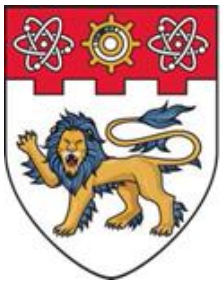


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

ENERGY-EFFICIENT AI HARDWARE DESIGN FOR EDGE INTELLIGENCE

CAO TIANCHENG

School of Electrical & Electronic Engineering

2023

ENERGY-EFFICIENT AI HARDWARE DESIGN FOR EDGE INTELLIGENCE

CAO TIANCHENG

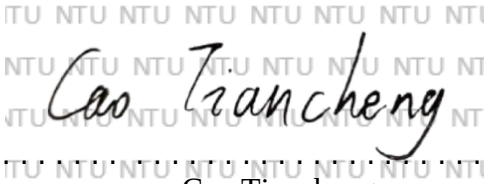
School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2023

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.



30/11/2023
Date

Cao Tiancheng

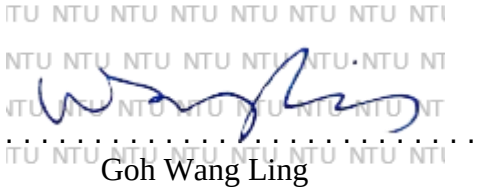
Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

7 December 2023

. . . . . .

Date



. . . . . .

Goh Wang Ling

Authorship Attribution Statement

This thesis contains material from two journal papers published in two different peer-reviewed journals, and two conference papers, all with the candidate as first author. The candidate is currently preparing one manuscript for submission to a peer-reviewed journal, and two other manuscripts for submission to peer-reviewed conferences, all as first author. Details of these papers are provided in the following paragraphs.

Chapter 3 is published as T. Cao, C. Liu, Y. Gao and W. L. Goh, "Parasitic-Aware Modeling and Neural Network Training Scheme for Energy-Efficient Processing-in-Memory with Resistive Crossbar Array," in *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 12, no. 2, pp. 436-444, June 2022, doi: 10.1109/JETCAS.2022.3172170.

The contributions of the co-authors are as follows:

- Prof. Goh Wang Ling supervised and revised the manuscripts.
- Dr. Gao Yuan coordinated the whole work and revised the manuscripts.
- Dr. Liu Chen helped with the experiments.
- The candidate conducted the experiments, processed the data, and prepared the manuscript.

Chapter 4 is published as T. Cao, C. Liu, W. Wang, T. Zhang, H.K. Lee, M.H. Li, W. Song, Z.X. Chen, V.Y.Zhuo, N. Wang, Y. Zhu, Y. Gao, W.L. Goh., "A Non-Idealities Aware Software–Hardware Co-Design Framework for Edge-AI Deep Neural Network Implemented on Memristive Crossbar," in *IEEE Journal on Emerging and*

Selected Topics in Circuits and Systems, vol. 12, no. 4, pp. 934-943, Dec. 2022, doi: 10.1109/JETCAS.2022.3214334.

The contributions of the co-authors are as follows:

- Prof. Goh Wang Ling supervised and revised the manuscripts.
- Dr. Gao Yuan coordinated the whole work and revised the manuscripts.
- Dr. Liu Chen, Dr. Weijie Wang, Dr. Tantan Zhang, Dr. Hock Koon Lee, Dr. Ming Hua Li, Dr. Wendong Song, Dr. Zhi Xian Chen, Dr. Victor Yi-Qian Zhuo, Dr. Nan Wang and Dr. Yao Zhu helped with the experiments.
- The candidate conducted the experiments, processed the data, and prepared the manuscript.

Chapter 5 is submitted to *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* as T. Cao, W. Yu, Y. Gao, C. Liu, T. Zhang, S. Yan, W.L. Goh., “Edge PoolFormer: A Memristor-based Complete Circuit PoolFormer Modeling and Learning Framework for Edge-AI Applications,”.

The contributions of the co-authors are as follows:

- Prof. Goh Wang Ling supervised and revised the manuscripts.
- Dr. Gao Yuan coordinated the whole work and revised the manuscripts.
- Mr. Yu Weihao, Dr. Liu Chen, Dr. Zhang Tantan, Dr. Yan Shuicheng helped with the experiments.
- The candidate conducted the experiments, processed the data, and prepared the manuscript.

Chapter 6.1 is published as T. Cao, Z. Zhang, W. L. Goh, C. Liu, Y. Zhu and Y. Gao, "A Ternary Weight Mapping and Charge-mode Readout Scheme for Energy Efficient FeRAM Crossbar Compute-in-Memory System," *2023 IEEE 5th International*

Conference on Artificial Intelligence Circuits and Systems (AICAS), Hangzhou, China, 2023, pp. 1-5, doi: 10.1109/AICAS57966.2023.10168639.

The contributions of the co-authors are as follows:

- Prof. Goh Wang Ling supervised and revised the manuscripts.
- Dr. Gao Yuan coordinated the whole work and revised the manuscripts.
- Mr. Zhang Zhongyi, Dr. Liu Chen and Dr. Zhu Yao helped with the experiments.
- The candidate conducted the experiments, processed the data, and prepared the manuscript.

Chapter 6.2 is published as T. Cao, Z. Zhang, W. L. Goh, C. Liu, Y. Zhu and Y. Gao, " ECG Classification Using Binary CNN on RRAM Crossbar with Nonidealities-Aware Training, Readout Compensation and CWT Preprocessing," *2023 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, Toronto, Canada, 2023.

The contributions of the co-authors are as follows:

- Prof. Goh Wang Ling supervised and revised the manuscripts.
- Dr. Gao Yuan coordinated the whole work and revised the manuscripts.
- Mr. Zhang Zhongyi, Dr. Liu Chen and Dr. Zhu Yao helped with the experiments.
- The candidate conducted the experiments, processed the data, and prepared the manuscript.

Chapter 6.3 is submitted to *IEEE Biomedical Circuits and Systems Conference (BioCAS)* as T. Cao, Z. Zhang, C. Liu, W. L. Goh, Y. Zheng and Y. Gao, "A Ternary Weight Mapping and Charge-mode Readout Scheme for Energy Efficient FeRAM Crossbar Compute-in-Memory System,".

The contributions of the co-authors are as follows:

- Prof. Goh Wang Ling and Prof. Zheng Yuanjin supervised and revised the manuscripts.
- Dr. Gao Yuan coordinated the whole work and revised the manuscripts.
- Dr. Liu Chen helped with the experiments.
- Mr. Zhang Zhengyuan and The candidate conducted the experiments, processed the data, and prepared the manuscript.

Chapter 7.2 is submitted to *IEEE Biomedical Circuits and Systems Conference (BioCAS)* as T. Cao, W. S. Ng, W. L. Goh and Y. Gao, " FPGA Implementation of PoolFormer Network using Python-Driven High-Level Synthesis Framework for Edge-AIoT Speech Recognition,".

The contributions of the co-authors are as follows:

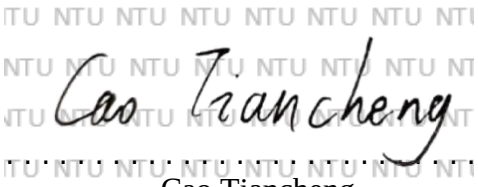
- Prof. Goh Wang Ling supervised and revised the manuscripts.
- Dr. Gao Yuan coordinated the whole work and revised the manuscripts.
- Mr. Ng Wei Soon helped with the experiments.
- The candidate conducted the experiments, processed the data, and prepared the manuscript.

Chapter 7.3 is in preparation for *IEEE Transactions on Biomedical Circuits and Systems* as T. Cao, W. S. Ng, W. L. Goh and Y. Gao, “Edge Cardiovascular Disease Recognition system with Quantized IncepResNet, CWT Approximation and FPGA Acceleration,”.

The contributions of the co-authors are as follows:

- Prof. Goh Wang Ling supervised and revised the manuscripts.
- Dr. Gao Yuan coordinated the whole work and revised the manuscripts.

- Mr. Ng Wei Soon helped with the experiments.
- The candidate conducted the experiments, processed the data, and prepared the manuscript.



30/11/2023
. . . . . .
Date

Cao Tiancheng

Acknowledgement

Throughout my Ph.D. journey and while preparing my thesis, I encountered numerous challenges that tested my perseverance. However, these trials also provided opportunities for personal and academic growth. I would like to express my heartfelt gratitude to the many individuals who extended their unwavering support and assistance during this transformative period.

First of all, I would like to express my sincere thanks to my supervisor Associate Professor Goh Wang Ling and co-supervisor Dr. Gao Yuan. Their insightful guidance has helped me a lot during the transformation from a learner to an active researcher.

I would like to thank my seniors, Dr. Chen Liu, Dr. Wei Zhou, Dr. Yisheng Chong, Dr. Tantan Zhang, and Dr. Yao Zhu. I appreciate their kind and patient assistance. Many thanks go to my group members: Mr. Zhongyi Zhang, Mr. Wei Soon Ng, Mr. Zhaoyang Cao, Ms. Ziming Liu and Mr. Chen Shen. They have provided me with a lot of help and support.

I would also like to extend my heartfelt gratitude to my dearest friends for their unwavering support. Their warm encouragement and companionship during my moments of distress are deeply appreciated.

My sincere thanks go to the technicians in the Centre for Integrated Circuits and System in the School: Ms. Lim-Tan Gek Eng and Ms. Seow-Guee Geok Lian. I am grateful for their kind help in resolving technical problems.

I am grateful to Nanyang Technological University for granting me the president scholarship and fostering a conducive research environment. My deep appreciation also goes to the examiners for dedicating their valuable time to review this thesis.

Finally, I want to express my deepest gratitude to my parents. Their selfless love and

unwavering support have consistently been a driving force, inspiring me to strive for continuous improvement.

Table of Contents

Summary

In the era of Artificial Intelligence (AI), the widespread adoption of AI applications has reached every facet of daily life, including the Internet of Things (IoT). However, the conventional Von Neumann architecture used in AI hardware implementations poses challenges, particularly for energy-efficient edge devices, where computational resources are constrained. Researchers are actively exploring solutions that leverage Computing-in-Memory (CIM) and Non-Volatile Memory (NVM) to optimize AI algorithms for these resource-limited environments. Deep neural networks (DNN) known for their remarkable performance in computer vision and natural language processing, demand substantial computational resources, making them unsuitable for edge devices. CIM and NVM technologies offer potential breakthroughs but face challenges like high static power and random telegraph noise. In conjunction with DNN, these innovations are poised to enable intelligent IoT applications, striking a balance between computational efficiency, memory constraints, and power consumption on edge devices. As another edge solution, FPGAs (Field-Programmable Gate Arrays) offer accelerated performance for deep neural networks by enabling parallel processing at the hardware level, resulting in faster inference and reduced latency compared to traditional CPU or GPU implementations. Additionally, FPGAs provide flexibility through reconfigurability, allowing optimization of hardware architectures tailored specifically to neural network models, enhancing efficiency and power consumption.

The challenges in adopting resistive nonvolatile memory (NVM) devices, specifically memristor-based crossbar arrays, for neural network applications are diverse. These hurdles encompass mitigating non-idealities introduced during

memristor fabrication, including parasitic resistances, device uniformity variations, and finite resistance states, which can significantly distort output currents and undermine vector-matrix multiplication (VMM) accuracy. Furthermore, the pressing demand for energy-efficient AI algorithms in areas like ECG signal analysis and edge AI devices intensifies the need to overcome these challenges. Capacitive crossbar arrays offer potential energy efficiency advantages but grapple with issues such as precise capacitance tuning and voltage sensitivity, while RRAM-based neural networks hold promise for energy efficiency but confront readout errors, uniformity, and non-ideal factor challenges. Addressing these multifaceted challenges is essential to harness the full potential of NVM devices and meet the growing demand for efficient neural network computations in diverse applications. On another side, developing an edge solution leveraging FPGA technology through high-level synthesis for real-time data processing and optimized performance in resource-constrained environments is also in a pressing need.

This Ph.D. thesis comprehensively investigates the implementation of AI accelerating on edge devices. In the introduction, background and motivations of the research work are described. Then state-of-art CIM system, AI algorithms, FPGA implementation and applications are reviewed in Chapter 2. In Chapter 3, the hybrid of complementary NVM modeling is proposed as a reinforced approach, and it can lead to significant improvements in the hardware simulation speed and performance. In Chapter 4, a general software-hardware co-design deep learning framework implemented by NVM is employed to mitigate the impact of hardware non-idealities and reduce the inference accuracy degradations. Afterwards, in Chapter 5, a more sophisticated deep learning framework, customized PoolFormer, implemented by measured RRAM-based

crossbar array is adopted to realize the implementation of Transformer-based DNN on hardware. The aim of Chapter 6 is to implement AI algorithms on hardware by CIM design for various applications. Next, Chapter 7 presents the advancement of high-level synthesis for FPGA in edge applications such as speech recognition and signal processing. Finally, Chapter 8 concludes the thesis and proposes the future work.

In conclusion, covering the modeling fundaments, algorithm developments, simulation, and application verifications, this Ph.D. thesis comprehensively investigates software-hardware co-design challenges, taking a large step towards practical edge AIoT applications.

List of Figures

List of Tables

Chapter 1. Introduction

This chapter serves to provide the background of the research work, highlighting the significance as well as the obstacles of non-volatile memory (NVM)-based computing-in-memory framework and edge Artificial Intelligence (AI) implementation of field-programmable gate array (FPGA). Following which, the motivation and objectives of the research work are given. The major contributions achieved in this research are stated, and the organization of the thesis is also provided at the end of the chapter.

1.1 Background

Artificial Intelligence (AI) is becoming ubiquitous in every aspect of daily life. Industries and research centers have put an enormous amount of effort into related areas and AI has shown its high efficiency in various tasks, such as natural language processing and computer vision, which is much better than humans [1]. As a key enabling technology for AI applications, deep neural network (DNN) has been implemented in different scale of hardware platforms from cloud cluster to edge devices [2-4]. To support the increasing computation demand, basic hardware implementation is required, such as the central processing unit (CPU) and the graphics processing unit (GPU). Nowadays, some local accelerators for neural network computing are designed, such as Tsinghua's Tianjic [5], MIT's Eyeriss [6] and Google's TPU [7]. The prevailing DNN hardware implementation is based on the conventional Von Neumann architecture, which is not optimized for the massive parallel operation nature of neural network (NN) [3].

A huge number of resources are consumed by the computation intensive matrix-based operation and the data movement between the memory and CPU due to the limited

bandwidth between processing unit and the memory. For mobile edge devices, energy efficiency is a major concern since it will directly affect the system operation lifetime. Therefore, new computation hardware architecture that is suitable to implement NN at edge devices with high energy efficiency is an emerging topic that attracted extensive research efforts [8-11]. Computing-in-memory (CIM) that processes data within the memory to overcome bandwidth limitation and speed mismatch between units has become one of the best choices among the existing hardware neural network accelerator. At present, several prototype chips based on traditional CMOS have been developed, such as flash memory [12], dynamic random-access memory (DRAM) [13] and static random-access memory (SRAM) [14].

With existing PIM prototype chips, the Von Neumann bottleneck (VNB) can be easily broken since simple neural networks can be processed in memory but still have some issues. For example, DRAM consumed large energy, flash memory needs a long time to write, and SRAM requires large chip area. Recent breakthroughs in memristor fabrication technologies open the door to various resistive nonvolatile memory (NVM) devices such as phase-change memory (PCM), magnetoresistive random access memory (MRAM) and resistive random-access memory (RRAM) [15]. With programmable resistance, the NN weights can be mapped to the multi-level memristor crossbar to realize vector-matrix multiplication (VMM), which is the most compute-intensive operation in the NN algorithm [16]. By applying input voltage vector to the word lines and detect the output current from the bit line, VMM is operated naturally in a parallel manner within the memory with time complexity of O(1). This computing-in-memory (CIM) architecture based on resistive crossbar array is an elegant solution to boost the speed of computation with multiple orders of energy efficiency improvement

[17-18].

On the other hand, amidst the same landscape, Field-Programmable Gate Arrays (FPGAs) have emerged as a compelling choice for edge AIoT applications [19]. FPGAs offer an array of advantages, including high parallelism, low power consumption, and hardware programmable reconfiguration, making them a versatile and energy-efficient solution for speech recognition and other edge AIoT tasks. Various efforts have been made to optimize deep neural networks for deployment on FPGAs, addressing challenges related to their computational complexity, memory footprint, and the resulting power consumption [20]. A primary emphasis has been on reducing memory usage, which consumes significantly more energy than arithmetic operations [21].

1.2 Motivation and Objectives

The recent breakthroughs in device fabrication have led to the emergence of analog memristor crossbar arrays with high resolutions, presenting a promising avenue for implementing neural networks (NNs) [22-24]. However, these advanced crossbars still suffer from various non-idealities, stemming from fabrication imperfections, peripheral circuit limitations, and system-level challenges [25-28]. These non-idealities can significantly compromise the accuracy and efficiency of NNs implemented on memristor crossbars [29-34]. Therefore, there is a critical need to develop a robust design and simulation tool that comprehensively accounts for these non-idealities, ultimately enabling the optimization of memristor-based NN systems for a wide range of applications. On the FPGA side, Transformer-based neural networks are gaining prominence over LSTM [35]. Transformers excel in parallel computation, capturing long-range dependencies more effectively, and providing better interpretability.

Nevertheless, the traditional Verilog development process faces constraints when it comes to compressing extensive Transformers for software-hardware co-design [36]. As a result, there is a pressing need for a high-level synthesis framework for Edge-AIoT speech recognition that leverages the advantages of Transformer-based models and FPGA implementations.

The aim of this thesis is to enhance the efficiency and dependability of neural network systems utilizing memristors across various applications. Additionally, it seeks to assess the potential for high-level synthesis development on Field-Programmable Gate Arrays (FPGAs) for signal processing and the implementation of neural networks on edge devices. Specifically, it contains the following four parts:

1. Accurate Non-Ideality Modeling: Develop precise modeling techniques for capturing non-idealities in memristor crossbar arrays, including device-level variations, failures, and array-level issues like line resistance and sneak paths.

2. Integration into NN Training: Incorporate the modelled non-idealities seamlessly into the NN training process, facilitating compensation and optimization during network design, and training phases, ultimately improving overall system performance.

3. Efficient and Scalable Framework: Create an efficient and scalable modelling and simulation framework capable of handling large training datasets, balancing computational complexity with accuracy, and addressing critical array-level non-idealities for realistic VMM simulations.

4. High-level synthesis development: Explore and assess the capability of high-level synthesis development using FPGA for optimizing signal processing algorithms and deploying neural networks effectively on edge devices.

1.3 Major Contributions

As the summary of my research work in the past few years, the thesis concentrates on tackling the limitations of pre-compensated wavefront shaping techniques stated above, and the major contributions can be reflected in four aspects.

First of all, a resistive crossbar model is presented that incorporates non-idealities for rapid and precise vector-matrix multiplication (VMM) simulations. The model achieves significantly faster computation speeds with minimal current distortion, reducing computation error by more than 10 times compared to existing acceleration methods.

Secondly, a unified simulation and training framework for edge-AI deep neural networks (DNNs) on memristor crossbars is introduced. This framework encompasses non-idealities at both individual element and array levels, along with peripheral circuit non-linearities. It leverages a dynamic quantization method to map NN weights to non-uniform conductance ranges and employs a transimpedance amplifier (TIA) to eliminate power-consuming ADC and DAC operations.

Thirdly, the crucial challenges in edge devices are addressed to meet the growing demand for efficient AI algorithms in various applications, including speech recognition, ECG signal analysis and biomedical imaging enhancement.

Last but not the least, the study confirms the effectiveness of FPGA-based solutions for meeting the requirements of edge applications like speech recognition while highlighting the crucial contribution of high-level synthesis frameworks in bridging the gap between software and hardware.

1.4 Organization of Thesis

The organization of this thesis is as follows:

Chapter II serves as foundations of the thesis work. The basics of NVM and NN is provided, followed by general reviews of state-of-art various CIM architecture.

In Chapter III, a fast and efficient crossbar interconnect line resistance estimation model named γ-compact model and the associated parasitic-aware neural network training method are presented. The relationship between the crossbar output error with the line resistance, crossbar size, RRAM device resolutions and resistance range have been analyzed. The proposed method reduced the VMM computation error by 186 and 17 times compared to the uncompensated method and the state-of-art compensation method, respectively.

In Chapter IV, a modeling and simulation framework for edge-AI DNN implemented on memristive crossbar is presented. Non-idealities aware training is proposed to achieve fast and accurate early exploration of the system design and reduce the degradation of inference accuracy. The impact of each non-ideality from the devices, arrays and circuits is analyzed, evaluated and then the modeling tool is demonstrated.

In Chapter V, a new light hardware-friendly Transformer like neural network structure, Edge PoolFormer, is presented. A fully circuit implementation framework is introduced for edge device implemented on memristive crossbar is proposed. The structure and performance of the neural network are analyzed. Integrating the nonideal factors into a unified training process allows for comprehensive evaluation of framework performance. In addition, compared with conventional DNNs, this structure can realize satisfactory accuracy with fewer parameters, and the training framework can effectively alleviate the effects of these non-idealities to minimize degradation of the

inference accuracy.

In Chapter VI, the focus is on CIM applications in edge devices. Novel designs are developed for efficient AI algorithms in various applications, including speech recognition, ECG signal analysis and biomedical imaging enhancement.

Chapter VII delves into the progressive evolution of high-level synthesis tailored for FPGA in diverse edge applications, prominently focusing on domains like speech recognition and signal processing. Through detailed analysis and case studies, it elucidates the pivotal role of high-level synthesis methodologies in empowering FPGA-based implementations, thereby facilitating the seamless integration of cutting-edge technologies into edge devices for robust and efficient functionality.

Finally, in Chapter VIII, the conclusions are drawn, and the suggestion for future work are provided and discussed.

Chapter 2. NVM-based Computing-in-Memory and High-Level Synthesis: A Review

2.1 Non-Volatile Memory

RSM device was firstly envisioned as memristor in 1971 [37] and introduced in 2008. Memristor connects the charge and the flux such that the resistance of the memristor depends on the previous total flux flowing through it (Fig. 2.1). In another word, RSM devices will respond to voltage pulses and switch their resistance according to the magnitude, direction, and duration of the pulses. The voltage that is just sufficient to switch the resistance is called the threshold voltage (VTH) in this report.

RSM devices have two kinds, digital and analog, due to different mechanisms. The electrical characteristics and behaviour [38-39] of the devices may not be much covered in this report since these factors are not important in simulation.

Digital RSM devices have a conductance switching from high resistance to low resistance abruptly when there is a set pulse and remain the resistance constant even

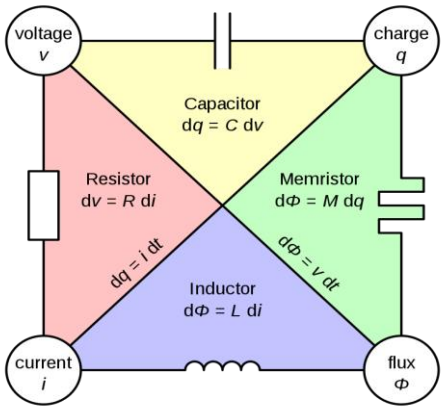


Fig 2.1. Memristor, the fourth fundamental non-linear circuit element, linking magnetic flux and charge.

there are more set pulses. When there is a reset pulse, the resistance will switch back to high and remain the same until the next set pulse (Fig. 2.2). Due to its 1-bit storage, digital RSM devices are also seen as an alternative of traditional memory such as DRAM and flash memory and some designs have proven that it can perform well in embedded system applications [40-41]. Nevertheless, the development of new neural network architecture, such as Binarized Neural Network (BNN) [42], made digital RSM suitable for computing in-memory (CIM) and the prototype chips have shown its high-speed computation and low energy consumption [43]. However, because of its digital characteristic, some devices are required to present one synapse and several external circuits to run the neural network. Thus, multiple-level devices are envisioned to overcome these limitations. However, set and rest asymmetry and bit-width make it complex to implement multi-level devices in hardware.

Compared to digital devices, analog RSM devices have a relative smooth set and reset processes when voltage pulses are introduced. It needs several pulses to achieve saturation low resistance during set process and several pulses to return to high

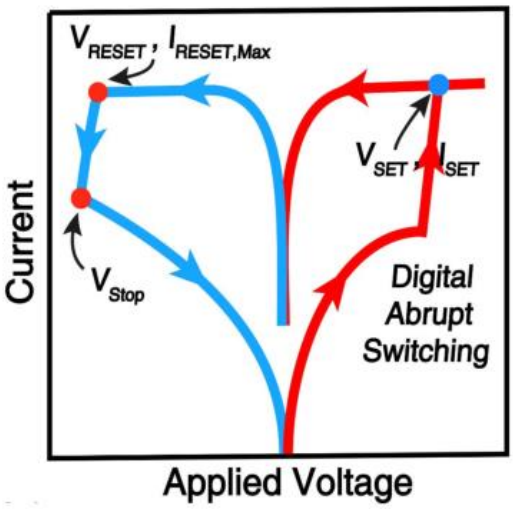


Fig 2.2. Digital RSM device I-V characteristic plot.

resistance (Fig. 2.3). Due to asymmetry of set and reset process, analog devices always appear in pair [44-46]. In theory, analog pair should have the ability to present any conductance among its switching boundary, but many internal and external variations make it impossible to reach desired resistance in reality [47]. The most significant works now for analog devices is to reduce the impacts of variations during implementation.

2.2 Deep Neural Network (DNN)

One of the most basic types of deep neural networks, the Full Connect Neural Network (FCNN) is a must-have for most entry-level deep learning fields, which fully reflects the characteristics of deep learning methods compared with traditional machine learning algorithms, that is, large Data-driven, formula derivation, self-iterative update, black box training, etc. Neural network is a kind of network model established by abstracting the neuron network of human brain from the perspective of information processing. Neural network is a breakthrough in science. It breaks through the limitations of traditional computer linear processing, adopts distributed storage and parallel computing processing methods, and is composed of a large number of parallel

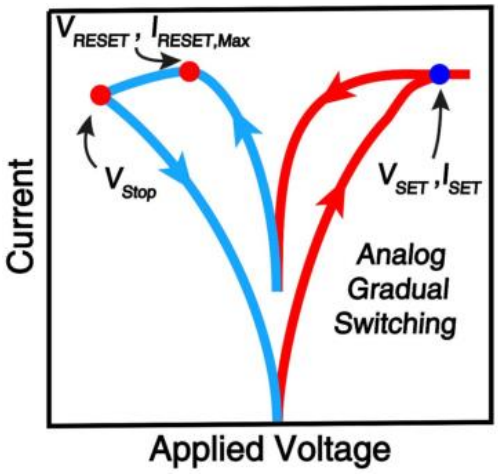


Fig 2.3. Analog RSM device I-V characteristic plot.

processing neurons. Although the structure and function of each neuron are extremely simple, the computing power brought by many neuron combinations, especially the application of distributed computing and storage methods, makes the neural network system have strong collective computing ability and adaptive learning.

The structure of a conventional deep neural network is shown in Fig. 2.4. The first layer that accepts input is the input layer, the last layer that provides output is the output layer, and all the layers in between are hidden layers. The more hidden layers and the more neurons in the hidden layer, the more complex the neural network can fit, but it is also harder to train. The way that neurons in every two adjacent layers are all connected to each other is called fully connected. Note that parameters such as weights and biases in the neural network are learned by the machine itself and are called model parameters. As for the number of layers in the neural network, and the number of nodes in each hidden layer, etc., they are pre-specified parameters that do not change during learning and are called hyper parameters.

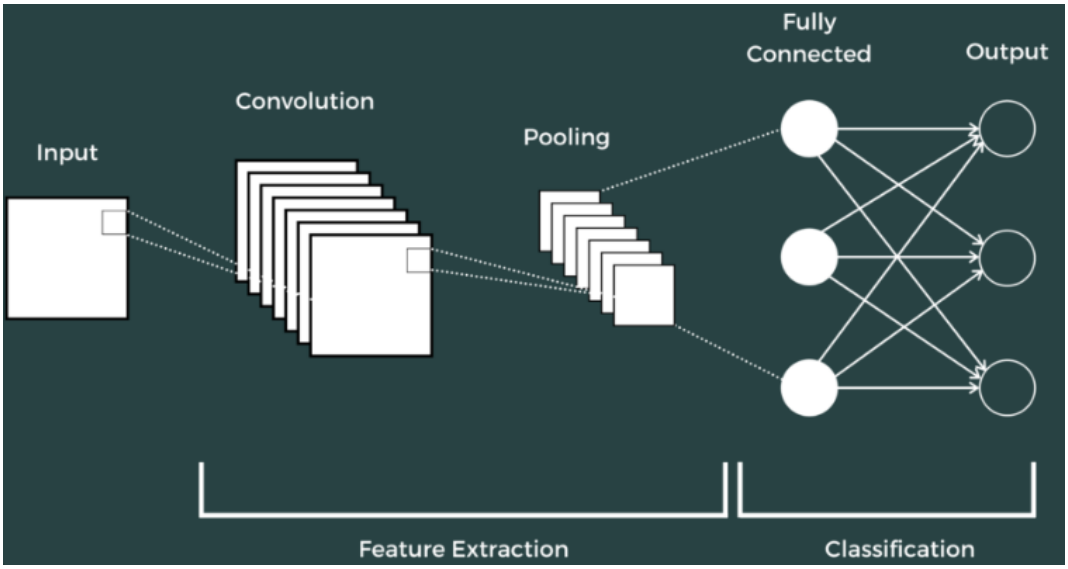


Fig 2.4. The conventional deep neural network with input layer, hidden layer and output layer.

2.3 Computing-in-Memory with NVM

During the 20th Century, human made a great success in exploring the working mechanism of our brain, which inspired many scientists to research on whether it is possible to simulate the mechanism of human brain based on electronic hardware to complete decision-making tasks. Compared to Von Neumann architecture widely used in computers, structure of the human brain is much more efficient with only 20 watts of energy consumption [48]. Data in the brain is stored in synaptic weights and neurons will collect and integral spikes from synapses then decide whether spikes are enough to

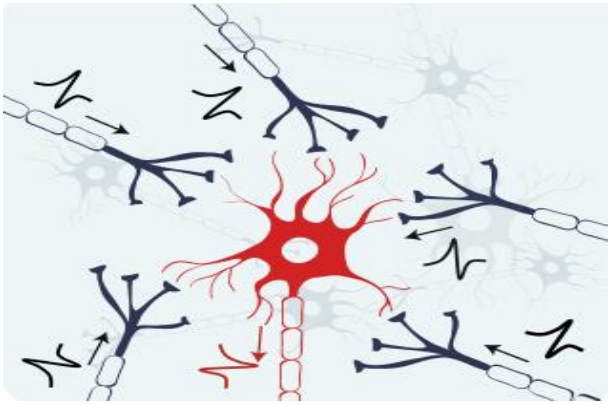


Fig 2.5. PM model simulates the process that the neuron in the center, which receives the pulse from synapses nearby, releases the pulse.

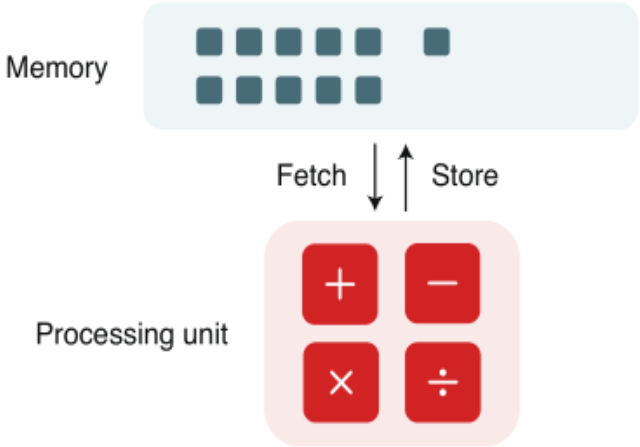


Fig 2.6. Memory store the synaptic weights and processing unit calculates whether to release a spike and store the spike value in memory.

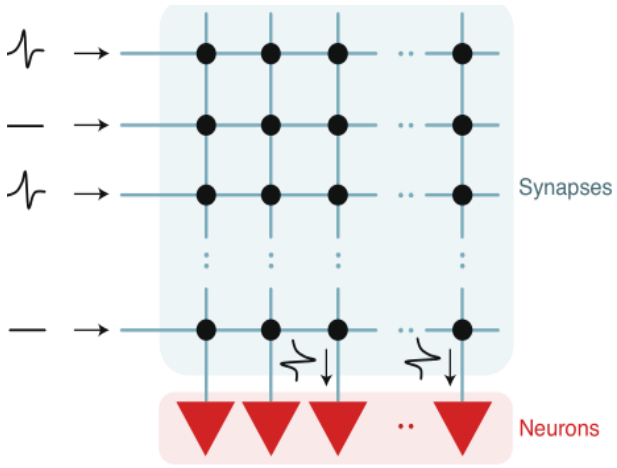


Fig 2.7. RSM devices store the synaptic weights and accumulative calculation results are received in bit lines.

Criteria	SRAM	DRAM	Flash	RRAM	PCM	FeFET
Non-volatility	No	No	Yes	Yes	Yes	Yes
Analog Storage	No	No	Yes	Yes	Yes	Yes
Area Efficiency	--	-	+	++	++	+
Program Speed	++	+	-	+	+	+
Power Efficiency	-	-	++	++	++	++
Technical maturity	High	High	High	Test chips	Test chips	Devices

Fig 2.8. RSM devices have shown its higher performance efficiency compared to traditional memory, like SRAM, DRAM and Flash.

release a spike to the following synapses, which is also known as PM model (Fig. 2.5) built by psychologist McCulloch and mathematician Pitts in 1943 [49]. Synapses work as memory to save the data and the neurons work as the processing unit to process the data from synapses (Fig. 2.6). Based on this PM model, the first-generation neural network, Perceptron, was published by F. Roseblatt in 1957 [50] and assembled with discrete resistors in 1961 [51]. Similarly, the computing-in-memory (CIM) architecture based on resistive crossbar array is an elegant solution to boost the speed of computation with multiple orders of energy efficiency improvement. It is important to note that the term non-volatile memory (NVM) is a larger concept and includes other types such as Flash memory, which operate differently from RRAM. Therefore, this thesis will

specifically use the term RRAM instead of NVM to accurately represent the technology discussed. However, the size and efficiency of Perceptron is limited by low-density hardware at that time. With the development of CMOS technology in 1980s, Bell Lab fabricated a 12*12 synaptic crossbar successfully, which can perform one of the most important parts of neural network, vector matrix multiplication (VMM) [52].

Neuro-inspired computing became possible when HP Labs introduced the first resistive switching device, i.e. the memristor in 2008 [53], and the first attempt on non-volatile memory based fabricated chip was reported in 2010 [54]. Therefore, many researches have placed their efforts on devices, algorithms and architectures for in-memory learning system based on resistive switching memory (RSM) (Fig. 2.7). Test chips based on different RSM devices, such as Ferroelectric Field Effect Transistor (FeFET) [55], phase change memory (PCM) [56] and resistive random access memory (RRAM) [57] have showed the exciting results compared to traditional memory [58-59].

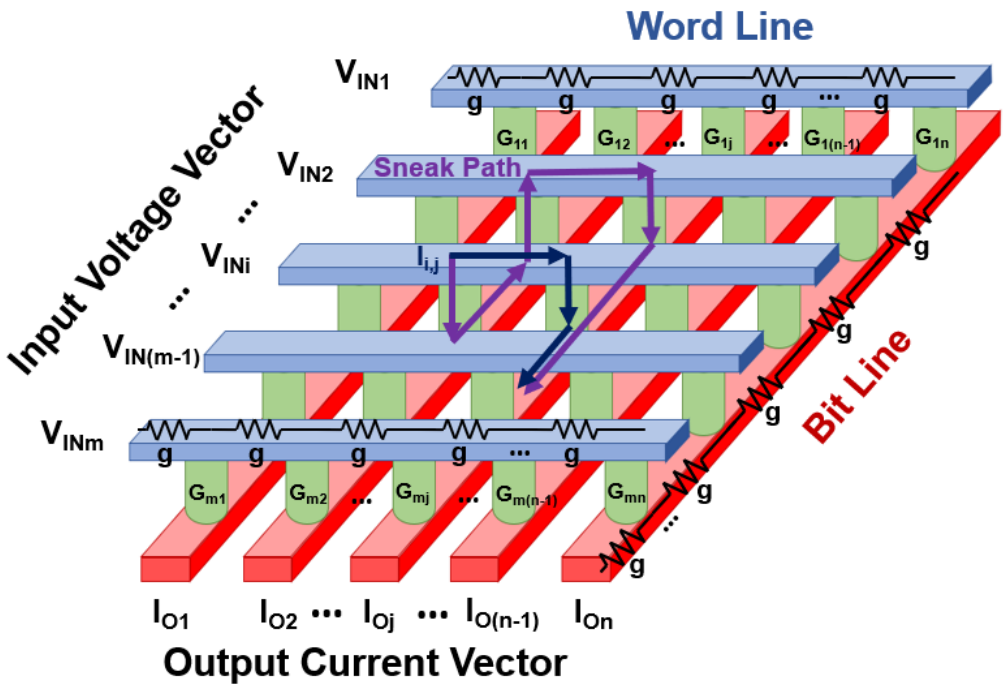


Fig. 2.9. Block diagram of a m×n crossbar array with horizontal word lines and vertical bit lines, including interconnect line resistances.

