POLITECNICO DI TORINO

Collegio di Ingegneria Chimica e dei Materiali

Master of Science Course

in Materials Engineering for Industry 4.0



Master of Science Thesis

Algorithm–Hardware Co-Design of Silicon Nanowire Devices for Neuromorphic Computing Applications

Supervisors :

Prof. Abdollah Saboori

Prof. Sandro Carrara (EPFL)

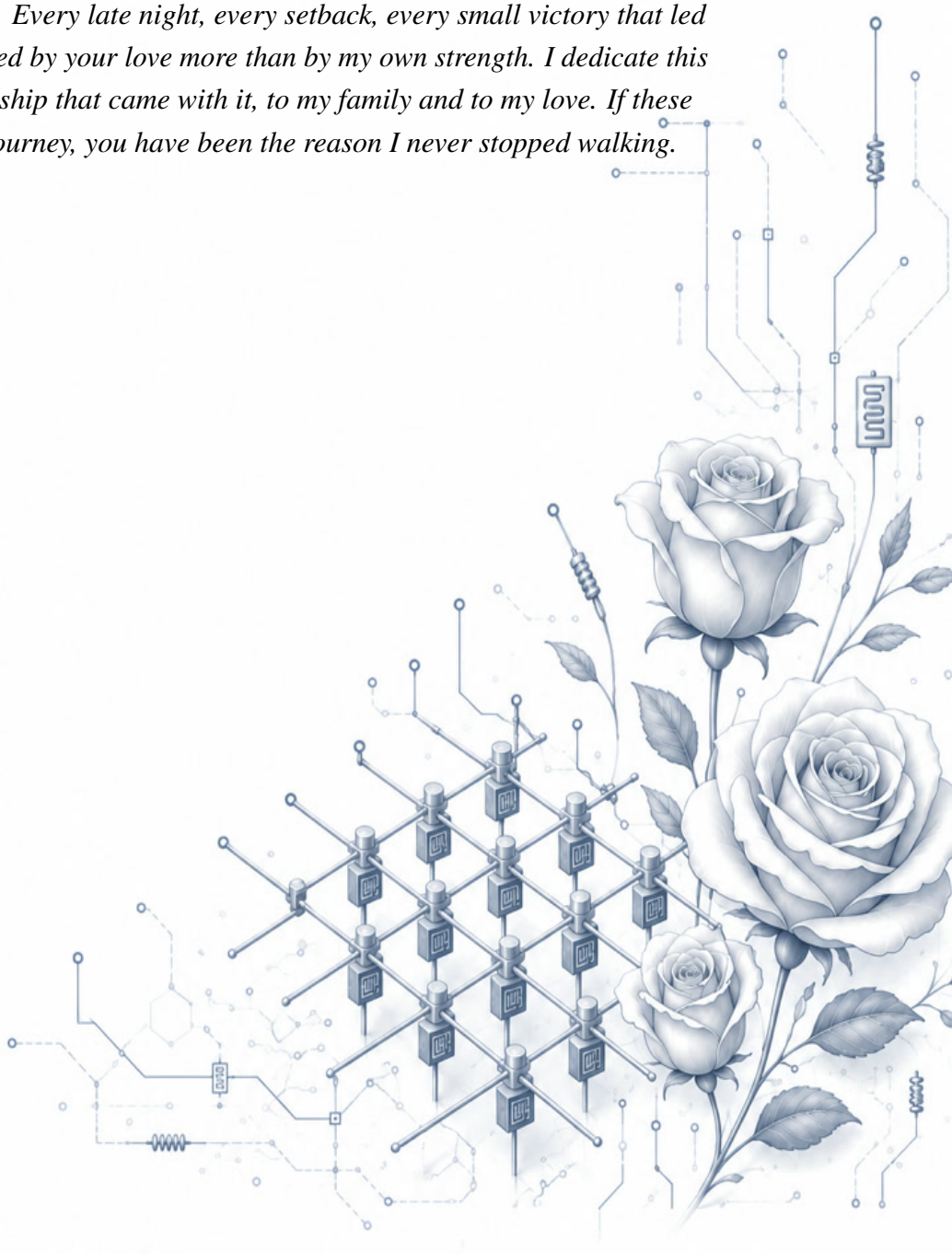
Dr. Junrui Chen (EPFL)

Candidate:

Ali Pilehvar Meibody

July 2026

I left home believing I was searching for a future. Instead, I discovered the value of my past. It was only after years spent far away that I understood the magnitude of what I had always had: the warmth of a family whose love asked for nothing in return, and a heart that remained beside me through every distance, every doubt, and every difficult day. Every late night, every setback, every small victory that led to this thesis was carried by your love more than by my own strength. I dedicate this work, and every hardship that came with it, to my family and to my love. If these pages represent a journey, you have been the reason I never stopped walking.



Acknowledgements

I would like to express my sincere gratitude to **Politecnico di Torino** and **École Polytechnique Fédérale de Lausanne (EPFL)** for providing the academic environment, resources, and opportunities that made this Master's journey possible. The collaboration between these two institutions was an invaluable experience that significantly contributed to both my academic and personal development.

I would like to express my sincere gratitude to **Professor Abdollah Saboori** for his support and guidance throughout my Master's studies. His advice and encouragement helped me overcome many challenges during this journey. I am especially thankful for the trust he showed in me and for being available whenever I needed help. I also appreciate his mentorship and the support he provided during my time at Politecnico di Torino.

I would also like to express my sincere gratitude to **Professor Sandro Carrara** for welcoming me into his research group at EPFL and providing me with the opportunity to work in an inspiring and highly interdisciplinary research environment. His vision and leadership allowed me to begin this research journey and explore the fascinating intersection of nanotechnology, biosensing, and neuromorphic computing.

I would like to give special thanks to **Junrui Chen**, a Ph.D. student in Professor Carrara's group, for all the support he provided during this thesis work. He was always available to share his knowledge and offer useful suggestions whenever I faced difficulties. Our discussions on device physics, neuromorphic computing, experimental observations, and research ideas helped me better understand many aspects of this project. I am particularly thankful for his expertise in silicon nanowire devices and fabrication techniques. The fabrication process used in this work benefited greatly from his previous research and experience. His guidance and continuous support were very important for the completion of this thesis.

I would also like to thank **Dr. Mohammad Bejat** and **Mohammad Taghian Todeshki**, Ph.D. researchers in the Integrated Additive Manufacturing (IAM) group, for their support and helpful discussions during my Master's studies. Their guidance and collaboration were valuable in several research activities and publications that I was involved in during this period.

I am also grateful to the staff of the **Center of MicroNanoTechnology (CMi)** at EPFL for their training and technical support. Their assistance in the use of the nanofabrication facilities was essential for carrying out the fabrication processes required for this work.

Finally, I would like to thank all my colleagues, collaborators, friends, and family for their support throughout this journey. Even though I cannot mention everyone individually, I truly appreciate all those who helped, encouraged, and supported me during my Master's studies. Their contributions, in different ways, made this work possible.

Abstract

In this thesis, silicon nanowire memristive devices are investigated as a multifunctional platform for neuromorphic computing and neuromorphic biosensing. Using CMOS-compatible fabrication and oxide engineering, silicon nanowire devices with stable resistive switching behavior were developed and experimentally characterized. The results show that oxide engineering allows a single silicon nanowire platform to support different neuromorphic functions, such as synaptic weight storage, nonlinear activation, pulse generation, and neuron-like behavior.

To evaluate their suitability for neuromorphic hardware, the devices were integrated into a hardware-aware simulation framework using CrossSim. Artificial neural networks and ternary neural networks were mapped onto memristive crossbar architectures using experimentally measured device characteristics. The optimized TNN achieved 93.78% inference accuracy while reducing hardware complexity by about eight times compared to a full-precision ANN. Hardware-aware simulations showed that programming variability caused only a small drop in accuracy, while ADC/DAC quantization reduced the accuracy to 81.14%. By including measured device non-idealities during training, device-aware training recovered the accuracy to 93.68%. These results show that performance losses caused by hardware effects can be largely reduced through co-optimization between the algorithm and the hardware.

Beyond computing applications, silicon nanowire memristive biosensors were investigated as pulse-generating neuromorphic sensing elements. A spiking neural network was developed to process pulse-encoded biomarker information for prostate cancer risk classification, achieving 92% classification accuracy. The presented results establish silicon nanowire memristors as promising candidates for future integrated neuromorphic systems that combine sensing, memory, and computation within a unified hardware platform.

Contents

List of Figures **xii**

List of Tables **xxvi**

List of Abbreviations **xxvii**

1 Introduction **1**

1.1 The Rise of Artificial Intelligence 1

1.1.1 The Von Neumann Bottleneck and In-Memory Computing . 4

1.1.2 Neuromorphic Computing 6

1.1.3 Emerging Neuromorphic Devices 8

1.1.4 Hardware–Algorithm Co-Design in Neuromorphic Computing 11

1.2 Silicon Nanowire Devices as Multifunctional Neuromorphic Elements 12

1.2.1 Motivation and Research Vision 12

1.2.2 Research Objectives 14

1.2.3 Thesis Organization 15

2 Silicon Nanowire Memristive Devices: Device Architecture and Fabrication **16**

2.1 Device Architecture and Physical Foundations of Silicon Nanowire Memristive Devices 16

2.2 Device Architecture of the Suspended Silicon Nanowire Memristor . 18

2.2.1 Suspended Silicon Nanowire Geometry 18
2.2.2 Silicon-on-Insulator Platform 19
2.2.3 Nickel Silicide Contact Formation 19
2.2.4 Schottky Barrier Formation 20
2.2.5 Back-to-Back Schottky Diode Architecture 21
2.2.6 Surface Oxide Engineering 23
2.2.7 Physical Origin of Volatile Memristive Behavior 24
2.3 Fabrication Technologies for Silicon Nanowire Devices 25
2.3.1 Top-Down and Bottom-Up Nanofabrication Approaches . . 25
2.3.2 Lithography as the Foundation of Nanofabrication 26
2.3.3 Positive and Negative Resist Systems 27
2.3.4 Electron-Beam Lithography 28
2.3.5 Thin-Film Deposition Technologies 29
2.3.6 Lift-Off Pattern Transfer 29
2.3.7 Silicidation Technology 30
2.3.8 Deep Reactive Ion Etching (DRIE) and the Bosch Process . 30
2.3.9 Surface Chemistry and Oxide Engineering 32
2.4 Fabrication Process of the Suspended Silicon Nanowire Memristive Device 32
2.4.1 Starting Material Selection 34
2.4.2 Alignment Marker Fabrication 34
2.4.3 Definition of Nickel Contact Regions 35
2.4.4 Nickel Deposition and Lift-Off 35
2.4.5 Formation of NiSi Schottky Contacts 36
2.4.6 High-Resolution Nanowire Definition 36
2.4.7 DRIE and Nanowire Release 37
2.4.8 Removal of HSQ and Surface Preparation 37

2.4.9 Final Electrode Fabrication 37
2.4.10 Final Device Structure 38

3 Structural and Electrical Characterization: Oxide Engineering 39

3.1 Structural Characterization 39
3.1.1 Scanning Electron Microscopy (SEM) Characterization . . . 39
3.1.2 Atomic Force Microscopy (AFM) Characterization 41
3.1.3 FIB-SEM Characterization of the Suspended Silicon Nanowire 43
3.1.4 Scanning Transmission Electron Microscopy (STEM) and High-Angle Annular Dark Field (HAADF)-STEM Structural Characterization 44
3.1.5 STEM-EDS Elemental Mapping and Line-Profile Analysis . 46
3.2 Electrical Characterization and Oxide Engineering 48
3.2.1 Schottky-Limited Transport in the As-Fabricated Device . . 48
3.2.2 Native Oxidation and Emergence of Memristive Hysteresis . 49
3.2.3 Oxygen Plasma Treatment as an Oxide Engineering Method 50
3.2.4 Time-Dependent Conductance Modulation 51
3.2.5 Oxide-Controlled Switching Regimes 52
3.2.6 From Device Physics to Neuromorphic Applications 54

4 Hardware-Aware Modeling of Analog In-Memory Neural Networks Using CrossSim 57

4.1 Analog In-Memory Computing (AIMC) Fundamentals 57
4.2 CrossSim Simulation Framework 63
4.2.1 Weight Resolution and Quantization 66
4.2.2 Bias Weight Implementation 67
4.2.3 Negative Number Representation 68
4.2.4 Weight Bit Slicing 69

4.3 Device Parameters 70
4.3.1 Modeling Device-Level Non-Idealities 70
4.3.2 Conductance On/Off Ratio 71
4.3.3 Conductance Programming Errors 72
4.3.4 Conductance Drift 74
4.3.5 Conductance Read Noise 75
4.4 Array Parameters 75
4.4.1 Array Size and Matrix Partitioning 75
4.4.2 Parasitic Resistance 77
4.5 ADC and Activation Quantization 77

5 Silicon Nanowire Devices as Synaptic Elements for Neuromorphic Computing 80

5.1 Motivation for Synaptic Computing 80
5.2 Digital Neural Network Baseline 83
5.3 Ternary Neural Networks (TNNs) 86
5.3.1 Optimization of TNN Backpropagation Surrogates 91
5.3.2 Digital TNN and ANN Comparison 94
5.3.3 CrossSim Implementation of ANN and TNN 100
5.3.4 Hardware Trade-Off Analysis and Selection of ANN versus TNN 106
5.4 Impact of Real Silicon Nanowire Device Characteristics on TNN Performance 111
5.4.1 Experimental Device Modeling 111
5.4.2 System-Level Peripheral Non-Idealities 124
5.4.3 Overall Impact of Non-Idealities 129
5.5 Recovery of Accuracy Degradation in Silicon Nanowire Neuromorphic Circuits 131

5.5.1 Using Silicon Nanowire Devices as Analog Neuromorphic Activation Functions . . . 131
5.5.2 Device-Aware Training for Accuracy Recovery . . . 133

6 Toward Silicon Nanowire Neuromorphic Biosensing Systems 139

6.1 Silicon Nanowire Memristive Biosensors for Neuromorphic Signal Generation . . . 139
6.1.1 Motivation for Pulse-Based Neuromorphic Biosensing . . . 139
6.1.2 Memristive Biosensor Structure and Operating Principle . . . 140
6.1.3 Biomarkers and PSA-Based Sensing . . . 141
6.1.4 Pulse Generation Model . . . 142
6.1.5 Overall Neuromorphic Front-End Architecture . . . 144
6.2 Spiking Neural Network (SNN) Fundamentals . . . 145
6.2.1 Introduction to SNNs . . . 145
6.2.2 Neuron Dynamics and Information Encoding . . . 145
6.2.3 Applications of SNNs . . . 146
6.2.4 Learning Paradigms in SNNs . . . 147
6.2.5 Neuron Model Used in This Work . . . 147
6.3 SNN for Pulse-Based Risk Classification . . . 149
6.3.1 Proposed SNN Architecture . . . 149
6.3.2 Synaptic Filtering and Temporal Processing . . . 149
6.3.3 Network Training and Decision Decoding . . . 151
6.3.4 Dataset and Evaluation Protocol . . . 151
6.3.5 Training Performance and Classification Results . . . 152
6.3.6 Temporal and Layer-Wise Representation Analysis . . . 155
6.4 Chapter Summary . . . 159

7 Conclusions, Limitations, and Future Work 160

7.1 Conclusion 160
7.2 Limitations and Assumptions 162
7.3 Future Perspectives 164

Data Availability Statement **167**

References **170**

Appendix A Detailed Training-Loss Evolution of Surrogate Gradient Configurations **179**

Appendix B Latent Weight Distributions and Ternarization Threshold Effects **183**

Appendix C Alternative ADC/DAC Models **190**

Appendix D Membrane Potentials and Spike Responses **191**

List of Figures

1.1 Growth in computational requirements of frontier AI models [1]. . . 2

1.2 Projected cost of training frontier AI models assuming continued exponential growth in computational requirements [1]. 3

1.3 A traditional structure in which the CPU is separated from the memory (left side) and a memory processing unit (MPU) (right side) for memory computing.[2]. 5

1.4 Historical evolution of neuroscience, psychology, ANNs, and machine learning leading to modern neuromorphic computing systems [3]. 6

1.5 Comparison between biological neural systems and conventional computing architectures. 7

1.6 Overview of main emerging device technologies for neuromorphic computing, including RRAM, CBRAM, PCM, ferroelectric devices, spintronic devices, and synaptic transistors. 8

1.7 Neuromorphic computing ecosystem at the interaction between hardware, software, and application layers. Emerging devices, circuits, crossbar arrays, neuromorphic accelerators, software frameworks, datasets, sensors, and actuators are jointly developed through hardware–software co-design methodologies [4]. 12

2.1 Generic energy band diagram illustrating Schottky barrier formation at a metal–semiconductor interface[5]. 21

2.2 (a) Energy band diagram of a metal-semiconductor-metal structure represented as two opposing Schottky junctions separated by a semiconductor channel. (b) Equivalent electrical circuit consisting of two Schottky diodes connected in series with the nanowire resistance.[6] 22

2.3 Comparison between bottom-up and top-down nanofabrication approaches. 26

2.4 (a) lithographic workflow consisting of resist deposition, exposure, development, and pattern transfer.(b) Comparison between positive and negative resist systems. 27

2.5 Schematic illustration of the Bosch DRIE process. (A) During the passivation step, a fluorocarbon polymer layer is deposited on all exposed surfaces. (B) Directional ion bombardment selectively removes the passivation layer at the trench bottom while preserving sidewall protection. (C) Fluorine radicals generated from SF_6 plasma etch the exposed silicon, resulting in highly anisotropic vertical etching. [7]. 31

2.6 Schematic fabrication sequence of the suspended silicon nanowire device: (A) PMMA coating, (B) EBL patterning and Ni deposition, (C) lift-off, (D) Ni silicidation, (E) HSQ nanowire definition, (F) Bosch DRIE etching, and (G) HSQ removal to obtain the final suspended silicon nanowire structure between NiSi Schottky contacts. 33

3.1 SEM images of the fabricated suspended silicon nanowire device. (a) Overall view of the complete device structure. (b) Magnified view of the central region. (c) High-magnification image of the silicon nanowire. 40

3.2 AFM characterization of the fabricated silicon nanowire device, (a) Height image(b) Corresponding peak-force-error image, (c) Large-area AFM height, (d) Large-area AFM Peak-force-error image. . . 42

3.3 FIB-SEM characterization of the fabricated suspended silicon nanowire device. (a) Cross-sectional view of the device structure obtained after FIB milling. (b) Side-view image of suspended nanowire geometry. (c) High-magnification view of the active nanowire region. 44

3.4 Cross-sectional STEM and HAADF-STEM characterization. (a) STEM image of overall morphology. (b) Higher-magnification STEM image. (c) HAADF-STEM image. 45

3.5 STEM-EDS elemental mapping and compositional profiling of the investigated structure. (a) HAADF-STEM image with nickel elemental mapping. (b,c) Nickel concentration and intensity profiles. (d) Combined oxygen and nickel elemental map. (e,f) Oxygen and nickel concentration and intensity profiles. (g) Composite elemental map showing oxygen, silicon, and nickel distributions. (h,i) Corresponding elemental concentration and intensity profiles. 47

3.6 Current–voltage characteristics of the silicon nanowire device during different stages of native oxidation. (a) As-fabricated device. (b,c) Intermediate oxidation stages. (d) Device after native oxidation [8]. 49

3.7 Effect of oxygen plasma treatment on the electrical response of the silicon nanowire device. (a) As-fabricated device, (b) after oxygen plasma treatment, (c) Repeatable volatile memristive response under repeated voltage sweeps [8]. 51

3.8 Time-dependent conductance response of the silicon nanowire device after oxide engineering [8]. 52

3.9 Oxide-controlled switching regimes in silicon nanowire devices. Light oxidation produces trap-controlled non-volatile switching, intermediate oxidation produces volatile memristive switching, and strong oxidation produces threshold-type volatile switching. 53

3.10 Silicon nanowire as a multifunctional platform for neuromorphic systems. 55

4.1 Conceptual correspondence between mathematical MVM and its implementation in a resistive crossbar array. Neural network weights are mapped to conductances (G_{ij}), input values are encoded as voltages (V_i), and output currents represent the accumulated dot products. 60

4.2 Representative non-idealities in AIMC systems, including conductance variability, device noise, conductance drift, nonlinear device behavior, parasitic interconnect resistance, and imperfections in peripheral circuits such as ADCs and DACs. 62

4.3 Structure of the CrossSim inference framework and interaction between neural network models, hardware configuration parameters, device characterization data, and inference datasets. 64

4.4 Overview of the CrossSim inference framework. A trained neural network, hardware configuration parameters, characterized device data, and inference datasets are combined to simulate analog MVM operations while incorporating realistic device- and circuit-level non-idealities [9]. 66

4.5 Comparison between digital and analog bias implementation. 67

4.6 Representation of signed neural network weights using (a) differential-cell encoding and (b) offset subtraction. 69

4.7 Example of balanced differential encoding with weight bit slicing. An 8-bit signed weight is decomposed into multiple lower-resolution slices, each stored in a separate conductance array. The outputs of individual slices are combined through digital shift-and-add reconstruction [9]. 70

4.8 Stages within the CrossSim inference workflow where device-level non-idealities can be incorporated. Device errors may be introduced during weight-to-conductance mapping, after conductance programming, or dynamically during analog MVM operations 71

4.9 Generic conductance programming error models implemented in CrossSim. (a) State-independent error model. (b) State-proportional error model. 73

4.10 Error characteristics of three exemplary devices provided with CrossSim. (a) SONOS device from Xiao et al. with a saturating exponential fit to the measured points [10], (b) PCM device from Joshi et al. with a quadratic fit [11], (c) ReRAM device from Milo et al. with a linear fit [12]. 74

4.11 Matrix partitioning in CrossSim. A large weight matrix is divided into multiple analog MVM arrays. Each partition computes a partial dot product that is digitized independently and subsequently aggregated to form the complete output. 76

4.12 ADC and activation quantization in CrossSim. (a) Analog signals are converted into discrete digital levels according to the ADC resolution and range, introducing quantization and clipping errors. (b) Quantization is applied both to analog MVM outputs through ADCs and to activation values before they are supplied to subsequent neural network layers. 79

5.1 Overview of the MNIST-based fully connected ANN implemented in PyTorch, showing the dataset, network architecture (784–256–128–10), training workflow, and the resulting digital baseline accuracy used for subsequent hardware-aware evaluations. . 84

5.2 Weight and bias distributions of the trained MNIST ANN. Histograms are shown for all three fully connected layers (fc1, fc2, and fc3). The dashed red line indicates the zero value. 85

5.3 Statistical analysis of the trained MNIST ANN. (a) Mean absolute weight and bias magnitude for each fully connected layer. (b) Standard deviation of the weight distributions. (c) Heatmap visualizations of representative weight matrices. 87

5.4 Comparison of TNN training performance using different surrogate-gradient families. 92

5.5 Detailed comparison of the three selected TNN surrogate-gradient candidates: Scaled STE with $s = 0.7$, Scaled STE with $s = 0.75$, and Smooth Threshold-Window surrogate with $r = 0.7$. 93

5.6 Comparison between the trained ANN and TNN models. (a) Training-loss evolution over 100 epochs. (b) Final test accuracy comparison. (c) Mean absolute weight magnitude for the full-precision ANN. (d) Ternary weight composition percentages. . 95

5.7 Layer-wise weight and bias distributions of the trained TNN. 97

5.8 Visualization of ternary connectivity patterns. (a) Active versus inactive connections in the first hidden layer. (b) Decomposition of a representative neuron into positive $(+1)$, negative (-1), and zero-valued weight locations. 98

5.9 Comparison of ANN and TNN weight matrices. (a) Full-precision ANN weight matrices. (b) TNN weight matrices. 99

5.10 Representative feature detectors learned by the full-precision and TNN. 100

5.11 Implementation strategies considered for mapping neural networks onto binary silicon nanowire memristors. The ANN approach employs 8-bit weight quantization and bit slicing, requiring eight positive and eight negative slices per layer. The TNN approach directly maps ternary weights using only positive and negative differential arrays. 101

5.12 Comparison of test accuracy obtained from the original PyTorch implementations and their corresponding ideal CrossSim implementations. 103

5.13 Confusion matrices for digital and ideal CrossSim implementations. (a) ANN implemented in PyTorch. (b) ANN implemented in ideal CrossSim. (c) TNN implemented in PyTorch. (d) TNN implemented in ideal CrossSim. 104

5.14 Validation of ANN weight mapping between PyTorch and ideal CrossSim. For each layer, the left column shows the original digital weights, the middle column shows the reconstructed CrossSim weights, and the right column shows the element-wise difference. . . 105

5.15 Validation of TNN weight mapping between PyTorch and ideal CrossSim. For each layer, the left column shows the original ternary weights, the middle column shows the reconstructed CrossSim weights, and the right column shows the difference matrix. 106

5.16 Bit-sliced conductance mapping of the third ANN layer. Each row corresponds to one of the eight binary slices used to represent an 8-bit weight. For every slice, separate positive and negative conductance arrays are required. The effective weight representation is obtained through differential subtraction of the positive and negative arrays. . 107

