Computation-In-Memory based Edge-AI for Healthcare
A Cross-Layer Approach

Diware, S.S.

DOI
10.4233/uuid:fe563dc8-6515-41d3-8d84-d86608ac0fc6

Publication date
2024

Document Version
Final published version

Citation (APA)
Diware, S. S. (2024). *Computation-In-Memory based Edge-AI for Healthcare: A Cross-Layer Approach*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:fe563dc8-6515-41d3-8d84-d86608ac0fc6

Computation-In-Memory based Edge-AI for Healthcare

A Cross-Layer Approach

Computation-In-Memory based Edge-AI for Healthcare

A Cross-Layer Approach

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. dr. ir. T.H.J.J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op woensdag 30 Oktober 2024 om 12:30 uur

door

Sumit Shaligram Diware

Master of Technology in VLSI Design Tools and Technology,
Indian Institute of Technology (I.I.T.) Delhi, India,
geboren te Akola, India.

Rector Magnificus,	voorzitter
Prof. dr. ir. S. Hamdioui,	Technische Universiteit Delft, promotor
Dr.-ing. R.K. Bishnoi,	Technische Universiteit Delft, copromotor
Onafhankelijke leden:	
Prof.dr.ir. G. Gaydadjiev	Technische Universiteit Delft
Prof.dr.ir. W. A. Serdijn	Technische Universiteit Delft
Prof.dr.-ing. M. Berekovic	Universität zu Lübeck, Germany
Prof.dr. A. Singh	Auburn University, USA
Dr. M. Ottavi	Universiteit Twente, The Netherlands

Keywords: Computation-in-memory, Processing-in-memory, Neural Networks, ECG, Memristor, Non-idealities, Neuromorphic Computing

Printed by: Ipskamp Printing, The Netherlands

Front & Back: By S. S. Diware.

ISBN 978-94-6366-928-3

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

CONTENTS

SUMMARY

Artificial intelligence (AI) is rapidly becoming an integral part of many real-world products and services. This is mainly facilitated by the extensive computing resources provided by the cloud infrastructure. However, cloud-based AI processing suffers from drawbacks like high latency, huge network costs, data privacy/security concerns, and service disruptions due to internet outage. Edge computing for AI (edge-AI) addresses these problems by combining data sources with on-board AI processing hardware. Such hardware must be energy efficient to achieve prolonged operation, given the limited energy resources on edge devices. Moreover, it should be compact in size to facilitate seamless system integration and enhanced portability. Conventional hardware cannot meet these requirements due to data transfer bottleneck in von Neumann architecture and limitations of conventional memory technologies. Computation-in-memory (CIM) overcomes these challenges by in-situ data processing using emerging memory technologies called memristors. Thus, CIM can facilitate energy efficient and compact edge-AI hardware design. Healthcare domain stands out as a prime target for CIM-based edge-AI hardware, due to two main reasons. Firstly, it holds significant real-world importance due to its direct impact on human well-being. Secondly, the increasing adoption of AI in healthcare can significantly benefit from efficient hardware for data processing. CIM-based edge hardware can greatly enhance the effectiveness of AI-based healthcare through rapid, reliable, and secure processing of medical data at its source. Hence, design of CIM-based edge-AI hardware for healthcare applications presents a promising research direction.

The process of designing CIM-based edge-AI hardware for healthcare can be expressed as a stack of six abstraction layers: application, algorithm, optimization, mapping, micro-architecture and circuits, and device. These abstraction layers can be further grouped into two distinct design phases. The first phase is application-dependent, covering the first three abstraction layers (application, algorithm and optimization). It involves creating a customized neural network model for the given healthcare application. The challenge in this phase is to achieve strong algorithmic performance, while incorporating features to exploit the full potential of CIM hardware. Conversely, the second phase is application-independent and comprises of the remaining abstraction layers (mapping, micro-architecture and circuits, and device). It solely focuses on translating the model computations into CIM hardware operations. However, the non-ideal characteristics of memristor devices introduce computational errors in hardware operations. This undermines the advantages of CIM as energy-efficient computations are of no use if they are incorrect. Hence, mitigating memristor non-idealities becomes the primary challenge in this phase. Moreover, it is important to integrate the customized model and non-ideality mitigation strategies into a comprehensive hardware solution and realize it through prototyping. This gives rise to the following three research topics: 1) healthcare AI models for CIM-based edge hardware, 2) dealing with memristor non-idealities, and 3) CIM edge-AI prototyping for healthcare.

We adopt a cross-layer approach in this thesis to address these research topics, covering all six layers of the CIM abstraction stack. We begin by creating neural network models for two healthcare applications: cardiac arrhythmia classification and diabetic retinopathy screening. Our contributions in this application-dependent design phase span across the first three abstraction layers (application, algorithm and optimization). At the application layer, we introduce new features in the model tailored to the specific healthcare application. This enhances its real-world impact by addressing the unique medical needs more effectively. Moving to the algorithm layer, we customize the computational flow within the model to exploit the characteristics of the healthcare data. This improves design performance in key aspects like accuracy and energy efficiency. Moreover, we strategically refine the model computations to further maximize post-deployment benefits on CIM hardware. At the optimization layer, we employ techniques like resampling, quantization and pruning to optimize hardware resource requirements, without compromising the model's algorithmic performance.

After creating the neural network models, we proceed to the application-independent design phase. Focusing on RRAM-based memristor devices, we first identify three key non-idealities that significantly impact inference accuracy on CIM hardware. We then devise mitigation strategies against these non-idealities, encompassing the remaining abstraction layers (mapping, micro-architecture and circuits, and device). At mapping layer, we propose a hardware-aware training methodology to combat the conductance variation non-ideality. Moving to the micro-architecture level, we present two mitigation strategies. The first addresses non-zero G_{min} error non-ideality through a novel approach to CIM micro-architecture design. The second introduces an adaptive micro-architecture that adjusts its sensing conditions to counteract the effects of read-disturb non-ideality. At the device level, these strategies indirectly contribute by circumventing the necessity for extensive device engineering, ensuring accurate inference even in the presence of non-idealities. Building upon this foundation of model development and non-ideality mitigation, we integrate the optimal ECG classification model with the proposed mitigation strategies to create a CIM edge-AI prototype. Thus, our contributions pave the way towards a future with enhanced effectiveness and efficiency of AI-powered healthcare.

SAMENVATTING

Kunstmatige intelligentie (AI) wordt in snel tempo een integraal onderdeel van veel producten en diensten in de echte wereld. Dit wordt voornamelijk mogelijk gemaakt door de uitgebreide computerbronnen die door de cloudinfrastructuur worden geboden. Cloudgebaseerde AI-verwerking heeft echter te kampen met nadelen zoals hoge latentie, enorme netwerkkosten, zorgen over gegevensprivacy/-beveiliging en serviceonderbrekingen als gevolg van internetstoringen. Edge computing voor AI (edge-AI) pakt deze problemen aan door databronnen te combineren met ingebouwde AI-verwerkingshardware. Dergelijke hardware moet energie-efficiënt zijn om langdurig gebruik te kunnen garanderen, gezien de beperkte energiebronnen op edge-apparaten. Bovendien moet het compact van formaat zijn om naadloze systeemintegratie en verbeterde draagbaarheid te vergemakkelijken. Conventionele hardware kan niet aan deze eisen voldoen vanwege het knelpunt in de gegevensoverdracht in de von Neumann-architectuur en de beperkingen van conventionele geheugentechnologieën. Computation-in-memory (CIM) overwint deze uitdagingen door in-situ gegevensverwerking met behulp van opkomende geheugentechnologieën die memristors worden genoemd. CIM kan dus een energiezuinig en compact edge-AI-hardwareontwerp mogelijk maken. Het gezondheidszorgdomein onderscheidt zich als een belangrijk doelwit voor op CIM gebaseerde edge-AI-hardware, vanwege twee belangrijke redenen. Ten eerste is het in de praktijk van groot belang vanwege de directe impact op het menselijk welzijn. Ten tweede kan de toenemende acceptatie van AI in de gezondheidszorg aanzienlijk profiteren van efficiënte hardware voor gegevensverwerking. Op CIM gebaseerde edge-hardware kan de effectiviteit van op AI gebaseerde gezondheidszorg aanzienlijk vergroten door snelle, betrouwbare en veilige verwerking van medische gegevens aan de bron. Daarom biedt het ontwerp van CIM-gebaseerde edge-AI-hardware voor toepassingen in de gezondheidszorg een veelbelovende onderzoeksrichting.

Het proces van het ontwerpen van CIM-gebaseerde edge-AI-hardware voor de gezondheidszorg kan worden uitgedrukt als een stapel van zes abstractielagen: applicatie, algoritme, optimalisatie, mapping, microarchitectuur en circuits, en apparaat. Deze abstractielagen kunnen verder worden gegroepeerd in twee verschillende ontwerpfasen. De eerste fase is applicatie-afhankelijk en omvat de eerste drie abstractielagen (applicatie, algoritme en optimalisatie). Het gaat om het creëren van een aangepast neuraal netwerkmodel voor de gegeven zorgtoepassing. De uitdaging in deze fase is het bereiken van sterke algoritmische prestaties, terwijl functies worden geïntegreerd om het volledige potentieel van CIM-hardware te benutten. Omgekeerd is de tweede fase toepassingsonafhankelijk en bestaat uit de overige abstractielagen (mapping, microarchitectuur en circuits, en apparaat). Het richt zich uitsluitend op het vertalen van de modelberekeningen naar CIM-hardwarebewerkingen. De niet-ideale kenmerken van memristorapparaten introduceren echter rekenfouten bij hardwarebewerkingen. Dit ondermijnt de voordelen van CIM, aangezien energie-efficiënte berekeningen nutteloos zijn als ze onjuist zijn.

Daarom wordt het verzachten van de niet-idealiteiten van memristors de belangrijkste uitdaging in deze fase. Bovendien is het belangrijk om het aangepaste model en de strategieën voor het beperken van niet-idealiteiten te integreren in een alomvattende hardwareoplossing en deze te realiseren door middel van prototyping. Dit geeft aanleiding tot de volgende drie onderzoeksthema's: 1) AI-modellen voor de gezondheidszorg voor CIM-gebaseerde edge-hardware, 2) omgaan met niet-idealiteiten van memristors, en 3) CIM edge-AI-prototyping voor de gezondheidszorg.

In dit proefschrift hanteren we een cross-layer benadering om deze onderzoeksonderwerpen aan te pakken, waarbij we alle zes lagen van de CIM-abstractiestapel bestrijken. We beginnen met het creëren van neurale netwerkmodellen voor twee toepassingen in de gezondheidszorg: classificatie van hartritmestoornissen en screening van diabetische retinopathie. Onze bijdragen in deze applicatie-afhankelijke ontwerpfase bestrijken de eerste drie abstractielagen (applicatie, algoritme en optimalisatie). Op de applicatielaag introduceren we nieuwe features in het model afgestemd op de specifieke zorgapplicatie. Dit vergroot de impact ervan in de echte wereld door effectiever in te spelen op de unieke medische behoeften. Als we naar de algoritmelaag gaan, passen we de rekenstroom binnen het model aan om de kenmerken van de gezondheidszorggegevens te benutten. Dit verbetert de ontwerpprestaties op belangrijke aspecten zoals nauwkeurigheid en energie-efficiëntie. Bovendien verfijnen we de modelberekeningen op strategische wijze om de voordelen na de implementatie op CIM-hardware verder te maximaliseren. Op de optimalisatielaag gebruiken we technieken zoals resampling, kwantisering en pruning om de hardwareresourcevereisten te optimaliseren, zonder de algoritmische prestaties van het model in gevaar te brengen.

Na het maken van de neurale netwerkmodellen gaan we over naar de applicatie-onafhankelijke ontwerpfase. Door ons te concentreren op op RRAM gebaseerde memristorapparaten, identificeren we eerst drie belangrijke niet-idealiteiten die een aanzienlijke invloed hebben op de nauwkeurigheid van de gevolgtrekkingen op CIM-hardware. Vervolgens bedenken we mitigatiestrategieën tegen deze niet-idealiteiten, waarbij we de resterende abstractielagen (mapping, microarchitectuur en circuits, en apparaat) omvatten. Op de kaartlaag stellen we een hardwarebewuste trainingsmethodologie voor om de niet-idealiteit van de geleidingsvariatie te bestrijden. Op het niveau van de microarchitectuur presenteren we twee mitigatiestrategieën. De eerste richt zich op de non-idealiteit van G_{min} fouten die niet nul zijn, door middel van een nieuwe benadering van CIM-microarchitectuurontwerp. De tweede introduceert een adaptieve microarchitectuur die de detectieomstandigheden aanpast om de effecten van niet-idealiteit van leesverstoring tegen te gaan. Op apparaatniveau dragen deze strategieën indirect bij door de noodzaak van uitgebreide apparaatengineering te omzeilen, waardoor nauwkeurige gevolgtrekkingen worden gegarandeerd, zelfs als er niet-idealiteiten aanwezig zijn. Voortbouwend op deze basis van modelontwikkeling en niet-idealiteitsmitigatie, integreren we het optimale ECG-classificatiemodel met de voorgestelde mitigatiestrategieën om een CIM edge-AI-prototype te creëren. Zo effenen onze bijdragen de weg naar een toekomst met verbeterde effectiviteit en efficiëntie van door AI aangedreven gezondheidszorg.

Acknowledgements

This thesis marks the end of my Ph.D. journey. I would like to take this opportunity for reflecting back on the path that brought me to this moment. First, I would like to express my sincere gratitude towards my promoter **Prof. Said Hamdioui**, for giving me the opportunity to be a part of his research group. His guidance has been instrumental in shaping my development as a researcher. I am really grateful for his support and for everything I learned from him throughout these years. Next, I would like to thank my co-promoter **Dr. Rajendra Bishnoi**. His constant encouragement and mentorship was the key in keeping me motivated and focused over the course of my doctoral studies. His technical expertise proved invaluable in helping me overcome the challenges in my research on numerous occasions. His belief in me, even during my moments of self-doubt, was a constant source of strength. I am truly grateful to him for consistently bringing out the best in me. Finally, I would like to thank all my colleagues at QCE, my family and friends, and anyone who directly or indirectly helped me in this journey.

Sumit Shaligram Diware
Delft, October 2024

1

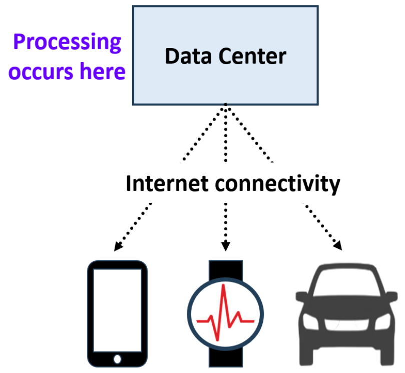
Introduction

1.1. MOTIVATION

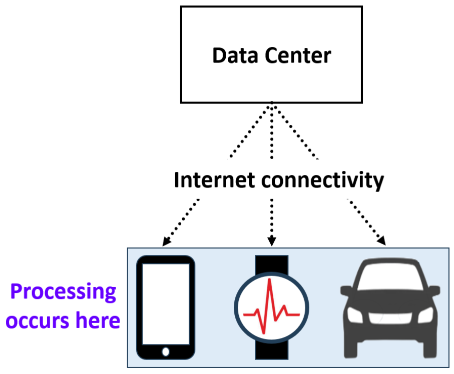
1.1.1. EDGE COMPUTING FOR AI (EDGE-AI)

Recent advancements in artificial intelligence (AI) have paved the way for its seamless integration into various real-world products and services [1]. Cloud computing plays a pivotal role in this integration, providing the users with access to vast computing resources in the data centers to execute AI algorithms. This enables extensive data processing without the need for managing the physical infrastructure and streamlines the deployment of AI-based solutions across diverse sectors like healthcare, finance, automotive etc. However, such cloud-based AI processing (cloud-AI), shown in Figure 1.1a, suffers from several drawbacks [2]. Firstly, it can introduce high response latency due to factors such as network congestion, bandwidth limitations, processing queues for server sharing etc. This poses a severe challenge in scenarios requiring real-time responsiveness. For instance, consider AI-based industrial automation where timely detection and response to safety hazards is paramount. A delay in data processing can lead to late emergency response and increase the risk of accidents. Secondly, continuous data transfer to the cloud can incur substantial network costs. For example, augmented reality applications streaming high-resolution images can consume significant bandwidth and lead to huge network expenses. Thirdly, transmitting sensitive data to remote servers is susceptible to privacy and security concerns. As an example, a breach of financial data can lead to severe consequences like substantial monetary losses and identity theft. Lastly, internet-dependency makes cloud computing susceptible to service disruptions, especially in environments with limited or unreliable network connectivity. This can prove catastrophic in fields like healthcare. For instance, limited internet access can impede AI-based health monitoring devices from providing timely alerts and endanger the user's life.

Edge computing for AI (edge-AI) addresses these challenges by processing the data directly at its source [3], shown in Figure 1.1b. It integrates the data source with on-board edge-AI hardware, which is a specialized chip that executes AI algorithms like neural networks. This effectively overcomes the response latency and network cost issues. It



(a) Cloud-AI: processing occurs in data center.



(b) Edge-AI: processing occurs at the data source.

Figure 1.1: Cloud-AI and edge-AI computing paradigms.

also enhances data privacy as the sensitive information is now maintained and processed locally. Moreover, it bolsters reliability by alleviating reliance on a centralized internet-based infrastructure. The importance of these edge-AI benefits is also reflected in the $14.7 Billion valuation of global edge-AI market in 2022 [4].

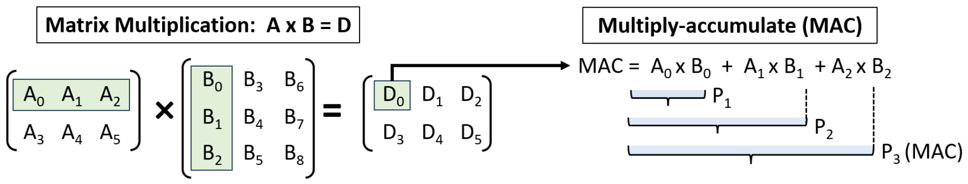
To fully harness these advantages of edge-AI, the on-board hardware must adeptly handle the resource constraints in edge environments. This necessity translates into two key design constraints for edge-AI hardware: i) It should be energy efficient to achieve extended battery life and prolonged uninterrupted operation. ii) It should have a small area footprint to facilitate seamless integration with the data source and enhanced portability. For instance, consider a drone equipped with edge-AI hardware to enhance navigation capabilities in complex environments. If the navigation hardware consumes excessive energy, it would quickly deplete the drone's battery. This would severely curtail its flight time and operational range, negating the benefits of AI-enabled navigation. Conversely, energy-efficient navigation hardware extends flight duration and operational reach. This, combined with improved navigation capabilities, results in comprehensive and superior exploration. Moreover, from an area perspective, a bulky navigation hardware would present a significant challenge for integration with other system components. Its bulkiness would also restrict the drone's range and maneuverability. These issues will essentially nullify the benefits of AI-based navigation. On the other hand, a compact hardware can be easily integrated into the drone system without any adverse effects. With these edge-AI design constraints in mind, the next section explores how conventional hardware struggles to meet them.

1.1.2. LIMITATIONS OF CONVENTIONAL HARDWARE

Conventional hardware designs adhere to von Neumann architecture for computing and use conventional memory technologies for data storage. They are not suited for edge-AI processing due to two main challenges: 1) data transfer bottleneck in von Neumann architecture and 2) limitations of conventional memory technologies, discussed next.

DATA TRANSFER BOTTLENECK

Most edge-AI applications use neural networks because of their ability to learn complex features directly from the data. Matrix multiplications form the computational backbone of neural networks, accounting for 70-90% of the total operations [5]. A matrix multiplication is fundamentally made up of several multiply-accumulate (MAC) operations shown in Figure 1.2. Hence, edge-AI hardware design incorporates optimized dataflow



Matrix Multiplication: A x B = D
$\begin{pmatrix} A_0 & A_1 & A_2 \\ A_3 & A_4 & A_5 \end{pmatrix} \times \begin{pmatrix} B_0 & B_3 & B_6 \\ B_1 & B_4 & B_7 \\ B_2 & B_5 & B_8 \end{pmatrix} = \begin{pmatrix} D_0 & D_1 & D_2 \\ D_3 & D_4 & D_5 \end{pmatrix}$
Multiply-accumulate (MAC)
MAC = $A_0 \times B_0 + A_1 \times B_1 + A_2 \times B_2$
P_1
P_2
P_3 (MAC)

Figure 1.2: Multiply-accumulate (MAC) operation as the fundamental computation unit of matrix multiplication. Notations P_1, P_2 and P_3 denote the partial sums within the MAC operation.

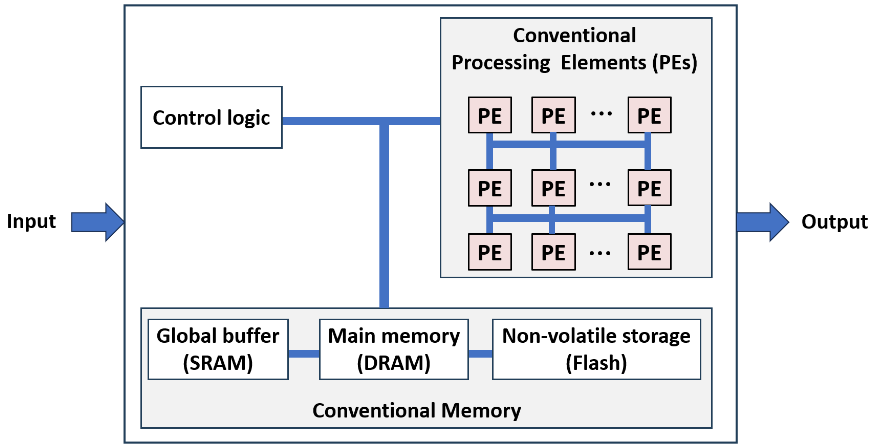
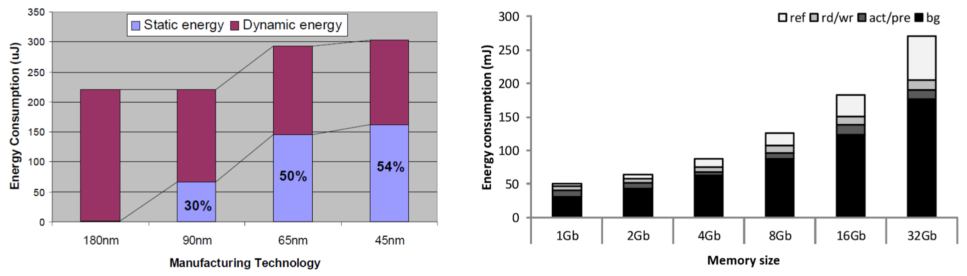


Figure 1.3: Conventional edge-AI hardware architecture based on [6].

for performing MAC operations. A typical example of conventional edge-AI hardware is shown in Figure 1.3. It employs a network of specialized processing elements (PEs) to perform MAC operations. These PEs are connected to a hierarchical memory system consisting of a high-speed global buffer (Static Random Access Memory, SRAM), a larger but slower main memory (Dynamic Random Access Memory, DRAM), and a non-volatile storage (Flash). The global buffer offers the fastest access for frequently used data, while the main memory provides more capacity at the cost of slower retrieval. Finally, the non-volatile storage retains data even after power loss, making it ideal for storing the trained neural network weights. Upon device boot-up, these weights are loaded from non-volatile storage to main memory and then transferred to the global buffer as needed for computations. Moreover, intermediate calculations/outputs also use global buffer and main memory for temporary storage. In this architecture, accessing data from global buffer and main memory consumes 6× and 200× more energy respectively, compared to the energy of a MAC operation [6]. This leads to a substantial amount of energy spending on data movement rather than the actual calculations. This becomes a significant barrier to deploying conventional hardware in energy constrained edge-AI environments.

LIMITATIONS OF CONVENTIONAL MEMORY TECHNOLOGIES

Conventional memory technologies such as SRAM and DRAM struggle to provide energy-efficiency desired for edge-AI. SRAM consumes energy even when it is not actively being accessed, known as static energy. Figure 1.4a shows that SRAM suffers from high static energy consumption, which increases further with each new technology node [7]. Similarly, DRAM wastes a significant portion of energy on non-operational tasks in the form of refresh energy and background energy in Figure 1.4b [8, 9]. Furthermore, these memory technologies also struggle to provide area-efficiency and high storage density required for edge-AI. A single SRAM cell incurs a large area footprint of 100F^2-200F^2 [10]. Although DRAM exhibits a smaller cell size of 8F^2-10F^2 [11], scaling it down to smaller technology nodes presents severe challenge to its reliability [12]. Thus, conventional memory



(a) Static Random Access Memory (SRAM) [7]. (b) Dynamic Random Access Memory (DRAM) [8]

Figure 1.4: Energy consumption breakdown for conventional memory technologies. In subfigure (b), rf: refresh energy, rd/wr: read/write energy, act/pre: activate/precharge energy, and bg: background energy.

technologies are not suited to fulfill the energy-efficiency and storage density needs of edge-AI.

1.1.3. COMPUTATION-IN-MEMORY (CIM)

Computation-in-memory (CIM) presents a promising alternative to overcome the aforementioned limitations of von Neumann architecture and conventional memory technologies. It achieves this by performing in-place computations and using emerging memory technologies known as memristors, as discussed next.

IN-PLACE COMPUTATION

CIM utilizes memristor devices for in-place MAC computations, leveraging their conductance states for data storage as shown in Figure 1.5. Here, operand A is encoded into voltages (V's) and applied to operand B encoded as memristor conductances (G's). The multiplication of A and B is achieved in analog domain through Ohm's law, in the form of currents I_1 and I_2. These currents get accumulated as per Kirchhoff's law to produce current I_{out}, which represents MAC output in analog domain. To obtain the final digital MAC result, I_{out} then undergoes analog-to-digital conversion. Thus, MAC is performed in-place, without fetching operand B out of the memristors and without storing/fetching

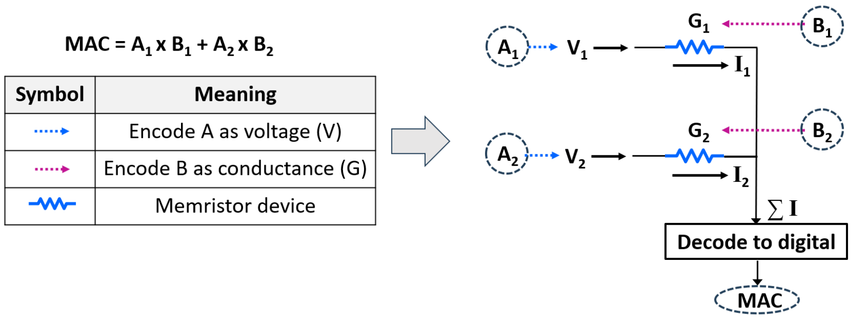


Figure 1.5: In-place multiply-accumulate (MAC) operation in memristor-based CIM.

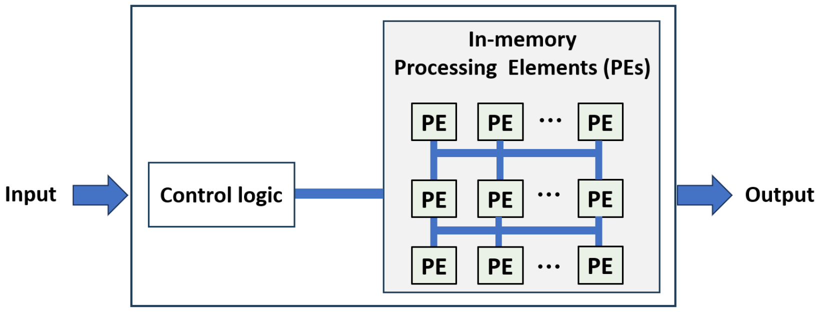


Figure 1.6: Computation-in-memory edge-AI hardware architecture.

the intermediate calculations like partial sums. A CIM-based edge-AI processor consists of in-memory processing elements (PEs) that leverage this in-place MAC computation ability of memristors as shown in Figure 1.6. These PEs store the network weights using an array of memristor devices and directly return the final MAC result via in-situ computations within the memory array. This eliminates the memory hierarchy, significantly reduces the data movement and alleviates the data transfer bottleneck.

Memristor Technology

Memristors offer non-volatile data storage, where they retain information without a power supply by virtue of of data storage as conductance instead of charge. As a result, memristors do not suffer from standby energy consumption like SRAMs (static energy) and DRAMs (refresh energy and background energy). Thus, memristors provide better energy-efficiency compared to conventional memory technologies [13]. Moreover, memristors are capable of achieving a compact cell size of $4F^2$ and a single memristor device can store multiple bits by exhibiting intermediate conductance levels. This enables memristors to achieve better area efficiency and storage density than conventional memory technologies [10]. Additionally, memristor devices are highly scalable and directly compatible with CMOS fabrication process [14]. Hence, memristors present a promising alternative to conventional memory technologies for energy-efficient and compact edge-AI hardware [15].

1.2. Research Topics

Having identified the benefits of CIM for edge-AI, we now delve into the design process of CIM-based edge-AI hardware. It encompasses the abstraction layer stack shown in Figure 1.7. Each layer within this stack serves a distinct purpose, described as follows:

- **Application layer**: It involves comprehensive assessment of functionality, requirements and practical implications of the given application to define the design specifications.
- **Algorithm layer**: It first determines the high-level computational structure to achieve efficient and effective data processing for the given the design specifications.

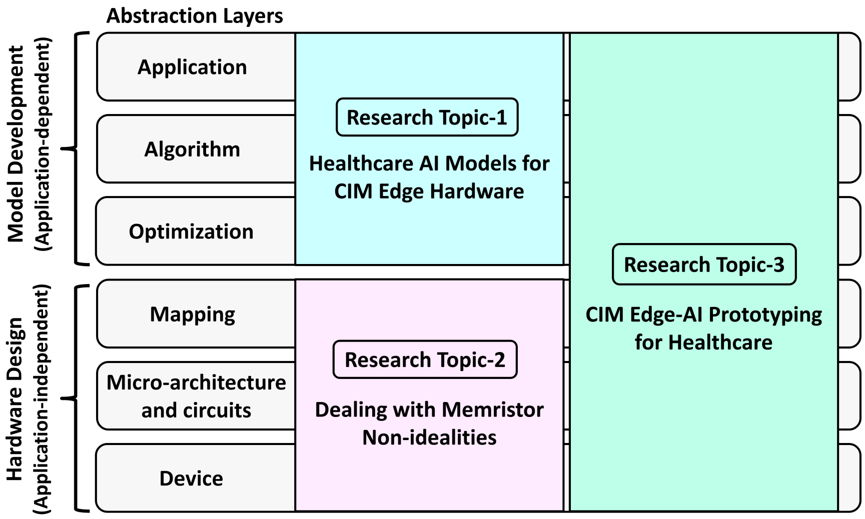

Abstraction Layers
Model Development (Application-dependent)
Hardware Design (Application-independent)
Application
Algorithm
Optimization
Mapping
Micro-architecture and circuits
Device
Research Topic-1
Healthcare AI Models for CIM Edge Hardware
Research Topic-2
Dealing with Memristor Non-idealities
Research Topic-3
CIM Edge-AI Prototyping for Healthcare

Figure 1.7: Research topics addressed in this thesis and their distribution across various CIM abstraction levels.

It then determines optimal AI algorithm for each computational block within this structure through design-space exploration.

- **Optimization layer**: It streamlines the AI model computations without compromising its functionality. This leads to reduced hardware resource utilization when such model is deployed for on-field processing.
- **Mapping layer**: It translates the AI model parameters and structure into functional unit configurations and control sequences on the hardware.
- **Micro-architecture and circuits layer**: It encompasses two key aspects: i) micro-architecture, which governs the dataflow within hardware functional units. and ii) circuits, which focuses on implementing the micro-architecture using fundamental electronic components such as transistors.
- **Device layer**: It includes development of fundamental electronic components like transistors, memristors etc. and engineering their physical properties.

The first three abstraction layers (application, algorithm, and optimization) are heavily influenced by the specific application for which CIM hardware is being designed. Hence, they represent an application-dependent design phase, where a customized neural network model is created for the given application. The core challenge in this phase is to develop a neural network model that: i) delivers strong algorithmic performance for the application, and ii) incorporates features to maximize the advantage of CIM hardware for the application. In contrast, the remaining abstraction layers (mapping, micro-architecture and circuits, and devices) constitute an application-independent design phase. This is because they focusing solely on translating model computations into executable operations on CIM hardware. The accuracy of these operations can be compromised by the non-ideal characteristics of memristor devices, which introduce computational errors. Therefore, dealing with memristor non-idealities becomes the primary challenge in this phase.

This thesis explores the challenges in both the aforementioned design phases, with a focus on healthcare applications and RRAM-based memristor devices. Moreover, it also puts emphasis on realizing the presented ideas and solutions through prototyping. This leads to the following three research topics: 1) healthcare AI models for CIM-based edge hardware, 2) dealing with RRAM non-idealities, and 3) CIM edge-AI prototyping for healthcare. We will now discuss them in detail.

1.2.1. Healthcare AI Models for CIM Edge Hardware

Healthcare domain stands out as a prime target for AI integration due to its direct and significant impact on human well-being. AI is already cementing its role in healthcare applications, encompassing crucial functions like health monitoring and diagnostics [16]. CIM-based edge-AI hardware can further enhance this integration by providing fast, reliable, and secure processing of medical data at the source, enabling prompt and effective healthcare interventions.

Developing neural network model for healthcare application aimed at deployment on CIM-based edge hardware presents several research opportunities. First, we can introduce new features in the model tailored to the specific healthcare application. This can enhance its real-world impact by addressing the medical needs more effectively. Second, dataflow within the model can be customized to exploit the unique characteristics of the healthcare application. This can improve its performance in key aspects like accuracy and energy efficiency. Third, strategically refining the model computations and dataflow can further maximize post-deployment benefits on CIM hardware. Last, techniques like quantization and pruning can be employed to optimize hardware resource requirements, without compromising the model's algorithmic performance. This thesis explores the above research opportunities in the context of following two healthcare applications:

- **Cardiac arrhythmia classification:** Heart-related disorders, known as cardiovascular diseases (CVDs), are one of the major causes of death globally [17]. Early diagnosis of CVDs can facilitate timely treatment to mitigate the health risks. This can be achieved by identifying abnormal heart activity, known as arrhythmia. Edge-AI can facilitate arrhythmia identification through wearable healthcare devices. These devices acquire and monitor heart activity via electrocardiogram (ECG) signals. These ECG signals are then classified by a neural network model into various arrhythmia types (classes). Our goal is to develop a neural network model customized for performing arrhythmia classification on CIM-based edge-AI hardware.

- **Diabetic retinopathy screening:** Diabetic retinopathy (DR) refers to irreversible retinal damage caused by elevated glucose levels and blood pressure. It is a leading cause of permanent vision impairment across the globe [18]. Moreover, every diabetic person is susceptible to the development of DR [19]. Regular screening for DR is necessary to detect it at an early stage and facilitate timely treatment to prevent further retinal damage. Edge-AI can achieve such screening in a fast, efficient, and convenient manner by employing neural networks to categorize retinal images into distinct screening classes. We aim to create a neural network model for DR screening, targeting deployment on CIM-based edge-AI hardware.

1.2.2. Dealing with Memristor Non-idealities

Using a developed neural network model to execute real-world tasks (e.g. image classification) with on-field data is referred to as inference. When a neural network model is deployed on CIM-based edge-AI hardware for inference, its weights are stored in the hardware using memristor conductances (G's) as shown in Figure 1.8. The memristor devices exhibit certain characteristics called non-idealities, which lead to deviation (ΔG's) from their expected (ideal) conductance behavior (G's). This gives rise to deviation (ΔI's) from their ideal current contributions (I's). These current deviations then get accumulated via Kirchhoffs law and introduce errors in the MAC output. Such erroneous computations can significantly diminish inference accuracy of CIM hardware.

In this thesis, we focus on RRAM-based memristor devices and identify their three key non-idealities which significantly affect the inference accuracy of CIM hardware. These non-idealities are depicted in Fig. 1.8 and described as follows:

- **Non-zero G_{min} Error**: Ideally, a memristor with zero conductance should be used in CIM to represent a zero weight in the neural network. However, real-world memristor devices exhibit a non-zero minimum conductance (G_{min}). Consequently, zero weights are represented with G_{min} conductance in CIM hardware. When an input voltage (non-zero input) is applied to a G_{min} memristor (zero weight), it yields

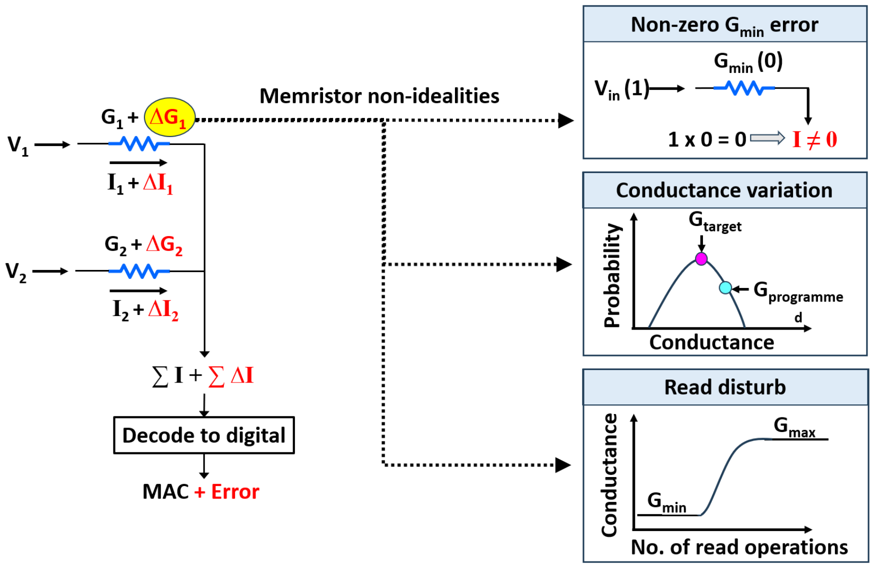


Figure 1.8: Illustration of the impact of memristor non-idealities on computational accuracy of CIM. We focus on three key non-idealities of RRAM-based memristors: i) non-zero G_{min} error, ii) conductance variation, and iii) read-disturb.

a non-zero output current. Such outcome contradicts the mathematical fact that multiplying a non-zero input by a zero weight should produce zero output. This inconsistency is termed as non-zero G_{min} error.

- **Conductance Variation**: The programmed conductance of a memristor deviates from its target value, due to the stochastic nature of device physics and fabrication imperfections [20]. This phenomenon is called conductance variation.
- **Read-disturb**: Read operations on memristor are carried out using low voltage to prevent any disturbance to its conductance state. However, unintended minuscule change in conductance can still occur despite this precaution [21, 22]. The accumulation of these small changes into a substantial conductance change over many read operations is known as read-disturb [23].

Dealing with these non-idealities is critical to uphold the practical value of CIM benefits in real-world scenarios. This is because energy-efficient computations serve no practical purpose if they are functionally incorrect. Moreover, we should ensure that non-ideality mitigation solutions do not introduce excessive overheads and undermine the advantages of CIM hardware. In this thesis, we aim to develop strategies that mitigate the impact of non-idealities while incurring minimal overheads, thereby facilitating accurate and energy-efficient CIM-based edge-AI design.

1.2.3. CIM EDGE-AI PROTOTYPING FOR HEALTHCARE

The groundwork for CIM-based edge-AI design for healthcare is established through the previous two research topics. The first topic explored the creation of AI models tailored for healthcare applications, while ensuring their efficient execution on CIM hardware. The second topic addressed the challenge of mitigating memristor non-idealities, to achieve error-free inference on CIM hardware. Building upon this foundation, our research now arrives at the final objective: prototyping of healthcare edge-AI solutions. This involves integrating the optimal AI models and non-ideality mitigation schemes from the first two research topics, into a hardware prototype using the ASIC design flow. In this thesis, our goal is to design a small-scale CIM edge-AI prototype for a healthcare application, incorporating our optimal AI model and non-ideality mitigation schemes.

1.3. THESIS CONTRIBUTIONS

We adopt a cross-layer approach in this thesis to address the aforementioned research topics, covering the entire CIM abstraction stack as shown in Figure 1.9. We will now describe our contributions in detail.

Contributions to healthcare AI models for CIM edge hardware:

- **Memristor-based CIM for ECG arrhythmia classification:** We present a severity-based, accurate, and energy-efficient ECG arrhythmia classifier, targeting deployment on CIM-based edge devices. Initially, we assess the severity impact of various arrhythmia classes and evaluate their implications for both end-users as well as

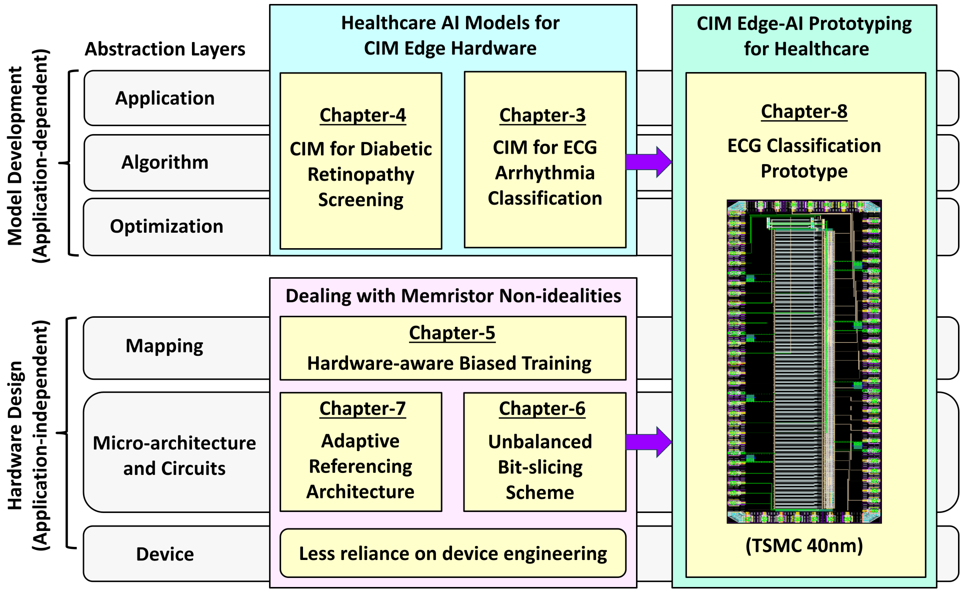

Model Development (Application-dependent)
Hardware Design (Application-independent)
Abstraction Layers
Application
Algorithm
Optimization
Mapping
Micro-architecture and Circuits
Device
Healthcare AI Models for CIM Edge Hardware
Chapter-4
CIM for Diabetic Retinopathy Screening
Chapter-3
CIM for ECG Arrhythmia Classification
Dealing with Memristor Non-idealities
Chapter-5
Hardware-aware Biased Training
Chapter-7
Adaptive Referencing Architecture
Chapter-6
Unbalanced Bit-slicing Scheme
Less reliance on device engineering
CIM Edge-AI Prototyping for Healthcare
Chapter-8
ECG Classification Prototype
(TSMC 40nm)

Figure 1.9: Thesis contributions (in yellow color) across the entire CIM abstraction stack covering three research topics. Each contribution is also annotated with its corresponding chapter number in the thesis.

medical professionals. Based on this analysis we develop a severity-based classification approach which serves two key purposes. First, it improves assistance offered by model to both end-users and medical professionals. Second, it decomposes the overall ECG classification task into a hierarchical arrangement of sub-classifiers, each managing simpler sub-tasks. This hierarchical classification structure enhances energy efficiency by selectively activating only the necessary sub-classifier for each input. It also achieves high accuracy as each sub-classifier deals with only a subset of total arrhythmia classes. Moreover, we perform design-space exploration to identify the optimal neural network topology for each sub-classifier, which provides the best balance between energy efficiency and accuracy. As a result, energy efficiency of the hierarchical classifier is improved while preserving its accuracy. This work was published in [24].