It is believed that the RSM based analog crossbar architecture could revolutionize the edge computing in artificial intelligence and internet of things (AIoT) and 5G era (Fig. 2.8).

Fig. 2.9 illustrates a generic memristor crossbar array that consists of a set of horizontal word lines (WL), vertical bit lines (BL) and memristor devices at each cross point. Line resistance refers to the ohm resistance of the interconnect traces along the top and bottom electrode of the memristor devices. The unit line resistance g between two adjacent memristor devices is in the range of a few ohms in the deep submicron CMOS process [60].

When an input voltage vector V_{IN} is applied to WL, neglecting the effect of line resistance first, the output current of j^{th} column I_{Oj} can be expressed as

$$I_{Oj} = \sum_{i=1}^{m} V_{INi} \cdot G_{ij} \tag{2.1}$$

where $1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n$. V_{INi} is the input voltage of i^{th} row, G_{ij} is the conductance of memristor device at i^{th} row, j^{th} column.

2.4 High-Level Synthesis Development

High-level synthesis (HLS) in neural networks involves the process of automatically translating high-level descriptions of neural network algorithms into optimized hardware designs. HLS tools enable designers to express neural network models using higher-level programming languages like C, C++, or Python instead of specifying the hardware details directly in hardware description languages (HDLs) like Verilog or VHDL. This approach aids in accelerating the deployment of neural network models onto hardware platforms like FPGAs as shown in Fig. 2.10, facilitating faster and more efficient inference for various applications, including edge devices and specialized

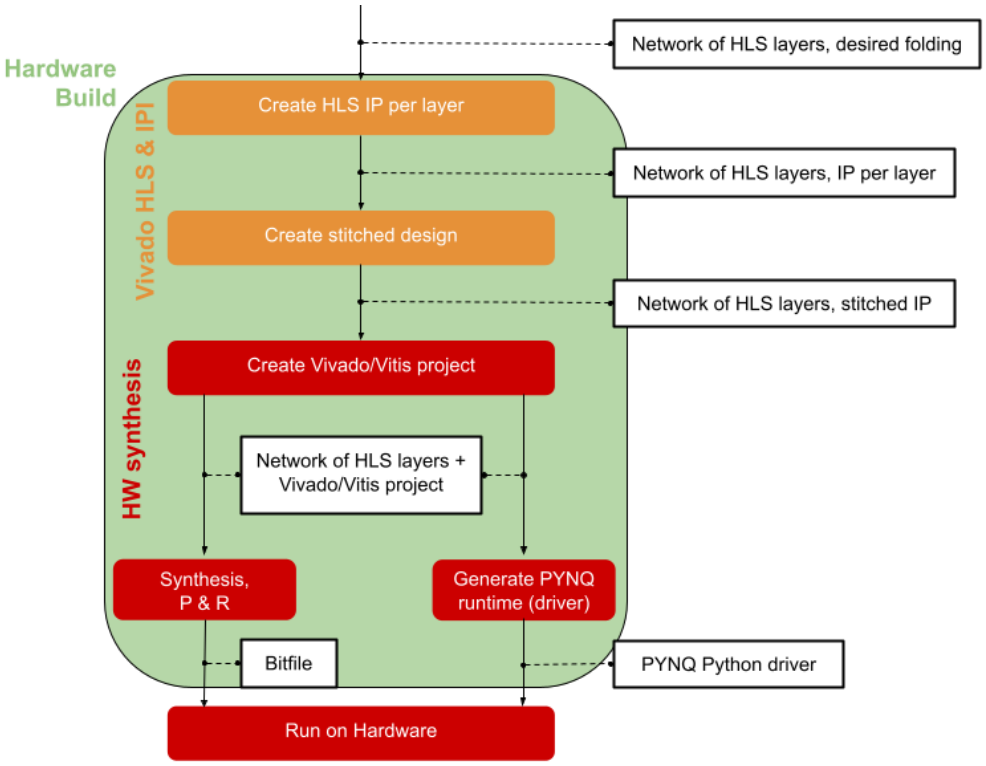


Fig. 2.10. The hardware build process for implementing neural networks on FPGA using high-level synthesis.

hardware accelerators.

2.5 Summary

This chapter provides a brief literature review of the history of processing-in-memory and current challenges and opportunities. Moreover, the structure of resistive crossbar array is introduced. The implementation of the vector matrix multiplication on crossbar array is briefly explained.

Chapter 3. Parasitic-Aware Modeling for Computing-in-Memory with Resistive Crossbar Array

As reviewed in Chapter 2, artificial Intelligence (AI) is becoming ubiquitous in every aspect of daily life. As a key enabling technology for AI applications, deep neural network (DNN) has been implemented in different scale of hardware platforms from cloud cluster to edge devices [2-3,5]. The prevailing DNN hardware implementation is based on the conventional Von Neumann architecture, which is not optimized for the massive parallel operation nature of neural network (NN) [3]. A great amount of resources are consumed by the computation intensive matrix-based operation and the data movement between the memory and CPU. For mobile edge devices, energy efficiency is a major concern since it will directly affect the system operation lifetime. Therefore, new computation hardware architecture that is suitable to implement NN at edge devices with high energy efficiency is an emerging topic that attracted extensive research efforts [8-11].

3.1 Recent Works

Recent breakthroughs in memristor fabrication technologies open the door to various resistive nonvolatile memory (NVM) devices such as phase-change memory (PCM), magnetoresistive random access memory (MRAM) and resistive random-access memory (RRAM) [15]. With programmable resistance, the NN weights can be mapped to the multi-level memristor crossbar to realize vector-matrix multiplication (VMM), which is the most compute-intensive operation in the NN algorithm [16]. By applying input voltage vector to the word lines and detect the output current from the bit line, VMM is operated naturally in a parallel manner within the memory with time

complexity of O(1). This processing-in-memory (PIM) architecture based on resistive crossbar array is an elegant solution to boost the speed of computation with multiple orders of energy efficiency improvement [17-18]. One notable study by W. Wang et al. (2019) demonstrated the integration of CMOS RRAM chips with programmable performance, providing a foundation for neuromorphic computing applications. This work forms a basis for understanding the device characteristics and their implications on system-level performance [73].

However, the key challenge for practical application of resistive crossbar array is that the actual memristor fabrication process introduces various types of non-idealities, including parasitic resistances [17], device uniformity variations [47, 61], and the finite states of resistance [29]. These non-idealities can severely distort the crossbar output current so that the VMM computation accuracy is deteriorated and in turn leads to degradation of NN inference accuracy [32-33]. Among these non-ideal factors, line resistance has been identified as the leading contributor, especially when the resistive crossbar is implemented in an advanced technology node or the array size scales up.

Several techniques have been proposed to model the crossbar non-idealities and include them into the system simulation and optimization process. Such as technology optimization [62] and modified training algorithms [63-64] have been explored for both on-chip and ex-situ learning to mitigate specific non-ideal effects such as IR-drop, synaptic device variations. It has been proven that the DNN inference accuracy can be improved significantly by including the non-idealities factors into the network training process. However, the NN training process involves multiple epochs and huge amount of data are required. Therefore, a fast and accurate parasitic-aware VMM computation method is the key enabling technique for PIM system implemented on resistive

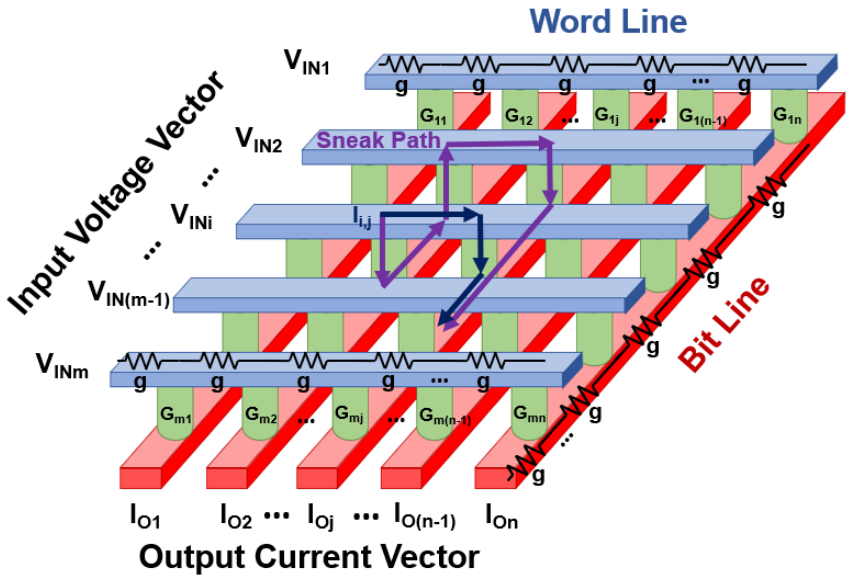


Fig. 3.1. Block diagram of a m×n crossbar array with horizontal word lines and vertical bit lines, including interconnect line resistances.

memristor crossbar.

Solutions to incorporate line resistance into VMM computation have been reported in many literatures [65-68]. The most straightforward solution is to calculate the crossbar current distribution by solving the Kirchhoff's laws equations. A m×n crossbar shown in Fig. 3.1 has 2mn network nodes and the current distribution can be calculated by performing nodal analysis to solve a 2mn × 2mn matrix equation [60]. Assuming Gaussian elimination method is adopted, the time complexity to solve the equation analytically is $O(m^3n^3)$ [69]. With iterative matrix solvers, the time complexity can be reduced to $O(m^2n^2)$ [70]. However, it is still time-consuming especially when the crossbar size scales up. To accelerate the simulation speed, a new approach was proposed to treat the voltage and current deviations as scaling factors applied to the input voltage and output current [67-68]. The scaling factors for word lines and bit lines are estimated separately as two one-dimensional arrays. Hence, the VMM computation is simplified, and the reported time complexity is further reduced to $O(m^2+n^2+mn)$ [68],

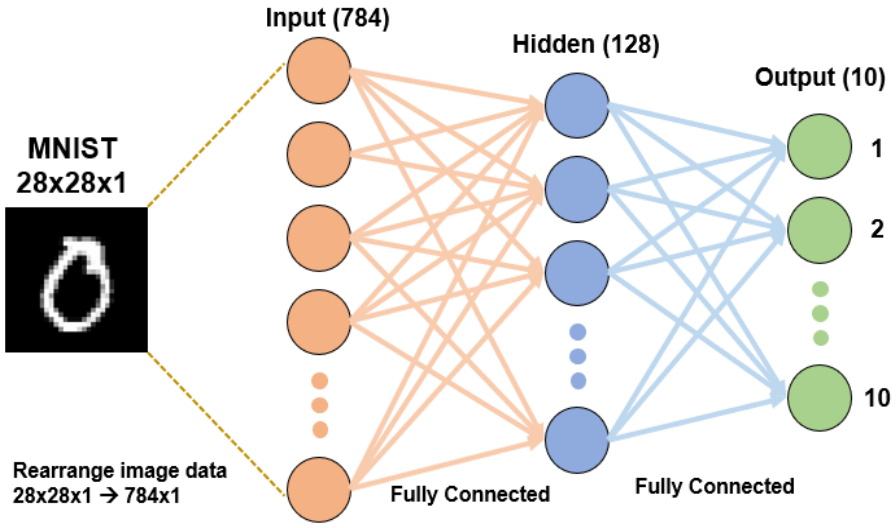


Fig. 3.2. Two-layer MLP for MNIST classification task.

which is a remarkable improvement compared to solving Kirchhoff's laws equations. However, the limitation of this method is the relatively large current distortion because this method uses the averaged memristor conductance in the scaling factors calculation. The current distortion will lead to accuracy degradation in both NN training and inference. In summary, a fast and accurate method to model the resistive crossbar interconnect line resistance into VMM simulation is still not available and it is highly desired for the design of crossbar-based PIM system.

Fig. 3.1 illustrates a generic memristor crossbar array that consists of a set of horizontal word lines (WL), vertical bit lines (BL) and memristor devices at each cross point. Line resistance refers to the ohm resistance of the interconnect traces along the top and bottom electrode of the memristor devices. The unit line resistance g between two adjacent memristor devices is in the range of a few ohms in the deep submicron CMOS process [60].

When an input voltage vector V_{IN} is applied to WL, neglecting the effect of line resistance first, the output current of j^{th} column I_{Oj} can be expressed as

$$I_{Oj} = \sum_{i=1}^{m} V_{INi} \cdot G_{ij} \quad (3.1)$$

where $1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n$. V_{INi} is the input voltage of i^{th} row, G_{ij} is the conductance of memristor device at i^{th} row, j^{th} column.

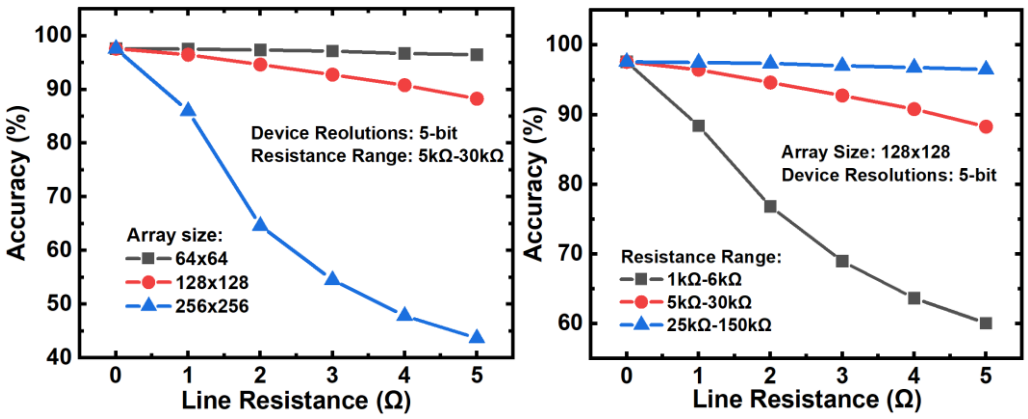
Line resistance has significant impact to the current flow due to the IR-drop and the sneak path. The actual voltage across the memristor device will be smaller than the theoretical value because of the accumulated IR-drop on the interconnect traces and the sneak paths. To explain the impact of line resistance to the overall PIM network performance, MNIST hand-written digits classification task is used as an example. Firstly, a two-layer MLP model with parameters shown in Fig. 3.2 is trained off-line with 32-bit weight resolution. Maximum 98.3% training accuracy is achieved on the test datasets. Next, the weights are mapped to the memristor crossbar. Considering the recently reported RRAM crossbar array hardware implementations [15, 17, 22, 60, 70-71], the trained network weights are mapped to 5-bit (32 levels) resistance in the range of 5kΩ to 30kΩ with the mapping method in [16]. With the mapped crossbar, the inference accuracy dropped slightly to 97.6%. However, as shown in Fig. 3.3 (a), after interconnect line resistance is included into VMM computation by solving the Kirchhoff's laws equations, the inference accuracy decreases drastically with the increase of line resistance and this trend is more obvious for larger array size. Besides, with the same array size, the accuracy drops more seriously for smaller RRAM resistance as shown in Fig. 3.3(b). Therefore, line resistance cannot be neglected and should be included in the crossbar-based NN design procedure.

3.2 Proposed γ-Compact Model

This section presented the details of the proposed parasitic-aware VMM model

which consists of two major improvements compared to [68]. Firstly, the conductance of individual RRAM device in the array is used in the scaling factors calculation, instead of the averaged conductance [68]. Secondly, a new set of calibration parameter γ, γ_α and γ_β are introduced to calibrate the distortions due to sneak path and scaling factors, which was not addressed in the previous designs. By combing these two techniques together, better VMM accuracy can be achieved with acceptable increase in computation overhead.

The crossbar array architecture suffers from two significant issues that impact the accuracy of neural network (NN) inference: sneak current and line resistance. Sneak current paths, which are unintended current paths through the array, cause interference and errors in the read operations of the memristor devices. This interference leads to distorted output signals, reducing the accuracy of the NN inference. On the other hand, line resistance refers to the resistance of the metal lines connecting the memristor devices in the crossbar array. The presence of line resistance causes a voltage drop across the array, leading to incorrect voltage levels being applied to the memristors. This voltage drop impacts the overall accuracy of the NN inference by introducing errors in the computed results. To evaluate the relative impact of sneak current and line resistance on accuracy, a series of experiments and simulations were conducted. The results indicate that while both factors contribute to accuracy degradation, the impact of line resistance is more pronounced than sneak current. The line resistance causes a more significant and consistent drop in accuracy across different array sizes and resistance values compared to the sneak current. This is due to the cumulative effect of voltage drops along the interconnects, which directly affects the voltage applied to the memristors and hence the output current.



(a) (b)

Fig. 3.3. (a) MLP inference accuracy versus the variation of line resistance in different array sizes. (b) MLP inference accuracy versus the variation of line resistance in different RRAM resistance ranges.

3.2.1 Model Derivation

To address the non-ideal effects such as sneak paths and scaling factor distortions, new calibration coefficients γ, γ_α and γ_β are introduced. These coefficients are derived to compensate for the distortions and ensure accurate voltage distributions across the crossbar array. The parameter γ calibrates the overall distortion due to sneak paths, while γ_α and γ_β specifically adjust the scaling factors for the word line (WL) and bit line (BL) resistances, respectively. These coefficients are calculated based on the normalized IR-drop (NIR) values, which consider the voltage drops and resistances along the interconnect lines. By solving the Kirchhoff equations for a reference crossbar array with uniform conductance, the calibration coefficients are determined to minimize the discrepancies in voltage drops and improve the accuracy of the VMM computation. This enhancement enables the model to accurately reflect the effects of parasitic resistances and maintain high inference accuracy even for large and complex crossbar arrays.

The normalized IR-drop (NIR) is defined as:

$$NIR_{i,j} = \frac{\Delta V_{i,j}}{V_{INi}} \tag{3.2}$$

where the voltage across $G_{i,j}$ is $\Delta V_{i,j}$, WL input voltage is V_{INi} and BL output is

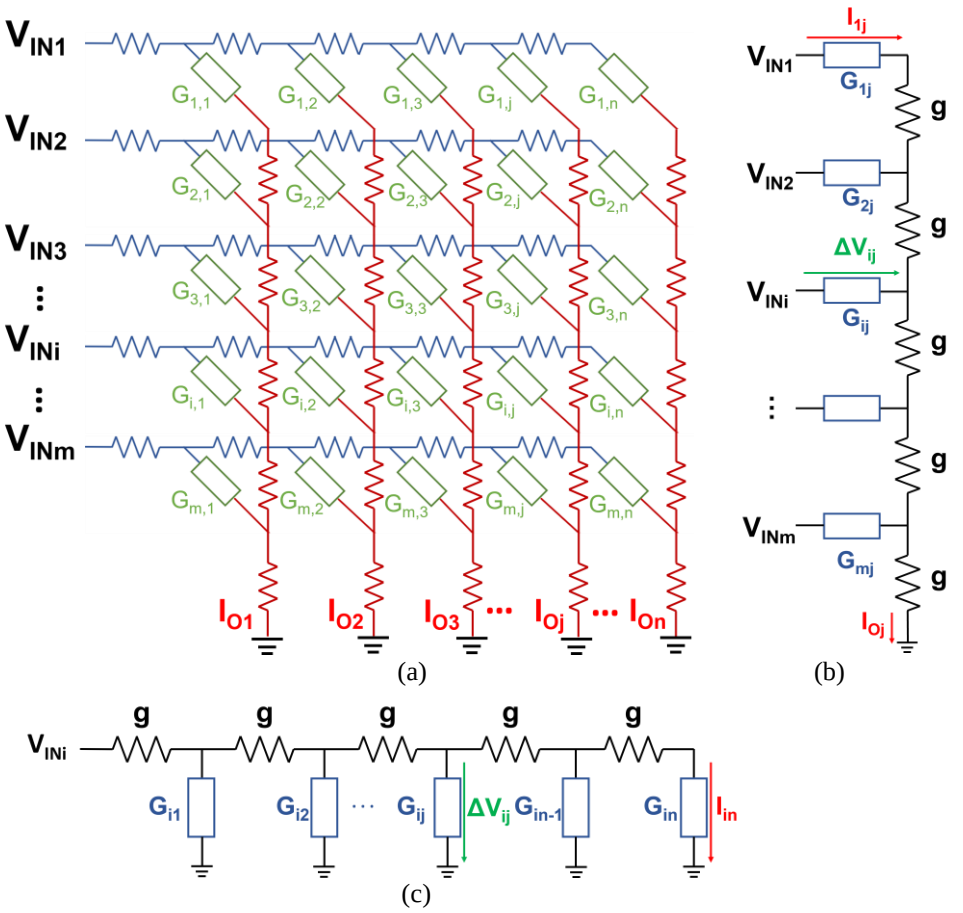


Fig. 3.4. (a) Equivalent circuit of m×n crossbar array with horizontal word lines and vertical bit lines. (b) The column model of the crossbar for j^{th} bit line with all word lines connected to 1V. (c) The row model of the crossbar for i^{th} word line with all bit lines tied to ground.

grounded. The ideal current flow through each RRAM is:

$$I_{i,j} = V_{INi} \cdot G_{i,j} \tag{3.3}$$

The BL and WL scaling factors $\alpha_{i,j}$ and $\beta_{i,j}$ are the special cases of $NIR_{i,j}$ when the line resistance along WL is neglected and the line resistance along BL is neglected, separately.

BL scaling factor $\alpha_{i,j}$ is derived as follows. For a $m \times n$ crossbar shown in Fig. 3.4(a), the j^{th} column model is shown in Fig. 3.4(b) when the line resistance along the word line is neglected and the input voltage V_{INi} = 1V, i =1, 2, 3, …, m.

According to Kirchhoff's voltage law (KVL), the voltage across $G_{i,j}$ is the same as

the sum of the voltage across $G_{1,j}$ and the voltage across 1st to i^{th} interconnect lines:

$$\Delta V_{i,j} = I_{1,j}/G_{1,j} + [(i-1)I_{1,j} + (i-2)I_{2,j} + \cdots + I_{i-1,j}]/g \tag{3.4}$$

Similarly, V_{INi} is derived as

$$V_{INi} = I_{1,j}/G_{1,j} + [mI_{1,j} + (m-1)I_{2,j} + \cdots + I_{m,j}]/g \tag{3.5}$$

Then, BL scaling factor $\alpha_{i,j}$ is written as

$$\alpha_{i,j} = \frac{\Delta V_{i,j}}{V_{INi}} = \frac{I_{1,j} \cdot g + \left[\sum_{k=1}^{i}(i-k)I_{k,j}\right] \cdot G_{1,j}}{I_{1,j} \cdot g + \left[\sum_{k=1}^{m}(m-k+1)I_{k,j}\right] \cdot G_{1,j}} \tag{3.6}$$

Similar processes are applied to derive the BL scaling factor $\beta_{i,j}$. The i^{th} row model of the crossbar is shown in Fig. 3.4(c). Assuming the line resistance along BL is neglected and the input voltage V_{INi} = 1V, for i = 1, 2, 3, …, m. WL scaling factor $\beta_{i,j}$ is written as

$$\beta_{i,j} = \frac{\Delta V_{i,j}}{V_{INi}} = \frac{I_{i,n} \cdot g + \left[\sum_{k=1}^{n-j+1}(k-1)I_{i,j+k-1}\right] \cdot G_{i,n}}{I_{i,n} \cdot g + \left[\sum_{k=1}^{n} kI_{i,k}\right] \cdot G_{i,n}} \tag{3.7}$$

To simplify the computation, the calculation of ideal current flows through each element of interconnect trace is extracted and computed in a separate process. The current accumulate matrix with size $(m+1) \times n$ and $m \times (n+1)$ are formed with time complexity of $O(mn)$.

$$I_{i,j}^{\alpha acc1} = \begin{cases} 0, & i = 1 \\ I_{i-1,j}^{\alpha acc1} + I_{i-1,j}, & 2 \le i \le m+1 \end{cases} \tag{3.8}$$

$$I_{i,j}^{\alpha acc2} = \begin{cases} 0, & i = 1 \\ I_{i-1,j}^{\alpha acc2} + I_{i,j}^{\alpha acc1}, & 2 \le i \le m+1 \end{cases} \tag{3.9}$$

$$I_{i,j}^{\beta acc1} = \begin{cases} 0, & j = n \\ I_{i,j+1}^{\beta acc1} + I_{i,j+1}, & 0 \le j \le n-1 \end{cases} \tag{3.10}$$

$$I_{i,j}^{\beta acc2} = \begin{cases} 0, & j = n \\ I_{i,j+1}^{\beta acc2} + I_{i,j}^{\beta acc1}, & 0 \le j \le n-1 \end{cases} \tag{3.11}$$

The sneak paths from the adjacent lines also affect the current distribution and the actual current flow through $G_{i,j}$ is not the ideal $I_{i,j}$ [16]. To compensate this effect and the errors due to the assumptions made to calculate $\alpha_{i,j}$ and $\beta_{i,j}$, a set of calibration coefficients γ, $\gamma_\alpha and\ \gamma_\beta$ are introduced. (3.6) and (3.7) can be modified as

$$\alpha_{i,j}{}' = \frac{I_{1,j} \cdot g \cdot \gamma + I_{i,j}^{\alpha acc2} \cdot G_{1,j}}{I_{1,j} \cdot g \cdot \gamma + I_{m+1,j}^{\alpha acc2} \cdot G_{1,j} \cdot \gamma_\alpha} \tag{3.12}$$

$$\beta_{i,j}{}' = \frac{I_{i,n} \cdot g \cdot \gamma + I_{i,j}^{\beta acc2} \cdot G_{i,n}}{I_{i,n} \cdot g \cdot \gamma + I_{i,0}^{\beta acc2} \cdot G_{i,n} \cdot \gamma_\beta} \tag{3.13}$$

γ is used to compensate the distortion on the first row and the last column which are affected more by the second order effects. γ_α and γ_β helps to compensate the distortions for the devices at the edge of input side.

A reference crossbar array of size $m \times n$ is created, all the device has the same conductance G_{mean} which is the average of maximum and minimum conductance of the RRAM device. Kirchhoff equations are solved to get the accurate voltage distribution. The following equations (3.14) and (3.15) are used to calculate the calibration coefficients, so that the scaling factors of the corner device $G_{1,n}$ and $G_{1m,1}$ are calibrated and in turn the entire array is calibrated.

$$\alpha'_{1,n} \cdot \beta'_{1,n} = NIR_{1,n} \tag{3.14}$$

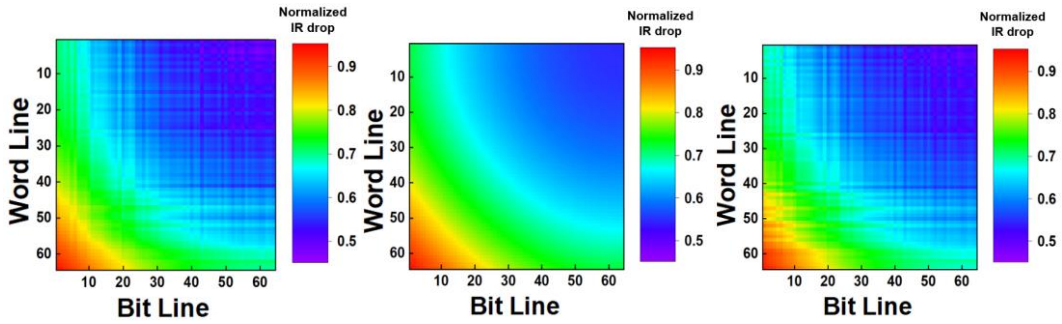
$$\alpha'_{m,1} \cdot \beta'_{m,1} = NIR_{m,1} \tag{3.15}$$

Solving the equations, the compensation coefficient γ can be calculated by

$$\gamma = \max\left\{\frac{\left(AXg\sqrt{NIR_{1,n}} + AX^2\sqrt{NIR_{1,n}}\right)}{g\left(g - g\sqrt{NIR_{1,n}} + AX - AX\sqrt{NIR_{1,n}} - BX\sqrt{NIR_{1,n}}\right)}, 0\right\} \tag{3.16}$$

where $A = \left(\frac{m-1}{m+1}\right)$, $B = \left(\frac{2}{m+1}\right)$ and $X = \frac{m(m+1)}{2} \cdot G_{mean}$.

Then γ_α and γ_β are derived as



(a) (b) (c)

Fig. 3.5. The IR-drop heatmap with randomized array weights on a 64 × 64 crossbar and 3Ω line resistance for (a) Kirchhoff's equations model, (b) dynamic compact model and (c) γ-compact model.

$$\gamma_\alpha = \frac{\left(\frac{m-1}{m+1}\right)\cdot\left(g + \frac{m(m+1)}{2}\cdot G_{mean}\right) + \frac{2}{m+1}\cdot g\cdot\gamma}{g + \left(\frac{m-1}{m+1}\right)\cdot\frac{m(m+1)}{2}\cdot G_{mean}} \tag{3.17}$$

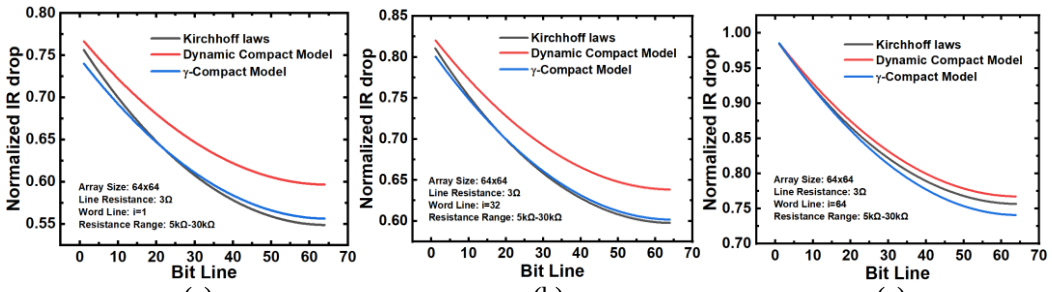
$$\gamma_\beta = \frac{\left(\frac{n-1}{n+1}\right)\cdot\left(g + \frac{n(n+1)}{2}\cdot G_{mean}\right) + \frac{2}{n+1}\cdot g\cdot\gamma}{g + \left(\frac{n-1}{n+1}\right)\cdot\frac{n(n+1)}{2}\cdot G_{mean}} \tag{3.18}$$

Final step is to compute the output which is also in $O(mn)$. The output current vector can be written as

$$I_O = V_{IN} \times (\alpha' \cdot G \cdot \beta') \tag{3.19}$$

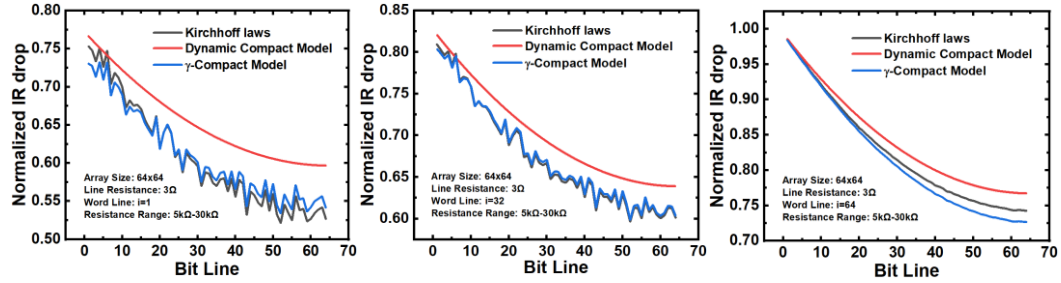
3.2.2 Model Validation

The accuracy of the proposed model is evaluated with extensive simulations and benchmarking with other state-of-art VMM computation methods. Three different computation methods based on Kirchhoff's laws equations [65], dynamic compact model [68] and the proposed γ-compact model are implemented in Python. The result based on Kirchhoff's laws equations is considered as the baseline reference since it is the most rigorous solution. The crossbar size is fixed at 64×64 with 5-bit RRAM devices with resistance range in 5 kΩ to 30 kΩ. Fig. 3.5 shows the crossbar voltage drop heatmap simulated by the three methods. Compared to the reference shown in Fig. 3.5(a),



(a) (b) (c)

Fig. 3.6. The normalized voltage drop of uniform weights simulations on 64 × 64 crossbar with 3Ω line resistance. The voltage drop of (a)1st word line, (b)32nd word line and (c) 64th word line with different models.

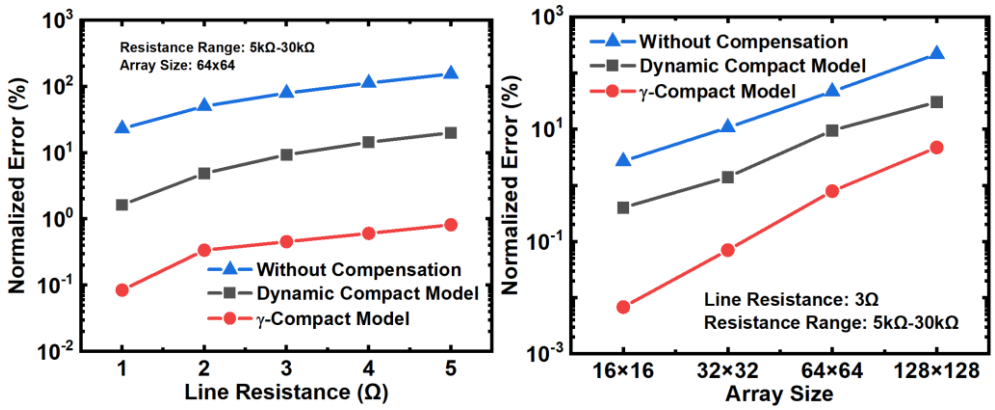


(a) (b) (c)

Fig. 3.7. The normalized voltage drop of random weights simulations on 64 × 64 crossbar with 3Ω line resistance. The voltage drop of (a)1st word line, (b)32nd word line and (c) 64th word line with different models.

dynamic compact model in Fig. 3.5(b) shows the correct voltage drop trend but cannot capture the details of individual RRAM due to the usage of averaged conductance for scaling factor calculation. By contrast, the proposed γ-compact model in Fig. 3.5(c) can estimate the voltage drop of each individual RRAM more accurately.

More detailed VMM computation accuracy analysis results are shown in Fig. 3.6 and Fig. 3.7. The normalized BL voltage on the 1st, 32nd and 64th row are calculated with three methods and compared. Fig. 3.6 shows results with uniform conductance mapped on the RRAM. Because of the uniform conductance, the effect of individual conductance scaling factor is removed. It can be observed that the proposed method matches well with the Kirchhoff's laws result. The discrepancy in dynamic compact model result is largely due to the second order effect which cannot be compensated in that method. This result also shows the efficacy of calibration factor γ in this work. The



(a) (b)

Fig. 3.8. Comparison of normalized output current error of dynamic compact model and γ-compact model with (a) different line resistance (b) different crossbar array size.

RMS error of γ-compact model at 1st row and 32nd row is reduced by 7.7 and 20 times compared to dynamic compact model.

Fig. 3.7 shows the same plots as Fig. 3.6, but with randomized RRAM conductance. The flutcations shown in Fig. 3.7(a) and Fig. 3.7(b) reflect the effect of BL line resistance which cause voltage drop related to the randomized RRAM conductance. This effect is minimal in the last row since there is only one unit line resistance contribute to the voltage drop, therefore the curve is much smoother. The proposed method can capture the individual conductance while the dynamic compact model cannot reflect the details. The RMS error of γ-compact model is reduced 32.3 times, 83.3 times and 3.2 times compared to dynamic compact model, respectively.

3.2.3 Output Error Verification

To evaluate the error of the output current, the normalized error of current is calculated by

$$I_{error} = \frac{\sum_j \left(\frac{I_j^{est} - I_j^{act}}{I_j^{act}}\right)}{n} \tag{3.20}$$

where I_j^{est} is the j^{th} column output current estimated by model and I_j^{act} is the actual

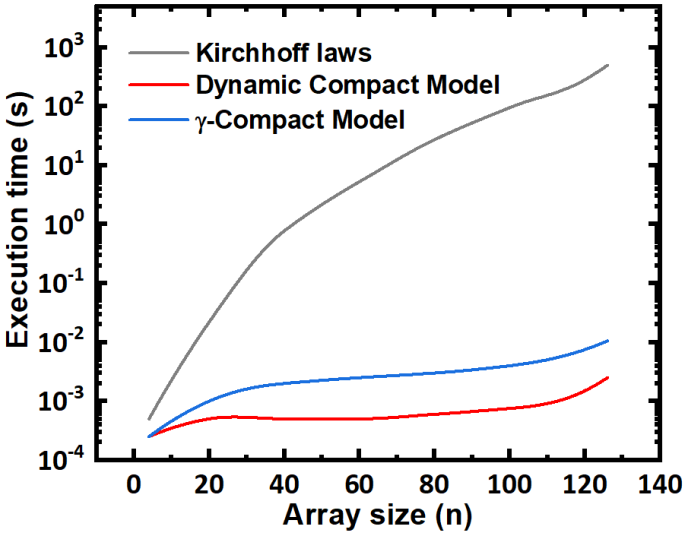


Fig 3.9. The comparison of VMM computation time with different models on a 16 cores 3.2GHz AMD Ryzen 5800H CPU with 32GB RAM.

j^{th} column output current calculated by Kirchhoff's laws equations.