5.17 Balanced conductance mapping of the TNN layers. For each layer, the positive array stores $+1$ weights and the negative array stores -1 weights. The effective ternary weight matrix is obtained by subtracting the negative conductance matrix from the positive conductance matrix. 108

5.18 Hardware trade-off comparison between ANN and TNN implementations. (a) Classification accuracy comparison. (b) Number of crossbar arrays required per layer. (c) Number of memristive devices required for each layer and for the complete network. (d) Estimated peripheral-circuit requirements. 110

5.19 Experimental characterization data used for programming-error extraction and device modeling of the silicon nanowire memristor. (a) Measured HRS and LRS distributions together with the extracted target conductance states. (b) Fifty repeated current–voltage sweeps used to characterize cycle-to-cycle variability. (c) Conductance distributions extracted at $V_{\text{read}} = 1$ V showing the actual programmed states. (d) Relative programming-error distributions. (e) Absolute conductance-error distributions. (f) Conductance error referenced to the target conductance values. These experimentally derived statistics were incorporated directly into the custom APM SiNW SBFET CrossSim device model. 114

5.20 Impact of experimentally measured programming variability on TNN inference. (a) Classification accuracy comparison between the digital TNN baseline, ideal CrossSim implementation (Programming OFF), and CrossSim implementation with experimentally modeled programming variability enabled (Programming ON). (b) Weight-matrix comparison for each network layer showing the ideal programmed weights, the programmed weights after applying experimental programming variability, and the corresponding element-wise difference matrices. 117

5.21 ON-state (LRS) drift characterization. (a) Raw measured current drift for three devices with five repeated measurements each over a period of 100 seconds. (b) Normalized ON-state drift curves. (c) Power-law fits applied to the normalized ON-state measurements. . 119

5.22 OFF-state (HRS) drift characterization. (a) Raw measured current drift for three devices with five repeated measurements each over a period of 100 seconds. (b) Normalized OFF-state drift curves. (c) Double-exponential fits applied to the normalized OFF-state measurements. 121

5.23 Validation of the final CrossSim drift model against experimental measurements. (a) ON-state drift model comparison showing experimental measurements (gray), randomly sampled CrossSim realizations (blue), and the empirical mean model (dark blue). (b) OFF-state drift model comparison showing experimental measurements (gray), randomly sampled CrossSim realizations (brown), and the empirical mean model (dark brown). 122

5.24 Impact of conductance drift on TNN performance. (a) TNN classification accuracy evaluated at different retention times after programming. Relative drift-induced weight changes in (b) Layer 1 , (c) Layer 2 and (d) Layer 3 at different retention times. 124

5.25 ADC/DAC optimization study for the TNN implementation. (a) Classification accuracy obtained using a 4-bit DAC under different ADC resolutions, ADC ranges, and input ranges. (b) Classification accuracy obtained using a 6-bit DAC. (c) Comparison of the best-performing ADC/DAC candidates. (d) Accuracy degradation relative to the programming-error-only baseline. (e) Confusion matrices comparing the digital TNN baseline, ideal CrossSim implementation without quantization, and the optimized ADC/DAC configuration. . 127

5.26 Impact of parasitic wire resistance on TNN performance. (a) Classification accuracy as a function of row/column parasitic resistance.(b) Estimated worst-case IR-drop as a function of parasitic resistance. . 128

5.27 Cumulative impact of device-level and system-level non-idealities on TNN inference accuracy. Starting from the digital TNN baseline, progressively more realistic hardware effects are introduced, including ideal CrossSim mapping, experimentally extracted programming variability, optimized ADC/DAC quantization, and parasitic interconnect resistance. 130

5.28 Conceptual illustration of using a silicon nanowire device as an analog activation function between successive crossbar layers. The nonlinear current-voltage characteristic of the device can directly transform analog crossbar outputs into activated analog signals, potentially reducing the need for intermediate ADC/DAC conversions. 132

5.29 Training evolution of the digital TNN and the device-aware TNN. . 133

5.30 Performance recovery using Device-aware training for APM SINW SBFET TNNs. 134

5.31 a) Classification performance comparison between the digital TNN and the device-aware trained TNN. b)confusion matrix difference of device-aware and digital network, c)Heat map of difference between device-aware and digital network. 135

5.32 Weight distributions of the device-aware trained TNN for all three network layers. The left column shows the latent shadow weights optimized during training, the middle column shows the corresponding ideal ternary states after thresholding, and the right column shows the physical analog weights after mapping to the silicon nanowire hardware model. 138

6.1 Silicon nanowire memristive biosensor and pulse-generation mechanism. (a) Functionalized SiNW biosensor for selective biomarker sensing. (b) Equivalent electrical circuit composed of a volatile memristor and an effective capacitance. (c) Pulse-domain output generated through self-sustained oscillatory dynamics under DC excitation. 141

6.2 Detailed sensing-to-decision workflow of the proposed neuromorphic biosensing system. Biomolecular binding events modulate the electrical dynamics of the silicon nanowire biosensor, resulting in spike generation through volatile memristive switching. The resulting pulse streams are processed by a SNN, which performs temporal integration and classification to generate the final risk assessment. . 144

6.3 Fundamental concepts of SNNs. (a) Illustration of spike-based information processing, including spike-train inputs, temporal integration, membrane potential evolution, threshold crossing, and output spike generation. (b) Comparison between conventional ANNs and SNNs in terms of information representation, computation model, network operation, efficiency, applications, and training challenges. 146

6.4 Architecture of the proposed SNN. Pulse streams generated from fPSA and tPSA measurements are transformed into synaptic currents and processed through multiple hidden layers of LIF neurons for prostate cancer risk classification [13]. 148

6.5 Representative filtered synaptic current signals obtained from pulse-encoded PSA measurements. The blue traces correspond to fPSA inputs and the red traces correspond to tPSA inputs. Variations in pulse frequency result in distinct temporal current patterns that are subsequently processed by the SNN [13]. 150

6.6 Distribution of the synthetic dataset used for system-level evaluation. (a) Distribution of fPSA and tPSA concentrations. (b) Distribution of free-to-total PSA ratios across the four prostate cancer risk categories [13]. 152

6.7 Training and testing performance of the proposed SNN over 210 epochs. (a) Evolution of training and testing loss. (b) Evolution of training and testing accuracy [13]. 153

6.8 Confusion matrices obtained using the proposed SNN for prostate cancer risk classification. (a) Training dataset. (b) Testing dataset [13]. 154

6.9 Representative membrane potentials and spike outputs from neurons in the first and second hidden layers for a high-risk sample [13]. . . 156

6.10 Representative membrane potentials and spike outputs from neurons in the third hidden layer for a high-risk sample [13]. 157

6.11 Output-layer membrane potentials and spike responses for a representative high-risk sample [13]. 158

A.1 Training-loss evolution of the initial surrogate-gradient methods evaluated for TNN training. The figure compares Plain STE, Clipped STE, multiple Scaled STE configurations ($s = 0.3$–0.9), the Bi-Real style triangular surrogate, and the Tanh/IR-Net style surrogate ($\beta = 1$). All methods exhibit stable convergence, while differences in the final loss values and convergence rates provide insight into the influence of surrogate-gradient design on optimization performance. 179

A.2 Training-loss curves for additional surrogate-gradient configurations evaluated in this work, including Tanh/IR-Net style surrogates ($\beta = 2$ and $\beta = 3$), threshold-window surrogates with different window widths, and smooth threshold-window surrogates with widths ranging from 0.3 to 0.9. 180

A.3 Parameter sensitivity analysis of the evaluated surrogate-gradient methods. The figure shows the influence of the scaling factor in Scaled STE, the parameter β in the Tanh/IR-Net surrogate, the window width in the threshold-window surrogate, and the window width in the smooth threshold-window surrogate on the final test accuracy. A direct comparison between hard and smooth threshold-window surrogates is also included. The dashed line indicates the best ANN baseline accuracy of 98.0%. 181

A.4 Latent full-precision weight distributions (top row) and corresponding ternary weight distributions (bottom row) for the three selected surrogate-gradient methods: Scaled STE ($s = 0.7$), Scaled STE ($s = 0.75$), and Smooth Threshold-Window ($r = 0.7$). Results are shown for FC1, FC2, and FC3. The latent weights remain continuous during optimization, whereas the forward computation uses ternary weights obtained through threshold-based quantization. The distributions demonstrate that all three methods produce highly similar weight representations, with only minor differences observed in the final classification layer. 182

B.1 Latent full-precision weight distributions (left column) and corresponding ternary weight distributions (right column) for the three fully connected layers of the MNIST TNN. The latent weights remain continuous during optimization and are subsequently mapped to ternary values $\{-1, 0, +1\}$ through threshold-based quantization. The figure illustrates how the underlying latent distributions influence the final ternary weight allocation in each layer. 183

B.2 Conductance mapping of the first ANN layer (FC1) using the bit-sliced crossbar implementation. For each of the eight weight-bit slices, the positive conductance array, negative conductance array, differential representation (positive minus negative), and the combined crossbar mapping are shown. The figure illustrates the hardware overhead associated with implementing signed 8-bit weights on a binary memristive crossbar, where each bit slice requires separate positive and negative arrays, resulting in sixteen physical crossbar arrays for a single neural-network layer. 184

B.3 Bit-sliced conductance mapping of the second ANN layer (FC2). Each row corresponds to one weight-bit slice and shows the positive array, negative array, differential conductance map, and combined crossbar representation used to implement signed weights on binary memristive hardware. The figure illustrates how the 8-bit signed weights of FC2 are decomposed into individual bit slices and mapped onto separate positive and negative crossbar arrays. 185

B.4 Validation of the proposed APM SiNW SBFET programming-error model against the reference SONOS model. The top row compares the resulting MNIST accuracy degradation and the mean absolute programming error of the positive and negative arrays across all network layers. The middle row shows the correlation between programmed effective weights obtained using the SONOS and APM SiNW SBFET models for FC1, FC2, and FC3. The bottom row presents the distributions of relative programming displacement from the ideal conductance values. The results demonstrate that the custom APM SiNW SBFET implementation reproduces the expected programming-error behavior while remaining fully compatible with the CrossSim device framework. 186

B.5 Drift-induced conductance variations in the first network layer (FC1) at retention times of 10 s, 25 s, 50 s, 75 s, and 100 s. For each time point, the effective conductance matrix (G_{eff}), conductance change relative to the initial state (ΔG), positive-core conductance variation (ΔG_{pos}), negative-core conductance variation (ΔG_{neg}), and the corresponding distribution of conductance changes are shown. The results illustrate the fluctuations in conductance values caused by the drift model over time. 187

B.6 Conductance drift visualization for the second network layer (FC2) at different retention times. The figure shows the effective conductance matrix (G_{eff}), conductance changes relative to the initial state (ΔG), positive-core conductance variations (ΔG_{pos}), negative-core conductance variations (ΔG_{neg}), and the corresponding distributions of conductance changes. The results illustrate the conductance fluctuations introduced by the drift model over time. 188

B.7 Drift-induced conductance variations in the third network layer (FC3) at retention times of 10 s, 25 s, 50 s, 75 s, and 100 s. The figure shows the effective conductance matrix (G_{eff}), conductance changes relative to the initial state (ΔG), positive-core conductance variations (ΔG_{pos}), negative-core conductance variations (ΔG_{neg}), and the corresponding distributions of conductance changes. The results illustrate the conductance fluctuations introduced by the drift model over time. 189

C.1 Comparison of ADC/DAC models for TNN inference using the APM SiNW SBFET device. Top panels show classification accuracy and loss relative to the digital TNN baseline. Realistic SignMagnitude DAC and SAR ADC implementations cause significantly larger accuracy degradation than the generic CrossSim quantizer model. . 190

D.1 Membrane-potential traces and corresponding spike trains for the first 20 neurons of the first hidden layer of the SNN. For each neuron, the left panel shows the membrane potential dynamics and the right panel shows the generated spike activity over the simulation interval. 192

D.2 Membrane-potential traces and corresponding spike trains for neurons 20–39 of the first hidden layer of the SNN. For each neuron, the left panel shows the membrane potential dynamics and the right panel shows the generated spike activity over the simulation interval. 193

D.3 Membrane-potential traces and corresponding spike trains for all 20 neurons in the second hidden layer of the SNN. For each neuron, the left panel shows the membrane potential dynamics and the right panel shows the generated spike activity over the simulation interval. 194

D.4 Membrane-potential traces and corresponding spike trains for all eight neurons in the third hidden layer of the SNN. For each neuron, the left panel shows the membrane potential dynamics and the right panel shows the generated spike activity over the simulation interval. 195

List of Tables

1 List of abbreviations. xxvii

2.1 Comparison of lithography tools used for micro and nanofabrication. 28

5.1 Surrogate-gradient configurations evaluated for TNN training. . . . 91

5.2 Ideal CrossSim configuration parameters used for ANN and TNN implementations. 102

6.1 Free-to-total PSA ratio, corresponding prostate cancer risk categories, and approximate probabilities [14, 15]. 142

6.2 Classification performance of the proposed SNN on the testing dataset [13]. 154

List of Abbreviations

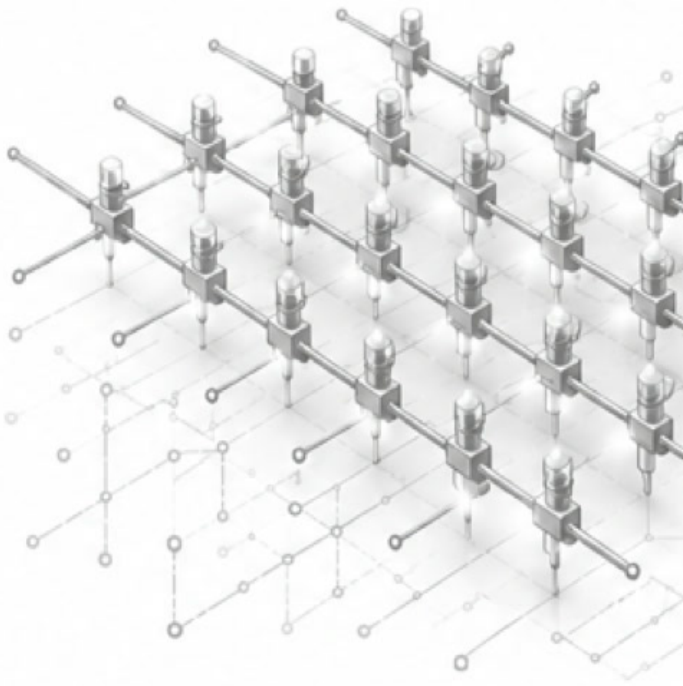
Table 1 List of abbreviations.

Abbreviation	Full Term
ADC	Analog-to-Digital Converter
AFM	Atomic Force Microscopy
AI	Artificial Intelligence
AIMC	Analog In-Memory Computing
ANN	Artificial Neural Network
BPTT	Backpropagation Through Time
CBRAM	Conductive-Bridge Random-Access Memory
CMOS	Complementary Metal-Oxide-Semiconductor
CPU	Central Processing Unit
DAC	Digital-to-Analog Converter
DC	Direct Current
DRIE	Deep Reactive Ion Etching
EBL	Electron-Beam Lithography
EDS	Energy-Dispersive X-ray Spectroscopy
FE-SEM	Field-Emission Scanning Electron Microscopy
FeFET	Ferroelectric Field-Effect Transistor
FIB	Focused Ion Beam
FIB-SEM	Focused Ion Beam–Scanning Electron Microscopy
GPU	Graphics Processing Unit
HAADF	High-Angle Annular Dark Field
HF	Hydrofluoric Acid
HRS	High-Resistance State
HSQ	Hydrogen Silsesquioxane

Abbreviation	Full Term
LIF	Leaky Integrate-and-Fire
LLM	Large Language Model
LRS	Low-Resistance State
MIBK	Methyl Isobutyl Ketone
MMA	Methyl Methacrylate
MRAM	Magnetic Random-Access Memory
MPU	Memory Processing Unit
MTJ	Magnetic Tunnel Junction
MVM	Matrix-Vector Multiplication
NiSi	Nickel Silicide
PCM	Phase-Change Memory
PMMA	Poly(methyl methacrylate)
PSA	Prostate-Specific Antigen
tPSA	Total Prostate-Specific Antigen
fPSA	Free Prostate-Specific Antigen
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
RRAM	Resistive Random-Access Memory
SAR	Successive Approximation Register
SBFET	Schottky Barrier Field-Effect Transistor
SE	Secondary Electron
SEM	Scanning Electron Microscopy
SNN	Spiking Neural Network
SOI	Silicon-on-Insulator
SONOS	Silicon-Oxide-Nitride-Oxide-Silicon
STDP	Spike-Timing-Dependent Plasticity
STE	Straight-Through Estimator
STEM	Scanning Transmission Electron Microscopy
STEM-EDS	Scanning Transmission Electron Microscopy–Energy-Dispersive X-ray Spectroscopy
TMAH	Tetramethylammonium Hydroxide
TNN	Ternary Neural Network
T-SNN	Ternary Spiking Neural Network
TPU	Tensor Processing Unit

Abbreviation	Full Term
VMM	Vector-Matrix Multiplication





Chapter 1

Introduction

1.1 The Rise of Artificial Intelligence

Artificial intelligence (AI) has grown rapidly over the past decade and has significantly impacted fields such as computer vision, natural language processing, healthcare, finance, autonomous systems, and scientific research. This rapid progress has been driven primarily by advances in deep learning, the availability of massive datasets, and increasingly powerful computing infrastructure. Landmark developments such as AlexNet, transformer-based architectures, and large language models (LLMs) have demonstrated that increasing model scale can significantly improve performance across a wide range of tasks [16–18]. As a result, AI systems have achieved human-level or even superhuman performance in numerous benchmark applications.

A major factor behind these advances has been the continuous increase in model size and computational resources. Modern neural networks have evolved from millions of trainable parameters to billions and, more recently, trillions of parameters. This growth has been accompanied by a dramatic increase in the computational requirements needed for training and deployment. Studies have shown that the compute used for state-of-the-art AI models has historically grown much faster than the rate predicted by Moore's law, requiring large-scale clusters of Graphics Processing Units (GPUs), Tensor Processing Units (TPUs), and other specialized AI accelerators [1]. As illustrated in Figure 1.1, the computational demand of frontier

AI systems increased exponentially following the introduction of AlexNet in 2012, marking a significant departure from historical computing trends.

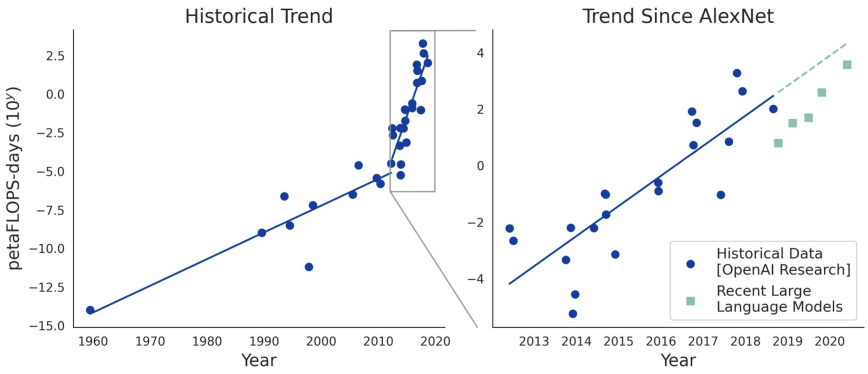


Fig. 1.1 Growth in computational requirements of frontier AI models [1].

While increasing computational resources has enabled substantial improvements in AI performance, this strategy presents important economic and practical limitations. The continued scaling of model size requires increasingly expensive hardware infrastructure, longer training times, and larger energy budgets. Based on past growth trends, the cost of training future frontier models may increase to a level that could become economically unsustainable. As shown in Figure 1.2, if current scaling trends continue, the financial cost associated with training state-of-the-art AI systems could eventually approach national-scale economic benchmarks, indicating that simply increasing computational resources cannot remain a long-term solution for advancing AI [1].

Beyond economic considerations, the rapid growth of AI has raised significant concerns regarding energy consumption and environmental sustainability. Training modern large-scale neural networks requires substantial electrical power, while deployed AI services generate continuous inference workloads that may collectively consume even more energy throughout their operational lifetime. Recent studies have shown that although advances in algorithms and hardware efficiency partially mitigate these effects, the overall energy demand of AI systems continues to increase due to the growing scale and widespread adoption of machine learning technologies [19, 20]. Consequently, reducing the energy cost of computation has become one of the most important challenges facing the future development of AI.

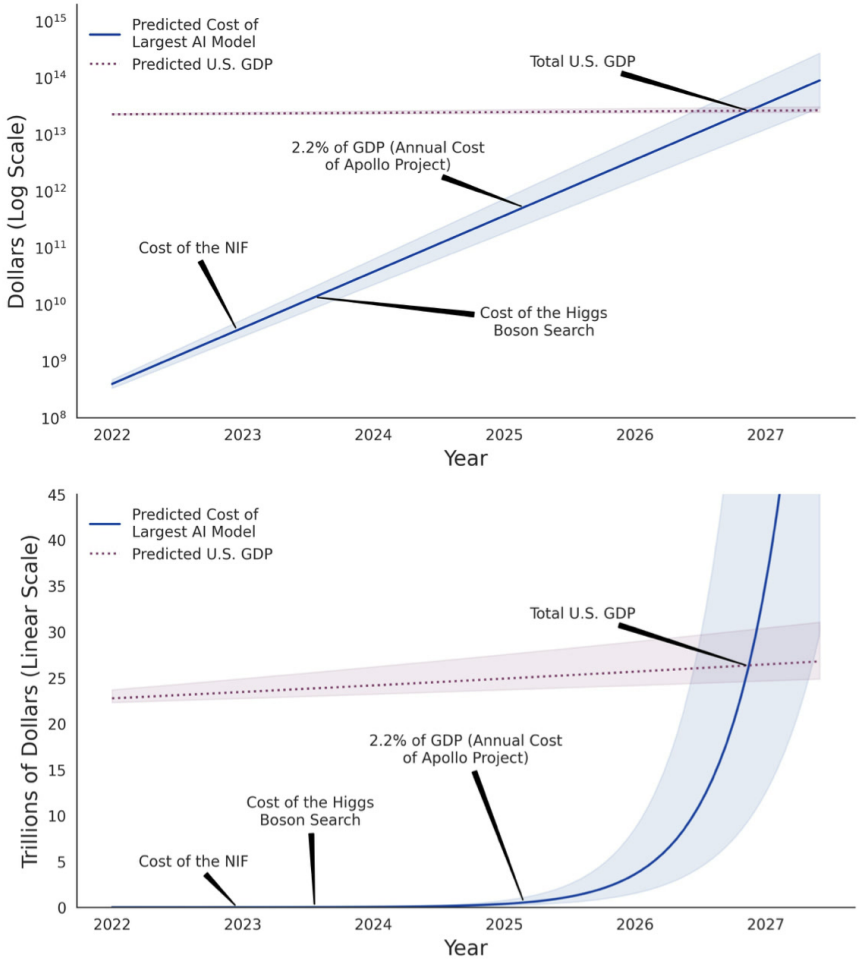


Fig. 1.2 Projected cost of training frontier AI models assuming continued exponential growth in computational requirements [1].

1.1.1 The Von Neumann Bottleneck and In-Memory Computing

The increasing computational demands of modern AI systems have led to a significant rise in energy consumption. For several decades, improvements in computing performance were largely achieved through transistor scaling, as described by Moore's law, which predicted a doubling of transistor density approximately every two years. This scaling enabled faster processors, larger memories, and more powerful computing systems while maintaining manageable costs and energy consumption. However, the rapid growth of AI workloads has exposed limitations that cannot be addressed solely through continued transistor miniaturization or incremental improvements in Central Processing Unit (CPU) and GPU performance.

Modern deep neural networks often contain millions, billions, or even trillions of parameters that must be repeatedly accessed during training and inference. Consequently, the primary challenge is no longer only the speed of computation itself, but also the efficient movement of data between memory and processing units. Despite significant advances in semiconductor technology, a fundamental architectural limitation remains in conventional computing systems. Most modern computers are based on the von Neumann architecture [21], where memory and processing units are physically separated. Data must therefore be continuously transferred between memory and processors during computation.

This separation creates a major performance and energy bottleneck known as the *von Neumann bottleneck* [22]. Every computational operation requires data to be fetched from memory, processed by the computing unit, and often written back to memory. As datasets and neural network models become larger, the amount of data movement increases dramatically. In many modern AI workloads, the energy required to transfer data can exceed the energy required for the computation itself. As a result, data movement has become one of the dominant contributors to power consumption and execution latency in AI systems [23, 24].

Figure 1.3 illustrates the difference between a conventional von Neumann architecture and an in-memory computing architecture [2]. In the traditional approach, memory and processing units are separated, requiring frequent communication through data buses. This repeated transfer of information limits performance, increases latency, and contributes significantly to overall energy consumption. As AI

models continue to grow in complexity and size, the inefficiencies associated with data movement become increasingly severe.