- **Memristor-based CIM for Diabetic Retinopathy Screening:** We propose a reliable and energy-efficient classifier for DR screening on CIM-based edge devices. We first analyze practical implications of training data quality and diagnostic information availability on neural network-based DR screening. To alleviate the reliability concerns due to training data quality, we create a custom training dataset which enables the model to effectively handle on-field data variations and minority classes. Moreover, we introduce a pseudo-binary classification scheme which couples multiclass DR classification with a decision-making logic to produce binary screening outcomes. This offers two key advantages. First, it extracts valuable diagnostic information by refining the multiclass classifier's internal outputs. Second, it bolsters the model reliability by improving its classification performance. This is achieved by considering the cumulative probabilities of various DR classes within each screening category, leading to more informed decisions. We then develop a pseudo binary DR screening model using our custom training dataset, through design-space exploration across two neural network architectures: Inception-V3 and DenseNet-121. Subsequently, we optimize the final model for CIM-based edge deployment through pruning and quantization. This work was published in [25].

Contributions to dealing with memristor non-idealities:

- **Hardware-aware Biased Training:** This training methodology mitigates the impact of conductance variation non-ideality. First, we analyze memristor conductance states to identify those inherently more immune to variation, called favorable states. We then establish a favorability constraint, which defines a range of weight values that directly get mapped to favorable states. Following this, we employ a two-stage training process. In the initial stage, we train the model in a hardware-unaware manner to establish a baseline. We then analyze the trained model to identify the weights important for CIM hardware accuracy. In the next stage, we retrain this model while enforcing the favorability constraint on important weights. Consequently, post-retraining values of important weights directly map to favorable states. As a result, execution of important calculations becomes error-free, leading to high inference accuracy on CIM hardware. This work was published in [26].

- **Unbalanced Bit-slicing Scheme:** This approach to CIM micro-architecture design effectively mitigates non-zero G_{min} error non-ideality. It involves two key elements. First, the memristor array is designed with a higher sensing margin allocation for most significant bits (MSBs). This directly combats the detrimental effects of non-zero G_{min} error on these crucial bits. Second, the digital post-processing circuitry utilizes 2's complement arithmetic, whose differential nature further minimizes the influence of non-zero G_{min} error. While this approach yields superior accuracy, it also incurs energy overhead due to extra sensing margin allocated to MSBs. To address this, we minimize hardware requirements by adjusting margin allocation for less critical bits, while maintaining high margins for MSBs. This reduces energy consumption without sacrificing the gains in accuracy. Moreover, it facilitates a trade-off between accuracy and energy efficiency. By exploring this trade-off, we can prioritize accuracy or energy efficiency as per the specific requirements of the application. This work was published in [27, 28].

- **Adaptive Referencing Architecture:** This CIM micro-architecture mitigates the impact of read-disturb non-ideality. Its design commences with an analysis to extract key insights into the read-disturb phenomenon. The findings from this analysis are then used to develop two architectural components: i) read-disturb detection unit ii) adaptive ADC equipped with control logic. The read-disturb detection unit identifies instances of read-disturb during operation. It achieves this by monitoring the most vulnerable column in the memristor array, that has been identified through pre-mapping profiling. Once the read-disturb event is detected, the adaptive ADC dynamically adjusts sensing conditions to counteract the effect of read-disturb. Thus, error-free operation is restored, resulting in high inference accuracy on CIM hardware. This work was published in [29].

- Besides mitigating the impact of non-idealities, these contributions also alleviate the need for device engineering to enhance memristor device characteristics. This paves the way for capitalizing on the energy-efficiency benefits of CIM, without relying on advancements in memristor device technology..

Contribution to CIM edge-AI prototyping for healthcare:

- **ECG Classification Prototype:** Our goal is to integrate the proposed ECG classification model and non-ideality mitigation strategies into a prototype. We select the first hierarchical level in the ECG classification model for prototyping, due to its significant impact of on human well-being. Moreover, we choose unbalanced bit-slicing for prototyping among the proposed non-ideality mitigation strategies. This is because we use polysilicon-based resistive storage due to memristor fabrication and integration unavailability, which only suffers from non-zero G_{min} error. We develop the prototype incorporating these choices through a three-phase process. First, we further optimize the model through resampling and quantization to reduce hardware resource usage. Second, we design a system architecture comprising of analog vector-matrix multiplication (VMM) units and digital processing logic. The analog VMM includes a novel CIM crossbar and a new ADC design (both out of the scope of this thesis). The digital processing logic is tailored to efficiently handle the outputs of the analog VMM units. Last, we implement the system-on-chip (SoC) layout for this architecture in TSMC 40nm technology. Its analog parts are implemented through custom layouts, while the layouts for digital parts are generated by standard cell-based physical design. These parts are integrated using analog-on-top flow and sent to the foundry for fabrication. Upon receiving the fabricated prototype, we create a testbench for its measurement and characterization. This evaluation phase is currently ongoing.

1.4. THESIS ORGANIZATION

The rest of this thesis consists of eight chapters that are grouped into five parts. We now present an overview of this organization structure.

PART-I Background (Chapter 2)

This part provides the fundamental knowledge necessary for understanding this thesis. Its constituent chapter is described next.

- Chapter 2 presents the fundamentals of neural networks and CIM. It begins by introducing basic terminologies in artificial intelligence. This is followed by a discussion on neural networks, covering their types and operational modes. We then delve into the CIM system architecture, focusing on its role in vector-matrix multiplication which is the dominant operation in neural networks. Finally, we explore different memristive crossbar array designs and memristor device technologies.

PART-II Healthcare AI Models for CIM Edge Hardware (Chapters 3 and 4)

This part focuses on model development for CIM-based edge-AI in healthcare domain. It includes the following chapters.

- Chapter 3 presents an ECG arrhythmia classifier for CIM-based edge-AI. We begin with an introduction that outlines the motivation, discusses state-of-the-art, and highlights our contributions. Following the introduction, we provide a concise overview of cardiac arrhythmia and its classification. We then describe the details of our proposed methodology. It leverages a severity-based classification approach specifically tailored for arrhythmia classification. This approach allows for the creation of multiple hierarchical classification architectures. We then outline the criteria for selecting the optimal architecture from these options. Furthermore, we provide a design-space exploration technique to efficiently design the network components within the chosen architecture, ensuring high accuracy while minimizing energy consumption. Later, we describe details of the simulation setup and present simulation results to demonstrate the effectiveness of our classifier. Finally, the conclusion section summarizes the key findings and insights.

- Chapter 4 proposes a diabetic retinopathy (DR) screening classifier tailored for deployment on CIM-based edge devices. We start with an introduction that covers the motivation, related work, and our key contributions. The background section then provides essential context on DR as a health condition and its screening process. We then delve into the details of our proposed methodology. This starts with an overview to facilitate a high-level understanding of our approach. Next, we focus on reliable model development, which includes custom dataset creation and a pseudo binary classification scheme. Furthermore, we explore model optimization through pruning and quantization to improve energy efficiency upon deployment on CIM hardware. Afterwards, we describe the simulation setup and present simulation results. Last, the conclusion section ends the chapter with a summary of key insights.

PART-III Dealing with Memristor Non-idealities (Chapters 5, 6 and 7)

This part presents solutions for dealing with memristor non-idealities, to improve the computational accuracy of CIM hardware. It consists of the following chapters.

- Chapter 5 describes the biased training method aimed at addressing conductance variation non-ideality. We commence with an introduction section, offering insights into motivation, prior art, and our contributions. Next, we present details of our proposed methodology. It starts with a high-level overview, followed by an analysis to determine the favorable conductance states. This analysis is subsequently used for developing the biased training algorithm. We then detail the simulation setup and present the simulation results. The chapter ends with a summary in conclusion section.

- Chapter 6 introduces an unbalanced bit-slicing approach for CIM micro-architecture design, that mitigates the impact of non-zero G_{min} error non-ideality. First, an introduction section describes the motivation, state-of-the-art, and our key contributions. Next, we dive into the details of the proposed methodology. It starts with a general overview of our approach. This is followed by a detailed explanation of two distinct variants of unbalanced bit-slicing. One variant prioritizes high accuracy, while the other explores a trade-off between accuracy and energy efficiency. Later, we describe the simulation setup and present the simulation results. We finally summarize important findings and takeaways in the conclusion section.

- Chapter 7 presents an adaptive referencing architecture for mitigating read-disturb non-ideality. We first include an introductory section that describes the motivation, discusses the state-of-the-art, and outlines our contributions. We then describe our proposed architecture in detail. This description begins with an overview of the proposed referencing approach. We then describe the design of two crucial architectural components: the read disturb detection unit which identifies read-disturb events, and the adaptive ADC which dynamically adjusts its referencing mechanism to counteract read-disturb errors. Next, we provide simulation setup details and present the simulation results. Lastly, the conclusion section offers a summary of the chapter.

PART-IV CIM Edge-AI Prototyping for Healthcare (Chapter 8)

This part focuses on prototype design by integrating the optimal AI model and non-ideality mitigation solutions. It comprises of the following chapter.

- Chapter 8 provides details of the ECG classification prototype. We begin with a brief introduction that covers the intuitions behind our design choices and an overview of our three-phase prototyping approach. We then describe the first phase, which optimizes the model for reducing hardware resource usage. This is followed by the details of second phase, where we design the system architecture. We then present the third phase, which involves layout implementation for the system architecture. Finally, we provide the details of PCB and testbench design for evaluating the fabricated prototype.

PART-V Conclusions (Chapter 9)

This is the final part which concludes this thesis. It contains the following chapter.

- Chapter 9 provides summary of all the chapters in the thesis and also offers insights into future research directions.

PART-I

Background

2
Fundamentals of Neural Networks and Computation-In-Memory

2.1. INTRODUCTION TO NEURAL NETWORKS

Figure 2.1 depicts relationship between three key terms used in the context of developing cognitive machines: artificial intelligence, machine learning, and neural networks. They can be distinguished from each other as follows:

- **Artificial intelligence (AI):** It is a broad field focusing on building intelligent machines, that can perform cognitive tasks like humans.
- **Machine learning (ML):** It is a sub-field of AI that enables machines to autonomously learn from the data without explicit programming.
- **Neural networks (NN):** It is a sub-field of ML which draws inspiration from the workings of human brain.

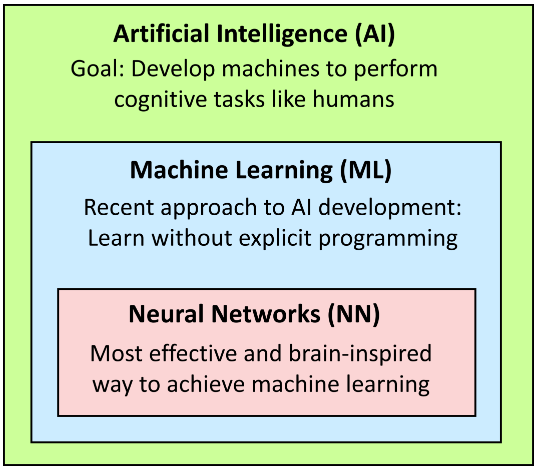

Artificial Intelligence (AI)
Goal: Develop machines to perform cognitive tasks like humans
Machine Learning (ML)
Recent approach to AI development: Learn without explicit programming
Neural Networks (NN)
Most effective and brain-inspired way to achieve machine learning

Figure 2.1: The relationship between three fundamental AI terminologies.

In today's world, neural networks stand as the most prevalent method for creating intelligent machines due to their exceptional capacity for cognitive processing and adaptability. At their core, neural networks operate by emulating the computational structure of the brain. The human brain achieves cognitive abilities through a network of specialized cells called neurons and their interconnection called synapses. Neurons are the brain's computational units and communicate with each other through synapses. The synapses act like weighted connections and modulate the biological signals passed between neurons. The computational neural networks mimic this brain structure, employing neuron functions interconnected by weighted connections (synapses) as shown in Figure 2.2. The neurons perform weighted sum of the inputs coming from input synapses. The subsequent outputs undergo non-linear transformations to enhance the network's cognitive capability. This emulation empowers neural networks to autonomously extract features from raw data and effectively handle complex relationships within data. Hence, they have demonstrated remarkable performance across a diverse array of real-world cognitive tasks, leading to their widespread adoption in modern AI applications.

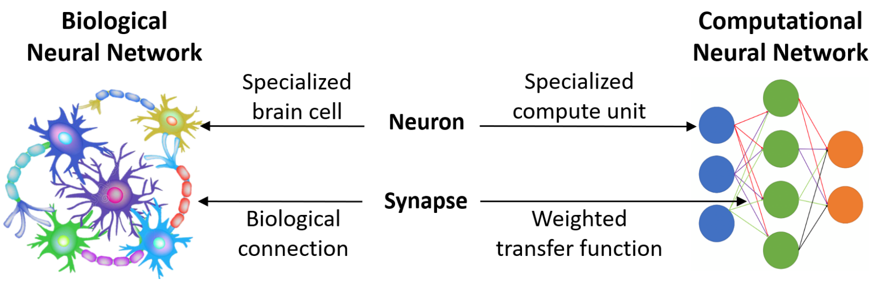


Figure 2.2: Biological neural network as the inspiration for computational neural network.

2.2. Types of Neural Networks

Achieving optimal performance for a given application typically requires using appropriate type of neural network suited for its characteristics. We will now describe the different types of neural network used within the scope of this thesis.

2.2.1. Fully-connected Neural Network

The most basic type of neural network is the fully-connected network shown in Figure 2.3. It consists of an input layer, one or more hidden layers and an output layer. All neurons in a layer are connected to all neurons in the next layer, hence the name fully connected network. The number of neurons in the input layer is determined by the dimensions of the input data. These input neurons receive the input data and transfer it to the first hidden layer. The hidden layers remain concealed from both the input and output of the network. Hidden layer neurons compute the weighted sum of inputs from the preceding layer, then apply a nonlinear activation function and pass the result to next layer. Output of the last hidden layer is passed onto the output layer. The number of neurons in the output layer is equal to the number of prediction classes. Weighted sum outputs of this layer undergo softmax operation to produce prediction probabilities for each class.

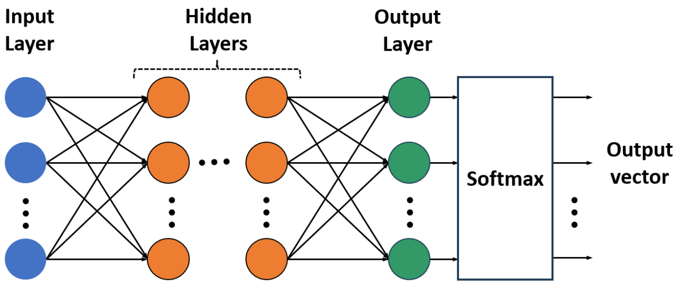


Figure 2.3: Fully-connected neural network

Fully connected neural network struggles to capture spatial or temporal information effectively due to its simple structure and computations. Hence, there exists a need for specialized neural networks that can more adeptly handle complex cognitive tasks like image processing, sequence learning etc. We will discuss such specialized networks in the upcoming subsections.

2.2.2. CONVOLUTIONAL NEURAL NETWORK

Convolutional neural network (CNN) is a specialized neural network primarily designed for image processing tasks. It excels at capturing spatial patterns in the images through convolution operations. CNN comprises of several stacks of convolution and pooling layers, followed by one or more fully connected layers as shown in Figure 2.4. Convolution layers apply a series of filters to the inputs and extract features at different spatial locations. These features are then passed through pooling layers, which compress the feature maps while retaining essential information. Finally, the extracted feature maps are flattened and fed into fully connected layers for classification. The effectiveness of CNNs stems from their hierarchical learning architecture. Early convolution layers learn fundamental features like edges and textures, which are progressively combined by subsequent layers to recognize increasingly complex patterns. This hierarchical approach allows CNNs to develop a sophisticated understanding of images, transitioning from low-level features to high-level concepts. This ability is a key factor behind their widespread adoption and success in computer vision tasks.

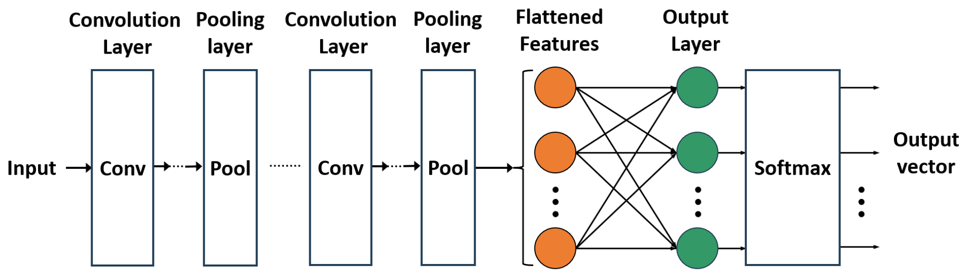


Figure 2.4: Convolutional neural network (CNN).

2.2.3. RECURRENT NEURAL NETWORK

Recurrent neural network (RNN) is a specialized neural network aimed at processing sequential data such as time series. It possess the ability to retain information about the previous inputs and uses this context for processing new sequential data. Long short-term memory (LSTM) network is a variant of RNN, that is widely used due to its effectiveness at capturing long-term dependencies in sequential data. A typical LSTM structure consists of an input layer, one or more hidden layers with LSTM cells as neurons and an output layer, as shown in Figure 2.5. The internal structure of an LSTM cell is depicted in Figure 2.6. It uses two memory variables: i) the cell state (C_t) which represents long term memory and ii) the hidden state (h_t) which represents short term memory. These memory variables are regulated through four gating functions, described as follows:

- Forget gate: It controls which existing part of previous information in the cell state

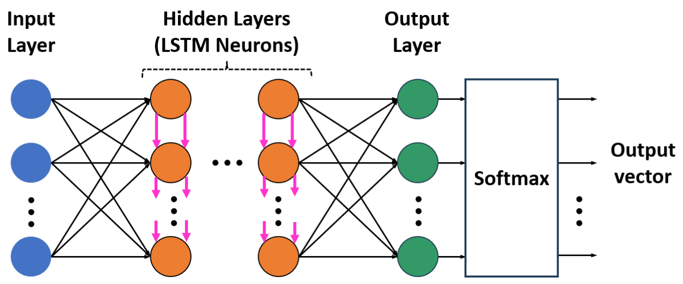


Figure 2.5: Long short-term memory (LSTM) neural network.

should be discarded.

- Input gate: It determines which new information should be incorporated in the cell state.
- Cell state update: It updates the cell state based on inputs from the forget gate and the input gate.
- Output gate: It determines the new hidden state based on the updated cell state.

These gating mechanisms capture long-term dependencies by selectively acquiring useful information and discarding irrelevant details over time.

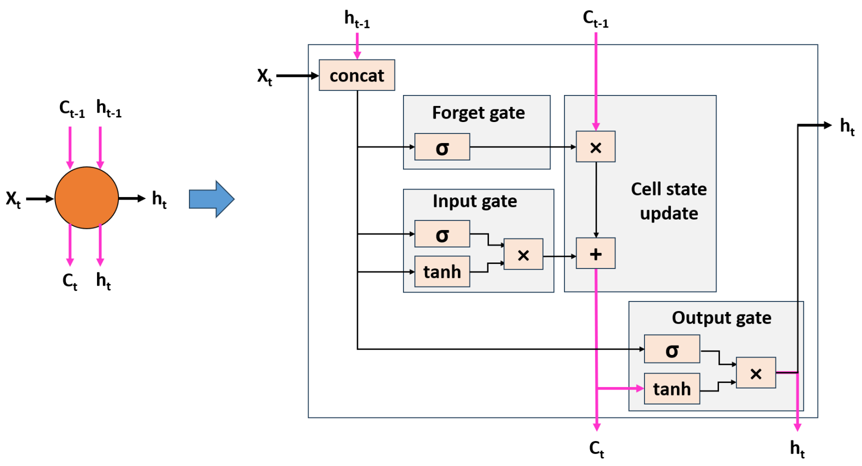


Figure 2.6: Long short-term memory (LSTM) neuron (also called LSTM cell) with X_t as the input. State inputs coming from previous LSTM neuron in the same layer are indicated by C_{t-1} and h_{t-1}. The outputs are given by C_t and h_t. Notations concat and σ denote concatenation and sigmoid function respectively.

Although LSTMs excel at using past information for predictions, many tasks like sentiment analysis require context from both past and future information. Bidirectional

LSTM (BLSTM) overcomes this issue by employing a combination of two LSTM layers as shown in Figure 2.7. Here, one LSTM layer processes the input sequence in the forward direction (past to future), while the other processes the sequence in the reverse direction (future to past). The final prediction is made by combining the outputs of both forward and backward LSTMs.

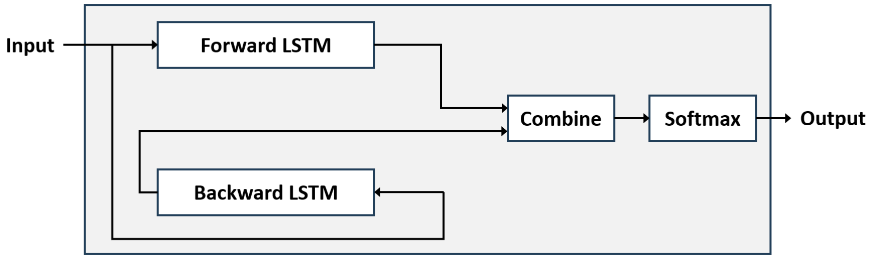


Figure 2.7: Bidirectional long short-term memory (BLSTM) neural network.

2.2.4. Temporal Convolutional Network

Temporal convolutional network (TCN) offers a compelling alternative to LSTM for processing sequential data with very long dependencies. It achieves this through dilated convolutions, which capture extensive temporal contexts without exponentially increasing the number of parameters. Additionally, TCN is better suited for hardware acceleration compared to LSTM. This is because its core operations are highly parallelizable, while LSTM relies on sequential computations. This advantage also becomes particularly significant when dealing with very long sequences.

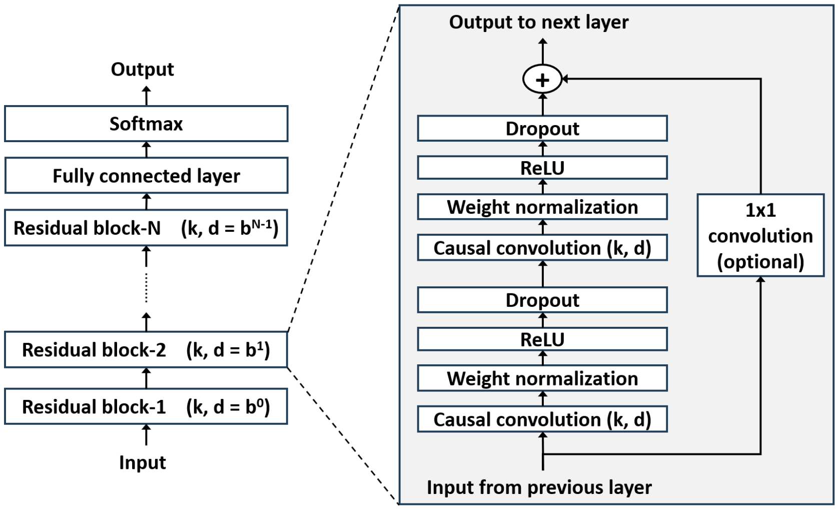

Output
Softmax
Fully connected layer
Residual block-N (k, d = b^{N-1})
Residual block-2 (k, d = b^1)
Residual block-1 (k, d = b^0)
Input
Output to next layer
+
Dropout
ReLU
Weight normalization
Causal convolution (k, d)
Dropout
ReLU
Weight normalization
Causal convolution (k, d)
1x1 convolution (optional)
Input from previous layer

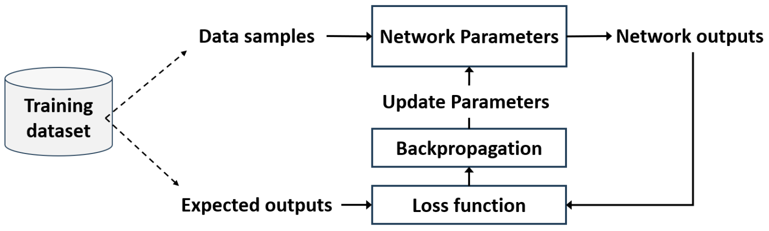
Figure 2.8: Temporal convolutional network (TCN).

A typical TCN consists of multiple layers of dilated convolutions organized into residual blocks as shown in Figure 2.8. Each residual block is characterized by kernel size (k) and dilation rate (d). The dilation rate controls the spacing between elements considered by the filter, allowing the model to increase the receptive field without extra parameters. The residual block employs causal convolutions. This ensures that only past and current information is considered during filtering to preserve the sequence order. Additionally, an optional 1×1 convolution adjusts feature map dimensions before aggregation. The dilation rate is exponentially increased across the residual blocks to progressively captures larger temporal contexts. Lastly, one or more fully connected layers followed by a softmax function are used to generate the prediction probabilities.

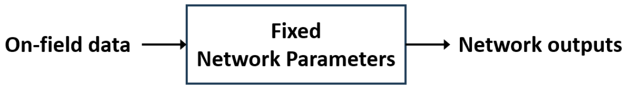
The choice between TCN and LSTM depends on the specific task and characteristics of the data. LSTM can be a preferred for handling short sequences. Additionally, its simpler architecture can facilitate faster development cycle. On the other hand, TCN is well suited for dealing with very long sequences. Also, TCN can be preferred over LSTM for projects that allow ample development time for meticulous design and optimization

2.3. Operational Stages of Neural Networks

Now that we have covered various neural network types, let us delve into the two stages of their operation: training and inference, shown in Figure 2.9. We will discuss both these stages from an algorithmic perspective as well as a hardware design perspective.



(a) Neural network training.



(b) Neural network inference.

Figure 2.9: Neural network operation stages.

2.3.1. Neural Network Training

The human brain contains a network of neurons and synapses, which forms its structural foundation. However, this structure becomes adept at cognitive tasks only after it undergoes a process called learning. The learning process involves modulation of synaptic connections, adjusting their strengths in response to the learning stimuli. Thus, it imparts cognitive abilities to the brain for processing the pertinent information, while preserving the brain's structural foundation. In a similar manner, computational neural networks have a foundational structure comprising of neuron units and weighted connections. This structure just provides the framework for acquiring the knowledge necessary to perform cognitive tasks. The actual development of cognitive abilities occurs through a process known as neural network training. It involves adjusting the neural network weights in response to training data, while keeping its underlying structure unchanged.

The neural network training process is depicted in Figure 2.9a. It begins with collecting the training dataset, consisting of data samples and corresponding expected outputs. The network then processes the training data samples and produces outputs. The extent of mismatch between network outputs and expected outputs is quantified using a loss function. After loss function calculation, the network undergoes a process called backpropagation. It involves calculating the gradient of loss function with respect to network weights. This gradient indicates how much each weight should be changed to minimize the loss function and bring the network output closer to the expected output. The network then continuously fine-tunes its weights across several iterations through backpropagation. With each iteration, the network outputs get closer to the desired outputs, improving its overall accuracy. The training process ends once the desired level of accuracy is attained. As training typically involves large training datasets and iterative weight update calculations, it requires extensive computing resources and can also extend over a long period of time. Hence, it is typically carried out in cloud environments due to their scalability and abundance of hardware resources.

2.3.2. NEURAL NETWORK INFERENCE

After completing the training stage, the neural network is deployed on-field to perform cognitive tasks using its acquired knowledge. This operational stage is known as inference. It involves using the trained (fixed) weights of the network to process new on-field data as shown in Figure 2.9b. For instance, consider an image recognition system performing inference. It feeds an input image into its neural network. The network then calculates a vector containing probabilities of the image belonging to various predefined classes. The class with the highest score becomes the prediction output for the given image.

Inference can take place either in the cloud or on edge devices. Nowadays, there is a growing preference for performing inference on edge devices. This is driven by several benefits of edge processing such as reduced latency, improved privacy, and better reliability. For example, consider real-time applications like autonomous vehicle or drone navigation. In these scenarios, edge inference can provide faster response by eliminating the communication delays with the cloud. Additionally, its local data processing can mitigate the data privacy and reliability risks associated with cloud infrastructure. Moreover, even for non real-time services like Apple Siri where cloud reliance is not a major issue, transitioning to edge inference can still yield benefits in terms of privacy and security. However, limited energy resources on edge devices pose a significant challenge in designing hardware for neural network inference on such platforms. This presents an interesting research opportunity from hardware design perspective.

2.4. Computation-In-Memory for Neural Networks

Memsristor-based computation-in-memory (CIM) has emerged as a promising computing paradigm, to achieve the energy efficiency desired for neural network inference on edge devices. We will now delve into the fundamentals of CIM, adopting a top-down approach. We begin with the CIM system architecture, followed by the details of CIM vector processing unit. We then explore the design of the memristor crossbar array and conclude with a discussion on memristor device technologies.

2.4.1. CIM System Architecture

A typical CIM system architecture leverages a tiled structure to exploit the inherent parallelism in neural network computations. Such architecture is depicted in Figure 2.10. It consists of tiles, functional units, global buffer, and global control logic. Tiles are responsible for performing vector-matrix multiplication (VMM) operations. Functional units perform other computations such as summing up partial outputs from tiles, pooling, or activation function. Global buffer serves as a storage for input/output feature maps, while global control logic governs the interactions among all system components.

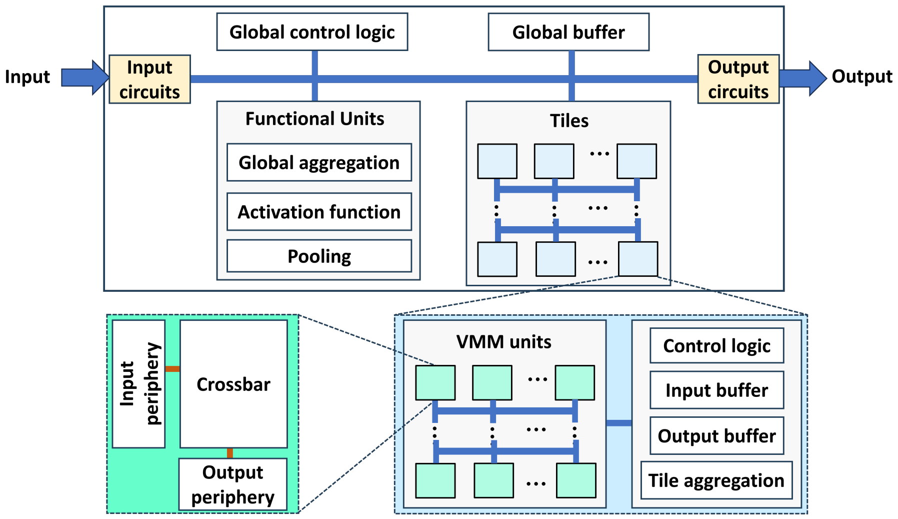


Figure 2.10: CIM system architecture.

Each tile is made up of CIM-based VMM units, input buffer, output buffer, tile aggregation units and control logic. This structure allows distribution of the workload across multiple VMM units for parallel execution. Control logic regulates the operation of all the tile components. Additionally, it synchronizes control sequences with the global control logic. VMM units receive input data from the the input buffer. Their outputs are post-processed by tile aggregation unit through appropriate shift-and-add operations. The final aggregated outputs are stored in the output buffer. Each VMM unit includes control logic, a memristive memory array called crossbar and peripheral circuits. The crossbar holds the network weights, enabling VMM between weights and inputs (activations) in the analog domain. The peripheral circuits act as interface between the crossbar and the rest of the system. They first convert digital inputs from input buffer into analog domain to perform VMM within the crossbar. Subsequently, they convert analog VMM outputs back into digital domain for further processing in the system. This approach allows the VMM unit to benefit from energy-efficiency of analog computations while maintaining compatibility with surrounding digital system.

The operation of this system can be described as follows. The global control unit first distributes the input data from global buffer across all tiles. Each tile stores the assigned inputs in its input buffer and feeds it into its VMM units. The partial outputs generated by the VMM units within each tile are then combined and stored in the tile output buffers. After all tiles finish their VMM computations, their output buffer data is consolidated by global aggregation units. This combined data is then passed through activation and/or pooling units and the resulting outputs are stored back in the global buffer.

2.4.2. CIM VECTOR-MATRIX MULTIPLICATION (VMM) UNIT

We will now explore the working of CIM-based VMM unit shown in Figure 2.11, which is the core component of CIM system architecture. It utilizes memristor-based memory cells for data storage. These cells are arranged in a grid-like structure known as crossbar,

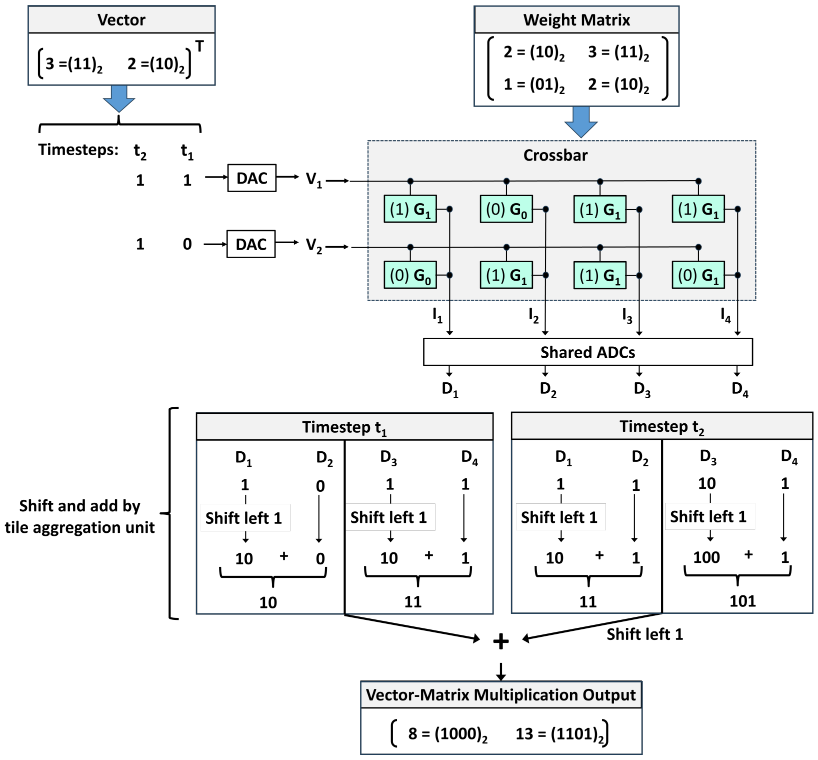


Figure 2.11: Illustration of the operation of CIM VMM unit with an example.

to enable dense data storage and parallel computation. Within the crossbar, weight matrix is stored as conductance values (G's) within each memory cell. Due to the limited bit-capacity of memristor devices, multi-bit weights are typically split across multiple memory cells in a single row. For instance, a 2-bit weight is distributed across two 1-bit memristors. This technique known as bit-slicing [30–32]. The input vector is converted into voltages (V's) using digital-to-analog converters (DACs) and applied to crossbar rows. Multi-bit inputs are often divided into smaller chunks and applied to crossbar using time-division multiplexing, due to the limitations associated with input voltage range and high-resolution DACs. For example, a 2-bit input is divided into single bit chunks that are converted to voltages and applied to the crossbar rows across two timesteps

At each timestep, the current flowing through each G is equivalent to element-wise multiplication of V's and G's as per Ohm's law. Currents from all the G's in a column get accumulated according to Kirchhoff's law to produce output currents (I's). These aggregated currents represent multiply-and-accumulate (MAC) operations in analog domain. The MAC operations across all the columns together represent a partial VMM operation (for that specific timestep). These partial VMM outputs are converted to digital domain by analog-to-digital converters (ADCs). The ADCs are typically shared across multiple columns due to their large pitch size compared to the crossbar. The ADC outputs undergo shift-and-add operation to account for weight slicing across multiple columns. An additional round of shift-and-add operations then merges this output with that of previous timestep, to account for time-division multiplexing of input bits. The final full precision output is obtained by repeating this process till the last timestep of input vector.

2.4.3. Memristive Crossbar Types

We now present three types of memristive crossbar designs for CIM-based VMM unit.

One-resistor (1R) Crossbar

The 1R crossbar, also known as passive crossbar, uses memory cells containing only the memristor device as shown in Figure 2.12. Programming and reading of a specific

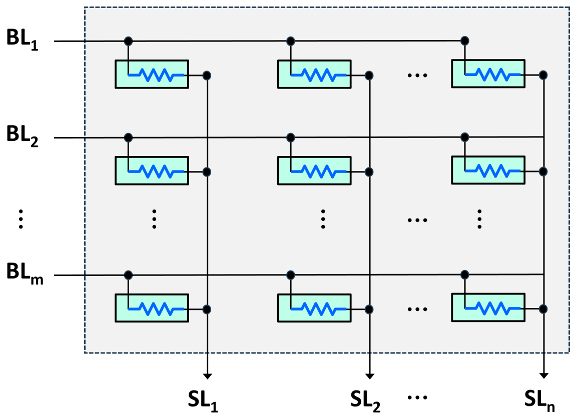


Figure 2.12: 1R (passive) crossbar design.

memristor is achieved by applying suitable voltages at the bit lines (BLs) and the source lines (SLs). Passive crossbar can achieve the best area efficiency and memory density due to the small size of memristor devices. However, it suffers sneak path issues, where SL and BL voltages to program/read a specific memristor also induce current flow in other memristors of the crossbar. During write operations, sneak path current causes an additional voltage drop along wire parasitics. This can result in write failures due to insufficient voltage at the intended memristor. During read operations, sneak currents from other memristors introduce deviations in the read current of the selected memristor. This can cause read failure due to incorrect interpretation of the target memristor's conductance state. Thus, sneak paths present a significant challenge for widespread adoption of 1R crossbars. This issue can be mitigated by integrating an additional selector device in series with the memristor, giving rise to the two crossbar types discussed next.

One-selector and one-resistor (1S1R) Crossbar

The selector is basically a device with bidirectional diode-like I–V characteristics [33]. The memory cells in 1S1R crossbar integrate such a selector device in series with each memristor as shown in Figure 2.13. This eliminates the sneak path issue. Moreover, the selector device can be stacked on top of the memristor, potentially yielding a high memory density and small area footprint. However, development of selector devices that meet the desired performance requirements remains a significant challenge. This limitation hampers the widespread adoption of 1S1R arrays.

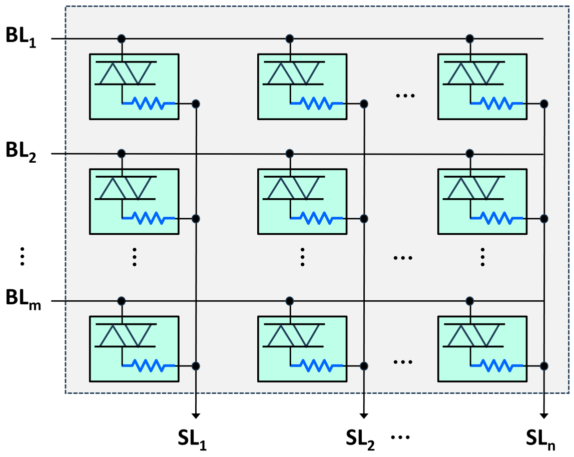


Figure 2.13: 1S1R crossbar design.

One-transistor and one-resistor (1T1R) Crossbar

This crossbar incorporates memory cells composed of transistors in series with the memristors as shown in Figure 2.14. It requires additional control lines, known as word lines (WLs), to selectively activate or deactivate the transistors. This approach effectively mitigates the sneak paths, ensuring accurate programming and reading of the memristors.

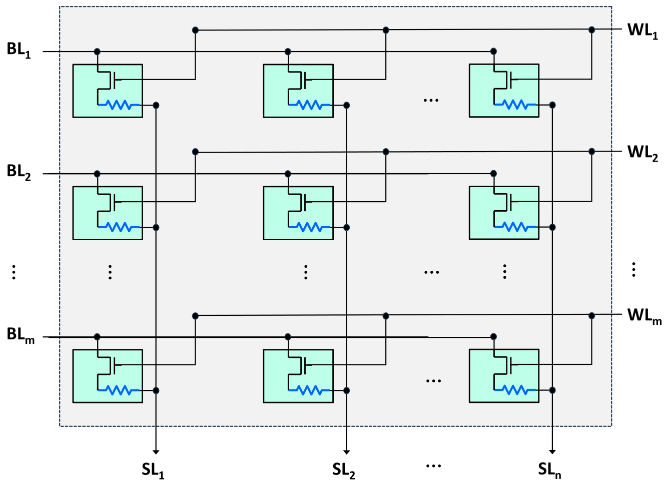


Figure 2.14: 1T1R crossbar design.

Although the inclusion of a transistor in each memory cell leads to a larger area footprint for 1T1R crossbar, it still remains the most preferred choice in CIM designs. This widespread adoption is driven by maturity CMOS fabrication processes, enabling by high-yield production of large-scale 1T1R crossbars.

2.4.4. Memristor Device Technologies

We will now discuss the device technologies used for implementing memristors in the crossbar memory cells. We focus on the four most prominent technologies namely RRAM, PCM, STT-MRAM and FeFET, followed by an assessment of their commercial availability.

Resistive Random Access Memory (RRAM)

The RRAM device is made up of an oxide material sandwiched between two metal electrodes [34, 35], as shown in Figure 2.15. Its conductance can be modulated by creating and disrupting the conductive filament (CF) composed of oxygen vacancies within the oxide layer. The RRAM device exhibits a high conductance (logic 1) when CF connects the electrodes, while a disrupted CF leads to low conductance (logic 0).

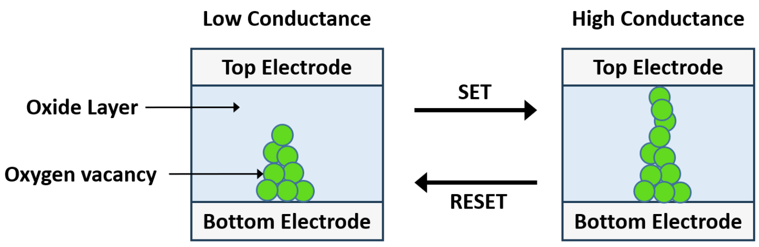


Figure 2.15: Resistive random access memory (RRAM) device.

The SET process achieves high conductance state through a high electric field that forces oxygen ions to drift towards one electrode. This leaves behind vacant oxygen sites in oxide layer, which form CF to increase its conductivity. The RESET process leads to low conductance state using an electric field with polarity opposite to that in SET process. This causes oxygen ions to migrate back into the oxide layer. These ions combine with oxygen vacancies to disrupt the CF , reducing the conductivity of oxide layer. To read data from RRAM device without altering the stored information, a small voltage is applied across it to detect its conductance state.

Phase Change Memory (PCM)

The PCM device consists of a top electrode, a phase-change material layer (e.g. $Ge_2Sb_2Te_5$) and a bottom electrode [36, 37], as shown in Figure 2.16. The phase change material exhibits high conductance in crystalline phase (logic 1) and low conductance in amorphous phase (logic 0).