The γ-compact model can achieve 0.5% current error with 3Ω interconnect line resistance on 64 × 64 array. The mean normalized output current error in Fig. 3.8 (a) shows better performance of the γ-compact model. The average error of γ-compact model is only 5.1% of the error of dynamic compact model under different line resistances. The simulation is repeated by changing the size of the crossbar. The γ-compact model also performs well as shown in Fig. 3.8 (b). The normalized average output error of γ-compact model is 5.7% of the error of dynamic compact model under different array size.

Fig. 3.9 shows the comparision of VMM computation time with different models on a 16-core 3.2GHz AMD Ryzen 5800H CPU with 32GB RAM. It can be observed that the computation time of Kirchhoff's laws method increases dramatically with larger array size, it is almost 10^5 times of γ-compact model when the array size is 128×128. On the other hand, the computation time of γ-compact model is approximately 5 times of the dynamic compact model. The longer computation time is mainly due to the larger

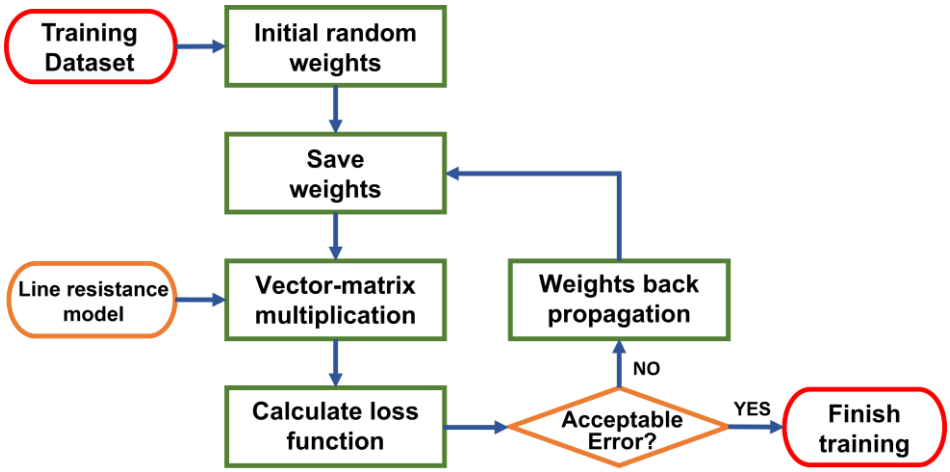
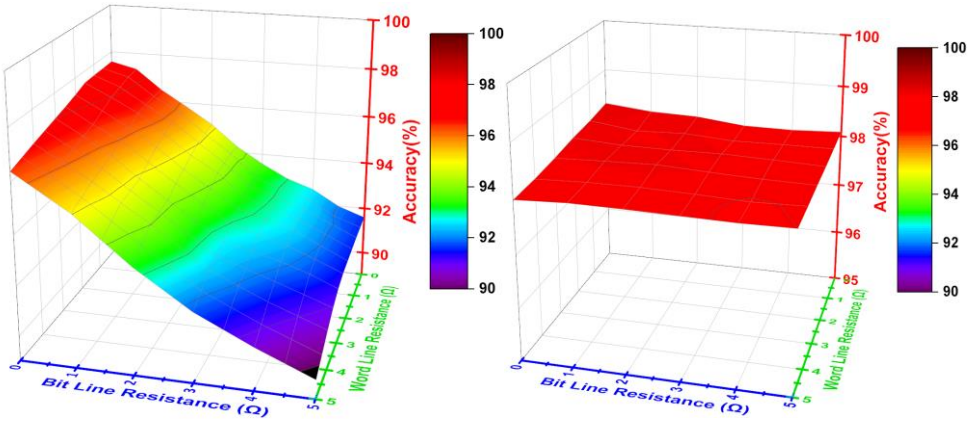


Fig 3.10. The flow chart of parasitic-aware training.



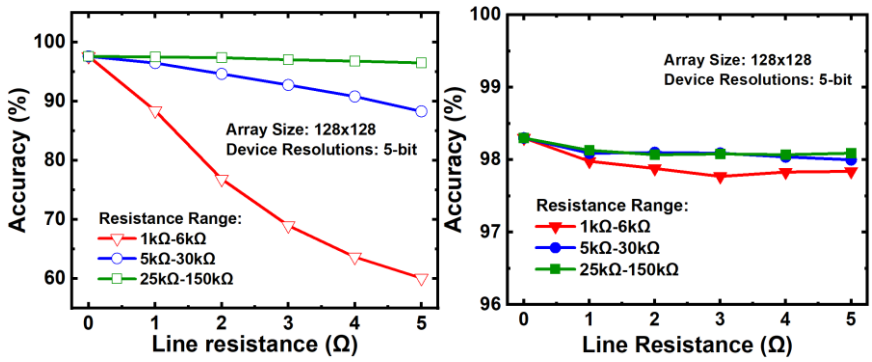
(a) (b)

Fig 3.11. MLP inference accuracy with different line resistance for (a) conventional training scheme that line resistance is not considered during training. (b) parasitic-aware training scheme.

amount of memory access.

3.3 Simulation Results

After the establishment of parasitic-aware VMM computation function, a parasitic-aware NN training scheme is proposed to compensate the impact of the line resistance to the NN inference performance. Fig. 3.10 shows the flow chart of the training scheme, the parasitic-aware VMM computation is included in the forward process of the back propagation training. The training is performed on a two-layer MLP with the same structure shown in Fig. 3.2 for MNIST classification task. Unless specified otherwise,



(a) (b)

Fig. 3.12. MLP inference accuracy with different RRAM resistance range for (a) conventional training method that line resistance is not considered during training, (b) parasitic-aware training scheme.

the default RRAM crossbar has an array size of 128 × 128, 5-bit resolution with resistance ranges from 5 kΩ to 30 kΩ. The interconnect line resistance is 3 Ω.

Fig. 3.11 shows the change of MLP inference accuracy with different line resistances for the conventional training scheme and the proposed method. As shown in Fig. 3.11 (a), the accuracy of conventional training method decreases when the line resistance increases. It is also observed that line resistance on

BL has larger impact to inference accuracy. This is because that the IR-drop of each row is dominated by BL line resistance. The inference accuracy of γ-compact model and the associated NN training scheme is less affected by line resistance since the impact has been taken into consideration during the training phase. As shown in Fig. 3.11(b), the accuracy can be maintained above 98% even with 5 Ω line resistance for both WL and BL.

Fig. 3.12 shows the effect of line resistance to MLP inference accuracy with different RRAM device resistance ranges. The RRAM device resistance range is selected from 3 groups, namely 1 kΩ to 6 kΩ, 5 kΩ to 30 kΩ and 25 kΩ to 150 kΩ. The line resistance is swept from 0 Ω to 5 Ω. Fig. 3.12 (a) shows that if line resistance is not considered during the training phase, the

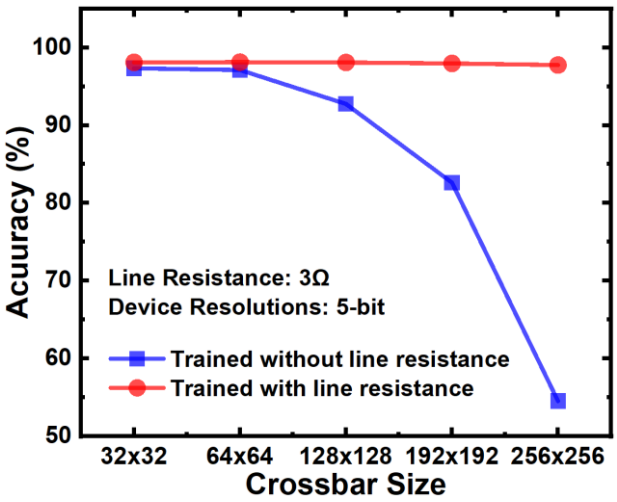
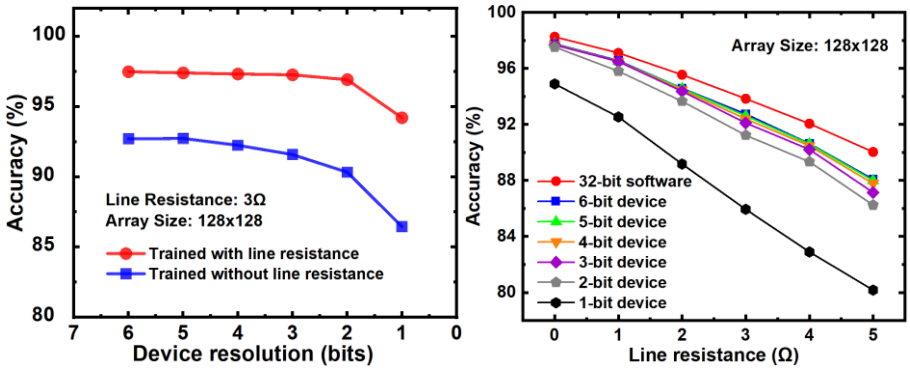


Fig. 3.13. MLP inference accuracy in different RRAM array size.



(a) (b)

Fig 3.14. (a) Comparison of MLP inference accuracy of models trained with and without line resistance. (b) MLP inference accuracy with model trained without line resistance under different device resolutions and line resistances.

inference accuracy will decease with larger line resistance and this change is more drastic for smaller resistance range. On the other hand, the result of parasitic-aware training scheme shown in Fig. 3.12(b) demonstrated good immunity to RRAM resistance range change. The proposed method achieved inference accuracy of 97.8% in the worst case of 1 kΩ to 6 kΩ resistance range, which is only 0.5% lower than the ideal model.

Fig. 3.13 compares the MLP inference accuracy under different training methods when the same NN model is mapped to different sizes of crossbar array. The crossbar array size is scaled up from 32 × 32 to 256 × 256. With larger crossbar size, the impact

of line resistance becomes more pronounced, therefore we can see more drastic accuracy decrease if the line resistance is not included in the training phase. With parasitic-aware training, good inference accuracy better than 97.9% is maintained even with large array size 256×256.

Fig. 3.14(a) compares the MLP inference accuracy of models trained without and with line resistance under varied line resistance and device resolutions. It shows that the proposed method can effectively improve the inference accuracy, the degradation is reduced to less than 1.2% even when the device resolution is reduced to 2-bit, indicating less sensitiveness to device resolutions compared to the conventional method. To further investigate the impact of line resistance and device resolutions, Fig. 3.14(b) shows the MLP inference results trained without line resistance under different device resolutions and line resistances. When line resistance is zero, the results show the impact of weight resolution on accuracy. It can be observed that the accuracy degradation is less than 0.8% for 6-bit to 2-bit weight resolutions. However, when the weight resolution further reduces to 1-bit (binary), huge accuracy degradation is observed. On the other hand, after line resistance's effect is included, inference accuracy drops for all weight resolution curves. These curves have similar slope with respect to the change of line resistance, but the lower resolution curves show the trend of higher accuracy drop and this gap increases with larger line resistance. This is probably due to the secondary effects such as sneak path. Lower weight resolution introduces larger weight quantization error, which will affect the crossbar IR drop via sneak path and in turn affect the crossbar output current accuracy. However, the analysis of the impact of this effect on inference accuracy is not straightforward, it is an interesting research topic to explore in the near future.

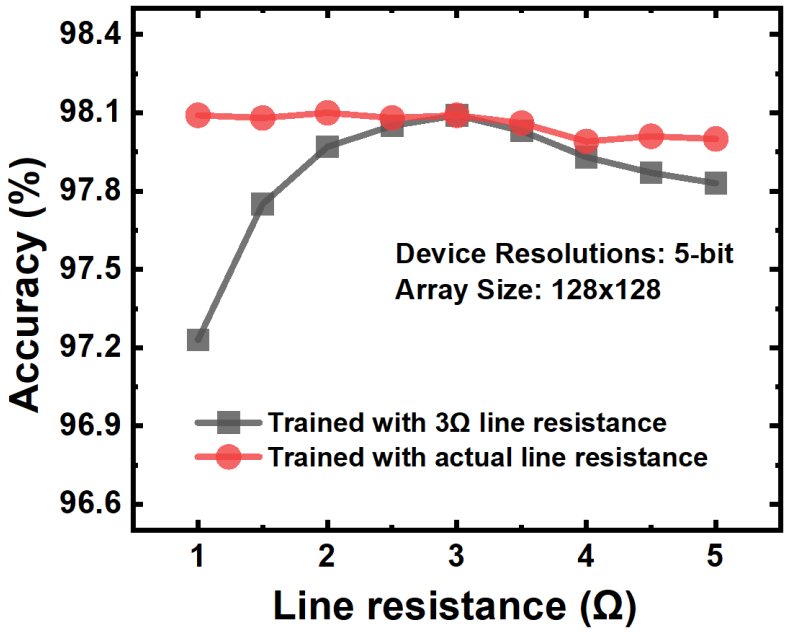


Fig 3.15. Comparison of inference accuracy of MLP with model trained with a fixed line resistance and model trained with actual line resistance.

Line resistance can be estimated with the crossbar fabrication process parameters such as metal layer resistivity, metal thickness and trace width [77]. However, due to the inevitable fabrication tolerance, there is discrepancy between the calculated and actual line resistances. As the parasitic-aware training is an off-line training process, it is not able to in-situ compensate this discrepancy. To investigate the impact to inference accuracy, the proposed model is firstly trained with 3Ω interconnect line resistance and then evaluated the inference accuracy with actual line resistance varied from 1Ω to 5Ω, corresponding to ±67% variations. As shown in Fig. 3.15, the inference accuracy decreases when the actual interconnect line resistance deviates from 3Ω used in parasitic-aware training. Larger inference accuracy drop is observed when the actual line resistance is smaller than 3Ω. This is because the change of normalized IR voltage due to line resistance is not a linear function and it decreases with larger line resistance. Since the network is trained with 3Ω line resistance, when the actual line resistance is less than this value, larger variation of normalized voltage is expected and in turn leads

to larger decrease of inference accuracy. Nevertheless, the accuracy can still maintain around above 97%, less than 1% degradation compared to the matched line resistance case.

The developed γ-compact model is versatile and can be applied to different RRAM configurations, including the 1T1R (one transistor-one resistor) array structure. The 1T1R architecture involves integrating a transistor with each RRAM cell, providing better control over the memory element and enhancing the overall performance of the array. This adaptation involves recalibrating the model parameters to account for the specific characteristics of the 1T1R configuration, such as transistor geometry and wire parasitic.

3.4 Summary

In this chapter, a fast and efficient crossbar interconnect line resistance estimation model named γ-compact model and the associated parasitic-aware neural network training method are presented. The relationship between the crossbar output error with the line resistance, crossbar size, RRAM device resolutions and resistance range have been analyzed. The proposed method reduced the VMM computation error by 186 and 17 times compared to the uncompensated method and the state-of-art compensation method, respectively. Maximum 98.1% inference accuracy is achieved with only 0.17% accuracy degradation compared to the ideal MLP model for MNIST classification task.

Chapter 4. A Non-Idealities Aware Software-Hardware Co-design Framework for Edge-AI

As reviewed in Chapter 3, in recent years, memristor crossbar has emerged as a promising candidate for its compelling advantages of high energy efficiency and high data throughput. Memristive devices has unique feature of non-volatile programmable conductance which can store the NN weights. Furthermore, by applying input voltage vector and sense the output current, the vector-matrix multiplication (VMM) operation can be implemented in a parallel manner naturally, leading to significant computation acceleration [10-11,15]. Memristor crossbar implemented in phase-change memory (PCM), magnetoresistive random-access memory (MRAM) and resistive random-access memory (RRAM) have been reported [16,78-80].

4.1 Recent works

Recently, with the recent advancements in device fabrication and novel device write-verify techniques, multi-level analog memristor crossbar arrays with resolution up to 8-bit have been demonstrated [25-27]. However, there are still challenges that limit the performance of NN implemented on memristor crossbar. Due to fabrication imperfectness, memristor crossbar exhibits various non-idealities, ranging from the device level conductance variation, device failure, programming failure [28-30] to the array level line resistance parasitics and sneak path [31]. In addition, the peripheral circuits such as analog to digital converter (ADC) and digital to analog converter (DAC) also introduce additional resolution limitation and nonlinearities [52,66,86]. These non-idealities can distort the crossbar output current so that the VMM computation accuracy is deteriorated. The errors will accumulate and propagate along the DNN layers and in

turn leads to degradation of training and inference accuracy. Hence, it is important to include these factors into the model of system, so that compensation and optimization can be performed during the network design and training phases to improve the final system performance.

The impact of memristor device non-idealities on NN performance have been studied extensively [29-34]. Many techniques have been explored to model these non-idealities into NN training and inference processes. For example, modified training algorithm have been explored to mitigate specific non-ideal effects such as device variation and IR-drop. There are also efforts to build comprehensive simulation and training framework to evaluate the DNN performance [31-34,82-87]. RxNN [31] and CxDNN [82] have included certain non-idealities into the inference and compensate the degradation of neural network performance. However, these inference modeling cannot be applied to training process directly due to the lack of the backward propagation and weight update phases [83]. Modeling tools like MNSIM [84] focused more on area and power of the system but ignored device variations and simplified non-idealities calculation. NeuroSim [85] can give a more comprehensive simulation including area, power estimation and execution time. However, for a large training dataset, such detailed simulation leads to long execution time and is unnecessary at the stage of system design. CrossSim [86] considers more non-idealities from memristive devices but neglected non-idealities from array and circuit levels and simulates with simple perceptron on a small dataset, MNIST [71]. Those tools didn't include function to model nonuniform distributed conductance in crossbar array. In addition, array level non-idealities including line resistance and sneak path are usually neglected or simplified in the VMM simulation, because of the high computation complexity overhead. Moreover,

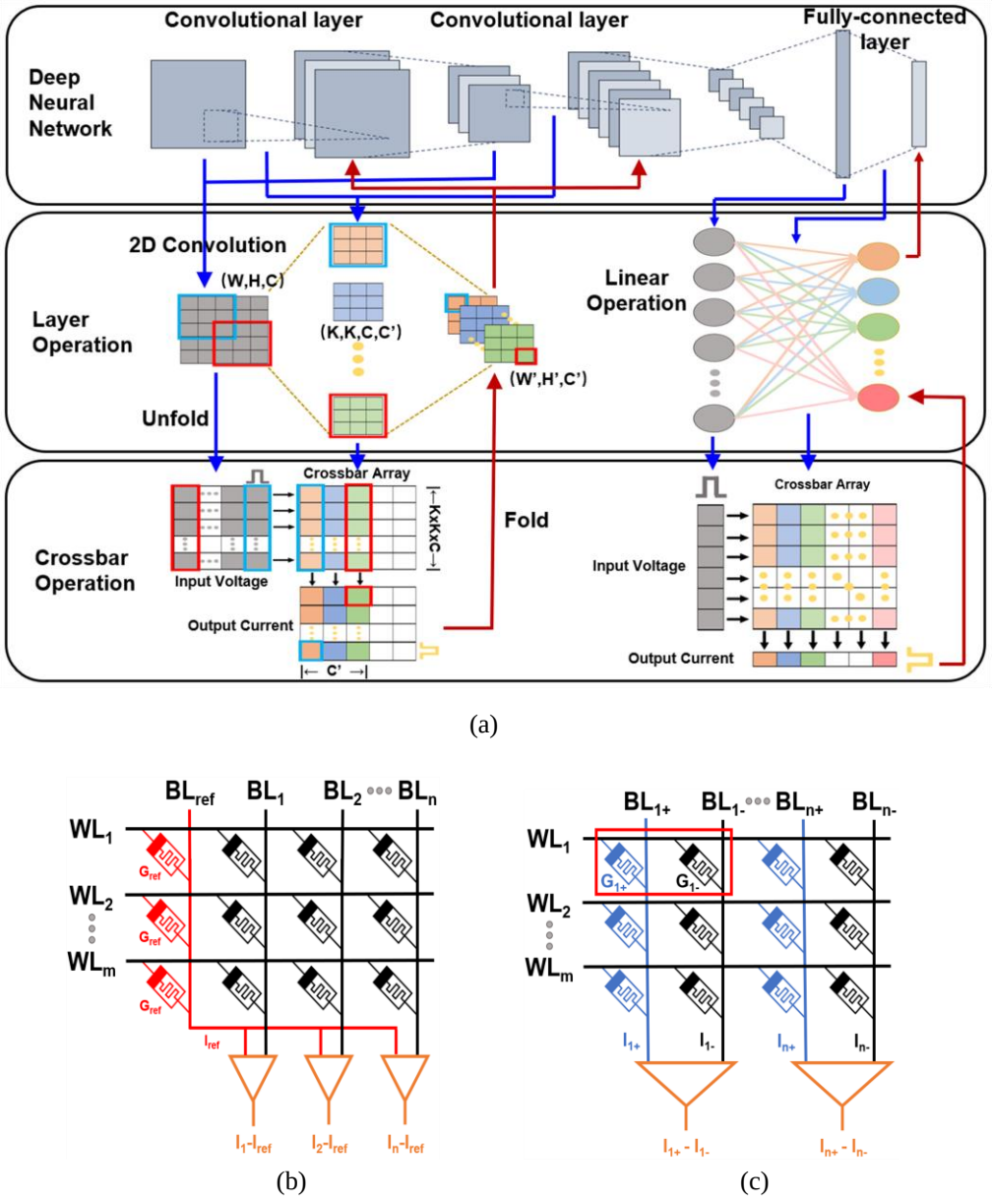


Fig. 4.1. (a) Block diagram of the mapping of a DNN convolution layer and a fully connected layer to memristor crossbars. (b) Structure of one-cell-one-weight scheme with reference current. (c) Structure of two-cell-one-weight scheme with differential pairs.

active function as one of the critical components in DNN, its hardware implementation is rarely presented in detail. In summary, a fast and accurate modeling and simulation tool for memristor crossbar-based NN performance evaluation is highly demanded.

4.2 Co-Design Framework

4.2.1 DNN Mapping to Crossbar

The two key components in a DNN are convolution layer and fully connected layer. They consume the majority of the computation resources. Fig. 4.1 illustrate the mapping relations of DNN algorithm, layer operation and the crossbar array implementation.

As shown in Fig. 4.1(a), convolutional layer input feature maps and kernels can be unfolded and transferred. C' kernels (K,K,C,C') slide over the feature maps (W,H,C) to generate outputs. When one kernel (K,K,C) slides, the corresponding feature map block (K,K,C) is selected, and the element-wise multiplication is applied. The sum of the multiplication result is the corresponding element of the output feature map. Reshape the kernel to one column (K×K×C,1) and the selected block to one row (1,K×K×C), the output element becomes the vector multiplication result. For multiple kernels and the same selected block, append the reshaped kernels column by column to form a kernel matrix (K×K×C,C') and results (1,C') of the selected block and all kernels are generated by vector-matrix multiplication. The unfolding process appends all reshaped selected blocks to form the input matrix which will be transferred to voltage pulse series and pumped into crossbar array from word lines. After folding the results, the feature map (W',H',C') for the next layer is generated.

The mapping of negative weights is an inevitable issue for crossbar-based computation. Several prior arts have presented one-cell-one-weight scheme with reference current [86] as shown in Fig. 4.1(b). Nevertheless, it is still impractical for large-scale array due to the large device variation. Therefore, the two-cell-one-weight scheme shown in Fig. 4.1(c) is adopted [15]. As shown in Fig. 4.1(c), every two adjacent columns are employed for one weight columns, left column cells represent positive

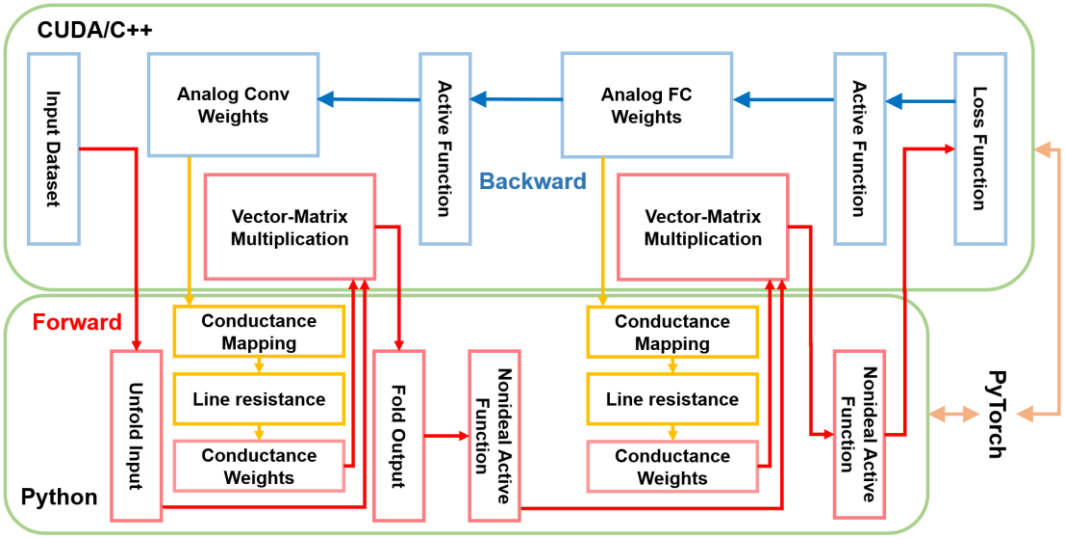


Fig. 4.2. Overall system architecture and data flow of the proposed framework.

weight and right column cells represent negative weight. If the weight is positive, the weight value is mapped to left cell and the right cell is adjusted to HRS, vice versa. The bit line current of two columns are subtracted to get the actual output current *I*.

$$I = I^{+} - I^{-} = V \times (G^{+} - G^{-}) \tag{4.1}$$

4.2.2 Proposed System Architecture

Fig. 4.2 shows the overall block diagram of the proposed DNN simulation framework. It consists of two parts implemented in PyTorch CUDA and Python, respectively. To speed up the computation, the most time consuming VMM operations are implemented in NVIDA CUDA and processed in GPU. In the forward convolutional layer, the analog weights are firstly mapped to conductance together with line resistance as the conductance weight. Both unfolded input feature map and conductance weight are fed into the VMM block. After folding the extracted output into feature map, nonideal ReLU activation function is applied, and the output is prepared for the next layer. For the fully-connect layer, the analog weights are also firstly mapped to conductance in a similarly way. The VMM is applied to input vector and conductance

weights. The final output after nonideal activation function will be transfer to GPU and compute the loss function and check the accuracy. The error will be used for backward propagation and update the analog weights.

The proposed non-idealities aware software-hardware co-design framework, while demonstrated on a crossbar structure, is equally applicable to 1T1R array structures. The techniques for modeling device, array, and circuit non-idealities, as well as the non-idealities aware training methods, can be adapted to the 1T1R array configuration. This ensures that the framework provides robust performance for different hardware implementations, thereby enhancing its versatility and applicability.

4.3 Non-idealities Aware Model

4.3.1 Device Non-Idealities

Non-ideal memristor device behaviors like conductance variation, nonlinearity, device to device (D2D), chip to chip (C2C) variations and programming failure are inevitable due to the fabrication imperfectness [47]. Therefore, nonuniform distributed levels and conductance variation should be considered in the DNN simulation. The proposed model tool assigns device to each level based on the conductance distribution in crossbar array. Then, normal distribution is added to simulate the non-ideal behaviors for each level [89]. To simulate the impact of programming failure, a random programming failure conductance mask matrix is generated in each write cycle based on the measured programming failure rate. The programmed conductance value is the Boolean AND operation of this programming failure mask and the array conductance matrix. The failure devices are mapped to HRS state. Similar to the case of programming failure, the stuck fault is simulated with a randomly generated mask matrix with stuck rate and the conductance of masked device is mapped to HRS. It should be highlighted

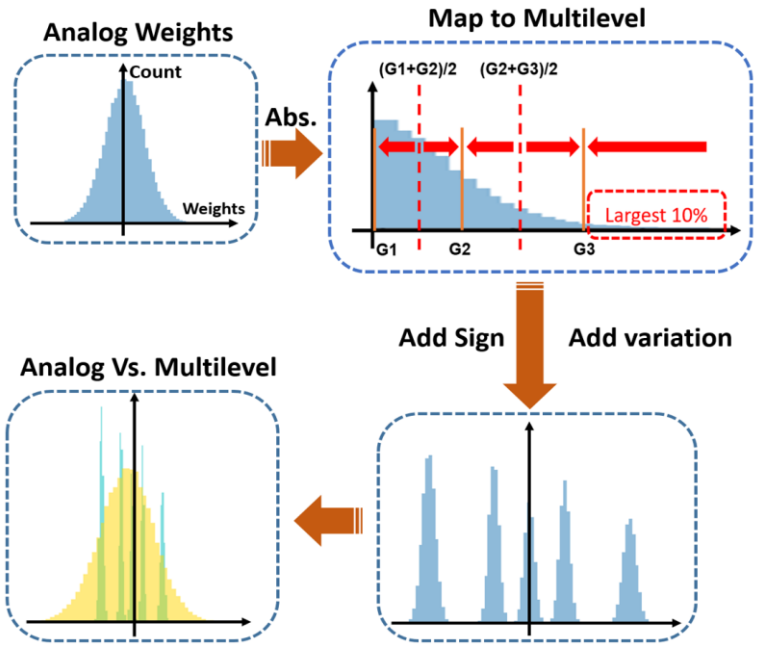
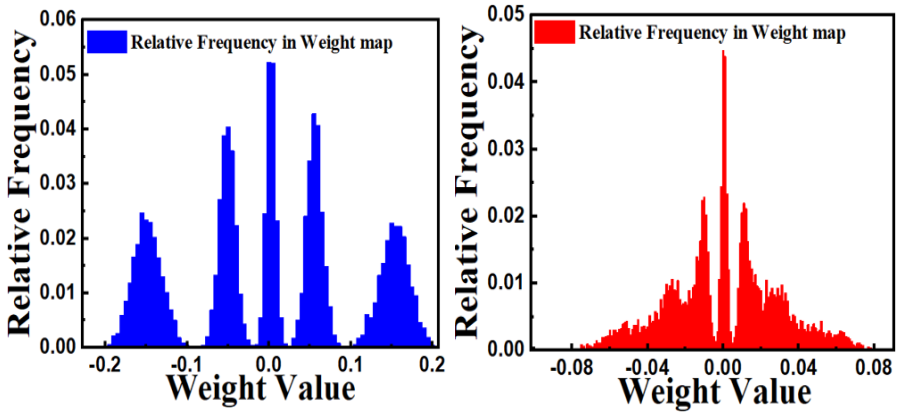


Fig. 4.3. Diagram of dynamic quantization and mapping with 3-level RRAM devices.

that different from the failure mask which will be updated in each programming cycle, the stuck mask keeps the same value during the training and inference since the stuck device cannot be fixed and the stuck position is fixed after array testing.

4.3.2 Dynamic Weight Quantization and Mapping

The conductance of memristor devices in crossbar array is programmed to store the trained neural network weights. Because of the limited conductance levels, the software weights need to be quantized and mapped to the memristor conductance. Ideally, the memristor conductance should be evenly distributed across the range from LRS and HRS. However, due to the device fabrication limitations and the inherent transport mechanism, the actual conductance distribution usually deviates from the ideal value. It becomes non-uniform distribution and even has severe overlapping between adjacent levels. Hence, a dynamic quantization and mapping method is proposed in this work to map the weights to non-uniformly distributed memristor conductance to minimize the impact of non-ideality.



(a) (b)

Fig. 4.4. Impact of the line resistance on the 'real' weights in the neural network. (a) Multilevel weights in a 128×128 array mapped from the trained analog weights, 25% device variation is added; (b) Equivalent 'real' weights in the 128×128 crossbar after IR-drop added, which is dependent on the relative location in the crossbar array. The interconnect line resistance between two cells is set to 1Ω.

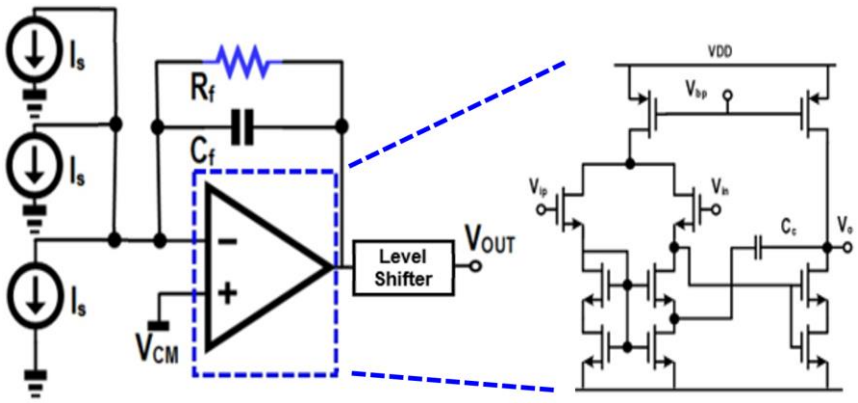
The process of dynamic quantization and mapping is illustrated in Fig. 4.3. As an example, trained ideal weights are mapped to 3-level RRAM devices. The conductance levels from low to high are G_1, G_2, G_3, respectively. Firstly, the ideal weights, w_a, are transferred to absolute value and the signs are recorded. Secondly, the largest $t\%$ tail weights are assigned to G_3 directly and left maximum weight is marked as $w_{max\text{-}t}$, where $t=10$ in the example. The left weights are mapped to conductance weights, w_c, according to the following equation:

$$w_c = w_a \frac{G_3 - G_1}{w_{max-t}} + G_1 \tag{4.2}$$

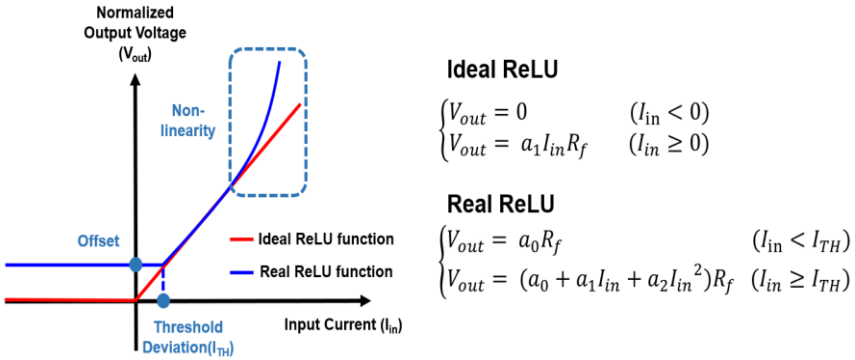
Then the conductance range is divided according to the intermediate value of each two adjacent levels. The conductance weights are assigned to each RRAM according to the regions. Finally, the recorded signs are added, and the variations are generated to each weight according to the levels.

4.3.3 Crossbar Non-idealities

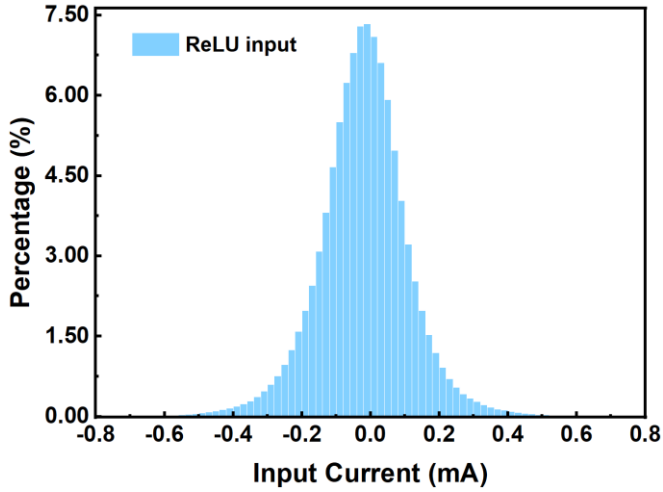
The non-idealities at crossbar array level mainly come from line resistance and sneak path [46]. The actual voltage across the memristor device will be smaller than the



(a)



(b)



(c)

Fig. 4.5. (a) Diagram of a typical neuron circuit for the ReLU and I-V converter function. (b) Diagram of non-idealities and the modelling of the realistic ReLU circuits. The non-idealities are input threshold, output offset and non-linearity. (c) The input current distribution of ReLU circuit.

theoretical value because of the accumulated IR-drop on the interconnect traces and sneak paths. As illustrated in Fig. 4.4, the combined impact of line resistance and conductance overlapping will make the actual conductance level undistinguishable and

affect the NN accuracy.

Several related works have been proposed using linear technique or iterative technique [66-67] to estimate IR-drop in crossbar. The most straightforward solution is to calculate the crossbar current distribution by solving the Kirchhoff's laws equations. With iterative matrix solvers, the time complexity can be reduced to $O(m^2n^2)$. However, it is still time-consuming especially when the crossbar size scales up. To accelerate the simulation speed, a new approach was proposed to treat the voltage and current deviations as scaling factors matrices applied to the input voltage and output current. This model achieved fast computation speed with low current distortion as well as low time complexity of $O(mn)$ at the same time. More technical details can be found in [90].