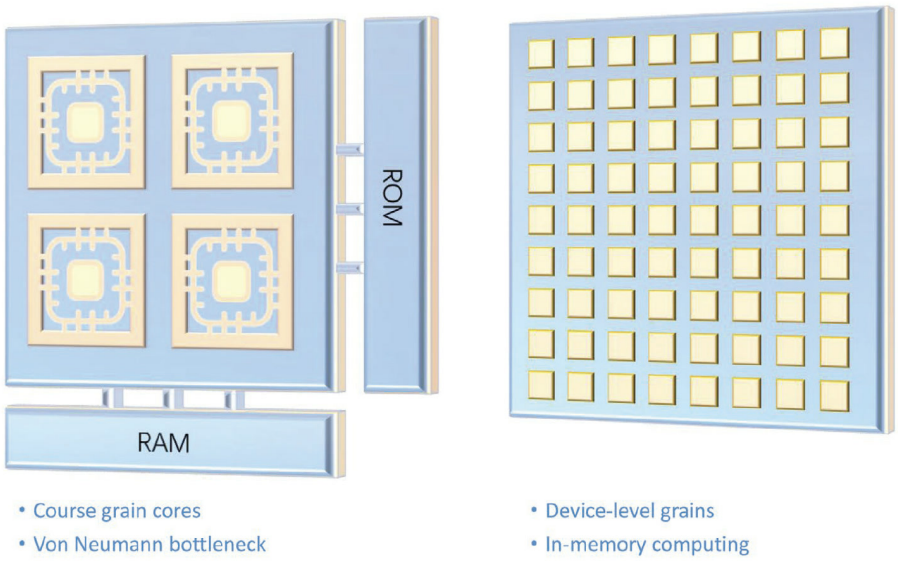


Fig. 1.3 A traditional structure in which the CPU is separated from the memory (left side) and a memory processing unit (MPU) (right side) for memory computing.[2].

To address these limitations, researchers have explored alternative computing paradigms that reduce or eliminate the need for continuous data movement. One of the most promising approaches is *in-memory computing*, in which computational operations are performed directly within memory arrays. By bringing computation closer to where data are stored, in-memory computing significantly reduces communication overhead, lowers energy consumption, and improves computational efficiency [25, 26].

This concept is particularly attractive for AI applications, where matrix–vector multiplication operations dominate the computational workload. Since these operations repeatedly access large amounts of stored data, performing computation directly within memory can substantially improve both performance and energy efficiency compared with conventional computing architectures. Consequently, in-memory computing has emerged as a key enabling technology for future energy-efficient AI hardware and neuromorphic computing systems.

1.1.2 Neuromorphic Computing

While in-memory computing addresses the inefficiencies associated with data movement in conventional computing systems, researchers have also explored more biologically inspired approaches to information processing. One of the most prominent paradigms is neuromorphic computing, which seeks to emulate key computational principles observed in the human brain [27].

The foundations of neuromorphic computing can be traced back to decades of interdisciplinary research spanning neuroscience, psychology, artificial neural networks (ANNs), machine learning, and computational intelligence. As illustrated in Figure 1.4, advances in our understanding of biological learning mechanisms, neural representations, temporal coding, and cognitive processing have continuously influenced the development of neural-inspired algorithms and hardware architectures [3]. From early perceptron models and Hopfield networks to modern deep learning and reservoir computing systems, these developments collectively contributed to the emergence of neuromorphic computing as a distinct research field.

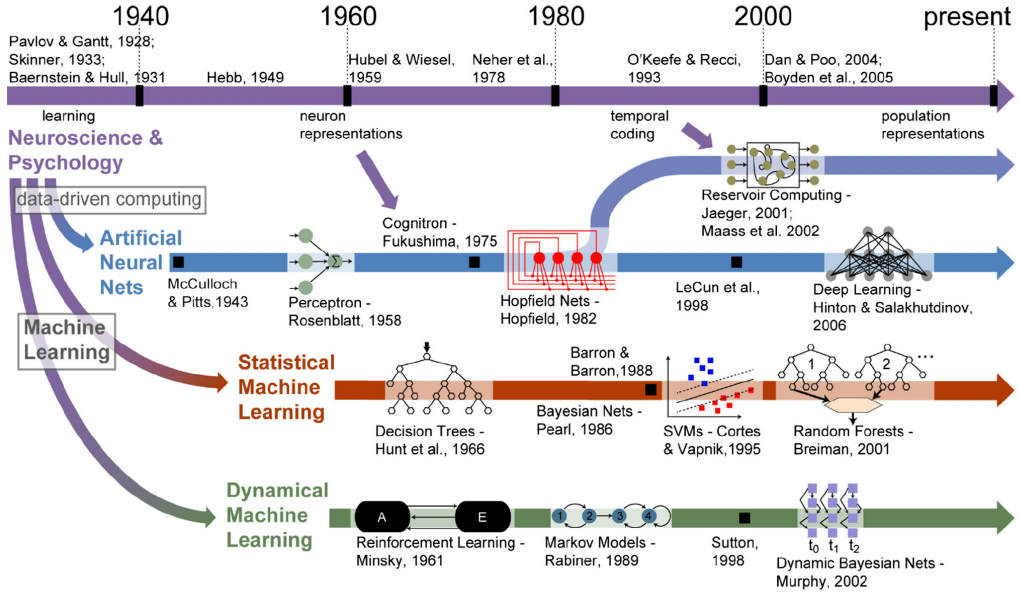


Fig. 1.4 Historical evolution of neuroscience, psychology, ANNs, and machine learning leading to modern neuromorphic computing systems [3].

The human brain is capable of performing complex cognitive tasks while consuming only a small amount of power. Despite containing approximately 10^{11} neurons connected through an estimated 10^{15} synapses, the brain typically operates at a

power level of only around 20 W [28]. In contrast, modern AI systems often require significantly larger amounts of computational resources and energy to perform comparable pattern-recognition tasks, as illustrated in Fig. 1.5.

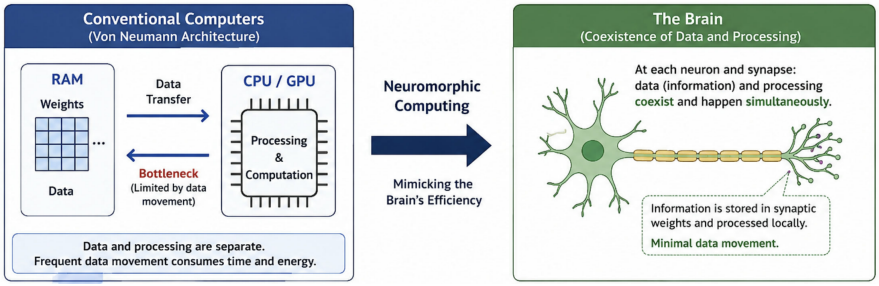


Fig. 1.5 Comparison between biological neural systems and conventional computing architectures.

One of goal of Neuromorphic computing is filling this efficiency gap by designing hardware architectures which is more closely resemble biological neural systems. Instead of separating memory and processing units, biological neurons and synapses naturally can combine information storage and computation inside highly interconnected networks. Information is transmitted through electrical spikes, and computation occurs in a massively parallel and event-driven manner.

Inspired by these characteristics, neuromorphic systems attempt to integrate memory and computation more closely while exploiting parallelism and event-driven operation. Modern neuromorphic platforms typically employ networks of artificial neurons and synapses that communicate through sparse spike events, enabling computation only when information is present. This event-driven behavior significantly reduces unnecessary switching activity and energy consumption compared with conventional clock-driven digital processors [29]. In addition, combining memory and computation in the same system helps overcome some of the limitations of the von Neumann architecture. As a result, neuromorphic systems are considered promising for future AI applications.

Among different technologies of neuromorphic computing, memory devices that are able to store and process information at the same time have attracted a lot of interest. They provide a robust platform for implementing artificial synapses and performing in-memory computation efficiently. In the next section, memristive devices and their role in neuromorphic hardware are introduced.

1.1.3 Emerging Neuromorphic Devices

The realization of practical neuromorphic computing systems requires hardware devices capable of emulating the behavior of biological synapses and neurons while operating with high energy efficiency and dense integration. Conventional Complementary Metal-Oxide-Semiconductor (CMOS) implementations can reproduce neural functions; however, they often require large circuit complexity and significant power consumption. Consequently, extensive research has focused on emerging electronic devices that intrinsically exhibit memory, plasticity, and nonlinear dynamics similar to those observed in biological neural systems [30].

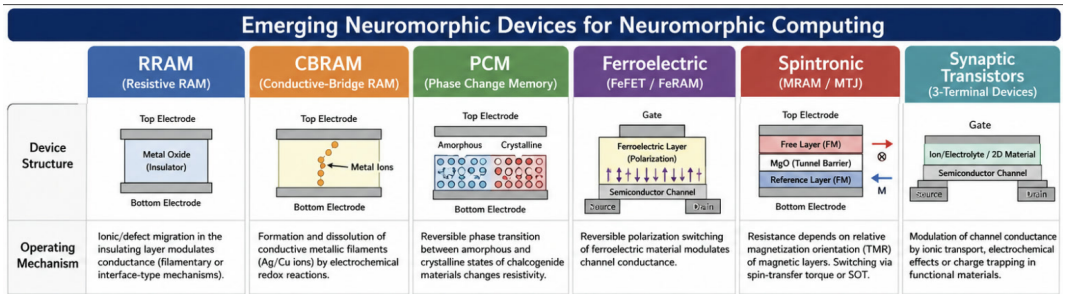


Fig. 1.6 Overview of main emerging device technologies for neuromorphic computing, including RRAM, CBRAM, PCM, ferroelectric devices, spintronic devices, and synaptic transistors.

Among the most widely investigated candidates are resistive random-access memory (RRAM), conductive-bridge random-access memory (CBRAM), phase-change memory (PCM), ferroelectric devices, spintronic memories, and various synaptic transistor technologies. These devices differ in their operating mechanisms, physical structures, and performance characteristics, but all aim to provide compact and energy-efficient implementations of artificial synapses and neurons [31, 32].

Figure 1.6 compares the main classes of neuromorphic devices and illustrate their typical structures and operating mechanisms. Although each technology is designed to reproduce synaptic or neuronal functionality, they are based on different physical processes. These mechanisms include ionic migration and conductive filament formation, phase transitions, ferroelectric polarization switching, spin-dependent effects, and electrochemical modulation of transistor channels.

Resistive Random-Access Memory (RRAM)

RRAM devices, often referred to as memristors, are among the most extensively studied technologies for neuromorphic computing. They typically consist of a metal–insulator–metal structure whose resistance can be changed through the movement of ions or defects within the insulating layer. The resulting conductance states can be programmed continuously, allowing the device to emulate analog synaptic weights. RRAM devices offer excellent scalability, low operating power, compatibility with crossbar architectures, and the ability to perform analog in-memory computation. However, challenges remain regarding device variability, nonlinearity of weight updates, and cycle-to-cycle fluctuations [33, 34].

Conductive-Bridge Random-Access Memory (CBRAM)

CBRAM, also known as electrochemical metallization memory, operates through the formation and dissolution of conductive metallic filaments, typically composed of silver or copper ions. Resistance switching occurs when metal cations migrate through a dielectric layer and form conductive bridges between electrodes. Due to their low programming voltage and low energy consumption, CBRAM devices are attractive candidates for neuromorphic systems [35]. Furthermore, volatile variants of conductive-bridge devices, often referred to as diffusive memristors, can emulate neuron-like behaviors such as leaky integrate-and-fire dynamics and short-term synaptic plasticity [36].

Phase-Change Memory (PCM)

PCM devices rely on reversible transitions between amorphous and crystalline phases of chalcogenide materials. These two phases exhibit significantly different electrical resistivities, allowing information to be stored as resistance states. PCM technology is one of the most mature emerging memory technologies and has demonstrated multi-level conductance control suitable for synaptic weight storage. Additionally, the intrinsic dynamics of phase transitions can be exploited to emulate neuronal behavior. Nevertheless, PCM devices often require relatively high programming currents and may suffer from resistance drift over time [37–39].

Ferroelectric Devices

Ferroelectric memories exploit the reversible polarization of ferroelectric materials. In particular, ferroelectric field-effect transistors (FeFETs) have attracted considerable attention because they provide nonvolatile memory operation while remaining compatible with modern CMOS processes. The gradual polarization

switching of ferroelectric domains enables multiple conductance states that can represent synaptic weights. FeFETs offer fast programming speeds, low power consumption, and highly symmetric potentiation and depression characteristics, making them promising candidates for neuromorphic learning systems [40, 41].

Spintronic Devices

Spintronic memories, such as magnetic random-access memory (MRAM), utilize electron spin rather than charge to store information. Their core element is the magnetic tunnel junction (MTJ), whose resistance depends on the relative orientation of magnetic layers. Spintronic devices offer exceptional endurance, nanosecond-scale switching speeds, and low energy consumption. Furthermore, stochastic switching behavior and domain-wall motion can be exploited to implement neuromorphic functions including probabilistic synapses and artificial neurons. However, their relatively small resistance contrast compared to other memory technologies remains a challenge for large-scale neuromorphic architectures [42–44].

Synaptic Transistors

A separate class of neuromorphic devices consists of three-terminal synaptic transistors. Unlike two-terminal memristive devices, synaptic transistors decouple the programming and read operations, enabling improved control of conductance modulation. Various implementations based on ionic transport, electrochemical effects, and two-dimensional materials have demonstrated both short-term and long-term plasticity, spike-timing-dependent plasticity (STDP), and extremely low energy consumption. Their main challenges involve scalability, fabrication complexity, and integration density compared with two-terminal memory devices [45, 46].

Overall, each emerging neuromorphic technology offers a different balance between scalability, power consumption, switching speed, endurance, and analog programmability. No single device technology currently satisfies all requirements simultaneously. As a result, the search for novel neuromorphic devices remains an active area of research. Among these technologies, memristive devices have attracted particular interest because of their simple structure, compatibility with crossbar architectures, and natural suitability for in-memory computing. The following section therefore focuses on memristive devices and their application in neuromorphic systems.

1.1.4 Hardware–Algorithm Co-Design in Neuromorphic Computing

Unlike conventional computing systems, where hardware and software are usually designed seperately, neuromorphic computing requires hardware and algorithms to be developed together. This concept is known as hardware–algorithm co-design. In this approach, the characteristics of the hardware influence how algorithms are designed, while the requirements of the algorithms help shape the development of devices, circuits, and system architectures.

This co-design approach is particularly important because neuromorphic systems often use emerging devices that do not behave ideally. For example, they may show programming variability, conductance drift, limited precision, stochastic switching, or nonlinear conductance updates. Since these effects can directly impact the performance of neural networks, they must be considered during the design process. As a result, algorithms should be developed while taking the hardware characteristics into account.

Figure 1.7 illustrates the broader neuromorphic computing ecosystem. At the hardware level, neuromorphic systems are constructed from emerging devices such as memristors, RRAM cells, and other memory technologies that are integrated into processing elements, crossbar arrays, accelerators, and complete neuromorphic chips. At the software level, hardware-specific programming layers, simulation frameworks, and high-level development environments are used to implement learning algorithms. These components are further connected to datasets, sensors, actuators, and real-world applications, forming a complete hardware–software ecosystem. This hierarchical structure highlights the need for co-design across all levels of the neuromorphic computing stack [4].

Two complementary co-design approaches are commonly employed. In a top-down approach, researchers start with insights from neuroscience and machine learning, they develop computational models and design hardware architectures which is capable of implementing these algorithms. In contrast, a bottom-up approach starts from the physical properties of devices to develop new learning mechanisms. In practice, successful neuromorphic systems combine both approaches, creating a feedback loop between hardware capabilities and algorithmic innovations.

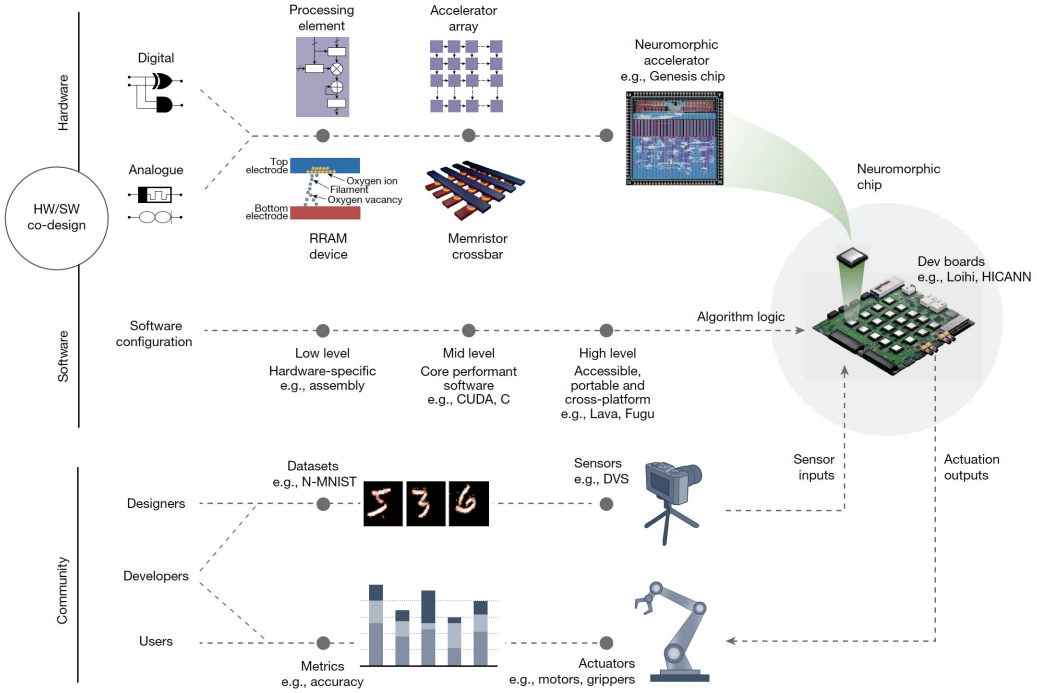


Fig. 1.7 Neuromorphic computing ecosystem at the interaction between hardware, software, and application layers. Emerging devices, circuits, crossbar arrays, neuromorphic accelerators, software frameworks, datasets, sensors, and actuators are jointly developed through hardware–software co-design methodologies [4].

1.2 Silicon Nanowire Devices as Multifunctional Neuromorphic Elements

1.2.1 Motivation and Research Vision

Among the various emerging technologies investigated for neuromorphic computing, silicon nanowire devices represent a particularly attractive platform because they combine sensing, memory, and computation within a single nanoscale structure. Unlike many neuromorphic devices that are designed exclusively for memory or computing applications, silicon nanowires offer the unique possibility of interacting directly with biological environments while simultaneously exhibiting rich electrical behaviors that can be exploited for neuromorphic information processing.

The origins of this research can be traced back to the development of Schottky Barrier Field-Effect Transistor (SBFET) silicon nanowire devices fabricated at EPFL

under the supervision of Professor Sandro Carrara [47]. These devices were initially developed as highly sensitive biosensors for the detection of biomolecules. Experimental measurements revealed the presence of a voltage hysteresis gap in the transfer characteristics of the devices. The magnitude of this hysteresis was found to depend strongly on surface functionalization, biomolecular interactions, and charge trapping phenomena occurring at the nanowire interface. This behavior enabled the detection of DNA molecules, cancer biomarkers, proteins, and other biological targets through purely electrical measurements.

Subsequent investigations revealed that the observed hysteretic behavior shares many characteristics with memristive systems. In particular, both volatile and non-volatile electrical responses were observed under different operating conditions. More recently, detailed studies demonstrated that oxide formation and oxide engineering play a central role in determining the electrical behavior of silicon nanowire devices [8]. The oxide layer influences charge trapping and detrapping mechanisms, thereby affecting memory retention, switching dynamics, volatility, and nonlinear electrical characteristics. Consequently, by controlling the oxidation conditions, it becomes possible to tune the device toward different neuromorphic functionalities.

This multifunctionality forms the central motivation of the present thesis. The non-volatile characteristics of silicon nanowire devices make them attractive candidates for artificial synapses in analog in-memory computing (AIMC) systems. In this role, the device conductance can represent synaptic weights and be integrated into crossbar architectures for ANN accelerators. Furthermore, the intrinsic nonlinear transfer characteristics of silicon nanowires can potentially be exploited as hardware activation functions, reducing the need for external digital processing and enabling more compact neural network implementations.

The volatile behavior of silicon nanowire devices provides an additional neuromorphic functionality. Volatile conductance dynamics can emulate thresholding and temporal integration mechanisms similar to those observed in biological neurons. As a result, silicon nanowires may be utilized as artificial neuron elements within spiking neural networks (SNNs). Moreover, because these devices inherently possess biosensing capabilities, they offer the possibility of directly converting biological signals into spike-based representations. Such a capability opens the door toward monolithic systems in which sensing, spike generation, and neuromorphic computation are integrated within a single hardware platform.

1.2.2 Research Objectives

This thesis investigates the potential of silicon nanowire devices for neuromorphic computing and also develops a hardware-aware framework that can connect experimentally observed device characteristics to system-level neural network performance.

More specifically, the objectives of this work are:

- To investigate the physical and electrical mechanisms responsible for the memristive behavior observed in silicon nanowire devices.
- To study the role of oxide engineering in controlling volatile and non-volatile electrical characteristics and enabling different neuromorphic functionalities.
- To develop accurate hardware-aware models of silicon nanowire devices suitable for neuromorphic system simulations.
- To evaluate the feasibility of employing silicon nanowire devices as artificial synapses in analog in-memory neural network accelerators.
- To investigate the impact of device-level and system-level non-idealities, including programming variability, conductance drift, ADC/DAC quantization, and parasitic effects, on neural network accuracy.
- To explore methods for mitigating accuracy degradation through device-aware training and algorithm–hardware co-design strategies.
- To investigate the integration of silicon nanowire devices within spike-based neuromorphic sensing and computing frameworks.
- To assess the long-term potential of silicon nanowires as multifunctional elements capable of simultaneously supporting sensing, memory, and computation.

To achieve these objectives, this thesis combines experimental device characterization, physical analysis, hardware-aware modeling, analog neural network simulation, and SNN development within a unified research framework.

1.2.3 Thesis Organization

The remainder of this thesis is organized as follows.

Chapter 2 introduces the silicon nanowire device architecture and the operating principles of SBFETs. It starts with a description of fabrication process and then discusses key physical mechanisms.

Chapter 3 presents the structural and electrical characterization of the fabricated devices. The effect of oxide engineering on device performance is examined and show that how different oxidation conditions can result in volatile or non-volatile electrical behavior. These different responses are particularly used for neuromorphic applications.

Chapter 4 introduces the CrossSim hardware-aware simulation framework and explains approach that used to model silicon nanowire devices in AIMC systems. It describes the crossbar architectures and neural network mapping strategies used in this work, as well as the impact of hardware non-idealities including device level and system level.

Chapter 5 investigates the first neuromorphic application of silicon nanowire devices as artificial synapses within neural network accelerators. Due to the two-state nature of the current device implementation, both bit-sliced ANNs and ternary neural networks (TNNs) are studied. The effects of programming variability, conductance drift, ADC/DAC quantization, and parasitic interconnect resistances are analyzed. Finally, device-aware training techniques are employed to recover accuracy losses introduced by hardware constraints.

Chapter 6 describe the integration of sensing and neuromorphic computation through SNNs. A neuromorphic biosensing framework is developed for prostate cancer biomarker detection using synthetic datasets derived. The chapter investigates spike-based processing strategies and discusses the potential of silicon nanowire devices as building blocks for future monolithic sensing-computation platforms.

Finally, Chapter 7 summarizes the principal contributions of this work, discusses current limitations and challenges, and outlines future research directions toward multifunctional silicon nanowire neuromorphic systems capable of combining sensing, memory, and computation within a unified hardware platform.



Chapter 2

Silicon Nanowire Memristive Devices: Device Architecture and Fabrication

2.1 Device Architecture and Physical Foundations of Silicon Nanowire Memristive Devices

The interest in nanowire-based resistive-switching devices arises from the fact that many memristive phenomena become more evident as device dimensions are reduced. In conventional bulk structures, processes such as defect migration, charge trapping, and interface-state modulation are distributed over relatively large volumes and masking their individual contributions. In nanowire devices, however, these phenomena are confined to much smaller active region. As a result, nanowires offer an effective way to study resistive-switching mechanisms and evaluate their potential for next generation memory and neuromorphic systems.

A broad range of nanowire materials have been investigated for resistive-switching applications. Metal-oxide nanowires such as titanium dioxide [48], zinc oxide [49], and vanadium oxide [50] typically exhibit switching behavior associated with oxygen-vacancy migration and conductive-filament formation. Under an external electric field, mobile ionic species redistribute inside the active material, producing localized conductive pathways that connect the electrodes. The creation and rupture of these conductive filaments result in transitions between

high-resistance and low-resistance states. Depending on the stability of the generated filament, the resulting switching behavior may be either volatile or non-volatile [51].

Chalcogenide nanowires represent another important class of resistive-switching materials. In these structures, reversible phase transitions, ion migration, or polarization effects govern the conductance modulation. Their ability to provide continuous and analog conductance updates has made them attractive candidates for artificial synapses and hardware neural networks. Similarly, metallic nanowires based on electrochemical metallization mechanisms exploit the migration and reduction of metal ions to create conductive pathways, allowing the realization of both memory and synaptic functionalities [52].