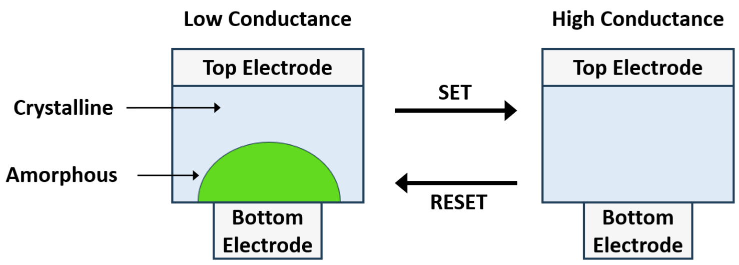


Figure 2.16: Phase change memory (PCM) device.

The transition from low to high conductance state is termed the SET process. Here, a current pulse heats the amorphous material above its crystallization threshold but below its melting point, for a duration sufficient enough for crystallization to occur. Conversely, transitioning from high to low conductance state involves a high-current pulse with an abrupt trailing edge. This pulse melts the phase-change material through Joule heating, followed by rapid cooling to solidify it in the amorphous state. To read the state of the PCM device without disturbing its existing state, a small electrical current is passed through it for detecting its conductance.

Spin-Transfer Torque Magnetic Random Access Memory (STT-MRAM)

The STT-MRAM device consists of an oxide barrier sandwiched between two ferromagnetic layers [38], as shown in Figure 2.17. One ferromagnetic layer (called reference layer) maintains a fixed magnetic field direction, while the other ferromagnetic layer (called free layer) has a programmable magnetic field direction. A parallel (P) alignment of magnetic fields in these layers yields high conductance (logic 1), while an anti-parallel (AP) alignment results in low conductance (logic 0).

Data is written to the STT-MRAM device using current, which must exceed a minimum threshold value to switch the magnetic field direction of the free layer. For the SET operation (changing from AP to P orientation), a write current surpassing this threshold flows from the reference layer to the free layer. The reference layer acts as a spin filter,

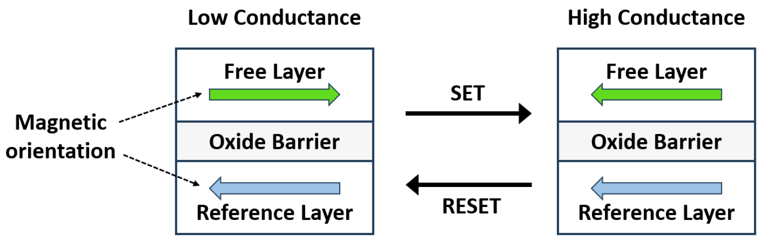


Figure 2.17: Spin-transfer torque magnetic random access memory (STT-MRAM) device.

only allowing electrons aligned with its magnetic field to pass through. These filtered electrons then apply spin-transfer torque to the free layer, altering its magnetic field direction. This results in parallel orientation of the two magnetic layers, which leads to high conductance [39, 40]. In the RESET operation (changing from P to AP orientation), a write current above the threshold flows from the free layer to the reference layer. Out of the electrons emerging from the free layer, reference layer reflects back those with spins opposite to its magnetic field. Such reflected electrons apply spin-transfer torque to the free layer. This alters free layer magnetic field direction toward anti-parallel orientation, resulting in low conductance [39, 40]. Reading a STT-MRAM device without disturbing its existing state involves applying a small current to detect its conductance.

FERROELECTRIC FIELD-EFFECT TRANSISTOR (FEFET)

The FeFET device is basically a transistor with an additional ferroelectric (FE) layer in the MOS structure [41], as shown in Figure 2.18. The polarization of FE layer can be modulated by applying gate to source voltage ($|V_{GS}|$) exceeding a coercive voltage (V_C). The device manifests high conductance (logic 1) if FE layer polarization helps in channel formation, while exhibits low conductance (logic 0) if it opposes channel formation.

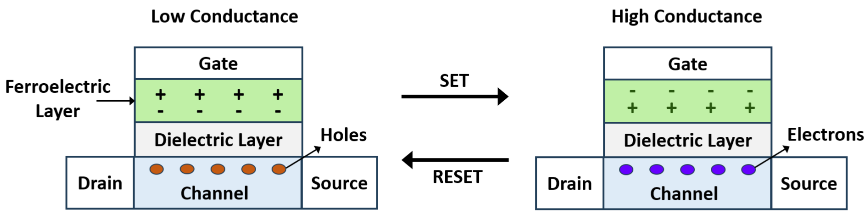


Figure 2.18: Ferroelectric field-effect transistor (FeFET) device.

In an nMOS-based FeFET (n-FeFET), SET process involves a positive V_{GS} ($V_{GS}>V_C$) to polarize FE layer in the direction pointing to the channel. This assists the electrons in the substrate to form a channel, leading to high conductance state. Conversely, RESET process uses a negative V_{GS} ($|V_{GS}|>V_C$) to polarize the FE layer in the direction pointing to the gate terminal. This obstructs the electrons from forming a channel, resulting in low conductance state. Similar concept can be directly extended to a pMOS-based FeFET (p-FeFET). Reading the data from FeFET device without disturbing its state involves applying a small read voltage at its gate to sense the drain-to-source current (I_{DS}) [42].

FABRICATION PROCESSES

Commercial fabrication processes for memristor technologies are now available from leading manufacturers like TSMC, Intel, STMicroelectronics, Globalfoundries, and Samsung as shown in Table 2.1. The RRAM process examples include TSMC 40/28/22 nm RRAMs [43–45], along with Intel 22 nm RRAM [46]. The advancements for PCM are evident with TSMC 40 nm PCM [47] and STMicroelectronics 28 nm PCM [48]. The STT-MRAM technology has seen progress with TSMC 22 nm STT-MRAM [49], Intel 22 nm STT-MRAM [50], Globalfoundries 22 nm STT-MRAM [51], and Samsung 28 nm STT-MRAM [52]. Lastly, Globalfoundries has made notable advancements in FeFET technology with implementations at 28 nm and 22 nm [53, 54].

Table 2.1: Examples of commercial fabrication processes for memristor device technologies.

Memristor device	**Fabrication process**
RRAM	TSMC 40/28/22 nm [43–45], Intel 22 nm [46]
PCM	TSMC 40 nm [47], STMicroelectronics 28 nm [48]
STT-MRAM	TSMC 22 nm [49], Globalfoundries 22 nm [51], Intel 22 nm [50], Samsung 28 nm [52]
FeFET	Globalfoundries 28/22 nm [53, 54]

The industry-mature memristor fabrication processes typically offer 1-bit storage per memristor. Multi-level cell capability, where a single memristor stores multiple bits using intermediate conductance states, has drawn particular interest from the research community. Efforts in this direction have shown promising results for RRAM and PCM technologies [55, 56]. This signifies a pivotal step towards enhancing memory density of CIM architectures.

PART-II

HEALTHCARE AI MODELS FOR CIM EDGE HARDWARE

3 Memristor-based CIM for ECG Arrhythmia Classification

This chapter is based on [24, 57].

3.1. INTRODUCTION

Heart plays an important role in human survival and any heart-related disorders, commonly known as *cardiovascular diseases* (CVDs), can present a significant danger to human life. CVDs are reported to be one of the leading causes of death worldwide [17] and are estimated to cause up to 23 million deaths by 2030 [58]. Diagnosis of CVDs at an early stage can facilitate timely medical treatment and greatly reduce CVD-related health risks. Such early diagnosis can be achieved by detecting the abnormal activity of the heart known as *arrhythmia*. There exist several types of arrhythmia based on the manner in which the heart activity deviates from its normal behavior. Timely detection of various types of arrhythmia requires monitoring of the activity of the heart. Wearable healthcare devices provide the most convenient way of achieving such monitoring. These devices are equipped with sensors that can record the heart activity in the form of *electrocardiogram* (ECG) signal. The task of identifying various types of arrhythmia is then expressed as the classification of heartbeats in the recorded ECG signal into different arrhythmia types (classes). As neural networks are inherently best suited for such classification tasks, these devices use neural networks as ECG classifiers to automatically identify various types of arrhythmia.

The neural network-based ECG classifier in a wearable healthcare device should have high classification accuracy to detect various arrhythmia types correctly. Moreover, it has to be energy-efficient as wearable healthcare devices are battery-powered and thereby have limited energy resources. Its classification outputs should also indicate the severity impact of detected arrhythmia classes which can help the users in knowing how urgently they need to seek medical attention, which can potentially prove to be life-saving. However, state-of-the-art neural network-based ECG classifiers fail to meet these requirements. Many works have adopted neural networks with a large number of layers to obtain high accuracy [59–61]. This results in high energy consumption as such big neural networks require a lot of hardware resources. Most of the existing works just focus on developing ECG classification models without taking into account the implications on hardware performance metrics such as energy [62–70]. Moreover, none of them take the severity impact into account. Hence, there is a strong need for ECG classification hardware that can deliver high accuracy and energy efficiency while also considering severity impact.

In this work, we address the challenge of designing a severity-based, accurate, and energy-efficient ECG classifier. We first create a classification architecture that consists of multiple small sub-classifiers connected in a hierarchical manner instead of a single large and complex classifier. Each sub-classifier deals with only a subset of arrhythmia classes which leads to good accuracy. This hierarchical design also allows us to activate various sub-classifiers only when needed, thereby saving energy. The proposed architecture uses a novel severity-based activation structure for sub-classifiers. The top levels of the hierarchy indicate how quickly the user should seek medical attention. The bottom hierarchical levels help the doctors in diagnosis (the process of finding the physiological root cause of arrhythmia) and then prescribing treatment (medicines, medical procedures, etc.) for arrhythmia based on the diagnosis. Moreover, we propose a hardware design methodology for each internal sub-classifier using the most energy-efficient neural network while still ensuring good accuracy. This hardware design is based on the computation-in-memory

(CIM) paradigm which uses emerging memory technologies such as resistive random access memory, also known as memristors, to provide higher energy efficiency compared to conventional von Neumann architecture-based implementation for neural networks. Our key contributions are summarized as follows:

- We develop a hierarchical classification architecture that breaks down the full classification task into smaller sub-tasks to achieve high accuracy and activates various architectural components only when required in order to save energy.
- We propose a severity-based activation structure that helps the users in seeking timely medical attention as well as helps the medical professional in speeding up the diagnosis and treatment.
- We provide a methodology for the hardware design of various components in the hierarchical ECG classification architecture using memristor-based computation-in-memory paradigm to achieve the best balance between energy efficiency and accuracy.

Simulation results show that the proposed architecture consumes an average energy of 0.11 μJ per heartbeat classification and requires 0.11 mm^2 area, which results in 25× less average energy consumption and 12× less area compared to the state-of-the-art while maintaining high accuracy.

3.2. CARDIAC ARRHYTHMIA

3.2.1. BASICS

The human heart is made up of four chambers. The upper two chambers are called *atria* and the lower two chambers are called *ventricles*. These chambers undergo contraction and relaxation in a periodic manner. This activity can be recorded as a graph of voltage versus time known as *electrocardiogram* (ECG). A single ECG recording contains multiple cycles of contraction and relaxation of the heart chambers. These cycles are known as ECG beats. A visualization of an ECG beat is shown in Figure 3.1 which begins with the contraction of atria represented by 'P'. This is followed by relaxation of the atria and contraction of the ventricles observed as the 'QRS' complex. 'Q' wave represents the

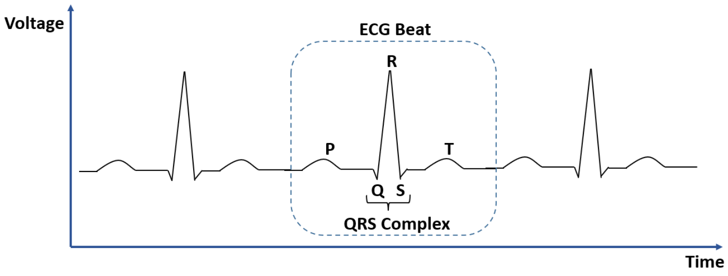


Figure 3.1: Illustration of 'PQRST' cycle for an ECG beat, where P: atrial contraction, Q: interventricular septum contraction, R: ventricular contraction (main mass), S: ventricular contraction (at heart's base), and T: ventricular relaxation. Atrial relaxation is obscured by QRS complex.

Table 3.1: AAMI [73] grouping of ECG arrhythmia classes in MIT-BIH dataset [71]

AAMI Class	Arrhythmia Class
Normal (N)	Normal Beat (N) Left Bundle Branch Block Beat (L) Right Bundle Branch Block Beat (R) Atrial Escape Beat (e) Nodal (Junctional) Escape Beat (j)
Supraventricular Ectopic Beat (S)	Atrial Premature Beat (A) Aberrated Atrial Premature Beat (a) Nodal (Junctional) Premature Beat (J) Supraventricular Premature Beat (S)
Fusion Beat (F)	Fusion of Ventricular and Normal Beat (F)
Ventricular Ectopic Beat (V)	Premature Ventricular Contraction (V) Ventricular Escape Beat (E)
Unknown Beat (Q)	Paced Beat (/) Fusion of Paced and Normal Beat (f) Unclassifiable Beat (Q)

contraction of the interventricular septum. 'R' wave indicates the contraction of the main mass of the ventricles. 'S' wave denotes the contraction of the ventricles at the base of the heart. The beat ends when the ventricles undergo relaxation denoted as 'T'. When a recorded ECG beat deviates from its expected normal behavior, it represents the abnormal activity of the heart chambers called *arrhythmia*. There exist several different classes (types) of arrhythmia based on the exact manner in which the recorded ECG beat deviates from its normal behavior. For instance, MIT-BIH Arrhythmia dataset [71] (provided through PhysioNet [72]) consists of 15 arrhythmia classes which are further grouped into 5 superclasses by *Association for the Advancement of Medical Instrumentation* (AAMI) [73] as shown in Table 3.1. The arrhythmia classes can be distinguished from each other (as well as the normal heart activity) by using different features of the 'QRS' complex such as timing, amplitude, etc. Hence, the 'QRS' complex in an ECG beat plays a crucial role in identifying arrhythmia classes.

3.2.2. DETECTION

Activity of the heart should be regularly monitored for timely detection of arrhythmia. This involves recording the ECG signal and identifying the types of abnormal beats in it. Various approaches used for such monitoring are as follows:

- **Manual**: In this case, medical professionals record the ECG signal at the hospital and identify the abnormal beats by visual inspection. This requires frequent visits to the hospital which are time-consuming and inconvenient for most people. Moreover, arrhythmia may get detected late as there is no monitoring of the heart activity in the time span between successive hospital visits.

- **Semi-automated**: Problems such as inconvenience in frequent hospital visits and late arrhythmia detection in the manual approach can be solved by using wearable healthcare devices. Such devices allow the monitoring of heart activity without hospital visits. These devices contain sensors that can directly record the ECG signal and are also equipped with hardware that can identify the types of abnormal beats. If the hardware in such devices uses traditional machine learning techniques (which do not involve neural networks) like support vector machine [74], the features have to be first manually extracted from the ECG recording and provided as inputs to the device. Hence, such an approach is known as semi-automated. It suffers from poor classification performance due to the imprecise nature of manual feature extraction.

- **Fully automated**: The need for manual feature extraction in the semi-automated approach can be eliminated by using neural networks. They are inherently capable of extracting the features from ECG recordings and then performing classification based on the extracted features. Hence, this approach is called fully automated. Moreover, automatic feature extraction results in superior classification performance compared to manual feature extraction. This can prove crucial for correct diagnosis and timely treatment.

Hence, neural network-based fully automated ECG classification is the most effective approach to building smart arrhythmia detection solutions. The generic flow for the development of neural network-based ECG classification solutions is shown in Figure 3.2. The recorded ECG data is pre-processed to remove the noise and enhance the regions of interest such as the 'QRS' complex in each ECG beat. It is then divided into a training set, a validation set, and a test set. The neural network training and hyperparameter tuning is performed using the training set and validation set, respectively. The classification performance of the trained network is then evaluated using the test set followed by the

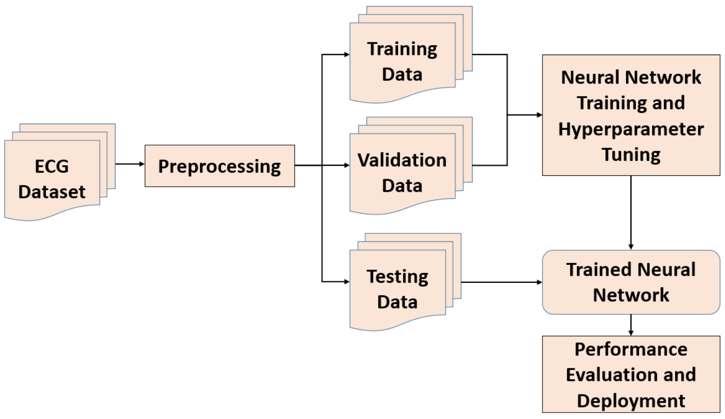

ECG Dataset
Preprocessing
Training Data
Validation Data
Testing Data
Neural Network Training and Hyperparameter Tuning
Trained Neural Network
Performance Evaluation and Deployment

Figure 3.2: Development flow for neural network-based ECG classification.

model deployment once the performance on the test set is deemed satisfactory.

3.3. PROPOSED METHODOLOGY

3.3.1. SEVERITY-BASED CLASSIFICATION APPROACH

A medical disorder represents improper functioning of a certain organ in the human body and has various subtypes based on the manner in which the malfunctioning occurs in that organ. For instance, arrhythmia is a disorder that indicates improper functioning of the heart, and there exist different arrhythmia subtypes based on which part of the heart is affected as well as the way in which it is affected. The disorder subtypes differ in severity impact based on the extent to which they obstruct the organ's normal functioning. Severe subtypes may highly impact the organ leading to life-threatening situations, while others may have a minor impact leading to just a temporary inconvenience. The severity impact determines the urgency with which a person should seek medical help for certain disorder subtypes. For example, severe subtypes may need medical attention immediately while the non-severe ones may need it within a few days. Moreover, the desired speed of diagnosis and treatment is also governed by the severity impact. For instance, severe subtypes may need faster diagnosis and treatment to prevent further damage to health over time, while such a speedup may not be necessary for the non-severe subtypes.

The influence of severity impact on the urgency in seeking medical attention as well as the speed of diagnosis and treatment can be leveraged to create severity-based classification with a two-level hierarchical structure as shown in Figure 3.3. First level of the hierarchy is intended for the user of the wearable device and indicates how urgently one needs to seek medical attention. Knowing the severity impact alone would suffice for this purpose, without knowing the exact disorder subtype. Hence, we can group the disorder subtypes into four broad classes based on their severity impact as follows:

- Normal: This class represents normal working of the human body and does not require any medical attention.
- Mild: This class includes disorder subtypes that have a very minor impact on normal organ functioning and do not lead to life-threatening scenarios over time. It is advisable to schedule a checkup in the upcoming few days if this class is detected.
- Moderate: This class includes disorder subtypes that have a minor impact on nor-

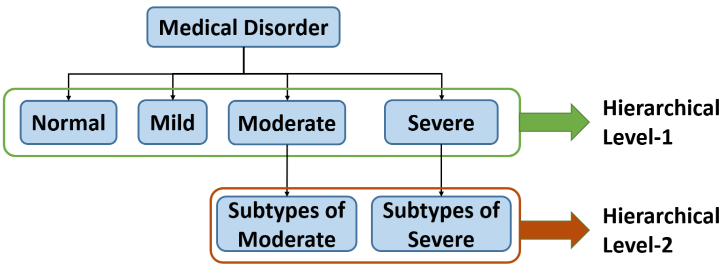


Figure 3.3: Severity-based hierarchical classification for a wearable device monitoring a medical disorder.

mal organ functioning at onset, but can potentially cause life-threatening scenarios over time. It requires faster medical attention than mild class, but not immediately.

- Severe: This class includes disorder subtypes that can significantly affect normal organ functioning at onset and are very likely to lead to life-threatening scenarios. It requires immediate medical attention upon detection.

Such a grouping can potentially improve the classification accuracy as the wearable device needs to detect only four broad classes instead of tens of subtypes of the considered medical disorder. Moreover, as the number of output classes is reduced, a smaller neural network can be used to reduce energy consumption while still maintaining high accuracy. The second level of this hierarchy is intended for speeding up the diagnosis and treatment. This can be achieved if the wearable device detects the exact disorder subtype and presents this information to the medical professional. Such speedup is only required for disorder subtypes that are either life-threatening from the onset or which become life-threatening over time. Hence, we need to only detect the disorder subtypes which are grouped together into moderate and severe classes at hierarchical level-1. As a result, hierarchical level-2 only consists of subtypes of moderate class and subtypes of severe class. We refer to the process of detecting the disorder types contained within a broad level-1 class as finer classification. As discussed earlier, it is clear that finer classification is only required for moderate and severe classes in level-1. Finer classification becomes redundant and is not required for mild class as everything will be thoroughly examined in a full checkup. Also, there is no need for finer classification of normal class as it needs no medical attention. As finer classification is not required for all disorder subtypes, this also simplifies the classification task as well as the hardware design providing further accuracy and energy efficiency benefits.

For this work, Table 3.2 shows the mapping of arrhythmia classes (subtypes) in the MIT-BIH arrhythmia dataset [71] to our severity-based classification structure. The rationale

Table 3.2: Severity-based ECG classification hierarchy for arrhythmia classes in MIT-BIH dataset, with details regarding advice for medical attention and need of finer classification.

Hierarchical Class	**MIT-BIH Dataset Class**	**Advice for Medical Attention**	**Finer Classification**
Normal	Normal Beat (N) Paced Beat (/)	No medical attention required.	Not required.
Mild	Left Bundle Branch Block Beat (L) Right Bundle Branch Block Beat (R) Atrial Escape Beat (e) Nodal (junctional) Escape Beat (j)	A checkup in the upcoming days.	Not required.
Moderate	Atrial Premature Beat (A) Aberrated Atrial Premature Beat (a) Nodal (junctional) Premature Beat (J) Supraventricular Premature Beat (S) Ventricular Escape Beat (E)	Medical attention within a few hours.	Required.
Severe	Fusion of Ventricular and Normal Beat (F) Premature Ventricular Contraction (V) Fusion of Paced and Normal Beat (f) Unclassifiable Beat (Q)	Medical attention immediately.	Required.

behind this mapping can be explained as follows [75, 76]:

- “N" beats belong to the normal class as they represent the normal working of the heart. Moreover, "paced" beats (/) also belong to the normal class as they indicate the normal working of the heart when aided by a pacemaker.
- “L", “R", “e", “j" beats belong to mild class because even though they deviate from perfectly normal beats (“N" and “paced"), they do not affect the functioning of the heart significantly and do not result in life-threatening scenarios over time.
- “A", “a" and “S" beats are related to improper functioning of atria which do not contribute significantly to the blood circulation process, while “J" and “E" beats indicate only a minor impact on ventricles which are vital for blood circulation. Hence, these beats have little impact on proper heart functioning at the onset. However, they can potentially lead to life-threatening scenarios over time and hence belong to the moderate class.
- “V" beat arises due to abnormal functioning of ventricles which are vital for blood circulation and indicates danger to human life. “F" and “f" beats represent superimposition of cardiac cell potentials which can also lead to life-threatening scenarios. Hence, “V", “F" and “f" belong to the severe class. Moreover, we conservatively include unclassifiable beat (“Q") in severe class as its exact nature is not clear.

3.3.2. HIERARCHICAL HARDWARE ARCHITECTURES

Human health often falls within normal and mild classes, outnumbering instances of moderate and severe classes. When health deteriorates to moderate or severe class, individuals seek medical intervention and return to normal or mild classes. Thus, ECG classifier predominantly deals with normal and mild classes, while encountering moderate and severe classes infrequently. This presents an opportunity to enhance energy efficiency by employing a hierarchical architecture of smaller classifiers. Each classifier would be dedicated to a specific subset of arrhythmia classes, leveraging the following strategy:

- Activate classifiers dealing with infrequent classes only when necessary. This conserves energy by keeping them inactive for extended periods.
- Simplify the design of classifiers handling frequent classes (e.g. by using smaller or simpler neural networks) to further reduce energy consumption.

This approach leads to four possible architectures for our severity-based hierarchy as shown in Table 3.3, discussed next.

ARCHITECTURE-1

This is the simplest architecture for the severity-based ECG classification. Classifier-1 classifies the input into four classes: normal, mild, moderate and severe. Classifier-2 and classifier-3 classify moderate and severe classes further into their subtypes. Classifier-1 activates classifier-2 or classifier-3 when it detects moderate or severe class.

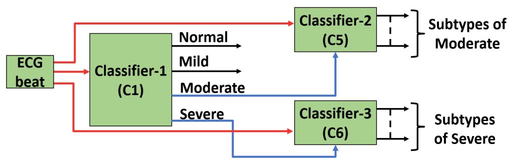
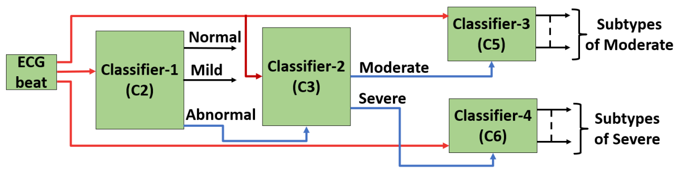
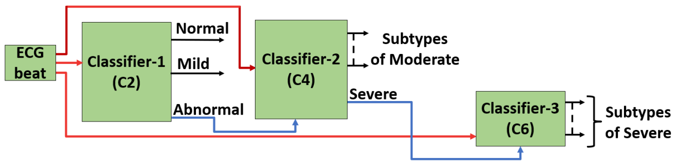
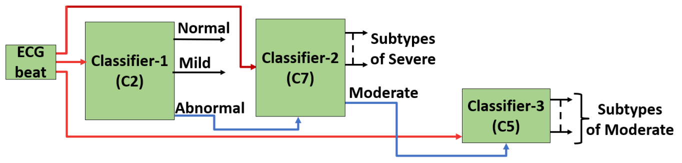
Name	Architecture Structure
Architecture-1	 ECG beat Classifier-1 (C1) Normal Mild Moderate Severe Classifier-2 (C5) Subtypes of Moderate Classifier-3 (C6) Subtypes of Severe
Architecture-2	 ECG beat Classifier-1 (C2) Normal Mild Abnormal Classifier-2 (C3) Moderate Severe Classifier-3 (C5) Subtypes of Moderate Classifier-4 (C6) Subtypes of Severe
Architecture-3	 ECG beat Classifier-1 (C2) Normal Mild Abnormal Classifier-2 (C4) Subtypes of Moderate Severe Classifier-3 (C6) Subtypes of Severe
Architecture-4	 ECG beat Classifier-1 (C2) Normal Mild Abnormal Classifier-2 (C7) Subtypes of Severe Moderate Classifier-3 (C5) Subtypes of Moderate

Table 3.3: Possible hardware architectures for severity-based ECG classification. ECG input data is indicated by red arrows, while the blue arrows represent classification outputs that also act as enable signals. Classifier names are shown in brackets, where the classifiers present in different architectures but having the same output classes are given the same name.

This architecture leads to energy savings as classifier-2 and classifier-3 remain inactive for most of the time. As classifier-1 is always on, it needs to use a smaller neural network to improve energy efficiency. Moreover, high accuracy for classifier-1 is important as its output advises the user about seeking medical help. However, a small neural network may not lead to high accuracy for classifier-1. Hence, there is a potential challenge of simultaneously achieving high accuracy and energy efficiency for classifier-1 in this architecture.

ARCHITECTURE-2

Architecture-2 can facilitate high accuracy with a small neural network for classifier-1 by using only three classes. This is achieved by grouping moderate and severe classes into a single abnormal class for classifier-1 and using an additional classifier-2 to split the abnormal class into moderate and severe classes. Classifier-2 activates classifier-3 and classifier-4 to classify moderate and severe classes further into their subtypes.

This architecture can potentially achieve high accuracy and low energy consumption for classifier-1 by using a smaller neural network, as classifier-1 now handles three classes unlike four classes in architecture-1. However, this architecture requires a total of four classifiers instead of three classifiers in architecture-1 which can increase overall energy consumption.

ARCHITECTURE-3

Architecture-3 can reduce the number of classifiers from four (in architecture-2) to three, while still maintaining only three classes in classifier-1. This is achieved by making classifier-2 handle six classes: five of them being subtypes of moderate (A, a, J, S, E) and sixth being the severe class. Thus, classifier-1 still handles three classes: normal, mild, and abnormal. It activates classifier-2 if it detects abnormal class. Classifier-2 then classifies abnormal class into six classes: A, a, J, S, E, and severe. If classifier-2 detects severe class then it activates classifier-3 which further classifies the severe class into its subtypes.

This architecture can retain all the benefits of classifier-1 in architecture-2 while reducing the overall energy consumption compared to architecture-2 as it needs only three total classifiers unlike four total classifiers in architecture-2. However, classifier-2 in this architecture has to deal with six classes unlike classifier-2 in architecture-2 which deals with only two classes. This can result in reduced accuracy.

ARCHITECTURE-4

Architecture-4 provides another way of reducing the total number of classifiers in architecture-2 by making classifier-2 handle five classes: four of those being subtypes of severe (F, V, f, Q) and the fifth one being the moderate class. Classifier-1 still has to deal with only three classes: normal, mild, abnormal and activates classifier-2 if it detects the abnormal class. Classifier-2 then classifies abnormal into five classes: F, V, f, Q, and moderate. If classifier-2 detects moderate class then it activates classifier-3 which further classifies moderate into its subtypes.

Classifier-1 in this architecture provides the same benefits as classifier-1 in architecture-3. However, classifier-2 and classifier-3 remain on for significantly more amount of time compared to those in architecture-3 as moderate classes occur more frequently than severe ones. This makes architecture-4 a worse version of architecture-3 in terms of energy efficiency. Hence, we do not select architecture-4, and it is just included here for completeness purpose.

3.3.3. ARCHITECTURE SELECTION PROCESS

Our goal is to select the appropriate architecture out of those presented in the previous section, considering accuracy and energy consumption. We can rule out architecture-4 as discussed in the previous subsection, which leaves us with architecture-1, architecture-2, and architecture-3 as possible choices. Table 3.4 lists various classifier components needed for implementing architecture-1, architecture-2, and architecture-3. We assign names (C1, C2....C6) to the individual classifier components to make it easy to refer to a particular classifier. The architecture selection process consists of two phases as

Table 3.4: List of classifier components required for various architectures.

Classifier	Output Classes	Used in
C1	Normal, Mild, Moderate, Severe	Arch1
C2	Normal, Mild, Abnormal	Arch2, Arch3
C3	Moderate, Severe	Arch2
C4	Moderate Subtypes (A, a, J, S, E) and Severe	Arch3
C5	Moderate Subtypes (A, a, J, S, E)	Arch1, Arch2
C6	Severe Subtypes (F, V, f, Q)	Arch1, Arch2, Arch3

depicted in Figure 3.4. In the first phase, our goal is to select an architecture that results in better accuracy on abnormal (moderate and severe) classes. This is governed by classifier C1 for architecture-1 and classifier C2 for architecture-2 as well as architecture-3. Thus, if classifier C1 is better at detecting abnormal classes then we select architecture-1. Otherwise, we discard architecture-1 and perform further exploration to select either architecture-2 or architecture-3 in the second phase.

The selection between architecture-1 and architecture-2 in the first phase depends on the choice between C1 and C2 as follows:

- If C1 has a significantly higher accuracy than C2, then select architecture-1.
- If C2 turns out to be significantly more accurate than C1, then select architecture-2.
- If C1 and C2 have similar overall accuracy, select architecture-1 if C1 performs better

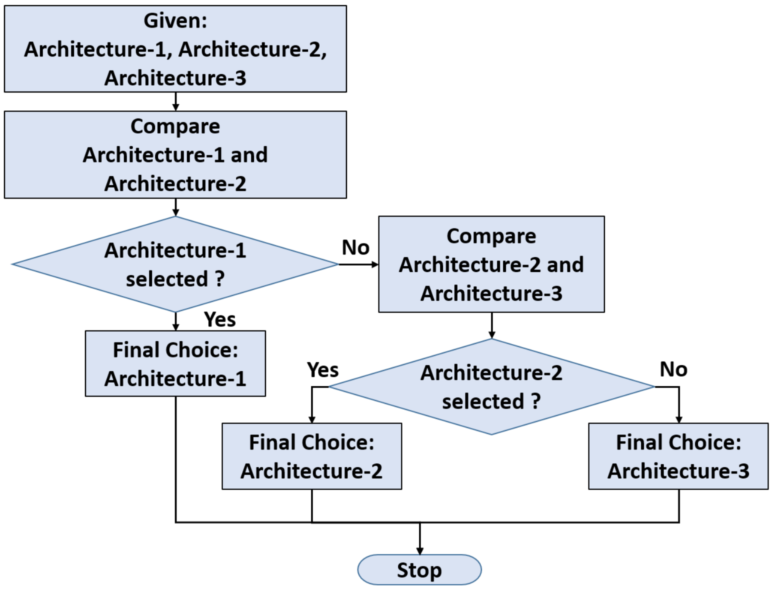

Given:
Architecture-1, Architecture-2, Architecture-3
Compare Architecture-1 and Architecture-2
Architecture-1 selected ?
No
Yes
Compare Architecture-2 and Architecture-3
Final Choice: Architecture-1
Architecture-2 selected ?
Yes
No
Final Choice: Architecture-2
Final Choice: Architecture-3
Stop

Figure 3.4: Architecture selection process.

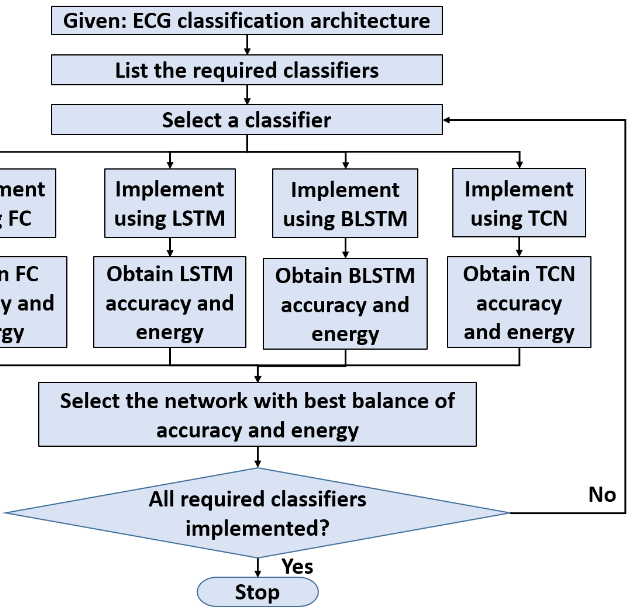

Given: ECG classification architecture
List the required classifiers
Select a classifier
Implement using FC
Implement using LSTM
Implement using BLSTM
Implement using TCN
Obtain FC accuracy and energy
Obtain LSTM accuracy and energy
Obtain BLSTM accuracy and energy
Obtain TCN accuracy and energy
Select the network with best balance of accuracy and energy
All required classifiers implemented?
No
Yes
Stop

Figure 3.5: Design space exploration flow for implementing a given hierarchical hardware architecture.

on moderate and severe classes (life-threatening scenarios). Otherwise, choose architecture-2 if C2 excels in these classes.

- If C1 and C2 have similar overall and critical (moderate and severe classes) accuracy, select the architecture that uses more energy-efficient classifier.

If we end up selecting architecture-2, the second phase involves comparing architecture-2 and architecture-3 based on classifiers C3 and C4 as follows:

- If C3 turns out to be significantly more accurate than C4, we select architecture-2.
- If C4 has much higher accuracy than C3, we select architecture-3.
- If C3 and C4 have similar overall accuracy, then select architecture-2 if C3 has higher accuracy on severe class (life-threatening), otherwise select architecture-3 if C4 has higher accuracy on severe class.
- If C3 and C4 have similar overall and critical (severe class) accuracy, choose the architecture that uses the less energy consuming classifier out of these two.

Thus, we have defined clear selection criteria for the architectures. To use it, we have to determine the topology and network configuration for each classifier which provides the best balance between accuracy and energy efficiency when implemented in CIM hardware. This is achieved by design space exploration as shown in Figure 3.5 across four types of neural networks: fully-connected network (FC) [77, 78], long short-term memory network (LSTM) [79, 80], bidirectional long short-term memory network (BLSTM) [81, 82] and temporal convolutional network (TCN) [83, 84]. We first list the various classifiers needed for a given hierarchical hardware architecture. We then choose a classifier from this list and implement it using all four aforementioned network types (FC, LSTM, BLSTM and TCN) so that each network achieves its maximum possible accuracy. Energy consumption for all four resulting networks is then estimated by considering a CIM-based hardware implementation. Finally, we select the network (FC or LSTM or BLSTM or TCN) which provides the best balance between accuracy and energy consumption for the classifier. This process is repeated for all the classifiers required in the given architecture. Standard convolutional neural network (CNN) [85] is not included for this design space exploration as we already consider TCN, which is an advanced form of CNN that deals more effectively with time series data like ECG. For completeness, we will compare our hierarchical ECG classification with state-of-the-art CNN-based ECG classification in Section 3.5.

3.4. SIMULATION SETUP

3.4.1. PERFORMANCE METRICS

Performance metrics for the evaluation of our hierarchical ECG classification are considered at two levels: algorithmic and hardware, described in detail as follows:

ALGORITHMIC METRICS

- **Accuracy**: It is the ratio of the total number of correctly classified beats to the total number of input beats, expressed as a percentage.

- **Critical Accuracy**: We define critical classes as a subset of the total output classes that can be more life-threatening and hence considered more important. Table 3.5 defines critical classes for various classifiers required in severity-based classification architectures presented in Section 3.3.2. The concept of critical classes is only applicable to classifiers that handle at least one of the broad classes (normal, abnormal, mild, moderate, and severe) which are fundamentally based on severity differences. It is not applicable to classifiers that only handle subtypes of moderate or subtypes of severe as the subtypes indicate similar severity levels. We now define *critical accuracy* as the ratio of the total number of correctly classified beats belonging to the critical classes to the total number of input beats belonging to the critical classes.

Table 3.5: Critical class definitions for classifiers.

Classifier	Output Classes	Critical Classes
C1	Normal, Mild, Moderate, Severe	Moderate, Severe
C2	Normal, Mild, Abnormal	Abnormal
C3	Moderate, Severe	Severe
C4	Severe and Moderate Subtypes: A, a , J, S, E	Severe
C5	Moderate subtypes (A, a, J, S, E)	-
C6	Severe subtypes (F, V, f, Q)	-

For instance, consider classifier C1 in Table 3.5 with four output classes: normal, mild, moderate, and severe. As moderate and severe can lead to life-threatening scenarios, they are considered critical classes. Critical accuracy for classifier C1 can then be obtained as follows:

$$\text{Critical accuracy for C1 (\%)} = 100 \times \frac{\text{Correct}_{\text{crit}}}{\text{Total}_{\text{crit}}}$$

$\text{Correct}_{\text{crit}}$: Correct classified moderate and severe beats
$\text{Total}_{\text{crit}}$: Total input moderate and severe beats

Hardware Metrics

- **Energy**: As wearable healthcare devices are battery-powered, the energy consumed by the CIM-based ECG classifier is an important hardware performance metric.
- **Area**: Apart from energy efficiency, wearable healthcare devices should be compact in size. Hence, the area occupied by the CIM-based ECG classifier is considered another hardware performance metric.

3.4.2. Simulation Platform

We use the MIT-BIH arrhythmia dataset [71] available in Physiobank [72] for our simulation experiments. It consists of ECG recordings from 48 patients across 15 arrhythmia types. The distribution of ECG beats for each of the arrhythmia types is given in Table 3.6. As this work focuses on ECG classification and not on QRS peak detection, we directly use the QRS peak annotations available in the MIT-BIH dataset. QRS peak detection at runtime can be achieved by algorithms like Pan-Tompkins algorithm [86] which can also be implemented in hardware [87].

Accuracy and critical accuracy are evaluated by implementing the neural networks using PyTorch [88] with RMSProp [89] optimizer. The details of the used neural networks are described below. Please note that only the number of output neurons (n_{out}) varies from two to six based on which classifier is being implemented, the rest stays the same.

- Fully-connected network (FC) [77, 78]: It has an input layer of 250 neurons, a hidden layer of 100 neurons, and n_{out} output neurons. FC network can thus be expressed as 250-100-n_{out}. The activation function used is ReLU.
- Long short-term memory network (LSTM) [79, 80]: An input sequence of 250 samples is fed to two cascaded standard LSTM units, each having a hidden state size of 30. The output from the last LSTM unit corresponding to the final timestep is flattened and connected to an output layer consisting of n_{out} neurons. LSTM structure can be expressed as 250-LSTM(30)-LSTM(30)-Flatten-n_{out}.
- Bidirectional long short-term memory network (BLSTM) [81, 82]: Its structure is exactly the same as the LSTM described earlier, with standard LSTM units being replaced by their bidirectional version. BLSTM structure can be expressed as 250-BLSTM(30)-BLSTM(30)-Flatten-n_{out}.

Table 3.6: Distribution of ECG Beats in MIT-BIH arrhythmia dataset.

MIT-BIH Arrhythmia Class	No. of Beats
Normal Beat (N)	75022
Paced Beat (/)	7025
Left Bundle Branch Block Beat (L)	8072
Right Bundle Branch Block Beat (R)	7255
Atrial Escape Beat (e)	16
Nodal (junctional) Escape Beat (j)	229
Atrial Premature Beat (A)	2546
Aberrated Atrial Premature Beat (a)	150
Nodal (junctional) Premature Beat (J)	83
Supraventricular Premature Beat (S)	2
Ventricular Escape Beat (E)	106
Fusion of Ventricular and Normal Beat (F)	802
Premature Ventricular Contraction (V)	7129
Fusion of Paced and Normal Beat (f)	982
Unclassifiable Beat (Q)	33
Total Heartbeats	109452

- Temporal convolutional network (TCN) [83, 84]: It is provided with a 250 sample long single channel input sequence. This sequence is fed into a cascade of six temporal blocks. The convolutions within each temporal block have a kernel size of four and 20 output channels. Output from the last temporal block corresponding to the final timestep is flattened and connected n_{out} neurons in the output layer. TCN structure can be expressed as 250-TB1-TB2-TB3-TB4-TB5-TB6-Flatten-n_{out}, where TBn represents n^{th} temporal block.

We split the ECG data as 60% for the training set, 20% for the validation set, and 20% for the test set. The networks are trained using the training set and the validation set is used for hyperparameter tuning. The test set is not exposed to the network during training or the hyperparameter tuning process. It is used only after the network is fully trained and tuned. All the accuracy and critical accuracy results are presented for the test set so that they correctly reflect the generalization performance on unseen test data.

We have developed a Python-based framework to estimate energy and area for neural networks using computation-in-memory hardware known as ISAAC presented in [30]. Its main building block is shown in Figure 3.6. The full-precision neural network weights and inputs are split into smaller bit-size chunks called slices. This is because i) the bit-capacity of memristor devices is typically less than bit-size needed for neural network weights and ii) digital-to-analog converters (DACs) and analog-to-digital converters (ADCs) with high bit-resolutions consume high energy and area. For example, as shown in Figure 3.6, 2-bit slices of the weights are converted to conductances and mapped to memristors in different crossbar columns, while 1-bit slices of the inputs are converted to voltages

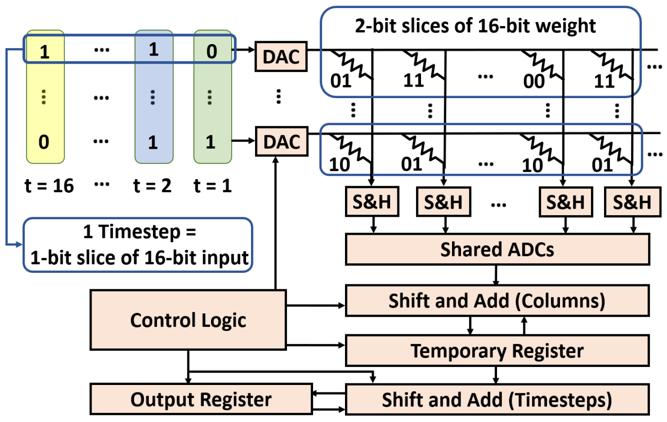


Figure 3.6: Computation-in-memory design for vector-matrix multiplication.