4.3.4 Peripheral Circuitry

In most previous works, ADC is used to read out the result of the crossbar array. Then the nonlinear activation functions such as RELU and Sigmoid are realized in digital domain. However, utilizing ADC will both sacrifice the circuit area and the power consumption, which is a major obstacle for the implementation [15]. To improve the efficiency of the system, a transimpedance amplifier (TIA) is designed as shown in Fig. 4.5(a) to realize the current to voltage conversion, partial sum of current for subarrays and ReLU activation function. The non-idealities of the TIA, such as the input offset, the threshold deviation, and the non-linearity, as illustrated in Fig. 4.5(b), will induce computation error and further reduce the accuracy.

To simulate the impact of the TIA non-idealities, the ReLU function is modelled as a polynomial with second order effect:

$$\begin{cases} V_{out} = a_0 R_f & (I_{in} < I_{TH}) \\ V_{out} = (a_0 + a_1 I_{in} + a_2 {I_{in}}^2) R_f & (I_{in} \geq I_{TH}) \end{cases} \tag{4.3}$$

where I_{in} and V_{out} are function input current and output voltage. I_{TH} is the threshold

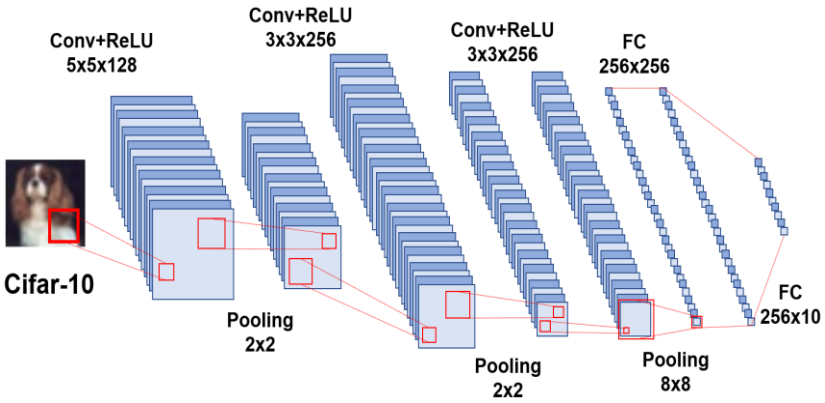


Fig. 4.6. Structure of the simplified 5-layer VGG network that includes 3 convolution layers (Conv) and 2 fully connected layers (FC).

deviation. a_0 is the output offset. a_1 is the linear coefficient and is set to 1 in this work. a_2 is the nonlinear coefficient. Fig. 4.5(c) shows the simulated distribution of the input current based on the simulation of the neural network in Fig. 4.5 (c). It can be observed that the current is within ±0.5 mA. Hence R_f is set to 1kΩ to constrain the output voltage to be within 1V.

4.3.5 Non-idealities Aware Training

To improve the reliability and accuracy of the neural network inference, the non-idealities aware training, an offline training method, is proposed. The weights are updated by the software and directly programmed into the devices. The training flow is illustrated in Fig. 4.2. Before the start of training, the above-mentioned non-idealities have been extracted from the crossbar measurement results. To calculate the loss function in the back propagation (BP) training algorithm, the calculations will be adjusted by incorporating the realistic non-idealities: first, the analog weights are stored, and mapped to discrete states based on the multilevel characteristics and program failure probability.

Then, line resistance is introduced during the VMM operations, next, the activation function output is calculated with the circuit non-idealities; finally, the loss function and

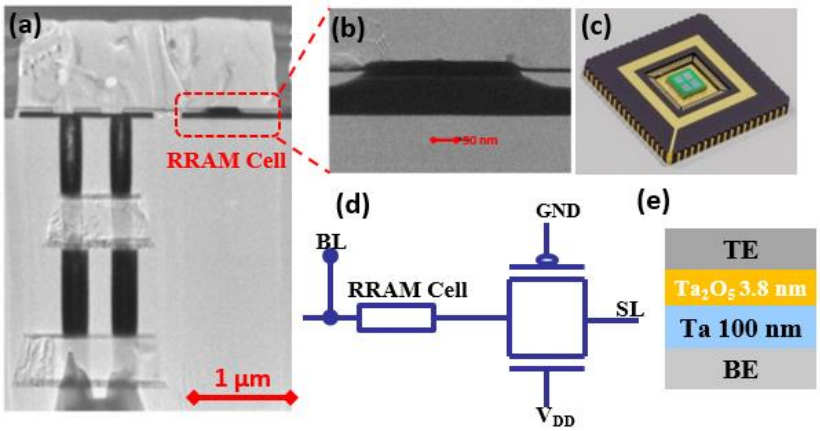


Fig 4.7. (a) SEM photo of RRAM cell cross-section; (b) zoom-in TEM photo of the RRAM cell; (c) 64kb array chip; (d) diagram of the 2T1R unit cell; (e) device stack of the RRAM.

the weight update is calculated. In this manner, the network is trained along with the non-idealities, adjusting itself to maintain the recognition accuracy.

4.4 Implementation and Results

To demonstrate the efficacy of the proposed software-hardware co-design framework, an 5-layer simplified VGG model is built for image classification task using CIFAR-10 dataset. As shown in Fig. 4.6, the network consists of 3 convolution layers, 3 pooling layers and 2 fully connected layers. It should be highlighted that the proposed framework can also be applied to other types of memristor devices and neural network architectures. The memristor model is based on the measurement results of 128×128 64kb TaOx 2T1R RRAM crossbar chip [73]. The RRAM device structure and material stacks are integrated with 0.18-μm CMOS baseline wafer via the backend-of-line (BEOL) integration process. The device SEM images, cell structure and the photo of packaged chip are shown in Fig. 4.7. The measured cumulative distribution function (CDF) in Fig. 4.8 shows huge resistance overlap among the 3 states. All the non-idealities are summarized in Table. 4.1. According to these parameters, the non-idealities aware training is performed, and the inference is simulated with the trained

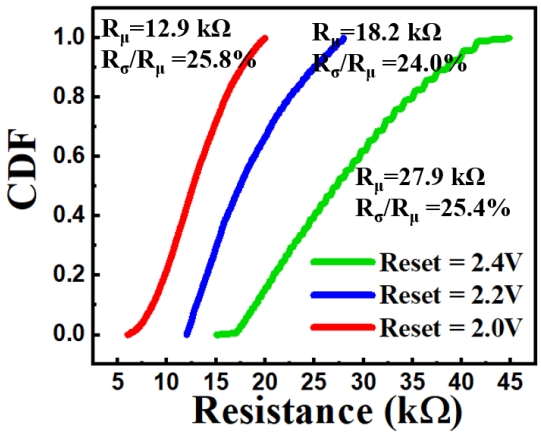


Fig 4.8. Measured cumulative distribution function (CDF) of the 3 states of the RRAM devices.

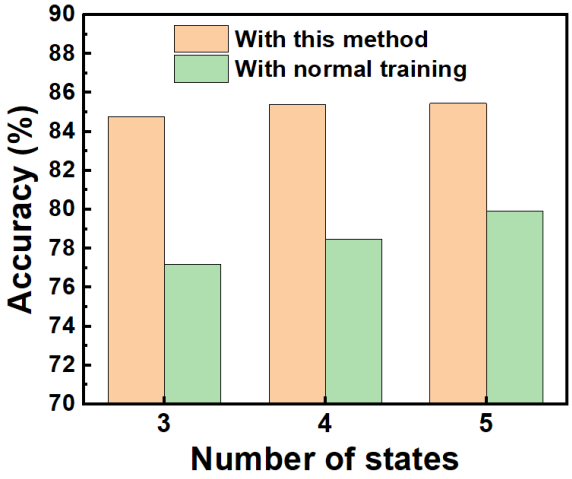
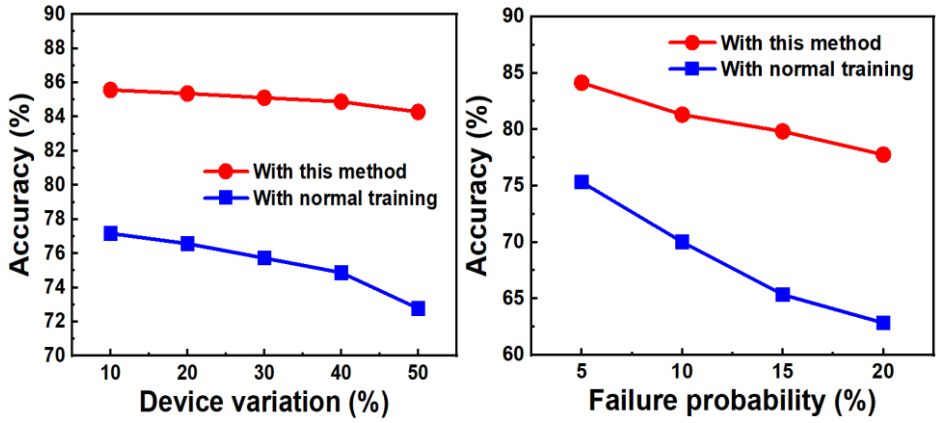


Fig 4.9. Comparison of recognition accuracy between non-idealities aware training and normal training method with different resistance states and 25% device variation.

weights mapped on the devices.

One of the significant considerations for the non-ideality-aware training method is the increased training cost compared to traditional methods. The incorporation of realistic non-idealities into the training process introduces additional computational overhead, as each training iteration must account for device-level conductance variations, programming failures, and peripheral circuit non-linearities. Despite this, the proposed method demonstrates superior inference accuracy, which justifies the higher training cost in scenarios where accuracy is critical. Future work will focus on optimizing the training algorithm to reduce this overhead and exploring hardware



(a) (b)

Fig 4.10. Recognition accuracy comparison of non-idealities aware training and normal training method with (a) different device to device resistance variation (defined as 3σ/μ) and (b) different program failure probability where 25% device variation exists.

acceleration techniques to further mitigate the increased training costs.

4.4.1 Device Non-Idealities

The impact of the number of device conductance state to training accuracy is shown in Fig. 4.9. Compared to the software accuracy, the proposed non-idealities aware training only suffers <2% accuracy loss after mapping the weights to the hardware for 3-5 states with 25% device variation (defined as 3σ/μ in this work) added, while the loss for the normal training is around 10%. Also, the normal training is more dependent on the number of states than non-idealities aware training. The number of states is set to 3 in this part. The impact of the device variation as shown in Fig. 4.10(a) indicates an excellent accuracy maintenance of the proposed method, with <2% degradation with the increase of the variation to 50%. The impact of the program failure probability is shown in Fig. 4.10(b). Though the non-idealities aware training will suffer around 5% accuracy loss compared to the software result, it is more robust than the normal training with the increase of the program failure probability.

4.4.2 Array Non-Idealities

As mentioned in Section 4.2.1, the line resistance connecting adjacent cells will reduce the voltage applied on the RSM cell, depending on the array size, the line

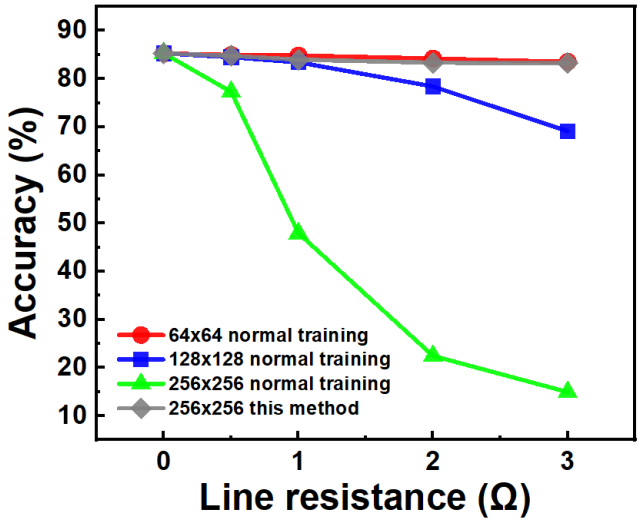
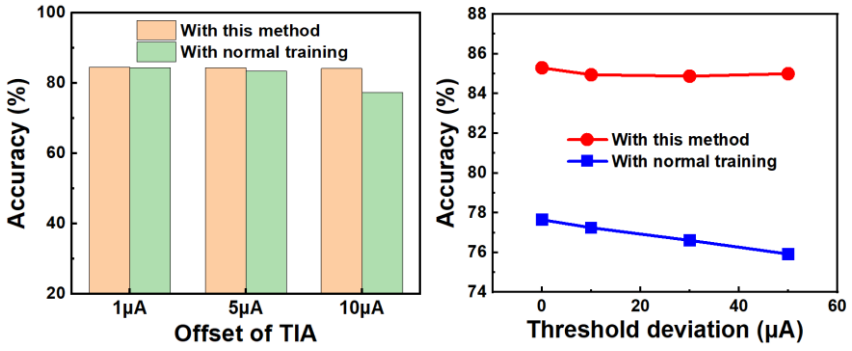


Fig 4.11. Impact of the line resistance and the array size on the accuracy. The impact is also dependent on the average resistance of the devices in the array.



(a) (b)

Fig 4.12. Comparison of non-idealities aware training and normal training with (a) increased output offsets of the ReLU circuits where threshold deviation is 0μA and (b) increased input threshold deviation of the ReLU circuit where the offset is 10μA. The non-linear part is 1% of the total output.

resistance and the average conductance. Refer to Fig. 4.4, in the distribution of the 'real' weights, the high weight states will be shifted to the center depending on the position in the array thus influencing the accuracy. Fig. 4.11 shows the severe degradation of the accuracy induced by the line resistance with various array size and 20kΩ average RSM resistance and the effect of the proposed training method against the array size and line resistance. Compared to the 256×256 array with the most severe loss, the non-idealities aware training can still maintain the accuracy when line resistance increased to 3Ω.

4.4.3 Circuits Non-Idealities

The hardware realization of the ReLU activation function can be achieved using the

TABLE 4.I MEASURED CROSSBAR PARAMETER SUMMARY

Non-idealities	Value
Number of states	3
Average R (kΩ)	12.9/18.2/27.9
Device variation (3σ/μ)	75%
Array size	128x128
Line resistance	0.72Ω
Output offset	10 μA
Threshold deviation	50 μA
Non-linear/Linear output	1%

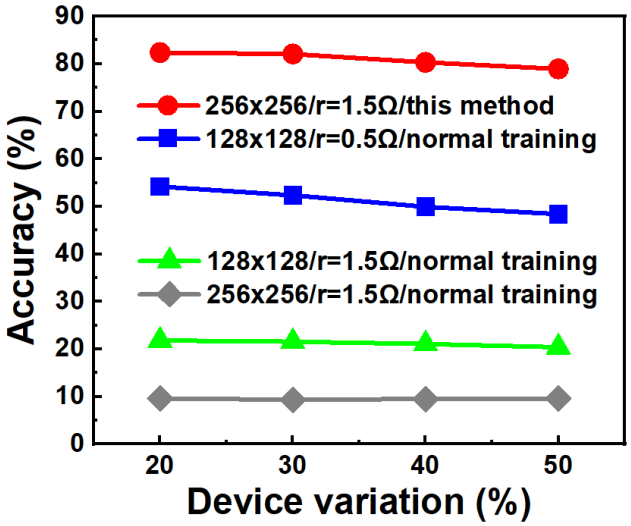
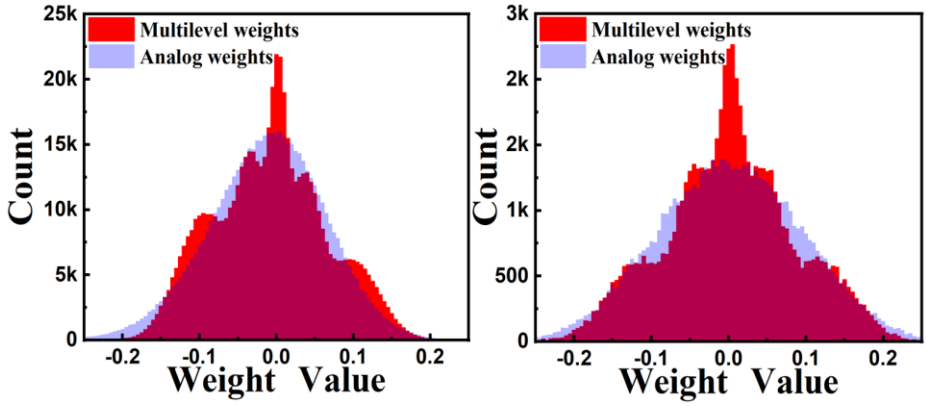


Fig 4.13. Comparison of non-idealities aware training and normal training with all non-idealities. For the ReLU circuits, the threshold deviation is 50μA, the offset is 10 μA and the non-linear part is 1% of the total output.

TIA as mentioned in Section II.C. The non-idealities of the TIA, such as the output offset, the threshold deviation and the non-linearity, are introduced in the modeling. Fig. 4.12 show the robustness of the non-idealities aware training against the output offset and the threshold deviation respectively, compared to the normal training. The results indicate that the proposed method effectively relaxes the requirements for the ReLU hardware realization.

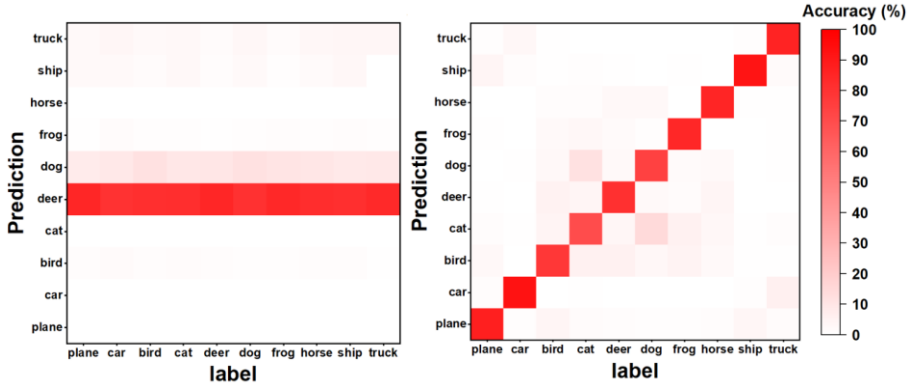
4.4.4 Interactions Among Non-Idealites

In the actual crossbar, all these non-idealities interact with each other thus



(a) (b)

Fig 4.14. Comparison of the weight distribution between the ideal analog weights trained by normal training and the 3-level weights in the inference system after the non-idealities aware training. (a) Weights in the 3rd convolution layer; (b) Weights in the 1st fully connected layer.



(a) (b)

Fig 4.15. The heat map of the confusion matrix referring to the classification correlation of the demonstrated inference system. (a) The normal training method; (b) The non-idealities aware training method. The overall recognition accuracy of the inference with normal training is around 10% while with non-idealities aware training is 83.05%.

magnifying the accuracy degradation induced by each separately. The impact of all non-idealities is evaluated simultaneously in the inference system and shown in Fig. 4.13. By offline training, the inference accuracy is around 75% with 256×256 array, 1.5Ω line resistance and 50% device variation.

The designed circuit mentioned in Section II.C realizes the function of ReLU and the I-V converter for the input of next layer. All the non-idealities are summarized in Table. 4.I. According to these parameters, the training is performed, and the inference is simulated with the trained weights mapped on the devices. Fig. 4.14 shows the

TABLE 4.II PERFORMANCE COMPARISON

	[91]	[92]	[93]			**This work**
Device	HfO2	HfO2/Al2O3	AlOx/HfO2	TaOx/HfOx	EpiRAM	TaOx
Network Structure	VGG-11	VGG	VGG-8	VGG-8	VGG-8	Simplified VGG-5
Crossbar Size	256×256	1152×128	128×128	128×128	128×128	128×128
Device Number of States	NR	16	40	128	64	3
Ron (Ω)	11k	50k	16.9k	100k	81k	12.9k
On/Off Ratio	1.9	16	4.43	10	50.2	2.2
Device Variation (3σ/μ)	NR	12%	15%	11.10%	6%	75%
Number of Weights	7.66M	6.6M	13M	13M	13M	1M
Line resistance (Ω)	NR	0.1	NR	NR	NR	0.72
SA Fault	10%	NR	NR	NR	NR	2%
Inference Accuracy (%)	83	79.4	37	81	85	83

*NR: Not Reported

distribution of the weight after the non-idealities aware training. Compared to the analog weight by normal training, the distribution clearly aggregates to 3 discrete states with an obvious overlap due to the large device variation. Fig. 4.15 shows the comparison of the recognition performance between the proposed training and the normal training. With all these non-idealities, the inference system with normally trained weights has lost the recognition function and will identify most figures as cat, while the non-idealities aware training trained inference can identify most categories correctly, with an 83.05% overall accuracy, around 3% loss compared to the software. The results also indicate that the best training method for the RSM-based neuromorphic system is on-chip performing the loss function computation.

The performance of the proposed method is summarized in Table. 4.II and compared with other recently published DNN implemented with memristor crossbar. For fair comparison, the inference results for the same CIFAR-10 image classification task are compared. It can be observed that the proposed framework achieved comparable inference accuracy with only 3-state device and a much smaller size network.

4.5 Summary

In this chapter, a modeling and simulation framework for edge-AI DNN implemented on memristive crossbar is presented. Non-idealities aware training is proposed to achieve fast and accurate early exploration of the system design and reduce the degradation of inference accuracy. The impact of each non-ideality from the devices, arrays and circuits is analyzed, evaluated and then the modeling tool is demonstrated. Implemented in a hybrid fashion of Python and PyTorch, the proposed framework is evaluated with a simplified 5-layer VGG network implemented on measured 128×128 RRAM array with 3-level weight resolution. For CIFAR-10 tasks, 83% inference accuracy is achieved with less than 3% accuracy drop compared to the ideal model. While the current verification is primarily based on simulation results, future work will involve validating these findings through hardware implementation. This will include fabricating prototype devices and conducting extensive experimental evaluations to confirm the simulation results and refine the models further.

Chapter 5. PoolFormer on RRAM Crossbar for Edge-AI

Chapter 4 aims at solving the problems of hardware aware training in co-design, which is inevitable in practice. In this chapter, a memristor-based PoolFormer modeling and training framework for edge-AI applications is presented. The original PoolFormer structure is further optimized for hardware implementation on crossbar by replacing the normalization operation with scaling. In addition, the non-idealities of RRAM crossbar from device to array level as well as peripheral readout circuits are analyzed. By integrating these factors into a single network training framework, the overall neural network performance is evaluated holistically. The impact of nonidealities to the network performance can be effectively mitigated by incorporating them into the training process.

5.1 Recent Works

Transformer has demonstrated remarkable improvement in classification accuracy in the areas of computer vision and natural language processing [94-99]. Transformer's excellent performance is attributed to the substantial computation resources needed for the self-attention process. Although the literature has reported on a cloud cluster and a tailored Transformer accelerator chip [2-3,6], both solutions aimed for large-scale applications. They are not suitable for running on battery-powered resource constraint edge-AI devices. A few lightweight transformer models have been reported [98-100]. HAT [100] focused on searching for the best Transformer structure that suit different hardware. LeViT [101] reintroduces the convolution operation to improve the tradeoff between the inference speed and accuracy, making an architecture similar to LeNet in convolutional neural network (CNN). Vit-Lite, CVT and CCT [102] deployed

convolution to extract low-level information and removing class tokens to improve the performance on small dataset. Recent research shown that the transformer model's competence is mostly related to its overall architecture with a new model called PoolFormer described in [103]. Instead of the computationally intensive self-attention process, only pooling operation is required. Since it requires no parameters in the memory and the processing procedure is much simplified, tremendous amount of computation resources can be saved.

A multitude of methods have been investigated to implement attention operation and peripheral operations like active function and logical operations. There are also efforts to build hardware-software co-design framework to simulate and evaluate the performance of Transfromer network [104-107]. For example, ReTransformer [104] decomposes the self-attention operation to reduce the number of frequent reloadings of intermediate results and transfers the softmax function to logical inference with look up table. ReBERT [105] also used look up table for softmax function and presented serial pipeline structure to accelerate the feed forward process. However, these reported solutions still require considerable computation resources. The ADC/DAC consumes around 80% of the total power to realize the essential signal transfer between analog and digital processes. A quantization and regularization method to reduce the power consumed by ADC has been proposed in [106] but the circuit nonidealities had not been considered in the model. Full analog circuit design has no requirement for ADC and had been proposed in [107] but the evaluation data is much simpler than real applications. Furthermore, the impact of nonidealities of analog memristor model and crossbar array on Transformer network are not examined. Moreover, to include the non-idealities into MLP and CNN training and inference procedures, a variety of approaches have been

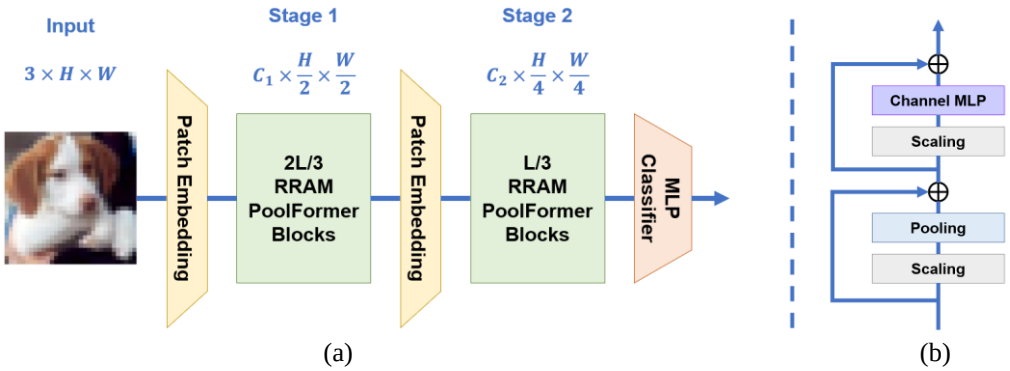


Fig. 5.1. (a) The overall framework of Edge PoolFormer that adopts hierarchical architecture with 2 stages. For a model with L Edge PoolFormer blocks, stage [1, 2] have [2L/3, L/3] blocks, respectively. The embedding dimension stage i is C_i shown in the figure. (b) The Edge PoolFormer block architecture in software training is presented, which differs from the Transformer block by using a simple non-parametric operator, pooling, instead of attention as basic token mixer.

investigated. There are additional attempts to provide extensive training and simulation frameworks to assess the performance of multi-layer perceptron (MLP) and CNN [31-34,82-85,108]. However, controlling the model size holds immense significance within edge device applications. Traditional neural network implementations like VGG necessitate millions of parameters, imposing an excessive resource burden at the edge. Thus, there is an urgent requirement for a modeling and simulation tool centered on memristor crossbars, particularly tailored for assessing the efficiency of lightweight transformer neural networks.

5.2 Edge PoolFormer

5.2.1 Edge PoolFormer Structure

The attention-based token mixer is widely recognized as the key factor behind the success of Transformers. However, attention operation is a challenge for processing-in-memory (PIM) since it contains large number of parameters, intermediate matrix and the nonlinear active function softmax. Although reorganizing the attention steps can reduce the number of intermediate matrix [104] and the pointwise exponent in softmax

can be approximated using Taylor series expansion [109], the crossbar array writing process as well as data transfer between analog and digital process are still required.

The structure of the Edge PoolFormer inspired from PoolFormer [103] is shown in Fig. 5.1. In the proposed structure, the attention operation is replaced by the non-parametric average pooling operation to avoid the implementation challenges stated above. Since it requires no parameters in the memory and the process is extremely simple, huge computation resources and storage are saved and simple analog process is applicable, which fits the demand of edge devices. Additionally, compared to the conventional PoolFormer, the proposed structure has following novelties to adapt to edge device applications:

- *Layer Normalization is replaced by Scaling*

Layer normalization plays an important role in the original PoolFormer. However, the computation of the average value and the variance in analog process for the real-time signal is complex. Although the previous work had proposed a circuit solution for this normalization process [107], this three-stage design includes subtraction sub-circuit, summation sub-circuit and square root sub-circuit, which consumes large static power and area.

In light of the aforementioned considerations, a re-evaluation of the rationale underlying the incorporation of layer normalization becomes warranted. Normalization is used in modern large-scale neural networks to increase their generalization ability, reduce their dependence on parameter initialization, and accelerate model training. In other words, it makes deep neural networks feasible while also speeding up the stability and convergence of the model. Nevertheless, in the context of small-scale models tailored for applications in edge computing, the integration of shortcut paths has already

proven to be satisfactory in fulfilling the prerequisites for achieving profound model convergence. To improve the generalization ability of the model, data standardization is necessary. However, most edge computing applications have small sample sizes and the same feature shapes, such as image recognition and keyword detection. Therefore, batch normalization is more suitable for training and the process is shown below.

$$\mu_c = \frac{1}{NHW}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W} x \tag{5.1}$$

$$\sigma_c = \sqrt{\frac{1}{NHW}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W}(x-\mu_c)^2 + \epsilon} \tag{5.2}$$

$$\begin{aligned} x_{BN} &= \gamma\left(\frac{x-\mu_c}{\sigma_c}\right) + \beta \\ &= \frac{\gamma}{\sigma_c}x + \left(\beta - \frac{\gamma\mu_c}{\sigma_c}\right) \\ &= scale \cdot x + shift \end{aligned} \tag{5.3}$$

where, x is the input sample of size (n, c, h, w), μ_c is the mean value of the channel, σ_c is the standard deviation of the channel, γ and β are the learnable parameters and x_{BN} is the batch normalization output.

It is obvious that the output of batch normalization is the scale and shift operation for each channel. The bias of MLP in each block already acts as the shift, so adding additional scaling to each channel is logical. Although the additional scaling constants need to be initialized and learned, which increases the instability of the model, it does not have a practical impact on small models. At the same time, the scaling constant, as a learnable parameter that is not actually related to the feature, increases the generalization ability and model accuracy of the model.

To balance the performance of neural network and its circuit efficiency, layer normalization is replaced by channel scaling. The effect of channeling scaling and layer

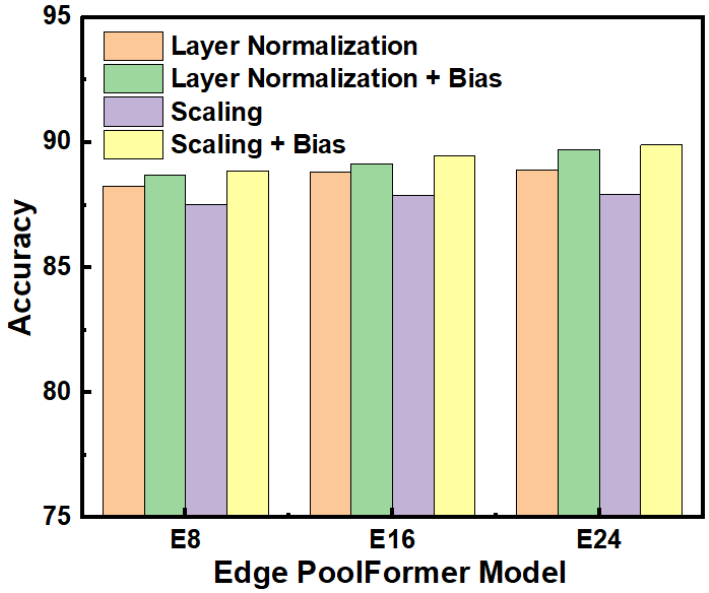
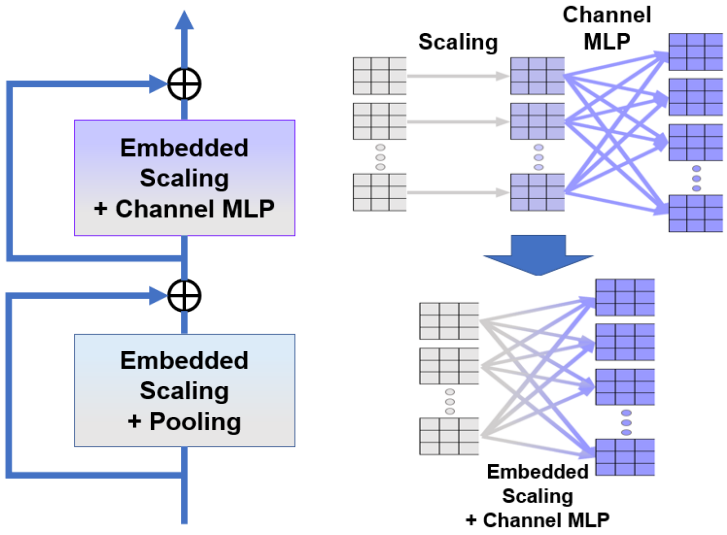


Fig. 5.2. The comparison of Edge PoolFormer model performance with conventional layer normalization and proposed channel scaling layer on CIFAR-10.

normalization is evaluated in 200 epoch Edge PoolFormer models and the result is shown in Fig. 5.2. The result verifies that the channel scaling layer with bias is able to replace the conventional layer normalization in this model. The scaling factors are respectively trained and multiplied by each channel. Compared to conventional normalization methods, channel scaling reduces the consumption of the circuits and achieves satisfactory performance on the small datasets in edge AI applications.

- *2-stage structure and smaller embedding dimension*

The conventional Vision Transformer setup typically involves four stages using block distributions of $L/6$, $L/6$, $L/2$ and $L/6$ for data processing, a structure that subsequent works commonly follow. However, within edge applications, specifically in scenarios dealing with smaller datasets and constrained computational resources, the priority lies in minimizing both storage and computational demands. Departing from this traditional approach, our proposed Edge PoolFormer implements a distinct 2-stage structure employing $2L/3$ and $L/3$ block distributions. Moreover, to effectively reduce



(a) (b)

Fig. 5.3. (a) The architecture of Edge PoolFormer block in hardware implementation. (b) The diagram of the change of scaling and channel MLP operations from software training to hardware implementation.

parameters, embedding dimensions of 32 and 64 are specifically chosen for each stage, catering to the constraints inherent in edge device environments.

- *Scaling is embedded in implementation*

During software training, scaling typically functions as a standalone layer. However, in our proposed system, during crossbar array implementation, scaling is strategically integrated with other operations like pooling and channel MLP within the training process. This consolidation optimizes processing efficiency, as depicted in the implementation framework sketch of the Edge PoolFormer block in Fig. 5.3(a). An illustrative example showcasing scaling embedded within the channel MLP is detailed in Fig. 5.3(b).

This cohesive integration of operations significantly economizes the space between layers, effectively reducing system latency and power consumption. Such optimization aligns perfectly with the requirements of edge devices, where resource efficiency is paramount. This approach ensures that the system maximizes its performance within the limitations inherent to edge environments.

Table. 5.I. Edge PoolFormer hyper-parameters

Stage	# Token	Layer Specification		Edge PoolFormer		
				E8	E16	E24
1	$\frac{H}{2} \times \frac{W}{2}$	Patch Embedding	Patch Size	3×3, stride 2		
			Embed. Dim.	32		
		Edge PoolFormer Block	Pooling Size	3×3, stride 1		
			MLP Ratio	4		
			# Block	6	12	18
2	$\frac{H}{4} \times \frac{W}{4}$	Patch Embedding	Patch Size	3×3, stride 2		
			Embed. Dim.	64		
		Edge PoolFormer Block	Pooling Size	3×3, stride 1		
			MLP Ratio	4		
			# Block	2	4	6
Parameters (M)				0.13	0.246	0.358
MACs (M)				1.75	2.1	2.44

5.2.2 Performance Analysis

In this work, three models with 8, 16 and 24 Edge PoolFormer blocks are evaluated on Cifar-10 dataset, and their hyper-parameters are shown in Table. 5.I. Each model has 2 stages with $\frac{H}{2} \times \frac{W}{2}$ and $\frac{H}{4} \times \frac{W}{4}$ tokens, respectively. The models underwent 300 training cycles using stochastic gradient descent (SGD) optimization methodwith batch size 125, weight decay $1e^{-4}$, peak learning rate $1e^{-3}$ and momentum 0.8. The dropout is set as 0.5 and the droppath is set to 0.1 to help train the models. Our implementation is built on the PoolFormer codebase [103], and the experiments are conducted using GPUs. Table. 5.II shows the performance of the Edge PoolFormer on Cifar-10 classification. Edge PoolFormer-E8 reaches the top-1 accuracy of more than 89%, requiring only 0.13M parameters and 1.7M MACs. In comparison, the widely used convolutional network ResNet18 [108] achieves a slightly higher accuracy of 90.27%. However, it requires 86 times more parameters and 24 times more multiply-accumulate operations. To obtain similar accuracy, a small-scale Transformer model ViT-Lite7/8 [102] needs

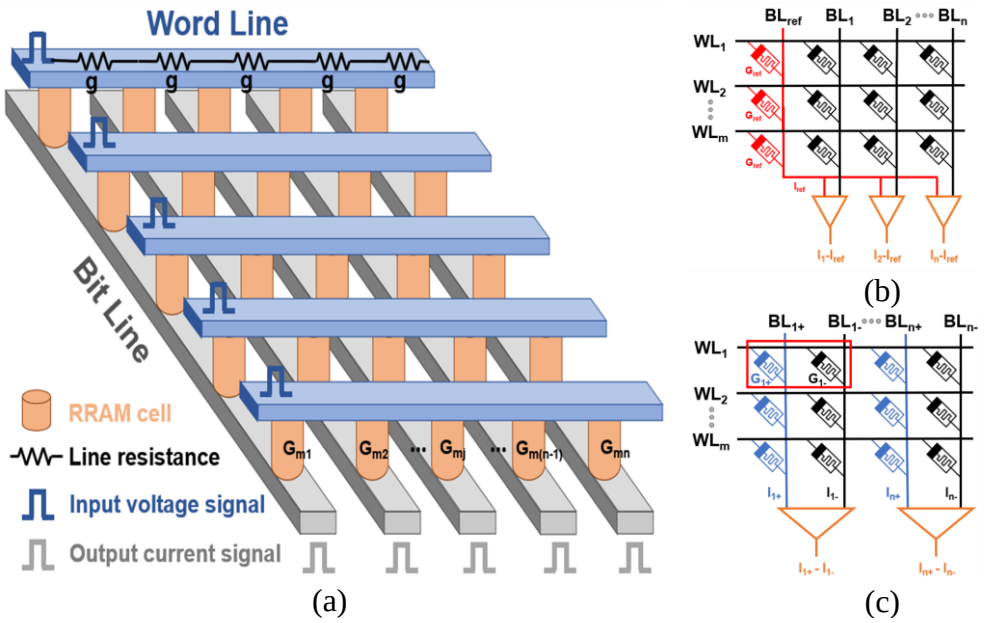


Fig. 5.4. (a) Block diagram of a m×n crossbar array with horizontal word lines and vertical bit lines, with interconnect line resistances. Line resistance refers to the ohm resistance of the interconnect traces along the top and bottom electrodes of the memristor devices. Structures of (b) one-cell-one-weight scheme with reference current and (c) two-cell-one-weight scheme with differential pairs.