Although these material systems show excellent switching performance, their operation is based on conductive filament formation or structural changes within the active material. Silicon nanowires have different behavior. Instead of filament formation, the switching behavior is linked to device interfaces. As a result, the mechanisms responsible for resistive switching can be different from those found in conventional memristive materials.

Silicon nanowires possess several characteristics that distinguish them from other nanowire-based devices. First, silicon is the foundation of modern microelectronics and benefits from decades of process optimization and industrial maturity. As a result, silicon nanowire devices can be fabricated using well-established CMOS-compatible technologies, facilitating large-scale integration and reproducibility. Second, silicon exhibits excellent crystalline quality and relatively low intrinsic defect density compared with many metal-oxide materials. This enables more controllable transport characteristics and often improves device-to-device consistency [53].

One of the defining features of silicon nanowires is their large surface-to-volume ratio. Because of their small dimensions, the electrical properties of the device are strongly affected by conditions at the surface and at material interfaces. Consequently, relatively minor surface modifications can lead to measurable changes in device behavior. This characteristic has been widely exploited in biological and chemical sensors, where surface interactions are detected through changes in channel conductance.

Together, these properties make silicon nanowires promising candidates for both sensing and neuromorphic applications. The suspended silicon nanowire structure investigated in this thesis is presented in the next section.

2.2 Device Architecture of the Suspended Silicon Nanowire Memristor

Before examining the electrical behavior of the device, it is useful to first describe its structure. In nanowire-based memristive devices, the observed characteristics depend not only on the semiconductor channel but also on factors such as the device geometry, the metal–semiconductor contacts, and the condition of the nanowire surface. Since all of these features influence the operation of the device, understanding the architecture is important for interpreting its switching behavior as well as its sensing and neuromorphic properties.

The devices investigated in this thesis are based on suspended silicon nanowires fabricated on silicon-on-insulator (SOI) substrates and connected to nickel silicide (NiSi) electrodes. The resulting structure forms a quasi-one-dimensional semiconductor channel positioned between two metal-semiconductor junctions. Unlike conventional oxide-based memristors, where switching occurs primarily within an insulating layer through conductive filament formation, the present device relies on the interaction between carrier transport inside the silicon nanowire, charge dynamics at the interfaces, and the modulation of Schottky barriers located at the contacts.

2.2.1 Suspended Silicon Nanowire Geometry

The active region of the device consists of a suspended silicon nanowire which connected between two NiSi contact pads. The nanowire is fabricated from the top silicon layer of a SOI wafer using a top-down lithographic process. During fabrication, the surrounding silicon is removed, leaving a narrow silicon bridge between the source and drain contacts.

In comparison with conventional planar structures, the suspended geometry is different that how the device interacts with its surroundings. Since the nanowire is physically separated from the underlying substrate, conduction paths are reduced and charge transport is mainly confined to the nanowire. In addition, a larger portion of the nanowire surface is exposed to the environment, making the device more responsive to changes occurring at the surface. The absence of the underlying silicon support also improves electrical isolation and allows external stimuli to interact more directly with the active channel.

As the dimensions of the nanowire are reduced, the influence of the surface becomes increasingly important. A larger fraction of the charge carriers interacts with surface states, interface traps, and local electric fields, making the electrical response highly dependent on the condition of the nanowire surface.

Because of this sensitivity, even small changes at the surface can affect charge that transport inside channel, which is one of the reasons that silicon nanowires are attractive for sensing applications. These can also contribute to charge trapping and relaxation effects that are important in memory-like behavior observed in the device.

2.2.2 Silicon-on-Insulator Platform

The fabrication of the devices is based on SOI technology.The SOI substrate consists of a 625 μm silicon handle wafer, a 1 μm buried silicon dioxide layer, and a 1 μm crystalline silicon device layer.

The buried oxide layer is an important part of the SOI structure. It electrically separates the nanowire from the bulk silicon substrate, it helps to reduce leakage currents and make sure that current flows primarily through the intended conduction path. As a result, the measured electrical response is dominated by the nanowire channel and its contact interfaces rather than by unwanted substrate effects.

The buried oxide layer is also beneficial during fabrication. During deep reactive ion etching, it acts as a reliable etch stop, it allows silicon device layer to be patterned without damaging the underlying substrate.

2.2.3 Nickel Silicide Contact Formation

While the silicon nanowire forms the active semiconductor channel, the electrical characteristics of the device are heavily influenced by the contact regions connecting the nanowire to the external circuitry. For this reason, the design and fabrication of the contacts are of central importance.

In the present work, nickel is initially deposited onto predefined regions of the silicon layer. Following thermal annealing, a silicidation reaction occurs between nickel and silicon, resulting in the formation of NiSi. The silicide regions serve

simultaneously as mechanical supports for the suspended nanowire and as electrical source-drain contacts.

NiSi is widely employed in advanced semiconductor technologies because of its low resistivity, excellent thermal stability, and compatibility with CMOS fabrication processes. More importantly, NiSi forms reproducible metal-semiconductor interfaces with silicon, enabling precise control of carrier injection into the nanowire channel [54].

The silicidation process is very important because the final electrical properties depend on the composition and morphology of the resulting silicide layer. Changes in annealing temperature, annealing duration, and local stoichiometry can change the interface properties and modify the electrical behavior of the device.

As a result, the formation of NiSi is an important part of the device fabrication process, since it strongly affects the transport behavior of the final structure.

2.2.4 Schottky Barrier Formation

When a metal comes into contact with a semiconductor, the difference between their electronic energy levels produces band bending at the interface. Depending on the materials involved, this contact may behave either as an ohmic contact or as a Schottky contact.

As illustrated in Fig. 2.1, a generic metal–semiconductor contact can lead to the formation of a depletion region and an associated Schottky barrier that governs carrier transport across the interface [5]. Although the figure shows a metal/n-type-semiconductor interface, the same physical principle applies to the lightly doped p-type SOI substrates used in this work: the metal–semiconductor interface forms an energy barrier that controls carrier injection.

In the present devices, the NiSi-silicon interfaces behave as Schottky junctions. A Schottky barrier is therefore formed at both ends of the silicon nanowire. These barriers control the injection and extraction of charge carriers between the electrodes and the semiconductor channel.

In contrast to an ideal resistor, the current through a Schottky junction is strongly influenced by the barrier height and also it follows an exponential relationship.

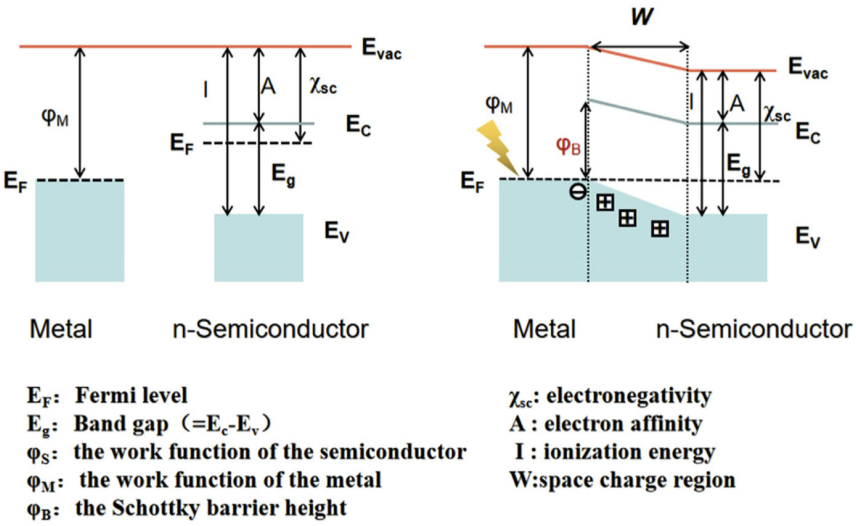


Fig. 2.1 Generic energy band diagram illustrating Schottky barrier formation at a metal–semiconductor interface[5].

Because of this, even small changes in the barrier can have a significant effect on the conductance of the device.

Several physical mechanisms may influence the effective Schottky barrier height, including charge trapping and detrapping, interface-state occupation, oxide charge accumulation, defect redistribution and local electric-field variations. This strong dependency off the barrier properties makes Schottky junctions well suited for conductance modulation and also can show memristive behavior.

2.2.5 Back-to-Back Schottky Diode Architecture

The overall device can be viewed as two Schottky diodes connected in series through a silicon nanowire channel. One Schottky junction is formed at the source side, while the second junction is formed at the drain side.

Figure 2.2 illustrates the back-to-back Schottky diode configuration of the device, where carrier transport is jointly controlled by the two metal-semiconductor interfaces and the resistance of the silicon nanowire channel [6].

As a result, the structure behaves as a back-to-back Schottky diode system. Under an applied voltage, one junction tends to become forward biased while the opposite junction becomes reverse biased. The electrical response of the entire device is

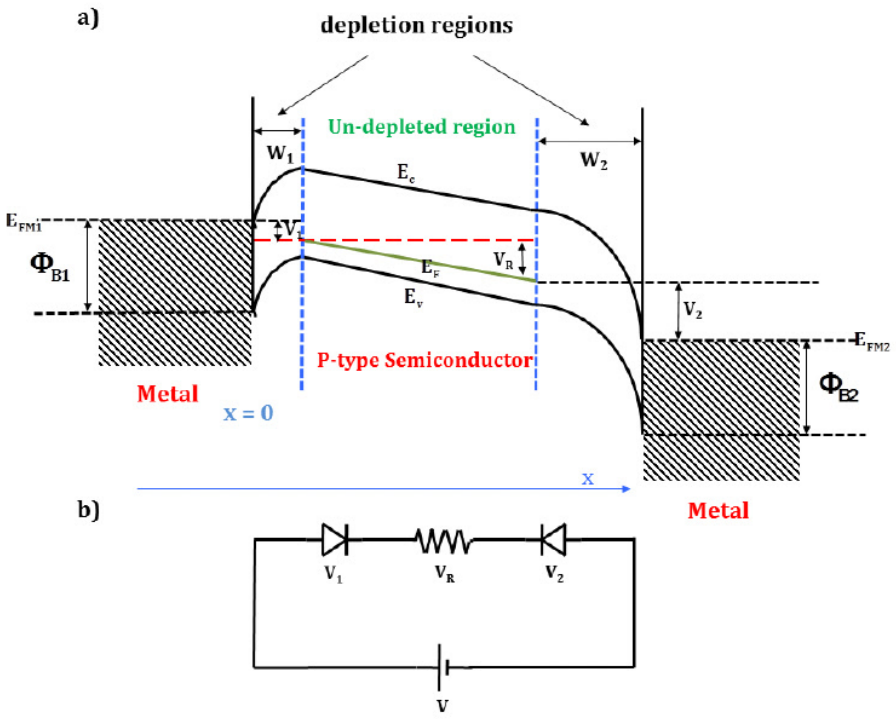


Fig. 2.2 (a) Energy band diagram of a metal-semiconductor-metal structure represented as two opposing Schottky junctions separated by a semiconductor channel. (b) Equivalent electrical circuit consisting of two Schottky diodes connected in series with the nanowire resistance.[6]

therefore determined by the combined behavior of both interfaces rather than by a single contact alone.

This architecture differs fundamentally from conventional filamentary memristors. In oxide-based devices, switching is typically associated with the formation and rupture of conductive pathways inside the active layer. In contrast, the present structure possesses no thin dielectric layer capable of supporting classical filament growth across the entire channel length [47].

Instead, the electrical response emerges from the interaction between the two Schottky barriers, the semiconductor nanowire, and the surrounding oxide environment. This distinction is one of the key scientific motivations underlying the study of silicon nanowire memristors [55].

2.2.6 Surface Oxide Engineering

Although the silicon nanowire and Schottky contacts form the structural foundation of the device, increasing evidence suggests that the surface oxide layer plays a decisive role in determining the memristive response.

Silicon surfaces are highly reactive and form a thin native oxide layer quickly when exposed to air. Additional oxide can also form during fabrication steps such as plasma processing, thermal treatments, or other surface modification procedures.

Traditionally, these oxide layers were viewed merely as unavoidable surface contaminants. However, recent studies have demonstrated that they actively participate in the switching process [56].

The oxide layer contains defect sites where charge can become temporarily trapped. As charge carriers are captured and released, the device does not respond immediately to changes in the applied voltage. Instead, its response is influenced by what happened previously (history), resulting in hysteresis in the current–voltage response. This behavior is closely associated with the memory effect observed in the device.

Importantly, experimental observations have shown that freshly fabricated devices often exhibit little or no hysteresis immediately after fabrication. As the oxide layer develops or is modified through oxygen plasma treatment, pronounced memris-

tive behavior emerges. These findings strongly suggest that the oxide is not a passive component but rather an active participant in the switching mechanism [8].

2.2.7 Physical Origin of Volatile Memristive Behavior

One of the most intriguing aspects of silicon nanowire memristors is their tendency to exhibit volatile switching rather than conventional non-volatile memory operation.

In volatile devices, the conductance is changed during electrical stimulation but relaxes back to its initial state slowly after the stimulus is removed. This behavior is different with filament-based memories which the resistance state can be retained for extended periods.

The currently accepted explanation involves a combination of charge trapping, interface-state dynamics, and Schottky barrier modulation.When a voltage is applied, electrical energy is temporarily stored within oxide traps and interface states. The resulting redistribution of charge modifies the local electric field and alters the effective Schottky barrier height. Consequently, the conductance of the device changes [57].

Once the external stimulus is removed, the trapped charge is gradually released and device returns back to its initial state. As this occurs, the barrier properties recover and the conductance slowly changes back to its original value. This relaxation process is responsible for the volatile memory behavior observed in the device and cause responses that are similar to short-term memory, adaptation, and other transient neural phenomena.

For this reason, silicon nanowire devices with Schottky contacts and engineered oxide interfaces have emerged as promising candidates not only for biosensing applications but also for neuromorphic computing systems that require dynamic and energy-efficient information processing [8].

2.3 Fabrication Technologies for Silicon Nanowire Devices

Fabricating nanoscale electronic devices requires careful control over device dimensions, interfaces, and material properties. At these scales, even small variations in geometry, surface roughness, interface quality, or fabrication-induced defects can have affect on the electrical behavior of the device. For this reason, the fabrication process has a significant impact on the final device performance.

The objective of this section is to introduce the fundamental fabrication technologies employed throughout this thesis and to discuss the physical principles underlying each process. Understanding these technologies is essential for interpreting the structural and electrical properties of the fabricated devices.

2.3.1 Top-Down and Bottom-Up Nanofabrication Approaches

Nanostructures can generally be fabricated using either bottom-up or top-down methodologies.

Bottom-up fabrication is based on the growth of nanoscale structures through chemical synthesis or self-assembly. Examples include vapor–liquid–solid nanowire growth, chemical vapor deposition, molecular self-assembly, and solution-based synthesis. While these approaches can produce high-quality nanostructures, controlling their placement, alignment, and large-scale integration remains a significant challenge.

Top-down fabrication has different approach. Instead of growing structures from the atomic scale upward, the device is formed by patterning and removing material from a larger substrate using techniques such as lithography, etching, and deposition.

The main advantages and limitations of the two fabrication approaches are summarized in Fig. 2.3. Bottom-up methods are well suited for producing high-quality nanostructures but top-down techniques provide better control over device dimensions and are generally more compatible with large-scale manufacturing.

Top-down fabrication offers several advantages for electronic devices intended for integration with conventional semiconductor manufacturing. Device dimensions can be controlled with high accuracy, alignment between different fabrication steps

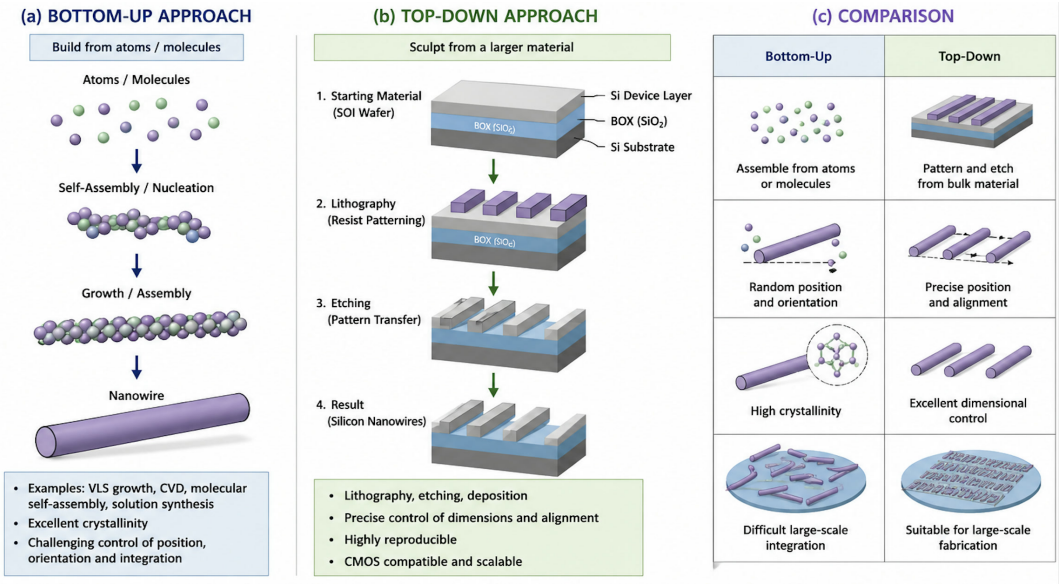


Fig. 2.3 Comparison between bottom-up and top-down nanofabrication approaches.

can be maintained at the nanometer scale, and large numbers of devices can be produced using a consistent fabrication process.

The silicon nanowires investigated in this work were therefore fabricated using a top-down approach derived from standard CMOS technology. In this process, the SOI substrate provides the isolated device layer, while lithography and etching define the suspended nanowire geometry with high dimensional control and compatibility with future large-scale integration.

2.3.2 Lithography as the Foundation of Nanofabrication

Lithography represents the central pattern-transfer technology in modern microelectronics. Virtually every nanoscale device requires lithographic patterning at multiple stages during fabrication.

Lithography is the process used to create the desired device pattern on the wafer surface. This pattern is then used to control other fabrication steps such as etching, deposition, implantation, and lift-off.

The general lithographic workflow consists of four fundamental steps: resist deposition, exposure, development, and pattern transfer.

Lithography starts by coating the substrate with a radiation-sensitive material which is called a resist. Portions of the resist are then exposed to ultraviolet light, electrons, or ions, depending on the lithography technique being used. The exposure changes the chemistry of the resist, it causes that selected regions to be removed during development. The remaining resist forms a patterned mask that defines the device features for the next fabrication steps.

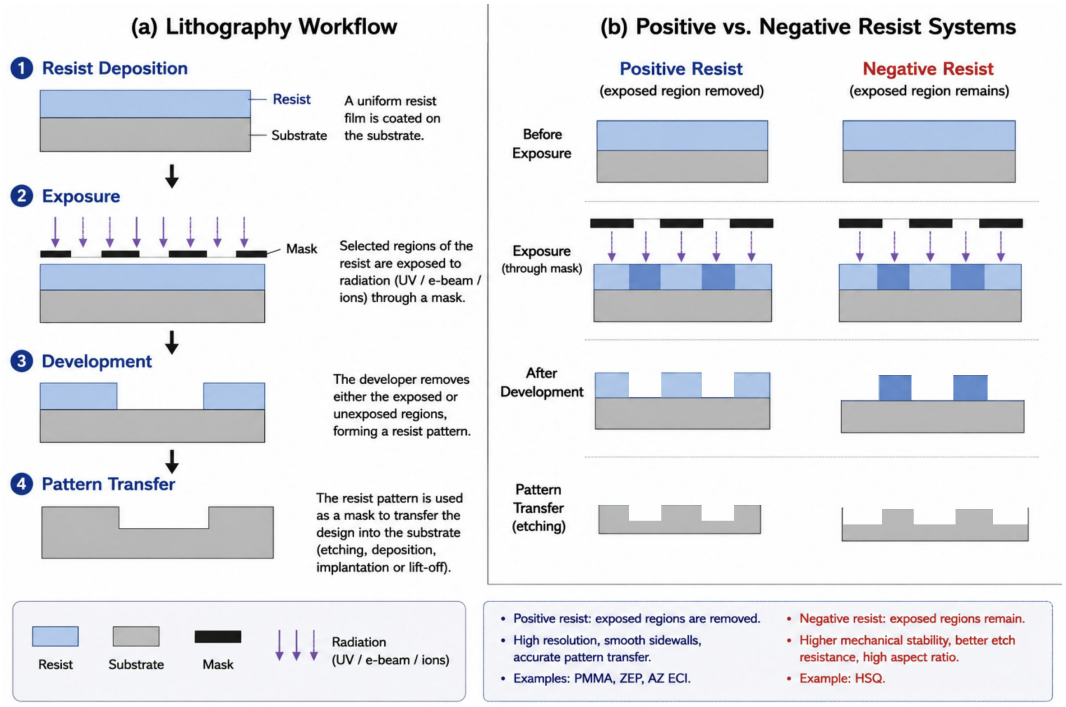


Fig. 2.4 (a) lithographic workflow consisting of resist deposition, exposure, development, and pattern transfer.(b) Comparison between positive and negative resist systems.

As illustrated in Fig. 2.4, lithography relies on a sequence of resist coating, exposure, development, and pattern-transfer steps, while the final resist profile depends on whether a positive or negative resist system is employed.

2.3.3 Positive and Negative Resist Systems

Photoresists are divided into two categories, positive and negative. It is based on how they respond to exposure. Although both serve the same purpose of defining patterns during lithography, the development process is different based on the type of resist used.

For positive resists, the exposed regions become more soluble in the developer and are removed during development and it leaves the unexposed regions behind. This behavior makes positive resists particularly useful when accurate pattern transfer and fine feature definition are required. In contrast, exposure of a negative resist causes chemical crosslinking and make the exposed areas less soluble. After development, the exposed regions remain while the unexposed areas are removed.

The choice of resist depends on the requirements of the fabrication process. In this work, PMMA (Poly(methyl methacrylate))-based resists were used for lift-off processes and Hydrogen Silsesquioxane (HSQ) was selected for the fabrication of high-resolution nanowire structures because of its ability to define nanoscale features with good dimensional control.

2.3.4 Electron-Beam Lithography

Electron-beam lithography (EBL) is one of the most common techniques used for fabricating nanoscale structures. Instead of using light, EBL employs a focused beam of electrons to expose the resist. Since electrons have much shorter wavelengths, much smaller features can be patterned. Another advantage of EBL is that it does not require a physical mask. Patterns can be written directly onto the resist from a computer-generated design and it makes the technique particularly useful during device development and prototyping.

Because of its high resolution and patterning flexibility, EBL has become a standard tool in nanofabrication research. Modern systems are capable of defining structures with dimensions well below 10 nm under optimized conditions. Table 2.1 summarizes the resolution, overlay accuracy, and line edge roughness of several lithography techniques commonly used in micro and nanofabrication.

Table 2.1 Comparison of lithography tools used for micro and nanofabrication.

Process (Tool)	**Typical Minimum Feature**	**Overlay Accuracy (3σ)**	**Typical LER (3σ)**
Maskless Aligner (MLA150)	0.6–1.0 μm	$\sim$500 nm (front), $\sim$1 μm (back)	–
Laser Pattern Generator (VPG+200)	0.75–1.0 μm	225–500 nm	$\sim$30–70 nm
Electron-Beam Lithography (100 kV)	5–10 nm	$\sim$10–15 nm	$\sim$2–3 nm

As shown in Table 2.1, EBL provides substantially higher resolution and overlay accuracy than optical lithography techniques, making it particularly suitable for the fabrication of nanoscale silicon nanowire devices.

One limitation of EBL is that the pattern is written point by point, which makes the process relatively slow compared with conventional photolithography. As a result, EBL is not suitable for high-volume manufacturing. Nevertheless, in research environments where feature size and pattern accuracy are more important than fabrication speed, it can be one of the most widely used lithography techniques.

2.3.5 Thin-Film Deposition Technologies

After lithographic pattern definition, functional materials must be deposited onto the substrate.

Thin-film deposition technologies can generally be classified into physical deposition methods and chemical deposition methods.

Thermal evaporation and electron-beam evaporation are two of the most commonly used techniques for depositing thin films. In these processes, the source material is heated in a high-vacuum environment until it vaporizes. The vaporized atoms reach the substrate surface and they accumulate and form a thin layer of material.

The vacuum environment has several important functions like preventing contamination, minimizing scattering, improving film purity and enhancing thickness uniformity. The deposited film properties depend on multiple factors, like substrate temperature, deposition rate, chamber pressure, and material selection. For nanoscale electronic devices, careful control of these parameters is critical because film morphology influences contact resistance, adhesion, and interface quality directly.

2.3.6 Lift-Off Pattern Transfer

One of the most widely used techniques for defining metallic structures is lift-off. Unlike subtractive patterning, where a continuous metal film was deposited and subsequently etched, lift-off uses a sacrificial resist mask to define the desired geometry before deposition.