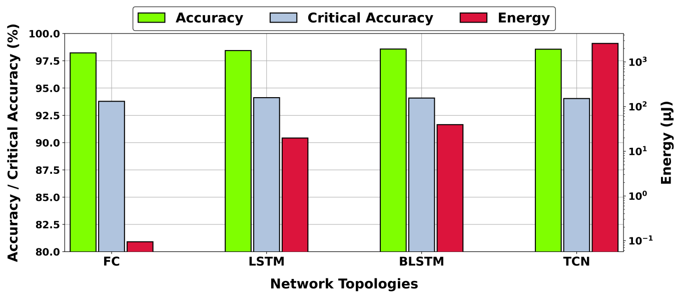
and mapped to different time-steps in which they are applied to the crossbar. With 1-bit DACs and 16-bit digital inputs as an example, 1-bit is fed at a time to all the DACs and this process is repeated 16 times (called 16 timesteps). The DACs convert the bits into equivalent voltage which produces a current at the output of every column in the crossbar. These currents are latched into sample and hold circuits (S&H) and then converted to digital outputs by ADCs. The outputs of ADCs belong to various weight slices based on which column they come from, and to different input data slices based on which timestep they belong to. To account for the slicing of weights across different crossbar columns, ADC outputs undergo shift and add operations across columns. Moreover, to account for time-multiplexed inputs (1-bit at a time), the shifted and added ADC outputs undergo another round of shift and add operations for merging with the outputs from previous timesteps to produce the full-precision digital output. We estimate the energy and area for neural networks using the design in [30] which utilizes this functionality.

3.5. Simulation Results

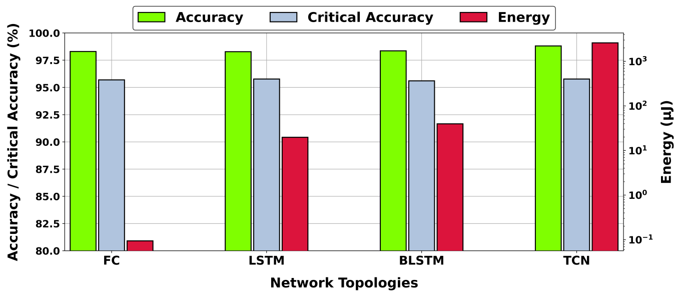
3.5.1. Hierarchical Architecture Design

As discussed in Section 3.3.3, we break the task of selecting the appropriate architecture into two phases. In the first phase, we make a choice between architecture-1 and architecture-2. The selection process stops if architecture-1 is selected. Otherwise, we proceed to the second phase to make a selection between architecture-2 and architecture-3, as our final choice.

For the first phase, the choice between architecture-1 and architecture-2 is governed by the comparison of classifiers C1 (Normal vs Mild vs Moderate vs Severe) and C2 (Normal vs Mild vs Abnormal) described in Table 3.4. We implement both C1 and C2 using all four types of neural networks (FC, LSTM, BLSTM and TCN). Figure 3.7 shows the performance metrics across various network topologies for C1 and C2. It is clear that FC provides the best balance between accuracy and energy efficiency for both C1 and C2. FC achieves accuracy comparable to other network topologies (LSTM, BLSTM, TCN) as the



(a) Design space exploration for classifier C1



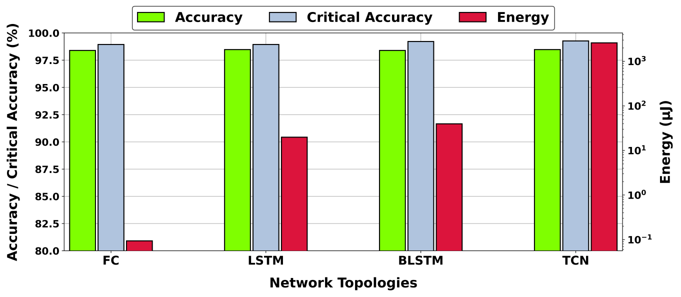
(b) Design space exploration for classifier C2

Figure 3.7: Comparison of classifiers C1 and C2 for selection between architecture-1 and architecture-2.

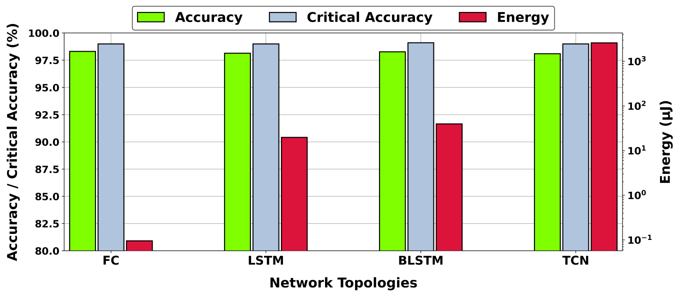
classification boundaries for ECG data seem to be simple and do not benefit much from the extra computational powers in other topologies. The low energy consumption of FC can be attributed to two factors:

- It needs fewer hardware resources compared to LSTM, BLTSM, and TCN as it doesn't involve complex computations like hidden state updates in LSTM/BLSTM or convolution operation in TCN.
- LSTM, BLSTM, and TCN involve iterative computations such as updating the hidden state after each input sample (LSTM and BLSTM) or sliding convolution windows across input feature maps (TCN). Thus, they use the same hardware multiple times and total energy is the sum of energies required for each iteration. The energy consumption increases further as such iterative computation is needed for each layer in the network. FC just requires a single non-iterative matrix-matrix multiplication per layer, saving a lot of energy.

CNN [85] is not included in the above comparison as clarified in Section 3.3.3. Nevertheless, CNN will also suffer from high energy consumption problem like TCN because both of them involve iterative sliding window convolution as the basic computation. We quantitatively demonstrate this later in Table 3.7 by comparing our FC-based ECG classification with CNN-based state-of-the-art ECG classification [59–61].



(a) Design space exploration for classifier C3



(b) Design space exploration for classifier C4

Figure 3.8: Comparison of classifiers C3 and C4 for selection between architecture-2 and architecture-3.

Mixing network topologies into a hybrid structure can potentially yield better results when the topologies that are being mixed have a significant difference in accuracy but not a large difference in energy consumption. However, Figure 3.7 shows that all of these networks deliver similar accuracy while LSTM, BLSTM, and TCN consume much more energy than that of FC. Hence, topology mixing is not useful as it would result in adding a large energy-consuming component to the FC network for almost no change in accuracy.

As shown in Figure 3.7, FC version of C2 achieves 2% higher critical accuracy (please see Table 3.5 for critical classes) than the FC version of C1. This is because combining the moderate and severe classes in C1 into a single abnormal class in C2 simplifies the classification task, as C2 has to learn only three classification boundaries (normal vs mild vs abnormal) unlike four classification boundaries in C1 (normal vs mild vs moderate vs severe). Higher critical accuracy also indicates that C2 correctly detects more scenarios where the user needs to seek medical help. Hence, we select the FC version of C2 and thereby architecture-2 in the first phase.

Having selected architecture-2 in the first phase, we then begin the second phase to make a selection between architecture-2 and architecture-3. This depends on the comparison between classifiers C3 (Moderate vs Severe) and C4 (A vs a vs J vs S vs E vs Severe) described in Table 3.4. Both C3 and C4 are implemented using all four types of neural networks (FC, LSTM, BLSTM, and TCN) and their performance comparison across various network topologies is shown in Figure 3.8. FC network ends up delivering the

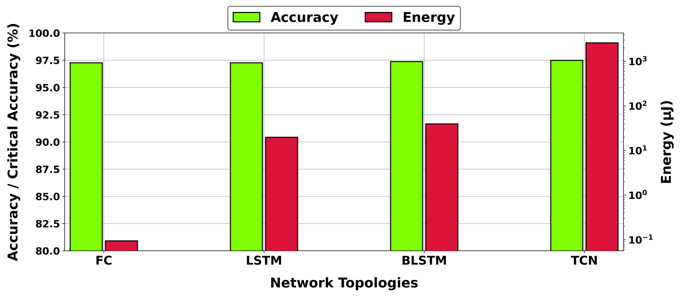


Figure 3.9: Design space exploration for classifier C6.

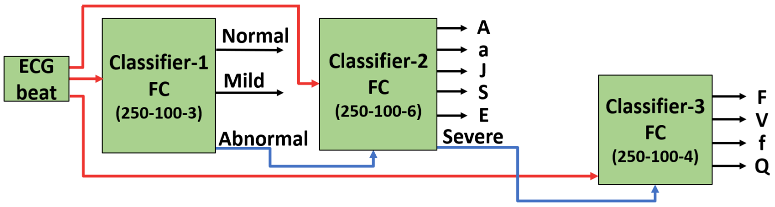


Figure 3.10: Final classifier architecture with network type and configuration annotated.

best balance between accuracy and energy efficiency for both C3 and C4, for the same reasons as discussed while comparing C1 and C2. Also, FC versions of C3 and C4 have almost identical performance across all the metrics. However, C4 leads to architecture-3 with a total of three classifiers while C3 leads to architecture-2 which needs a total of four classifiers. Thus, we select C4 (its FC version) and thereby architecture-3 as it needs fewer hardware resources and less energy, with no impact on accuracy and critical accuracy.

After finalizing architecture-3, the only remaining thing is to figure out the neural network type and configuration to use for its remaining classifier (classifier C6 in Table 3.4) which deals with the classification of subtypes of severe class (F, V, f, and Q). We implement it using all four possible types of neural networks (FC, LSTM, BLSTM, and TCN) and show their performance comparison in Figure 3.9. The FC version is selected as it provides the best balance between accuracy and energy efficiency. Thus, our final selection is architecture-3 with all of its classifiers being FC networks with configurations as shown in Figure 3.10. Classifier-1, classifier-2 and classifier-3 achieve accuracy of 98.29%, 98.31%, 97.26% and energy consumption of 0.094 μJ, 0.095 μJ, 0.094 μJ, respectively. As the selection between architecture-1 and architecture-2 depends on the choice between classifiers C1 and C2 only, while that between architecture-2 and architecture-3 depends on the comparison between classifiers C3 and C4 only, we have not evaluated the accuracy of classifier C5 in Table 3.5. Moreover, as we end up selecting architecture-3 which does not include C5, there is no further need to evaluate its accuracy.

3.5.2. PERFORMANCE EVALUATION

Performance comparison of our hierarchical ECG classification with state-of-the-art is shown in Table 3.7. It includes [60] which represents the most accurate ECG classifica-

Table 3.7: Comparison of the proposed hierarchical classification with state-of-the-art ECG classifiers. Values marked with * are estimated by our framework assuming memristor-based CIM implementation. Unavailable or not applicable values are indicated by "-".

Performance Metric	Wu-IEEE Access'2019 [59]	Xiao-JBHI'2022 [60]	Wang-TBCAS'2019 [61]	This Work
Output Classes	AAMI	AAMI	AAMI	Severity-based
Finer Classification	No	No	No	Yes
Accuracy (%)	96.06	99.10	98.40	98.29
Energy per classification (μJ)	2.78	710.00	488.81*	0.11
Area (mm^2)	1.40	-	1.11*	0.11
Hardware Design Complexity	High	High	High	Low

tion and [59] which represents the most energy-efficient ECG classification for AAMI classes among state-of-the-art works. We also include [61] in Table 3.7 as its architectural approach (selectively turning off classification components) is close to our work.

The reported accuracy of 98.29% for the proposed ECG classification architecture in Table 3.7 is the accuracy of classifier-1 in Figure 3.10. This is because classifier-1 classifies the ECG beats only into the broad severity classes similar to the state-of-the-art works which classify the ECG beats into broad AAMI classes only and do not detect the actual arrhythmia classes. The accuracy comparison in Table 3.7 shows that we achieve classification accuracy on par with state-of-the-art ECG classification solutions. Even though the accuracies obtained by our work and state-of-the-art works are very similar, the classification boundaries addressed by our work are different than the state-of-the-art. For instance, the normal beat in MIT-BIH dataset belongs to AAMI class “N", while paced beat belongs to AAMI class “Q". However, both normal beat and paced beat in MIT-BIH dataset belong to the same “Normal" class in our severity-based ECG hierarchy. Thus, the accuracy results do not reflect a fair comparison. Hence, the emphasis should be on the fact that our work achieves good accuracy on severity-based classes, rather than comparing the absolute accuracy values.

For a heartbeat that belongs to a broad severity class or a broad AAMI class, finer classification refers to detecting its actual arrhythmia class. For instance, once we classify a beat into the broad severity class “Moderate”, then finer classification determines the arrhythmia class of that beat out of the five classes (“A”, “a”, “J”, “S”, and “E”) contained within the broad “Moderate” class. More details about finer classification can be found in Section 3.3.1. As shown in Table 3.7, only the proposed severity-based architecture provides such finer classification which can help doctors with faster diagnosis and treatment.

The energy consumption for various severity-based classes in our proposed hierarchical ECG classification architecture (Figure 3.10) is shown in Table 3.8. If the input beat falls into the “Normal” or “Mild” class, only classifier-1 is active which consumes 0.094 μJ. If the input beat gets classified as the “Moderate” class, both classifier-1 and classifier-2 get utilized consuming a total of 0.094 μJ + 0.095 μJ = 0.19 μJ. If the input beat is identified as the “Severe” class, all classifiers get utilized consuming a total of 0.094 μJ + 0.095 μJ + 0.094 μJ = 0.28 μJ which is the worst-case energy consumption for any single heartbeat in our architecture. For a fair comparison with state-of-the-art works in Table 3.7 which report average energy consumption, we derive the average energy

Table 3.8: Energy consumption and test set fraction for severity classes.

Severity Class	Energy per heartbeat classification	Heartbeats in test set (N)	Fraction of test set (N ÷ 21,891)
Normal	0.094 μJ	16410	0.75
Mild	0.094 μJ	3115	0.14
Moderate	0.19 μJ	577	0.03
Severe	0.28 μJ	1789	0.08

consumption for our architecture as the weighted average of the energy consumption across the severity classes. Here, the weight coefficients used for averaging indicate what fraction of total heartbeats in the test set (21,891) belongs to a specific severity class as shown in Table 3.8. This results in an average energy consumption of 0.11 μJ per heartbeat classification with a standard deviation of 0.052 μJ.

Table 3.7 shows that the proposed hierarchical ECG classifier consumes 25× less energy and 12× less area compared to state-of-the-art while keeping the accuracy benefits intact. Energy savings can be attributed to the fact that hierarchical architecture simplifies the design, activates hardware components only when necessary, and uses computation-in-memory which further improves energy efficiency. Area savings arise from the design simplification due to hierarchical architecture and high scalability of memristor devices.

We have presented an architecture for ECG classification which can be transformed into a hardware chip, where neural network algorithms are implemented as hardware components. Hence, the complexity comparison with state-of-the-art refers to the complexity of designing such a chip. The simple fully-connected (FC) neural network topology in our proposed architecture greatly simplifies the dataflow, storage of intermittent calculations, and control logic compared to complex network topologies in [59–61]. This results in the simplification of various chip design processes like placement, routing, and timing analysis resulting in faster chip development. Hence, our architecture greatly reduces the hardware design complexity compared to state-of-the-art as shown in Table 3.7.

3.6. CONCLUSIONS

This chapter proposed a severity-inclusive, accurate, and energy-efficient ECG classification, using hierarchical hardware architecture and computation-in-memory (CIM) paradigm. The hierarchical structure achieved high accuracy by breaking down the classification task into smaller subtasks and energy efficiency by activating its components only when needed. It also accounted for severity impact of arrhythmia classes to help the wearable device users and also assist medical professionals. We further performed design space exploration to implement the classification subtasks using neural networks which provide best balance between energy efficiency and accuracy on CIM hardware. The proposed ECG classification achieved 25× improvement in terms of average energy consumption and 12× improvement in terms of area compared to the state-of-the-art. Thus, we have shown that tailoring the computation architecture to characteristics of the application and the underlying hardware can lead to significant improvements in energy efficiency and area footprint.

4 Memristor-based CIM for Diabetic Retinopathy Screening

This chapter is based on [25, 90].

4.1. INTRODUCTION

Diabetic retinopathy (DR) refers to a condition where elevated glucose levels and blood pressure lead to irreversible retinal damage. It is a leading cause of permanent vision impairment across the globe, and the number of affected people is expected to reach 70 million by 2045 [18]. Moreover, every diabetic person is susceptible to the development of DR [19]. As the vision loss caused by DR is irreversible, detecting it at an early stage is crucial for timely treatment to prevent further retinal damage. Regular screening for DR is essential for such early detection. Recent advancements in artificial intelligence have paved the way for developing automated systems to provide fast, efficient, and convenient DR screening. These systems employ neural network-based DR classifiers to categorize retinal images into distinct screening classes, capitalizing on the inherent proficiency of neural networks in classification tasks.

The publicly available DR datasets exhibit inherent image inconsistencies to pose a tougher classification challenge than private ones, resulting in more robust and adaptive models. Moreover, their wider accessibility is valuable for driving further innovation in automated DR classification. Hence, we focus on DR classification literature based on publicly available datasets. Such works are susceptible to reliability issues, where the model performs well during development but exhibits poor performance upon deployment. This can arise due to several factors such as small training data size [19, 91–96], absence of external test data [97–101], and lack of diversity in training data [102, 103]. Furthermore, the inherent class imbalance in public datasets can bias the model performance towards majority classes. This can hinder the identification of minority DR classes (indicating retinal damage), further aggravating the reliability concern. Additionally, supplementary information about model prediction is crucial for widespread adoption of automated DR classification [101, 104]. For instance, when a DR model assists human specialists in double reading [105], supplementary information bolsters specialist's confidence when human diagnosis matches the model prediction, and helps resolve conflicts when these two differ. However, none of the aforementioned works provide supplementary information about model prediction. Lastly, deploying automated DR classification on portable edge devices can address the global scarcity of DR screening facilities [106, 107]. This requires energy-efficient hardware design of DR classifiers to achieve uninterrupted operation despite limited energy resources, enabling large-scale screening programs even in remote regions. However, the aforementioned literature only focuses on software model development while neglecting hardware design considerations. Hence, there exists a pressing need for hardware solution that facilitates reliable and energy-efficient DR screening at the edge.

In this work, we present a reliable and energy-efficient DR screening hardware targeting deployment on edge devices. We first develop a reliable DR classification model via training on a newly created custom dataset. This dataset encompasses image quality inconsistencies, diverse image sources, and reduced class imbalance. This enables our trained model to effectively handle real-world retinal images and perform well on minority classes, ensuring post-deployment reliability. Furthermore, we introduce a pseudo-binary classification scheme that internally uses multiclass classification to achieve binary screening. This enhances our model's classification performance and also provides supplementary information to aid its wider adoption. We then present energy-efficient hardware design of our model based on computation-in-memory (CIM) paradigm. It uses emerging memory devices known as memristors, to perform computations directly within the memory. This eliminates the data transfer bottleneck and provides superior energy efficiency suitable for edge device deployment. Our key contributions are as follows:

- We develop a reliable DR classification model by using inconsistent quality images collected from diverse sources and addressing the class imbalance problem.
- We propose a pseudo-binary classification scheme to improve the classification performance and provide a more informative classification output.
- We present an energy-efficient hardware design for our DR classification model using memristor-based CIM to facilitate its deployment on edge devices.

Simulation results show that we achieve reliable DR classification while consuming three orders of magnitude less energy compared to the state-of-the-art hardware platforms.

4.2. DIABETIC RETINOPATHY

4.2.1. BASICS

Diabetic retinopathy is an irreversible condition arising from elevated glucose levels and hypertension. It damages blood vessels in the retina and can potentially cause permanent vision impairment. Severity of DR is assessed based on the presence of specific features, known as lesions, within the retina. Figure 4.1 depicts the four most common lesions: microaneurysms, hemorrhages, hard exudates, and soft exudates. They can be described as follows [19, 108]:

- Microaneurysms (MA): These are the earliest visible signs of retinal damage. They manifest as tiny red dots arising due to capillary dilation.
- Hemorrhages (HM): These are red spots with irregular margins and/or uneven density. They are bigger than MA and occur due to leakage of weak capillaries.

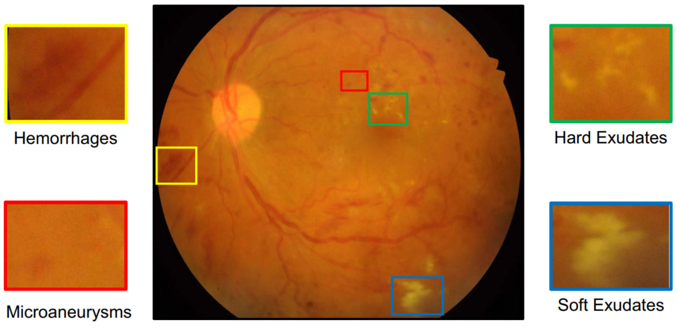


Figure 4.1: Retinal image annotated with the four most common lesions: microaneurysms, hemorrhages, hard exudates, and soft exudates [108].

Table 4.1: Lesion-based diagnosis of DR classes.

DR Severity Level	DR Class	Lesion-based Diagnosis
No DR	DR-0	No lesions
Mild non-proliferative	DR-1	Only MA present
Moderate non-proliferative	DR-2	MA and other lesions present Less prominence than DR-3
Severe non-proliferative	DR-3	At least one of the following present: HM (>20 in each quadrant), Blood spillage (>2 quadrants), No indicators of DR-4
Proliferative	DR-4	Vitreous/preretinal HM and/or NV

- Hard exudates (HE): These are yellow-white deposits in outer layers of retina caused by leakage of plasma.
- Soft exudates (SE): These are greyish oval or round-shaped patches arising due to the swelling of the nerve fiber. They are also called cotton wool spots.
- Neovascularization (NV): It refers to the abnormal growth of new blood vessels on the inner surface of the retina. Such blood vessels often bleed into the vitreous cavity and lead to obscured vision.

Table 4.1 provides mapping of the five DR severity levels (classes) defined in international standards [109] to the composition of retinal lesions.

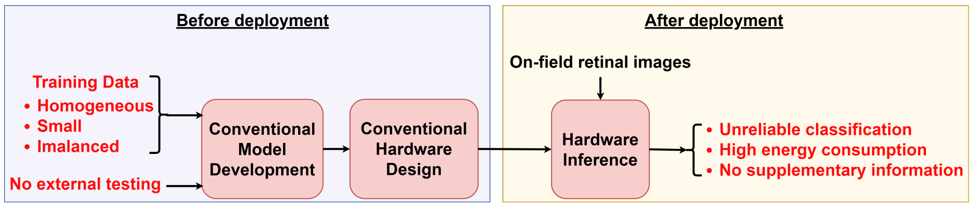
4.2.2. DETECTION

Conventional DR detection typically begins with pre-capture medical procedures on patients to enlarge their pupils and facilitate better coverage of the retinal area during image capture. Skilled operators then employ specialized fundus cameras and meticulously adjust settings such as focus, exposure, alignment, etc., to capture high-quality retinal images. Subsequently, the severity of DR is evaluated through visual inspection of various lesions within the captured retinal image. The advancements in artificial intelligence have opened avenues to employ automated systems to achieve DR identification from retinal images. They leverage neural networks, which inherently excel at extracting crucial lesion information from retinal images and autonomously categorize them into distinct DR classes. Thus, neural network-based DR classification systems offer an effective and efficient approach to DR detection.

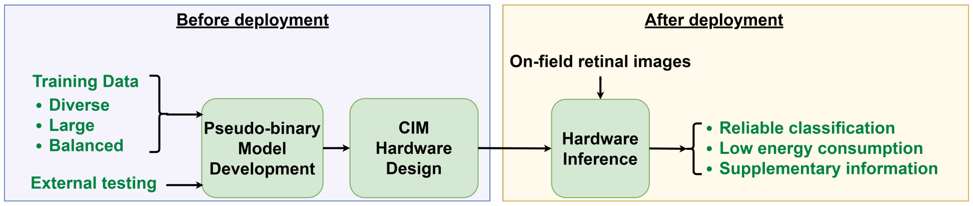
4.3. PROPOSED METHODOLOGY

4.3.1. OVERVIEW OF DR SCREENING APPROACHES

An overview of both conventional and proposed approaches for developing neural network-based DR screening solutions is shown in Figure 4.2. They both involve two phases: 1) pre-deployment phase, where the model is trained and hardware is designed for the


Before deployment
Training Data
• Homogeneous
• Small
• Imalanced
No external testing
Conventional Model Development
Conventional Hardware Design
After deployment
On-field retinal images
Hardware Inference
• Unreliable classification
• High energy consumption
• No supplementary information

(a) Conventional Approach



(b) Proposed Approach

Figure 4.2: Overview of conventional and proposed neural network-based DR screening approaches.

trained model, and 2) post-deployment phase, where the hardware performs inference using on-field images. Models developed using conventional approach are susceptible to reliability issue, where they perform well during development but fail after deployment. This is a consequence of model's poor generalization ability arising from small-sized, non-diverse and imbalanced training data, coupled with absence of external test data. Moreover, they do not provide supplementary information about the model prediction. This severely limits their widespread adoption by both specialists and patients. Additionally, there has been almost no effort directed towards energy efficient hardware design for DR classification models targeting edge device deployment, which is critical to improve their global accessibility.

Our proposed approach overcomes all of these challenges. We first create a large, diverse and balanced custom dataset by combining data from multiple sources and taking measures to reduce class imbalance. We then train our DR classification model with this dataset and also assess the model reliability with external test data. Moreover, we propose a pseudo-binary classification scheme that improves the model performance and also provides supplementary information to facilitate its widespread adoption. Furthermore, we present energy-efficient hardware design for our model using memristor-based CIM, to facilitate its deployment on edge devices for improved accessibility. Thus, we provide a solution that offers reliable DR classification, supplementary information and energy efficient edge deployment. We will now delve into the details of our approach in the next subsections.

4.3.2. RELIABLE MODEL DEVELOPMENT

DATASET CREATION

The DR classification models often encounter inconsistent quality retinal images post-deployment. This arises due to various factors such as improper exposure, misalignment, incomplete retinal coverage etc. Furthermore, the distribution of post-deployment data can diverge substantially from the data used during model development. Hence, the model must be trained using data that encompasses these inconsistencies and reflects diversity of post-deployment data to ensure reliability. We build such a comprehensive training dataset by leveraging the following publicly available datasets:

- EyePACS dataset [111]: It is provided by EyePACS Inc. for DR detection competition sponsored by California Healthcare Foundation in 2015. It contains 88,702 images collected from different parts of the USA.
- DDR dataset [101] : It contains 13,673 images collected across 147 hospitals in China from 2016 to 2018. The dataset actually has 12,522 usable images as 1,151 images are deemed ungradable.
- APTOS dataset [110]: It is a part of DR detection competition organized by Asia Pacific Tele-Ophthalmology Society in 2019. It contains 3662 retinal images provided by Aravind Eye Hospital in India.

After acquiring these datasets, we filter out corrupt images and merge them in varying proportions to create three merged datasets. We then undersample the majority classes in each merged dataset to limit class imbalance to 10× or less. This is because 10× or less imbalance suffices for neural networks perform well on minority classes [112], and achieving perfect class balance is impractical due to huge number of healthy retina images. As a result, we end up with three new proposed datasets: Small (S), Medium (M), and Large (L). The classwise distributions of the original and new datasets is shown in Table 4.2. It can be seen that the new M and L datasets exhibit better class balance than original EyePACS and DDR datasets. Moreover, merging enhances the data diversity in M and L datasets compared to EyePACS and DDR datasets. As a result, models developed using M and L datasets can potentially exhibit better reliability in handling both on-field image inconsistencies and minority DR classes. Furthermore, the three datasets can be used to obtain crucial insights into how dataset size influences classification performance.

Table 4.2: Overview of the original public datasets and newly proposed datasets. Imbalance is the ratio of the sizes of the largest and smallest class.

	Original Datasets			**Proposed Datasets**		
	APTOS [110]	**DDR** [101]	**EyePACS** [111]	**Small (S)**: APTOS	**Medium (M)**: APTOS & DDR	**Large (L)**: APTOS, DDR & EyePACS
DR-0	1805	6266	65343	1798	3000	10000
DR-1	370	630	6205	365	991	7180
DR-2	999	4477	13153	991	3000	10000
DR-3	193	236	2087	188	424	2504
DR-4	295	913	1914	292	1204	3117
Total	3662	12522	88702	3634	8619	32801
Imbalance	9.4×	26.5×	34.1×	9.6×	7.1×	4.0×

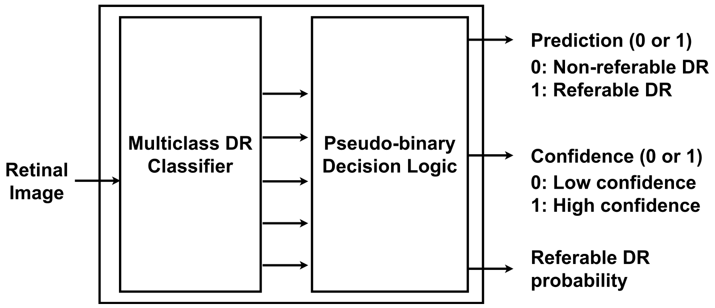


Figure 4.3: Pseudo-binary classification concept.

PSEUDO-BINARY CLASSIFICATION

The recommended management guidelines for various DR classes are as follows [113]:

- Annual screening for DR-0 or DR-1.
- A follow-up every six months for DR-2.
- Referral to an ophthalmologist for DR-3 or DR-4.

Thus, the recommended DR management approach shifts from annual screening to more frequent monitoring as the severity reaches DR-2. Consequently, grouping the five DR classes into the following two categories suffices for screening [103]: non-referable DR (consisting of DR-0 and DR-1) and referable DR (consisting of DR-2, DR-3, and DR-4). Thus, DR screening becomes a binary classification task involving referable DR and non-referable DR as its two classes.

A straightforward approach to binary screening involves relabeling the five original classes into these two broad categories and developing a binary classification model. However, this leads to a hard decision between the two categories which is more susceptible to misclassifications. It also weakens the interpretability by hindering the derivation of supplementary information. To alleviate this problem, we introduce an approach called pseudo-binary classification. It internally employs a multiclass DR classifier and uses additional decision-making logic to ultimately produce a binary classification outcome, as shown in Figure 4.3. It capitalizes on cumulative probabilities within non-referable (0) and referable (1) categories instead of hinging on a single maximum probability for decision-making, reducing susceptibility to misclassifications. Moreover, it presents the outcome as a tuple containing prediction, confidence level, and referable DR probability, providing better interpretability.

Algorithm 4.1 describes the pseudo-binary classification process. It begins with multiclass classification to obtain prediction probabilities for the five original DR classes. Subsequently, it calculates a score for the non-referable (0) class by adding the probabilities of DR-0 and DR-1. Similarly, a score for the referable (1) class is computed by adding the probabilities of DR-2, DR-3, and DR-4. The broad class (referable 0 or non-referable 1) with the higher score is selected as the prediction value. Furthermore, if the scores differ by more than a predefined confidence threshold, we indicate high confidence ('H'); otherwise low confidence ('L'). Additionally, the referable class score indicates the probability

Algorithm 4.1: Pseudo-binary classification algorithm.

```
input  : Confidence threshold $C_{th}$, retinal image $I$
output : Prediction tuple $P$
1  softmax ← multiclass_inference($I$);
2  non_ref_score ← softmax(DR-0) + softmax(DR-1);
3  ref_score ← softmax(DR-2) + softmax(DR-3) + softmax(DR-4);
4  $\Delta$ ← ref_score - non_ref_score;
5  if $\Delta > 0$ then
6      prediction ← 1;
7      if $\Delta > C_{th}$ then
8          confidence ← H;
9      else
10         confidence ← L;
11 else
12     prediction ← 0;
13     if $|\Delta| > C_{th}$ then
14         confidence ← H;
15     else
16         confidence ← L;
17 $P$ ← (prediction, confidence, ref_score);
18 return $P$;
```

Table 4.3: Interpretation of the pseudo-binary prediction tuples. Here, S denotes probability of referable DR.

Prediction Tuple	**Interpretation**
(0, H, S)	Healthy (no DR).
(0, L, S)	DR developing, checkup recommended.
(1, L, S)	DR found, seek medical help soon.
(1, H, S)	DR found, seek medical help immediately.

of referable DR. For example, consider a scenario with confidence threshold 0.25 and softmax probabilities as [0.20 (DR-0), 0.33 (DR-1), 0.4 (DR-2), 0.03 (DR-3), 0.04 (DR-4)]. This leads to a pseudo-binary prediction tuple as (0, L, 47%), indicating that the patient has non-referable DR (class 0), detected with low confidence (L) and a 47% likelihood of referable DR. Thus, the patient appears to be developing DR-1 and is recommended to have a checkup in the near future. The interpretation of the various output tuples resulting from pseudo-binary classification is summarized in Table 4.3, augmenting the prediction with supplementary information.

4.3.3. ENERGY-EFFICIENT HARDWARE DESIGN

Pruning and Quantization

Before mapping our trained pseudo-binary DR classification model to CIM hardware, we perform pruning and quantization to reduce its hardware resource requirements. Pruning refers to selectively removing a user-defined portion of low-magnitude weights from each layer. The reduction in hardware resource requirements due to pruning often comes at the cost of accuracy degradation. To counter this, we adopt pruning followed by retraining to recover lost accuracy. An essential consideration for such post-pruning retraining is the selection of hyperparameters, particularly the learning rate. An excessively low learning rate can hinder the network's adaptability to recover the pruning-induced accuracy loss. Hence, we dynamically adjust the learning rate within a narrow range centered around its original value during the retraining process. This iterative cycle continues until we achieve the desired level of pruning while preserving the network's original accuracy. We then quantize the weights of the pruned model to further reduce hardware resource requirements. However, an aggressive quantization can lead to high quantization error and degraded classification performance. Hence, we adopt a design space exploration approach to minimize bit-sizes for weights while ensuring minimal impact on classification performance.

Mapping to CIM Architecture

We map our pruned and quantized pseudo-binary DR classification model to the memristor-based CIM architecture described in [30]. The fundamental building block of this architecture is depicted in Figure 4.4. It divides the full-precision neural network weights and inputs into smaller slices. This is because memristors have limited bit capacity and high-resolution data converters (DACs and ADCs) consume significant energy and area.

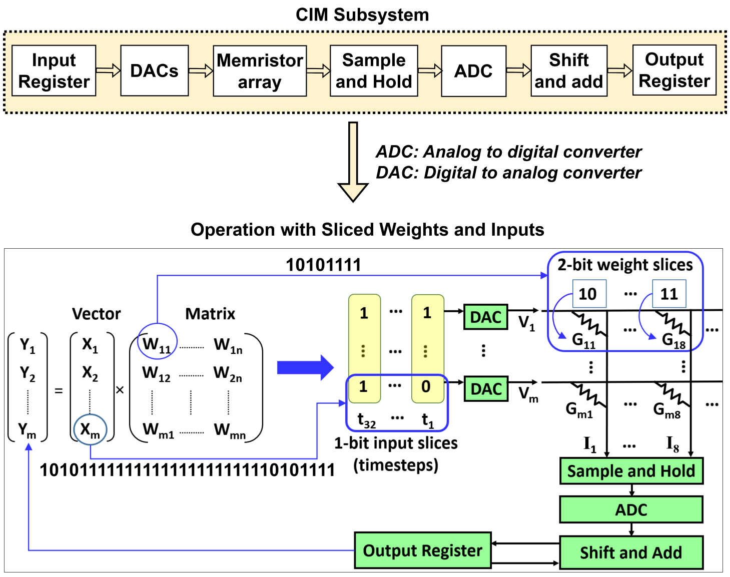


Figure 4.4: CIM hardware architecture for DR model implementation.

We transform 2-bit slices of the weights into conductance values, which are then mapped onto distinct columns within the memristor crossbar. We also convert 1-bit slices of the inputs into voltages that are applied to the crossbar at different timesteps. For instance, with 1-bit DACs for 32-bit digital inputs, the DACs are fed with 1-bit at a time across 32 timesteps. The DACs convert the bits at each timestep into voltages, generating a current in each column of the crossbar. These currents are captured by sample and hold circuits (S&H) and then converted into digital outputs by ADCs. To account for the slicing of weights across crossbar columns, a shift and add operation is performed across the columns for the ADC outputs. Furthermore, an additional round of shift and add operations is performed to merge such partial outputs from various timesteps to produce the final full-precision digital output.

4.4. SIMULATION SETUP

4.4.1. PERFORMANCE METRICS

The key performance metrics for evaluation of the proposed DR classification are:

- **Accuracy**: It is defined as the ratio of the number of correctly classified retinal images to the total number of input retinal images, expressed as a percentage.
- **F1-score**: While accuracy is a valuable indicator of overall classification performance, there is a need for metrics that delve deeper into the model's behavior. The F1-score is one such metric that reflects the model's ability to make correct predictions while keeping false alarms to a minimum. It is calculated using a table called the confusion matrix with true labels as column headers and predicted labels as row headers, shown in Figure 4.5.

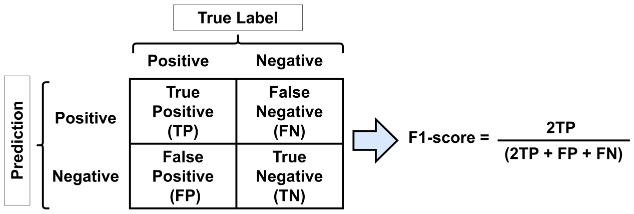


Figure 4.5: F1-score calculation from confusion matrix representation.

- **Energy consumption**: Deployment of automated DR screening on portable edge devices can significantly improve its global accessibility, even in remote areas. To achieve this, such devices must be able to operate with limited and interrupted energy availability. Hence, energy consumed by a DR classification hardware is an important performance metric.

4.4.2. SIMULATION PLATFORM

The overview of the simulation platform is shown in Table 4.4. We use TensorFlow [114] framework for developing our DR classification model. Inception-v3 (IV3) [115] and DenseNet121 (DN121) [116] neural network architectures are selected for exploration, as

Table 4.4: Simulation platform details.

Component	Specification
Deep learning framework	TensorFlow [114]
Network architectures	Modified inception-v3 (IV3) Modified densenet121 (DN121)
Datasets	New merged datasets (in Table 4.2): Small (S), Medium (M), Large (L)
CIM hardware	ISAAC [30]
Conventional hardware	CPU: Intel Core i7-9750H [118] GPU: NVIDIA GeForce GTX 1650 [119] mTPU: Google Edge TPU on Coral dev board [120]
Power profiling tools	CPU: s-tui [121] GPU: nvidia-smi [122] mTPU: datasheet [123] CIM: data provided in [30]
Latency profiling tools	CPU: Tensorflow profiler [124] GPU: Tensorflow profiler [124] mTPU: Python datetime package [125] CIM: data provided in [30]

they have demonstrated remarkable performance on complex image datasets [117]. We adapt these architectures for DR classification by introducing three new fully-connected layers (IV3: 2048×128, 128×128, 128×5 and DN121: 1024×128, 128×128, and 128×5) and dropout layers (probability 0.5). We train these networks with each of our new S/M/L datasets, following the train-validation-test split shown in Table 4.5. Employing transfer learning, we only train the newly added fully connected layers while freezing the pre-trained weights from the ImageNet dataset for all the other layers. During this training phase, we perform grid search followed by manual fine-tuning to establish optimal values for the hyperparameters. The post-training model performance is evaluated with the corresponding S/M/L test set.

To evaluate the reliability of the trained models, we employ publicly accessible Messidor-2 dataset [126–128] as an external training set. It contains 1748 images where 1058 images are provided by the Messidor program partners [126] and the remaining are collected at Brest University Hospital in France between 2009 and 2010. The labels for Messidor-2 dataset are sourced from [129], following the study in [130]. This labeling process has deemed four images ungradable, leaving 1744 usable images with classwise

Table 4.5: Training-validation-test split for the newly proposed datasets: Small (S), Medium (M) and Large (L).

Dataset	Training Set					Validation Set					Test Set				
	DR-0	DR-1	DR-2	DR-3	DR-4	DR-0	DR-1	DR-2	DR-3	DR-4	DR-0	DR-1	DR-2	DR-3	DR-4
Small (S)	1059	229	589	112	191	382	67	197	33	48	357	69	205	43	53
Medium (M)	1775	598	1808	257	733	620	202	565	87	250	605	191	627	80	221
Large (L)	6021	4314	5992	1465	1887	2034	1432	1975	517	603	1945	1434	2033	522	627

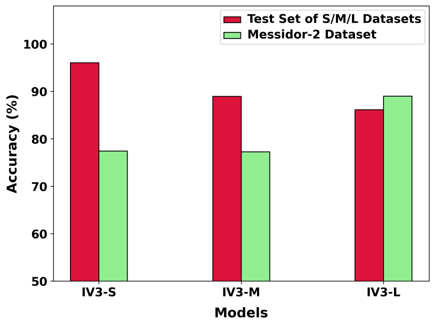
distribution as follows - DR-0: 1017, DR-1: 270, DR-2: 347, DR-3: 75, and DR-4: 35. As Messidor-2 embodies a data characteristics distinct from S/M/L train-validation-test sets, model performance on Messidor-2 serves as an indicator of its reliability.

To map the trained reliable model onto the memristor-based CIM hardware, we employ the architecture described in Section 4.3.3. We leverage power consumption and latency data as presented in [30] to assess CIM energy consumption. This energy consumption is then compared against three conventional state-of-the-art hardware platforms: CPU (Intel Core i7-9750H [118]), GPU (NVIDIA GeForce GTX 1650 [119]), and mTPU (Google Edge TPU on Coral development board [120]). CPU and GPU represent general-purpose conventional hardware, while the mTPU embodies AI-optimized conventional hardware. To quantify energy consumption across these conventional hardware platforms, we first measure their latency and power consumption, and then calculate energy consumption as a product of these two values. The latency for both CPU and GPU is obtained via TensorFlow profiler [124], while that for mTPU is measured using Python datetime package [125]. The power consumption of the CPU is recorded using s-tui [121], while nvidia-smi [122] is used to record GPU power consumption. We use mTPU's datasheet to obtain its power consumption [123].

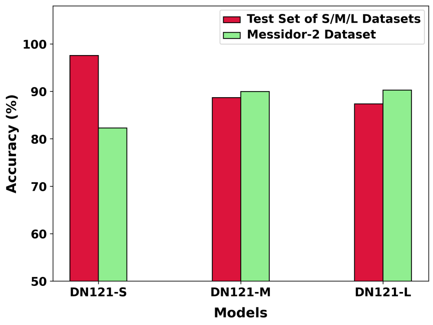
4.5. SIMULATION RESULTS

4.5.1. MODEL RELIABILITY ASSESSMENT

We train IV3 and DN121 networks across our three new datasets (S, M, and L in Table 4.2) by employing the pseudo-binary classification approach. This yields six distinct models: IV3-S (IV3 network trained on S dataset), IV3-M (IV3 network trained on M dataset), IV3-L (IV3 network trained on L dataset), DN121-S (DN121 network trained on S dataset), DN121-M (DN121 network trained on M dataset), and DN121-L (DN121 network trained on L dataset). The development phase performance of these models is assessed as their accuracy and F1-score on the corresponding S/M/L test set. To emulate post-deployment scenarios, we evaluate their accuracy and F1-score on Messidor-2 as an external test dataset. A reliable neural network model should exhibit consistent accuracy and F1-

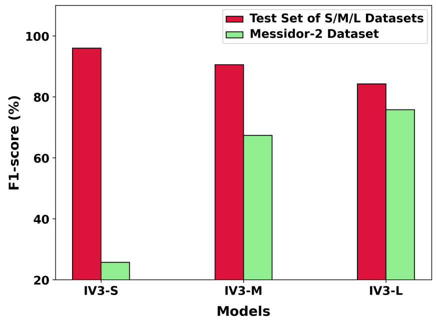


(a) IV3 accuracy.