29 times parameters as well as 35 times computation while only 89.1% accuracy is attained. When compared with the much-improved compact transformer variants [102], the Edge PoolFormer still shows a better balance between the performance and resources. Specifically, CCT-2/3 obtains 89.75% top-1 accuracy with 0.28M parameters and 40M MACs while the Edge PoolFormer-E16 reaches 89.52% with 15% fewer parameters (0.24M) and 95% fewer MACs (2.1M).

5.3 Memristor Crossbar Array

5.3.1 Vector Matrix Multiplication (VMM)

A typical memristor crossbar array is shown in Fig. 5.4(a). It comprises of memristor devices at each cross point along with a group of horizontal word lines (WL) and vertical bit lines (BL). Line resistance refers to the resistance in ohms of the connection traces that run between the top and bottom electrodes of the memristor devices. Using a deep submicron CMOS technology, the unit line conductance g

Table. 5.II. Comparison of network top-1 validation accuracy on CIFAR-10 dataset

Architecture	Model	Image Size	Parameters (M)	MACs (M)	Epoch	Top-1%
Convolutional Network	ResNet18	32	11.18	40	300	90.27
	ResNet34	32	21.29	80	300	90.51
	MobileNetV2/0.5	32	0.7	10	300	84.78
	MobileNetV2/1.0	32	2.24	10	300	89.07
Vision Transformer	ViT-12/16	32	85.63	430	300	83.04
	ViT-Lite7/16	32	3.89	20	300	78.45
	ViT-Lite6/16	32	3.36	20	300	78.12
	ViT-Lite7/8	32	3.74	60	300	89.1
	ViT-Lite7/8	32	3.22	60	300	88.29
Compact Transformer	CVT-7/8	32	3.74	60	300	89.79
	CVT-6/8	32	3.21	50	300	89.5
	CCT-2/3	32	0.28	40	300	89.75
PoolFormer	Edge PoolFormer-E8	32	**0.13**	**1.7**	300	**89.13**
	Edge PoolFormer-E16	32	**0.24**	**2.1**	300	**89.52**
	Edge PoolFormer-E24	32	**0.35**	**2.4**	300	**90.28**

between two neighboring memristor devices is in the region of a few Ohms [60].

When a voltage vector, V_{IN}, is applied to the WL, ignoring the influence of line resistance initially, the output current, I_{Oj}, of the j^{th} column can be written as follows.

$$I_{Oj} = \sum_{i=1}^{m} V_{INi} \cdot G_{ij} \tag{5.4}$$

where $1 \leq i \leq m, 1 \leq j \leq n$. V_{INi}, represents the input voltage of i^{th} row, and G_{ij} represents the conductance of memristor device at i^{th} row and j^{th} column.

For crossbar-based computing, the mapping of negative weights is an unavoidable problem. The approach of having one weight per cell with a reference current shown in Fig. 5.4(b) has been reported in several earlier works [88]. Large-scale arrays are currently not possible because of the fluctuation of devices. Thus, the two-cell-one-weight technique shown in Fig. 5.4(c) is applied [15]. One weight is used demonstrated by every two neighboring cells. Positive weight is represented by cells in the left column, while negative weight is indicated by cells in the right column. The difference in bit line current between two columns is detected to determine the real output current I_O.

$$I_O = {I_O}^+ - {I_O}^- = V_{IN} \times (G^+ - G^-) \tag{5.5}$$

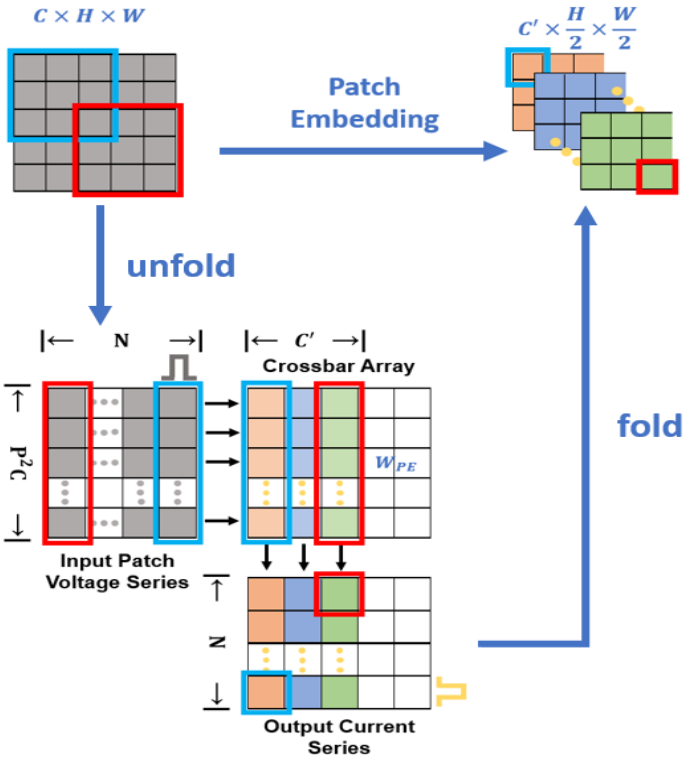


Fig. 5.5. Crossbar array implementation of Patch Embedding.

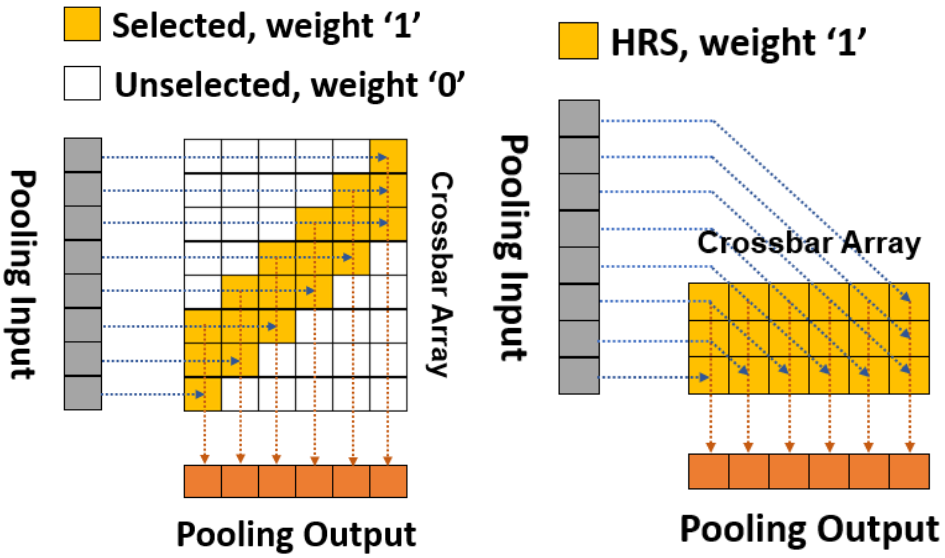
5.3.2 Layer Implementation

- *Implementation of Patch Embedding*

Like ViT, Patch Embedding (PE) is used to downscale the sample, to extract the features before each stage.

$$X = PE(I) = fold(unfold(I) \times W_{PE}) \tag{5.6}$$

where $I \epsilon (C, H, W)$ and $X \epsilon (C', \frac{H}{2}, \frac{W}{2})$, W_{PE} is the learnable weight, P is the path size, C is the input channel, N is the path number and C' is the embedding dimension. The patch size is set to 3 and the stride is fixed at 2 in this work. To implement the embedding process, the input is firstly unfolded to N patches with size (N, P^2C). Thereafter, each patch is converted to the input voltage of the crossbar array and fed into the word lines. After collecting the VMM results with size (N, C'), the output of embedding is folded to the feature maps as shown in Fig. 5.5.



(a) (b)

Fig. 5.6. Crossbar array implementation of Pooling.

- *The Implementation of Pooling*

An average Pooling takes the mean value of the kernel after the layer scaling, and can be represented as

$$Y = avgPool(Scale(X)) \tag{5.7}$$

where $avgPool(\cdot)$ stands for average pooling. The average pooling is the sum of the multiplication result of the elements and the matrix of ones, and then divided by the number of the elements. Since the stride of the average pooling is 1, the high reuse rate of elements makes it possible to process in parallel with crossbar array as shown in Fig. 5.6(a). To save the wasted area of unused devices, the array is reconstructed to Fig. 5.6(b) where HRS devices are engaged to reduce the static power consumption.

- *The Implementation of Channel MLP*

The channel MPL block consists of 2-layer MLP with a scaling and non-linear active function. The detial structure is shown in Fig. 5.7 and the formula is

$$Z = \sigma(Scale(Y) \times W_1) \times W_2 \tag{5.8}$$

where $W_1 \in R^{C\times 4C}$ and $W_2 \in R^{4C\times C}$ are learnable parameters in MLP. $\sigma(\cdot)$ denotes the

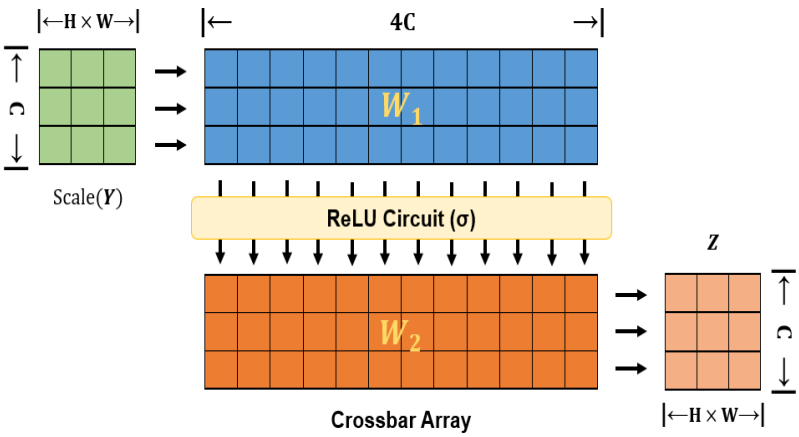


Fig. 5.7. Crossbar array implementation of Channel MLP.

non-linear active function and ReLU is used in this work.

The weights of MLP layer are mapped on crossbar array directly. When the input voltage is applied to the word lines, the result of vector matrix multiplication is collected as output current from the bit lines.

5.3.3 Crossbar Array Nonidealities

- *Device Non-idealities*

Due to manufacturing imperfections, non-ideal memristor device characteristics such nonlinearity, conductance variation and programming failure are inevitable, and they can bring significant impact to the network performance [47]. Therefore, the non-idealities should be considered during the network design stage and compensated appropriately. Therefore, limited conductance levels and variation are considered in the Edge PoolFormer implementation simulation.

The suggested structure divides the devices into levels according to the resolution in the crossbar array. First, the ideal weights, w_a, are converted to an absolute value and the signs are recorded. Second, the largest $t\%$ tail weights are assigned to G_{LRS} directly and left maximum weight is marked as w_{max-t}. According to the following equation, the left weights are transferred to the conductance weights, w_c:

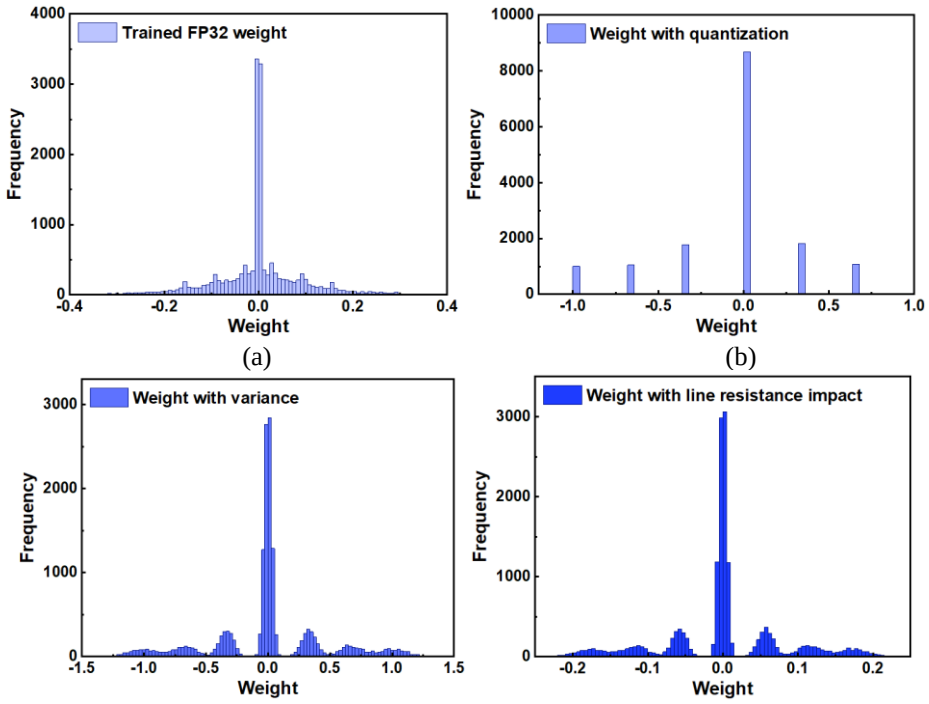


Fig. 5.8. An example of weights mapping with trained weight from channel MLP in stage-2 block-4. The figures show: (a) the trained FP32 weight, (b) the quantized weight, (c) the weight mapping with device nonidealities and (d) the weight mapping with crossbar nonidealities.

$$w_c = w_a \frac{G_{LRS} - G_{HRS}}{w_{max-t}} + G_{HRS} \tag{5.9}$$

The conductance range is then partitioned based on the intermediate value between each pair of neighboring levels. Each RRAM level is given a set of conductance weights based on the regions. The recorded signs are then combined, and variations for each weight are generated based on the levels, with the addition of a normal distribution to reflect the non-ideal behaviors at each level [89].

Each training cycle generates a random mask matrix that represents programming failure based on the measured programming failure rate in order to simulate the effects of writing failure. To obtain the programmed conductance value, the array conductance matrix and this programming failure mask are merged using the AND operation. It should be noticed that the failing devices have HRS status assigned to them.

- *Crossbar Non-Idealities*

Crossbar array non-idealities are primarily caused by sneak path and line resistance [65]. The node voltage between the theoretical and real cases will change due to sneak paths and cumulative IR-drop on the connection traces. As shown in Fig. 5.8, conductance overlapping and line resistance work together to obfuscate real conductance levels and decrease accuracy.

To estimate the IR-drop in the crossbar, iterative and linear approaches have been applied in various studies of a similar research [66-67]. The typical approach is to compute the crossbar voltage distribution using Kirchhoff's laws equations. The temporal complexity can be lowered to $O(m^2n^2)$ using iterative matrix techniques. Even so, it continues to take a long time, particularly as the crossbar size increases. In order to speed up simulation, a novel method was suggested to manage voltage and current degradations as scaling factor matrices that are applied to the output current and input voltage. This model achieves reduced current distortion while combining rapid computation with time complexity $O(mn)$. [88] provides more technical explanations.

- *Peripheral Circuitry*

In more recent works, the crossbar array's result is read out using the ADC. In the digital domain, nonlinear activation functions like ReLU and Sigmoid are implemented. However, using an ADC will consume up power and circuit area, both of which are significant issues [15]. A transimpedance amplifier (TIA), as illustrated in Fig. 4.5(b), is created to generate partial sum of current for the subarrays, the current to voltage conversion, and ReLU activation function with the aim of enhancing the system's efficiency. The TIA non-idealities, such as the non-linearity, threshold deviation, and input offset, as shown in Fig. 5.9, would result in computation error, lowering accuracy

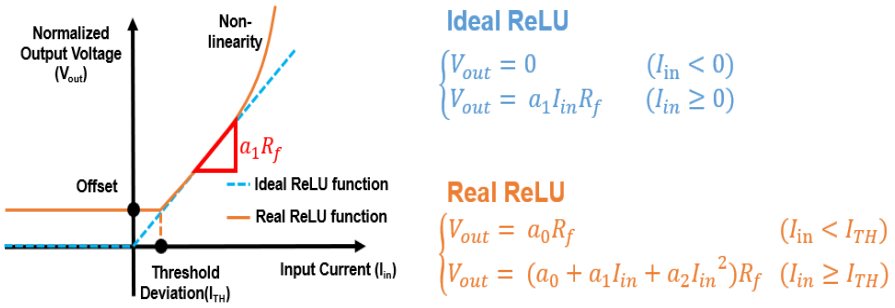


Fig. 5.9. A diagram that depicts the non-idealities and the modeling of practical ReLU circuits is shown. The non-idealities include the input threshold, output offset, and non-linearity.

even further.

The ReLU function is modeled as a polynomial formula with second order effect to replicate the effects of the TIA non-idealities:

$$\begin{cases} V_{out} = a_0 R_f & (I_{in} < I_{TH}) \\ V_{out} = (a_0 + a_1 I_{in} + a_2 {I_{in}}^2) R_f & (I_{in} \geq I_{TH}) \end{cases} \quad (5.10)$$

where I_{in} represents the input current to the function, and V_{out} represents the output voltage, respectively. I_{TH} represents the deviation in the threshold, a_0 represents the output offset, a_1 represents the linear coefficient, which is set to 1 in this study and, a_2 is the nonlinear coefficient.

5.4 Implementation Results

Edge PoolFormer-E16 model is developed for image classification job using CIFAR-10 dataset to show the effectiveness of the suggested software-hardware co-design architecture. The model consists of two stages, with embedding dimension 32 and 64, respectively. The first stage has 12 blocks and the second stage has 4 blocks followed by the MLP classifier. It should be noted that the suggested framework can be used for edge AI vision tasks and various kinds of memristor devices. The memristor model is based on the TaOx 2T1R RRAM crossbar chip measurement results [73]. The RRAM device structure and layer of materials are integrated with a 0.18-μm

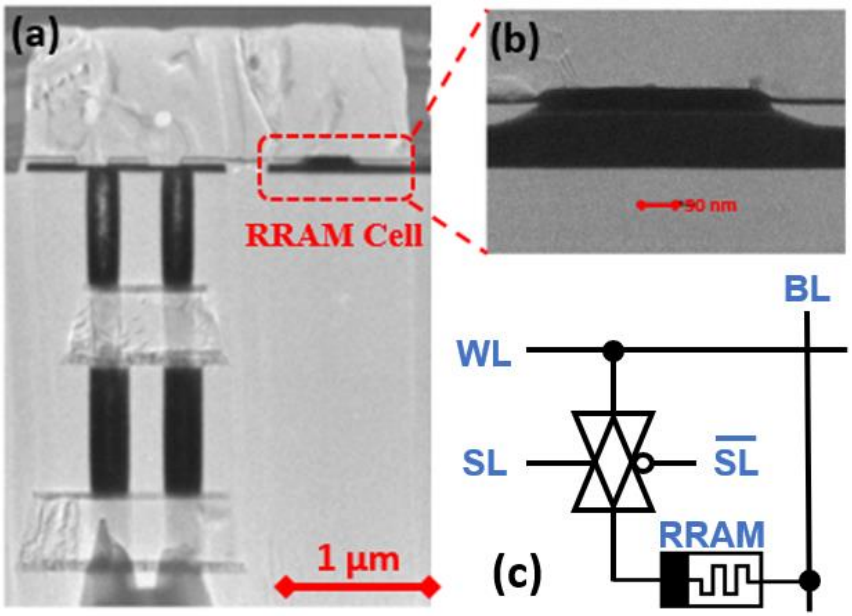


Fig. 5.10. (a) A scanning electron microscope (SEM) photograph of the cross-section of an RRAM cell; (b) a close-up transmission electron microscope (TEM) image of the RRAM cell; (c) a circuit model diagram of the 2T1R unit cell with Ta_2O_5 RRAM.

Table. 5.III. Crossbar parameter summary

Nonidealities	Value
Number of levels	4
Average R (kΩ)	5 / 6.9 / 11.3 / 27.9
Device variation (σ/μ)	10%
Array size	64x64
Line resistance(Ω)	0.5
Output offset	5μA
Threshold deviation	5μA
Nonlinearity	1%

complementary metal-oxide-semiconductor (CMOS) baseline wafer through a backend-of-line (BEOL) integration process and 4-level RRAM device is used in the project as shown in Fig. 5.10.

Table. 5.III provides a summary of all the non-idealities. The mentioned deviations from ideality were only taken from the crossbar measurement results ahead of the beginning of training. The non-idealities aware training [111] is carried out using these parameters, and the inference is assessed using trained weights that are mapped onto the devices. The weights are directly programmed into the devices and are updated by the

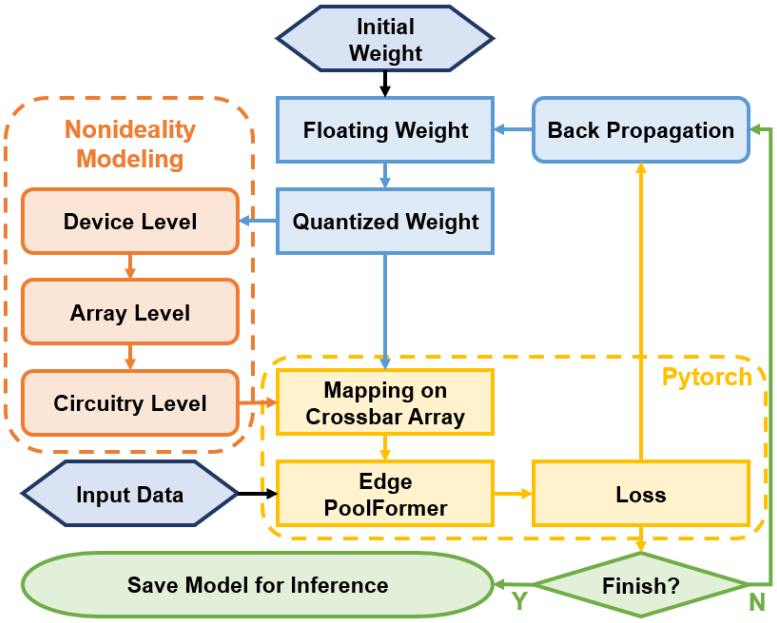


Fig. 5.11. The flowchart of the aware training.

software. The flowchart of the aware training used in this work is shown in Fig. 5.11. In the PyTorch platform, the fundamental training modules are implemented. During the forward pass, weights undergo quantization to specific levels corresponding to RRAM cells. Our modeling accounts for non-idealities during array mapping. In the backward pass, the floating-point weights are updated using local Python functions, leveraging training loss from PyTorch for this purpose. With this approach, the network can be trained with non-ideals and adapts to preserve recognition accuracy.

5.4.1 Nonidealities Impact

Model quantization is the first step for implementing Edge PoolFormer trained weights on crossbar array. As a result, accuracy degradation is inevitable due to the limited resolution. The impact of the device resolution on the training accuracy is shown in Fig. 5.12. The accuracy loss for the conventional offline training is substantially more than that for the nonidealities aware training used on the Edge PoolFormer-E16 after mapping the weights to 2–8 bits resolution where <0.5% loss observed.

Also, the normal offline training is more dependent on the device resolution than the

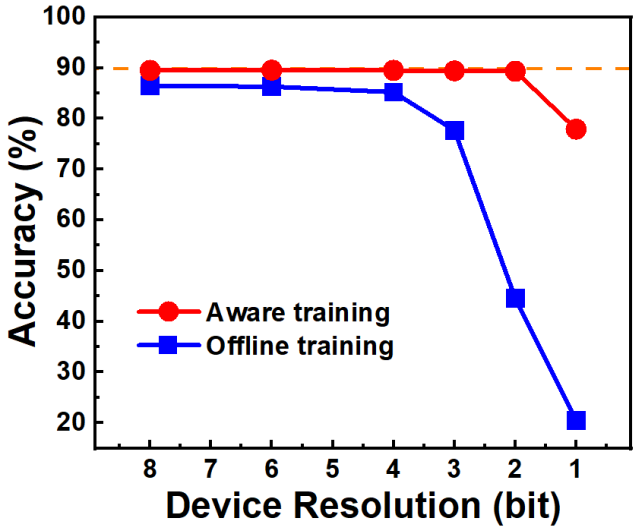
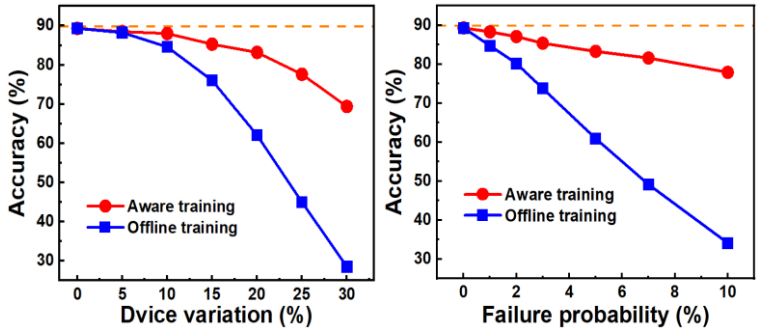


Fig. 5.12. The impact of the device resolution. The software accuracy is shown as dash line for reference.



(a) (b)

Fig 5.13. A comparison of the recognition accuracy between the non-idealities aware training method and the conventional offline training method in Edge PoolFormer, for (a) different device to device variation (defined as σ/μ) and (b) different program failure probability.

nonidealities aware training in Edge PoolFormer. The device resolution is set to 2 bits in training during the following parts.

- *Device Non-Idealities*

With a 6% degradation as the variation up to 20%, the influence of device variation as shown in Fig. 5.13(a), which shows a remarkable preservation of accuracy with the proposed method. Fig. 5.13(b) illustrates the effect of the program failure probability. Even though the non-idealities aware training may experience a 10% drop in accuracy compared to the software result with a 10% programming failure, it is still more reliable than the conventional offline training method, which has a higher probability of

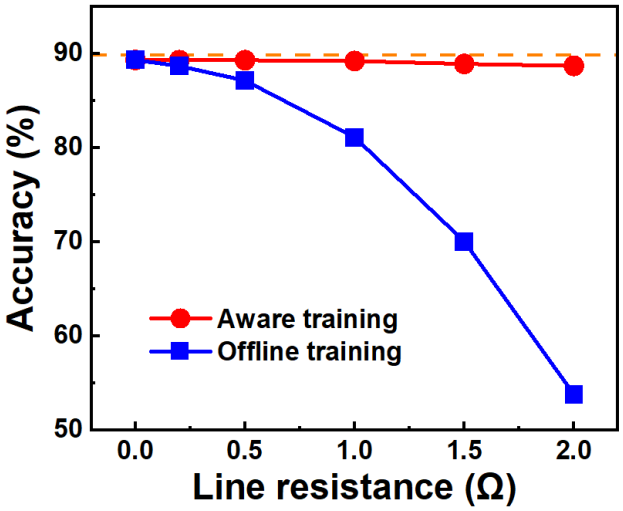


Fig. 5.14. The impact of the crossbar array line resistance. The software accuracy is shown as dash line for reference.

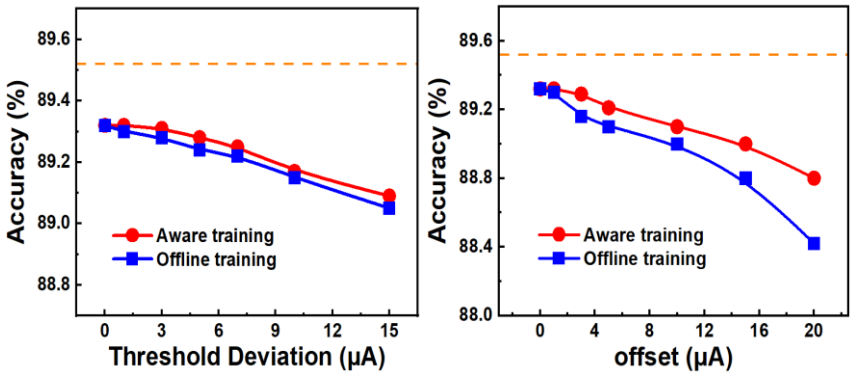
programming failure.

- *Array Non-Idealities*

The line resistance between adjacent cells, as described in Section III.C, will lower the voltage provided to the memristor cell. It is based on the size of the array, the resistance of the lines, and the average conductance. It is dependent on the average conductance, the line resistance and the array size. Referring to the distribution of the mapping weights, Fig. 5.8, the high weight states in the distribution of the mapping weights will be moved toward the center and overlapped depending on the cell location in the array, which will affect accuracy. The effect of the suggested training framework against the line resistance is shown in Fig. 5.14 along with the loss of accuracy against different line resistances. When line resistance is severely increased to 2Ω, the framework can still maintain accuracy compared to the conventional offline training on 64×64 array.

- *Circuits Non-Idealities*

The TIA can be used to implement the ReLU activation function on hardware, as mentioned in section 5.3, which also introduces the TIA non-idealities, such as the non-



(a) (b)

Fig 5.15. This diagram compares the results of the non-idealities aware training and the conventional offline training in Edge Poolformer, with (a) an increase in the output offsets of the ReLU circuits and (b) an increase in the deviation of the input threshold in the ReLU circuit. The non-linear component represents 1% of the total output.

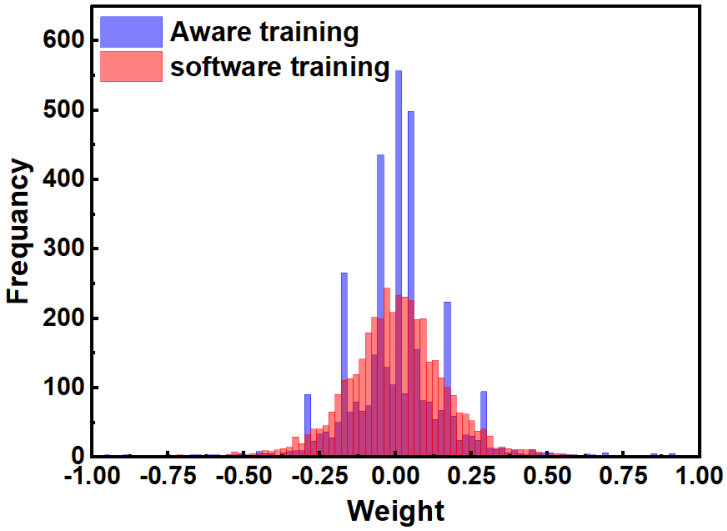


Fig. 5.16. A comparison of the weight distribution between the ideal software weights trained through normal training and the trained weights in the inference system after undergoing non-idealities aware training. Weights are extracted from the MLP classifier.

linearity, the threshold deviation, and the output offset. Fig. 5.15 illustrates the robustness of the aware training against these non-idealities in comparison to the conventional offline training. The results reveal that Edge PoolFormer framework is not sensitive to the impact of designed ReLU circuit, which relaxes the standard requirements for the ReLU hardware implementation.

5.4.2 Interaction Among Non-Idealities

All of these non-idealities interact in the real crossbar, which magnifies the accuracy

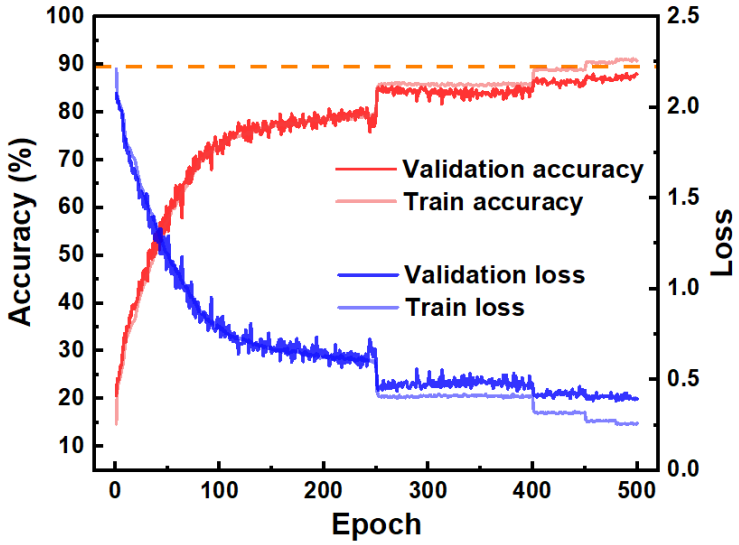
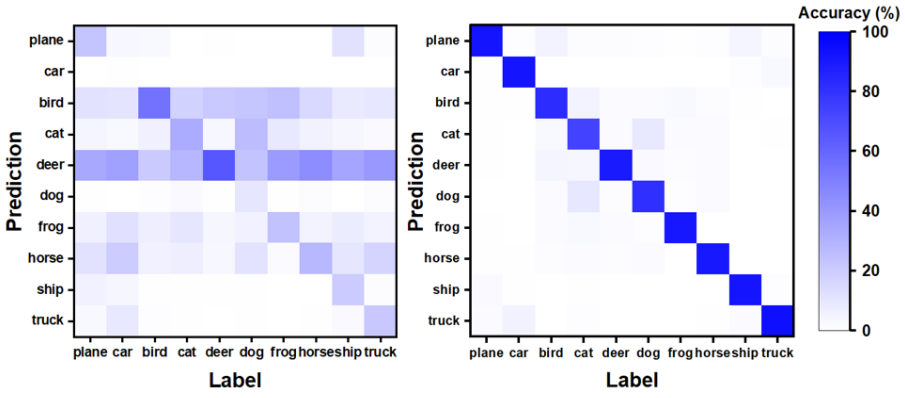


Fig. 5.17. Validation inference accuracy with aware training under incorporated crossbar array nonidealities.

loss caused by each one individually. The impact of all non-idealities on Edge PoolFormer framework is examined simultaneously in the inference system. The training is carried out using the parameters from Table. 5.III, and the trained weights used in the inference system have been loaded onto the hardware. The weight distribution following the training with non-idealities is shown in Fig. 5.16. The distribution clearly aggregates to four separate levels with a noticeable overlap due to device variance when compared to the analog weight by software training. The training loss and accuracy of the proposed Edge PoolFormer-E16 framework on CIFAR-10 is given in Fig. 5.17, where the validation accuracy has more oscillation and is close to the software accuracy after 500 epochs. The contrast between the framework with aware training and the conventional offline training is shown in Fig. 5.18. The inference system with typically trained weights has lost the identification ability because of all these non-idealities and will mistaken most figures for deers and birds. On the other hand, the inference with non-idealities aware training can classify most labels correctly, with an 88.07% overall accuracy for 64×64 array, 0.5 Ω line resistance and 10% device variation,



(a) (b)

Fig 5.18. The confusion matrix of the classification correlation for the Edge PoolFormer inference system is shown in the form of a heat map. (a) The result from normal offline training and (b) The outcome of training that takes into account non-idealities. The recognition accuracy of the inference system with normal training is approximately 28.21%, while with training that accounts for non-idealities, it is 88.07%.

Table. 5.IV. RRAM-based frameworks on CIFAR-10 comparisons

	[89]	[104]	[90]	[91]	[110]	[111]	This work
Model	VGG-11	ResNet-18	VGG	VGG-8	ResNet-20	ResNet-20	**PoolFormer**
Device	HfO2	NR	HfO2/Al2O3	TaOx/HfOx	NR	NR	**TaOx**
# of Conductance Levels	NR	256	16	128	32	10	**4**
# cells per weight	2	2	2	2	2	2	**2**
Ron(Ω)	11k	3k	50k	100k	3k	25k	**5k**
On/Off Ratio	1.9	1000	16	10	1000	12	**6**
D2D Variation (σ/μ)	NR	20%	4%	3.70%	0.9%	NR	**10%**
# of parameters	7.66M	NR	6.6M	13M	0.27M	0.27M	**0.24M**
Line resistance (Ω)	NR	NR	0.1	NR	NR	NR	**0.5**
Crossbar Array	256x256	64x64	1152x128	128x128	32x32	NR	**64x64**
Failure	10%	NR	NR	NR	NR	NR	**1%**
Nonidealities Aware	Yes	Yes	Yes	Yes	Yes	Yes	**Yes**
Software Accuracy (%)	85.2	87	85.17	NR	88.27	84.94	**89.52**
Inference Accuracy (%)	83	86.3	79.4	81	85.58	84.11	**88.07**

*NR：Not Reported

around 1.5% loss compared to the software.