The process begins with the lithographic patterning of a resist layer on the substrate. fter defining the ppatern, metal is deposited over the entire surface. Then , the resist is dissolved and removing the metal deposited on top of it and leaving

behind only the metal that is in direct contact with the substrate. As a result, the desired metal pattern is obtained without the need for a separate metal etching step.

One advantage of this approach is that it avoids exposing the metal layer to additional etching processes. This makes lift-off particularly suitable for materials that are difficult to etch and for delicate structures that may be affected by more aggressive fabrication steps.

2.3.7 Silicidation Technology

As device dimensions continue to shrink, contact resistance becomes an increasingly important limitation. One of the most effective solutions is silicidation, whereby a deposited metal reacts with silicon during thermal annealing to form a conductive silicide compound.

Among various silicide systems, NiSi has become especially attractive due to Low resistivity, CMOS compatibility, Low formation temperature and Excellent scalability.

During annealing, nickel atoms diffuse into silicon and react to form silicide phases. The NiSi formed during annealing provides electrical contact for silicon and establishes the Schottky junctions which is used in the device. Because these interfaces play a key role in charge transport, careful control of the silicidation process is important for obtaining consistent device behavior.

2.3.8 Deep Reactive Ion Etching (DRIE) and the Bosch Process

The fabrication of suspended silicon nanowires requires highly anisotropic silicon etching capable of producing narrow structures with large aspect ratios.

DRIE satisfies this requirement through a cyclic process known as the Bosch process. The Bosch process alternates between Silicon etching and Sidewall passivation.

As illustrated in Fig. 2.5, the Bosch process achieves anisotropic silicon etching through alternating passivation and etching cycles, allowing vertical structures to be formed while minimizing lateral undercutting [7].

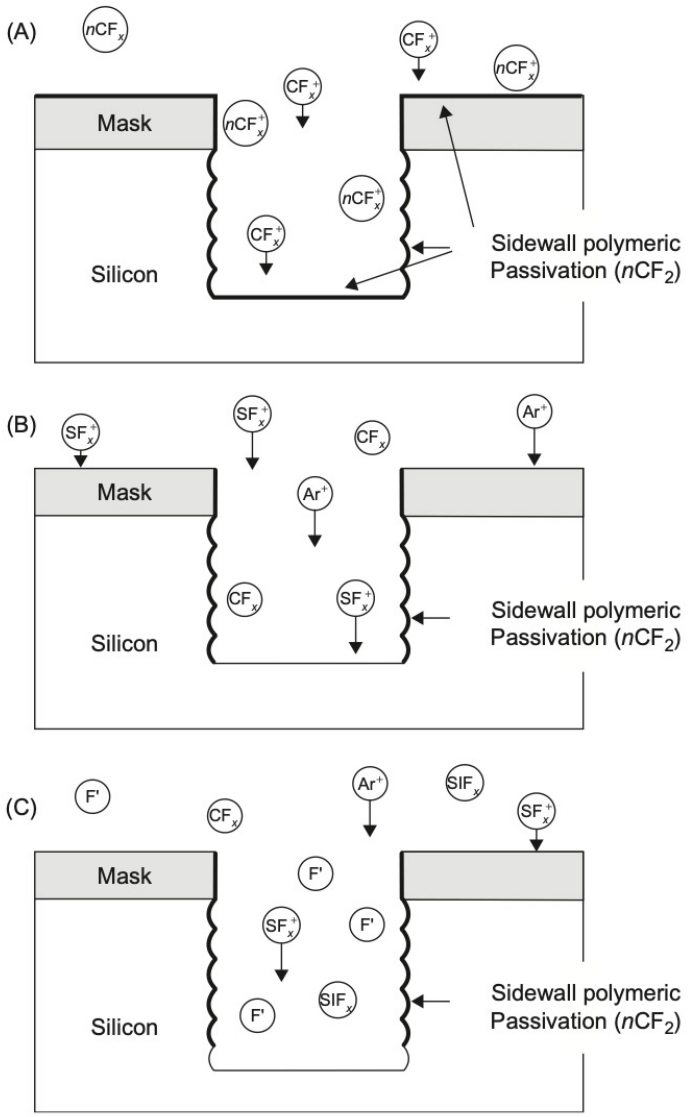


Fig. 2.5 Schematic illustration of the Bosch DRIE process. (A) During the passivation step, a fluorocarbon polymer layer is deposited on all exposed surfaces. (B) Directional ion bombardment selectively removes the passivation layer at the trench bottom while preserving sidewall protection. (C) Fluorine radicals generated from SF_6 plasma etch the exposed silicon, resulting in highly anisotropic vertical etching. [7].

During the etching phase, SF_6 plasma generates fluorine radicals that chemically remove silicon. During the passivation phase, fluorocarbon species deposit a protective polymer layer on all exposed surfaces.

When the next etching cycle begins, ion bombardment removes the polymer mainly from the bottom of the trench while most of the sidewall protection remains intact. As a result, the etch progresses in the vertical direction, with only limited lateral etching. Repeating these cycles allows deep structures with vertical sidewalls to be fabricated. In the case of suspended silicon nanowires, this level of control is important for defining narrow channels.

2.3.9 Surface Chemistry and Oxide Engineering

At nanoscale dimensions, the surface becomes an active component of the device rather than merely a boundary.

Silicon surfaces naturally form a thin oxide layer when exposed to air, and also additional oxide can develop during fabrication steps like plasma treatments or thermal processing. These oxide layers (Altough be thin) can affect device behavior by influencing charge trapping, carrier transport, interface states, Schottky barrier properties, and the overall memristive response. For this reason, surface preparation and oxide control are important considerations during device fabrication.

For the devices studied in this thesis, the oxide layer has a strong influence on the observed hysteretic behavior. As a result, the fabrication steps that affect the oxide must be considered when interpreting the electrical response of the final devices.

2.4 Fabrication Process of the Suspended Silicon Nanowire Memristive Device

Following the discussion of the device architecture and the fabrication technologies employed throughout this work, this section presents the complete fabrication process used to realize the suspended silicon nanowire memristive devices. The fabrication strategy was designed to satisfy several simultaneous objectives. First, the process must produce nanowires with dimensions sufficiently small to maximize

surface sensitivity and interface effects. Second, the resulting structure must remain mechanically stable after suspension. Third, the contact regions must provide reproducible Schottky barriers capable of supporting memristive behavior. Finally, all fabrication steps should remain compatible with conventional CMOS processing technologies to facilitate future scalability.

The overall process follows a top-down approach implemented on SOI substrates. Starting from a crystalline silicon device layer, successive lithography, deposition, silicidation, and etching steps are used to transform the wafer into a suspended nanowire structure connected to NiSi contacts. The complete process flow is illustrated in Figure 2.6.

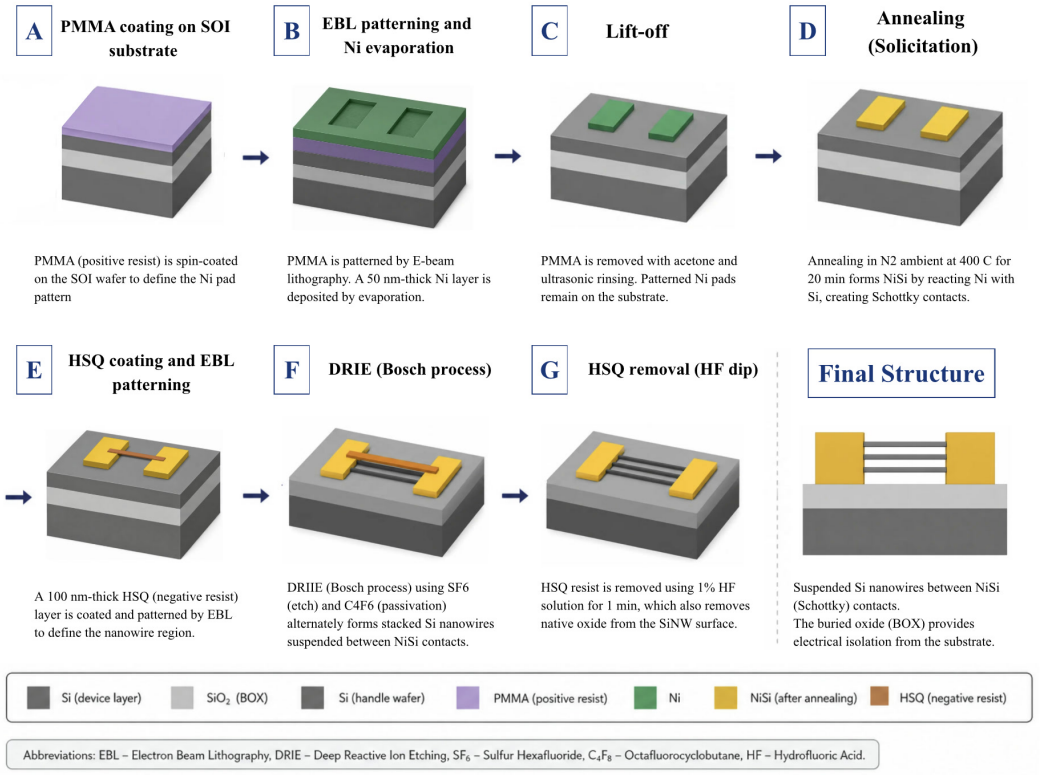


Fig. 2.6 Schematic fabrication sequence of the suspended silicon nanowire device: (A) PMMA coating, (B) EBL patterning and Ni deposition, (C) lift-off, (D) Ni silicidation, (E) HSQ nanowire definition, (F) Bosch DRIE etching, and (G) HSQ removal to obtain the final suspended silicon nanowire structure between NiSi Schottky contacts.

2.4.1 Starting Material Selection

The fabrication begins with commercially available SOI wafers. The selection of the substrate is one of the most important decisions because it directly influences both fabrication feasibility and electrical performance.

The devices were fabricated on 4-inch p-type SOI wafers with a (100) crystal orientation and a resistivity of 14–22 Ω·cm. This resistivity corresponds to an estimated acceptor concentration of approximately 5.9×10^{14} to 9.4×10^{14} cm^{-3} and it cause the silicon device layer be in the lightly doped regime. The SOI structure consisted of a 1 μm silicon device layer which is separated from the silicon handle wafer by a 1 μm buried oxide layer. The handle wafer itself had a thickness of approximately 625 μm.

Low doping concentration was selected in order to preserve the influence of Schottky barriers and interface-related effects on charge transport. It was important for the present study, since the memristive behavior of the device is closely related to processes occurring at interfaces and within the surface oxide layer.

Before fabrication, wafers were cleaned using standard semiconductor cleaning procedures. The goal of this cleaning step was to remove organic residues, metallic contaminants, and any native oxide present on the surface. Proper surface preparation was important for achieving reliable lithographic patterning, good resist adhesion, and consistent metal–semiconductor interfaces throughout the fabrication process.

2.4.2 Alignment Marker Fabrication

Before the fabrication of the active devices began, alignment markers were first defined on the substrate. While these markers are not part of the final device structure, they are essential for ensuring that the different lithography steps are correctly aligned during fabrication process. Since nanowire and contact regions are patterned in separate steps, even small alignment offsets can influence the final device geometry.

For this reason, alignment markers fabricated by EBL were used. These markers served as fixed reference points, it allows each lithography level to be positioned accurately with respect to the previous one. This was particularly important during

the patterning of the nanowire and contact structures, where precise alignment was required.

The markers were patterned by EBL and subsequently transferred into the silicon device layer by plasma etching. The etch extended slightly into the buried oxide, creating permanent topographical features that remained visible during later processing steps. These features were then used as reference marks for all subsequent lithographic alignments.

2.4.3 Definition of Nickel Contact Regions

After marker alignment, the regions for the nickel contacts were patterned using EBL. A bilayer resist stack consisting of approximately 150 nm Methyl Methacrylate (MMA) (EL6) and 150 nm PMMA (495K A4) was used for this step. The two-layer resist structure was chosen to facilitate the subsequent lift-off process.

During development, the bilayer resist naturally forms an undercut profile beneath the patterned openings. This undercut helps separate the deposited metal from the resist sidewalls, resulting in cleaner pattern definition and a more reliable lift-off process after metal deposition.

Following resist coating and baking, the contact structures were exposed by EBL and developed in a Methyl Isobutyl Ketone (MIBK) (1:3) solution. This process created openings in the resist that defined the locations of the future nickel contacts. A short oxygen-plasma descum treatment (10 s) was then carried out using an Oxford PRS900 RF plasma system to remove any residual resist remaining on the exposed silicon surface.

2.4.4 Nickel Deposition and Lift-Off

After completion of the lithographic patterning, nickel was deposited onto the substrate by electron-beam evaporation. Electron-beam evaporation was selected because it provides highly pure metallic films with excellent thickness control and minimal contamination.

A 50 nm thick nickel layer was deposited onto the patterned substrate by electron-beam evaporation. Lift-off was subsequently performed in acetone at 23 °C for

25 min with ultrasonic agitation (power level 7), removing the unwanted metal and leaving only the nickel structures defined by the lithographic pattern.

2.4.5 Formation of NiSi Schottky Contacts

After lift-off, the samples were annealed in a nitrogen atmosphere. During this step, the deposited nickel reacted with the underlying silicon, resulting in the formation of NiSi at the contact regions.

In this work, annealing was carried out at 400 °C for 20 minutes in an inert nitrogen environment. This condition was used consistently throughout the fabrication process. The silicidation step changes the nature of the contacts by creating a metal–semiconductor interface rather than a simple metal layer deposited on the silicon surface.

The electrical properties of the resulting contacts are related to quality of this interface. For this reason, the annealing conditions must be carefully controlled because variations in temperature or annealing time can affect the behavior and reproducibility of the final devices.

2.4.6 High-Resolution Nanowire Definition

Once the contact regions had been fabricated, the silicon nanowire was defined. For this step, HSQ 006 was used as a negative-tone electron-beam resist because of its suitability for high-resolution patterning. The resist was spin-coated at 4000 rpm, resulting in a film thickness of approximately 130–140 nm.

The nanowire patterns were written using a Vistec EPBG5000ES EBL system with an exposure dose of 6800 μC/cm^2. After exposure, the samples were developed in Tetramethylammonium Hydroxide (TMAH), which removed the unexposed resist and left behind the cross-linked HSQ structures which are correspond to nanowire pattern.

These HSQ structures were then used as an etch mask during the subsequent DRIE step. The remaining HSQ layer provided sufficient resistance to the plasma etch and allow the nanowire geometry to be transferred accurately into the underlying silicon layer.

2.4.7 DRIE and Nanowire Release

The nanowire pattern was transferred into the silicon device layer using Bosch DRIE. The Bosch process alternates between SF_6-based silicon etching and fluorocarbon sidewall passivation cycles, enabling highly anisotropic etching with excellent control of vertical sidewalls.

To ensure that the silicon surrounding the nanowire was completely removed, the AMS SOI 4- recipe was used with a total etch time of 38 s. The etch was intentionally extended slightly beyond the nominal endpoint to provide a small over-etch margin and reduce the risk of residual silicon remaining between adjacent structures. During the process, the buried oxide layer served as an effective etch stop, helping to preserve the underlying substrate.

2.4.8 Removal of HSQ and Surface Preparation

Following nanowire formation, the HSQ mask must be removed. The HSQ mask was removed by immersing the samples in a 1% Hydrofluoric Acid (HF) solution for 1 min. This treatment simultaneously removed the HSQ mask, residual Bosch-process passivation layers, and the native silicon oxide present on the nanowire surface, leaving a clean silicon interface for subsequent characterization.

This step is particularly important because the oxide-free silicon surface subsequently undergoes natural reoxidation and surface-state evolution, phenomena that may influence the electrical behavior discussed in later chapters.

As the HF treatment removes the native oxide without introducing intentional doping, the silicon nanowires remain close to intrinsic conditions, resulting in a very low free-carrier concentration at room temperature.

2.4.9 Final Electrode Fabrication

To facilitate electrical characterization, larger probing electrodes were fabricated after completion of the suspended nanowire structure using a conventional photolithography, metal deposition, and lift-off process. These electrodes provide a reliable interface between the nanoscale device and external measurement equipment.

2.4.10 Final Device Structure

Upon completion of all fabrication stages, the resulting device consists of a suspended silicon nanowire bridging two NiSi contact regions and connected to larger probing electrodes.

Electrically, the device can be described as a silicon channel coupled to two Schottky junctions which are arranged in a back-to-back configuration.

Mechanically, the nanowire remains suspended above the buried oxide region and it maximize exposure of the active surface.

From a functional perspective, the completed structure provides a unique platform where Schottky-barrier modulation, surface-state dynamics, oxide-related charge trapping, and nanoscale carrier transport interact simultaneously. These interactions ultimately give rise to the volatile memristive behavior investigated throughout the remainder of this thesis.

The overall fabrication process is a top-down approach and is compatible with standard CMOS technology. This provides a practical route for the fabrication and future integration of silicon nanowire devices for neuromorphic computing applications.

Chapter 3

Structural and Electrical Characterization: Oxide Engineering

3.1 Structural Characterization

3.1.1 Scanning Electron Microscopy (SEM) Characterization

SEM was used to examine the morphology and structural quality of the fabricated silicon nanowire devices. In this technique, a focused electron beam scans the sample surface, and the emitted electrons are collected to form high-resolution images. Because of high spatial resolution of SEM, it is useful for studying nanoscale structures, while its large depth of field allows detailed observation of complex device geometries. For these reasons, SEM was used throughout this work to evaluate the fabricated nanowire devices.

The structural characterization was carried out using a Zeiss MERLIN field-emission scanning electron microscope (FE-SEM). The instrument is capable of sub-nanometer resolution and can be operated over a wide range of accelerating voltages, from 0.02 to 30 kV. It is equipped with secondary electron (SE), In-Lens, and energy-selective backscattered electron detectors, allowing different imaging modes to be used depending on the features of interest. The system also includes energy-dispersive X-ray spectroscopy (EDS), which can be used to obtain compositional information from the sample.

SEM characterization was used to assess the quality of the fabricated devices. It allows examination of the overall device geometry and direct visualization of the suspended nanowire structures. Also, SEM imaging was useful for identifying fabrication-related defects such as structural collapse, fractures, discontinuities, or excessive surface roughness.

Figure 3.1 shows SEM images of a fabricated suspended silicon nanowire device at different magnifications. The lower-magnification images show the overall device layout and higher-magnification views provide a closer look at the active nanowire region.

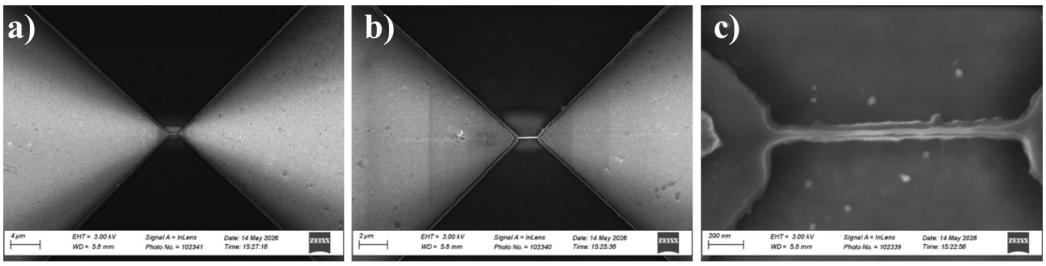


Fig. 3.1 SEM images of the fabricated suspended silicon nanowire device. (a) Overall view of the complete device structure. (b) Magnified view of the central region. (c) High-magnification image of the silicon nanowire.

The low-magnification SEM images in Figures 3.1a and 3.1b show an overview of the fabricated device. The structure consists of two contact regions which are connected by a narrow suspended nanowire at center of the device. The overall geometry appears well defined, and the different features are aligned as intended. No obvious discontinuities or major defects can be observed in the active region.

A higher magnification image of the nanowire region is shown in Figure 3.1c. The suspended structure remains intact after the release process and it maintains a uniform shape along its length. No signs of collapse, fracture, or significant sidewall damage are visible in the image. These observations suggest that the lithography and etching steps successfully produced the desired nanowire structure.

In overall, the SEM images confirm that the target device geometry was successfully fabricated. The nanowire remains suspended between the two contact regions and keeps its structural integrity after processing. This provides a suitable starting point for the compositional and electrical characterization presented in the following sections.

3.1.2 Atomic Force Microscopy (AFM) Characterization

AFM was used to investigate the surface morphology of the fabricated silicon nanowire devices. In contrast to electron microscopy techniques, AFM generates three-dimensional topographical information by scanning a sharp probe across the sample surface. The measurements were carried out using a Bruker AFM system equipped with closed-loop positioning and peak-force imaging capabilities which is suitable for characterization of surface features with nanometer-scale resolution.

Figure 3.2 illustrates representative AFM height and peak-force-error images of fabricated devices. The AFM height maps provide direct information regarding the surface morphology and topographical variations across the patterned structures and peak-force-error images improve the visibility of edges and structural boundaries, making it easier to distinguish different regions of the device.

Figures 3.2a and 3.2b focus on a region near the edge of the fabricated structure. A clear height difference can be observed between the patterned device area and the surrounding etched region. This transition is more clearly visible in the corresponding peak-force-error image, which highlights the boundary between the two regions and confirms the formation of well-defined device edges during fabrication.

Figures 3.2c and 3.2d show AFM images of larger area of the device. The overall layout of the structure can be identified including the contact regions and the central active area. The observed height contrast between different regions of the device shows that the lithography and etching steps were successfully transferred into the final structure.

Although AFM is well suited for examining the overall surface morphology of the device, it provides only limited information about the suspended nanowire itself. Because the nanowire is both extremely small and suspended above the substrate and probing this region with high accuracy can be challenging. As a result, techniques with higher spatial resolution are needed to investigate the nanowire geometry and structural details.

Even with these limitations, the AFM measurements provide useful information about the surface topography and structural integrity of the fabricated devices. Together with the SEM results, the AFM observations confirm the successful fabrication of the suspended silicon nanowire structure.

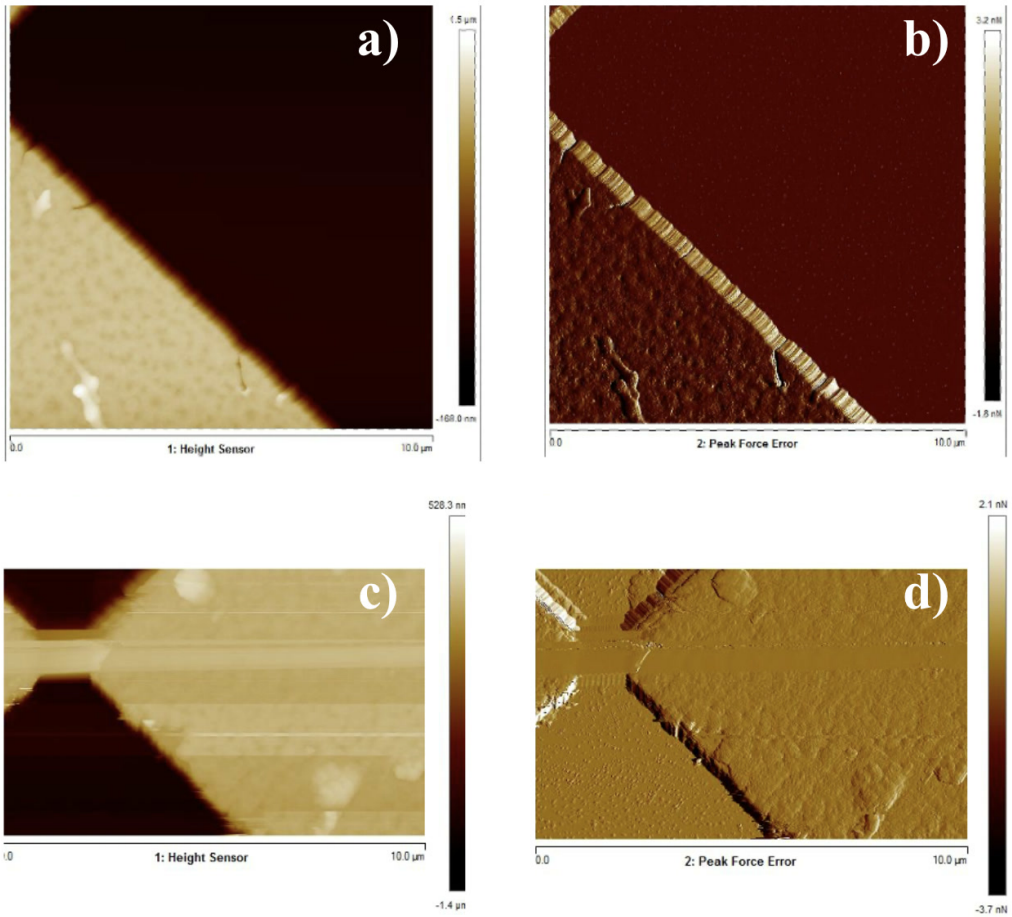


Fig. 3.2 AFM characterization of the fabricated silicon nanowire device, (a) Height image(b) Corresponding peak-force-error image, (c) Large-area AFM height, (d) Large-area AFM Peak-force-error image.