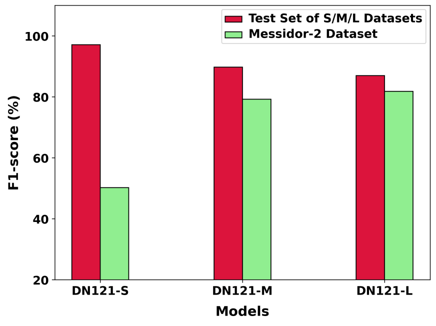


(b) DN121 accuracy.

Figure 4.6: Reliability assessment of IV3 and DN121 models in terms of accuracy.



(a) IV3 F1-score



(b) DN121 F1-score.

Figure 4.7: Reliability assessment of IV3 and DN121 models in terms of F1-score.

score, both during the development phase (on S/M/L test sets) and in post-deployment situations (on Messidor-2 dataset). This criterion forms the basis to assess the reliability of these six models in Figures 4.6 and 4.7.

Models developed using S dataset (IV3-S and DN121-S) show commendable performance on S/M/L test sets but struggle on Messidor-2, indicating low reliability. The other models trained with M and L datasets exhibit a marked improvement in reliability, consistently maintaining robust performance on both S/M/L test sets and Messidor-2 data. Also, model reliability improves as we transition from the M to the L dataset. This can be observed as IV3-L and DN121-L models outperform their M dataset counterparts IV3-M and DN121-M. This also highlights the pivotal role of large datasets in ensuring model reliability. As DN121-L exhibits better reliability than IV3-L in terms of both accuracy and F1-score, it becomes our final choice.

We now compare the reliability of our DN121-L model with other works from the literature. While several works [91, 94, 95, 97] have conducted evaluations on Messidor-2 dataset, they incorporate it within the training data rather than exclusively reserving it for testing. Consequently, a fair comparison with such studies is not possible. Additionally, works like [102] use datasets other than Messidor-2 for external testing and cannot be directly compared with our work. Therefore, we compare our approach with [103, 131] which use Messidor-2 dataset only for external testing. As shown in Table 4.6, DN121-L achieves competitive accuracy and F1-score despite testing on 65% more Messidor-2

Table 4.6: Comparison with other works using Messidor-2 as external testing dataset. Notation ‘N/A’ indicates that the corresponding value is not available from the paper.

Screening Approach	**Test Images**	**Accuracy (%)**	**F1-score (%)**
This Work (DN121-L)	1744	90.3	81.8
Blair et al. [131]	1054	93.5	N/A
Ludwig et al. [103]	1058	N/A	83

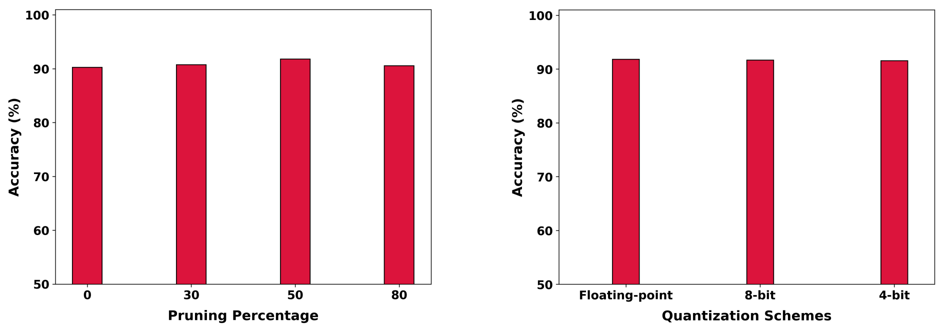
images compared to [103, 131], validating its reliability. We will discuss CIM hardware design for this reliable DN121-L model next.

4.5.2. HARDWARE DESIGN FOR RELIABLE MODEL

In this subsection, we first optimize the reliable DN121-L model for hardware design through pruning and quantization, and then assess its energy consumption on CIM hardware. We will now delve into the details of these steps.

PRUNING AND QUANTIZATION

Figure 4.8a shows the impact of pruning on DN121-L model. The pruning percentage indicates the fraction of model parameters with low magnitudes that undergo removal during the pruning process. Following the removal of the specified fraction of parameters, we conduct retraining for a few epochs to recover the accuracy loss incurred during pruning. Observing the classification performance across various pruning percentages, it becomes evident that the 50% pruned version of DN121-L exhibits the best classification performance. We select this version, denote it as DN121-L-P50 and subject it to weight quantization. Figure 4.8b shows that we can use 4-bit weights with almost no accuracy loss. Hence, we select the 4-bit quantized version of DN2121-L-P50 and denote it as CIM-DN121. We will analyze its energy consumption on CIM hardware next.



(a) Pruning of DN121-L model.

(b) Quantization of DN121-L-P50 model.

Figure 4.8: Impact of pruning and quantization on model accuracy.

ENERGY EFFICIENCY ASSESSMENT

The energy consumption for CIM-DN121 model, along with that of DN121 on state-of-the-art conventional hardware platforms like CPU, GPU, and edge TPU (mTPU) is depicted in Figure 4.9. It quantifies the energy required for executing inference on a single retinal image. The mTPU turns out to be the least energy-efficient despite being designed for AI applications. This is because mTPU's efficiency is limited by other resources on Coral dev board which handle tasks such as code context management and input/output data processing. For a large neural network model like DN121, these resources become the bottleneck, leading to significantly longer execution latency and increased energy consumption compared to CPUs or GPUs. Therefore, mTPU dev board may not be the

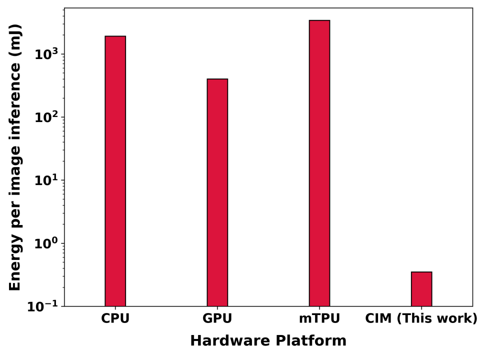


Figure 4.9: Energy per image inference for various hardware platforms.

best choice for energy-efficient execution of large neural network models. On the other hand, CIM-DN121 demonstrates 5441× reduction in energy consumption compared to CPU. Furthermore, it consumes 1144× and 9686× less energy compared to GPU and mTPU respectively. This highlights the tremendous potential of memristor-based CIM for developing energy-efficient hardware for DR screening.

4.6. CONCLUSIONS

This chapter presented a reliable and energy-efficient hardware design for DR screening. We accomplished reliable classification by training the model with diverse and inconsistent quality data, while addressing class imbalance issue. We then proposed a pseudo-binary classification technique to further improve the model performance and provide supplementary information. Furthermore, we explored energy-efficient hardware design for our reliable DR model targeting deployment on edge devices for enhanced healthcare accessibility. Our DR screening solution based on DenseNet121 model achieved reliable classification with three orders of magnitude less energy consumption compared to the state-of-the-art hardware platforms. Thus, our work has laid the groundwork for reliable and accessible healthcare through the intersection of technology and medical science.

PART-III

Dealing with Memristor Non-idealities

5

Mapping-aware Biased Training for CIM-based Neural Networks

This chapter is based on [26].

5.1. INTRODUCTION

Memristors suffer from conductance variation problem where their programmed conductance deviates from the target value, due to fabrication imperfections and stochastic device physics [10]. This leads to an undesired change in the neural network weights stored as memristor conductances, resulting in low accuracy. Prior works addressing the conductance variation issue in CIM-based neural networks can be grouped into four categories: i) on-chip training, ii) off-chip training or mapping based on hardware characterization, iii) hardware compensation, and iv) write-verify programming. First, on-chip training inherently adapts the weights to conductance variation as the network is trained on the CIM chip [132, 133]. However, it is not scalable due to individual training necessity for each chip, high energy consumption, and endurance issues. Second, off-chip training using a hardware-calibrated software model of conductance variation [134, 135] is also not scalable, as each chip requires individual characterization and training. Moreover, some works [20, 136] prevent large weights from mapping to high variation memristors. This requires extensive chip characterization and does not address errors due to the accumulation of variations in small weights. Alternatively, noise estimated from a non-extensive chip characterization can be injected in off-chip training [137–142] to enhance the network's tolerance towards errors due to conductance variation. However, this approach fails to address the issue of reducing such errors, as memristors can still get mapped to high variation conductance states, rendering it ineffective. Last, the hardware compensation and write-verify programming involve significant energy and area overheads with increased design complexity [143–146]. Hence, there is a strong need for an effective, scalable, and low-overhead solution to mitigate conductance variation impact on CIM-based neural networks.

This work presents a mapping-aware biased training methodology to improve the accuracy of CIM-based neural networks in the presence of conductance variation. We first identify memristor conductance states with low variation impact (favorable states). We then derive a favorability constraint that only allows weight values that map to these favorable states. During training, we determine which weights are important for CIM hardware accuracy and impose the favorability constraint on them. The resulting post-training values of these important weights then directly map to favorable states, leading to high inference accuracy on CIM hardware. Our key contributions can be summarized as follows:

- A favorability constraint analysis to find the weight values desirable for reducing conductance variation errors.
- An approach to identify the important weights which significantly influence the hardware accuracy.
- A mapping-aware biased training with favorability constraint on important weights for high hardware accuracy.

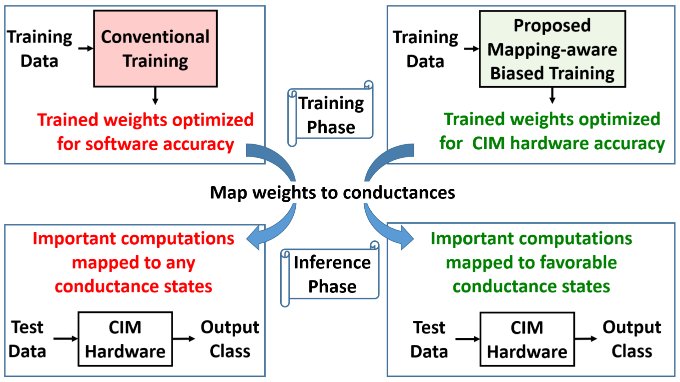

Training Data
Conventional Training
Trained weights optimized for software accuracy
Training Data
Proposed Mapping-aware Biased Training
Trained weights optimized for CIM hardware accuracy
Training Phase
Map weights to conductances
Important computations mapped to any conductance states
Test Data
CIM Hardware
Output Class
Inference Phase
Important computations mapped to favorable conductance states
Test Data
CIM Hardware
Output Class

Figure 5.1: Overview of the conventional and proposed training methodologies.

5.2. PROPOSED METHODOLOGY

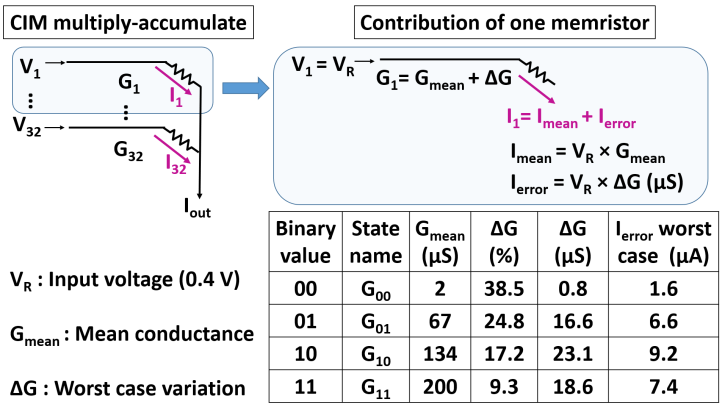
5.2.1. OVERVIEW OF TRAINING APPROACHES

The deployment of a neural network on CIM hardware for inference involves two phases as shown in Fig. 5.1: i) Training the neural network weights to obtain high classification accuracy. ii) Mapping the trained weights to memristor conductances for inference on CIM hardware. Conventional training can result in the mapping of weights to conductance states having a high variation impact (unfavorable states). This can lead to low hardware accuracy despite high software accuracy. Our proposed mapping-aware biased training restricts the neural network weights during training, so that their post-training values directly get mapped to conductance states having a low variation impact (favorable states). However, restricting too many weights hinders backpropagation and leads to low software accuracy. This in turn results in low hardware accuracy, as it is upper bounded by software accuracy. Conversely, if too few weights are restricted, the hardware accuracy will be poor as many memristors can get mapped to unfavorable states. Hence, our proposed mapping-aware biased training only restricts the important weights. This leads to high software accuracy due to the adaptability of non-important weights and also provides high hardware accuracy as important weights get mapped to favorable memristor states.

5.2.2. FAVORABLE CONDUCTANCE STATES ANALYSIS

Fig. 5.2 shows a CIM-based multiply-accumulate operation, where I_{error} is the error current in a single memristor device due to conductance variation. As small I_{error} is desirable, the preference order of states in Fig. 5.2 is: G_{00} (best), G_{01}, G_{11}, G_{10} (worst). Despite having a higher variation percentage, G_{00} and G_{01} are preferred over G_{11} and G_{10} as their small mean values result in small I_{error}. Hence, the preference order of conductance states must be based on I_{error} contribution instead of the variation percentage. The ordered conductance states are then grouped into: i) unfavorable states (U) to avoid for mapping, and ii) favorable states (F) to prefer for mapping. Based on Fig. 5.2, the possible grouping configurations are:

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


CIM multiply-accumulate
Contribution of one memristor
V_1
G_1
I_1
V_{32}
G_{32}
I_{32}
I_{out}
$V_1 = V_R$
$G_1 = G_{mean} + \Delta G$
$I_1 = I_{mean} + I_{error}$
$I_{mean} = V_R \times G_{mean}$
$I_{error} = V_R \times \Delta G$ (µS)
V_R : Input voltage (0.4 V)
G_{mean} : Mean conductance
ΔG : Worst case variation

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

Figure 5.2: Favorable conductance states analysis for a 2-bit memristor (four conductance states). The used conductance variation data is obtained from [147].

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

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

We now determine a favorability constraint on the weights to ensure the mapping of desired weight bits to favorable conductance states. This constraint depends on memristor bit capacity and CIM mapping scheme details like fixed-point format, underlying CIM architecture, etc. For example, consider 2-bit memristors (slices), 8-bit fixed-point weights (6-bit fraction), and CIM architecture in [30]. The mapping scheme first converts trained

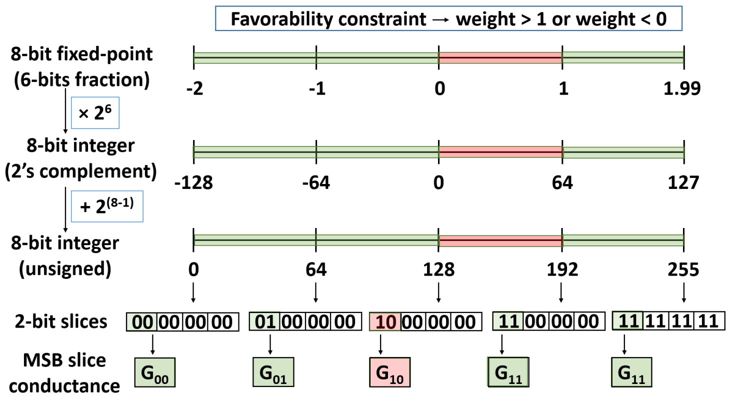


Figure 5.3: Illustration of favorability constraint derivation for mapping MSB slice of 8-bit weight to favorable conductance states in a 2-bit memristor.

weights to 2's complement fixed-point format. It then shifts the 2's complement weight range by 2^7 to overcome the difficulty in isolating the sign contribution from a multi-bit slice [30]. Fig. 5.3 then shows the favorability constraint to map the most significant 2-bit slice to favorable states in Section 5.2.2 (G_{00}, G_{01}, and G_{10}) for this example.

5.2.3. BIASED TRAINING ALGORITHM

The flowchart of our mapping-aware biased training is shown in Fig. 5.4. We first train the neural network in a standard (hardware-unaware) manner. These weights are used as initial weights for mapping-aware biased training for faster convergence. After this, we determine the favorability constraint on the weights for mapping to variation-immune states. We now perform a new epoch of backpropagation using training data and then determine which weights are important for high hardware accuracy. In a neural network, some weights have more importance than others for high software accuracy. However, in CIM hardware design for the same network, instead of individual weights, some crossbar columns (groups of weights) are more important than others for high hardware accuracy. This is because the basic computation in CIM is the column-wise multiply-accumulate operation. Let HI_c denote the importance of a CIM column for high hardware accuracy and SI_w denote the importance of a weight for high software accuracy. If P denotes the network output (without softmax) and L denotes the one hot label, then SI_w is given by Eq. 5.1.

$$\mathrm{SI}_w = \frac{\partial Q}{\partial w}, \text{ where } Q = \sum_{i=1}^{\text{Batch size}} P_i \times L_i \tag{5.1}$$

HI_c is then obtained by dividing the software weight matrix into crossbar-sized chunks and adding SI_w of weights per column across all chunks. A high HI_c value indicates

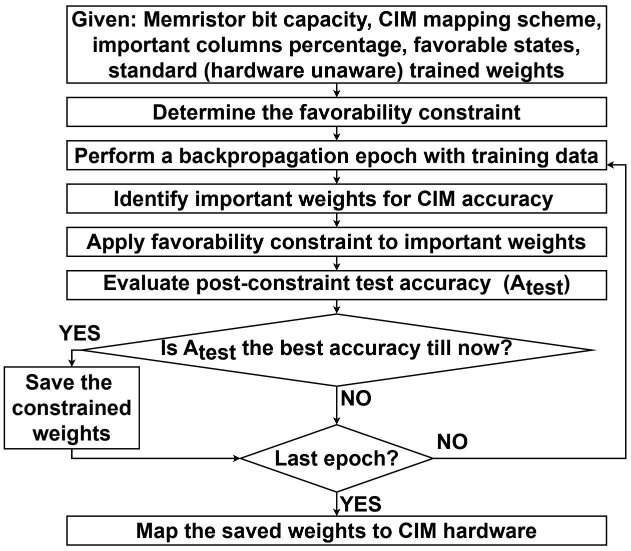

Given: Memristor bit capacity, CIM mapping scheme, important columns percentage, favorable states, standard (hardware unaware) trained weights
Determine the favorability constraint
Perform a backpropagation epoch with training data
Identify important weights for CIM accuracy
Apply favorability constraint to important weights
Evaluate post-constraint test accuracy (Atest)
Is Atest the best accuracy till now?
YES
Save the constrained weights
NO
Last epoch?
NO
YES
Map the saved weights to CIM hardware

Figure 5.4: Flowchart of the proposed mapping-aware biased training.

more influence on hardware accuracy. We now select m% columns with the highest HI_c (m is obtained by design-space exploration, details in Section 5.4.1). Weights in these columns are restricted as per the favorability constraint and test accuracy is evaluated. This process is repeated for a given number of biased training epochs and weights with the best post-restriction test accuracy are mapped to CIM hardware.

5.3. SIMULATION SETUP

We have developed a Python-based framework for behavioral simulation of neural network inference on CIM hardware. It is based on in-situ multiply-accumulate (IMA) unit in state-of-the-art CIM architectures [30, 31]. Power and area for various IMA components are also obtained from [30]. We consider 8-bit weights split across four memristors of 2-bit capacity. Memristor device parameters and conductance variation data are obtained from [147] which presents experiments on real memristor devices. We have performed evaluations using MNIST [85], Fashion MNIST (FMNIST) [148], and EMNIST letters (EMNIST-L) [149] datasets on LeNet-5 neural network [85]. The mapping-aware biased training is carried out in software and trained weights are used in our Python-based framework to evaluate the hardware inference accuracy.

5.4. SIMULATION RESULTS

5.4.1. NEURAL NETWORK ACCURACY

We perform design space exploration to determine the optimal percentage of crossbar columns, which are designated as important and subjected to favorability constraints in all neural network layers, as depicted in Fig. 5.5. A high percentage results in low software accuracy as restricting more weights obstructs the minimization of the cost function. The hardware accuracy is also reduced as it is upper bounded by software accuracy. A moderate percentage can provide high accuracy in both software and hardware by bal-

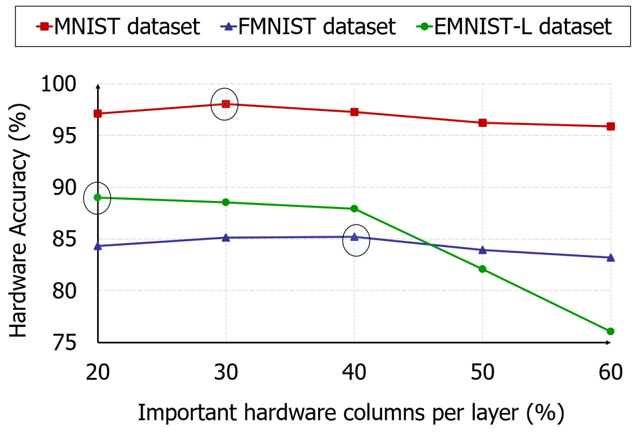


Figure 5.5: Design-space exploration for the percentage of important columns per neural network layer. Circle denotes the peak hardware accuracy per dataset.

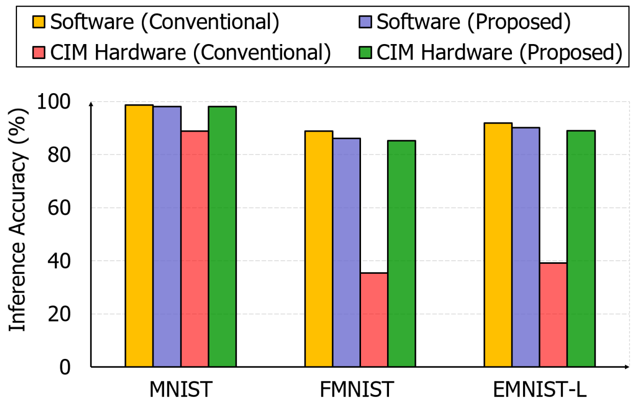


Figure 5.6: Neural network inference accuracy comparison across various datasets.

ancing the freedom of non-important weights and restriction on important weights. The optimal percentage varies from one dataset to another as indicated in Fig. 5.5. The hardware accuracy using weights at the optimal percentages is used for comparison between proposed mapping-aware biased training and conventional training (backpropagation) in Fig. 5.6. The proposed mapping-aware biased training has a slightly lower software accuracy compared to conventional training. This is because cost function minimization during training becomes difficult due to the favorability constraint on important weights. Our proposed biased training provides up to 2.4× hardware accuracy compared to conventional training. This can be attributed to the mapping of important weights to conductance states having a low variation impact. The accuracy improvement is higher for complex datasets (FMNIST, EMNIST-L) than simpler ones (MNIST), as they need more error-free computations for correct classification.

5.4.2. HARDWARE PERFORMANCE METRICS

The comparison of hardware metrics between the proposed mapping-aware biased training and the conventional training (backpropagation) is shown in Table 5.1. They both need identical hardware components and hence consume the same energy and area. We define a new metric “correct operations per unit energy" as the ratio of the number

Table 5.1: Hardware metrics per in-situ multiply-accumulate unit.

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

of correct operations to energy consumption (unit: Giga-operations per joule (GOP/J)). Here, the number of correct operations is the product of accuracy (as fraction) and the total number of operations. Table 5.1 shows that the proposed mapping-aware biased training achieves up to 2.4× correct operations per unit energy than conventional training without any hardware overhead.

5.5. CONCLUSIONS

This chapter presented a mapping-aware biased training to mitigate the impact of conductance variation on CIM-based neural networks. This was achieved by restricting the important weights during training, so that their post-training values directly get mapped to conductance states with low variation impact. The proposed biased training achieved up to 2.4× hardware accuracy and up to 2.4× correct operations per unit energy compared to the conventional training, with no hardware overhead. Such high accuracy and energy efficiency can facilitate the deployment of CIM-based neural networks for edge-AI.

6 Unbalanced Bit-slicing for CIM-based Neural Networks

This chapter is based on [27, 28].

6.1. INTRODUCTION

CIM architectures face the limitation of not being able to support the high bit-precision demands of neural network applications [32]. Therefore, a bit-slicing scheme [30, 31] is commonly employed in CIM architectures where multiple memristor devices represent a full-precision neural network weight. Bit-slicing CIM architectures represent a zero weight in the neural network using a memristor device with non-zero conductance equal to the minimum possible memristor device conductance denoted as G_{min} [150]. Multiplication of any non-zero digital input with a zero digital weight must produce a zero digital output. However, a non-zero output current is produced when a non-zero input in the form of a voltage is applied to a memristor with G_{min} conductance. This is known as non-zero G_{min} error, which violates the functional equivalence between the digital output and CIM output, leading to errors in VMM and degraded neural network accuracy.

State-of-the-art bit-slicing CIM architectures cannot provide good accuracy in presence of non-zero G_{min} error. For instance, ISAAC [30] and PUMA [31] use *balanced* bit-slicing (BBS) scheme which suffers from accuracy degradation due to non-zero G_{min} error. PANTHER [32] proposes *heterogeneous* bit-slicing (HBS) scheme, which is an extension of BBS and thereby struggles to provide good accuracy in the presence of non-zero G_{min} error. *Current subtraction technique* (CST) [150] can be utilized to mitigate the impact of non-zero G_{min} on BBS and HBS. However, it becomes less effective when conductance variation is considered along with non-zero G_{min} error. Hence, there is a strong need for an effective solution to mitigate the impact of non-zero G_{min} error on bit-slicing CIM architectures while taking conductance variation into account.

6

We propose an *unbalanced* bit-slicing (UBS) scheme for CIM architectures to mitigate the impact of non-zero G_{min} error. UBS provides higher sensing margin for more important bits; i.e. most significant bits (MSBs) to reduce the impact of non-zero G_{min}. Moreover, UBS is supported with 2's complement arithmetic whose inherent differential nature further helps in this mitigation task leading to improved neural network accuracy. Assigning higher sensing margin for more important bits i.e. most significant bits (MSBs) leads to achieve high accuracy at the expense of energy overheads. We extend UBS with a new variant which allocates just good enough sensing margin to the slices to improve the energy-efficiency by reducing the hardware requirements while retaining the accuracy benefits. We also provide an algorithm for optimal energy-efficient slice size selection which leverages an inherent accuracy versus energy tradeoff. Lastly, we also demonstrate the effectiveness of UBS across different datasets and neural networks. Our key contributions can be summarized as follows:

- We propose an unbalanced bit-slicing scheme which provisions high sensing margin for more important slices to achieve high accuracy in presence of non-zero G_{min} error and an algorithm to find slice sizes for optimal accuracy under given resource constraints.
- We develop a methodology to tailor the unbalanced bit-slicing scheme for energy-efficiency by constraining the sensing margin for slices in order to reduce the hardware resources while maintaining good accuracy.
- We present a holistic solution consisting of unbalanced bit-slicing logic and 2's

complement arithmetic which mitigates non-zero G_{min} error impact in presence of conductance variation.

- We demonstrate the effectiveness of the proposed unbalanced bit-slicing scheme by performing comprehensive analysis and comparison with state-of-the-art across different datasets and neural networks.

Simulation results show that UBS achieves up to 7.3× accuracy and up to 7.8× correct operations per unit energy consumption compared to state-of-the-art with reasonable overheads.

6.2. PROPOSED METHODOLOGY

6.2.1. OVERVIEW OF BIT-SLICING SCHEMES

A bit-slicing scheme consists of bit-slicing logic and associated crossbar arithmetic. The bit-slicing logic determines how a full-precision neural network weight is split into smaller slices, whereas the crossbar arithmetic governs the way in which the partial outputs from the crossbar columns are combined to obtain the final full-precision output. Along the same lines, an overview of the state-of-the-art as well as the proposed bit-slicing schemes is shown in Figure 6.1. For the state-of-the-art bit-slicing scheme, balanced bit-slicing logic splits the full-precision neural network weight into equal-sized (*balanced*) slices and unsigned binary arithmetic is used to combine the column-wise partial outputs. Whereas, in our proposed unbalanced bit-slicing (UBS) scheme, we split the full-precision neural network weight into a mix of equal-sized and unequal-sized (*unbalanced*) slices in such a way that the resulting allocated sensing margins minimize the impact of non-zero G_{min} error. The sensing margin allocation strategy in UBS can be adapted based on whether the goal is to achieve high accuracy or high energy-efficiency. If the target is to achieve high accuracy, UBS delivers high sensing margin to more important bits (MSBs). On the other hand, if energy-efficiency is the primary goal then UBS provides just enough

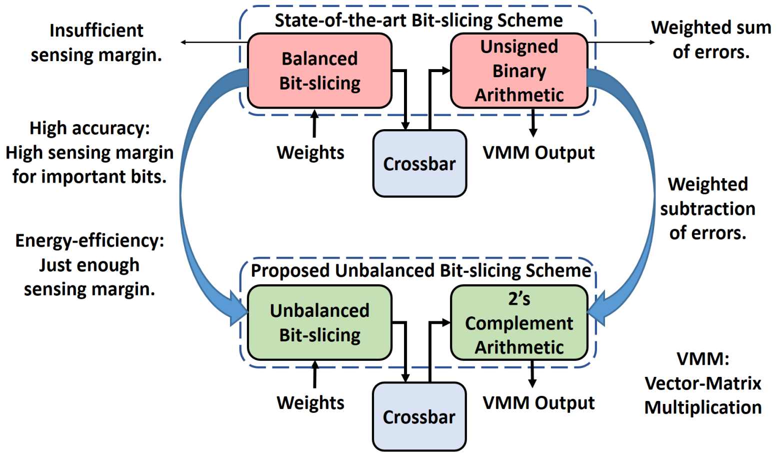

Insufficient sensing margin.
State-of-the-art Bit-slicing Scheme
Balanced Bit-slicing
Unsigned Binary Arithmetic
Weighted sum of errors.
Weights
Crossbar
VMM Output
High accuracy: High sensing margin for important bits.
Energy-efficiency: Just enough sensing margin.
Weighted subtraction of errors.
Proposed Unbalanced Bit-slicing Scheme
Unbalanced Bit-slicing
2's Complement Arithmetic
Weights
Crossbar
VMM Output
VMM: Vector-Matrix Multiplication

Figure 6.1: Overview of state-of-the-art and proposed bit-slicing schemes.

sensing margins to the slices to reduce the hardware resource requirements resulting in a high energy-efficiency while maintaining a sufficiently good accuracy. Moreover, UBS uses 2's complement arithmetic which results in negative scaling factor for the column containing MSB slices when combining the column-wise partial outputs using shift-and-add operations. The full-precision output denoted as D_{acc} is obtained by performing shift-and-add operations on n columns as per Eq. 6.1a, where D_is and S_is denote the column-wise partial outputs and scaling factors respectively.

$$D_{\text{acc}} = \sum_{i=1}^{n} S_i \cdot D_i \tag{6.1a}$$

$$D_{\text{acc}} = \sum_{i=1}^{n} S_i \cdot T_i + \sum_{i=1}^{n} S_i \cdot E_i \tag{6.1b}$$

$$E_{\text{acc}} = \sum_{i=1}^{n} S_i \cdot E_i = (-S_1 \cdot E_1 + \sum_{i=2}^{n} S_i \cdot E_i) \tag{6.1c}$$

Expressing $D_i = T_i + E_i$ leads to Eq. 6.1b for D_{acc}, where T_i is the ideal column output ($\text{G}_{\text{min}} = 0$ scenario) and E_i is the error due to non-zero G_{min}. The summation over E_i in Eq. 6.1b gives the accumulated non-zero G_{min} error denoted as E_{acc} which is present in the full-precision output D_{acc}. The contribution of the negatively scaled 2's complement MSB column (i = 1) towards E_{acc} can be separated from the other positively scaled columns (i = 2 to n) as shown in Eq. 6.1c, where E_1 is the error in MSB column and $-S_1$ is the MSB column scaling factor where $S_1 > 0$ denotes magnitude of the MSB column scaling factor. It is clear from Eq. 6.1c that the negative MSB column scaling factor reduces the overall accumulated error E_{acc} in the final output due to weighted subtraction of column-wise errors. However, such weighted subtraction will not be perfectly zero resulting in a non-zero accumulated error which can be large enough to cause substantial deviation from the correct VMM computation. The sensing margins provided by the unbalanced slice sizes in UBS ensure that this accumulated error remains small. Thus, sensing margin allocation and 2's complement arithmetic work together to minimize the impact of non-zero G_{min} error for improved neural network accuracy. In the next subsections, we first describe the details of UBS for achieving high accuracy followed by its more energy-efficient variant.

6.2.2. Unbalanced Bit-slicing for High Accuracy

Bit-slicing Logic

Goal of the bit-slicing logic is to minimize the accumulated error represented by E_{acc} in Eq. 6.1c that remains after combining the column-wise partial outputs using 2's complement arithmetic. It is clear that the accumulated error can be minimized by achieving a good matching between the magnitudes of negatively scaled error in MSB column and sum of the positively scaled errors in the rest of the columns as illustrated in Figure 6.2. If all the column-wise errors (E_i in Eq. 6.1c, i = 1 to n) are individually large, then the mismatch between the scaled errors will be large leading to a high accumulated error. On the other hand, a small accumulated error can be obtained if the columns with high scaling factors have low errors. As columns with high scaling factors correspond to MSB slices, our bit-slicing logic aims at reducing the errors in as many MSB slices as possible for

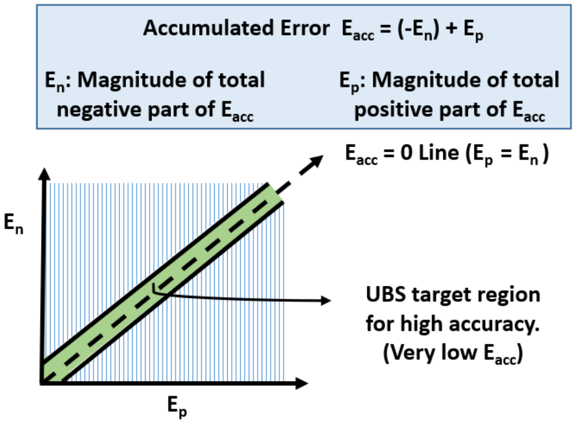

Accumulated Error E_{acc} = (-E_n) + E_p
E_n: Magnitude of total negative part of E_{acc}
E_p: Magnitude of total positive part of E_{acc}
E_{acc} = 0 Line (E_p = E_n)
E_n
E_p
UBS target region for high accuracy. (Very low E_{acc})

Figure 6.2: Error reduction approach for high accuracy.

achieving low accumulated error and high neural network accuracy. Non-zero G_{min} error at the output of a crossbar column with a certain slice size can be reduced by providing the slices with higher sensing margin. This can be achieved by using a memristor device with n-bit capacity as an m-bit memory-cell (slice) such that $m<n$. This results in wider separation between the conductance states leading to higher sensing margin and reduced error in the crossbar column output due to non-zero G_{min} as shown in Figure 6.3.

As discussed earlier, reducing the individual errors in as many MSB slices as possible will result in better accuracy. Different UBS configurations can be obtained based on how many MSBs are provided with high sensing margin to reduce the individual errors. For instance, using 2-bit memristors for 8-bit weights, [1,1,2,2,2] bits/slice (first 2 MSBs with

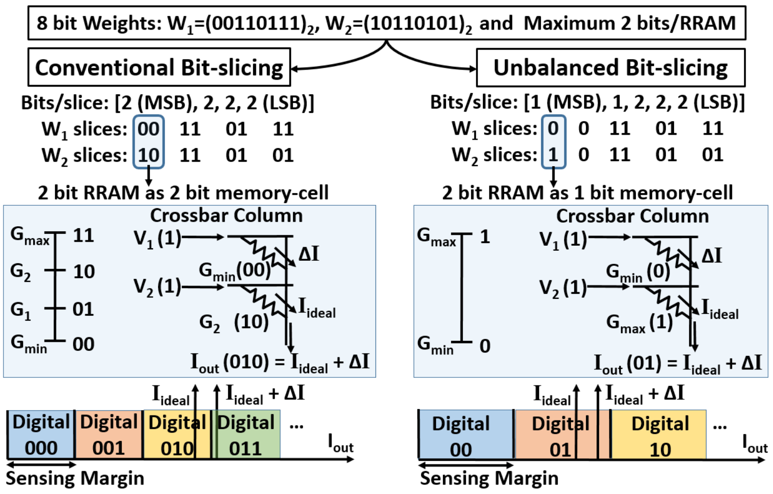


Figure 6.3: Illustration of the impact of sensing margin on bit-slicing schemes.

Algorithm 6.1: UBS slice configuration selection for high accuracy.

```
input  : CIM system architecture (A), Energy constraint (C),
         memristor bit-capacity (B), Bits per full-precision neural weight (N)
output : UBS slice configuration for high accuracy (S)
1  FSC ← fundamental_slice_config(B, N);
2  S ← FSC;
3  R ← compute_resource_req(A, FSC);
4  while $R < C$ do
5      config ← next_possible_UBS_config(S, B, N);
6      R ← compute_resource_req(A, config);
7      if $R < C$ then
8          S ← config;
9      end
10 end
11 return S
```

high sensing margin, total five slices) and [1,1,1,1,2,2] bits/slice (first 4 MSBs with high sensing margin, total six slices) are some of the possible configurations. UBS configuration with more number of slices results in better accuracy due to better matching between the positively and negatively scaled errors leading to smaller accumulated error. However, it needs more energy due to more analog to digital conversion operations. For minimum energy requirement, an UBS configuration should have:

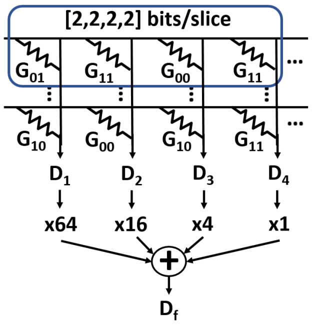
- Minimum number of slices per weight.
- High sensing margin for MSB slices.
- First MSB slice of 1-bit size for 2's complement arithmetic compatibility (details in Section 11).

Such configuration is called *fundamental slice configuration* (FSC) which is obtained for N-bit weights and m-bit memristors as follows: 1 bit for the first MSB slice and remaining N-1 bits divided into nearly equal chunks of m bits with small-sized chunks assigned to MSB slices. For example, with 8-bit weights and m=2 bits per memristor, we obtain FSC = [1,m-1,m,m,m]=[1,1,2,2,2] bits/slice which is used in Figure 6.3.

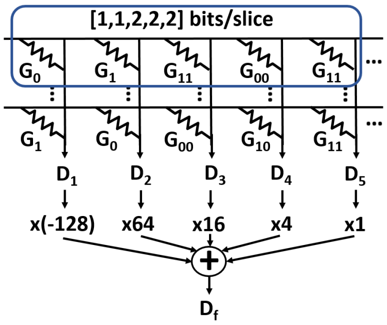
UBS provides high accuracy at the cost of additional energy. Algorithm 6.1 gives UBS slice sizes for optimal accuracy in presence of non-zero G_{min} error subjected to energy constraint. It starts with FSC having minimum energy requirement and then progressively assigns smaller slices to the next MSBs. Finally, the UBS slice configuration having the highest accuracy (highest number of slices) within the specified energy limit is selected.

CROSSBAR ARITHMETIC

Weights represented in 2's complement format cannot be mapped to a crossbar that uses BBS. This is due to the difficulty in isolating the negative contribution of the MSB from a multi-bit slice in 2's complement format [30]. Hence, BBS converts signed weights into equivalent positive weights using an offset and utilizes unsigned binary arithmetic



(a) Balanced bit-slicing (BBS).



(b) Unbalanced bit-slicing (UBS).

Figure 6.4: Accumulation of partial digital outputs in CIM crossbar.

to combine the column-wise partial outputs. In UBS, we force the MSB slice to always be of 1-bit size for compatibility with 2's complement weight encoding and utilize 2's complement arithmetic to combine the column-wise partial outputs.

Figure 6.4 shows the accumulation of partial digital outputs (D_i's) for 8-bit weights with 2-bit memristors using both conventional BBS and proposed UBS. We use [2,2,2,2] bits/slice for BBS and FSC with [1,1,2,2,2] bits/slice for UBS. We obtain Eq. 6.2a and Eq. 6.2b as the expressions for the accumulated output error for BBS and UBS respectively, by using Eq. 6.1c for dataflow graphs shown in Figure 6.4.

$$E_{\text{acc-BBS}} = 64 \cdot E_1 + 16 \cdot E_2 + 4 \cdot E_3 + E_4 \tag{6.2a}$$

$$E_{\text{acc-UBS}} = (-128) \cdot E_1 + 64 \cdot E_2 + 16 \cdot E_3 + 4 \cdot E_4 + E_5 \tag{6.2b}$$

These equations show that UBS can lead to lower accumulated error compared to BBS due to weighted subtraction of column-wise errors in 2's complement arithmetic. This holds true for all UBS configurations, as they use 1-bit slice for the first MSB to be compatible with 2's complement crossbar arithmetic. Moreover, impact of non-zero G_{min} error is further reduced as unbalanced bit-slicing logic (described earlier in Section 6.2.2) ensures that the error accumulated after weighted subtraction remains small.

6.2.3. UNBALANCED BIT-SLICING FOR ENERGY-EFFICIENCY

UBS leads to more slices per weight compared to BBS due to utilizing memristors for less than their bit-capacity for some slices to achieve higher sensing margin. The extra slices in UBS introduce additional analog-to-digital conversion operations resulting in higher energy consumption. Hence, it is necessary to decrease the number of slices per weight in UBS for reducing the energy overhead which can be achieved by using bigger slice sizes. However, the low sensing margin in bigger slice sizes leads to higher accumulated error and reduced accuracy. Thus, bigger slice sizes provide higher energy-efficiency at the expense of reduced accuracy and there exists a potential accuracy versus energy tradeoff.

Neural networks can inherently tolerate some deviation from ideal VMM computations. For instance, correct classification needs the output corresponding to the correct

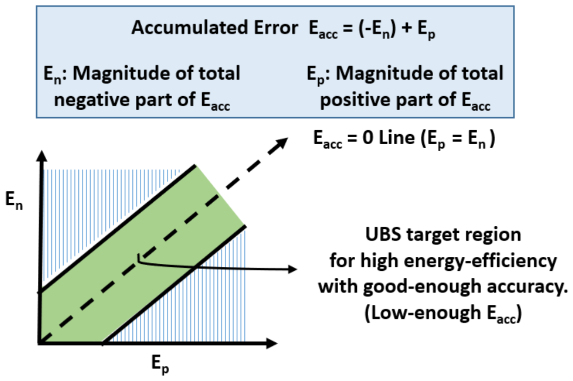

Accumulated Error $E_{acc} = (-E_n) + E_p$
E_n: Magnitude of total negative part of E_{acc}
E_p: Magnitude of total positive part of E_{acc}
$E_{acc} = 0$ Line ($E_p = E_n$)
E_n
UBS target region for high energy-efficiency with good-enough accuracy. (Low-enough E_{acc})
E_p

Figure 6.5: Error reduction approach for high energy-efficiency.

class to be maximum, but it does not matter if it is slightly higher or much higher than the other classes. So, instead of the targeting a very low accumulated error as done earlier in Section 6.2.2 to obtain high accuracy, sufficiently good accuracy can still be achieved with a higher accumulated error which is just small enough to get the correct classification output. Thus, we can leverage bigger slices for higher energy-efficiency while incurring reasonable accuracy loss by adjusting the sizes of the bigger slices in such a way that the accumulated error remains small enough for correct classification. This is illustrated in Figure 6.5. Hence, an energy-efficient UBS configuration that uses bigger slice sizes must satisfy the following conditions to achieve high energy-efficiency while maintaining sufficiently good accuracy:

- Less number of slices per weight than UBS FSC.
- Small enough accumulated error for correct classification.