The results also reveal that the Edge PoolFormer framework implemented on memristor-based PIM system is a potential outstanding software and hardware co-design solution for edge AI applications on small dataset. In Table. 5.IV, the performance of the suggested framework is outlined and contrasted with that of other previously published DNNs and Transformer that was implemented with a memristor crossbar. The results of the image classification task using the CIFAR-10 dataset are evaluated for a fair comparison. The highlighted framework showcases a notable

achievement by leveraging a considerably smaller model and utilizing only a 4-level device, ultimately leading to enhanced inference accuracy. Notably, PoolFormer not only rivals the software and inference accuracies of established architectures like VGG and ResNet but also excels in adapting to diverse hardware environments. Its efficiency in utilizing conductance levels and parameters strikes a delicate balance between model intricacy and effectiveness, while demonstrating a keen awareness of hardware-related non-idealities. The documented minimal failure rate and the successful implementation framework underscore Edge PoolFormer practical viability, solidifying its position as a resilient and versatile model tailored specifically for the intersection of software efficacy and hardware-conscious design, particularly in the realm of resource constrained edge-AI applications.

5.5 Summary

In this chapter, a new light hardware-friendly Transformer like neural network structure, Edge PoolFormer, is presented. A modeling and training framework is introduced for edge device implemented on memristive crossbar is proposed. The structure and performance of the neural network are analyzed. Integrating the nonideal factors into a unified training process allows for comprehensive evaluation of framework performance. In addition, compared with conventional DNNs, this structure can realize satisfactory accuracy with fewer parameters, and the training framework can effectively alleviate the effects of these non-idealities to minimize degradation of the inference accuracy. Designed using a combination of Python and PyTorch, the suggested framework is tested using a 16-block network built on a measured 64x64 RRAM array with a 4-level weight resolution. 88.07% inference accuracy is achieved

on CIFAR-10 image classification tasks with less than 1.5% accuracy degradation compared to the ideal model.

While this chapter primarily focuses on the training performance of the proposed Edge PoolFormer model against non-ideal device properties, it is also essential to analyze the inference performance against long-term reliability problems, such as the drift of the conductance over time. Although the current research is limited to modeling and providing guidance for applications, future work will extend to the analysis of inference performance considering the long-term reliability of the memory devices under fabrication. This future research will ensure the robustness and practical viability of the Edge PoolFormer model in real-world applications.

Chapter 6. Computing-in-memory (CIM) Systems for Edge AI Applications

In Chapters 4 and 5, a modeling and simulation framework for edge-AI DNN implemented on memristive crossbar is presented. Thereafter, various edge system applications are explored in this chapter.

6.1 ECG Classification

In recent years, convolutional neural network (CNN) has achieved impressive results in various classification tasks such as image recognition [1,132-134] and language processing [135]. CNN has also shown great potential in biomedical signal processing, particularly in the analysis of electrocardiogram (ECG) signal for cardiovascular diseases (CVD) [136]. Wearable ECG monitoring devices offer critical information for the early detection of heart arrhythmias and heart failure [137]. So that early clinical intervention can be triggered. There has been an increasing interest in integrating artificial intelligence function into ECG devices to achieve intelligence at the edge. However, implementing CNN on resource constraint wearable devices poses great design challenges, due to the limited memory size and computation capacity in the wearable device [138]. Binary neural network (BNN), in which weights are quantized to only binary values [139] is considered as a promising alternative to CNN. BNN offers significant reduction in memory size while maintaining comparable classification accuracy [140]. The objective is to strike a balance between model complexity, inference accuracy, computing resources, and power consumption [141-142].

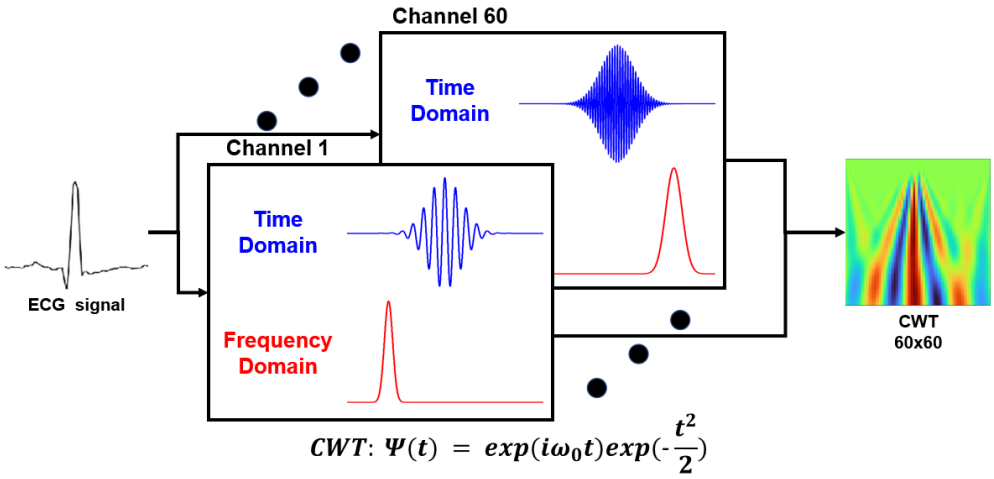


Fig. 6.1. BPF-based 60-channel CWT-approximation signal processing of ECG in this work.

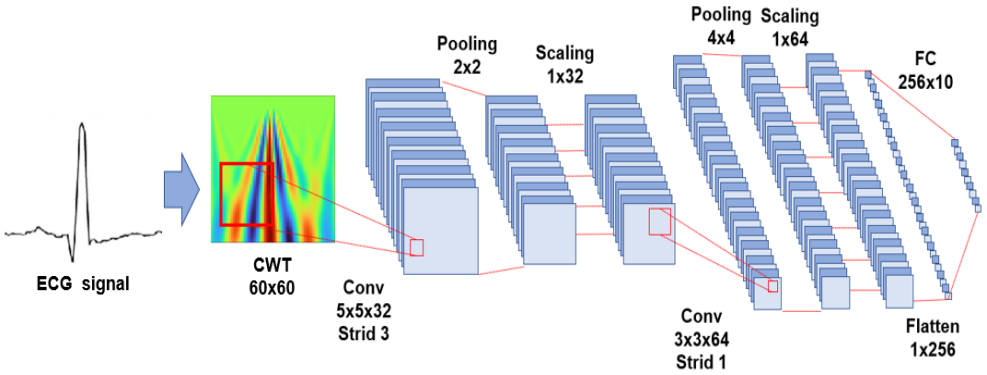


Fig. 6.2. The structure of the binary convolutional neural network used in this work for training and test.

6.1.1 Binary Neural Network (BNN)

The original one-dimensional (1D) ECG signal is firstly converted to 2D time-frequency feature by applying continuous wavelet transform (CWT). Compared with short-time Fourier transform (STFT), CWT provides better time-frequency resolution, and it can capture non-stationary characteristics of ECG signals. The employed CWT has scales from 1 to 60 and Morlet wavelet is used as mother wavelet with sampling frequency of 360Hz. The Transform is implemented with band pass filter (BPF) as shown in Fig.6.1. BPF with a high Q-factor and high order can approximate the high frequency resolution of CWT. Additionally, the time-domain resolution of CWT can be accomplished via additional time window steps. This BPF-based approximation

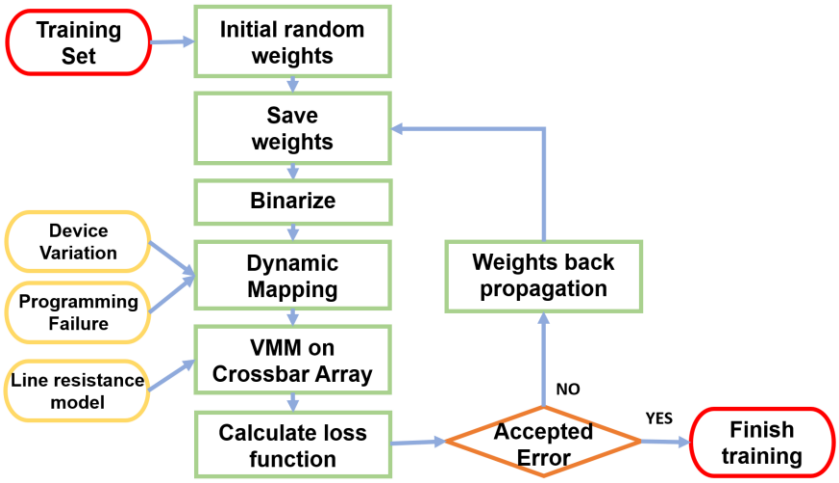


Fig. 6.3. Flow chart of binary nonidealities aware training.

technique offers an advantage over traditional CWT signal processing [154] by facilitating more convenient adjustment of time-frequency characteristics.

Fig. 6.2 illustrates the structure of the binary CNN network. The network comprises 2 convolutional layers, 2 pooling layers, 2 channel scaling layers, and 1 fully connected (FC) layer, totaling 21.5k parameters. In contrast to conventional CNN that employ batch normalization, the output of batch normalization in this network includes a scale and shift operation for each channel. Since the bias of the convolution layer in each block already serves as the shift function, an additional simple scaling operation is introduced to each channel. Through software training with floating point weights, a test sensitivity of 99.6% is achieved.

The convolutional layer is responsible for feature extraction, which is a critical step for binary CNN. Referring to the previous work that applied the convolution operation on multi-level RRAM-based crossbar [153], the binary convolution operation is designed based on the process reported in [108]. Both the binary kernels and the feature maps from the input can be unfolded. The kernel is reshaped into a column on the crossbar array. The reshaped kernels are appended column by column to form a multiple-kernel matrix. Similarly, the feature maps of next layer are created after the

folding.

The conductance of RRAM follows the normal distribution because memristors typically have a conductance variation [155]. Thus, the random normal distribution mask is applied to HRS and LRS level in the model as the variation. The impact of line resistance to crossbar-based system been explored in previous work [67]. It is negligible in relatively small size crossbars. However, as the array size grows, the line resistance effects become more severe, resulting in significant degradation. In this work, a highly efficient line resistance impact estimation model [90] is applied in the training and inference process, which compensates the error accumulation and voltage headroom when turning on the array cells simultaneously. With the line resistance, the voltage across the cell is reduced, the effective conductance also decreases. As a result, the currents collected from the bit lines are smaller than expected and can further reduce NN inference accuracy. The programming failure is simulated by randomly turning off the RRAM cells in crossbar simulation. Considering the above nonidealities, the nonidealities aware training shown in Fig. 6.3 is conducted during the system training.

6.1.2 System evaluation

The binary CNN model shown in Fig. 6.2 is implemented in Python. The nonidealities parameters are based on the measured TaOx RRAM 64×64 crossbar array model [73]. Additionally, due to the extreme imbalance in the distribution of the dataset samples, a random sampling approach is employed during training for the more frequent classes, while the less frequent classes are augmented to increase their sample size, thus ensuring equal representation of each class per epoch. For the same reason, sensitivity is used to evaluate the system instead of accuracy as the focus should be on identifying the minority of abnormal ECG signals rather than the majority of normal signals.

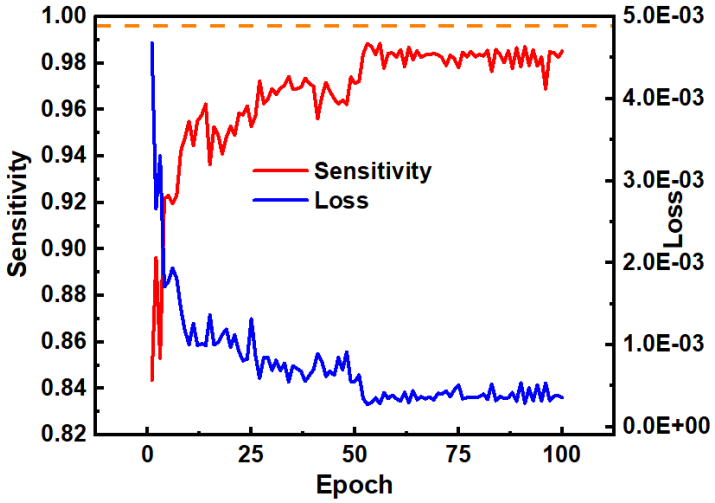


Fig. 6.4. Test inference sensitivity with nonidealities incorporated training under crossbar array nonidealities.

Sensitivity is calculated as the ratio of true positives (TP) to the sum of true positives and false negatives (FN) as below:

$$sensitivity = \frac{TP}{TP+FN} \tag{6.1}$$

By taking the proposed readout compensation and nonidealities into the aware training with Pytorch, the inference sensitivity of the model can be trained to 98.89% as shown in Fig. 6.4, where the sensitivity of the software model is given as the reference line. The result has only 0.7% degradation as compared to the software model. It can be observed that the proposed mapping and training offer the most comprehensive analysis. With only 1T1R cell and 21.5k parameters, a higher inference sensitivity is achieved.

6.2 Speech Recognition

The ferroelectricity is recently found in the aluminum scandium nitride ($Al_xSc_{1-x}N$) thin film in 2019 [123]. Unlike HfO_2, the distribution of the coercive field of $Al_xSc_{1-x}N$ is quite tight [123], making it possible to construct the selector-free array without unexpected state changing. Derived from the experiments on a CMOS (BEOL)

compatible FeRAM array, [124], the edge-AI application framework implemented on $Al_{0.7}Sc_{0.3}N$-based capacitive crossbar array are explored in this work. Readout circuit for FeRAM-based synapse is designed and a high efficiency ternary weight mapping method is proposed. To evaluate the performance of the framework, a speech recognition task is performed with ResNet [110] on the proposed framework.

While we predominantly use the term "FeRAM" to describe the ferroelectric capacitors in the crossbar array that employs tunable small-signal capacitances, it is important to note that the term FeRAM is typically used to refer to the 1T1C cell utilizing the large-signal polarization effect. As pioneered by the Georgia Tech team, the small-signal capacitive sensing concept is referred to as “nvCap” (non-volatile capacitor) [173]. However, the concept of using small-signal capacitances is similar to the work described by C. Liu et al. on Al0.7Sc0.3N-based FeRAM. In this research, the steep switching, leakage, and selector-free array characteristics of FeRAM are explored within the context of compute-in-memory applications. For consistency with the terminology used in this foundational work, we occasionally refer to these capacitors as FeRAM in this thesis, recognizing the similarities in the underlying concepts.

6.2.1 Capacitive FeRAM Crossbar

The structure of a general capacitive crossbar array is shown in Fig. 6.5(a). The synaptic devices are assigned at every cross point between word line (WL) and bit line (BL). The voltage collected at each BL is the sum of multiplication of WL input voltage and capacitance of synaptic device. Thanks to the tight coercive field distribution of $Al_{0.7}Sc_{0.3}N$-based synapse, it is possible to build an ultra-high density selector-free FeRAM array. Furthermore, input voltage can be represented in analog way and

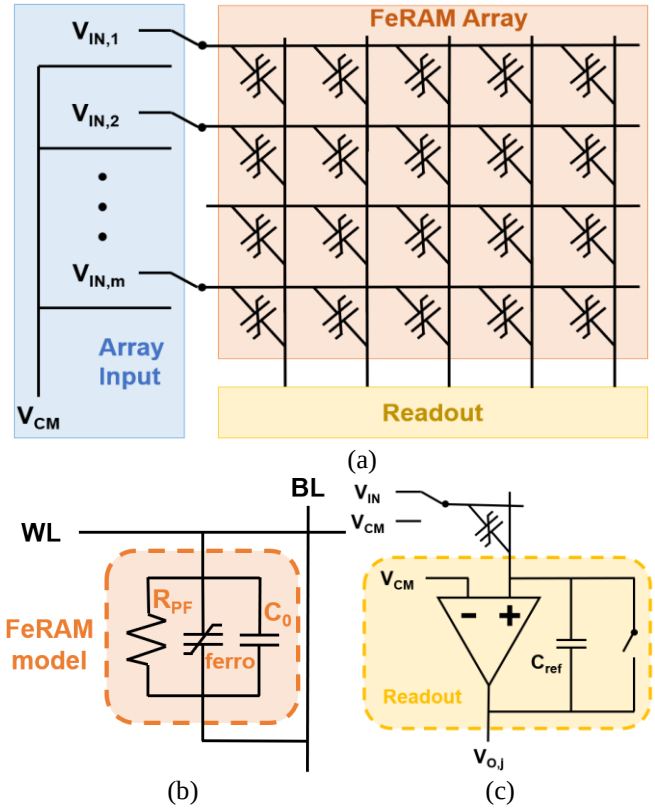


Fig. 6.5. (a) The structure of general FeRAM capacitive crossbar array with word line connected to array input and bit line connected to readout circuit. (b) The circuit model of an $Al_{0.7}Sc_{0.3}N$-based FeRAM cell. (c) The structure of the proposed readout circuit.

implemented in only one cycle without concerns about the state change of FeRAM.

The readout process of capacitive crossbar includes two phases, charging phase and charge transfer phase [118]. However, conventional readout circuit in two phases is not appropriate for FeRAM -based synapse due to leakage current and the circuit model of the synapse cell is shown in Fig. 6.5(b). Therefore, a novel readout circuit shown in Fig. 6.5(c) is designed in this work.

During the charging phase, output switch is closed to avoid the influence of leakage current on reference capacitance, C_{ref}, BLs are connected to common-mode voltage, V_{CM}, and input vector is transferred to voltage, V_{IN}, applied to WLs. The synapse is then charged based on the input voltage and the FeRAM state. During the charge transfer phase, output switch is open, BLs and WLs are connected to V_{CM}. Leakage current is

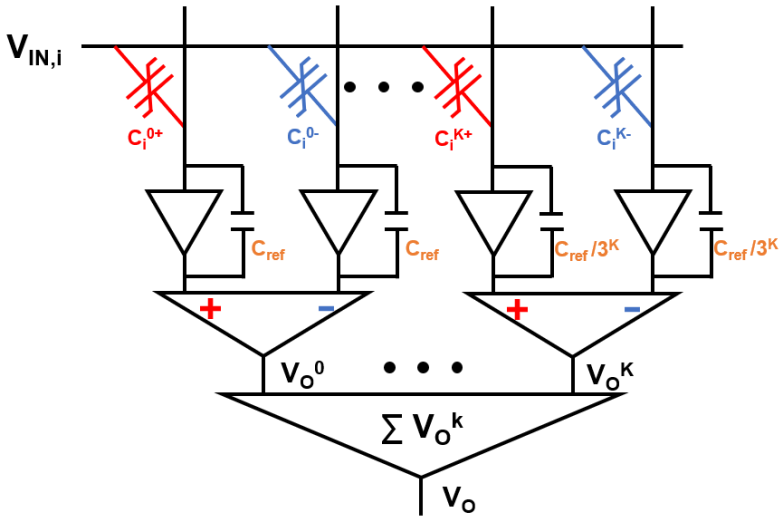


Fig. 6.6. The diagram of one weight representation with the proposed ternary mapping. Two column bit lines represent one digit, one for positive and one for negative. Each digit is different from reference capacitance and summed.

neglected because of zero potential difference. Thereafter, the charges on the synapses is transferred to C_{ref} at the end of each BL. The output voltage of the j^{th} column, $V_{O,j}$, is represented by following equation:

$$V_{O,j} = \frac{\sum V_{IN,i} \times C_{i,j}}{C_{ref}} \tag{6.2}$$

where $V_{IN,i}$ is the input voltage of i^{th} row and $C_{i,j}$ is the conductance of the FeRAM at i^{th} row and j^{th} column.

To represent the negative weight in FeRAM, differential pair is employed here [18]. Two adjacent FeRAM cells are used to represent one weight, one cell for positive and another for negative. The interim output voltage collected from BLs are subtracted to get the differential output as:

$$V_O = V_O{}^+ - V_O{}^- = \frac{\sum V_{IN,i} \times (C_i{}^+ - C_i{}^-)}{C_{ref}} \tag{6.3}$$

where $C_i{}^+$ and $C_i{}^-$ are the conductance of differential pair FeRAM at i^{th} row.

Because of the small on/off ratio of FeRAM, binary synapse is chosen for large array considering the precision of weight and conventional binary mapping method

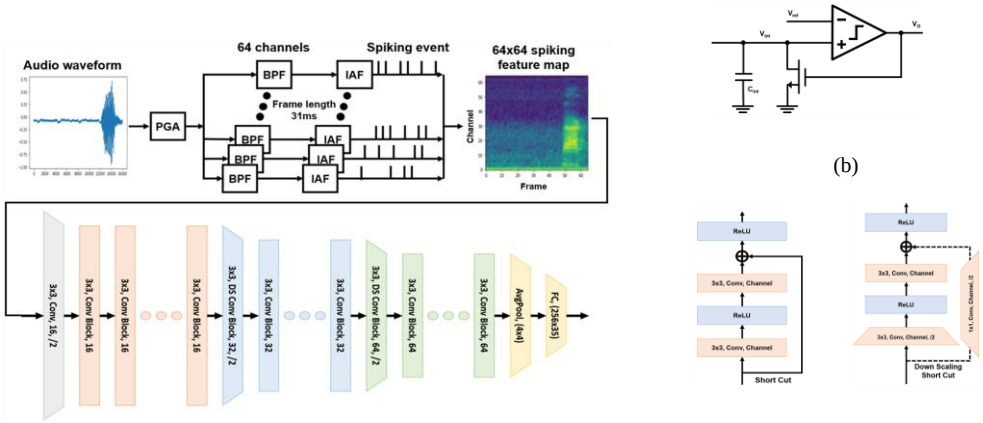


Fig. 6.7. (a) The diagram of the speech recognition system with feature extraction module and ResNet implemented by FeRAM crossbar array. (b) The circuit of integrate-and-fire (IAF) module. (c) Architecture of convolution block (Conv Block), consisting of convolution, ReLU and short cut operations. (d) Architecture of down scaling convolution block (DS Conv Block), consisting of convolution, ReLU and down scaling short cut operations.

implements multi-bit weight by multiple columns and shift&add. To improve the efficiency of the weight mapping, a ternary mapping method is

proposed here. The example of K ternary digits is given in Fig. 6.6. To map k^{th} ternary digit, $k = 0, 1, \dots, K$, the reference capacitance is enlarged 3^k times to decrease the weight of digit. The output voltage can be described as following equation:

$$V_O = \sum {V_O}^k = \sum_k \frac{\sum_i V_{IN,i} \times ({C_i}^{k+} - {C_i}^{k-})}{3^k \times C_{ref}} \tag{6.4}$$

where ${V_O}^k$ is the k^{th} ternary digit of output.

6.2.2 Keyword Spotting System and Results

To demonstrate the feasibility of the proposed FeRAM -based capacitive crossbar array and ternary mapping method on edge-AI applications, a speech recognition system of spiking feature extraction and ResNet model is built for Google Speech Commands dataset [125]. As shown in Fig. 6.7(a), the system transfers the waves into feature maps and applied the ResNet model implemented by capacitive crossbar array.

In the design of a 64-channel analog spiking feature extraction (SFE) system, shown in Fig. 6.7(a), the system exhibits strong performance in accurately detecting and

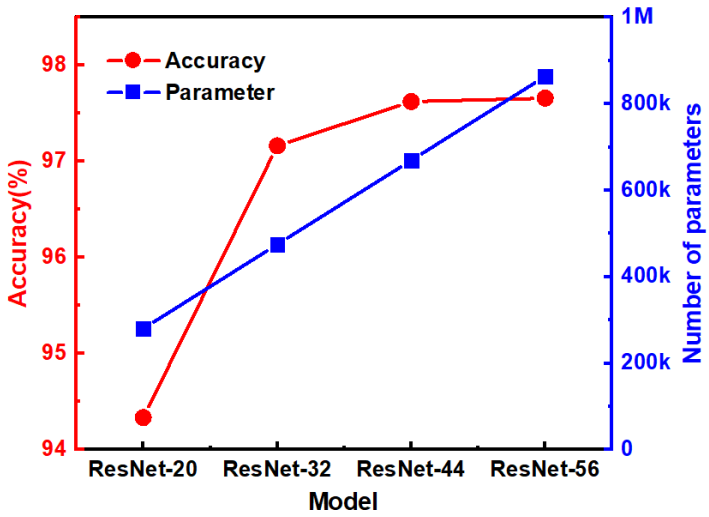
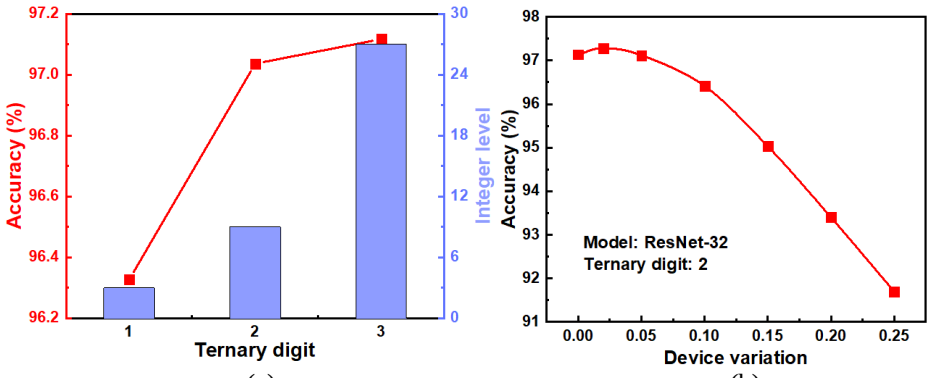


Fig. 6.8. The performance comparison of different ResNet models on speech recognition task.

responding to variations in the amplitude, frequency, and noise characteristics of speech signals. This ensures a high degree of robustness in the system's ability to extract relevant features from the input signal.

The first block in the analog SFE is a programmable gain amplifier (PGA) [126] that can adjust its gain from -12dB to 12dB in order to accommodate various microphone signal intensities. The PGA employs a two-stage, fully differential, Miller-compensated operational amplifier with an active-RC configuration. This amplifier exhibits an open-loop DC gain of 58 dB and has a unity gain bandwidth of 100 kHz. The speech signal is subsequently input into a 64-channel bandpass filter (BPF) [127] with a Q-factor that can be varied from 1 to 10 and a frequency range from 20 to 4000 Hz. Finally, the features generated by the BPF are transformed into asynchronous events by the integrate-and-fire (IAF) module, and then transmitted to the digital ResNet model for further processing.

ResNet is a widely used deep convolutional neural network model in AI applications. VMM operation is the foundation of the key components of ResNet, such as identical dimension convolutional layer, down scaling short cut and fully-connected classifier.



(a) (b)

Fig. 6.9. (a) The effect of ternary digits on ResNet-32 model with corresponding integer levels. (b) The impact of device variation (σ/μ) on inference accuracy.

Consequently, FeRAM-based capacitive crossbar array is a potential candidate for low power implementation.

The mapping of convolutional layer and fully-connected layer on resistive crossbar array have been discussed in previous works [111] and is applicable for capacitive crossbar in this work. Besides, Down scaling short cut in this work is a modified 2-D convolutional operation with 1 kernel size and 2 stride size. Thus, complete implementation of ResNet on capacitive crossbar array is feasible and simulated in this work.

The synapse model is built with the experiments on multiscale modeling of $Al_{0.7}Sc_{0.3}N$-based FeRAM [123]. The individual device area is $10^{-6}cm^2$ with measured C_0 around $0.15nF/cm^2$ and 64×64 array size is selected. The system is trained 200 epochs on a 16-core 3.2GHz AMD Ryzen 5800H CPU with 32GB RAM with the non-idealities-aware training [111].

To find the appropriate model for the system, three models with different sizes are evaluated in software, ResNet-20, ResNet-32, ResNet-44 and ResNet-56, with 0.27M, 0.45M, 0.64M and 0.82M parameters, respectively. The performance of different models on speech recognition task is given in Fig. 6.8 where ResNet-32 shows high

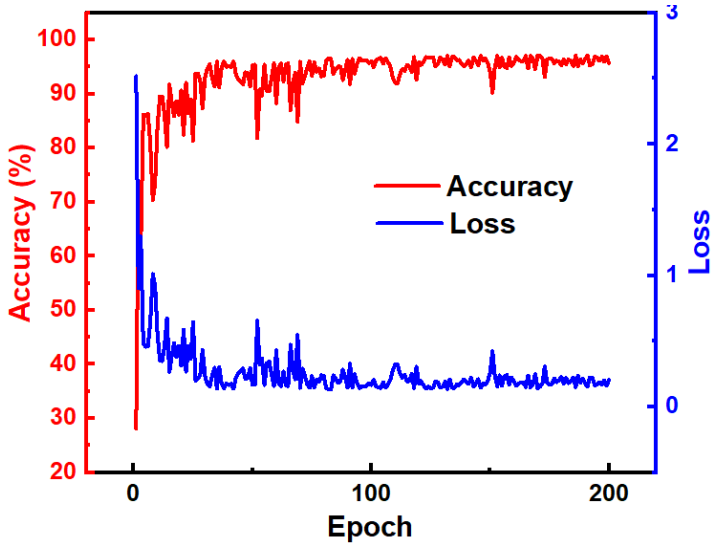


Fig. 6.10. The performance of the final $Al_{0.7}Sc_{0.3}N$ -based FeRAM crossbar array system on the Google Speech Commands dataset 35-command task

recognition accuracy with small model size. Therefore, ResNet-32 is chosen for the system implementation in following evaluation.

Thereafter, the proposed ternary mapping is applied to quantize the ResNet-32 model. Fig. 6.9(a) illustrates how the ternary resolution affects training accuracy. The results show high efficiency of ternary mapping. The training accuracy achieves over 97% with only 2 ternary digits, 9 quantized integer levels.

Although FeRAM device is relatively more stable during the reading procedure, device-to-device variation still affects the precision of the synapse. Fig. 6.9(b) shows the impact of device variation (σ/μ) to inference accuracy. It is worth noting that with the increase of the variation, the system shows its robustness and still achieves >97% accuracy with up to 5% device variation. Besides, because of the small random unpredicted variation during the training, the model reduces the overfitting effect, and the system even performs better with 2% device variation.

The performance of the final system built with Python on the Google Speech Commands dataset 35-command task is shown in Fig. 6.10. The proposed FeRAM

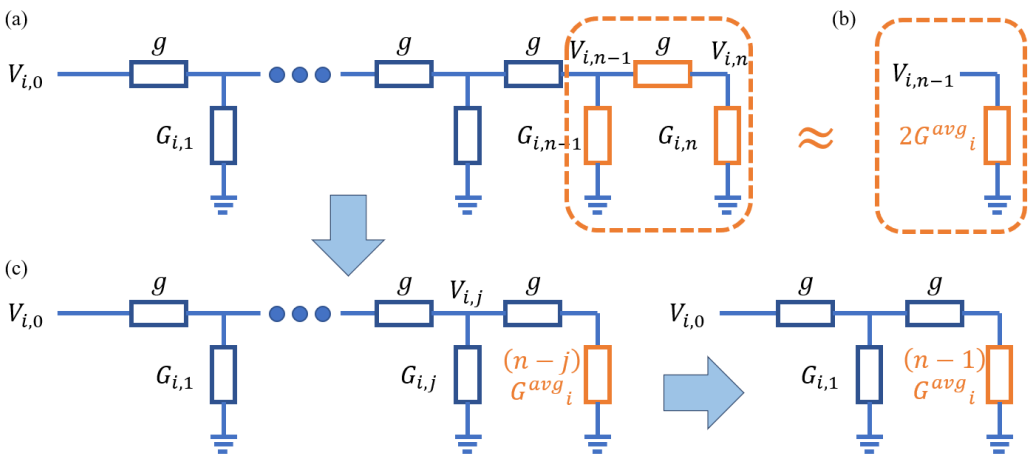


Fig. 6.12. word line charging model.

crossbar with ternary mapping achieved outstanding 97.12% inference accuracy with only 2 ternary digits and 5% high device variation. The overall network size is reduced significantly to only 0.45M parameters. Considering the power and area efficiency of FeRAM crossbar array, the proposed system is a promising candidate for edge-AI applications.

6.3 Imaging Enhancement

6.3.1 FeRAM Array Charging Model

Due to the line conductance and parasitic conductance in the FeRAM, the input voltage cannot be passed to capacitive FeRAM completely. As a result, the performance of the well-trained software model will become extremely poor when implemented on hardware. Thus, hardware aware training is indispensable [111]. To evaluate the impact of the array during the training, a novel conductance FeRAM crossbar model is proposed here to achieve fast and accurate model as shown in Fig. 6.11.

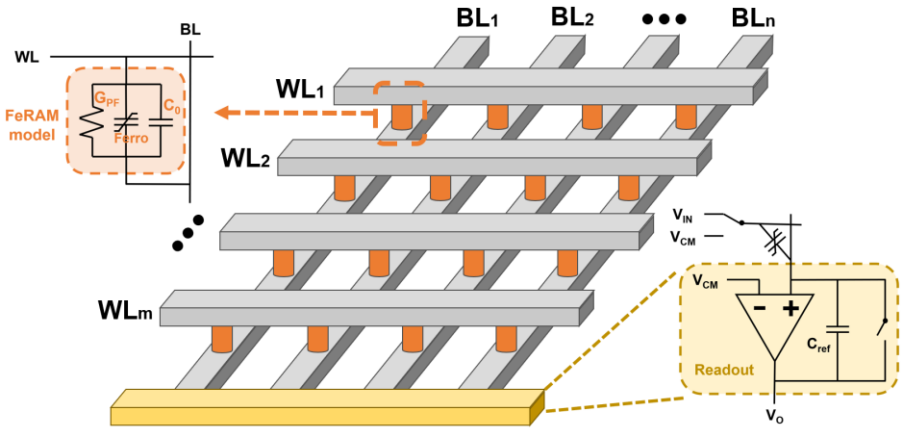


Fig. 6.11. the structure of a capacitive crossbar array

At the end of charging process, the FeRAM synapses are fully charged, and the array becomes a conductance crossbar array system [65]. The system is divided into word line model and bit line model separately. Word line model has positive input with bit line grounded and the bit line model has negative input with word line grounded. A single i^{th} word line model is shown in Fig. 6.12(a) where only word line conductance and FeRAM synapse conductance are considered. For the last two synapses in the row, the equivalent conductance at $V_{i,n-1}$ can be evaluated as

$$G^{eq}{}_{i,n-1} = G_{i,n-1} + (G_{i,n} // g) \tag{6.5}$$

where $G_{i,n}$ is the conductance of FeRAM at i^{th} row, n^{th} column and g is the line conductance between interconnects. Since the line conductance is much larger than the parasitic conductance, which means,

$$g \gg G_{i,n} \tag{6.6}$$

Thus, the equivalent conductance can be simplified as

$$G^{eq}{}_{i,n-1} \approx G_{i,n-1} + G_{i,n} \approx 2G^{avg}{}_{i} \tag{6.7}$$

as shown in Fig. 6.12(b), where $G^{avg}{}_{i}$ is the average conductance of i^{th} row. Recurse the procedure and the Kirchoff's current law (KCL) equation at j^{th} FeRAM

synapse shown in Fig. 6.12(c) can be obtained by

$$(V_{i,j-1} - V_{i,j})g = V_{i,j}G_{i,j} + V_{i,j}(G^{eq}{}_{i,j+1}//g) \tag{6.8}$$

$$V_{i,j-1}g = V_{i,j}(G_{i,j} + g + ((n-j)G^{avg}{}_i//g)) \tag{6.9}$$

$$V_{i,j} = \frac{V_{i,j-1}g}{G_{i,j} + g + (\frac{(n-j)G^{avg}{}_i g}{(n-j)G^{avg}{}_i + g})} \tag{6.10}$$

Thus, the general term formula is derived, and the first term is provided as

$$V_{i,0} = V_{IN,i} \tag{6.11}$$

where $V_{IN,i}$ is the input of the i^{th} row.

Similarly, the general term formula of the single bit line model can be described as

$$-V_{i,j} = \frac{-V_{i+1,j}g}{G_{i,j} + g + \left(\frac{iG^{avg}{}_j g}{iG^{avg}{}_j + g}\right)} \tag{6.12}$$

where $G^{avg}{}_j$ is the the average conductance of j^{th} column.