3.1.3 FIB-SEM Characterization of the Suspended Silicon Nanowire

To further examine the three-dimensional structure of the fabricated devices, focused ion beam–scanning electron microscopy (FIB-SEM) was carried out using a Zeiss CrossBeam 550 dual-beam system. By combining SEM imaging with focused ion beam (FIB) milling, this technique enables cross-sectional analysis of nanoscale structures that cannot be fully assessed through surface imaging alone. In addition to revealing the internal geometry of the device, FIB-SEM provides insight into the quality of the suspension process and the structural integrity of the fabricated nanowires.

Cross-sectional characterization was very important for verifying the final device architecture. Conventional SEM imaging provides detailed information from the top view, it offers only limited access to the vertical dimensions of the structure. FIB-SEM overcomes this limitation by exposing internal cross-sections of the device, allowing the spatial relationship between suspended nanowire, the buried oxide layer, and underlying substrate to be examined directly.

Figure 3.3 presents representative FIB-SEM images acquired at different magnifications. The images confirm the successful fabrication of the suspended silicon nanowire architecture and provide direct evidence of the released device geometry.

Figure 3.3(a) shows a cross-sectional view of the fabricated device after FIB milling. The suspended portion of the structure is clearly visible and can be distinguished from the underlying substrate. The image confirms that the release process successfully created the suspended geometry while preserving the overall structure of the device. The different material layers and interfaces can also be identified, indicating good agreement with the designed architecture.

A side-view image of the suspended region is shown in Figure 3.3(b). The nanowire extends from the anchored region into the suspended section and remains intact after fabrication. A clear gap can be observed between the nanowire and the substrate, confirming that the active channel was successfully released. Within the imaged area, no obvious signs of collapse, fracture, or major deformation are visible.

Figure 3.3(c) provides a higher-magnification view of the active nanowire region. The image reveals the nanoscale dimensions of the suspended structure and demonstrates good dimensional control following lithography, etching, and release process-

ing. Representative dimensions of approximately 69 nm, 93 nm, and 208 nm are measured within the nanowire region, confirming that the fabricated device features remain well within the nanoscale regime. The nanowire maintains a well-defined geometry with smooth structural boundaries, indicating successful fabrication of the intended device architecture.

Overall, the FIB-SEM analysis confirms the successful fabrication of a mechanically stable suspended silicon nanowire structure with well-defined nanoscale dimensions and excellent structural integrity. These results provide direct structural validation of the fabricated devices and establish the physical foundation for the compositional and electrical characterizations presented later in this chapter.

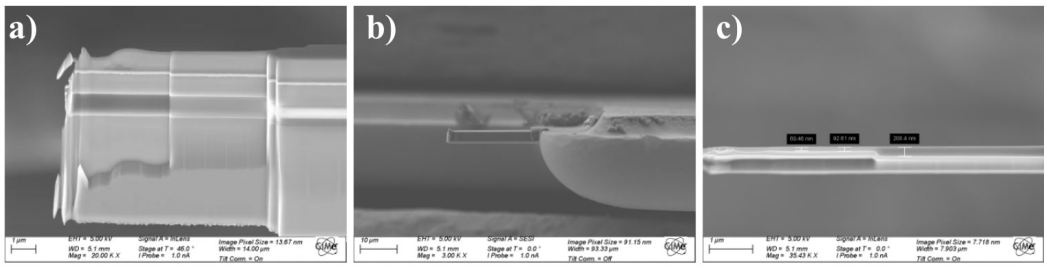


Fig. 3.3 FIB-SEM characterization of the fabricated suspended silicon nanowire device. (a) Cross-sectional view of the device structure obtained after FIB milling. (b) Side-view image of suspended nanowire geometry. (c) High-magnification view of the active nanowire region.

3.1.4 Scanning Transmission Electron Microscopy (STEM) and High-Angle Annular Dark Field (HAADF)-STEM Structural Characterization

To obtain a more detailed view of the fabricated devices, cross-sectional STEM and HAADF-STEM analyses were performed using a Talos F200S transmission electron microscope.The internal structure of the device can be characterized at nanoscale because of high spatial resolution of these techniques and also it is possible to observe material interfaces, layer thicknesses, and structural features that cannot be clearly distinguished using surface imaging techniques alone.

Figure 3.4(a) presents a cross-sectional STEM image of the investigated region. A distinct upper layer is visible above the underlying substrate which is separated by a well-defined interface. The upper layer exhibits a heterogeneous nanoscale

morphology together with a relatively rough surface profile. No evidence of large-scale cracking, delamination, or structural discontinuities is observed within the imaged region.

Figure 3.4(b) shows a higher-magnification view of the structure. Variations in contrast can be seen throughout the upper layer, including several vertically oriented features that extend across its thickness. These observations indicate that the layer is not completely uniform at the nanoscale. However, the structure remains continuous across the analyzed region, and no clear evidence of defects is visible.

Figure 3.4(c) presents a HAADF-STEM image of the same region. HAADF imaging provides atomic-number-sensitive contrast and allows clearer visualization of the layer architecture and interfaces. The image confirms the presence of a continuous upper layer with a clearly distinguishable interface relative to the underlying region. This contrast remains approximately uniform throughout the layer thickness and it shows good structural continuity and the absence of major large-scale inhomogeneities.

Taken together, the STEM and HAADF-STEM results provide a detailed view of the fabricated structure at the nanoscale. The images show a continuous layer with a rough surface profile, clearly distinguishable interfaces, and several vertically oriented structural features. These observations help to clarify the morphology of the device and provide useful context for the compositional characterization discussed in the following section.

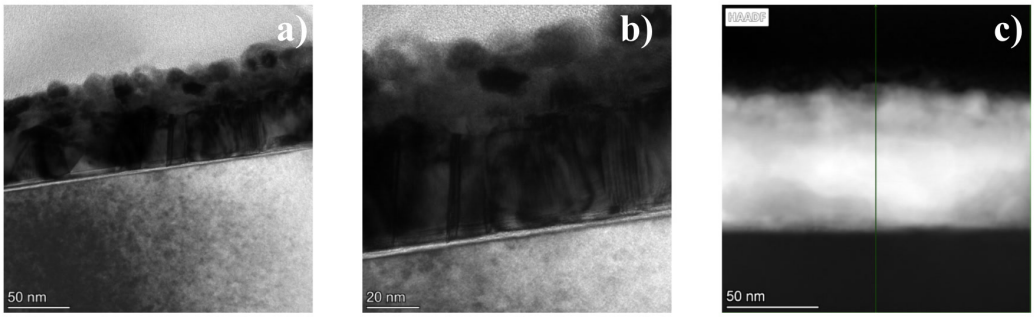


Fig. 3.4 Cross-sectional STEM and HAADF-STEM characterization. (a) STEM image of overall morphology. (b) Higher-magnification STEM image. (c) HAADF-STEM image.

3.1.5 STEM-EDS Elemental Mapping and Line-Profile Analysis

To investigate the elemental composition and spatial distribution of the constituent elements within the fabricated structure, EDS was performed in STEM mode using the Talos F200S transmission electron microscope.Combining STEM imaging with EDS measurements makes it possible to relate the structural features observed in the microscope images to their corresponding elemental composition.

Figure 3.5(a) presents a HAADF-STEM image together with the corresponding nickel elemental map. The nickel signal is primarily observed within the central region of the structure, corresponding to the intermediate layer visible in the STEM image. The corresponding compositional profiles in Figures 3.5(b) and 3.5(c) show a distribution that closely follows the HAADF contrast and it supports the continuity of this layer across the analyzed region.

The oxygen and nickel distributions are illustrated in Figures 3.5(d)–3.5(f). While nickel remains concentrated within the central layer, the oxygen signal is observed near the surrounding interfaces. The line-profile analysis shows regions that the two signals overlap, suggesting the presence of oxygen-containing interfacial layers adjacent to the nickel-rich region.

Figures 3.5(g)–3.5(i) present the combined elemental maps for silicon, nickel, and oxygen. Silicon is mainly located in the lower part of the structure, nickel is concentrated within the intermediate layer, and oxygen is found predominantly near the upper region and at the interfaces surrounding the nickel-containing layer. The corresponding line profiles show a clear separation between the silicon-rich substrate and the nickel-rich region and also it highlights the presence of oxygen at the interfaces between these layers.

The coexistence of silicon-, nickel-, and oxygen-containing regions provides direct compositional evidence for the formation of a well-defined multilayer structure. In particular, the oxygen enrichment observed near the interfaces suggests the presence of oxide-related regions that may significantly influence charge transport and interfacial electronic properties. Such oxide-containing interfaces are expected to play an important role in the electrical behavior and memristive switching mechanisms investigated in the subsequent sections of this chapter.

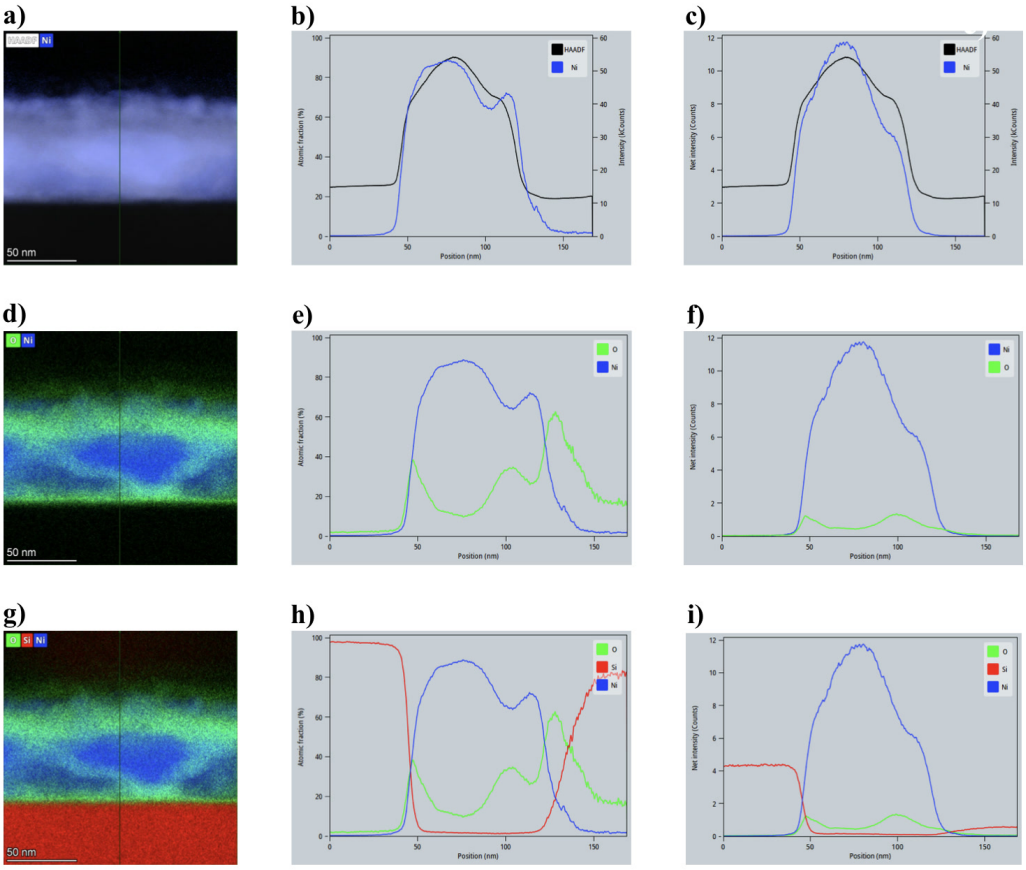


Fig. 3.5 STEM-EDS elemental mapping and compositional profiling of the investigated structure. (a) HAADF-STEM image with nickel elemental mapping. (b,c) Nickel concentration and intensity profiles. (d) Combined oxygen and nickel elemental map. (e,f) Oxygen and nickel concentration and intensity profiles. (g) Composite elemental map showing oxygen, silicon, and nickel distributions. (h,i) Corresponding elemental concentration and intensity profiles.

3.2 Electrical Characterization and Oxide Engineering

The suspended silicon nanowire devices investigated in this thesis exhibit electrical behavior that cannot be understood only from the geometry of the nanowire channel. As discussed in the structural characterization section, the device contains a silicon nanowire connected through metal–semiconductor contacts, while native oxide and oxygen-containing interfacial regions are present around the active structure. These interfaces are particularly important because the current transport is strongly influenced by Schottky-barrier modulation, charge trapping, and oxide-related surface states.

Previous studies from our research group have shown that similar silicon nanowire devices, originally developed for biosensing applications, can exhibit pinched hysteresis before any biofunctionalization step [57]. This observation indicates that the memristive response is not necessarily caused by biomolecular layers, but can originate from the intrinsic device structure itself. In particular, the surface oxide layer, the metal–silicon interface, and the back-to-back Schottky diode configuration play central roles in determining the electrical response.

The purpose of this section is therefore to discuss how surface oxidation and oxygen plasma treatment modify the electrical characteristics of the silicon nanowire device. This discussion is important for two reasons. First, it provides a physical explanation for the observed volatile and non-volatile memristive behavior. Second, it shows that oxide engineering can tune the same silicon nanowire platform toward different neuromorphic functionalities, including synaptic memory, volatile dynamics, and threshold-type neuron-like switching.

3.2.1 Schottky-Limited Transport in the As-Fabricated Device

In the freshly fabricated state, the device behaves primarily as a back-to-back Schottky diode structure. Since the silicon nanowire is contacted by two metal–semiconductor junctions, one contact becomes forward biased while the opposite contact becomes reverse biased under an applied voltage. As a result, the total current is limited by the reverse-biased Schottky junction.

The reverse saturation current can be described using the thermionic emission relation:

$$I_0 = AA^{**}T^2 \exp\left(-\frac{q\phi_B}{k_B T}\right), \tag{3.1}$$

where A is the Schottky contact area, A^{**} is the effective Richardson constant, T is the absolute temperature, q is the elementary charge, k_B is the Boltzmann constant, and ϕ_B is the Schottky barrier height.

Figure 3.6a shows the electrical response of a newly fabricated device. The current–voltage characteristic is dominated by the back-to-back Schottky configuration, and no pronounced hysteresis is observed during the forward and backward sweeps. This indicates that, immediately after fabrication, the interfacial charge-trapping processes are not yet strong enough to produce a clear memristive response.

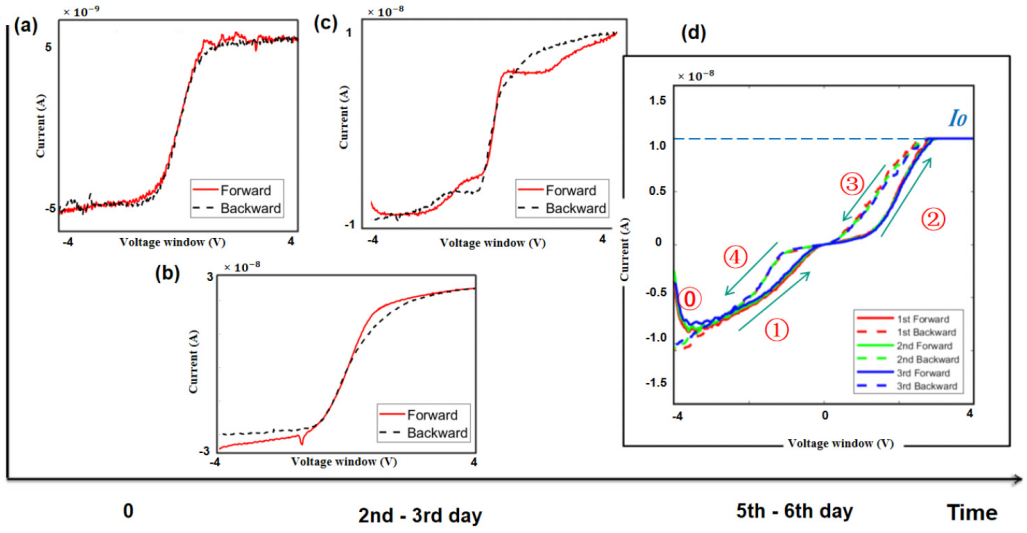


Fig. 3.6 Current–voltage characteristics of the silicon nanowire device during different stages of native oxidation. (a) As-fabricated device. (b,c) Intermediate oxidation stages. (d) Device after native oxidation [8].

3.2.2 Native Oxidation and Emergence of Memristive Hysteresis

Silicon surfaces naturally oxidize when exposed to air. For nanoscale silicon structures, this process is especially important because a large fraction of the active

volume is influenced by the surface. Therefore, even a thin native oxide layer can significantly modify the local electric field, the density of surface states, and the effective Schottky barrier at the metal–silicon interface.

Figures 3.6b and 3.6c show intermediate electrical states during the native oxidation process. In this period, the device exhibits unstable and evolving hysteretic behavior. This suggests that the oxide layer and associated trap states are still developing, leading to a non-equilibrium distribution of interfacial charges.

After longer exposure to ambient conditions, the electrical response becomes more stable, as illustrated in Figure 3.6d. At this stage, the current–voltage curve exhibits a repeatable pinched hysteresis loop which is sign of memristive behavior. The switching event occurs before the current reaches the Schottky-limited regime and it means that the observed conductance changes are linked to processes occurring at or near the interface rather than to an increase in bulk conduction.

These observations indicate that the native oxide layer plays a more active role than simply protecting the silicon surface. The oxide appears to participate directly in the electrical response of the device through charge trapping and release processes, which can influence the Schottky barrier and introduce memory effects into the current–voltage behavior.

3.2.3 Oxygen Plasma Treatment as an Oxide Engineering Method

Although native oxidation can induce memristive behavior, the process is not fully controlled and may lead to device-to-device variability. Oxygen plasma treatment provides a more direct method for modifying the surface oxide and activating oxygen-related states in the silicon nanowire structure.

Figure 3.7 compares the device behavior before and after oxygen plasma treatment. Before plasma exposure, the semilogarithmic current–voltage response shows negligible hysteresis. After treatment, a clear and repeatable hysteresis loop appears within the applied voltage window. This transition demonstrates that oxygen incorporation and oxide modification strongly influence the electrical response of the device.

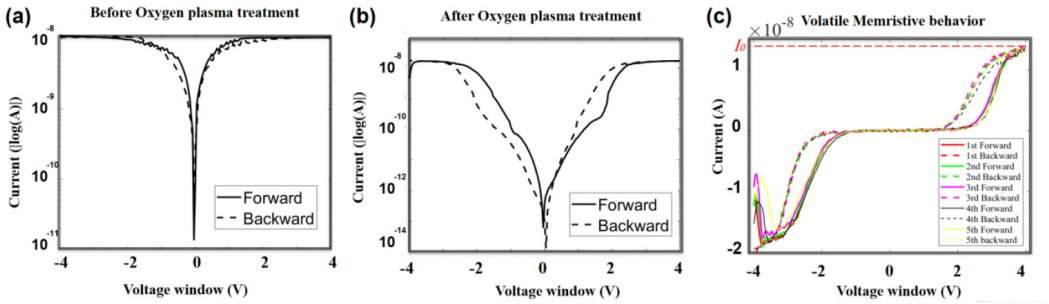


Fig. 3.7 Effect of oxygen plasma treatment on the electrical response of the silicon nanowire device. (a) As-fabricated device, (b) after oxygen plasma treatment, (c) Repeatable volatile memristive response under repeated voltage sweeps [8].

The role of oxygen plasma can be understood through two coupled effects. First, plasma treatment modifies the surface oxide layer and increases the density of oxide-related trap states. Second, oxygen-related defects can influence the local carrier distribution and the effective barrier profile at the metal–silicon interface. Since the device has a quasi-one-dimensional geometry, these surface and interface modifications strongly affect the total current passing through the nanowire.

This hysteresis is commonly attributed to charge trapping and release processes that occur during voltage sweeping. As the applied bias changes, charge can become trapped in defect states within the oxide or near the silicon interface. The presence of this trapped charge changes the local electric field and affects carrier transport. When the bias is reduced or reversed, part of the trapped charge is gradually released and allows the conductance to move back at its initial state. This reversible process is consistent with the volatile memristive behavior which is observed experimentally.

3.2.4 Time-Dependent Conductance Modulation

The dynamic nature of the switching response can be further understood through time-dependent measurements. Figure 3.8 shows the current evolution under applied voltage stress before and after oxygen plasma treatment.

Before oxygen plasma treatment, the current remains nearly constant, indicating that the device does not undergo significant conductance modulation. After oxygen plasma treatment, however, the current gradually increases under the applied voltage.

This gradual transition corresponds to a movement from a high-resistance state toward a low-resistance state.

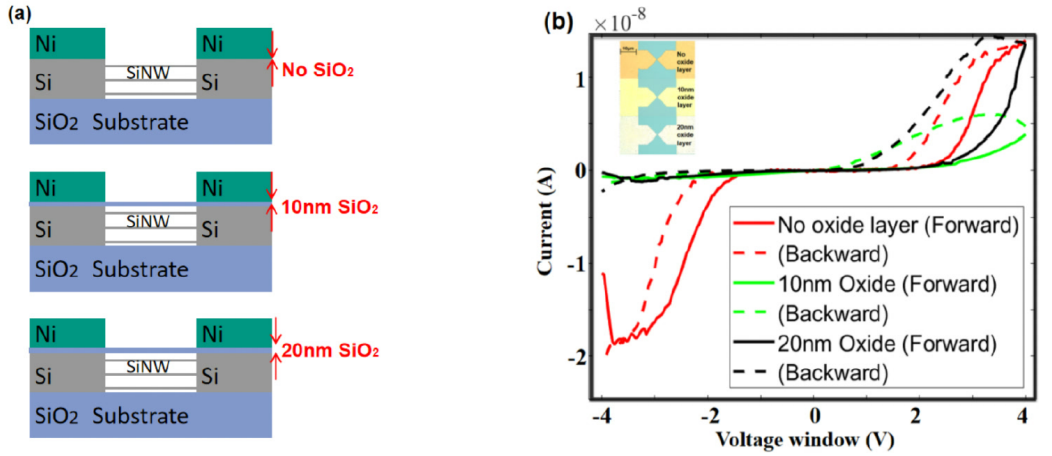


Fig. 3.8 Time-dependent conductance response of the silicon nanowire device after oxide engineering [8].

An important feature of this result is that the conductance does not change abruptly but evolves over time. This behavior is consistent with interfacial trapping and detrapping processes that gradually change the device conductance. When the applied voltage is reduced, the current returns back to its initial level and it shows the conductive state is only temporary under these conditions. This type of dynamic response is very important for neuromorphic systems, where gradual relaxation of conductance can emulate biological processes such as short-term memory and adaptation.

3.2.5 Oxide-Controlled Switching Regimes

Oxygen plasma treatment does not produce only one type of electrical behavior. Depending on the oxidation condition, the same silicon nanowire platform can exhibit different switching regimes. This tunability is summarized in Figure 3.9.

Under light oxidation, only a limited number of deep traps are introduced near the native oxide and silicon interface. Once charged, these traps can retain their state for a longer time, leading to non-volatile conductance modulation. This regime is suitable for synaptic memory, where the device conductance should preserve information after the programming stimulus is removed.

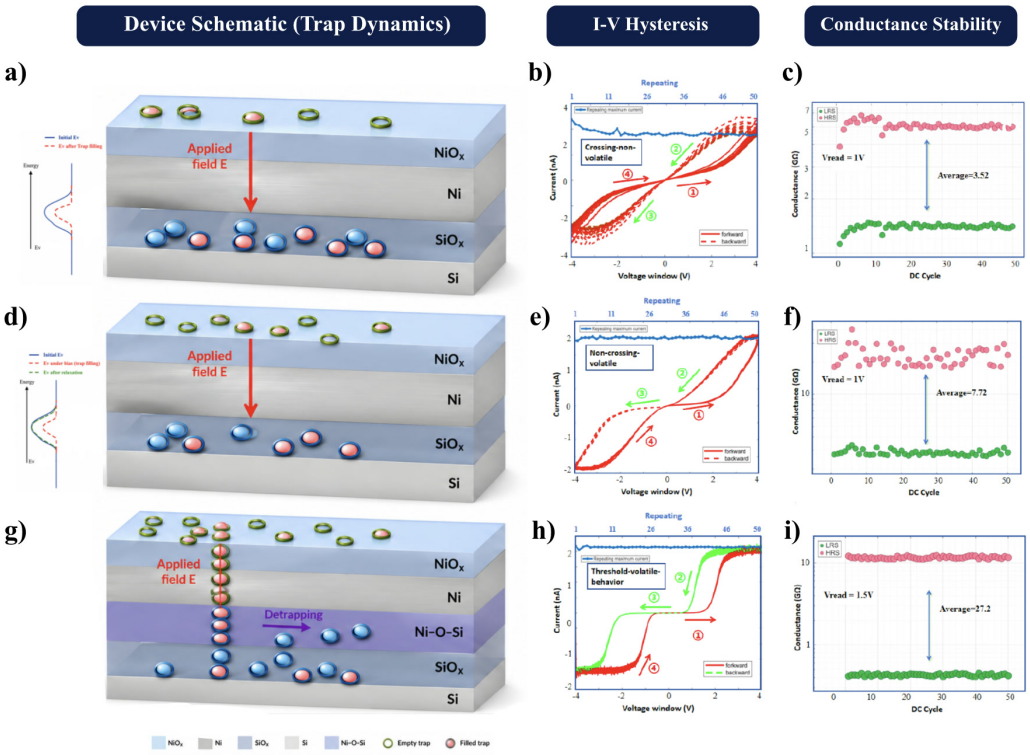


Fig. 3.9 Oxide-controlled switching regimes in silicon nanowire devices. Light oxidation produces trap-controlled non-volatile switching, intermediate oxidation produces volatile memristive switching, and strong oxidation produces threshold-type volatile switching.