To satisfy the constraint for number of slices, we need to utilize some slice sizes that are bigger than the memristor bit-capacity. For instance, with 2-bit memristors for 8-bit weights, UBS FSC ([1,1,2,2,2] bits/slice) leads to five slices. If we want the UBS configuration to fit in 4 or less slices, the only option is to keep the first MSB slice as 1-bit (for 2's complement compatibility) and increase the sizes of other slices so that the remaining seven bits fit in three or less slices. This leads to slice configurations such as [1,2,2,3] bits/slice where some slices exceed the 2-bit capacity of our memristor devices. Slices bigger than memristor bit-capacity can be implemented by overloading the existing memristor device. Overloading refers to the process of using n-bit memristor as an m-bit memory-cell (slice) such that $m > n$, where $n = 2$ and $m = 3$ for our scenario. A non-overloaded memristor (n-bit memristor used to hold n or fewer bits) has a certain available sensing margin and no overlap exists between the variation profiles of its conductance states. Overloading a memristor device leads to a larger number of conductance states within the same conductance range. This results in reduced sensing margin and overlap between the variation profiles of the conductance states. The more we overload a device (more difference between m and n), the lower is the sensing margin and the higher is the variation profile overlap. This can lead to increased errors due to non-zero G_{min} in presence of conductance variation for a column with overloaded memristors.

Overloading memristor devices to meet the first constraint for number of slices inherently increases the error in overloaded columns. However, we can still use overloading and satisfy the second constraint of small-enough accumulated error by overloading the columns which contribute less towards the accumulated error. The scaling factors for individual output errors (E_is) in MSB columns are much higher compared to the LSB columns as evident from Eq. 6.2b. Hence, low accumulated error can be achieved if MSB slices have smaller individual output errors compared to LSB slices. Hence, MSB slices in an energy-efficient UBS configuration should have higher sensing margin i.e. they should be smaller and less overloaded compared to LSB slices. Thus, a possible energy-efficient UBS configuration which can achieve low accumulated error in presence of memristor overloading is obtained as follows:

- First MSB slice must be of 1-bit size.
- Slices per weight must be less than UBS FSC.
- LSB slices must be bigger or equal to MSB slices.

For 8-bit weights and 2-bit memristors (UBS FSC as [1,1,2,2,2] bits/slice) various possible energy-efficient UBS configurations which satisfy the aforementioned conditions can be listed as: [1,2,2,3] bits/slice, [1,1,2,4] bits/slice, [1,1,3,3] bits/slice, [1,1,1,5] bits/slice, [1,3,4] bits/slice, [1,2,5] bits/slice, [1,1,6] bits/slice and [1,7] bits/slice. They provide different levels of accuracy while consuming different amounts of energy. Algorithm 6.2 describes the method of selecting appropriate energy-efficient UBS configuration according to the energy and accuracy constraints. It begins with FSC whose accuracy is taken as the

Algorithm 6.2: UBS slice configuration selection for energy-efficiency.

```
input :CIM system architecture (A), Permissible accuracy loss w.r.t. FSC (P), Energy
       constraint (C), memristor bit-capacity (B), Bits per full-precision neural weight (N)
output:UBS slice configuration for energy-efficiency (S)
1  FSC ← fundamental_slice_config(B, N);
2  FSC_Acc ← compute_accuracy(A, FSC);
3  $Acc_{min}$ ← FSC_Acc - P;
4  E ← list_energy_efficient_UBS_configs(FSC, B, N);
5  Eligible_configs_list ← ∅;
6  Eligible_configs_list.insert(FSC);
7  foreach config in E do
8      Accuracy ← compute_accuracy(A, config);
9      Energy ← compute_energy(A, config);
10     if Accuracy > $Acc_{min}$ and Energy < C then
11         Eligible_configs_list.insert(config);
12     end
13 end
14 S ← minimum_energy_config(Eligible_configs_list);
15 return S
```

baseline. The permissible accuracy loss with respect to this baseline is also specified as an input. Out of the various possible energy-efficient UBS configurations, the ones having accuracy and energy within the specified limits are shortlisted. Finally, slice configuration having the least energy requirement among the shortlisted ones is selected.

6.3. SIMULATION SETUP

We have developed a simulation framework in Python based on in-situ multiply-accumulate (IMA) unit in [30] (shown in Figure 6.6) which is compatible with UBS, BBS and HBS. The IMA design follows 32 nm CMOS technology [30]. The power and area details for various IMA components are also obtained from [30]. The power consumption of ADCs in the IMA is modelled based on the ADC design presented in [151] which is also referenced in [30]. In order to estimate the power/area for the aforementioned ADC at different bit resolutions while keeping rest of the ADC specifications same, we followed the methodology given in [30]. It involves scaling the power/area of all the ADC components except capacitive DAC (CDAC) linearly with ADC resolution and scaling the power/area of the CDAC exponentially with ADC resolution. The ADC power/area at new bit resolution is then obtained by summing all these scaled component-wise powers/areas. The memristor device-related simulation parameters are taken from HfO_x-based device presented in [147]. All of our experiments take into account both non-zero G_{min} error and conductance variation together. We consider 2-bit memristors (same as [30, 31]) and 8-bit weights. This leads to [1,1,2,2,2] bits/slice (FSC obtained using Algorithm 6.1) for UBS, [2,2,2,2] bits/slice for BBS [30, 31] and [1,1,2,2,1,1] bits/slice for HBS [32]. Other slice configurations are specified within the corresponding result figures.

We evaluate the performance of five datasets shown in Table 6.1 on a fully-connected neural network (FC-NN) as well as a convolutional neural network (CNN). All of our datasets consist of 28x28 images. The details of the used neural networks are as follows:

- FC-NN: It consist of an input layer of 784 neurons followed by two hidden layers with 100 and 50 neurons. It has 47 output neurons for EMNIST [149] dataset and 10 output neurons for all other datasets. Thus, FC-NN can be expressed as 784-100-50-(47 or 10). The activation function used is ReLU.

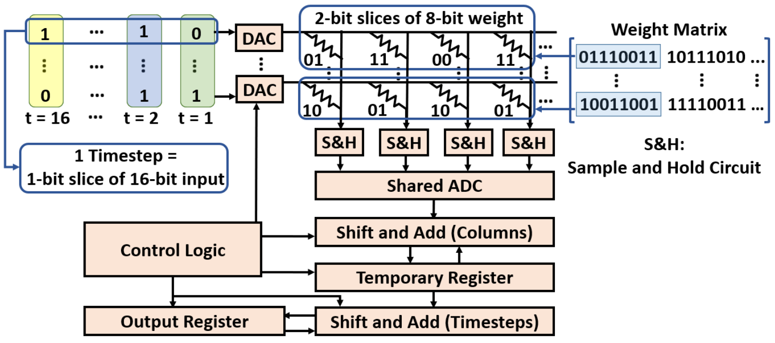


Figure 6.6: Memristor-based CIM architecture with bit-slicing.

Table 6.1: Accuracy on the datasets considered for simulation.

Dataset	Baseline Accuracy	
	Fully-connected Neural Network (FC-NN)	Convolutional Neural Network (CNN)
MNIST [85]	97.85%	98.75%
FMNIST [148]	88.57%	88.69%
KMNIST [152]	87.99%	92.47%
RMNIST [153]	87.94%	93.75%
EMNIST [149]	82.26%	85.40%

- CNN: Its structure is similar to LeNet-5 [85] with two convolution layers having 6 kernels and 16 kernels of size 5×5, each followed by a max-pooling layer. Flattened output from the second max-pooling layer connects to fully-connected layers with 120 neurons and 84 neurons. The output layer has 47 neurons for EMNIST [149] dataset and 10 neurons for all other datasets. So, we can represent the CNN as 6c5-maxpool-16c5-maxpool-flatten-120-84-(47 or 10), where mCn means m kernels of size n×n. ReLU is used as activation function.

Moreover, to demonstrate the applicability of UBS to complex datasets and large neural networks, we use CIFAR-10 [154] dataset on VGG-16 [155] neural network. As the original VGG-16 network is intended for 224x224 RGB images, we adapt it for 32x32 RGB images in CIFAR-10 by making a slight change in the final fully-connected layers. The resulting network can be represented as follows (nCm means n convolution filters of m x m size): Input - 64c3 - 64c3 - Maxpool - 128c3 - 128c3 - Maxpool - 256c3 - 256c3 - 256c3 - Maxpool - 512c3 - 512c3 - 512c3 - Maxpool - 512c3 - 512c3 - 512c3 - Maxpool - 512 - 10. This network achieves a software baseline accuracy of 89%.

After training these neural networks using PyTorch [88], their inference on CIM hardware is simulated using our Python-based framework. The Python-based simulation framework is validated by comparison with SPICE simulation. All the results are shown in terms of relative accuracy. It is calculated by expressing the accuracy obtained in Python-based hardware simulation for a given dataset as a percentage of its ideal (software) baseline accuracy in Table 6.1. Thus, relative accuracy acts as a measure of how much software accuracy is preserved in CIM hardware in presence of non-zero G_{min} error and conductance variation.

6.4. SIMULATION RESULTS

6.4.1. NEURAL NETWORK ACCURACY

UBS achieves the highest accuracy while HBS achieves the lowest accuracy among BBS, HBS and UBS for both FC-NN and CNN as shown in Figure 6.7 and Figure 6.8. The intuition behind this can be explained as follows. Non-zero G_{min} error at the output of i^{th} crossbar column can be expressed as:

$$E_i = \frac{N_i \cdot \delta}{S} \tag{6.3}$$

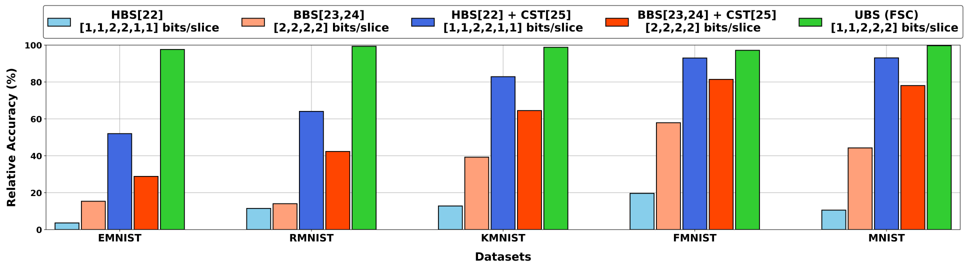


Figure 6.7: Accuracy comparison of bit-slicing schemes for fully-connected neural network.

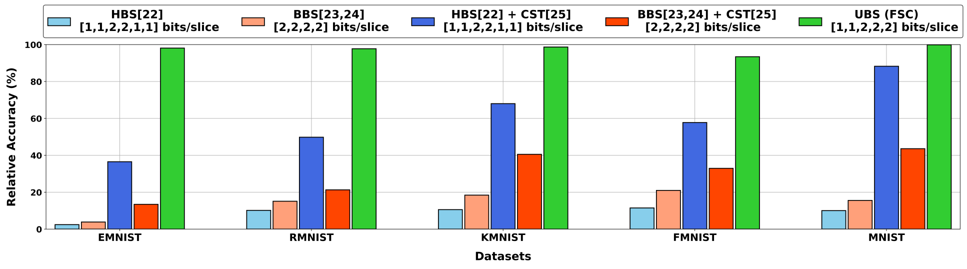


Figure 6.8: Accuracy comparison of bit-slicing schemes for convolutional neural network.

where N_i is the number of memristors (in the i^{th} column) having $G_{\min}$ conductance, δ is the error due to a single $G_{\min}$ conductance and S is the sensing margin for such an architecture design. The column-wise errors E_i's result in final accumulated error E_{acc} as per Eq. 6.2a and Eq. 6.2b for BBS and UBS respectively. UBS intends to reduce the error E_i compared to BBS by providing larger S (see Eq. 6.3) using smaller slice sizes for some important columns like MSBs. However, this also leads to higher N_i for some UBS columns compared to their BBS counterparts as smaller slice sizes produce more digital zero chunks which get mapped to $G_{\min}$, contributing towards increase in E_i. Nevertheless, weighted sum in Eq. 6.2a leads to high accumulated error for BBS despite having smaller N_i's. On the other hand, the impact of higher N_i's on the accumulated error in UBS is severely diminished thanks to weighted subtraction due to 2's complement arithmetic as shown in Eq. 6.2b. The sensing margin allocation in UBS further reduces the impact of accumulated error that remains after the weighted subtraction. Hence, UBS has a very small accumulated error and thereby provides much higher accuracy compared to BBS. Even though HBS provides high sensing margin for some slices, increase in N_i's coupled with weighted accumulation of errors (similar to Eq. 6.2a) due to its unsigned binary arithmetic leads to higher accumulated error and lower accuracy compared to both BBS and UBS.

CST [150] mitigation for BBS and HBS relies on current subtraction using a common dummy column for the entire crossbar. Such current subtraction is not perfect due to conductance variation and yields non-zero current residues. These current residues get accumulated across the columns leading to errors in the digital output after analog-to-digital conversion. These errors are further amplified during weighted sums to combine the column-wise partial digital outputs in HBS and BBS, leading to a high overall accumulated error. UBS does not rely on a common column and leverages weighted subtraction across various groups of columns. Hence, UBS outperforms HBS and BBS even when they are augmented with CST as shown in Figure 6.7 and Figure 6.8.

The final accumulated error at the output layer is high in CNN compared to FC-NN as our considered CNN has more layers than FC-NN. Thus, the impact of non-zero G_{min} error becomes more severe in CNN. Hence, the accuracy for BBS and HBS (with as well as without CST) reduces on CNN when compared to that on FC-NN. On the other hand, higher number of layers in CNN result in negligible impact on UBS accuracy as it always inherently leads to low error for the output in each layer. This is evident in Figure 6.7 and Figure 6.8. Thus, UBS outperforms BBS and HBS (both with and without CST) across various datasets as well as types of neural networks.

6.4.2. ACCURACY VERSUS ENERGY TRADEOFF EXPLORATION

We only consider in-situ multiply-accumulate (IMA) unit in [30] for energy comparison as bit-slicing schemes and configurations have no impact on its other components. Different bit-slicing schemes and configurations lead to changes in total number of crossbar columns as well as the number of bits per memristor device (slice size) for a given column. These changes affect various IMA components such as analog-to-digital converters (ADCs), crossbar arrays, sample-and-hold circuits etc. However, the contribution of ADCs towards the overall IMA energy is significantly higher compared to other components [30]. Hence, we focus on the impact of bit-slicing schemes and configurations on the energy of ADCs which can be directly correlated to the IMA energy. A change in the number of crossbar columns leads to a change in total number of ADCs in the IMA to maintain constant throughput. A change in the number of bits per memristor device for a column causes a change in the ADC resolution for correct analog-to-digital conversion. Thus, the overall IMA energy for a bit-slicing scheme or configuration is determined by the combined effect of the number of required ADCs and their resolutions.

Both BBS and HBS are augmented with CST for comparison in Figure 6.9 and Figure 6.10 as their accuracy is higher with CST (discussed in Section 6.4.1). UBS FSC ([1,1,2,2,2] bits/slice), BBS + CST ([2,2,2,2] bits/slice) and HBS + CST ([1,1,2,2,1,1] bits/slice) in Figure 6.9 and Figure 6.10 all have slices which are either 1-bit or 2-bit in size. Thus,

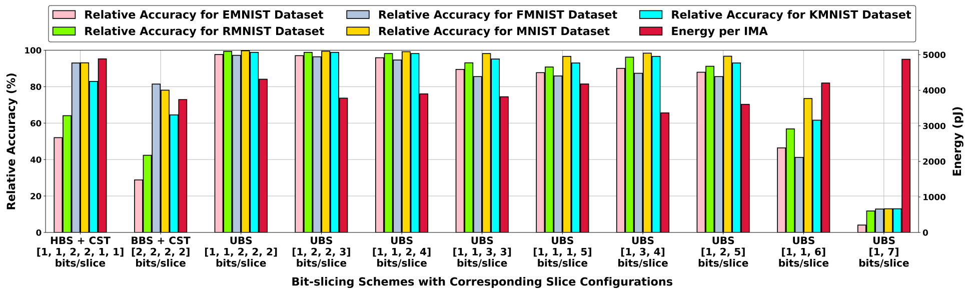


Figure 6.9: Accuracy versus energy tradeoff for bit-slicing schemes on fully-connected neural network.

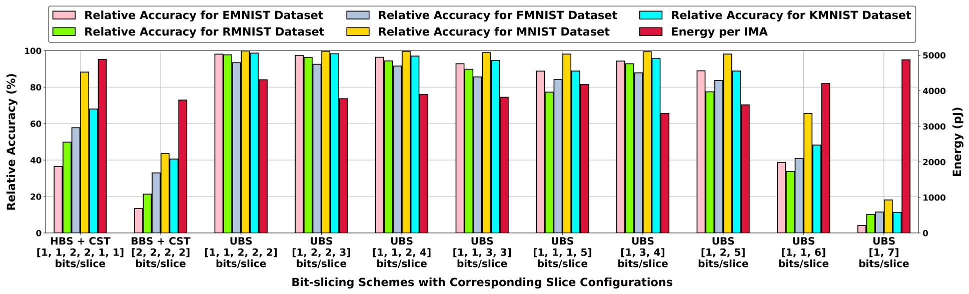


Figure 6.10: Accuracy versus energy tradeoff for various bit-slicing schemes on convolutional neural network.

ADC resolutions used in these schemes are quite similar. Hence, their IMA energy performance is mainly governed by total number of ADCs. The total number of ADCs required for UBS FSC is less than that in HBS + CST as it has less number of slices per weight. Hence, UBS FSC results in less energy than HBS + CST. On the other hand, UBS FSC has more slices per weight than BBS + CST which results in more number of ADCs and higher energy when compared to BBS + CST. This is shown in Figure 6.9 and Figure 6.10.

The energy overhead for UBS FSC can be lowered at the expense of reduced accuracy by decreasing the total number of slices. As discussed in Section 6.2.3, the possible energy-efficient UBS configurations can be listed as: [1,2,2,3] bits/slice, [1,1,2,4] bits/slice, [1,1,3,3] bits/slice, [1,1,1,5] bits/slice, [1,3,4] bits/slice, [1,2,5] bits/slice, [1,1,6] bits/slice and [1,7] bits/slice. Two opposite effects regarding the energy consumption come into picture when we move from UBS FSC to energy-efficient UBS configurations:

- Energy-efficient UBS configurations have less number of slices than UBS FSC. Hence, they need less crossbar columns and less ADCs compared to UBS FSC. This contributes towards reduction in energy requirements.

- Energy-efficient UBS configurations use bigger slice sizes compared to UBS FSC. Hence, they need higher resolution ADCs which demand more energy. This contributes towards increase in energy requirements.

The overall energy consumption depends on which of these two effects dominates for a given slice configuration. It can be seen in Figure 6.9 and Figure 6.10 that [1,3,4] bits/slice consumes the least energy among all schemes and configurations (even less than BBS). This indicates that the impact of number of slices (number of ADCs) is dominant over the impact of slice sizes (ADC resolution) for [1,3,4] bits/slice.

The energy-efficient UBS configuration in which bit-sizes of the adjacent slices are similar, leads to better balancing between the scaled column-wise errors. This results in smaller accumulated error and high accuracy. Hence, [1,2,2,3] bits/slice gives the best accuracy among all the listed energy-efficient UBS configurations. This is because the maximum bit-size difference between its adjacent slice-sizes is 1, which is the smallest among all the listed energy-efficient UBS configurations. Similarly, among configurations with 3 slices, [1,3,4] bits/slice gives the highest accuracy as the maximum difference between its adjacent slices is 2, while it is 3 and 5 for [1,2,5] bits/slice and [1,1,6] bits/slice respectively. Note that even though energy-efficient UBS configurations achieve slightly less accuracy compared to UBS FSC having [1,1,2,2,2] bits/slice, they still achieve much higher accuracy compared to BBS and HBS (both with CST) as evident in Figure 6.9 and Figure 6.10. The configurations [1,1,6] bits/slice and [1,7] bits/slice result in significantly poor accuracy compared to all other energy-efficient UBS configurations as depicted in Figure 6.9 and Figure 6.10. Hence, it is not recommended to overload a 2-bit memristor device for holding more than 5 bits. The analysis in Figure 6.9 and Figure 6.10 can be used to find the UBS configuration having the lowest energy consumption for a given accuracy constraint as well as the UBS configuration with the highest accuracy for a given energy budget.

It is also interesting to note that the bit-sizes for weights in different layers of the neural network can be selected independent of UBS. The bit-sizes for weights in each layer are given as a design constraint to UBS (like number of bits per memristor) and UBS tries to deliver the best possible performance for these given bit-sizes. For designs which use the same bit-size for weights in every layer, all layers should be provided with the same slice configuration based on the accuracy versus energy-efficiency tradeoff. For instance, with 16-bit weights and 2-bit memristors, [1,1,2,2,2,2,2,2,2] bits/slice should be used for all layers if accuracy is the primary concern and [1,2,2,2,2,3,4] bits/slice should be used for all layers if energy-efficiency is the major concern. If the design uses different bit-sizes for weights in different layers, the same strategy can be extended to layers that use the same bit-size. For instance, consider a network where half the layers use 8-bit weights, and half the layers use 16-bit weights. If accuracy is the primary target, then one should use [1,1,2,2,2] bits/slice for 8-bit layers and [1,1,2,2,2,2,2,2,2] bits/slice for 16-bit layers. On the other hand, for energy-efficiency, one should use [1,3,4] bits/slice for 8-bit layers and [1,2,2,2,2,3,4] bits/slice for 16-bit layers. Moreover, UBS is designed to make the vector-matrix multiplication error free. Provided that the core computation involved in a layer is vector-matrix multiplication or matrix-matrix multiplication, any such layer can reap the benefits of UBS by following the slice configuration allocation logic discussed earlier. Hence, UBS is applicable to any type of deep learning layer as almost all types of layers in deep-learning involve matrix-matrix multiplication as the core computation.

6.4.3. SCALABILITY ASSESSMENT

The accuracy comparison of various bit-slicing schemes on VGG-16 network using CIFAR-10 dataset is shown in Figure 6.11. We have considered HBS and BBS empowered with CST for this comparison because they achieve their highest possible accuracy with CST as evident in Figure 6.7 and Figure 6.8. We use 16-bit weights as CIFAR-10 is a complex dataset and VGG-16 is a large network [30]. Considering 2 bits per memristor as in Section 6.3, we get [1,1,2,2,2,2,2,2,2] bits/slice for UBS, [2,2,2,2,2,2,2,2] bits/slice for BBS+CST and [1,1,2,2,2,2,2,2,1,1] bits/slice for HBS+CST. To improve the energy efficiency of UBS with 9 slices ([1,1,2,2,2,2,2,2,2] bits/slice), we follow the same logic as in Section 6.4.2 to reduce the number of slices. This leads to [1,2,2,2,2,2,2,3] bits/slice as the best (most accurate) configuration having 8 slices and [1,2,2,2,2,3,4] bits/slice as the best (most accurate) configuration having 7 slices. These slice configurations are also included for the comparison in Figure 6.11. It is evident from Figure 6.11 that UBS outperforms HBS+CST and BBS+CST on VGG-16 with CIFAR-10 dataset in terms of accuracy. This

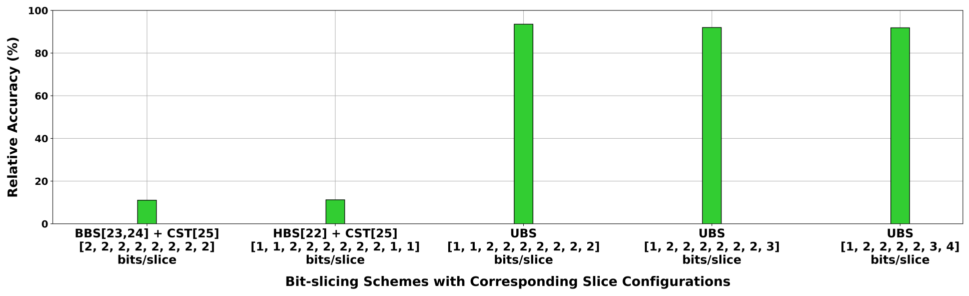


Figure 6.11: Accuracy comparison of bit-slicing schemes on VGG-16 neural network for CIFAR-10 dataset.

can be attributed to the mitigation provided by the sensing margin allocation which is also supported by 2's complement crossbar arithmetic. Moreover, energy-efficient UBS configurations ([1,2,2,2,2,2,2,3] bits/slice and [1,2,2,2,2,3,4] bits/slice) incur a very minor accuracy loss compared to the UBS with 9 slices ([1,1,2,2,2,2,2,2,2] bits/slice) as they sacrifice sensing margin for LSBs to reduce the number of slices.

6.4.4. HARDWARE PERFORMANCE METRICS

The performance metrics per IMA for various bit-slicing schemes are summarized in Table 6.2. We consider both BBS and HBS augmented with CST so that their most accurate versions are considered for comparison. Out of the various energy-efficient UBS configurations discussed in Section 6.4.2, [1,2,2,3] bits/slice and [1,3,4] bits/slice are selected as they provide the highest accuracy and the lowest energy among UBS configurations having 4 slices per weight and 3 slices per weight respectively. UBS FSC ([1,1,2,2,2] bits/slice) provides 7.3× accuracy at the expense of 15.3% energy overhead compared to BBS, while it achieves 2.7× accuracy and consumes 11.7% less energy when compared to HBS. The UBS energy overhead with respect to BBS can be reduced from 15.3% to mere 1.1% by using UBS with [1,2,2,3] bits/slice. We can even achieve 10% energy savings compared to BBS by leveraging UBS with [1,3,4] bits/slice.

The net energy-efficiency represents the number of operations performed per unit

Table 6.2: Summary of the performance metrics per IMA, unit indicated in brackets with each metric. Accuracy values for EMNIST dataset on CNN are reported. Correct operations per unit energy = (Total operations×Accuracy/100)/Energy, unit: giga operations per joule (GOP/J).

Metric	**UBS [This work]**			**BBS [30, 31] + CST [150]**	**HBS [32] + CST [150]**
	[1,1,2,2,2] bits/slice	**[1,2,2,3] bits/slice**	**[1,3,4] bits/slice**	**[2,2,2,2] bits/slice**	**[1,1,2,2,1,1] bits/slice**
Energy (pJ)	4311	3778	3365	3738	4883
Net energy-efficiency (GOPS/W)	237.5	271.0	304.3	273.9	209.7
Accuracy (%)	83.77	83.23	80.54	11.46	31.19
Correct operations/energy (GOP/J)	198.9	225.5	245.1	31.4	65.4
Area (μm^2)	23324	23121	26985	21765	24883

energy consumption and is expressed as giga operations per second per watt (GOPS/W) in Table 6.2. The energy-efficiency comparison can be correlated to the energy comparison discussed earlier by taking into account the fact that these two have inverse relationship. For instance, the more energy-efficient scheme is the one that consumes less energy or has higher GOPS/W. However, metrics like energy and energy-efficiency do not take into account the fact that even though BBS appears to be energy-efficient, it is spending that energy in performing incorrect computations as reflected in its poor accuracy. Hence, we define a metric called “correct operations per unit energy" which takes into account both energy and computational correctness. The number of correct operations can be computed as a product of accuracy (in fraction format, not as a percentage) and total operations. Correct operations per unit energy is then simply obtained by dividing the number of correct operations by total energy consumption. UBS FSC ([1,1,2,2,2] bits/slice) provides 6.3× and 3× correct operations per unit energy compared to BBS and HBS respectively. [1,2,2,3] bits/slice provides 7.2× and 3.4× correct operations per unit energy compared to BBS and HBS respectively, at the cost of just 0.54% accuracy loss compared to UBS FSC. [1,3,4] bits/slice results in 7.8× and 3.7× correct operations per unit energy compared to BBS and HBS respectively, at the expense of 3.23% accuracy loss compared to UBS FSC. This shows that the number of correct computations performed per unit energy consumption is much higher for UBS and it increases further with the use of energy-efficient UBS configurations for a very small decrease in accuracy.

UBS FSC ([1,1,2,2,2] bits/slice) requires 7.2% more area with respect to BBS. UBS with [1,2,2,3] bits/slice and [1,3,4] bits/slice incur area overheads of 6.2% and 24% respectively when compared to BBS. Thus, UBS with [1,2,2,3] bits/slice is the best choice if the target is to achieve high energy-efficiency with accuracy closest to UBS FSC and minimum area overhead with respect to BBS. On the other hand, [1,3,4] bits/slice is the best choice if we are willing to trade further accuracy and area for even higher energy-efficiency.

6.5. CONCLUSIONS

This chapter presented an unbalanced bit-slicing scheme to mitigate the impact of non-zero minimum conductance of memristors on CIM architectures. It achieved this by appropriate utilization of sensing margin and leveraging the weighted subtraction effect of 2’s complement crossbar arithmetic. Two different sensing margin allocation strategies were proposed based on whether the final goal is to achieve high accuracy or energy-efficiency. The proposed scheme provided up to 7.3× accuracy and up to 7.8× correct operations per unit energy consumption compared to state-of-the-art. Such high accuracy and energy-efficiency can benefit a wide spectrum of AI applications.

7
Adaptive Referencing Architecture for CIM-based Neural Networks

This chapter is based on [29].

7.1. INTRODUCTION

Read-disturb is a phenomenon where a large number of read operations lead to a significant resistance change in the memristor [22]. As neural network inference involves numerous read operations on memristors, read-disturb causes an undesired change in the weights stored as memristor resistances and results in degraded inference accuracy [23]. To reduce the impact of read-disturb, some works have recommended using low read voltages [21, 22, 156]. However, this leads to a reduced sensing margin and increased influence of process variation, resulting in erroneous output. Alternatively, CIM-aware training can be leveraged to address read-disturb. Such approaches fall into two categories: i) ex-situ where software models of non-idealities are incorporated into the training [134, 157], and ii) in-situ where training involves forward path execution directly on the CIM chip [55, 158]. The ex-situ approach only deals with design-time non-idealities, leaving run-time non-idealities like read-disturb unaddressed. The in-situ approach requires frequent on-chip training iterations, leading to high energy consumption and endurance issues. A few works provide only detection of the occurrence of read-disturb [159, 160], while [161, 162] recommend periodic reprogramming of memristors for read-disturb mitigation which leads to excessive write energy. Some techniques periodically reverse the direction of the read current to compensate for the resistance change due to read-disturb [23, 163]. However, this strategy falls short due to the asymmetric behavior of read-disturb in opposite read directions, leading to incorrect compensation. Furthermore, it adds extra complexity to the hardware design, which can negatively affect inference accuracy in the presence of process variation. Hence, there is a strong need for an effective read-disturb mitigation technique to improve the accuracy of CIM-based neural networks.

We present an adaptive referencing architecture for dynamic detection and mitigation of read-disturb in CIM-based neural networks. It begins with an analysis to extract insights about the read-disturb phenomenon. We then develop a dynamic detection mechanism capable of identifying instances of read-disturb occurrence at run-time. Moreover, we propose an adaptive method based on the aforementioned analysis, that adjusts sensing conditions in CIM hardware upon detecting read-disturb to prolong the error-free operation. Our key contributions are as follows:

- An analysis to derive insights about the read-disturb phenomenon in memristor-based neural networks.
- A mechanism for detecting the occurrence of read-disturb at run-time.
- An adaptive design to adjust the CIM sensing conditions to extend the correct functionality.

Our proposed architecture provides up to 2× accuracy and up to 2× correct operations per unit energy than conventional CIM architectures.

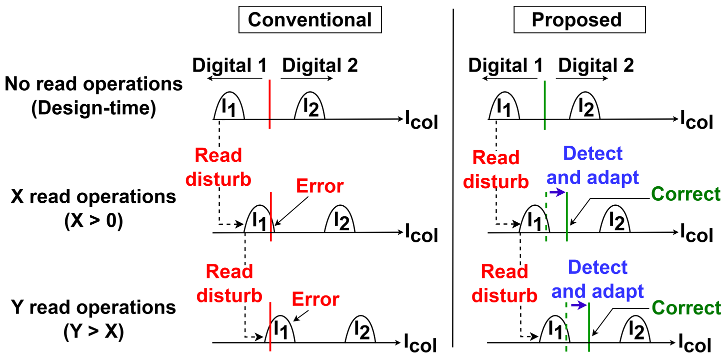


Figure 7.1: Overview of conventional and proposed referencing schemes in the presence of read disturb.

7.2. PROPOSED METHODOLOGY

7.2.1. OVERVIEW OF REFERENCING SCHEMES

Based on the experiments in [23], read disturb only impacts R_H memristors and decreases their resistance. Thus, the column currents increase over time and their distributions shift towards the right as shown in Fig. 7.1. Conventional design practice sets sensing reference at the midpoint between adjacent column current distributions, to optimize the sensing margin [30, 31, 164]. However, it leads to erroneous operation at run-time due to right shift induced by read-disturb, as shown in Fig. 7.1. This challenge is also not effectively addressed by prior works as discussed in Section 7.1. Our proposed methodology overcomes this problem by dynamically detecting the occurrences of read-disturb and then adapting the sensing references to restore correct operation as depicted in Fig. 7.1. This requires new hardware components such as read-disturb detection unit, ADC adaptation control, and adaptive ADC as shown in Fig. 7.2. The design details of these components are discussed next.

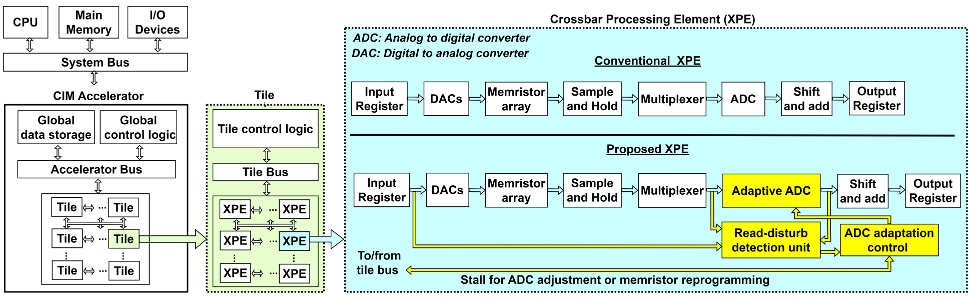


Figure 7.2: CIM system architecture with conventional and proposed crossbar processing element (XPE), with new/modified components indicated in yellow.

Algorithm 7.1: Identifying the most vulnerable columns.

```
input  : Weights (W), inputs (I), fixed point weight format ($F_W$), fixed point input format
         ($F_I$), memristor bit-size (R), DAC resolution (D), crossbar size (X)
output : A matrix containing the location of the most vulnerable column in each crossbar
         (MVC_matrix)
1 $W_{fxp} \leftarrow$ fixed_point_conversion(W, $F_W$);
2 $W_{sliced} \leftarrow$ bit_slicing($W_{fxp}$, R);
3 $W_{xbar} \leftarrow$ split_into_xbar_chunks($W_{sliced}$, X);
4 $I_{fxp} \leftarrow$ fixed_point_conversion(I, $F_I$);
5 $I_{sliced} \leftarrow$ bit_slicing($I_{fxp}$, D);
6 $I_{xbar} \leftarrow$ split_into_xbar_chunks($I_{sliced}$, X);
7 RVS_matrix $\leftarrow$ track_nonzero_inputs_to_$R_H$($I_{xbar}$, $W_{xbar}$);
8 MVC_matrix $\leftarrow$ max(RVS_matrix, X);
9 return MVC_matrix
```

7.2.2. DYNAMIC READ-DISTURB DETECTION

Dynamic read-disturb detection starts with software profiling in Algorithm 7.1. We use test dataset for profiling as it closely emulates the post-deployment stimuli that the network may encounter. The neural network inputs and weights are first quantized to fixed-point numbers and bit-sliced to match the DAC resolution (for inputs) or memristor bit capacity (for weights). These matrices with bit-sliced values are then divided into crossbar-sized submatrices to be used for profiling. As read-disturb builds up only when an R_H memristor (digital 0 weight) receives non-zero voltage (digital 1 input), our profiling tracks instances where each digital 0 weight in a submatrix encounters a digital 1 input. This tally corresponds to the number of read operations (N_{read}) affecting each R_H memristor throughout the profiling dataset. Given that the output of a CIM crossbar is the column current, the sum of N_{read} across all R_H memristors in a column gives its read-disturb vulnerability score (RVS). For each crossbar, the column with the highest RVS is designated as the most vulnerable column (MVC).

Read-disturb detection unit (RDU), shown in Fig. 7.3, within each crossbar then monitors its MVC to dynamically detect the occurrence of read-disturb. The RDU has two registers: one stores the index of MVC, while the other stores a digital version of MVC weights. When XPE's multiplexer select line matches the MVC index, RDU performs

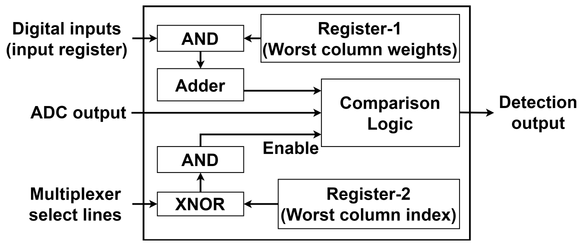

Digital inputs (input register)
AND
Register-1 (Worst column weights)
Adder
ADC output
Comparison Logic
Detection output
Enable
AND
Multiplexer select lines
XNOR
Register-2 (Worst column index)

Figure 7.3: Read-disturb detection unit.

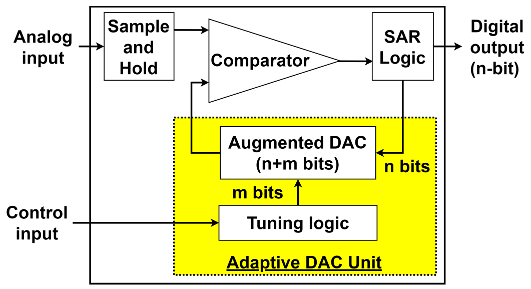
a multiply-accumulate operation between the digital MVC weights register and digital inputs retrieved from XPE's input register. A mismatch between this multiply-accumulate output and XPE's ADC output serves as an indicator of read-disturb, prompting activation of the ADC adaptation control unit. The column under RDU monitoring can be changed by altering the contents of its internal register offering hardware adaptability for scenarios such as varying datasets, retrained networks, etc.

7.2.3. ADAPTIVE REFERENCING DESIGN

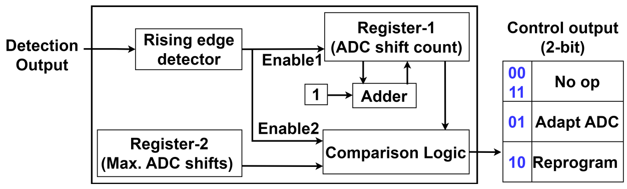
As the column current distributions shift towards right over time due to read-disturb, shifting the sensing references towards right upon detecting read-disturb can restore the correct operation. This can be achieved using an adaptive ADC and a control logic.

We design adaptive ADC using successive approximation-register (SAR) ADC as the base. This is because SAR ADC is widely used in CIM due to its low power consumption [165]. It uses a binary search approach to map the analog input to the appropriate digital output. Reference shifting in n-bit SAR ADC is achieved by augmenting its internal digital-to-analog converter (DAC) with m additional least significant bits (LSBs) and a tuning logic, as shown in Fig. 7.4a. The control input activates the tuning logic to increment the m-bit LSB value by 1. As a result, the reference values change by $V_{FS}/2^{(m+n)}$, where V_{FS} denotes the full-scale voltage of the ADC. The references can be shifted at the most 2^m times, which is the maximum possible LSB value.

The adaptive ADC control logic is depicted in Fig. 7.4b. It initiates reference adaptations in the adaptive ADC and coordinates essential control sequences with the tile control logic. It contains a register that holds the value of maximum possible shifts (2^m) and a counter that keeps track number of shifts that have already occurred (C). The



(a) Adaptive SAR ADC.



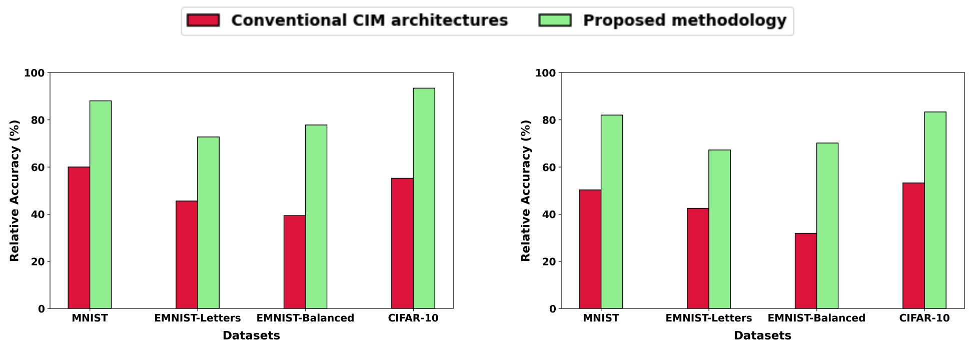
(b) ADC adaptation control unit.

Figure 7.4: Adaptive ADC and control logic design.

counter is reset to zero on system startup and only gets activated by the output of the read-disturb detection unit. Upon activation, if $C < 2^m$, it increments by 1. It also triggers the tuning logic within the adaptive ADC and dispatches a control sequence stall request to the tile control unit. This prompts the tile control unit to wait for the completion of ADC reference shifting. Conversely, if $C = 2^m$, the control logic initiates a memristor reprogramming request to the tile control unit.

7.3. Simulation Setup

We use the following four datasets for evaluation: MNIST [85], EMNIST-Letters [149], EMNIST-balanced [149] and CIFAR-10 [154]. We modify the VGG [155] network for CIFAR-10 as: 32c3 → 32c3 → maxpool → 64c3 → 64c3 → maxpool → 128c3 → 128c3 → maxpool → flatten → 128 → 10. Here, nCm denotes a block of n filters of m kernel size with batch normalization and relu, while a single number indicates neurons in a fully connected layer with batch norm and relu (except the last layer). Lenet-5 [85] with batch normalization is used for all other datasets. After training these networks in PyTorch, we perform behavioral simulation of their inference on CIM hardware using our Python-based framework. This framework leverages the crossbar processing element (XPE) in Fig. 7.2 and adaptive ADC obtained by modifying the SAR ADC in [30, 31] as per Section 7.2.3 with four extra LSBs. Power and area for the adaptive ADC are obtained using [166]. We performed RTL synthesis in TSMC 40nm technology to derive power and area for read-disturb detection unit and ADC adaptation control unit. The power and area of other XPE components are obtained from [31]. We consider full-precision weights distributed across a group of 1-bit memristors, programmed using write-verify method in [167]. Read-disturb models are extracted from [23] which presents experimental investigations on real memristors. Our simulations consider both finite on-off ratio and resistance variation in addition to read-disturb.



(b) Read voltage 0.5V. (c) Read voltage 0.6V.

Figure 7.5: Accuracy comparison between conventional CIM architectures [30, 31] and proposed methodology.