Then the elements of word line and bit line scaling matrix, α and β, can be calculated as

$$\alpha_{i,j} = \frac{V_{i,j}}{V_{i,j-1}} = \frac{g}{G_{i,j} + g + (\frac{(n-j)G^{avg}{}_i g}{(n-j)G^{avg}{}_i + g})} \tag{6.13}$$

$$\beta_{i,j} = \frac{-V_{i,j}}{-V_{i+1,j}} = \frac{g}{G_{i,j} + g + (\frac{iG^{avg}{}_j g}{iG^{avg}{}_j + g})} \tag{6.14}$$

Finally, the compensated weight matrix, W_{comp} ,considered the crossbar nonidealities during the training is,

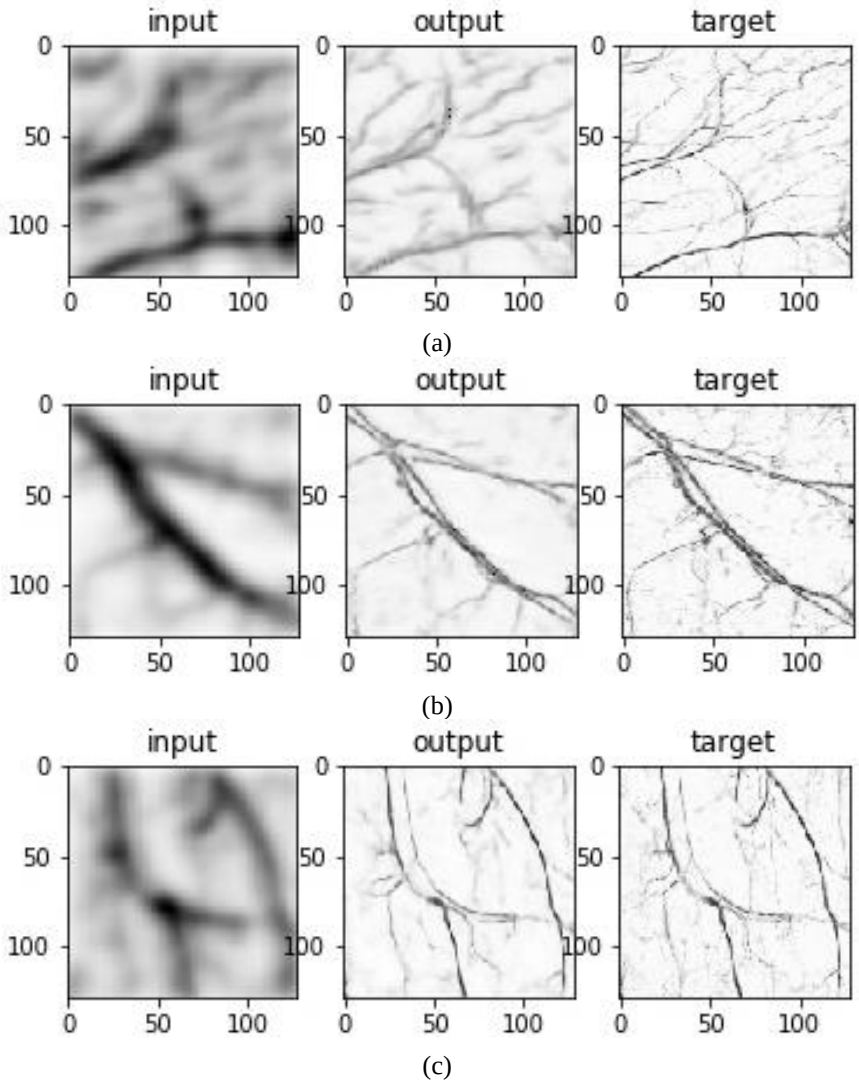


Fig. 6.13. The training results after (a)1, (b)30, (c)100 epochs.

$$W_{comp} = \alpha \odot \beta \odot W \tag{6.15}$$

where W is the weight matrix saved in the network model and $\odot$ is the element-wise multiplication.

6.3.2 System Evaluation

The synapse model is built with the experiments on multiscale modeling of CMOS back end of line (BEOL) compatible $Al_{0.7}Sc_{0.3}N$-based FeRAM [124]. The individual device area is $10^{-6}cm^2$ with measured C_0 around $0.15nF/cm^2$ and 64×64 array size is selected. The system is trained 100 epochs on a 16-core 3.2GHz AMD Ryzen 5800H

Table 6.I

Comparison and summary of our proposed platform with state-of-the-art handheld PAI system

		Y. Zhou *et al.* 2014 [169]	K. Park *et al.* 2017 [170]	J. Aguirre *et al.* 2018 [171]	S. Liu *et al.* 2019 [172]	**Platform in this work**
System parameters	Probe Volume	n.a.	162 g	n.a.	n.a.	**44 g**
	Wavelength	650 nm	532 nm	532 nm	450 nm	**532 nm**
	Scanning mechanism	Raster scanning	Raster scanning	Raster scanning	Without scanning	**Without scanning**
	Ultrasound transducer	Spherically focused UST	UST with acoustic lens	Spherically focused UST	Planar UST with 72 matrix arrays	**Planar UST with 72 matrix arrays**
Imaging criteria	Penetration depth	4 mm	800 μm	5 mm	⩾ 5 mm	**≥ 1 cm**
	Imaging resolution	230 μm	n.a.	30 μm	740 μm	**78 μm**
	Imaging speed	10 s	20 s	70 s	3 Hz	**30 Hz**

CPU with 32GB RAM with the non-idealities-aware training [111]. The ground truth data is vasculature images obtained with optical resolution photoacoustic microscopy, which is famous for its high imaging resolution and quality. The corresponding degraded imaging results from handheld PA imager is synthesized and the paired low/high resolution data can be used for training.

The proposed MultiResU-Net system is trained with FeRAM array model on 2 ternary digits. The results during the training process of 1, 30, 100 epochs are shown in Fig. 6.13. It can be observed that during the training process, the noise presented in the original image is removed, while the vasculature structure is more prominent, and the resolution is also significantly enhanced.

6.4 Summary

This chapter introduces three significant advancements in the fields of edge-AI and biomedical imaging. Firstly, it presents an edge-AI system utilizing capacitive ferroelectric random-access memory (FeRAM) crossbar arrays, compatible with CMOS backend-of-line fabrication. A novel capacitive crossbar circuit and ternary mapping technique enhance weight resolution efficiency exponentially. Secondly, an electrocardiogram (ECG) signal classification method using binary CNNs on RRAM crossbar arrays is presented. It includes a unique RRAM crossbar structure for adaptive readout quantization, mitigating crossbar nonidealities. A bandpass filter-based continuous wavelet transform enhances ECG signal preprocessing. Lastly, the work introduces a novel approach to enhance imaging quality in handheld photoacoustic tomography using FeRAM crossbar arrays and an optimized algorithm, promising improved biomedical imaging applications.

Chapter 7. FPGA Implementation for Neural Network and Signal Processing Based on High-Level Synthesis

In this chapter, the high-level synthesis framework implemented on the Xilinx PYZQ platform is explained. A tailored PoolFormer-based model for edge speech recognition and a continuous wavelet transform (CWT) approximation for ECG signal processing on FPGA through PYNQ-Z2 board are proposed.

7.1 PYNQ Framework

The Xilinx PYNQ platform revolutionizes edge computing and neural network implementation by combining FPGA technology with Python programming. The flowchart shown in Fig. 7.1 outlines the process of NN accelerator deployment on the platform. This innovative platform simplifies FPGA development, making it accessible to developers from different background. The flexibility and adaptability of PYNQ

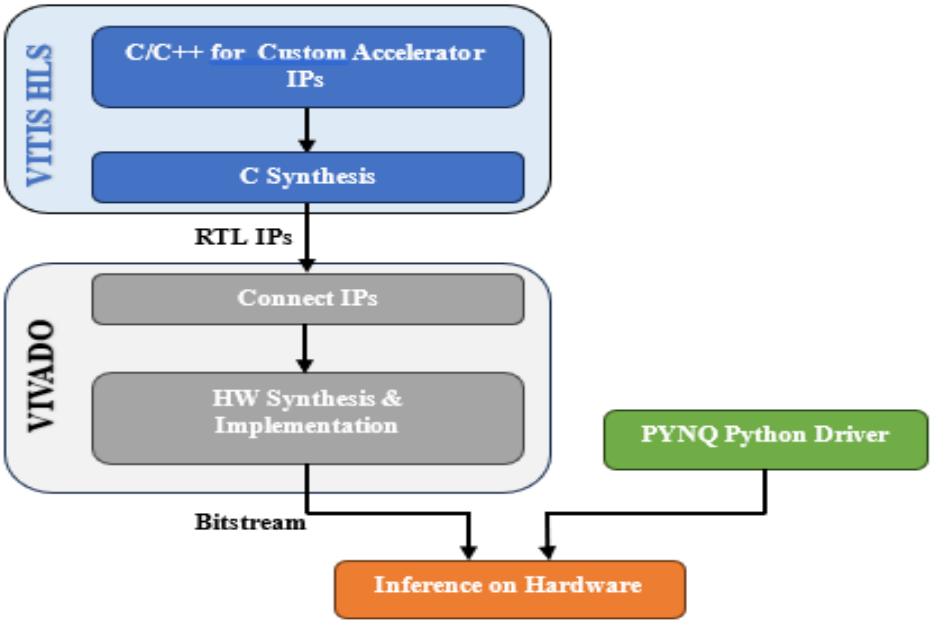


Fig. 7.1. The flowchart of the PYNQ platform

make it ideal for deployment of high performance custom neural network hardware accelerator. Additionally, it streamlines the deployment of custom hardware accelerators for neural networks, largely reducing the deployment time, while offering high performance and high energy efficiency. All these traits make PYNQ a valuable platform as edge AI devices.

In this chapter, we utilized the PYNQ-Z2 platform, a System-on-Chip (SoC) solution featuring both a processing system (PS) and programmable logic (PL) components. This platform enables the deployment of a high performance customized PoolFormer (PF) model. The PS is leveraged for its control logics while the PL is harnessed for neural network hardware acceleration. The architectural configuration of the developed accelerator is illustrated in Fig. 7.2.

Within this setup, PS executes the python driver and is responsible for all the control logics including memory allocation for both weights and activations in the dynamic random-access memory (DRAM). On the other hand, PL consists of three hardware

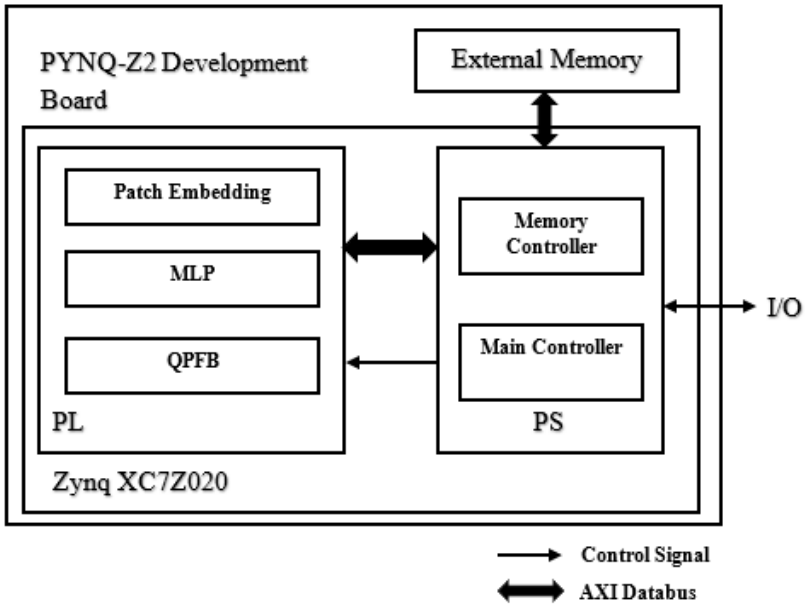


Fig. 7.2. The architectural configuration of the developed accelerator.

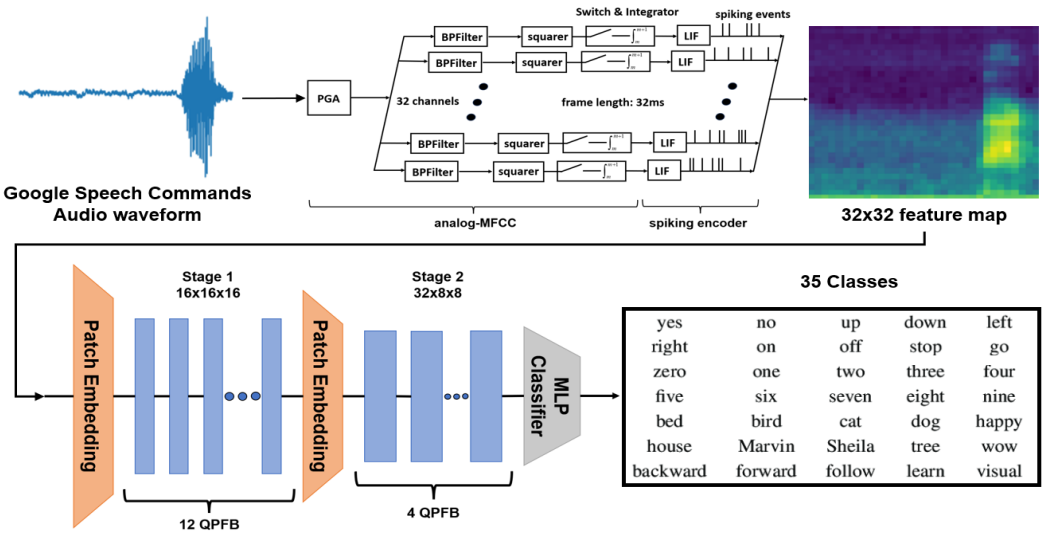


Fig. 7.3. The overall structure of the Edge-AIoT speech recognition framework with a 2-stage PoolFormer structure including 12 and 4 Quantized PoolFormer Blocks (QPFB).

accelerator layer modules for three different neural network layers. Notably, each layer module running on the PL predominantly comprises of multiply and accumulations (MACs) array customized to compute each layer most efficiently. Due to the constrained on-chip memory capacity and the substantial number of parameters in the current model, the parameters associated with the target model and the resulting output feature tensor for each layer are stored in external memory. Consequently, the AXI4 master data movers play a critical role in establishing a connection between the on-chip buffers and the external memory. Furthermore, AXI4 burst mode data transfers are employed to facilitate higher data throughput rates.

7.2 Edge Speech Recognition

The schematic representation of the proposed edge speech recognition system is depicted in Fig. 7.3. Initially, the time-domain signal undergoes a novel spiking feature extraction module, transforming it into a frequency domain feature map. Subsequently,

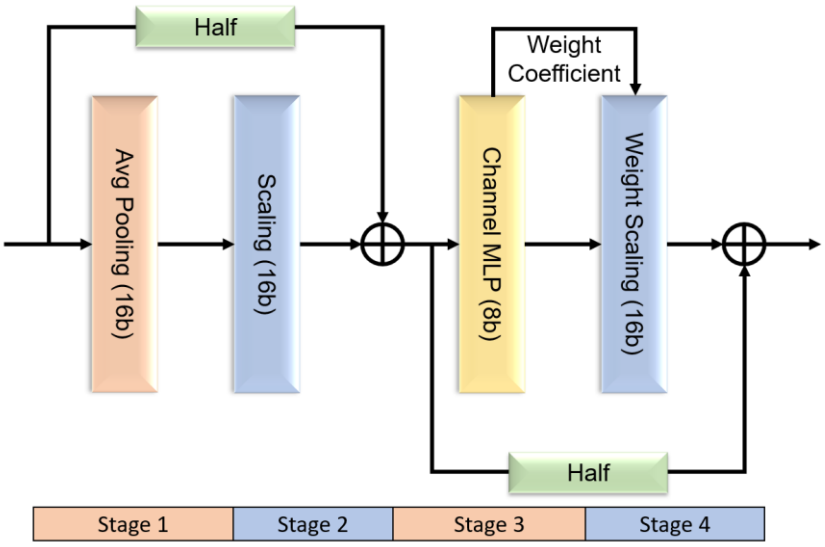


Fig. 7.4. The quantized PoolFormer block structure

this feature map traverses a customized FPGA-implemented PoolFormer network for classification.

7.2.1 Spiking Feature Extraction

Inspired by recent research [36], a novel Spiking Feature Extraction architecture has been employed. This innovative structure seamlessly combines analog-MFCC calculation with a spiking encoder, thus exhibiting a harmonious amalgamation of high accuracy, robustness, and ultra-low power consumption inherent to analog circuits.

To begin with, the speech signal captured by the microphone undergoes amplification via a programmable gain amplifier (PGA) [126]. Subsequently, within the analog circuitry, a high energy-efficient bandpass filter (BPF) is utilized to perform the function of the Fast Fourier Transform (FFT). This choice stems from the fact that the energy spectrum computed in the frequency domain, post-FFT and mel filtering, is approximately equivalent to the energy in the BPF domain after mel scaling in the time domain. Following this, Squarer, Switch, and Integrator circuits transform the signals from each channel into analog energy spectra in every frame. The frame length is set at

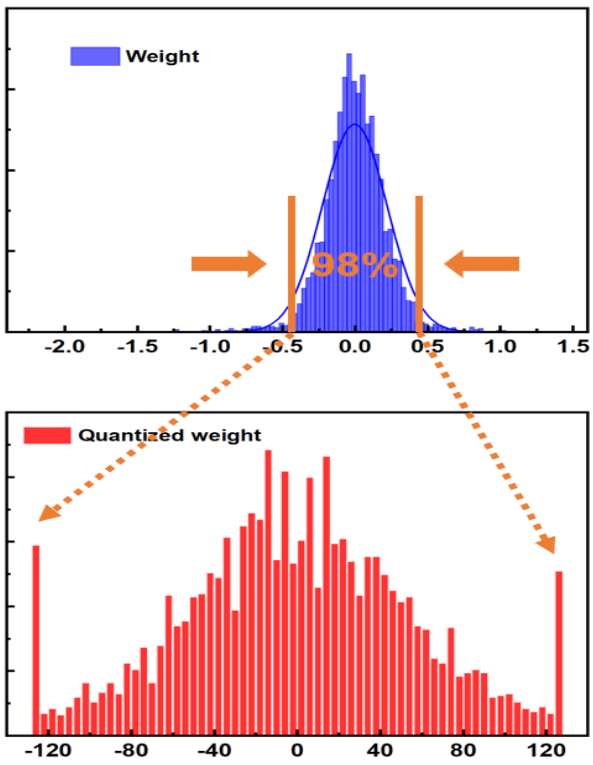


Fig. 7.5. The example of the proposed substep symmetric quantization approach.

32ms and is collectively controlled by the switches of all channels. At this point, 32-dimensional analog approximations of MFCC features are obtained. Ultimately, the Leaky Integrate-and-Fire (LIF) block, functioning as the spiking encoder, facilitates the conversion of analog signals into asynchronous spiking events.

7.2.2 Quantized PoolFormer Block (QPFB)

PoolFormer, which has previously been investigated in the context of edge computing, has demonstrated its compatibility with hardware implementation, primarily owing to its straightforward linear operations and a reduced need for trainable parameters [108]. In this study, we enhance the PoolFormer block structure, as illustrated in Fig. 7.4, to fully leverage the advantages offered by FPGA technology.

Since the pooling operation lacks parameters, the primary weights pertain to the

TABLE. 7.I. Performance Comparison for Speech Recognition on FPGA

	This Work	[166]	[167]	[168]
FPGA/Rpi	Zynq XC7Z020	Zynq Ultrascale+ XCZU19	Intel Cyclone V	Raspberry Pi 3B+
Precision (W/A)	8b/16b	32b/32b	1b/NR	32b/32b
class	35	10	10	10
processing	Spiking Feature Extraction	MFCC	MFCC	MFCC
Network	PoolFormer	CNN	CNN	RNN
Input size	32x32	40x40	1024x16	152x181
No. weight	39k	93k	433k	830k
Accuracy	95.41	95.11	90.3	96.62

channel MLP. To optimize FPGA resource utilization, we employ quantization for the channel MLP weights, specifically utilizing an INT8 format with a substep symmetric quantization approach. The substep symmetric quantization process, depicted in Fig. 7.5, involves forcing the top 2% absolute weights to align with the 98% weight boundary, while the remaining weights are scaled to the nearest levels within a range spanning from -127 to 127. This weight coefficient is then integrated into the scaling layer.

$$weight\ coefficient = \frac{weight\ boundary}{127} \tag{7.1}$$

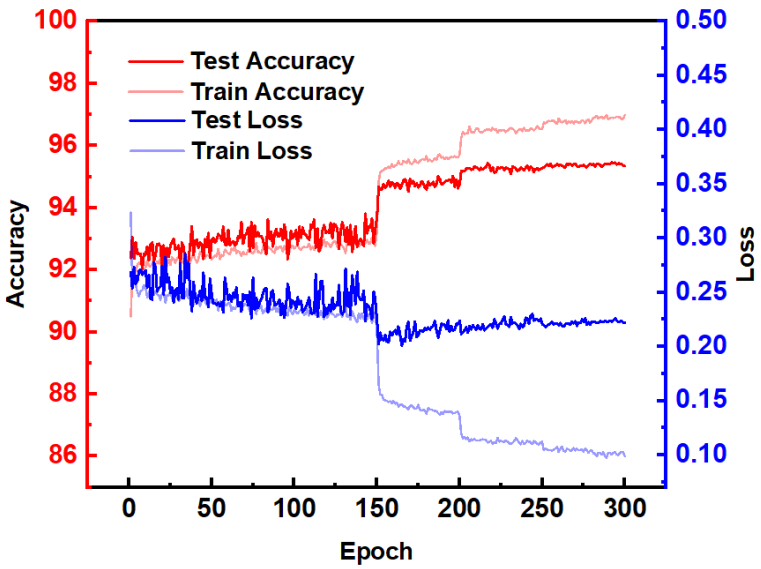
$$weight\ scaling_i = half(scaling_i \times weight\ coefficient) \tag{7.2}$$

where, i is the channel number and $half(\cdot)$ transfer the datatype of output into FP16 to reduce the memory utilization.

Additionally, the channel ratio has been decreased from 4 to 2 to further diminish the number of parameters within the PoolFormer block.

7.2.3 Simulation Results

The 16-layer PoolFormer model, initially well-trained in FP32, underwent a



Resource	Utilization	Available	Utilization%
LUT	35640	53200	66.99
FF	42528	106400	39.97
BRAM	27.5	140	19.64
DSP	156	220	70.91

Fig. 7.6. The resultss of the fine-tuned retraining process and resource utilization on PYNQ-Z2 board.

refinement process in which its weights were fine-tuned and substep symmetrically quantized to INT8, and its features were represented in FP16 format. Subsequently, this optimized model was deployed on the PYNQ-Z2 platform, which incorporates the ZYNQ XC7Z020 System-on-Chip (SoC). Following the preprocessing step involving spiking feature extraction, the input data undergoes classification through the custom PYNQ framework hosted on the board.

The initial training phase for the PoolFormer network is conducted using FP32, achieving an impressive accuracy rate of 95.64%. Subsequently, the model is transitioned to the proposed quantized system. Fig. 7.6 displays the outcomes of the fine-tuned retraining process and resource utilization. After post-implementation, our proposed edge speech recognition system attains an impressive accuracy rate of 95.41%

on the 35-class Google Commands dataset [125], all while maintaining resource utilization levels that are compatible with the PYNQ-Z2 board.

In TABLE. 7.I, we provide a comprehensive summary of results, facilitating a detailed comparison with recent speech recognition solutions. To ensure an equitable evaluation of network performance, we focus specifically on the Google Commands classification task, aligning our analysis with findings from various contemporary works in the field. Notably, our research showcases a remarkable achievement in this context. Despite the relatively modest parameter count of our model, consisting of only 39k parameters, our proposed edge speech recognition system achieves outstanding results for 35-class recognition. This underscores the efficiency and effectiveness of our approach, positioning it as a highly competitive and viable solution within the realm of speech recognition technologies. Such accomplishments hold significant potential to advance the field of edge computing, particularly in scenarios where resource constraints and real-time processing are pivotal considerations.

7.3 CWT Approximation

The ever-growing demand for real-time, efficient solutions in healthcare has sparked a keen interest in edge-based cardiovascular disease (CVD) recognition systems. Leveraging advanced techniques like Continuous Wavelet Transform (CWT) and FPGA technology, coupled with High-Level Synthesis (HLS), offers a promising avenue to develop robust and rapid CVD recognition models. This integration empowers the creation of sophisticated, hardware-accelerated algorithms capable of processing intricate cardiovascular data at the edge, ensuring timely and accurate diagnoses while optimizing resource utilization.

7.3.1 Approximation Derivation

Continuous Wavelet Transform (CWT) is a mathematical tool used to analyze and represent a signal frequency content at different scales and time intervals simultaneously. It characterizes how the signal frequency components change over time, offering a time-frequency representation suitable for analyzing non-stationary signals. The original formula can be represented as

$$CWT(a,b) = \frac{1}{\sqrt{a}}\int_{-\infty}^{\infty} f(t)\cdot\varphi(\frac{t-b}{a})dt \tag{7.3}$$

where, $f(t)$ represents the input time domain signal, a and b are the scale and translation of the transform, respectively and φ is the wavelet function. In this project, Morlet wavelet is chosen as the wavelet function, which is a multiplication of a complex exponential and a Gaussian window as shown below

$$\varphi(t) = e^{i\omega t}e^{-\frac{t^2}{2}} \tag{7.4}$$

In practical applications, the complex exponential function in the $e^{i\omega t}$ Morlet wavelet can often be approximated by the cosine function $\cos(\omega t)$.

$$\varphi'(t) = \cos(\omega t)\cdot e^{-\frac{t^2}{2}} \tag{7.5}$$

This approximation is valid especially when the value of ω is relatively small. For ECG signals, where the dominant frequency components are typically within lower to mid-range frequencies, the cosine function approximation serves as a computationally efficient and accurate representation of the Morlet wavelet. The cosine function approximation simplifies computations while reasonably representing the frequency modulation in the wavelet. Replace the wavelet function in CWT, we have

$$CWT'(a,b) = \frac{1}{\sqrt{a}}\int_{-\infty}^{\infty} f(t)\cdot\cos(\omega\cdot\frac{t-b}{a})\cdot e^{-\frac{(\frac{t-b}{a})^2}{2}}dt \tag{7.6}$$

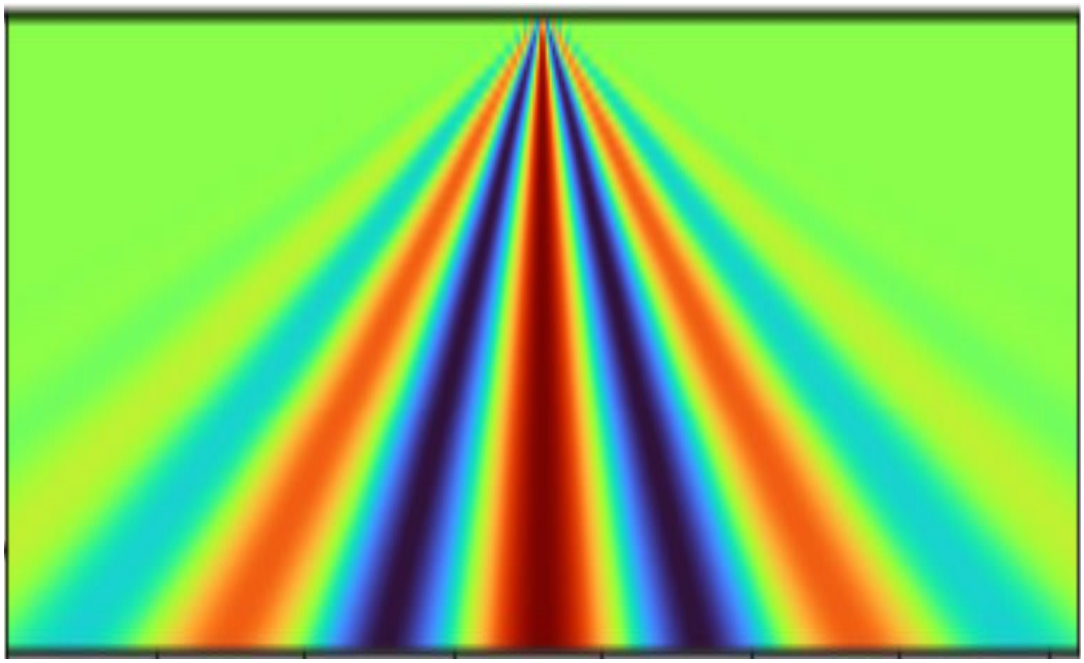


Fig. 7.7. The simplified Morlet wavelet matrix under scales from 1 to 90.

To implement the CWT on FPGA, the above simplified representation is transferred to a discrete representation of the signal where integration is replaced with summation and the continuous parameters are discretized.

$$DWT'(j,k) = \sum_n f[n] \cdot \varphi'[n-2k]$$

$$= \frac{1}{\sqrt{j}} \sum_n f[n] \cdot \cos\left(\omega \cdot \frac{n-2k}{j}\right) \cdot e^{-\frac{\left(\frac{n-2k}{j}\right)^2}{2}} \qquad (7.7)$$

where, $f[n]$ represents the discrete signal at sample n, j and k are the scale and translation of the transform, respectively.

7.3.2 Half Precision FPGA Implementation

To implement the simplified CWT in above section, high-level synthesis is employed here. Following equation (7.5), the simplified CWT can be segmented into two main components: wavelet generation and 1D signal convolution.

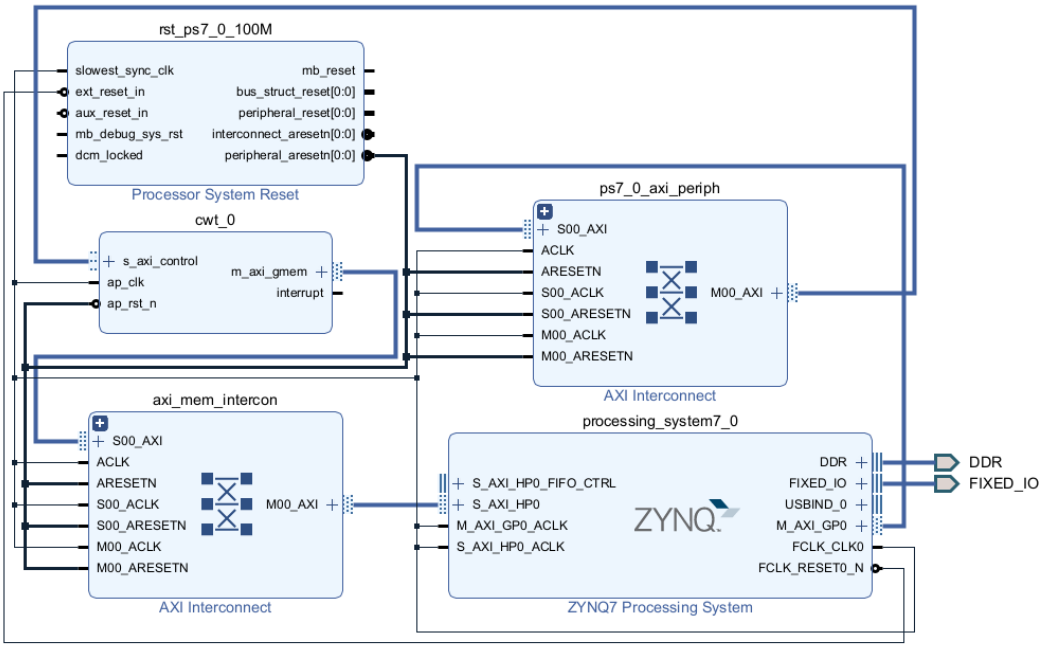
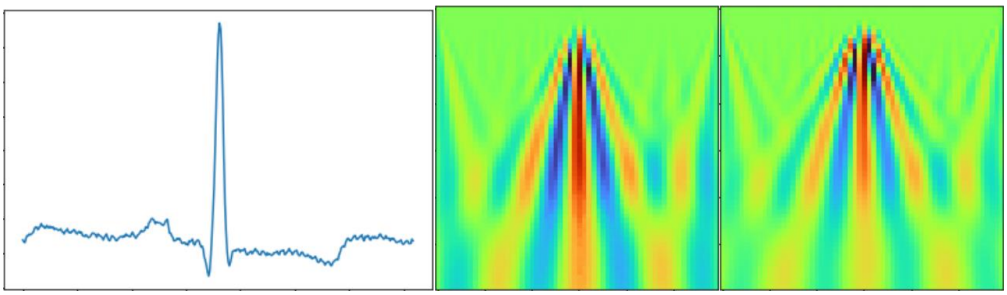


Fig. 7.8. The block design for the simplified half precision CWT implementation.



(a) (b) (c)

Fig. 7.9. An ECG signal preprocessing with (a) original signal wave, (b) software CWT and (c) proposed simplified half precision CWT on FPGA.

For the wavelet generation, once the scale and translation parameters of the Morlet wavelet are configured, the matrix of wavelets across scales ranging from 1 to 90 is computed and depicted in Fig. 7.7.

The conversion of the 1D structure was built using Vitis HLS with all variables constraint to half precision to reduce computation resources. Subsequently, the IP core was exported to Vivado for the block design, as illustrated in Fig. 7.8. An example of

the proposed simplified CWT FPGA implementation is compared with software CWT in Fig.7.9.

7.4 Summary

This chapter has explored leveraging FPGA technology for edge applications in speech recognition and cardiovascular signal processing. The utilization of FPGA technology, coupled with the high-level synthesis (HLS) framework, has been instrumental in crafting customized and high-performance solutions for edge AI devices. Specifically, the proposed tailored PoolFormer-based model for speech recognition has exhibited remarkable accuracy rates of 95.41% on the Google Commands dataset, showcasing its competitiveness against contemporary speech recognition solutions. Moreover, the fine-tuning and quantization of the PoolFormer model, while maintaining resource utilization compatible with the PYNQ-Z2 board, underscore the potential of our approach in addressing real-world edge computing constraints.

Furthermore, our foray into Continuous Wavelet Transform (CWT) approximation for cardiovascular signal analysis has demonstrated the promise of FPGA-driven implementations in healthcare applications. By employing high-level synthesis, we successfully designed a simplified CWT model, showcasing efficient wavelet generation and 1D signal convolution on FPGA. This implementation, optimized for half precision, exemplifies the potential of FPGA technology in processing complex medical data at the edge. The development of hardware-accelerated algorithms for CWT heralds a significant stride towards real-time and accurate cardiovascular disease recognition systems, aligning with the demand for efficient solutions in healthcare settings. Overall, our study underscores the viability of FPGA-based solutions for edge

AI and healthcare applications, promising advancements in resource-constrained environments while maintaining high performance and accuracy levels.

Chapter 8. Conclusion and Future Work

8.1. Conclusion

This thesis represents a comprehensive and pioneering exploration into the realm of NVM-based CIM and high-level synthesis FPGA development, traversing the spectrum from theoretical analysis to tangible implementations. The foundational journey began with an intricate exposition within the introductory phase, elucidating the backdrop and impetuses driving this research pursuit. Subsequently, an extensive review of the prevailing landscape encompassing Computing-in-Memory (CIM) systems, AI algorithms, hardware implementation paradigms, and high-level synthesis was meticulously undertaken, serving as the bedrock for the introduction of a groundbreaking paradigm termed complementary NVM modelling.

The introduction of the complementary NVM modelling approach marks a notable contribution, showcasing substantial advancements in hardware simulation speed and overall performance. The novel design has laid a significant foundation for the evolution of efficient hardware implementations, enabling accelerated processing and heightened efficacy within edge devices.

Further innovation was achieved through the deployment of a software-hardware co-design framework leveraging Non-Volatile Memory (NVM). This pioneering framework was instrumental in mitigating the deleterious impact of hardware non-idealities, thereby augmenting inference accuracy. This strategic integration of software and hardware elements stands as a testament to the thesis's commitment to addressing and mitigating critical challenges encountered in contemporary AI hardware.

Moreover, the implementation of a sophisticated deep learning framework,

exemplified by the custom-designed PoolFormer on measured RRAM-based crossbar arrays, represents a substantial leap forward. This deployment effectively realized Transformer-based DNN hardware implementations, demonstrating a tangible and functional application of cutting-edge technology in the field.

Chapter 6 further solidifies the thesis's contributions by focusing on CIM design tailored to diverse AI applications. Simultaneously, Chapter 7 accentuates the versatility and efficacy of high-level synthesis on FPGA as an additional solution for edge application implementations, particularly in domains such as speech recognition and signal processing.

In conclusion, this comprehensive thesis, encompassing modeling, algorithmic development, rigorous simulation, and meticulous application verification, stands as an exceptional scholarly endeavor. Its multifaceted approach and innovative solutions to software-hardware co-design challenges underscore its significant contribution to the advancement of practical edge AIoT applications. By overcoming hurdles, breaking new ground, and offering novel methodologies, this work fundamentally pushes the boundaries of AI hardware implementations in resource-constrained environments, paving the way for transformative advancements in the field.

8.2. Future Work

Based on the preliminary results obtained from the current study, there are several bottlenecks of the current resistive crossbar array system and high-level synthesis that are worthy of further investigation. The perspective of the future work may concern the following aspects:

1. *System-Level Analysis and Algorithm Design:* Extending research into system-level analysis will involve the creation of novel learning and inference algorithm architectures aimed at enhancing overall performance. This exploration could lead to breakthroughs in optimizing the operation of resistive crossbar arrays for various applications.

2. *Circuit-Level Design:* Further investigation into circuit-level design will delve into the intricacies of key building blocks, including analog circuits and data converters. Developing innovative circuit designs and strategies will be pivotal in overcoming the challenges posed by resistive crossbar arrays and improving their efficiency.

3. *CMOS Technology Implementation and Chip Measurement:* The next step involves translating theoretical concepts into practical solutions by implementing circuits in CMOS technology. Conducting chip measurements will provide valuable insights into real-world performance and help validate the effectiveness of the proposed designs and algorithms.

4. *FPGA power and resource constraints:* optimizing hardware designs to meet stringent power and resource constraints while ensuring high performance is still a challenge compared to conventional FPGA development. Balancing the trade-offs between computational complexity and resource utilization is essential for efficient deployment in edge devices.