At intermediate stages of oxidation, the oxide layer contains a larger number of reversible trapping sites. These traps can capture charges during electrical stimulation and then gradually release it when the bias is removed. As a result, the device exhibits volatile hysteresis and a temporary change in conductance. Such behavior is very suitable for neuromorphic applications that rely on fading memory, short-term plasticity, or other time-dependent responses.

At strong oxidation ,the density of defect states within the oxide increases. Under high electric fields, charge transport can rise abruptly through trap-assisted mechanisms and show threshold-like switching behavior. When the applied bias is removed, the conductive state relaxes and the device returns to its initial high-resistance condition. This type of response shares similarities with the activation and relaxation processes commonly used to emulate neuron-like behavior in neuromorphic systems.

3.2.6 From Device Physics to Neuromorphic Applications

The results discussed above show that oxide engineering provides a practical route for tuning the function of silicon nanowire memristive devices. Rather than treating oxide formation as an uncontrolled surface effect, it can be understood as a functional design parameter.

These different switching behaviors are interesting for neuromorphic computing. Non-volatile operation can be used to store synaptic weights and also in the other hand, volatile switching provides a mechanism for short-term memory and conductance relaxation. Threshold-type switching is also another neuron-inspired functions, where a sudden change in conductance can be used to emulate activation or spike-generation processes.

As a result, the same silicon nanowire device may be used to perform different neuromorphic functions depending on its oxidation state and operating conditions.

This functional flexibility is central to the motivation of this thesis. The silicon nanowire is not only a biosensing structure, but also a reconfigurable nanoelectronic platform whose oxide-mediated dynamics can be exploited for a wide range of neuromorphic computing functions.

Figure 3.10 highlights the evolution of silicon nanowire devices from biosensing device to multifunctional neuromorphic elements. Early studies focused on their

sensing capabilities, where voltage gap and strong surface sensitivity were used to detect biological interactions at the nanowire surface. Because surface charges can directly influence charge transport within the device, the presence of proteins, DNA strands, or cancer biomarkers can produce measurable changes in the electrical response. These properties established silicon nanowires as highly sensitive biosensors and later motivated their exploration for memory and neuromorphic applications. As a result, silicon nanowires have become attractive candidates for next-generation biosensing technologies and bioelectronic interfaces.

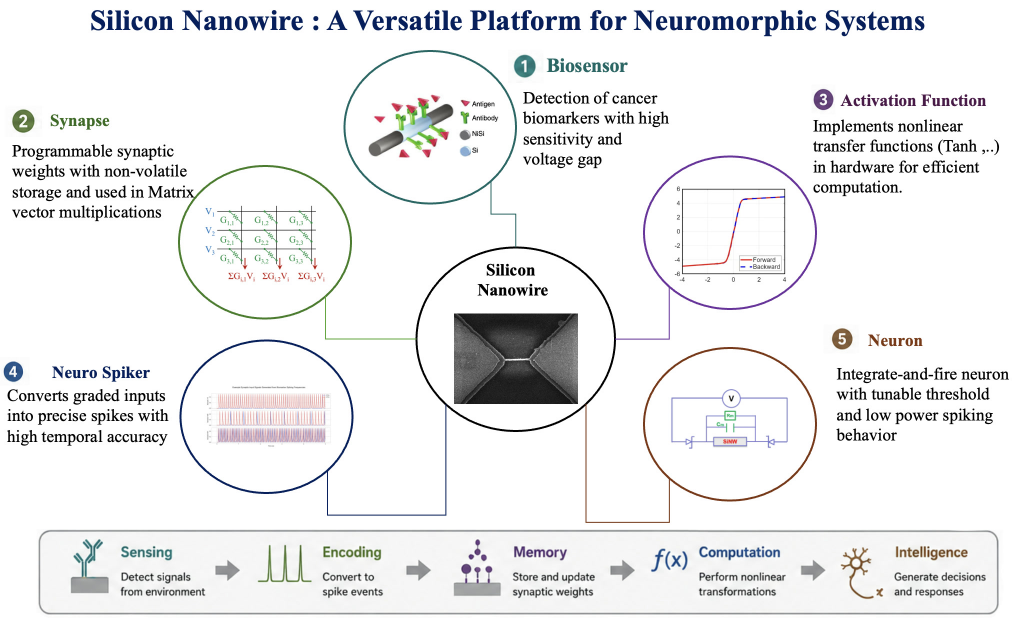


Fig. 3.10 Silicon nanowire as a multifunctional platform for neuromorphic systems.

The findings presented in this chapter demonstrate that the same device platform can be extended far beyond sensing through controlled oxide engineering. By modifying the density, distribution, and dynamics of oxygen-related trap states, the electrical response of the silicon nanowire can be tailored toward multiple neuromorphic functionalities. Light oxidation can enable non-volatile conductance modulation suitable for synaptic weight storage and matrix–vector multiplication. Intermediate oxidation can produce volatile memristive behavior that mimics short-term plasticity and temporal information processing. Stronger oxide engineering can generate threshold-type switching and spiking dynamics, enabling neuron-like activation and spike generation. Furthermore, the nonlinear current–voltage characteristics arising from oxide-mediated barrier modulation can be exploited as hardware activation

functions. Collectively, these functionalities allow the silicon nanowire to serve as a unified platform spanning sensing, spike encoding, memory, nonlinear computation, and intelligent decision-making, as illustrated in Figure 3.10.

One of the most appealing aspects of silicon nanowire devices is their potential to combine sensing and computation within the same physical device. Rather than seperation of sensing, memory, and information processing into different components, these devices may enable several functions to be worked within a single nanoscale structure. This combination, makes silicon nanowires an interesting candidate for future hardware systems inspired by biological information processing.

Chapter 4

Hardware-Aware Modeling of Analog In-Memory Neural Networks Using CrossSim

4.1 Analog In-Memory Computing (AIMC) Fundamentals

Modern ANNs have become the dominant computational paradigm for machine learning applications, including image classification, speech recognition, natural language processing, and scientific data analysis. Although neural network architectures may vary considerably in complexity, their underlying mathematical operations are remarkably simple. At their core, neural networks repeatedly perform matrix-vector multiplications (MVMs) followed by nonlinear activation functions [58].

A neural network consists of multiple interconnected layers of artificial neurons. Each neuron receives a set of input values, multiplies them by trainable weights, then adds a bias term, and then applies a nonlinear activation function. Mathematically, the operation of a single neural network layer can be expressed as

$$\mathbf{y} = f(\mathbf{W}\mathbf{x} + \mathbf{b})$$

where $\mathbf{x}$ is the input vector, $\mathbf{W}$ is the weight matrix, $\mathbf{b}$ is the bias vector, $f(\cdot)$ represents the activation function, and $\mathbf{y}$ is the output vector. The activation function can be Rectified Linear Unit (ReLU), sigmoid function, hyperbolic tangent, or another nonlinear function.

From a computational perspective, neural network execution can be divided into two different main phases: inference phase and training phase. During inference, a trained network processes new input data to generate predictions. The input propagates through successive layers according to

$$\mathbf{x}^{(1)} = \mathbf{x}$$

$$\mathbf{x}^{(l+1)} = f\left(\mathbf{W}^{(l)}\mathbf{x}^{(l)} + \mathbf{b}^{(l)}\right)$$

until the final output layer produces the predicted result $\hat{\mathbf{y}}$.

At each layer, three operations are performed:

1. Matrix-vector multiplication (MVM):

$$\mathbf{z} = \mathbf{W}\mathbf{x}$$

2. Bias addition:

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

3. Nonlinear activation:

$$\mathbf{y} = f(\mathbf{z})$$

Among these operations, MVM dominates the computational workload. Numerous studies have shown that the majority of inference time and energy consumption in neural network accelerators originates from repeated execution of these MVM operations. Consequently, accelerating MVM is one of the primary objectives of modern AI hardware design [59].

Training introduces an additional computational stage in which the network parameters are iteratively optimized to minimize prediction error. A forward pass first computes the network output. A loss function then quantifies the discrepancy

between predicted and target values. For example, the mean squared error (MSE) loss is given by

$$L = \frac{1}{2} \|\hat{\mathbf{y}} - \mathbf{y}\|^2$$

The gradients of loss function are calculated using backpropagation and the chain rule with respect to the network parameters.

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial W}$$

allowing the weights to be updated through gradient descent,

$$W = W - \eta \frac{\partial L}{\partial W}$$

where η denotes the learning rate. Although training involves additional computations, MVM remains the dominant operation during both forward and backward propagation.

Traditionally, neural networks are executed on digital processors such as CPUs and GPUs. In these systems, the weight matrices are stored in memory while arithmetic operations are performed within separate computational units. Consequently, large amounts of data must continuously move between memory and processing cores. This repeated transfer of weights and activations consumes substantial energy and creates a bottleneck commonly referred to as the *von Neumann bottleneck* [60]. As neural network models continue to increase in size, data movement often becomes more expensive than the mathematical operations themselves.

AIMC has emerged as a promising alternative architecture for overcoming this limitation. Instead of physically moving weight values between memory and processors, AIMC performs computation directly within the memory array where the weights are stored. This approach exploits the intrinsic electrical properties of memory devices to execute MVM in parallel and with significantly reduced data movement [61].

Figure 4.1 illustrates the fundamental relationship between the mathematical MVM used in neural networks and its physical realization in a resistive crossbar array. In the mathematical domain, the output vector is obtained through multiplication of a

weight matrix and an input vector. In the hardware domain, the matrix elements are represented by device conductances while the input values are encoded as voltages applied to the rows of the crossbar. The resulting column currents naturally perform the weighted summation required for neural network computation.

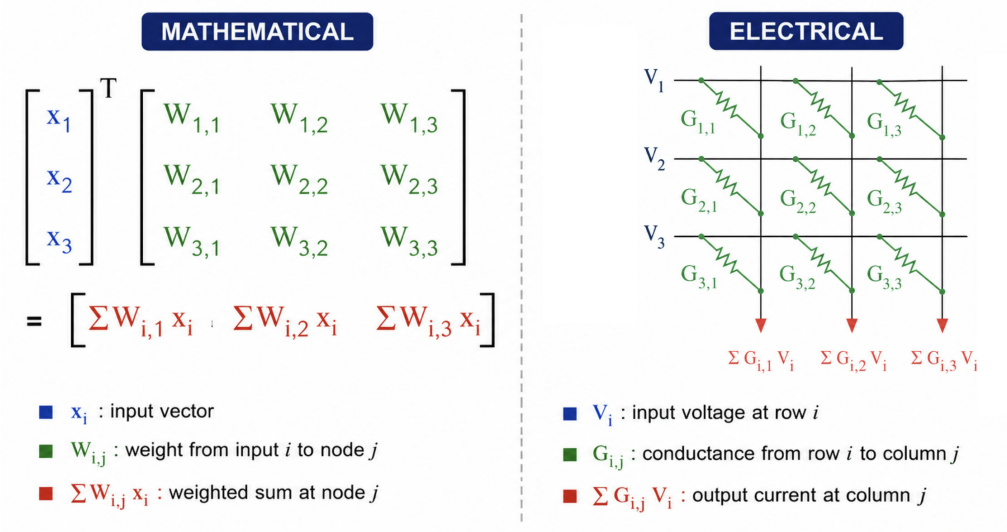


Fig. 4.1 Conceptual correspondence between mathematical MVM and its implementation in a resistive crossbar array. Neural network weights are mapped to conductances (G_{ij}), input values are encoded as voltages (V_i), and output currents represent the accumulated dot products.

The fundamental computational element of many AIMC systems is the resistive crossbar array. In such architectures, memory devices are arranged at the intersections of horizontal and vertical metal lines. The conductance of each memory cell represents a neural network weight, while input values are encoded as voltages applied to the rows of the array. According to Ohm's law,

$$I = GV$$

where G is the device conductance and V is the applied voltage. Each memory cell therefore performs an analog multiplication between the stored weight and the input signal. Simultaneously, Kirchhoff's current law causes all currents flowing into a column to accumulate naturally, producing the weighted summation required for the dot-product operation,

$$I_j = \sum_i G_{ij} V_i$$

which directly corresponds to the MVM operation performed in neural networks.

As a result, an entire MVM can be executed in a single physical operation rather than through millions of sequential digital multiply-and-accumulate instructions. This parallel analog computation provides the primary source of the energy-efficiency advantages associated with in-memory computing.

In practical systems, the outputs of the crossbar are analog currents and must therefore be converted back into digital values using analog-to-digital converters (ADCs). Likewise, digital inputs are transformed into analog voltages using digital-to-analog converters (DACs). The overall inference pipeline can therefore be summarized as

Digital Input $\rightarrow$ DAC $\rightarrow$ Crossbar MVM $\rightarrow$ ADC $\rightarrow$ Activation Function $\rightarrow$ Next Layer

This architecture enables the computationally intensive MVM to be performed directly inside the memory hardware while preserving compatibility with conventional digital neural network processing.

The potential benefits of AIMC are substantial. Since MVM occurs through the underlying device physics rather than sequential digital instructions, AIMC can achieve significantly higher computational parallelism, lower energy consumption, and reduced latency compared with conventional digital accelerators. These advantages have motivated extensive research into resistive memories, phase-change memories, floating-gate devices, ferroelectric transistors, and memristive technologies as candidate hardware platforms for future neuromorphic and AI computing systems [62].

However, the discussion presented thus far assumes an ideal analog computing system. In practice, every stage of the analog inference pipeline introduces non-idealities that can affect computational accuracy. These imperfections originate from the memory devices themselves, the crossbar interconnects, and the peripheral circuitry used to interface analog and digital domains.

Figure 4.2 summarizes several important sources of error in AIMC systems. At the device level, conductance variability, programming uncertainty, electrical noise, nonlinear switching behavior, and long-term conductance drift can cause the stored weights to deviate from their intended values. At the array level, wire resistance introduces parasitic voltage drops (IR drops), causing the effective voltages and currents within the crossbar to differ from ideal theoretical predictions. Additional errors comes from peripheral circuits such as DACs, ADCs, amplifiers, and sensing circuitry, which introduce quantization effects, noise, finite resolution, and signal distortion.

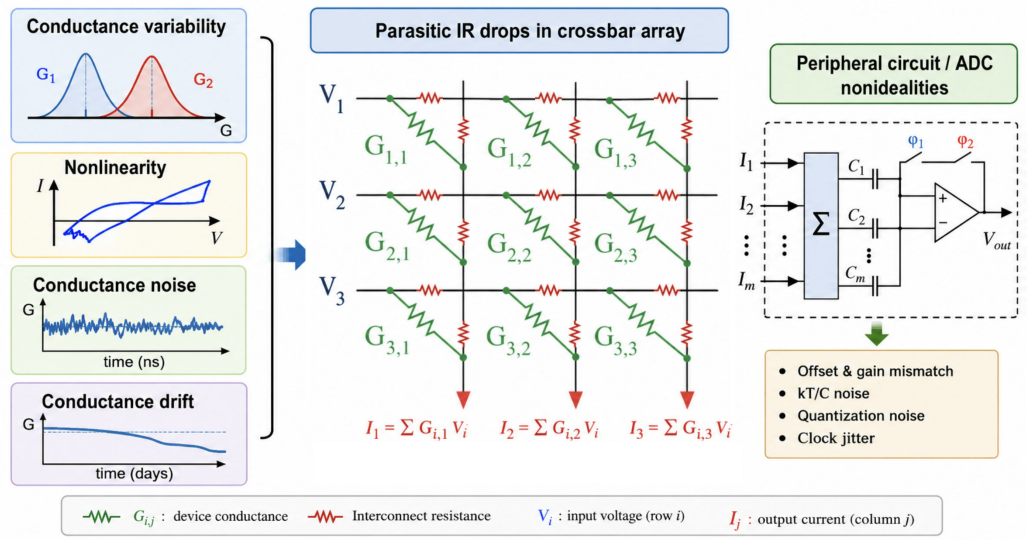


Fig. 4.2 Representative non-idealities in AIMC systems, including conductance variability, device noise, conductance drift, nonlinear device behavior, parasitic interconnect resistance, and imperfections in peripheral circuits such as ADCs and DACs.

As a result, the computation carried out by a physical crossbar array is not identical to the ideal MVM which is in software-based neural network models. Non-idealities originating from the device, circuit, and system levels can introduce errors that accumulate throughout the computation and ultimately affect inference accuracy. Understanding the impact of these effects is very important part of neuromorphic hardware design.

Evaluating such behavior requires simulation tools that account for the characteristics of real hardware. Instead of assuming ideal mathematical operations, hardware-aware frameworks incorporate the effects of physical non-idealities and allow their influence on neural network performance to be studied. CrossSim is one

such framework, providing a link between experimentally measured device behavior and the resulting performance of AI models at the system level [9].

4.2 CrossSim Simulation Framework

CrossSim is a hardware-aware simulation framework developed to evaluate the impact of analog hardware non-idealities on neural network inference accuracy [9]. Unlike conventional hardware simulators that focus primarily on estimating metrics such as energy consumption, chip area, or computational throughput, CrossSim is designed to answer a fundamentally different question: *How does the accuracy of a neural network change when inference is executed on a realistic analog computing system instead of an ideal digital processor?*

CrossSim simulate analog neural network inference through a hierarchy of layers and it connects device-level properties to system-level behavior. As illustrated in Figure 4.3, the framework combines four essential inputs:

1. A trained neural network model which includes the network topology and weight parameters.
2. A set of hardware configuration parameters that define the analog accelerator architecture which include data mapping strategies, conductance representations, ADC specifications, and activation ranges.
3. Experimentally characterized device data describing analog memory behavior, particularly programming variability, read noise, and other device non-idealities.
4. A dataset consisting of input samples and corresponding labels used to evaluate inference performance.

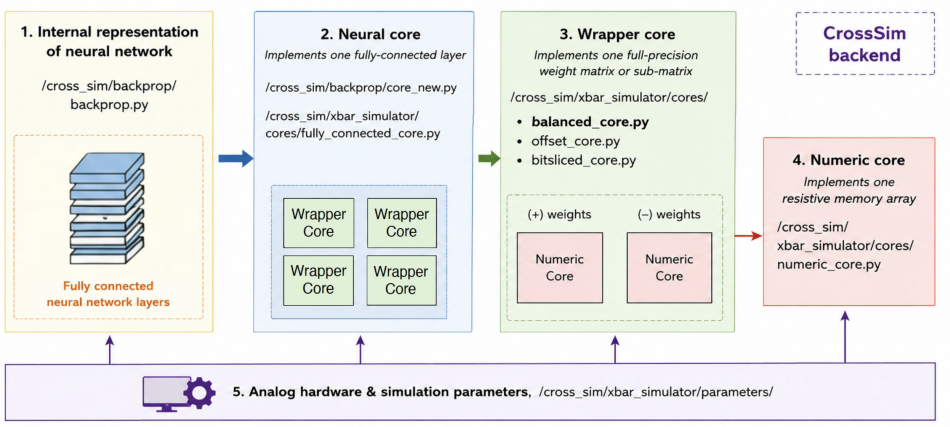


Fig. 4.3 Structure of the CrossSim inference framework and interaction between neural network models, hardware configuration parameters, device characterization data, and inference datasets.

In CrossSim, the analog MVM operation follows the resistive-crossbar principles described in Section 4.1. This section focuses on how CrossSim organizes trained neural-network weights, hardware parameters, device non-idealities, and inference data into a simulation workflow.

Before inference can be performed, the trained neural network weights must be mapped onto the analog hardware representation. This process starts with quantization of the weights to the target resolution and then mapping them to the conductance states of the memory cells. In CrossSim, the resulting conductance matrices are implemented through analog MVM cores that emulate the memory arrays and the surrounding peripheral circuits used during computation.

A single neural network weight matrix can be partitioned into multiple analog cores depending on hardware constraints and architectural choices. Partitioning can occur for several reasons. First, differential representations may be employed to encode positive and negative weight values using separate positive and negative conductance arrays. Second, digital weights may be divided into multiple bit slices, with each slice stored in a separate array. Third, large weight matrices may be spatially partitioned across several smaller arrays to satisfy practical crossbar size limitations [9].

CrossSim implements a hierarchical architecture composed of several abstraction layers. At the highest level, the *Neural Core* represents a complete neural network layer, such as a fully connected or convolutional layer. Each Neural Core contains one

or more Wrapper Cores that collectively represent the full weight matrix associated with that layer.

The *Wrapper Core* manages the representation of a complete section of the weight matrix at full numerical precision. It contains all positive and negative weight components as well as all significance bits associated with the stored values. Depending on the selected mapping scheme, a Wrapper Core may be subdivided into multiple lower-level computational units.

Numeric Core is at the lowest hardware abstraction level which represents a single resistive memory array containing strictly positive conductance values. Device-level and circuit-level analog effects are also modeled inside the Numeric Core which includes programming variability, read noise, conductance drift, parasitic resistance, ADC quantization, and other hardware non-idealities. Consequently, the Numeric Core serves as the primary location where experimentally characterized device behavior influences neural network inference accuracy.

To manage this multi-level simulation environment, CrossSim employs a collection of parameter objects that define the behavior of the framework at each abstraction level. These parameters specify device characteristics, array configurations, peripheral circuitry models, data mapping strategies, quantization settings, and system-level architectural constraints. Through this hierarchical organization, CrossSim provides a comprehensive framework for evaluating how device-level non-idealities propagate through analog hardware and ultimately affect neural network inference performance.

The overall workflow of the CrossSim inference framework is illustrated in Figure 4.4. After a trained neural network is provided, CrossSim maps its weights into hardware-compatible conductance representations and distributes them across one or more analog MVM cores according to the selected architecture, precision, and partitioning settings.

Within each analog MVM core, CrossSim applies the selected device, array, and peripheral-circuit models to the mapped conductance matrix. Non idealities such as programming errors, conductance drift, read noise, ADC quantization, and parasitic resistance are incorporated during the simulated inference process. The resulting outputs are then digitized, aggregated when multiple cores are used, combined with bias terms, and passed to the next neural-network layer.

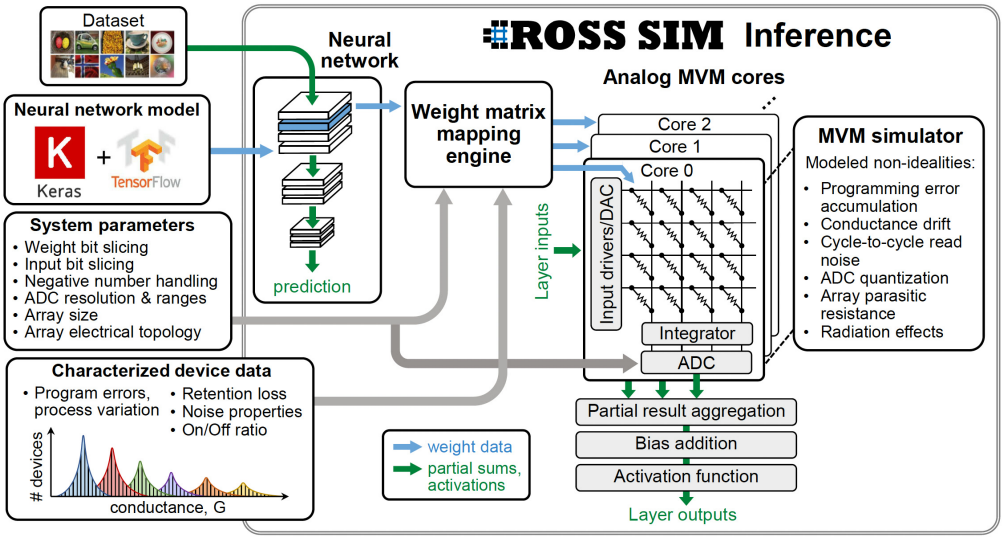


Fig. 4.4 Overview of the CrossSim inference framework. A trained neural network, hardware configuration parameters, characterized device data, and inference datasets are combined to simulate analog MVM operations while incorporating realistic device- and circuit-level non-idealities [9].

4.2.1 Weight Resolution and Quantization

Weight quantization determines the numerical precision available for representing neural network weights before they are mapped to device conductances. In CrossSim, the precision of this representation is controlled by the parameter `weight_bits`. When `weight_bits = 0`, no quantization is applied and the original floating-point weights are directly mapped to conductance values. For nonzero values, the weights are discretized into a limited number of levels, reducing the precision with which they can be represented on the hardware.

The range over which quantization is applied is determined by the parameter `weight_percentile`. This parameter specifies the portion of the weight distribution considered during the quantization process and therefore influences how the available quantization levels are allocated across the network weights. Smaller values increase the effective resolution at the expense of clipping extreme weights, whereas larger values reduce clipping but may underutilize the available conductance range.