7.4. Simulation Results

7.4.1. Neural Network Accuracy

The accuracy comparison between proposed adaptive referencing and conventional CIM architectures [30, 31] is shown in Fig. 7.5. These results are presented in terms of relative accuracy, obtained by normalizing the accuracy of behavioral CIM hardware simulation with corresponding software baseline accuracy. This normalization effectively quantifies how faithfully computational correctness in software is maintained in CIM hardware. The proposed methodology achieves up to 2× accuracy compared to conventional CIM architectures (EMNIST-Balanced dataset with 0.5V read voltage), providing effective read-disturb mitigation. It also accommodates diverse dataset complexities (from MNIST to CIFAR-10) and adapts seamlessly to various network sizes (from Lenet-5 to VGG-like architecture). Furthermore, it delivers these benefits across two different read voltages, highlighting its robustness.

7.4.2. Hardware Performance Metrics

The hardware metrics per XPE for the proposed methodology and conventional CIM architectures [30, 31] are shown in Table 7.1. Metrics like energy and net energy-efficiency expressed in giga operations per second per watt (GOPS/W) do not inherently account for the energy devoted to correct computations. Hence, we introduce a new metric called “correct operations per unit energy". It is defined as the ratio of correct operations to total energy consumption (unit: Giga operations per joule, GOP/J), where correct operations are given by the product of accuracy (as a fraction) and total operations. Our proposed methodology achieves up to 2× correct operations per unit energy at the expense of 2.2% energy overhead and 23.6% area overhead. This additional cost can be attributed to the increased resolution of the DAC within the SAR ADC, which is necessary to accommodate reference shifting. Thus, it is clear that its advantages outweigh the overheads.

Table 7.1: Comparison of crossbar processing element (XPE) metrics.

Metric	**Conventional XPE [30, 31]**	**Proposed XPE [This work]**
Energy consumption (pJ)	407.06	416.00
Net energy-efficiency (GOPS/W)	157.22	153.85
EMNIST-Balanced accuracy at 0.5V (%)	33.45	66.12
Correct operations per unit energy (GOP/J)	52.59	101.72
Area (μm^2)	3735.43	4617.82

7.5. Conclusions

This chapter presented an adaptive referencing architecture for dynamic detection and mitigation of read-disturb, to improve the accuracy of CIM-based neural networks. This was achieved through a combination of run-time monitoring and adaptation of sensing references. Our proposed architecture achieved up to 2× accuracy compared to that of conventional CIM architectures. Thus, we have shown that a shrewd hardware design can facilitate correct operation despite memristor non-idealities.

PART-IV

CIM Edge-AI Prototyping for Healthcare

8

CIM-based ECG Classification Prototype

This chapter is based on[168, 169].

8.1. INTRODUCTION

This chapter describes our CIM hardware prototype for ECG classification. It combines our hierarchical ECG classification model (Chapter 3) with non-ideality mitigation strategies (Chapters 5, 6, and 7), as depicted in Figure 8.1. This integration offers energy efficiency through the CIM-tailored hierarchical classification model and ensures accurate hardware computations due to non-ideality mitigation strategies. From the three classifiers in hierarchical model, we select classifier 1 for prototyping. This decision is based on the practical implication of each classifier. Classifier 1 plays a crucial role in alerting end-users about the urgency of seeking medical attention. On the other hand, the other two classifiers provide additional insights useful for medical practitioners treating the individual. Therefore, classifier 1 becomes the prime choice for prototyping due to its direct impact of on human well-being. Among the three non-ideality mitigation strategies, we select unbalanced bit-slicing for our prototype. This choice arises from our use of polysilicon-based resistive storage in place of memristors, owing to difficulties in obtaining a commercial memristor fabrication process. Unlike memristors, polysilicon resistive elements do not suffer from conductance variation and read disturb problems. However, they are still affected by non-zero G_{min} error non-ideality. Hence, we leverage unbalanced bit-slicing in the prototype to address this particular challenge.

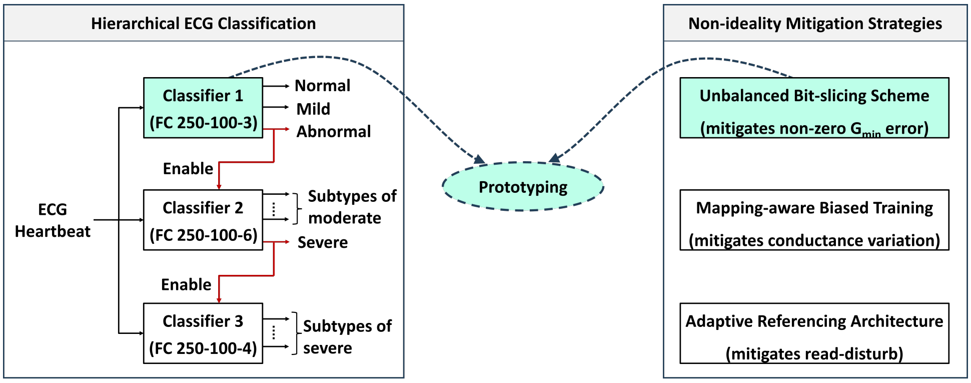

Hierarchical ECG Classification
ECG Heartbeat
Classifier 1 (FC 250-100-3)
Normal
Mild
Abnormal
Enable
Classifier 2 (FC 250-100-6)
Subtypes of moderate
Severe
Enable
Classifier 3 (FC 250-100-4)
Subtypes of severe
Prototyping
Non-ideality Mitigation Strategies
Unbalanced Bit-slicing Scheme (mitigates non-zero G_{min} error)
Mapping-aware Biased Training (mitigates conductance variation)
Adaptive Referencing Architecture (mitigates read-disturb)

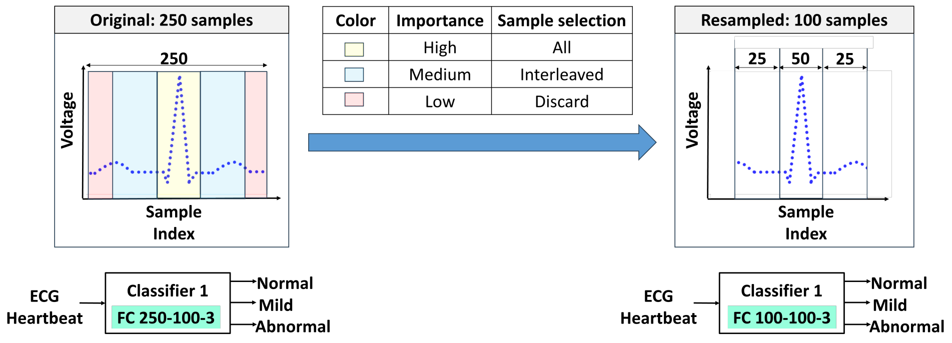
Figure 8.1: Our prototyping approach to combine hierarchical ECG classification model with non-ideality mitigation strategies. FC indicates fully connected neural network.

We develop the prototype incorporating classifier 1 and unbalanced bit-slicing through a three-phase process. The first phase focuses on optimizing classifier 1 further for efficient hardware implementation. This is achieved through resampling and quantization. The resampling process reduces the model complexity by decreasing the number of input neurons and size of the hidden layer matrix. This leads to a lower computational resource usage. We then train the streamlined model to adapt to the information loss caused by resampling. Following the retraining, we quantize the resampled model. It involves determining the smallest bit-sizes for various model components, while ensuring minimal accuracy degradation. This reduces hardware resource usage while maintaining high accuracy. In the second phase, we design CIM hardware architecture for the resampled and quantized model. We translate the model computations into CIM hardware operations and associated control logic. This translation incorporates unique hardware features such as unbalanced bit-slicing and a new ADC design (out of the scope of this thesis). Moreover, it includes additional logic blocks to handle off-chip data and control signal exchanges. In the final phase, we perform the physical layout implementation for this hardware architecture. The analog parts of the architecture are implemented as custom layouts, while the digital parts are implemented using the automated place and route tools. These parts are integrated in an analog-on-top flow and the combined final layout is sent to the foundry for fabrication. Upon receiving the fabricated chip, we create a testbench to conduct measurements and characterization. The upcoming sections will describe each of these three prototyping phases in detail.

8.2. MODEL OPTIMIZATION

The objective of this optimization phase is to minimize the hardware resource requirements of the model, while still preserving its classification performance. We begin by analyzing the ECG heartbeat signal which is a recording of voltage against sample index as shown in Figure 8.2. The sample indices represent discrete time instances spaced apart by the sampling duration of the acquisition sensor. The specifics of the data acquisition system are beyond the scope of this thesis and our focus lies on understanding the features within the ECG heartbeat. We observe that the central segment around the heartbeat peak holds the highest influence on the classification outcome. The impact of samples progressively diminishes as we move away from this region. Moreover, the samples in the tail section contribute minimally to classification performance.

We adopt a resampling strategy based on this insight to represent the ECG heartbeat using a strategically chosen subset of data samples. It involves selecting all samples within a 50-sample window from the high-influence area around the peak. Within the moderate influence zone, we select 25 samples within a 50-sample window by selecting every alternate sample. Samples beyond these two windows are discarded. This results in a reduction of the input neurons from 250 to 100, thereby altering the network structure from 250-100-3 to 100-100-3 as depicted in Figure 8.2. This streamlines the model computations and reduces the hardware resource requirements.



Color	Importance	Sample selection
🟨	High	All
🟦	Medium	Interleaved
🟥	Low	Discard

Figure 8.2: Resampling of ECG heartbeat to reduce the number of input neurons in the model.

We now need to train the new 100-100-3 model with resampled ECG heartbeat data. The training process incorporates advanced techniques such as weighted loss function and learning rate scheduler. This helps the network in overcoming the information loss due to ECG resampling. We use a weighted loss function with scaling factors [1,1,5] for [normal, mild, abnormal] classes respectively. It penalizes the mistakes on normal and mild classes equally, while those on abnormal classes incur a 5× penalty. This enables emphasis on correct classification of abnormal class, while maximizing the overall accuracy. Furthermore, we explore various learning rate schedules such as cosine annealing, step, multistep and exponential decay. Among these, the multistep scheduler emerges as the most effective choice for our model. The 100-100-3 model trained with resampled heartbeats using aforementioned advanced features achieves 98% overall accuracy and 95.4% critical accuracy. Thus, it incurs just 0.3% loss on both overall accuracy and critical accuracy compared to the original 250-100-3 model. This demonstrates the effectiveness of resampling in streamlining the model computations, while still maintaining high classification accuracy.

We quantize the trained resampled model to further reduce its hardware resource usage. This quantized model structure is shown in Figure 8.3. Each data sample in the input heartbeat data is quantized to 8 bits. The weights and biases in both layer 1 and layer 2 are also represented using 8 bits. Layer-1 outputs are quantized to 16 bits, while layer-2 outputs quantized to 32 bits. Softmax layer is not required for inference as the class with the highest softmax probability also has the highest raw score. Hence, we employ argmax to determine the output class based on layer-2 outputs. This saves hardware resources that would otherwise be required for softmax computations. The quantized model achieves 97.9% overall accuracy and 94.9% critical accuracy. Thus, it incurs a tiny loss of 0.4% and 0.8% in overall and critical accuracy compared to the original 250-100-3 model, while significantly reducing the hardware resource demands. This makes the resampled and quantized classifier 1 model (100-100-3) well suited for hardware design.

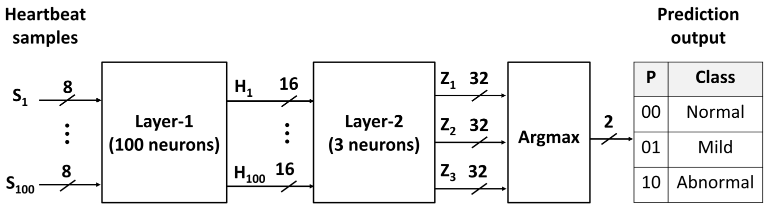


Figure 8.3: The optimized (quantized and resampled) classifier 1 model.

8.3. Hardware Architecture Design

In this phase, we design the hardware architecture for our resampled and quantized classifier 1 model. We first create a hardware-oriented view of this model as shown in Figure 8.4, distinguishing analog and digital blocks through color coding. Each layer involves two-stage mixed-signal data processing. First stage performs vector-matrix multiplication (VMM) in analog domain. This requires time-multiplexed inputs due to large energy consumption associated with high resolution digital-to-analog converters.

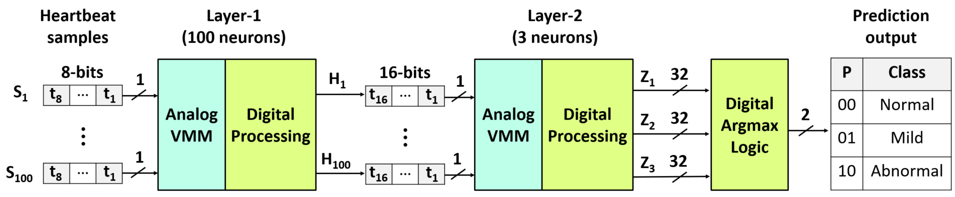


Figure 8.4: Hardware-aware view of the optimized classifier 1 model. The analog and digital blocks are indicated using two different colors.

Moreover, analog VMM internally employs unbalanced bit-slicing as the resistive storage with a 2-bit capacity cannot hold a 8-bit weight. The analog VMM outputs are then converted to digital and undergo subsequent digital processing in the second stage to produce the layer outputs. The argmax logic then determines the output class based on layer-2 outputs.

We now look into the internal operation of layer-1 in Figure 8.5. Its 100x100 weight matrix translates to 100x500 crossbar due to unbalanced bit-slicing that splits each 8-bit weight across 5 columns as [1,1,2,2,2] bits. Similarly, the 1x100 bias vector expands to a 1x500 crossbar structure. The biases are stacked below the weights to create an overall 101x500 crossbar, which enables bias addition as a part of crossbar outputs. The 8-bit heartbeat samples are fed to the crossbar in a time-multiplexed manner. At each timestep,

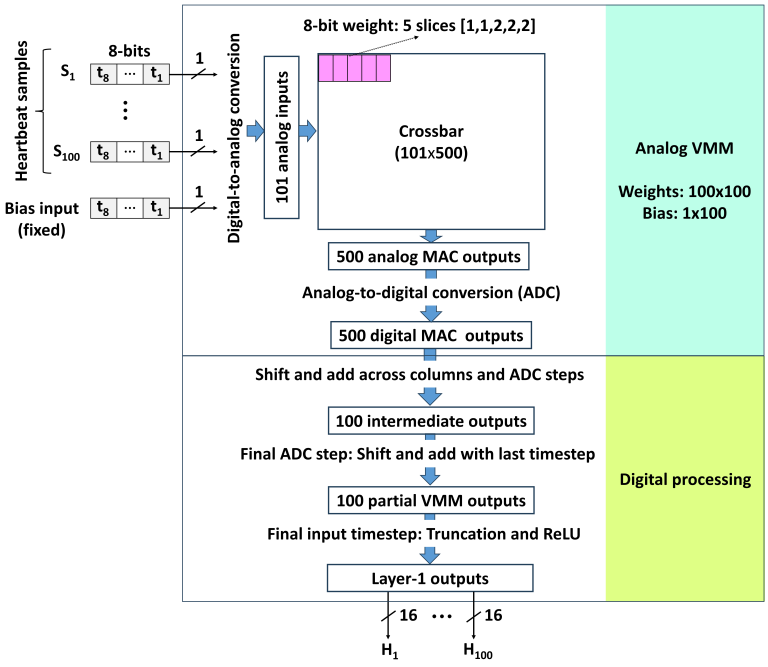


Figure 8.5: Internal working of layer-1 on CIM hardware.

the corresponding bits from all heartbeat samples are converted to analog domain and applied to crossbar rows. The bias row is provided with a constant input. As a result, the crossbar outputs include MAC operation between time-multiplexed inputs and sliced weights along with bias addition. These outputs are converted to digital domain using analog-to-digital converters. The novel ADC in our design delivers bit-by-bit digital conversion across multiple steps (its design details are out of the scope of this thesis). For instance, conversion of analog MAC value to a 7-bit digital output occurs over seven steps, one bit at a time. A straightforward approach would involve collecting all ADC bits and then performing a shift-and-add operation to account for weight slicing across columns. However, this is computationally expensive. To address this, we combine these two steps into a single step through selection of appropriate scaling factors. Once this accumulation is completed across all ADC steps, we merge it with the outcome from previous input timestep. After repeating this process over all the input timesteps, we truncate the resulting values to 16-bits and apply ReLU activation to obtain the final layer-1 outputs.

Layer-2 operates in a similar manner as layer-1, shown in Figure 8.6. Its 100x3 weight matrix requires 100x15 crossbar due to slicing of 8-bit weights across 5 columns through unbalanced bit-slicing. Similarly, its 1x3 bias vector translates to 1x15 crossbar. Stacking biases below the weights then leads to 101x15 crossbar. This crossbar receives 16 bit inputs from layer-1 across 16 timesteps. It produces 15 analog outputs at each timestep,

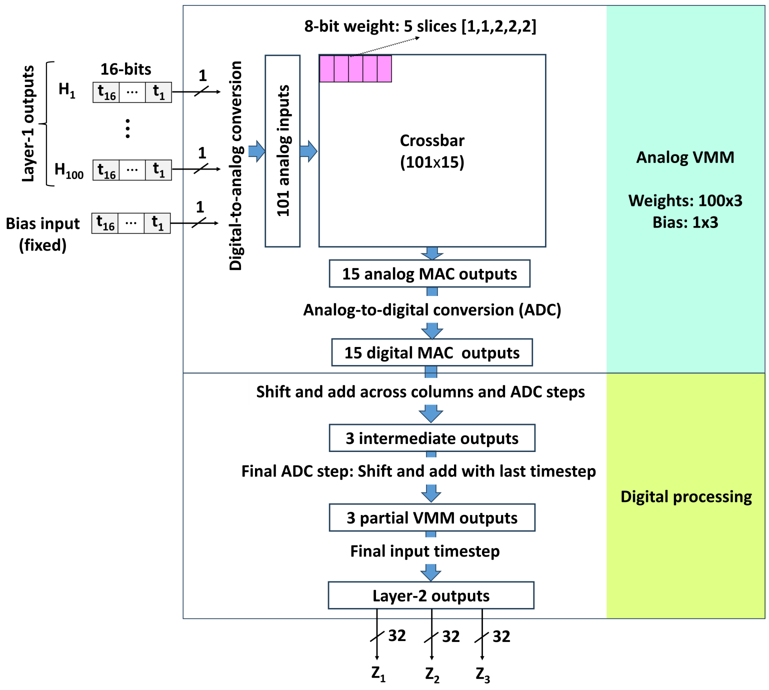


Figure 8.6: Internal working of layer-2 on CIM hardware.

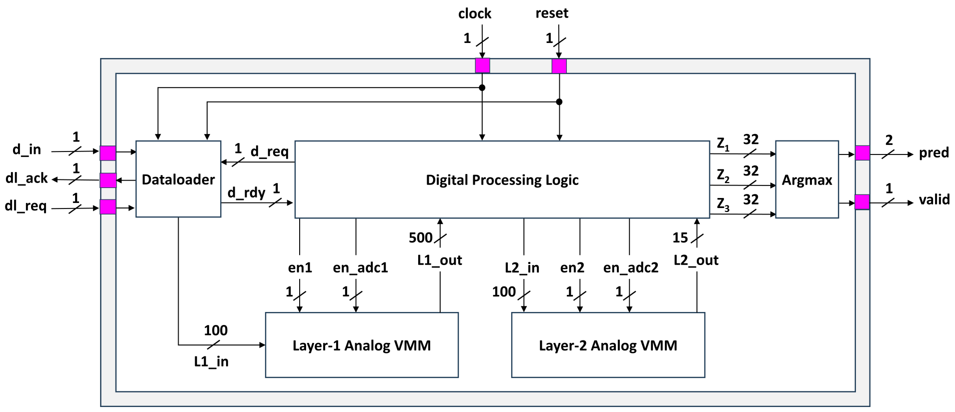


Figure 8.7: ECG classification system architecture.

which undergo a single cumulative shift-and-add operation to account for ADC steps and slicing across columns. At the last ADC step, these values get merged with outcome of the previous input timestep. Repeating this process over all 16 timesteps produces layer-2 outputs. Truncation is not needed as 32 bits suffice to hold layer-2 outputs. Moreover, ReLU is not needed as layer-2 is the final layer of the network. Outputs of layer-2 are then fed into argmax to make a prediction.

We integrate the functionalities of layer-1 and layer-2 with other essential components to form the ECG classification system, illustrated in Figure 8.7. A dataloader serially imports ECG heartbeat into the prototype from an external source (details in the next section). Upon acquiring the heartbeat data, it undergoes analog VMM and subsequent digital processing across the two layers. Each layer's analog VMM is handled by a dedicated hardware block. The digital processing of both layers is encapsulated in the digital processing logic block. Finally, argmax block determines the class with highest classification score as the prediction outcome.

8.4. IMPLEMENTATION AND TESTBENCH

This phase focuses on creating a physical layout for the system architecture in Figure 8.7. The resulting chip layout in TSMC 40nm technology is shown in Figure 8.8. It consists of three analog components and one digital component. The analog components include two analog VMM units and ADC characterization circuits. Each analog VMM unit incorporates a novel CIM crossbar with a new ADC design. The ADC characterization circuits facilitate ADC evaluation independent of the rest of the system. The internal design details of the novel crossbar and ADC are out of the scope of this thesis. All of these analog components are implemented through custom layout process. It involves manual placement and routing of the transistors to achieve the desired functionality. The sole digital component of the layout is denoted as the digital processing unit in Figure 8.8. We leverage RTL (register-transfer level) to layout flow for its implementation. It starts with creating a VHDL description for the digital processing unit, which internally merges

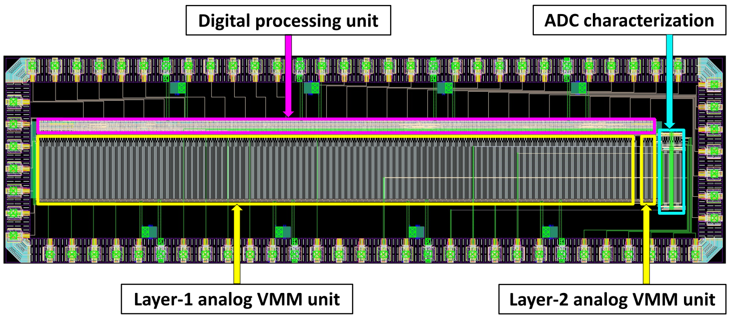


Figure 8.8: Layout of the ECG classification system.

the dataloader, argmax, and digital processing logic blocks of Figure 8.7. This allows sharing of internal computations across these blocks, enabling better optimization in later steps. The VHDL description is then converted to an optimized gate-level netlist through RTL synthesis. Finally, the synthesized netlist undergoes automated placement and routing to produce the optimal transistor-level digital layout. Once the layouts for all the individual components are ready, we combine them into a full chip layout using analog-on-top approach. It begins with exporting the digital component layout into analog layout editor. The imported layout is then manually integrated with the layouts of analog components and the padring. This integrated full-chip layout is sent to the foundry for fabrication. The fabricated chip die is shown in Figure 8.9. It occupies 2.9 mm^2 silicon area and operates at 100 MHz clock frequency. The IO pads of the fabricated chip are bonded to the PCB in Figure 8.9. The PCB features a power distribution network, integrated switches, and biasing circuits for analog crossbars. It also includes jumpers, BNC connectors, level-shifters etc. for interfacing with off-board components.

We next develop an evaluation testbench for the prototype using this PCB, as shown in Figure 8.10. Power supply ports and connections are omitted here for the sake of brevity. The testbench utilizes a waveform generator to provide clock and reset signals to the PCB. The heartbeat data first undergoes preprocessing on a PC. It involves quantizing the

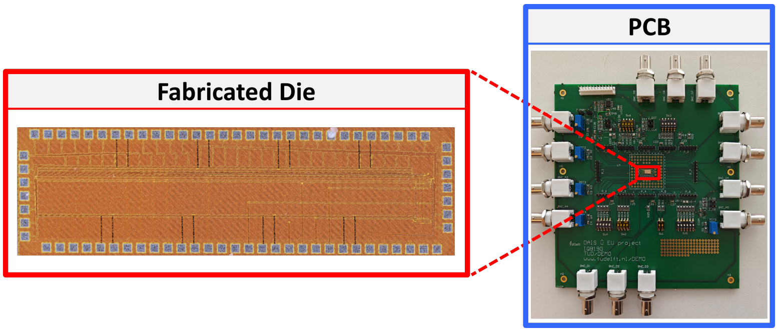


Figure 8.9: The prototype die and its incorporation in the printed circuit board (PCB).

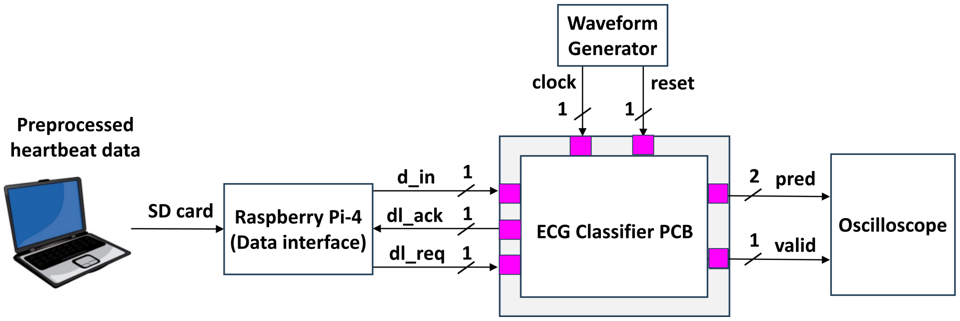


Figure 8.10: Testbench for prototype evaluation.

heartbeat samples to 8 bits, followed by bit-slicing for compatibility with time-multiplexed CIM processing on the prototype. This preprocessed data is loaded onto an SD card, which is subsequently transferred to a Raspberry Pi 4 module. Python script running on the Raspberry Pi transmits the data from the SD card to the PCB via its general-purpose input/output (GPIO) ports. The prototype on the PCB processes this input to produce prediction outcome. Its outputs are then observed using an oscilloscope. Measurements on our prototype using this testbench are currently ongoing.

PART-V

Conclusions

9
Summary and Outlook

9.1. Thesis Summary

We now present a chapter-by-chapter summary of this thesis.

Chapter 1: Introduction
This chapter began with highlighting the importance of edge-AI. We then described the drawbacks of traditional computing and introduced memristor-based computation-in-memory (CIM) paradigm as a promising alternative for edge-AI. Following this, we outlined three main research topics covered in this thesis. The first topic involved developing models suitable for edge-AI in healthcare applications, with a focus on optimizing them for deployment on CIM hardware. The second topic addressed preserving the model accuracy on CIM hardware by dealing with memristor non-idealities. The third topic focused on prototyping the ideas and solutions presented in the previous two research topics. We then described our contributions to each research topic in detail, followed by an overview of thesis organization.

Chapter 2: Fundamentals of Neural Networks and Computation-In-Memory
This chapter started with introducing fundamental terminologies in the field of AI. We then narrowed our focus to neural networks, discussing their types and operational aspects such as training and inference. Afterwards, we introduced CIM system architecture for neural network inference and provided a detailed description of its functioning. We further covered various types of memristive crossbar designs and concluded with a discussion on memristor device technologies.

Chapter 3: Memristor-based CIM for ECG Arrhythmia Classification
This chapter presented energy efficient and accurate cardiac arrhythmia classification on CIM-based edge-AI hardware. Initially, we evaluated the severity impact of various arrhythmia classes. We then devised a severity-based classification approach to enhance support for both end users and medical professionals. Next, we established a hierarchical

classification structure for this approach. It simplifies the overall classification task and activates only the essential network components for a given input. This leads to improved energy efficiency while maintaining high accuracy. Within this structure, we incorporated the most energy-efficient neural network topology customized for CIM hardware, identified through design space exploration. This further reduces energy consumption and area footprint due to in-situ data processing and memristor scalability in CIM hardware. Our proposed classification approach achieved 25× less average energy consumption and 12× less area compared to the state-of-the-art while maintaining high accuracy.

Chapter 4: Memristor-based CIM for Diabetic Retinopathy Screening
This chapter proposed a reliable and energy-diabetic retinopathy (DR) screening suitable for deployment on CIM-based edge-AI hardware. We first assessed the impact of training data quality and diagnostic information on AI-based DR screening. We first created a custom training dataset to overcome reliability issues arising from training data quality. Our dataset incorporated diverse image quality and sources, while also addressing class imbalance issue. This enabled the trained model handle variations in on-field retinal images and perform well on minority DR classes, enhancing its post-deployment reliability. Subsequently, we proposed a pseudo-binary classification approach to provide diagnostic information along with model predictions. This enhanced the real-world effectiveness of our model and also helped in facilitating its widespread adoption. Moreover, this approach improved the classification performance due to its knowledge distillation structure. We then developed a pseudo-binary DR classification model using our custom dataset. Furthermore, we optimized this model for deployment on CIM-based edge hardware through quantization and pruning. Our DR screening solution achieved reliable classification with three orders of magnitude less energy consumption compared to the state-of-the-art hardware platforms.

Chapter 5: Mapping-aware Biased Training for CIM-based Neural Networks
This chapter described a biased training approach to mitigate conductance variation non-ideality in CIM-based neural networks. First, we identified memristor conductance states that exhibited inherent immunity to variation, termed ’favorable states’. Next, we defined a favorability constraint to specify the range of weight values that directly get mapped to these favorable states. Subsequently, we created a baseline model by training in a hardware-unaware manner. We analyzed this baseline model to determine the weights that significantly influence CIM hardware accuracy, called ’important weights’. We then retrained the baseline model while imposing the favorability constraint on important weights. As a result, the retrained values of important weights directly got mapped to favorable states. This led to error-free important computations and improved inference accuracy on CIM hardware. Our biased training approach achieved up to 2.4× hardware accuracy compared to the conventional training.

Chapter 6: Unbalanced Bit-slicing for CIM-based Neural Networks
This chapter introduced a CIM micro-architecture design approach that mitigates non-zero G_{min} error non-ideality. It consisted of two key features. Firstly, the memristor array

design included high sensing margins for most significant bits (MSBs). This countered the detrimental effects of non-zero G_{min} error on these important bits. Secondly, it leveraged 2's complement arithmetic for digital post-processing, whose inherent differential nature further reduced the impact of non-zero G_{min} error. In this approach, the extra sensing margin allocation to MSBs improved accuracy at the cost of some energy overhead. We addressed this issue by proposing a new variant of our approach. It minimized hardware requirements by adjusting sensing margins for less important bits, while keeping high sensing margins for MSBs. Thus, it reduced energy consumption while maintaining the accuracy benefits. It also facilitated a trade-off between accuracy and energy efficiency, to fine-tune these metrics based on application requirements. Our proposed approach achieved up to 7.3× accuracy and up to 7.8× correct operations per unit energy consumption compared to state-of-the-art.

Chapter 7: Adaptive Referencing Architecture for CIM-based Neural Networks
This chapter proposed a CIM micro-architecture design to mitigate read-disturb non-ideality. We began with an analysis to extract key insights about the read-disturb phenomenon. We then used the findings of this analysis to design two key components of this micro-architecture: i) read-disturb detection unit ii) adaptive ADC with control logic. The design of read-disturb detection unit involved a pre-mapping profiling, to determine that crossbar column most vulnerable to read-disturb. The read-disturb detection unit then monitored this column to detect the occurrences of read-disturb during operation. Upon detecting read-disturb, the adjustment of sensing conditions within the adaptive ADC counteracted the effect of read-disturb. This restored error-free operation, resulting in improved inference accuracy. Our proposed architecture provided up to 2× accuracy compared to conventional CIM architectures.

Chapter 8: CIM-based ECG Classification Prototype
This chapter presented a CIM prototype for ECG classification. It incorporated the first hierarchical level in the ECG classification model and unbalanced bit-slicing. This prototype was developed through a three-phase process. First, the model was optimized through resampling and quantization to reduce hardware resource usage. Second, a mixed-signal system architecture was designed. It comprised of analog vector-matrix multiplication (VMM) units and digital processing logic. Last, layout implementation for this architecture was carried out in TSMC 40nm technology and sent to the foundry for fabrication. Upon receiving the fabricated prototype, we created a testbench for its measurement and characterization.

9.2. FUTURE RESEARCH DIRECTIONS

In this section, we outline potential future research avenues that can advance the contributions of this thesis.

On-chip Training for CIM Hardware

Neural networks are typically trained on massive datasets stored in the cloud. Edge devices then download the trained models for real-time inference. The accuracy of this inferences depends heavily on how well the training data reflects real-world scenarios encountered by edge devices. However, it is not possible for cloud-based training datasets to capture every single on-field variation. The potential discrepancies between training data and on-field data can then lead to degraded inference accuracy. A potential solution to this problem involves sending real-time data back to the cloud for retraining. However, waiting for model updates from the cloud can stall the real-time inference for a long duration. Such delays are not acceptable in real-time applications and scenarios demanding quick adaptation to evolving environments. Therefore, equipping edge hardware with on-board training capabilities presents a promising avenue for future research. This would facilitate real-time model adaptation in response to on-field data, ensuring high inference accuracy.

CIM-based Spiking Neural Networks

Spiking neural networks (SNNs) offer a compelling avenue for healthcare applications, due to their distinctive data processing abilities and potential for high energy efficiency. Unlike conventional neural networks that rely on real-valued computations, SNNs use discrete events called spikes for data processing. This event-driven nature makes them significantly more energy-efficient compared to conventional neural networks. Thus, combining SNNs with CIM-based hardware has the potential to achieve brain-like energy-efficiency. Additionally, SNNs excel at handling spatio-temporal information, a characteristic prevalent in biomedical signals. The spatio-temporal data processing and potential for brain-like energy efficiency makes CIM-based SNNs ideal for energy-constrained healthcare devices like wearables, implants etc. Furthermore, SNNs are inherently robust against memristor non-idealities as their computations rely on spikes instead of precise numeric values. Hence, CIM-based SNN design emerges as an attractive research direction towards realizing accurate and energy-efficient healthcare edge-AI.

On-chip Communication Strategies for CIM Architecture

CIM systems leverage a tiled architecture to exploit the inherent parallelism in neural network computations. These tiles communicate with each other using on-chip interconnects. As neural networks follow a layered structure, tiles of a given layer send their outputs to the tiles of next layer, through these interconnects. However, the ever-growing complexity of neural networks creates a corresponding surge in on-chip communication. This can lead to a substantial increase in energy consumption, potentially negating the energy efficiency advantages of CIM. To address this issue, designing energy-efficient communication strategies for tiled CIM architectures is crucial. Furthermore, due to diverse computational needs of different neural network types, a single solution might not be effective. Therefore, developing network-specific on-chip communication strategies is a promising area for future research in CIM system architecture.

Bibliography

[1] *Google, Facebook, and Microsoft Are Remaking Themselves around AI.* URL: https://www.wired.com/2016/11/google-facebook-microsoft-remaking-around-ai/.

[2] M. Capra et al. "Edge Computing: A Survey On the Hardware Requirements in the Internet of Things World". *Future Internet* 11.4 (2019).

[3] R. Parloff. *From 2016: Why Deep Learning Is Suddenly Changing Your Life and Will Soon Transform Corporate America.* 2016. URL: https://fortune.com/longform/ai-artificial-intelligence-deep-machine-learning/.

[4] *What is edge AI?* URL: https://www.ibm.com/topics/edge-ai.

[5] S. Shukla, B. Fleischer, M. Ziegler, J. Silberman, J. Oh, V. Srinivasan, J. Choi, S. Mueller, A. Agrawal, T. Babinsky, N. Cao, C.-Y. Chen, P. Chuang, T. Fox, G. Gristede, M. Guillorn, H. Haynie, M. Klaiber, D. Lee, S.-H. Lo, G. Maier, M. Scheuermann, S. Venkataramani, C. Vezyrtzis, N. Wang, F. Yee, C. Zhou, P.-F. Lu, B. Curran, L. Chang, and K. Gopalakrishnan. "A Scalable Multi-TeraOPS Core for AI Training and Inference". *IEEE Solid-State Circuits Letters* 1.12 (2018), pp. 217–220.

[6] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer. "Efficient Processing of Deep Neural Networks: A Tutorial and Survey". *Proceedings of the IEEE* 105.12 (2017), pp. 2295–2329.

[7] M. Goudarzi, T. Ishihara, and H. Noori. "Variation-aware software techniques for cache leakage reduction using value-dependence of SRAM leakage due to within-die process variation". *Proceedings of the 3rd International Conference on High Performance Embedded Architectures and Compilers.* 2008, pp. 224–239.

[8] I. Bhati, M.-T. Chang, Z. Chishti, S.-L. Lu, and B. Jacob. "DRAM Refresh Mechanisms, Penalties, and Trade-Offs". *IEEE Transactions on Computers* 65.1 (2016), pp. 108–121.

[9] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu. "Memory power management via dynamic voltage/frequency scaling". *Proceedings of the 8th ACM International Conference on Autonomic Computing.* 2011, pp. 31–40.

[10] P. Y. Chen and S. Yu. "Technological Benchmark of Analog Synaptic Devices for Neuroinspired Architectures". *IEEE Design and Test* 36.3 (2019), pp. 31–38.

[11] *Memory Technology: Putting the "nano" in your iPod, Prof. Eric Pop, University of Illinois at Urbana-Champaign.* URL: http://poplab.stanford.edu/pdfs/EPop-UniHigh-Memory-May2008.pdf.

[12] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun. “A Modern Primer on Processing in Memory”. *Emerging Computing: From Devices to Systems: Looking Beyond Moore and Von Neumann.* Ed. by M. M. S. Aly and A. Chattopadhyay. Singapore: Springer Nature Singapore, 2023, pp. 171–243.

[13] P. Y. Chen, X. Peng, and S. Yu. “NeuroSim: A Circuit-Level Macro Model for Benchmarking Neuro-Inspired Architectures in Online Learning”. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37.12 (2018), pp. 3067–3080.

[14] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou. “Memory Devices and Applications for In-Memory Computing”. *Nature Nanotechnology* 15.7 (2020), pp. 529–544.

[15] X. Yang, B. Taylor, A. Wu, Y. Chen, and L. O. Chua. “Research Progress on Memristor: From Synapses to Computing Systems”. *IEEE Transactions on Circuits and Systems I: Regular Papers* 69.5 (2022), pp. 1845–1857.

[16] *How Edge Computing is Transforming Healthcare.* URL: https://resources.nvidia.com/en-us-fleet-command/healthcare-at-the-edge.

[17] J. Rehm, C. Mathers, S. Popova, M. Thavorncharoensap, Y. Teerawattananon, and J. Patra. “Global, Regional, and National Age-Sex Specific Mortality for 264 Causes of Death, 1980-2016: A Systematic Analysis for the Global Burden of Disease Study 2016”. *The Lancet* 373.9682 (2009), p. 2223.

[18] *Sight for All.* URL: https://sightforall.org/diabetic-retinopathy-initiative/.

[19] Y. Zhou, B. Wang, L. Huang, S. Cui, and L. Shao. “A Benchmark for Studying Diabetic Retinopathy: Segmentation, Grading, and Transferability”. *IEEE Transactions on Medical Imaging* 40.3 (2021), pp. 818–828.

[20] L. Chen, J. Li, Y. Chen, Q. Deng, J. Shen, X. Liang, and L. Jiang. “Accelerator-Friendly Neural-Network Training: Learning Variations and Defects in RRAM Crossbar”. *Proceedings of the 2017 Design, Automation and Test in Europe, DATE 2017.* 2017, pp. 19–24.

[21] J. C. Babu, Y. Suresh, R. S. Rani, S. Yasmeen, K. S. R. K. Reddy, and K. Harshavardhan. “Accelerated Addition in Resistive Ram Array Using Parallel-Friendly Majority Gates”. *Studies in Computational Intelligence* 1117 (2024), pp. 111–119.

[22] W. Shim, Y. Luo, J. S. Seo, and S. Yu. “Impact of Read Disturb on Multilevel RRAM Based Inference Engine: Experiments and Model Prediction”. *IEEE International Reliability Physics Symposium Proceedings.* Vol. 2020-April. 2020.

[23] W. Shim, Y. Luo, J. S. Seo, and S. Yu. “Investigation of Read Disturb and Bipolar Read Scheme on Multilevel RRAM-Based Deep Learning Inference Engine”. *IEEE Transactions on Electron Devices* 67.6 (2020), pp. 2318–2323.

[24] S. Diware, S. Dash, A. Gebregiorgis, R. V. Joshi, C. Strydis, S. Hamdioui, and R. Bishnoi. “Severity-Based Hierarchical ECG Classification Using Neural Networks”. *IEEE Transactions on Biomedical Circuits and Systems* 17.1 (2023), pp. 77–91.

[25] S. Diware, K. Chilakala, R. V. Joshi, S. Hamdioui, and R. Bishnoi. “Reliable and Energy-Efficient Diabetic Retinopathy Screening Using Memristor-Based Neural Networks”. *IEEE Access* 12 (2024), pp. 47469–47482.

[26] S. Diware, A. Gebregiorgis, R. V. Joshi, S. Hamdioui, and R. Bishnoi. “Mapping-aware Biased Training for Accurate Memristor-based Neural Networks”. *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. 2023, pp. 1–5.

[27] S. Diware, A. Singh, A. Gebregiorgis, R. V. Joshi, S. Hamdioui, and R. Bishnoi. “Accurate and Energy-Efficient Bit-Slicing for RRAM-Based Neural Networks”. *IEEE Transactions on Emerging Topics in Computational Intelligence* 7.1 (2023), pp. 164–177.

[28] S. Diware, A. Gebregiorgis, R. V. Joshi, S. Hamdioui, and R. Bishnoi. “Unbalanced Bit-slicing Scheme for Accurate Memristor-based Neural Network Architecture”. *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. 2021, pp. 1–4.

[29] S. Diware, M. A. Yaldagard, A. Gebregiorgis, R. V. Joshi, S. Hamdioui, and R. Bishnoi. “Dynamic Detection and Mitigation of Read-disturb for Accurate Memristor-based Neural Networks”. *2024 IEEE 6th International Conference on AI Circuits and Systems (AICAS)*. 2024, pp. 393–397.

[30] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar. “ISAAC: A Convolutional Neural Network Accelerator with in-Situ Analog Arithmetic in Crossbars”. *Proceedings - 2016 43rd International Symposium on Computer Architecture, ISCA 2016*. 2016, pp. 14–26.

[31] A. Ankit, I. El Hajj, S. Rahul Chalamalasetti, G. Ndu, M. Foltin, R. S. Williams, P. Faraboschi, W. M. Hwu, J. Paul Strachan, K. Roy, and D. S. Milojicic. “PUMA: A Programmable Ultra-Efficient Memristor-Based Accelerator for Machine Learning Inference”. *International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS*. 2019, pp. 715–731.

[32] A. Ankit, I. E. Hajj, S. R. Chalamalasetti, S. Agarwal, M. Marinella, M. Foltin, J. P. Strachan, D. Milojicic, W. M. Hwu, and K. Roy. “PANTHER: A Programmable Architecture for Neural Network Training Harnessing Energy-Efficient ReRAM”. *IEEE Transactions on Computers* 69.8 (2020), pp. 1128–1142.

[33] Y. Deng, P. Huang, B. Chen, X. Yang, B. Gao, J. Wang, L. Zeng, G. Du, J. Kang, and X. Liu. “RRAM Crossbar Array With Cell Selection Device: A Device and Circuit Interaction Study”. *IEEE Transactions on Electron Devices* 60.2 (2013), pp. 719–726.

[34] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou. “Memory devices and applications for in-memory computing”. *Nature Nanotechnology* 15.7 (2020), pp. 529–544.