These future directions will contribute to the advancement of resistive crossbar array systems, making them more capable, efficient, and practical for a wide range of applications.

Author's Publications

Journal Papers:

1. **Tiancheng Cao**, Wei Son Ng, Yuan Gao, Wang Ling Goh, "Edge Cardiovascular Disease Recognition system with Quantized IncepResNet, CWT Approximation and FPGA Acceleration," *IEEE Trans. Biomed. Circuits Syst.*, **in preparation**.
2. Zhengyuan Zhang, **Tiancheng Cao**, Siyu Liu, Haoran Jin, Zesheng Zheng, Goh Wang Ling, Yuan Gao, Yuanjin Zheng, "Enhancing Handheld Photoacoustic Imaging Quality based on Energy-Efficient FeRAM Crossbar Compute-in-Memory System," *IEEE Trans. Circuits Syst. Video Technol.*, **in preparation**.
3. **Tiancheng Cao**, Weihao Yu, Yuan Gao, Chen Liu, Shuicheng Yan, Wang Ling Goh, "Edge PoolFormer: A Memristor-based Complete Circuit PoolFormer Modeling and Learning Framework for Edge-AI Applications," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, **submitted**.
4. **Tiancheng Cao**, Chen Liu, Weijie Wang, Tantan Zhang, Hock Koon Lee, Ming Hua Li, Wendong Song, Zhi Xian Chen, Victor Yi-Qian Zhuo, Nan Wang, Yao Zhu, Yuan Gao, Wang Ling Goh, "A non-idealities aware software–hardware co-design framework for edge-AI deep neural network implemented on memristive crossbar," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 12, no. 4, pp. 934-943, Dec. 2022.
5. **Tiancheng Cao**, Chen Liu, Yuan Gao, Wang Ling Goh, "Parasitic-aware modeling and neural network training scheme for energy-efficient processing-in-memory with resistive crossbar array," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 12, no. 2, pp. 436-444, June 2022.

Conference Papers:

6. **Tiancheng Cao**, Wei Son Ng, Yuan Gao, Wang Ling Goh, “FPGA Implementation of PoolFormer Network using Python-Driven High-Level Synthesis Framework for Edge-AIoT Speech Recognition*,” 2024 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, **submitted**.
7. **Tiancheng Cao**, Zhengyuan Zhang, Chen Liu, Wang Ling Goh, Yuanjin Zheng, Yuan Gao, “Energy Efficient FeRAM Crossbar Compute-in-Memory System for Handheld Photoacoustic Imaging System Enhancement” *2024 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, **submitted**.
8. **Tiancheng Cao**, Zhongyi Zhang, Wang Ling Goh, Chen Liu, Yao Zhu, Yuan Gao, “ECG Classification Using Binary CNN on RRAM Crossbar with Nonidealities-Aware Training, Readout Compensation and CWT Preprocessing,” *2023 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, Toronto, Canada, 2023.
9. **Tiancheng Cao**, Zhongyi Zhang, Wang Ling Goh, Chen Liu, Yao Zhu, Yuan Gao, "A Ternary Weight Mapping and Charge-mode Readout Scheme for Energy Efficient FeRAM Crossbar Compute-in-Memory System," *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, Hangzhou, China, pp. 1-5, 2023
10. **Tiancheng Cao**, Weihao Yu, Yuan Gao, Chen Liu, Shuicheng Yan, Wang Ling Goh, "RRAM-PoolFormer: A Resistive Memristor-based PoolFormer Modeling and Training Framework for Edge-AI Applications," *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, Monterey, CA, USA, pp. 1-5, 2023
11. **Tiancheng Cao**, Chen Liu, Yuan Gao, Wang Ling Goh, "Parasitic-Aware Modelling for Neural Networks Implemented with Memristor Crossbar Array," *2021 IEEE 14th*

International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), Singapore, Singapore, pp. 122-126, 2021

12. Chen Liu, Qiang Wang, Wenyuan Yang, **Tiancheng Cao**, Li Chen, Minghua Li, Fei Liu, Desmond Loke, Jinfeng Kang, Yao Zhu, "Multiscale Modeling of Al0.7Sc0.3N-based FeRAM: the Steep Switching, Leakage and Selector-free Array," *2021 IEEE International Electron Devices Meeting (IEDM)*, San Francisco, CA, USA, pp. 8.1.1-8.1.4, 2021.

Bibliography

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[2] N.P. Jouppi, et al., "In-datacenter performance analysis of a tensor processing unit," *Proc. 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, pp. 1-12, Jun. 2017.

[3] J. Lee, et al., "UNPU: an energy-efficient deep neural network accelerator with fully variable weight bit precision," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp: 173-185, Jan. 2019.

[4] W.Wan, R.Kubendran, C.Schaefer et al., "A compute-in-memory chip based on resistive random-access memory," *Nature* 608, pp.504–512, 2022.

[5] J. Pei et al., "Towards artificial general intelligence with hybrid Tianjic chip architecture," *Nature*, vol. 572, no. 7767, pp. 106–111, Aug. 2019.

[6] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[7] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," presented at *the 44th Annu. Int. Symp. Comput. Archit.*, Toronto, ON, Canada, 2017.

[8] X. Peng, R. Liu and S. Yu, "Optimizing weight mapping and data flow for convolutional neural networks on RRAM based processing-in-memory architecture," *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 67, no. 4, pp. 1333-1343, Apr. 2020.

[9] Y. Luo and S. Yu, "Accelerating deep neural network in-situ training with non-volatile and volatile memory based hybrid precision synapses," *IEEE Trans. on Computers*, vol. 69, no. 8, pp. 1113-1127, 1 Aug. 2020.

[10] X. Wang, M. A. Zidan and W. D. Lu, "A crossbar-based in-memory computing architecture," *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, pp. 4224-4232, Dec. 2020.

[11] W. Zhang, B. Gao, J. Tang, et al., "Neuro-inspired computing chips," *Nat Electron* 3, 371–382, 2020.

[12] J. Hasler and S. Shah, "VMM+WTA embedded classifiers learning algorithm implementable on SoC FPAA devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 1, pp. 65–76, Mar. 2018.

[13] H. Shin, D. Kim, E. Park, S. Park, Y. Park, and S. Yoo, "McDRAM: Low latency and energy-efficient matrix computations in DRAM," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2613–2622, Nov. 2018.

[14] W.-S. Khwa et al., "A 65 nm 4 Kb algorithm-dependent computing-in-memory SRAM unit-macro with 2.3 ns and 55.8TOPS/W fully parallel product-sum operation for binary DNN edge processors," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 496–498.

[15] S. Yu, "Neuro-inspired computing with emerging nonvolatile memorys," *Proceedings of the IEEE*, vol. 106, no. 2, pp. 260-285, Feb. 2018.

[16] T. Cao, W.L. Goh, Y. Gao, "Performance analysis of convolutional neural network using multi-level memristor crossbar for edge computing," *International Conf. on Intelligent Autonomous Systems (ICoIAS)*, pp. 107-111, Singapore, Feb 2020.

[17] P. Yao, H. Wu, B. Gao, et al., "Fully hardware-implemented memristor convolutional neural network," *Nature* 577, 641–646, 2020.

[18] Y. Xi, et al., "In-memory learning with analog resistive switching memory: a review and perspective," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 14-42, Jan. 2021.

[19] M. Ahn et al., "AIX: a high performance and energy efficient inference accelerator on FPGA for a DNN-based commercial speech recognition," *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Florence, Italy, 2019.

[20] R. Dorrance, F. Ren, et al., "A scalable sparse matrix-vector multiplication kernel for energy-efficient sparse-blas on FPGAs," *2014 ACM/SIGDA international symposium on Field-programmable gate arrays*, 2014.

[21] S. Han, et al., "ESE: Efficient Speech Recognition Engine with Sparse LSTM on FPGA," *2017 ACM/SIGDA international symposium on Field-programmable gate arrays*, 2017.

[22] Kim, H., et al. "4K-memristor analog-grade passive crossbar circuit," *Nature Comm.*, vol. 12, no. 1, p. 5198, Aug. 2021.

[23] F. Kiani, et al. "A fully hardware-based memristive multilayer neural network," *Science Advances*, vol. 7, no. 48, p. eabj4801, 2021.

[24] X. Xue et al., “A 22nm 4Mb 8b-precision ReRAM computing-in-memory macro with 11.91 to 195.7TOPS/W for tiny AI edge devices,” *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2021, pp. 245–247

[25] M. Hu, H. Li, Y. Chen, Q. Wu, G. S. Rose and R. W. Linderman, “Memristor crossbar-based neuromorphic computing system: a case study,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 10, pp. 1864–1878, Oct. 2014.

[26] P. Yao, H. Wu, B. Gao et al, “Fully hardware-implemented memristor convolutional neural network,” *Nature* 577, pp. 641–646, 2020.

[27] Y. Xi et al., “In-memory learning with analog resistive switching memory: a review and perspective,” *Proceedings of the IEEE*, vol. 109, no. 1, pp. 14-42, Jan. 2021.

[28] V. Ntinas, P. Karakolis, G. C. Sirakoulis and P. Dimitrakis, “Neuromorphic circuits on segmented crossbar architectures with enhanced properties,” *2020 European Conference on Circuit Theory and Design (ECCTD)*, pp. 1-6, 2020.

[29] Y. Zhang et al., “An improved RRAM-based binarized neural network with high variation-tolerated forward/backward propagation module,” in *IEEE Trans. on Electron Devices*, vol. 67, no. 2, pp. 469-473, Feb. 2020.

[30] P. Chen, X. Peng and S. Yu, “NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 12, pp. 3067-3080, Dec. 2018.

[31] S. Jain, A. Sengupta, K. Roy and A. Raghunathan, “RxNN: A framework for evaluating deep neural networks on resistive crossbars,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst*., vol. 40, no. 2, pp. 326-338, Feb. 2021.

[32] Y. Jeong, M. A. Zidan and W. D. Lu, “Parasitic effect analysis in memristor-array-based neuromorphic systems,” IEEE Trans. on *Nanotechnology*, vol. 17, no. 1, pp. 184-193, Jan. 2018.

[33] J. Woo and S. Yu, “Impact of selector devices in analog RRAM-based crossbar arrays for inference and training of neuromorphic system,” *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 9, pp. 2205-2212, Sept. 2019.

[34] Y. Chen et al., “A multifault-tolerant training scheme for nonideal memristive neural networks,” *Advanced Intelligent Systems*, Feb. 2022.

[35] A. Vaswani, et al., “Attention is all you need,” *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[36] S. Zhang, F. Su, Y. Wang, S. Mai, K. P. Pun and X. Tang, “A low-power keyword spotting system with high-order passive switched-capacitor bandpass filters for analog-MFCC feature extraction, ” *IEEE Trans. Circuits Syst. I, Reg. Papers*, pp. 1–14, Aug. 2023.

[37] L. O. Chua, “Memristor - the missing circuit element,” *IEEE Trans. Circuit Theory* 18, pp. 507–519, 1971.

[38] Z. Wang et al., “Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing,” *Nature Mater*., Article, vol. 16, p. 101, Sep. 2016.

[39] W. Sun et al., “Understanding memristive switching via in situ characterization and device modeling,” *Nature Commun*., vol. 10, no. 1, p. 3453, Dec. 2019.

[40] M.-F. Chang et al., “A 0.5 V 4 Mb logic-process compatible embedded resistive RAM (ReRAM) in 65 nm CMOS using low-voltage current-mode sensing scheme with 45ns random read time,” in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, pp. 434–436, Feb. 2012.

[41] M.-F. Chang et al., “19.4 Embedded 1Mb ReRAM in 28nm CMOS with 0.27-to-1 V read using swing-sample-and-couple sense amplifier and self-boost-write-termination scheme,” in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, pp. 332–333, Feb. 2014.

[42] M. Bocquet et al., “In-memory and error-immune differential RRAM implementation of binarized deep neural networks,” in *IEDM Tech. Dig.*, pp. 20.6.1–20.6.4, Dec. 2018.

[43] A. Shafiee et al., “ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars,” in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, pp. 14–26, Jun. 2016.

[44] A. M. Sheri, H. Hwang, M. Jeon, and B.-G. Lee, “Neuromorphic character recognition system with two PCMO memristors as a synapse,” *IEEE Trans. Ind. Electron*., vol. 61, no. 6, pp. 2933–2941, Jun. 2014.

[45] J. Woo et al., “Improved synaptic behavior under identical pulses using AlO x /HfO 2 bilayer RRAM array for neuromorphic systems,” *IEEE Electron Device Lett*., vol. 37, no. 8, pp. 994–997, Aug. 2016.

[46] W. Wu, H. Wu, B. Gao, N. Deng, S. Yu, and H. Qian, "Improving analog switching in HfO x based resistive memory with thermal enhanced layer," *IEEE Electron Device Lett.*, vol. 38, no. 8, pp. 1019–1022, 2017.

[47] X. Sun and S. Yu, "Impact of non-ideal characteristics of resistive synaptic devices on implementing convolutional neural networks, " *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 3, pp. 570-579, Sep. 2019.

[48] E. R. Kandel, et al. "Principles of Neural Science," vol. 4, McGraw-hill New York, 2000.

[49] W.S. McCulloch, W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics* 5, 115–133, 1943.

[50] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, Vol. 65, pp. 368-408, 1958.

[51] B. Widrow, W. H. Pierce, J. B. Angell, "Birth, Life, and Death in Microelectronic Systems," *IRE Trans. Mil. Electron*, pp. 191–201, 1961.

[52] L. D. Jackel, "Artificial neural networks for computing," *Journal of Vacuum Science and Technology*. B 4, 61, 1986.

[53] D. B. Strukov, G. S. Snider, D. R. Stewart, R. S. Williams, "The missing memristor found," *Nature* 453, pp. 80–83, 2008.

[54] S. H. Jo, et al. "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Lett*. 10, pp. 1297–1301, 2010.

[55] M. Jerry et al., "Ferroelectric FET analog synapse for acceleration of deep neural network training," in *IEDM Tech. Dig*., Dec. 2017

[56] D. Kuzum, R. G. D. Jeyasingh, B. Lee, and H.-S.-P. Wong, "Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing," *Nano Lett.*, vol. 12, no. 5, pp. 2179–2186, May 2012.

[57] S. Yu, Y. Wu, R. Jeyasingh, D. Kuzum, and H.-S.-P. Wong, "An electronic synapse device based on metal oxide resistive switching memory for neuromorphic computation," *IEEE Trans. Electron Devices*, vol. 58, no. 8, pp. 2729–2737, Aug. 2011.

[58] G. W. Burr et al., "Neuromorphic computing using non-volatile memory," *Adv. Phys*. X, vol. 2, no. 1, pp. 89–124, 2017.

[59] G. C. Adam, B. D. Hoskins, M. Prezioso, F. Merrikh-Bayat, B. Chakrabarti, and D. B. Strukov, "3-D memristor crossbars for analog and neuromorphic computing applications," *IEEE Trans. Electron Devices*, vol. 64, no. 1, pp. 312–318, Jan. 2017.

[60] ITRS, "International technology roadmap for semiconductors," 2013.

[61] J. Chen, et al., "A parallel multibit programing scheme with high precision for RRAM-based neuromorphic systems*," IEEE Trans. on Electron Devices*, vol. 67, no. 5, pp. 2213-2217, May 2020.

[62] M. Hu, et al., "Memristor-based analog computation and neural network classification with a dot product engine," *Adv. Mater*. 30, 1705914, 2018.

[63] C. Liu et al., "A memristor crossbar based computing engine optimized for high speed and accuracy," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, pp. 110–115, Jul. 2016.

[64] Y. Li, et al., "The impact of interconnect resistance on one-selector one-resistor (1S1R) crossbar array performance," *2019 China Semicon. Tech. International Conference (CSTIC)*, Shanghai, China, Mar 2019.

[65] A. Chen, "A comprehensive crossbar array model with solutions for line resistance and nonlinear device characteristics," *IEEE Trans. on Electron Devices*, vol. 60, no. 4, pp. 1318-1326, Apr. 2013.

[66] B. Liu et al., "Reduction and IR-drop compensations techniques for reliable neuromorphic computing systems," *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2014.

[67] M. E. Fouda, et al, "Overcoming crossbar nonidealities in binary neural networks through learning," *2018 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH),* Athens, Greece, 2018.

[68] Y. Liao et al., "A compact model of analog RRAM with device and array nonideal effects for neuromorphic systems," *IEEE Trans. on Electron Devices*, vol. 67, no. 4, pp. 1593-1599, Apr. 2020.

[69] C.D. Meyer, Matrix analysis and applied linear algebra, *SIAM*, 2000.

[70] K. Chen, Matrix preconditioning techniques and applications, Cambridge University Press, 2005.

[71] Y. LeCun. (1998). The MNIST database of handwritten digits. [Online]. Available: http://yann.lecun.com/exdb/mnist/.

[72] Z. Fahimi, et al., "The impact of device uniformity on functionality of analog passively-integrated memristive circuits, " *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 68, no. 10, pp. 4090-4101, Oct. 2021.

[73] W. Wang, et al, "Enabling neuromorphic computing: BEOL integration of CMOS RRAM chip and programmable performance," *IEEE International System-on-Chip Conference (SOCC)*, pp. 354-358, Singapore, Sep. 2019.

[74] T. Cao, W. L. Goh and Y. Gao, "Parasitic-aware modelling for neural networks implemented with memristor crossbar array," *IEEE International Symp. on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pp. 122-126, Singapore, Dec 2021.

[75] Y. Xi et al., "In-memory learning with analog resistive switching memory: a review and perspective," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 14-42, Jan. 2021.

[76] F. Cai, J.M. Correll, S.H. Lee, et al., "A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations," *Nat. Electron*, vol. 2, pp. 290–299, 2019.

[77] M.E. Fouda, et. al. "Modeling and analysis of passive switching crossbar arrays," *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 65, no. 1, pp. 270-282, Jan. 2017.

[78] H..-S. P. Wong et al., "Phase change memory," *Proceedings of the IEEE*, vol. 98, no. 12, pp. 2201-2227, Dec. 2010.

[79] D. Apalkov, B. Dieny and J. M. Slaughter, "Magnetoresistive random access memory," *Proceedings of the IEEE*, vol. 104, no. 10, pp. 1796-1830, Oct. 2016.

[80] H. . -S. P. Wong et al., "Metal–oxide RRAM," in *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951-1970, June 2012.

[81] Y. Jiang et al., "HARNS: high-level architectural model of RRAM based computing-in-memory NPU," *2021 IEEE International Conference on Integrated Circuits, Technologies and Applications (ICTA)*, pp. 35-36, 2021.

[82] S. Jain and A. Raghunathan, "CxDNN: Hardware-software compensation methods for deep neural networks on resistive crossbar systems," *ACM Trans. Embedded Comput. Syst.*, vol. 18, no. 6, pp. 1–23, Jan. 2020.

[83] S. Roy, S. Sridharan, S. Jain and A. Raghunathan, "TxSim: Modeling training of deep neural networks on resistive crossbar systems," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 4, pp. 730-738, April 2021.

[84] Z. Zhu et al., "MNSIM 2.0: A behavior-level modeling tool for memristor-based neuromorphic computing systems," *2020 Great Lakes Symposium on VLSI (GLSVLSI)* , pp. 83-88, Sept. 2020.

[85] A. Lu, X. Peng, W. Li, H. Jiang and S.Yu, "NeuroSim simulator for compute-in-memory hardware accelerator: validation and benchmark," *Frontiers in Artificial Intelligence*, vol. 4, 2021.

[86] S. Agarwal et al., "Resistive memory device requirements for a neural algorithm accelerator," *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, pp. 929–938, Jul. 2016.

[87] Alex Krizhevsky (2009). Learning Multiple Layers of Features from Tiny Images. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html.

[88] W. Wang et al., "Efficient training of the memristive deep belief net immune to non-idealities of the synaptic devices," *Advanced Intelligent Systems*, Feb. 2022.

[89] Y. Long, X. She and S. Mukhopadhyay, "Design of reliable DNN accelerator with un-reliable ReRAM," *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1769-1774, 2019.

[90] T. Cao, C. Liu, Y. Gao and W. L. Goh, "Parasitic-aware modeling and neural network training scheme for energy-efficient processing-in-memory with resistive crossbar array," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 12, no. 2, pp. 436-444, Jun. 2022.

[91] L. Xia, M. Liu, X. Ning, K. Chakrabarty and Y. Wang, "Fault-Tolerant Training Enabled by On-Line Fault Detection for RRAM-Based Neural Computing Systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 9, pp. 1611-1624, Sept. 2019.

[92] X. Fei, Y. Zhang and W. Zheng, "XB-SIM*: A simulation framework for modeling and exploration of ReRAM-based CNN acceleration design," in *Tsinghua Science and Technology*, vol. 26, no. 3, pp. 322-334, June 2021.

[93] X. Peng, S. Huang, H. Jiang, A. Lu and S. Yu, "DNN+NeuroSim V2.0: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators for On-Chip Training," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 11, pp. 2306-2319, Nov. 2021.

[94] A. Vaswani, et al.,"Attention is all you need," in *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[95] N. Carion and F. Massa, et al., "End-to-end object detection with transformers," in *European conference on computer vision (ECCV)*, 2020, pp. 213-229.

[96] A. Dosovitskiy and L. Beyer, et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations (ICLR)*, 2020.

[97] Z. Liu and Y. Lin, et al., "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 10012-10022.

[98] L. Yuan and Y. Chen, et al., "Tokens-to-token vit: Training vision transformers from scratch on imagenet," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 558-567.

[99] P. Ramachandran and N. Parmar, et al., "Stand-alone self-attention in vision models," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 32, 2019.

[100] H. Wang and Z. Wu, et al., "HAT: Hardware-aware transformers for efficient natural language processing," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020, pp. 7675-7688.

[101] B. Graham and A. El-Nouby, et al., "Levit: a vision transformer in convnet's clothing for faster inference," *in Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*, 2021, pp. 12259-12269.

[102] A. Hassani and S. Walton, et al., "Escaping the big data paradigm with compact transformers," arXiv preprint arXiv:2104.05704, 2021.

[103] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, and S. Yan, "Metaformer is actually what you need for vision," In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10819-10829.

[104] X. Yang, B. Yan, H. Li and Y. Chen, "ReTransformer: ReRAM-based processing-in-memory architecture for transformer acceleration," *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1-9.

[105] M. Kang, H. Shin and L. -S. Kim, "A framework for accelerating transformer-based language model on ReRAM-based architecture," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 41, no. 9, pp. 3026-3039, Sept. 2022

[106] H. Sun, Z. Zhu, Y. Cai, X. Chen, Y. Wang and H. Yang, "bit-efficient quantized and regularized training framework for processing-in-memory accelerators," *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020, pp. 325-330

[107] C. Yang, X. Wang and Z. Zeng, "Full-circuit implementation of transformer network based on memristor," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 69, no. 4, pp. 1395-1407, April 2022.

[108] T. Cao, W. Yu, Y. Gao, C. Liu, S. Yan and W. L. Goh, "RRAM-PoolFormer: A Resistive Memristor-based PoolFormer Modeling and Training Framework for Edge-AI Applications," *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, Monterey, CA, USA, 2023.

[109] M. Zhou, W. Xu, J. Kang and T. Rosing, "TransPIM: A memory-based acceleration via software-hardware co-design for transformer," *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2022, pp. 1071-1085.

[110] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pages 770– 778.

[111] T. Cao et al., "A non-idealities aware software–hardware co-design framework for edge-AI deep neural network implemented on memristive crossbar," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 12, no. 4, pp. 934-943, Dec. 2022.

[112] Y. Bi, Q. Xu, H. Geng, S. Chen, and Y. Kang, "AD2VNCS: adversarial defense and device variation-tolerance in memristive crossbar-based neuromorphic computing systems," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 29, no. 1, pp. 8-27, Jan. 2024. doi: 10.1109/TCAD.2023.3600231.

[113] S. K. Roy, A. Patil and N. R. Shanbhag, "Fundamental limits on the computational accuracy of resistive crossbar-based in-memory architectures," *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, Austin, TX, USA, pp. 384-388, 2022.

[114]C. Wang et al., "Multi-state memristors and their applications: an overview," *IEEE J. Emerg. Sel. Topics Circuits Syst.,* vol. 12, no. 4, pp. 723-734, Dec. 2022.

[115]J. Kang et al., "Time-dependent variability in RRAM-based analog neuromorphic system for pattern recognition," *2017 IEEE International Electron Devices Meeting (IEDM),* pp. 6.4.1-6.4.4, 2017.

[116]Q. Zheng et al., "Artificial neural network based on doped HfO2 *ferroelectric* capacitors with multilevel characteristics," *IEEE Electron Device Lett.,* vol. 40, no. 8, pp. 1309–131, Aug. 2019.

[117]D. Kwon and I.-Y. Chung, "Capacitive neural network using chargestored memory cells for pattern recognition applications," *IEEE Electron Device Lett.,* vol. 41, no. 3, pp. 493–496, Mar. 2020.

[118]Y. -C. Luo, A. Lu, J. Hur, S. Li and S. Yu, "Design of non-volatile capacitive crossbar array for in-memory computing," *2021 IEEE International Memory Workshop (IMW),* pp. 1-4, 2021.

[119]Z. Wang, M. Rao, JW. Han, et al., "Capacitive neural network with neuro-transistors," *Nat. Commun.,* vol. 9, no. 3208, Aug. 2018.

[120]U. Schroeder, M.H. Park, T. Mikolajick, et al., "The fundamentals and applications of ferroelectric HfO2," *Nat. Rev. Mater.,* vol. 7, pp. 653–669, Mar. 2022.

[121]Y. -C. Luo, A. Lu, J. Hur, S. Li and S. Yu, "Design and optimization of non-volatile capacitive crossbar array for in-memory computing," *IEEE Trans. Circuits Syst. II Express Briefs,* vol. 69, no. 3, pp. 784-788, Mar. 2022.

[122]T. Mikolajick et al., "Next generation ferroelectric materials for semiconductor process integration and their applications," *J. Appl. Phys.,* vol. 129, no. 10, Mar. 2021.

[123]S. Fichtner, N. Wolff, F. Lofink, L. Kienle, and B. Wagner, "AlScN: A III-V semiconductor based ferroelectric," *J. Appl. Phys.,* vol. 125, no. 11, 2019, doi: 10.1063/1.5084945.

[124]C. Liu et al., "Multiscale modeling of Al0.7Sc0.3N-based FeRAM: the steep switching, leakage and selector-free array," *2021 IEEE International Electron Devices Meeting (IEDM),* pp. 8.1.1-8.1.4, 2021.

[125]P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," arXiv:1804.03209, 2018, [online] Available: https://arxiv.org/abs/1804.03209.

[126]M. Croce, B. Friend, F. Nesta, L. Crespi, P. Malcovati and A. Baschirotto, "A 760-nW, 180-nm CMOS fully analog voice activity detection system for domestic environment," *IEEE J. Solid-State Circuits,* vol. 56, no. 3, pp. 778-787, Mar. 2021.

[127]M. Yang, C. -H. Chien, T. Delbruck and S. -C. Liu, "A 0.5V 55µW 64×2 channel binaural silicon cochlea for event-driven stereo-audio sensing," *IEEE J. Solid-State Circuits,* vol. 51, no. 11, pp. 2554-2569, Nov. 2016.

[128]C. J. Schaefer, M. Horeni, P. Taheri and S. Joshi, "LSTMs for keyword spotting with ReRAM-based compute-in-memory architectures," *2021 IEEE International Symposium on Circuits and Systems (ISCAS),* Daegu, Korea, pp. 1-5, 2021.

[129]W.Wan, R.Kubendran, C.Schaefer et al., "A compute-in-memory chip based on resistive random-access memory," *Nature* 608, pp.504–512, 2022.

[130]X. Jia et al., "A 11.6µW computing-on-memory-boundary keyword spotting processor with joint MFCC-CNN ternary quantization," *2022 IEEE International Symposium on Circuits and Systems (ISCAS),* Austin, TX, USA, pp. 2816-2820, 2022.

[131]V. P. K.Miriyala, M.Ishii, "Ultra-low power on-chip learning of speech commands with phase-change memories," arXiv: 2010.11741, 2020, [online] Available: https://arxiv.org/abs/2010.11741.

[132]A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM,* vol. 60, no. 6, pp. 84–90, May 2017.

[133]Y. Lu, V. L. Le and T. T. -H. Kim, "9.7 A 184 µ W real-time hand-gesture recognition system with hybrid tiny classifiers for smart wearable devices," *2021 IEEE International Solid- State Circuits Conference (ISSCC),* San Francisco, CA, USA, pp. 156-158, 2021.

[134]Y. Lu, V. L. Le and T. T. -H. Kim, "A 184-µW error-tolerant real-time hand gesture recognition system with hybrid tiny classifiers utilizing edge CNN," *IEEE J. Solid-State Circuits,* vol. 58, no. 2, pp. 530-542, Feb. 2023.

[135]X. He, K. Wang, H. Huang, T. Miyazaki, Y. Wang, and S. Guo, "Green resource allocation based on deep reinforcement learning in content-centric IoT," *IEEE Trans. Emerg. Topics Comput.,* Feb. 13, 2018.

[136]S. S. Virani et al., "Heart disease and stroke statistics—2021 update," *Circulation,* vol. 143, no. 8, pp. 1–6, Feb. 2021.

[137] T. Yang, L. Yu, Q. Jin, L. Wu, and B. He, "Localization of origins of premature ventricular contraction by means of convolutional neural network from 12-Lead ECG," *IEEE Trans. Biomed. Eng*., vol. 65, no. 7, pp. 1662–1671, Jul. 2018.

[138] X. Sun, R. Liu, X. Peng and S. Yu, "Computing-in-Memory with SRAM and RRAM for Binary Neural Networks," *2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, 2018.

[139] M. Courbariaux, et al., "Binarized neural network: Training deep neural networks with weights and activations constrained to+ 1 or-1," arXiv: 1602.02830, 2016.

[140] M. Rastegari, et al., "XNOR-net: ImageNet classification using binary convolutional neural networks," arXiv: 1603.05279, 2016.

[141] Y. Zhao, Z. Shang, and Y. Lian, "A 13.34 μw event-driven patient-specific ANN cardiac arrhythmia classifier for wearable ECG sensors," *IEEE Trans. Biomed. Circuits Syst*., vol. 14, no. 2, pp. 186–197, Apr. 2020.

[142] M. Saeed et al., "Evaluation of level-crossing ADCs for event-driven ECG classification," *IEEE Trans. Biomed. Circuits Syst*., vol. 15, no. 6, pp. 1129–1139, Dec. 2021.

[143] S. Agarwal, O. D. Parekh, T. Quach, C. D. James, J. B. Aimone and M. Marinella, "The energy scaling advantages of RRAM crossbars," *2015 Fourth Berkeley Symposium on Energy Efficient Electronic Systems (E3S)*, Berkeley, CA, pp. 1-3, 2015.

[144] D. L. T. Wong, Y. Li, D. John, W. K. Ho and C. H. Heng, "Resource and energy efficient implementation of ECG classifier using binarized CNN for edge AI devices," *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, Daegu, Korea, pp. 1-5, 2021.

[145] E. Esmanhotto et al., "High-density 3D monolithically integrated multiple 1T1R multi-level-cell for neural networks," *2020 IEEE International Electron Devices Meeting (IEDM)*, San Francisco, CA, USA, 2020.

[146] S. Diware et al., "Severity-based hierarchical ECG classification using neural networks," *IEEE Trans. Biomed. Circuits Syst.*, vol. 17, no. 1, pp. 77-91, Feb. 2023.

[147] Y. -F. Qin, R. Kuang, X. -D. Huang, Y. Li, J. Chen and X. -S. Miao, "Design of high robustness BNN inference accelerator based on binary memristors," in *IEEE Trans. Electron Devices*, vol. 67, no. 8, pp. 3435-3441, Aug. 2020.

[148] Y. Jeong and W. Lu, "Neuromorphic computing using memristor crossbar networks: a focus on bio-inspired approaches," in *IEEE Nanatechnol. Mag*., vol. 12, no. 3, pp. 6-18, Sept. 2018.

[149] Z. Dong et al., "Convolutional neural networks based on RRAM devices for image recognition and online learning tasks," in *IEEE Trans. Electron Devices*, vol. 66, no. 1, pp. 793-801, Jan. 2019.

[150] M. Prezioso, F. Merrikh-Bayat, B. Hoskins et al, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, pp. 61–64, 2015.

[151] L. Xu, C. Li and L. Chen, "Analog memristor based neuromorphic crossbar circuit for image recognition," *2015 Sixth International Conference on Intelligent Control and Information Processing (ICICIP)*, Wuhan, pp. 155-160, 2015.

[152] T. Hirtzlin et al., "Outstanding bit error tolerance of resistive RAM-based binarized neural networks," *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2019.

[153] X. Sun, X. Peng, P. Chen, R. Liu, J. Seo and S. Yu, "Fully parallel RRAM synaptic array for implementing binary neural network with (+1, −1) weights and (+1, 0) neurons," *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2018.

[154] A. J. Casson and E. Rodriguez-Villegas, "A 60 pW gmC continuous wavelet transform circuit for portable EEG systems," *IEEE J. Solid-State Circuits*, vol. 46, no. 6, pp. 1406-1415, June 2011.

[155] C. Li, M. Hu, Y. Li et al, "Analogue signal and image processing with large memristor crossbars," *Nat Electron*, vol 1, pp. 52–59, 2018.

[156] A. Ullah, S. M. Anwar, M. Bilal, and R. M. Mehmood, "Classification of arrhythmia by using deep learning with 2-D ECG spectral image representation," *Remote Sens*., vol. 12, no. 10, p. 1685, May 2020.

[157] D. Li, J. Zhang, Q. Zhang and X. Wei, "Classification of ECG signals based on 1D convolution neural network," *2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom)*, Dalian, China, 2017.

[158] L. V. Wang, "Multiscale photoacoustic microscopy and computed tomography," *Nat. Photonics*, vol. 3, no. 9, pp. 503–509, 2009.

[159] Z. Zhang et al., "Deep and domain transfer learning aided photoacoustic microscopy: Acoustic resolution to optical resolution," *IEEE Trans. Med. Imag*., vol. 41, no. 12, pp. 3636–3648, 2022.

[160] Z. Zhang et al., "Adaptive enhancement of acoustic resolution photoacoustic microscopy imaging via deep CNN prior," *Photoacoustics*, vol. 30, p. 100484, 2023.

[161] Z. Zhang, H. Jin, Z. Zheng, Y. Luo, and Y. Zheng, "Photoacoustic microscopy imaging from acoustic resolution to Optical Resolution Enhancement with Deep Learning," *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021.

[162] Z. Zhang, H. Jin, Z. Zheng, and Y. Zheng, "Learning-based algorithm for Real Imaging System Enhancement: Acoustic Resolution to optical resolution photoacoustic microscopy," *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2022.

[163] Z. Zhang, H. Jin, Z. Zheng, and Y. Zheng, "Super acoustic resolution photoacoustic microscopy imaging enhancement," *2022 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2022.

[164] T. Cao et al., "A ternary weight mapping and charge-mode readout scheme for energy efficient FeRAM crossbar compute-in-memory system," *2023 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2023.

[165] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, pp. 2818-2826, 2016.

[166] H. Gulzar, M. Shakeel, K. Nishida, K. Itoyama, K. Nakadai and H. Amano, "CASE: CNN acceleration for speech-classification in edge-computing," *2021 IEEE Cloud Summit (Cloud Summit)*, Hempstead, NY, USA, pp. 63-68, 2021.

[167] J. Yoon, D. Lee, N. Kim, S. -J. Lee, G. -H. Kwak and T. -H. Kim, "A real-time keyword spotting system based on an end-to-end binary convolutional neural network in FPGA," *2023 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS)*, Tokyo, Japan, 2023.

[168] S. Yang, Z. Gong, K. Ye, Y. Wei, Z. Huang and Z. Huang, "EdgeRNN: a compact speech recognition network with spatio-temporal features for edge computing," *IEEE Access*, vol. 8, pp. 81468-81478, 2020.

[169] Y. Zhou, W. Xing, K. I. Maslov, L. A. Cornelius, and L. V. Wang, "Handheld photoacoustic microscopy to detect melanoma depth in vivo," *Optics Letters*, vol. 39, no. 16, p. 4731, Aug. 2014.

[170] K. Park *et al.*, "Handheld photoacoustic microscopy probe," *Sci. Rep.*, vol. 7, no. 1, Oct. 2017.

[171] J. Aguirre *et al*., "Assessing nailfold microvascular structure with ultra-wideband Raster-scan optoacoustic mesoscopy," *Photoacoustics*, vol. 10, Jun. 2018.

[172] S. Liu, W. Song, X. Liao, T. T. -H. Kim and Y. Zheng, "Development of a handheld volumetric photoacoustic imaging system with a central-holed 2D matrix aperture," *IEEE Trans. Biomed. Circuits Syst*., vol. 67, no. 9, pp. 2482-2489, Sept. 2020.

[173] S. Yu, Y.-C. Luo, T.-H. Kim, O. Phadke, "Non-volatile capacitive synapse: Device candidates for charge domain compute-in-memory," *IEEE Electron Devices Magazine*, vol. 1, no. 2, pp. 23-32, 2023.