[35] Rajendran, Gokulnath and Banerjee, Writam and Chattopadhyay, Anupam and Aly, Mohamed M. Sabry. “Application of Resistive Random Access Memory in Hardware Security: A Review”. *Advanced Electronic Materials* 7.12 (2021), p. 2100536.

[36] M. L. Gallo and A. Sebastian. "An overview of phase-change memory device physics". *Journal of Physics D: Applied Physics* 53.21 (2020), p. 213002.

[37] W. Xu and I. Koren. "A Scalable Wear Leveling Technique for Phase Change Memory". *ACM Trans. Storage* 20.1 (2024).

[38] A. M. Hosseini Monazzah, A. M. Rahmani, A. Miele, and N. Dutt. "CAST: Content-Aware STT-MRAM Cache Write Management for Different Levels of Approximation". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39.12 (2020), pp. 4385–4398.

[39] N. Mahdavi, F. Razaghian, and H. Farbeh. "An Architectural-Level Reliability Improvement Scheme in STT-MRAM Main Memory". *Microprocessors and Microsystems* 90 (2022), p. 104462.

[40] T. Kawahara, K. Ito, R. Takemura, and H. Ohno. "Spin-transfer torque RAM technology: Review and prospect". *Microelectronics Reliability* 52.4 (2012), pp. 613–627.

[41] S. K. Kim, T. W. Oh, S. Lim, D. H. Ko, and S.-O. Jung. "High-Performance and Area-Efficient Ferroelectric FET-Based Nonvolatile Flip-Flops". *IEEE Access* 9 (2021), pp. 35549–35561.

[42] S. Kumar, S. Chatterjee, S. Thomann, Y. S. Chauhan, and H. Amrouch. "Cross-Layer Reliability Modeling of Dual-Port FeFET: Device-Algorithm Interaction". *IEEE Transactions on Circuits and Systems I: Regular Papers* 70.7 (2023), pp. 2891–2903.

[43] C.-C. Chou, Z.-J. Lin, P.-L. Tseng, C.-F. Li, C.-Y. Chang, W.-C. Chen, Y.-D. Chih, and T.-Y. J. Chang. "An N40 256K×44 embedded RRAM macro with SL-precharge SA and low-voltage current limiter to improve read and write performance". *2018 IEEE International Solid-State Circuits Conference - (ISSCC)*. 2018, pp. 478–480.

[44] C.-F. Yang, C.-Y. Wu, M.-H. Yang, W. Wang, M.-T. Yang, T.-C. Chien, V. Fan, S.-C. Tsai, Y.-H. Lee, W.-T. Chu, and A. Hung. "Industrially Applicable Read Disturb Model and Performance on Mega-Bit 28nm Embedded RRAM". *2020 IEEE Symposium on VLSI Technology*. 2020, pp. 1–2.

[45] C.-C. Chou, Z.-J. Lin, C.-A. Lai, C.-I. Su, P.-L. Tseng, W.-C. Chen, W.-C. Tsai, W.-T. Chu, T.-C. Ong, H. Chuang, Y.-D. Chih, and T.-Y. J. Chang. "A 22nm 96KX144 RRAM Macro with a Self-Tracking Reference and a Low Ripple Charge Pump to Achieve a Configurable Read Window and a Wide Operating Voltage Range". *2020 IEEE Symposium on VLSI Circuits*. 2020, pp. 1–2.

[46] P. Jain, U. Arslan, M. Sekhar, B. C. Lin, L. Wei, T. Sahu, J. Alzate-vinasco, A. Vangapaty, M. Meterelliyoz, N. Strutt, A. B. Chen, P. Hentges, P. A. Quintero, C. Connor, O. Golonzka, K. Fischer, and F. Hamzaoglu. "13.2 A 3.6Mb 10.1Mb/mm2 Embedded Non-Volatile ReRAM Macro in 22nm FinFET Technology with Adaptive Forming/Set/Reset Schemes Yielding Down to 0.5V with Sensing Time of 5ns at 0.7V". *2019 IEEE International Solid-State Circuits Conference - (ISSCC)*. 2019, pp. 212–214.

[47] J. Wu, Y. Chen, W. S. Khwa, S. M. Yu, T. Y. Wang, J. Tseng, Y. Chih, and C. H. Diaz. “A 40nm Low-Power Logic Compatible Phase Change Memory Technology”. *2018 IEEE International Electron Devices Meeting (IEDM)*. 2018, pp. 27.6.1–27.6.4.

[48] F. Arnaud, P. Zuliani, J. Reynard, A. Gandolfo, F. Disegni, P. Mattavelli, E. Gomiero, G. Samanni, C. Jahan, R. Berthelon, O. Weber, E. Richard, V. Barral, A. Villaret, S. Kohler, J. Grenier, R. Ranica, C. Gallon, A. Souhaite, D. Ristoiu, L. Favennec, V. Caubet, S. Delmedico, N. Cherault, R. Beneyton, S. Chouteau, P. Sassoulas, A. Vernhet, Y. Le Friec, F. Domengie, L. Scotti, D. Pacelli, J. Ogier, F. Boucard, S. Lagrasta, D. Benoit, L. Clement, P. Boivin, P. Ferreira, R. Annunziata, and P. Cappelletti. “Truly Innovative 28nm FDSOI Technology for Automotive Micro-Controller Applications embedding 16MB Phase Change Memory”. *2018 IEEE International Electron Devices Meeting (IEDM)*. 2018, pp. 18.4.1–18.4.4.

[49] Y.-D. Chih, Y.-C. Shih, C.-F. Lee, Y.-A. Chang, P.-H. Lee, H.-J. Lin, Y.-L. Chen, C.-P. Lo, M.-C. Shih, K.-H. Shen, H. Chuang, and T.-Y. J. Chang. “13.3 A 22nm 32Mb Embedded STT-MRAM with 10ns Read Speed, 1M Cycle Write Endurance, 10 Years Retention at 150°C and High Immunity to Magnetic Field Interference”. *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*. 2020, pp. 222–224.

[50] L. Wei, J. G. Alzate, U. Arslan, J. Brockman, N. Das, K. Fischer, T. Ghani, O. Golonzka, P. Hentges, R. Jahan, P. Jain, B. Lin, M. Meterelliyoz, J. O’Donnell, C. Puls, P. Quintero, T. Sahu, M. Sekhar, A. Vangapaty, C. Wiegand, and F. Hamzaoglu. “13.3 A 7Mb STT-MRAM in 22FFL FinFET Technology with 4ns Read Sensing Time at 0.9V Using Write-Verify-Write Scheme and Offset-Cancellation Sensing Technique”. *2019 IEEE International Solid-State Circuits Conference - (ISSCC)*. 2019, pp. 214–216.

[51] V. B. Naik, K. Lee, K. Yamane, R. Chao, J. Kwon, N. Thiyagarajah, N. L. Chung, S. H. Jang, B. Behin-Aein, J. H. Lim, T. Y. Lee, W. P. Neo, H. Dixit, S. K, L. C. Goh, T. Ling, J. Hwang, D. Zeng, J. W. Ting, E. H. Toh, L. Zhang, R. Low, N. Balasankaran, L. Y. Zhang, K. W. Gan, L. Y. Hau, J. Mueller, B. Pfefferling, O. Kallensee, S. L. Tan, C. S. Seet, Y. S. You, S. T. Woo, E. Quek, S. Y. Siah, and J. Pellerin. “Manufacturable 22nm FD-SOI Embedded MRAM Technology for Industrial-grade MCU and IOT Applications”. *2019 IEEE International Electron Devices Meeting (IEDM)*. 2019, pp. 2.3.1–2.3.4.

[52] Y. J. Song, J. H. Lee, S. H. Han, H. C. Shin, K. H. Lee, K. Suh, D. E. Jeong, G. H. Koh, S. C. Oh, J. H. Park, S. O. Park, B. J. Bae, O. I. Kwon, K. H. Hwang, B. Seo, Y. Lee, S. H. Hwang, D. S. Lee, Y. Ji, K. Park, G. T. Jeong, H. S. Hong, K. P. Lee, H. K. Kang, and E. S. Jung. “Demonstration of Highly Manufacturable STT-MRAM Embedded in 28nm Logic”. *2018 IEEE International Electron Devices Meeting (IEDM)*. 2018, pp. 18.2.1–18.2.4.

[53] M. Trentzsch, S. Flachowsky, R. Richter, J. Paul, B. Reimer, D. Utess, S. Jansen, H. Mulaosmanovic, S. Müller, S. Slesazeck, J. Ocker, M. Noack, J. Müller, P. Polakowski, J. Schreiter, S. Beyer, T. Mikolajick, and B. Rice. “A 28nm HKMG super low power embedded NVM technology based on ferroelectric FETs”. *2016 IEEE International Electron Devices Meeting (IEDM)*. 2016, pp. 11.5.1–11.5.4.

[54] S. Dünkel, M. Trentzsch, R. Richter, P. Moll, C. Fuchs, O. Gehring, M. Majer, S. Wittek, B. Müller, T. Melde, H. Mulaosmanovic, S. Slesazeck, S. Müller, J. Ocker, M. Noack, D.-A. Löhr, P. Polakowski, J. Müller, T. Mikolajick, J. Höntschel, B. Rice, J. Pellerin, and S. Beyer. "A FeFET based super-low-power ultra-fast embedded NVM technology for 22nm FDSOI and beyond". *2017 IEEE International Electron Devices Meeting (IEDM)*. 2017, pp. 19.7.1–19.7.4.

[55] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian. "Fully Hardware-Implemented Memristor Convolutional Neural Network". *Nature* 577.7792 (2020), pp. 641–646.

[56] W. Kim, R. Bruce, T. Masuda, G. Fraczak, N. Gong, P. Adusumilli, S. Ambrogio, H. Tsai, J. Bruley, J.-P. Han, M. Longstreet, F. Carta, K. Suu, and M. BrightSky. "Confined PCM-based Analog Synaptic Devices offering Low Resistance-drift and 1000 Programmable States for Deep Learning". *2019 Symposium on VLSI Technology*. 2019, T66–T67.

[57] S. Diware, Y. Dong, M. A. Yaldagard, and R. Bishnoi. "Adaptive Multi-Threshold Encoding for Energy-Efficient ECG Classification Architecture using Spiking Neural Network". *accepted in Design, Automation and Test in Europe (DATE)*. 2025.

[58] Go AS, Mozaffarian D, Roger VL, Benjamin EJ, Berry JD, and E. al. *Heart Disease and Stroke Statistics*. 2013. URL: https://www.heart.org/idc/groups/ahamah-public/@wcm/@sop/@smd/documents/downloadable/ucm_470704.pdf.

[59] J. Wu, F. Li, Z. Chen, Y. Pu, and M. Zhan. "A Neural Network-Based ECG Classification Processor with Exploitation of Heartbeat Similarity". *IEEE access : practical innovations, open solutions* 7 (2019), pp. 172774–172782.

[60] J. Xiao et al. "ULECGNet: An Ultra-Lightweight End-to-End ECG Classification Neural Network". *IEEE Journal of Biomedical and Health Informatics* 26.1 (2022), pp. 206–217.

[61] N. Wang, J. Zhou, G. Dai, J. Huang, and Y. Xie. "Energy-Efficient Intelligent ECG Monitoring for Wearable Devices". *IEEE Transactions on Biomedical Circuits and Systems* 13.5 (2019), pp. 1112–1121.

[62] F. C. Bauer, D. R. Muir, and G. Indiveri. "Real-Time Ultra-Low Power ECG Anomaly Detection Using an Event-Driven Neuromorphic Processor". *IEEE transactions on biomedical circuits and systems* 13.6 (2019), pp. 1575–1582.

[63] F. Qiao, B. Li, Y. Zhang, H. Guo, W. Li, and S. Zhou. "A Fast and Accurate Recognition of ECG Signals Based on ELM-LRF and BLSTM Algorithm". *IEEE access : practical innovations, open solutions* 8 (2020), pp. 71189–71198.

[64] O. Yildirim, U. B. Baloglu, R. S. Tan, E. J. Ciaccio, and U. R. Acharya. "A New Approach for Arrhythmia Classification Using Deep Coded Features and LSTM Networks". *Computer Methods and Programs in Biomedicine* 176 (2019), pp. 121–133.

[65] Ö. Yıldırım, P. Pławiak, R. S. Tan, and U. R. Acharya. "Arrhythmia Detection Using Deep Convolutional Neural Network with Long Duration ECG Signals". *Computers in Biology and Medicine* 102 (2018), pp. 411–420.

[66] Ö. Yildirim. “A Novel Wavelet Sequences Based on Deep Bidirectional LSTM Network Model for ECG Signal Classification”. *Computers in Biology and Medicine* 96 (2018), pp. 189–202.

[67] J. H. Tan, Y. Hagiwara, W. Pang, I. Lim, S. L. Oh, M. Adam, R. S. Tan, M. Chen, and U. R. Acharya. “Application of Stacked Convolutional and Long Short-Term Memory Network for Accurate Identification of CAD ECG Signals”. *Computers in Biology and Medicine* 94 (2018), pp. 19–26.

[68] Y. J. Lin, C. W. Chuang, C. Y. Yen, S. H. Huang, P. W. Huang, J. Y. Chen, and S. Y. Lee. “Artificial Intelligence of Things Wearable System for Cardiac Disease Detection”. *Proceedings 2019 IEEE International Conference on Artificial Intelligence Circuits and Systems, AICAS 2019*. 2019, pp. 67–70.

[69] F. Corradi, S. Pande, J. Stuijt, N. Qiao, S. Schaafsma, G. Indiveri, and F. Catthoor. “ECG-based Heartbeat Classification in Neuromorphic Hardware”. *Proceedings of the International Joint Conference on Neural Networks*. Vol. 2019-July. 2019, pp. 1–8.

[70] A. M. Hassan, A. F. Khalaf, K. S. Sayed, H. H. Li, and Y. Chen. “Real-Time Cardiac Arrhythmia Classification Using Memristor Neuromorphic Computing System”. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*. Vol. 2018-July. 2018, pp. 2567–2570.

[71] G. B. Moody and R. G. Mark. “The Impact of the MIT-BIH Arrhythmia Database”. *IEEE Engineering in Medicine and Biology Magazine* 20.3 (2001), pp. 45–50.

[72] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C. K. Peng, and H. E. Stanley. “PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals.” *Circulation* 101.23 (2000), e215–e220.

[73] ECAR, AAMI. *Recommended Practice for Testing and Reporting Performance Results of Ventricular Arrhythmia Detection Algorithms*. Tech. rep. Arlington, VA, USA: Association for the Advancement of Medical Instrumentation, 1987.

[74] R. Thilagavathy, R. Srivatsan, S. Sreekarun, D. Sudeshna, P. Lakshmi Priya, and B. Venkataramani. “Real-Time ECG Signal Feature Extraction and Classification Using Support Vector Machine”. *2020 International Conference on Contemporary Computing and Applications, IC3A 2020*. 2020, pp. 44–48.

[75] V. Fuster. *Hurst’s the Heart*. McGraw Hill, 2017.

[76] J. Jameson et al. *Harrison’s Manual of Medicine 20th Edition*. McGraw Hill, 2020.

[77] *CSC321 Lecture 5: Multilayer Perceptrons, Roger Grosse*. URL: https://www.cs.toronto.edu/~rgrosse/courses/csc321_2018/slides/lec05.pdf.

[78] *Multilayer Perceptrons & Neural Networks: Basics, Kalev Kask*. URL: https://www.ics.uci.edu/~kkask/Spring-2018%20CS273P/slides/08-mlpercept.pdf.

[79] S. Hochreiter and J. Schmidhuber. “Long Short-Term Memory”. *Neural Computation* 9.8 (1997), pp. 1735–1780.

[80] *Understanding LSTM Networks*. URL: https://www.cse.iitk.ac.in/users/sigml/lec/Slides/LSTM.pdf.

[81] Z. Cui, R. Ke, Z. Pu, and Y. Wang. "Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-Wide Traffic Speed Prediction". *ArXiv* abs/1801.0 (2018). URL: http://arxiv.org/abs/1801.02143.

[82] R. Brueckner and B. Schulter. "Social Signal Classification Using Deep Blstm Recurrent Neural Networks". *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. 2014, pp. 4823–4827.

[83] S. Bai, J. Z. Kolter, and V. Koltun. "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling". *ArXiv* abs/1803.0 (2018). URL: http://arxiv.org/abs/1803.01271.

[84] C. Pelletier, G. I. Webb, and F. Petitjean. "Temporal Convolutional Neural Network for the Classification of Satellite Image Time Series". *Remote Sensing* 11.5 (2019).

[85] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-Based Learning Applied to Document Recognition". *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2323.

[86] J. Pan and W. J. Tompkins. "A Real-Time QRS Detection Algorithm". *IEEE Transactions on Biomedical Engineering* BME-32.3 (1985), pp. 230–236.

[87] N. Bayasi, H. Saleh, B. Mohammad, and M. Ismail. "65-Nm ASIC Implementation of QRS Detector Based on Pan and Tompkins Algorithm". *2014 10th International Conference on Innovations in Information Technology, IIT 2014*. Nov. 2014, pp. 84–87.

[88] A. Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[89] T. Tieleman and G. Hinton. *Divide the Gradient by a Running Average of Its Recent Magnitude*. 2012. URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[90] S. Diware, K. Chilakala, and R. Bishnoi. "Computation-In-Memory for Reliable and Energy-Efficient Diabetic Retinopathy Screening". *Springer book chapter (under publishing logistics)*. 2025.

[91] M. Feng, J. Wang, K. Wen, and J. Sun. "Grading of Diabetic Retinopathy Images Based on Graph Neural Network". *IEEE access : practical innovations, open solutions* 11 (2023), pp. 98391–98401.

[92] M. B. Khan, M. Ahmad, S. B. Yaakob, R. Shahrior, M. A. Rashid, and H. Higa. "Automated Diagnosis of Diabetic Retinopathy Using Deep Learning: On the Search of Segmented Retinal Blood Vessel Images for Better Performance". *Bioengineering* 10.4 (2023).

[93] W. K. Wong, F. H. Juwono, and C. Apriono. "Diabetic Retinopathy Detection and Grading: A Transfer Learning Approach Using Simultaneous Parameter Optimization and Feature-Weighted ECOC Ensemble". *IEEE access : practical innovations, open solutions* 11 (2023), pp. 83004–83016.

[94] M. R. Islam, L. F. Abdulrazak, M. Nahiduzzaman, M. O. F. Goni, M. S. Anower, M. Ahsan, J. Haider, and M. Kowalski. “Applying Supervised Contrastive Learning for the Detection of Diabetic Retinopathy and Its Severity Levels from Fundus Images”. *Computers in Biology and Medicine* 146 (2022), p. 105602.

[95] S. S. Mohammadi et al. “A User-Friendly Approach for the Diagnosis of Diabetic Retinopathy Using ChatGPT and Automated Machine Learning”. *Ophthalmology Science* (2024), p. 100495.

[96] N. Sikder, M. Masud, A. K. Bairagi, A. S. M. Arif, A. A. Nahid, and H. A. Alhumyani. “Severity Classification of Diabetic Retinopathy Using an Ensemble Learning Algorithm through Analyzing Retinal Images”. *Symmetry* 13.4 (2021).

[97] V. K. R. Poranki and B. Srinivasarao. “Computer-Aided Diagnosis-Based Grading Classification of Diabetic Retinopathy Using Deep Graph Correlation Network with IRF”. *SN Computer Science* 5.2 (2024).

[98] A. Sadeghzadeh, M. S. Junayed, T. Aydin, and M. B. Islam. “Hybrid CNN+Transformer for Diabetic Retinopathy Recognition and Grading”. *2023 Innovations in Intelligent Systems and Applications Conference, ASYU 2023*. 2023, pp. 1–6.

[99] A. R. Wahab Sait. “A Lightweight Diabetic Retinopathy Detection Model Using a Deep-Learning Technique”. *Diagnostics* 13.19 (2023).

[100] W. L. Alyoubi, M. F. Abulkhair, and W. M. Shalash. “Diabetic Retinopathy Fundus Image Classification and Lesions Localization System Using Deep Learning”. *Sensors* 21.11 (2021).

[101] T. Li, Y. Gao, K. Wang, S. Guo, H. Liu, and H. Kang. “Diagnostic Assessment of Deep Learning Algorithms for Diabetic Retinopathy Screening”. *Information Sciences* 501 (2019), pp. 511–522.

[102] E. Korot, Z. Guan, D. Ferraz, S. K. Wagner, G. Zhang, X. Liu, L. Faes, N. Pontikos, S. G. Finlayson, H. Khalid, G. Moraes, K. Balaskas, A. K. Denniston, and P. A. Keane. “Code-Free Deep Learning for Multi-Modality Medical Image Classification”. *Nature Machine Intelligence* 3.4 (2021), pp. 288–298.

[103] C. A. Ludwig, C. Perera, D. Myung, M. A. Greven, S. J. Smith, R. T. Chang, and T. Leng. “Automatic Identification of Referral-Warranted Diabetic Retinopathy Using Deep Learning on Mobile Phone Images”. *Translational Vision Science and Technology* 9.2 (2020), pp. 1–7.

[104] G. Quellec, K. Charrière, Y. Boudi, B. Cochener, and M. Lamard. “Deep Image Mining for Diabetic Retinopathy Screening”. *Medical Image Analysis* 39 (2017), pp. 178–193.

[105] R. Sayres, A. Taly, E. Rahimy, K. Blumer, D. Coz, N. Hammel, J. Krause, A. Narayanaswamy, Z. Rastegar, D. Wu, S. Xu, S. Barb, A. Joseph, M. Shumski, J. Smith, A. B. Sood, G. S. Corrado, L. Peng, and D. R. Webster. “Using a Deep Learning Algorithm and Integrated Gradients Explanation to Assist Grading for Diabetic Retinopathy”. *Ophthalmology* 126.4 (2019), pp. 552–564.

[106] Z. L. Teo, Y. C. Tham, M. Yu, M. L. Chee, T. H. Rim, N. Cheung, M. M. Bikbov, Y. X. Wang, Y. Tang, Y. Lu, I. Y. Wong, D. S. W. Ting, G. S. W. Tan, J. B. Jonas, C. Sabanayagam, T. Y. Wong, and C. Y. Cheng. "Global Prevalence of Diabetic Retinopathy and Projection of Burden through 2045: Systematic Review and Meta-Analysis". *Ophthalmology* 128.11 (2021), pp. 1580–1591.

[107] N. Panwar, P. Huang, J. Lee, P. A. Keane, T. S. Chuan, A. Richhariya, S. Teoh, T. H. Lim, and R. Agrawal. "Fundus Photography in the 21st Century -a Review of Recent Technological Advances and Their Implications for Worldwide Healthcare". *Telemedicine and e-Health* 22.3 (2016), pp. 198–208.

[108] L. Lin, M. Li, Y. Huang, P. Cheng, H. Xia, K. Wang, J. Yuan, and X. Tang. "The SUSTech-SYSU Dataset for Automated Exudate Detection and Diabetic Retinopathy Grading". *Scientific Data* 7.1 (2020), p. 409.

[109] S. D. Solomon and M. F. Goldberg. "ETDRS Grading of Diabetic Retinopathy: Still the Gold Standard?" *Ophthalmic Research* 62.4 (2019), pp. 190–195.

[110] Asia Pacific Tele-Ophthalmology Society. *APTOS 2019 Blindness Detection*. 2019. URL: https://www.kaggle.com/competitions/aptos2019-blindness-detection.

[111] *Diabetic Retinopathy Detection, Kaggle*. URL: https://www.kaggle.com/competitions/diabetic-retinopathy-detection/overview.

[112] *Determining Whether a Dataset Is Imbalanced or Not*. URL: https://datascience.stackexchange.com/questions/122571/determining-whether-a-dataset-is-imbalanced-or-not.

[113] R. Chakrabarti, c. A. Harper, and J. E. Keeffe. "Diabetic Retinopathy Management Guidelines". *Expert Review of Ophthalmology* 7.5 (2012), pp. 417–439.

[114] M. Abadi et al. "TensorFlow: A System for Large-Scale Machine Learning". *USENIX Conference on Operating Systems Design and Implementation*. 2016, pp. 265–283.

[115] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the Inception Architecture for Computer Vision". *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 2016-Decem. 2016, pp. 2818–2826.

[116] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. "Densely Connected Convolutional Networks". *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. Vol. 2017-Janua. 2017, pp. 2261–2269.

[117] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. *ImageNet Large Scale Visual Recognition Challenge*. 2015. URL: https://www.image-net.org/challenges/LSVRC/.

[118] *Intel Core I7-9750H Processor*. URL: https://ark.intel.com/content/www/us/en/ark/products/191045/intel-core-i7-9750h-processor-12m-cache-up-to-4-50-ghz.html.

[119] *GTX 1650 Gaming Laptops*. URL: https://www.nvidia.com/en-eu/geforce/gaming-laptops/gtx-1650/.

[120] G. Coral. *Dev Board*. 2019. URL: https://coral.ai/products/dev-board.

[121] A. Manuskin. *The Stress Terminal UI: S-Tui*. URL: https://github.com/amanusk/s-tui.

[122] *System Management Interface Tool (SMIT)*. 2000. URL: https://developer.nvidia.com/nvidia-system-management-interface.

[123] Google. *Coral Dev Board*. 2020. URL: https://coral.ai/products/dev-board.

[124] *Optimize TensorFlow Performance Using the Profiler*. URL: https://www.tensorflow.org/guide/profiler.

[125] Python Software Foundation. *Datetime — Basic Date and Time Types*. 2020. URL: https://docs.python.org/3/library/datetime.html.

[126] *Messidor*. URL: https://www.adcis.net/en/third-party/messidor/.

[127] E. Decencière, X. Zhang, G. Cazuguel, B. Laÿ, B. Cochener, C. Trone, P. Gain, J. R. Ordóñez-Varela, P. Massin, A. Erginay, B. Charton, and J. C. Klein. "Feedback on a Publicly Distributed Image Database: The Messidor Database". *Image Analysis and Stereology* 33.3 (2014), pp. 231–234.

[128] M. D. Abràmoff, J. C. Folk, D. P. Han, J. D. Walker, D. F. Williams, S. R. Russell, P. Massin, B. Cochener, P. Gain, L. Tang, M. Lamard, D. C. Moga, G. Quellec, and M. Niemeijer. "Automated Analysis of Retinal Images for Detection of Referable Diabetic Retinopathy". *JAMA Ophthalmology* 131.3 (2013), pp. 351–357.

[129] *MESSIDOR-2 DR Grades*. URL: https://www.kaggle.com/datasets/google-brain/messidor2-dr-grades.

[130] J. Krause, V. Gulshan, E. Rahimy, P. Karth, K. Widner, G. S. Corrado, L. Peng, and D. R. Webster. "Grader Variability and the Importance of Reference Standards for Evaluating Machine Learning Models for Diabetic Retinopathy". *Ophthalmology* 125.8 (2018), pp. 1264–1272.

[131] J. P. Blair, J. N. Rodriguez, R. M. Lasagni Vitar, M. A. Stadelmann, R. Abreu-González, J. Donate, C. Ciller, S. Apostolopoulos, C. Bermudez, and S. De Zanet. "Development of LuxIA, a Cloud-Based AI Diabetic Retinopathy Screening Tool Using a Single Color Fundus Image". *Translational Vision Science and Technology* 12.11 (2023), p. 38.

[132] S. R. Nandakumar, M. Le Gallo, C. Piveteau, V. Joshi, G. Mariani, I. Boybat, G. Karunaratne, R. Khaddam-Aljameh, U. Egger, A. Petropoulos, T. Antonakopoulos, B. Rajendran, A. Sebastian, and E. Eleftheriou. "Mixed-Precision Deep Learning Based on Computational Memory". *Frontiers in Neuroscience* 14 (2020).

[133] C. Li, D. Belkin, Y. Li, P. Yan, M. Hu, N. Ge, H. Jiang, E. Montgomery, P. Lin, Z. Wang, W. Song, J. P. Strachan, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, and Q. Xia. "Efficient and Self-Adaptive in-Situ Learning in Multilayer Memristor Neural Networks". *Nature Communications* 9.1 (2018).

[134] G. Charan, A. Mohanty, X. Du, G. Krishnan, R. V. Joshi, and Y. Cao. "Accurate Inference with Inaccurate RRAM Devices: A Joint Algorithm-Design Solution". *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* 6.1 (2020), pp. 27–35.

[135] W. Jiang, Q. Lou, Z. Yan, L. Yang, J. Hu, X. S. Hu, and Y. Shi. “Device-Circuit-Architecture Co-Exploration for Computing-in-Memory Neural Accelerators”. *IEEE Transactions on Computers* 70.4 (2021), pp. 595–605.

[136] Z. Song, Y. Sun, L. Chen, T. Li, N. Jing, X. Liang, and L. Jiang. “ITT-RNA: Imperfection Tolerable Training for RRAM-Crossbar-Based Deep Neural-Network Accelerator”. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40.1 (2021), pp. 129–142.

[137] A. Antolini, C. Paolino, F. Zavalloni, A. Lico, E. F. Scarselli, M. Mangia, F. Pareschi, G. Setti, R. Rovatti, M. L. Torres, M. Carissimi, and M. Pasotti. “Combined HW/SW Drift and Variability Mitigation for PCM-Based Analog in-Memory Computing for Neural Network Applications”. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 13.1 (2023), pp. 395–407.

[138] J. Doevenspeck, R. Degraeve, A. Fantini, S. Cosemans, A. Mallik, P. Debacker, D. Verkest, R. Lauwereins, and W. Dehaene. “OxRRAM-Based Analog in-Memory Computing for Deep Neural Network Inference: A Conductance Variability Study”. *IEEE Transactions on Electron Devices* 68.5 (2021), pp. 2301–2305.

[139] M. Fritscher, J. Knödtel, M. Mallah, S. Pechmann, E. P. B. Quesada, T. Rizzi, C. Wenger, and M. Reichenbach. “Mitigating the Effects of RRAM Process Variation on the Accuracy of Artificial Neural Networks”. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).* Vol. 13227 LNCS. 2022, pp. 401–417.

[140] C. Huang, N. Xu, J. Wang, and L. Fang. “An Efficient Variation-Tolerant Method for RRAM-based Neural Network”. *2022 IEEE 5th International Conference on Electronics Technology, ICET 2022.* 2022, pp. 90–95.

[141] V. Joshi, M. Le Gallo, S. Haefeli, I. Boybat, S. R. Nandakumar, C. Piveteau, M. Dazzi, B. Rajendran, A. Sebastian, and E. Eleftheriou. “Accurate Deep Neural Network Inference Using Computational Phase-Change Memory”. *Nature Communications* 11.1 (2020).

[142] Q. Wang, Y. Park, and W. D. Lu. “Device Variation Effects on Neural Network Inference Accuracy in Analog In-Memory Computing Systems”. *Advanced Intelligent Systems* 4.8 (2022).

[143] V. Milo, F. Anzalone, C. Zambelli, E. Perez, M. K. Mahadevaiah, O. G. Ossorio, P. Olivo, C. Wenger, and D. Ielmini. “Optimized Programming Algorithms for Multilevel RRAM in Hardware Neural Networks”. *IEEE International Reliability Physics Symposium Proceedings.* Vol. 2021-March. 2021.

[144] N. Lepri, A. Glukhov, and D. Ielmini. “Mitigating Read-Program Variation and IR Drop by Circuit Architecture in RRAM-based Neural Network Accelerators”. *IEEE International Reliability Physics Symposium Proceedings.* Vol. 2022-March. 2022, pp. 3C21–3C26.

[145] J. He, Y. Huang, M. Lastras, T. T. Ye, C. Y. Tsui, and K. T. Cheng. “RVComp: Analog Variation Compensation for RRAM-Based in-Memory Computing”. *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC.* 2023, pp. 246–251.

[146] C. C. Chang, S. T. Li, T. L. Pan, C. M. Tsai, I. T. Wang, T. S. Chang, and T. H. Hou. "Device Quantization Policy in Variation-Aware in-Memory Computing Design". *Scientific Reports* 12.1 (2022).

[147] A. Prakash and H. Hwang. "Multilevel Cell Storage and Resistance Variability in Resistive Random Access Memory". *Nano Devices and Sensors* 1.6 (2016), pp. 49–72.

[148] H. Xiao, K. Rasul, and R. Vollgraf. "Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms". *arXiv* (2017). URL: http://arxiv.org/abs/1708.07747.

[149] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik. "EMNIST: Extending MNIST to Handwritten Letters". *Proceedings of the International Joint Conference on Neural Networks*. Vol. 2017-May. 2017, pp. 2921–2926.

[150] P. Y. Chen, B. Lin, I. T. Wang, T. H. Hou, J. Ye, S. Vrudhula, J. S. Seo, Y. Cao, and S. Yu. "Mitigating Effects of Non-Ideal Synaptic Device Characteristics for on-Chip Learning". *2015 IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2015*. 2016, pp. 194–199.

[151] L. Kull, T. Toifl, M. Schmatz, P. A. Francese, C. Menolfi, M. Brändli, M. Kossel, T. Morf, T. M. Andersen, and Y. Leblebici. "A 3.1 mW 8b 1.2 GS/s Single-Channel Asynchronous SAR ADC with Alternate Comparators for Enhanced Speed in 32 Nm Digital SOI CMOS". *IEEE Journal of Solid-State Circuits* 48.12 (2013), pp. 3049–3058.

[152] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha. "Deep Learning for Classical Japanese Literature". *arXiv* (2018). URL: http://arxiv.org/abs/1812.01718%0Ahttp://dx.doi.org/10.20676/00000341.

[153] *MNIST Variations*. URL: https://sites.google.com/a/lisa.iro.umontreal.ca/public_static_twiki/variations-on-the-mnist-digits.

[154] *The CIFAR-10 Dataset*. URL: https://www.cs.toronto.edu/~kriz/cifar.html.

[155] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. 2015.

[156] Y. Lin, J. Tang, B. Gao, Q. Qin, Q. Zhang, H. Qian, and H. Wu. "A High-Speed and High-Efficiency Diverse Error Margin Write-Verify Scheme for an RRAM-Based Neuromorphic Hardware Accelerator". *IEEE Transactions on Circuits and Systems II: Express Briefs* 70.4 (2023), pp. 1366–1370.

[157] D. Joksas, E. Wang, N. Barmpatsalos, W. H. Ng, A. J. Kenyon, G. A. Constantinides, and A. Mehonic. "Nonideality-Aware Training for Accurate and Robust Low-Power Memristive Neural Networks". *Advanced Science* 9.17 (2022).

[158] W. Ye, G. L. Zhang, B. Li, U. Schlichtmann, C. Zhuo, and X. Yin. "Aging Aware Retraining for Memristor-Based Neuromorphic Computing". *Proceedings - IEEE International Symposium on Circuits and Systems*. Vol. 2022-May. 2022, pp. 3294–3298.

[159] J. H. Yoon, M. Chang, W. S. Khwa, Y. D. Chih, M. F. Chang, and A. Raychowdhury. "29.1 a 40nm 64Kb 56.67TOPS/W Read-Disturb-Tolerant Compute-in-Memory/Digital RRAM Macro with Active-Feedback-Based Read and in-Situ Write Verification". *Digest of Technical Papers - IEEE International Solid-State Circuits Conference.* Vol. 64. 2021, pp. 404–406.

[160] M. A. Yaldagard, S. Diware, R. V. Joshi, S. Hamdioui, and R. Bishnoi. "Read-Disturb Detection Methodology for RRAM-based Computation-in-Memory Architecture". *AICAS 2023 - IEEE International Conference on Artificial Intelligence Circuits and Systems, Proceeding.* 2023, pp. 1–5.

[161] W. Li, X. Sun, H. Jiang, S. Huang, and S. Yu. "A 40nm RRAM Compute-in-Memory Macro Featuring on-Chip Write-Verify and Offset-Cancelling ADC References". *European Solid-State Device Research Conference.* Vol. 2021-Septe. 2021, pp. 79–82.

[162] W. Li, X. Sun, S. Huang, H. Jiang, and S. Yu. "A 40-Nm MLC-RRAM Compute-in-Memory Macro with Sparsity Control, on-Chip Write-Verify, and Temperature-Independent ADC References". *IEEE Journal of Solid-State Circuits* (2022).

[163] B. Yan, J. Yang, Q. Wu, Y. Chen, and H. Li. "A Closed-Loop Design to Enhance Weight Stability of Memristor Based Neural Network Chips". *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD.* Vol. 2017-Novem. 2017, pp. 541–548.

[164] Y. Park, Z. Wang, S. Yoo, and W. D. Lu. "RM-NTT: An RRAM-Based Compute-in-Memory Number Theoretic Transform Accelerator". *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* 8.2 (2022), pp. 93–101.

[165] W. Wan, R. Kubendran, C. Schaefer, S. B. Eryilmaz, W. Zhang, D. Wu, S. Deiss, P. Raina, H. Qian, B. Gao, S. Joshi, H. Wu, H. S. Wong, and G. Cauwenberghs. "A Compute-in-Memory Chip Based on Resistive Random-Access Memory". *Nature* 608.7923 (2022), pp. 504–512.

[166] M. Saberi, R. Lotfi, K. Mafinezhad, and W. A. Serdijn. "Analysis of Power Consumption and Linearity in Capacitive Digital-to-Analog Converters Used in Successive Approximation ADCs". *IEEE Transactions on Circuits and Systems I: Regular Papers* 58.8 (2011), pp. 1736–1748.

[167] Y. Luo, X. Han, Z. Ye, H. Barnaby, J. S. Seo, and S. Yu. "Array-Level Programming of 3-Bit per Cell Resistive Memory and Its Application for Deep Neural Network Inference". *IEEE Transactions on Electron Devices* 67.11 (2020), pp. 4621–4625.

[168] S. Diware, Y. Biyani, A. Gebregiorgis, S. Hamdioui, and R. Bishnoi. "Energy-Efficient Computation-In-Memory Prototype for Hierarchical ECG Classification". *in preparation (to be submitted to Nature).* 2025.

[169] S. Diware, M. A. Yaldagard, and R. Bishnoi. "Hardware-Aware Quantization for Accurate Memristor-Based Neural Networks". *International Conference on Computer-Aided Design (ICCAD).* 2024.

Curriculum Vitæ

Sumit Shaligram Diware

13-06-1992	Born in Akola, India.

Education

October 2024	Ph.D. in Computer Engineering, Technische Universiteit Delft (TU Delft), Delft, The Netherlands	
	Thesis:	Computation-In-Memory based Edge-AI for Health-care: A Cross-Layer Approach
	Promotor:	Prof. Said Hamdioui
	Co-promotor:	Dr. Rajendra Bishnoi
July 2018	M.Tech. in VLSI Design Tools and Technology, Indian Institute of Technology (I.I.T.) Delhi, India	
	Thesis:	Early Adaptive Verification Flow for System-on-Chip Power Management Architecture
	Supervisor:	Prof. Kolin Paul

Main Publications

10. **S. Diware**, Y. Biyani, A. Gebregiorgis, S. Hamdioui and R. Bishnoi, *"Energy-Efficient Computation-In-Memory Prototype for Hierarchical ECG Classification"*, in preparation (to be submitted to Nature), 2025.

9. **S. Diware**, K. Chilakala and R. Bishnoi, *"Computation-In-Memory for Reliable and Energy-Efficient Diabetic Retinopathy Screening"*, Springer book chapter (under publishing logistics), 2025.

8. **S. Diware**, Y. Dong, M. A. Yaldagard, and R. Bishnoi, *"Adaptive Multi-Threshold Encoding for Energy-Efficient ECG Classification Architecture using Spiking Neural Network"*, accepted in Design, Automation and Test in Europe (DATE), 2025.

7. **S. Diware**, M. A. Yaldagard, and R. Bishnoi, *"Hardware-Aware Quantization for Accurate Memristor-Based Neural Networks"*, International Conference on Computer-Aided Design (ICCAD), 2024.

6. **S. Diware**, K. Chilakala, R. V. Joshi, S. Hamdioui and R. Bishnoi, *"Reliable and Energy-Efficient Diabetic Retinopathy Screening using Memristor-based Neural Networks"*, IEEE Access, 2024.

5. **S. Diware**, M. A. Yaldagard, A. Gebregiorgis, R. V. Joshi, S. Hamdioui and R. Bishnoi, *"Dynamic Detection and Mitigation of Read-disturb for Accurate Memristor-based Neural Networks"*, International Conference on Artificial Intelligence Circuits and Systems (AICAS), 2024.

4. **S. Diware**, S. Dash, A. Gebregiorgis, R. V. Joshi, C. Strydis, S. Hamdioui and R. Bishnoi, *"Severity-Based Hierarchical ECG Classification Using Neural Networks"*, Transactions on Biomedical Circuits and Systems (TBioCAS), 2023.

3. **S. Diware**, A. Gebregiorgis, R. V. Joshi, S. Hamdioui and R. Bishnoi, *"Mapping-aware Biased Training for Accurate Memristor-based Neural Networks"*, International Conference on Artificial Intelligence Circuits and Systems (AICAS), 2023.

2. **S. Diware**, A.Singh, A. Gebregiorgis, R. V. Joshi, S. Hamdioui and R. Bishnoi, *"Accurate and Energy-Efficient Bit-Slicing for RRAM-Based Neural Networks"*, Transactions on Emerging Topics in Computational Intelligence (TETCI), 2023

1. **S. Diware**, A. Gebregiorgis, R. V. Joshi, S. Hamdioui and R. Bishnoi, *"Unbalanced Bit-slicing Scheme for Accurate Memristor-based Neural Network Architecture"*, International Conference on Artificial Intelligence Circuits and Systems (AICAS), 2021.

Other Publications

7. A. Sehgal, A. K. Shukla, **S. Diware**, S. Soni, S. Dhull, S. Shreya, S. Roy, and R. Bishnoi, *"Enhancing Parallelism and Energy-Efficiency in SOT-MRAM based CIM Architecture for On-Chip Learning"*, submitted to Design Automation Conference (DAC), 2025.

6. A. Singh, R. Bishnoi, A. Kaichouhi, **S. Diware**, R. V. Joshi, and S. Hamdioui, *"A 115.1 TOPS/W, 12.1 TOPS/mm^2 Computation-in-Memory using Ring-Oscillator based ADC for Edge AI"*, International Conference on Artificial Intelligence Circuits and Systems (AICAS), 2023.

5. M. A. Yaldagard, **S. Diware**, R. V. Joshi, S. Hamdioui and R. Bishnoi, *"Read-disturb Detection Methodology for RRAM-based Computation-in-Memory Architecture"*, International Conference on Artificial Intelligence Circuits and Systems (AICAS), 2023.

4. H. Aziza, C. Zambelli, S. Hamdioui, **S. Diware**, R. Bishnoi and A. Gebregiorgis, *"On the Reliability of RRAM-Based Neural Networks"*, International Conference on Very Large Scale Integration (VLSI-SoC), 2023.

3. R. Bishnoi, **S. Diware**, A. Gebregiorgis, S. Thomann, S. Mannaa, B. Deveautour, C. Marchand, A. Bosio, D Deleruyelle, I. O'Connor, H. Amrouch and S. Hamdioui, *"Energy-efficient Computation-In-Memory Architecture using Emerging Technologies"*, International Conference on Microelectronics (ICM), 2023.

2. A. Gebregiorgis, A.Singh, **S. Diware**, R. Bishnoi and S. Hamdioui, *"Dealing with Non-Idealities in Memristor Based Computation-In-Memory Designs"*, International Conference on Very Large Scale Integration (VLSI-SoC), 2022

1. A. Singh, **S. Diware**, A. Gebregiorgis, R. Bishnoi, F. Catthoor, and S. Hamdioui, *"Low-Power Memristor-Based Computing for Edge-AI Applications"*, International Symposium on Circuits and Systems (ISCAS), 2021.