4.2.2 Bias Weight Implementation

Most neural network layers include a bias vector that is added to the output of the MVM operation. CrossSim supports both analog and digital implementations of bias addition with `digital_bias` parameter.

In the digital approach, the bias values are stored separately from the crossbar arrays and then after analog-to-digital conversion added to the outputs. Since this addition is performed digitally, it does not introduce any additional analog non-idealities.

In the analog approach, the bias values are embedded directly within the memory array as extra rows. These rows are supplied with a constant input value of one, allowing the bias contribution to be accumulated as part of the MVM itself. As a result, both the weights and biases are processed within the same crossbar computation.

Because digital bias addition avoids analog programming errors, noise, and drift effects, it generally provides slightly higher inference accuracy than the analog alternative.The two bias implementation strategies are illustrated in Figure 4.5.

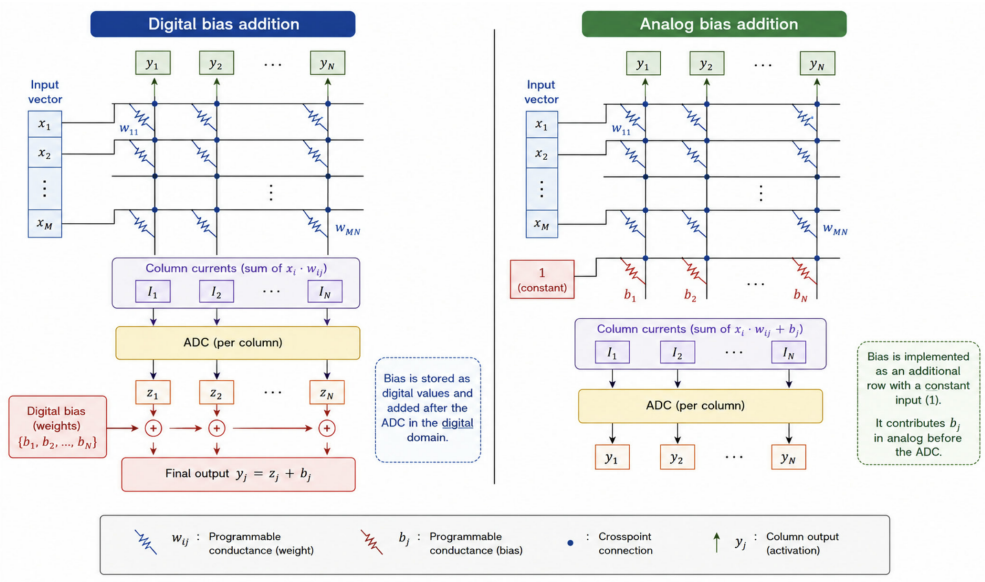


Fig. 4.5 Comparison between digital and analog bias implementation.

4.2.3 Negative Number Representation

One of the fundamental challenges in AIMC is the representation of signed neural network weights using memory devices whose conductance values are strictly positive. CrossSim supports two primary approaches for representing negative numbers: differential-cell encoding (BALANCED) and offset subtraction (OFFSET).

Differential Cells (BALANCED)

In differential-cell approach, each weight is described by two non-negative components,

$$W = W^{+} - W^{-},$$

Both components can then be mapped directly to physical conductances, while the effective signed weight is obtained from the difference between the two values.

CrossSim has two approaches for implementing differential-cell mappings, called `one_sided` and `two_sided`. For inference studies, the `one_sided` configuration is generally preferred. In this approach, the weight magnitude is stored in only one of the two conductance cells, while the other remains fixed at the minimum conductance value, G_{min}. Positive weights are represented using the positive cell, whereas negative weights are assigned to the negative cell.

A weight value of zero is represented by setting both devices to G_{min}. Because two devices are used to represent a single weight, the required conductance resolution per device is reduced by approximately one bit compared with the overall weight resolution.

Offset Subtraction (OFFSET)

Offset subtraction represents signed weights by introducing a constant offset:

$$W = W_{\text{prog}} - W_{\text{offset}},$$

where W_{prog} is strictly non-negative and therefore directly mappable to device conductance values.

The offset is selected such that a weight value of zero corresponds to a conductance value near the middle of the available conductance range. After the MVM operation is completed, the offset contribution is removed digitally or through additional arithmetic operations.

Compared with differential encoding, offset subtraction requires only one memory device per weight. However, it typically utilizes a larger portion of the conductance range to represent the offset and may exhibit different sensitivity to device non-idealities.Figure 4.6 compares the differential-cell and offset-subtraction approaches implemented in CrossSim.

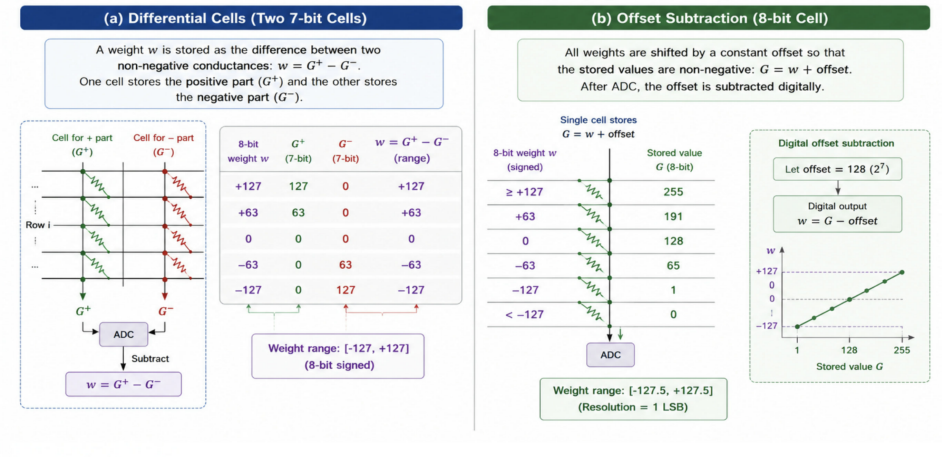


Fig. 4.6 Representation of signed neural network weights using (a) differential-cell encoding and (b) offset subtraction.

4.2.4 Weight Bit Slicing

Weight bit slicing is used when the desired weight precision exceeds the conductance resolution available in a single memory device. In this approach, each weight is decomposed into multiple lower-precision slices that are stored in separate crossbar arrays.

Each slice performs an independent analog MVM operation. The resulting partial outputs are subsequently combined using a digital shift-and-add procedure to reconstruct the full-precision result.

In CrossSim, the number of slices is controlled by the parameter `Nslices`. Setting `Nslices=1` disables bit slicing, whereas larger values increase representational precision at the expense of additional memory arrays and peripheral circuitry.An example of bit-sliced weight representation is shown in Figure 4.7.

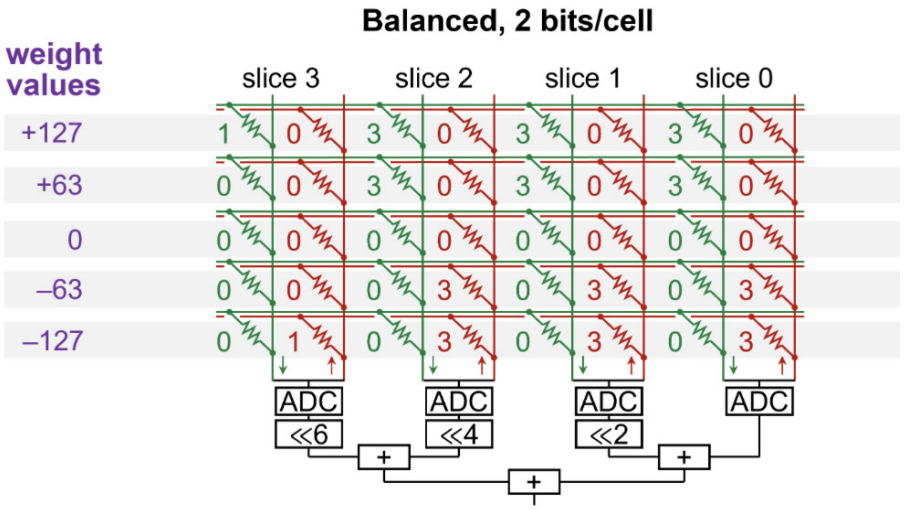


Fig. 4.7 Example of balanced differential encoding with weight bit slicing. An 8-bit signed weight is decomposed into multiple lower-resolution slices, each stored in a separate conductance array. The outputs of individual slices are combined through digital shift-and-add reconstruction [9].

4.3 Device Parameters

4.3.1 Modeling Device-Level Non-Idealities

The accuracy of AIMC systems is strongly influenced by the physical characteristics of the underlying memory devices. Unlike digital hardware, where arithmetic operations can be performed with extremely high precision, analog memory arrays are subject to device variability, programming inaccuracies, conductance drift, and noise. Consequently, realistic evaluation of neural network inference requires these effects to be incorporated into the simulation framework.

CrossSim provides a comprehensive set of device-level parameters that model the impact of memory-device non-idealities on inference accuracy. These effects

can be introduced at multiple stages of the simulation flow, including during the mapping of neural network weights to conductance values, immediately after conductance programming, and during MVM operations. This hierarchical modeling approach allows CrossSim to capture the propagation of device-level errors through the complete inference pipeline.

The overall locations at which device-level errors can be introduced within the CrossSim workflow are illustrated in Figure 4.8. Depending on the selected configuration, non-idealities may affect the programmed conductance values, the stored conductance states over time, or the readout process during analog computation.

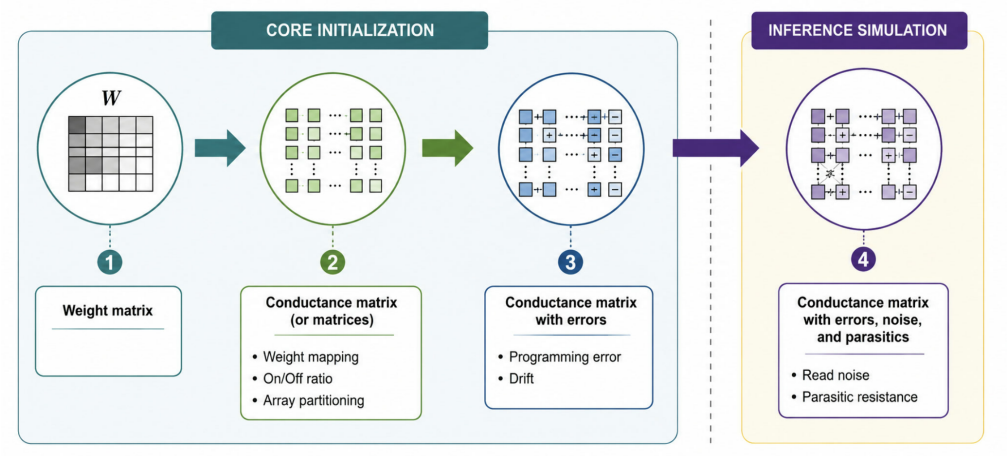


Fig. 4.8 Stages within the CrossSim inference workflow where device-level non-idealities can be incorporated. Device errors may be introduced during weight-to-conductance mapping, after conductance programming, or dynamically during analog MVM operations

4.3.2 Conductance On/Off Ratio

The dynamic range of a memory device is characterized by its conductance On/Off ratio, defined as

$$\text{On/Off Ratio} = \frac{G_{\max}}{G_{\min}},$$

where $G_{\max}$ and $G_{\min}$ denote the maximum and minimum programmable conductance states, respectively.

In CrossSim, the On/Off ratio is controlled through the parameter `on_off_ratio`. This parameter determines the minimum achievable conductance that can be represented within a memory array and therefore influences the range over which neural network weights can be mapped.

4.3.3 Conductance Programming Errors

CrossSim includes a generic statistical model for conductance programming errors that can be used when device-specific experimental data are unavailable [9]. Although not tied to a particular memory technology, the model can be used to investigate the impact of programming variability on inference accuracy and overall system performance. This functionality is enabled through the setting `error_model = ''alpha''`.

Two variants of the model are available: a *state-independent model*, in which all conductance states are assumed to experience the same level of programming variability, and a *state-proportional model*, where the magnitude of the error increases with conductance. In both cases, programming errors are generated from zero-mean Gaussian distributions, and the overall level of variability is controlled by the parameter `alpha_error`. Figure 4.9 illustrates the two generic programming-error models implemented in CrossSim.

Although these generic error models do not represent the behavior of every memory technology, they provide a convenient framework for studying the impact of conductance variability on neural network inference. Experimental studies have reported memory devices whose behavior resembles either state-independent or state-proportional programming errors [10, 63]. The two models can affect network accuracy differently, particularly because the state-proportional model depends on both the neural network weight distribution and the selected conductance mapping strategy. As a result, the interaction between weight mapping and programming variability is important for evaluattion of AIMC systems.

Custom Programming Error Models

Although the state-independent and state-proportional models are useful for exploring the general impact of programming variability, they provide only a simplified

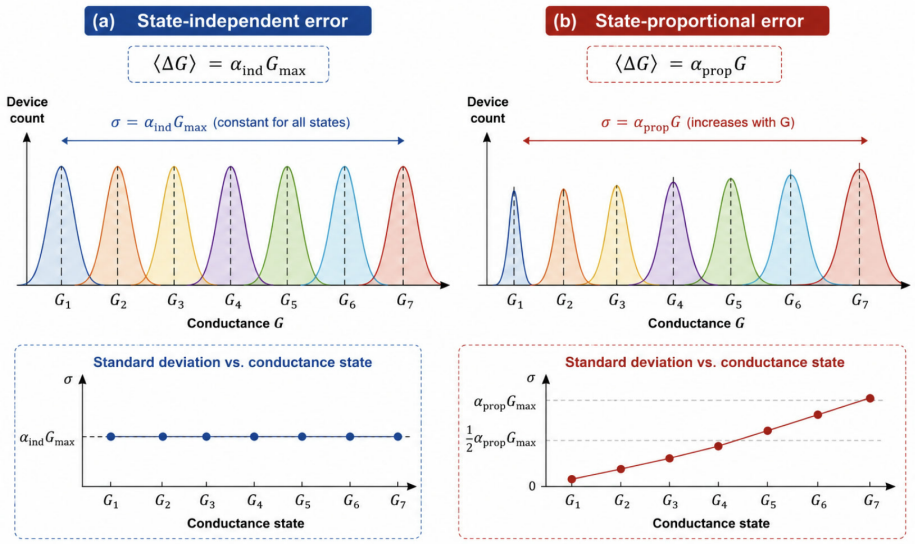


Fig. 4.9 Generic conductance programming error models implemented in CrossSim. (a) State-independent error model. (b) State-proportional error model.

description of device behavior. In many practical memory technologies, programming errors depend on the conductance state in more complex ways and may comes from several underlying sources of variability. As a result, experimental devices often need more detailed models to accurately capture their programming characteristics.

To address this limitation, CrossSim supports custom conductance programming error models derived from experimental measurements or lower-level device simulations. A custom error model is implemented through a user-defined Python function that specifies how programming variability depends on the target conductance value. During array initialization, the function is applied to the conductance matrix before inference begins, allowing arbitrary state-dependent error behavior to be incorporated into the simulation [9].

Several example device models are distributed with CrossSim, including PCM [11] , Silicon-Oxide-Nitride-Oxide-Silicon (SONOS) charge-trapping memory [10], and HfO_2-based RRAM [12]. These models are derived from experimentally reported device characteristics and provide realistic examples of how programming variability can depend on conductance. Unlike the generic models, which assume a simple linear or constant relationship between conductance and error magnitude, custom models can reproduce arbitrary conductance-dependent error profiles observed in real devices.

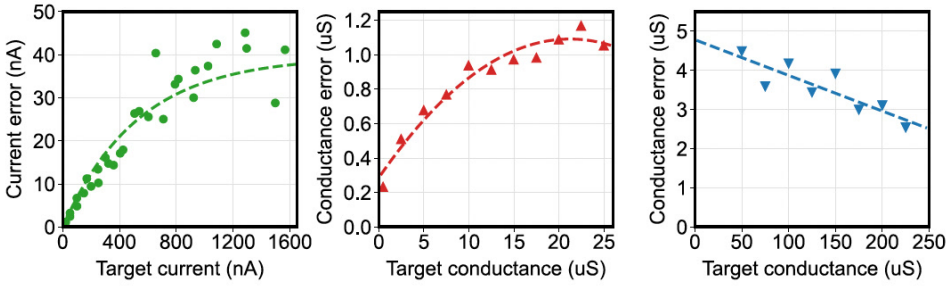


Fig. 4.10 Error characteristics of three exemplary devices provided with CrossSim. (a) SONOS device from Xiao et al. with a saturating exponential fit to the measured points [10], (b) PCM device from Joshi et al. with a quadratic fit [11], (c) ReRAM device from Milo et al. with a linear fit [12].

Examples of custom conductance-error profiles included in CrossSim are shown in Figure 4.10. These profiles demonstrate that programming-error characteristics can vary substantially across memory technologies.

4.3.4 Conductance Drift

Conductance drift refers to the gradual change in a device's programmed conductance state after programming. Physically, drift may arise from structural relaxation mechanisms in PCM and resistive switching devices, or from charge leakage in flash-based memory technologies. Although drift is often discussed in relation to long-term retention, it can also affect analog neural network accelerators, where small conductance changes across many memory cells may collectively influence MVM accuracy.

CrossSim models drift through the parameter `t_drift`, which specifies the elapsed time after programming and it is in units of days. When `t_drift` is greater than zero, a user-selected drift model is applied to the programmed conductance matrix before inference.

Unlike programming-error models, CrossSim does not provide a generic drift model because drift behavior is highly dependent on the underlying device physics. Instead, technology-specific drift models are provided for experimentally characterized devices. These include a SONOS charge-trapping memory model based on

measured conductance evolution over time and a PCM drift model based on the analytical formulation proposed by Joshi *et al.* [11].

4.3.5 Conductance Read Noise

Read noise is another important non-ideality in analog memory devices. Unlike programming errors and conductance drift, which remain fixed after programming, read noise is dynamic and is re-sampled during every MVM operation. Because of that, the conductance values used during inference fluctuate continuously, introducing variability from one computation to the next.

CrossSim has both generic and custom approaches for modeling read noise. The generic implementation is enabled with `noise_model = ‘‘alpha’’` and is available in two forms: a *state-independent* model, where the noise level is the same across all conductance states, and a *state-proportional* model, where the noise magnitude increases with conductance. In both cases, random conductance fluctuations are sampled from Gaussian distributions during each MVM operation.

For applications which need more realistic model of device behavior, CrossSim also supports custom read-noise models. This allows experimentally measured noise roperties to be incorporated directly into the simulation framework. As a result, the impact of read noise on neural network inference can be evaluated using models that more closely reflect the behavior of a specific memory technology.

4.4 Array Parameters

4.4.1 Array Size and Matrix Partitioning

In practical analog accelerators, the dimensions of a resistive crossbar array are constrained by fabrication limitations, peripheral-circuit overhead, programming reliability, and the accumulation of analog non-idealities. Although larger arrays improve storage density and computational throughput, they are generally more susceptible to programming variability, read noise accumulation, and parasitic voltage drops. Consequently, large neural-network weight matrices are divided across multiple physical memory arrays.

CrossSim supports matrix partitioning by dividing a neural-network weight matrix into multiple MVM arrays. Partitioning is performed along the dimension over which cell currents are accumulated to form a dot product. The maximum number of rows assigned to a single partition is controlled by the parameter `NrowsMax`. Whenever the dimensions of the weight matrix exceed this limit, the matrix is automatically decomposed into multiple partitions.

The partitioning procedure is illustrated in Figure 4.11. Each partition independently performs an analog MVM operation and produces a partial dot product. The outputs of the individual partitions are subsequently digitized by separate ADCs and combined digitally to reconstruct the full MVM result.

If the total number of rows N_1 is less than or equal to `NrowsMax`, a single array is used. Otherwise, the rows are distributed as evenly as possible among the required partitions. When digital bias addition is disabled, the bias row is included in the partitioning process. It should be noted that a matrix partition may itself contain multiple arrays corresponding to different weight bit slices or positive/negative weight representations.

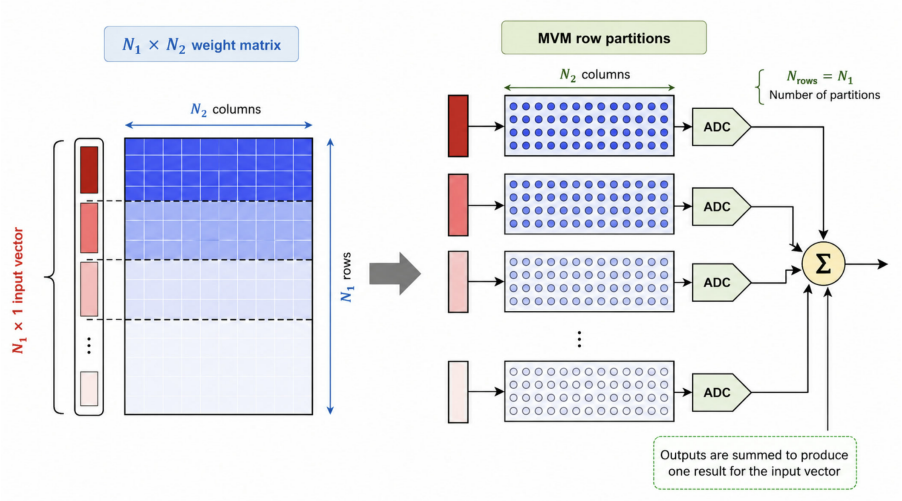


Fig. 4.11 Matrix partitioning in CrossSim. A large weight matrix is divided into multiple analog MVM arrays. Each partition computes a partial dot product that is digitized independently and subsequently aggregated to form the complete output.

The matrix-partitioning strategy used by CrossSim is illustrated in Figure 4.11.

4.4.2 Parasitic Resistance

The metal interconnects that form the rows and columns of a crossbar array possess finite electrical resistance. As current flows through these interconnects, voltage drops occur along the array according to Ohm's law. These parasitic voltage drops alter the voltages experienced by individual memory cells and therefore distort the resulting MVM.

The magnitude of the parasitic effect depends on several factors, including the array dimensions, device conductance values, input vector distribution, weight-mapping strategy, and memory-cell topology. Because each additional row and column contributes both resistance and current flow, the resulting error generally increases superlinearly with array size.

Parasitic resistance is one of the principal factors limiting the practical dimensions of analog MVM arrays. Consequently, it strongly influences accelerator architecture and motivates the use of matrix partitioning in large-scale systems.

CrossSim incorporates parasitic resistance directly into the analog MVM simulation through a dedicated circuit solver. The parasitic model is evaluated independently for each matrix partition, each weight bit slice, and each input bit slice. This allows the simulator to capture the complex interactions between interconnect resistance, conductance distributions, and neural-network inputs.

4.5 ADC and Activation Quantization

Analog MVM produces continuous-valued output currents that must be converted into digital values before subsequent neural network operations can be performed. This conversion is carried out by an ADC, which represents one of the primary sources of numerical error in AIMC systems.In CrossSim, the behavior of the ADC is primarily determined by two parameters: the number of quantization bits, which sets the conversion resolution, and the operating range over which analog signals can be accurately represented. Together, these parameters define how the analog output of the crossbar array is converted into a digital value.

During this conversion process there are two main sources of error. Quantization error comes because a continuous analog signal must be represented using a finite

set of digital levels. Clipping error occurs when the signal falls outside the ADC operating range, causing values beyond the limits to be truncated to the nearest boundary. Both effects can influence the accuracy of MVM and, consequently, the performance of the neural network.

In addition to ADC quantization, CrossSim also supports quantization of neural network activations before they are applied to the next layer. Although these parameters are referred to as DAC parameters within the framework (`dac_bits` and `dac_ranges`), they do not necessarily correspond to a physical DAC. Instead, they define the numerical precision used to represent activation values entering subsequent MVM operations. Because activation quantization typically occurs after layer outputs have been aggregated and activation functions have been applied, its operating range is often different from the ADC range used in the preceding layer.

The quantization process used by CrossSim is deterministic and employs uniformly spaced quantization levels. The resolution of the ADC and activation quantizers is specified by the parameters `adc_bits` and `dac_bits`, respectively. If parameter is set to zero, the quantization stage is disabled. For a quantizer with resolution B bits, the number of available quantization levels is equal to 2^B. Signals that exceed the specified range are clipped to the maximum or minimum representable value.

Another feature of CrossSim is ability to define quantization parameters separately for each layer of a neural network. For this reason, different layers can be evaluated using different ADC resolutions and activation precisions rather than applying the same settings throughout the entire model.

The way in which analog signals are digitized in CrossSim depends on the hardware architecture being modeled. For example, when differential weight encoding is used, the positive and negative currents are combined in the analog domain before being passed to the ADC. As a result, the digitization process operates on the differential signal rather than on the individual current contributions.

Selection of appropriate ADC or activation range is a trade-off between clipping and quantization errors. A narrow range can reduce spacing between quantization levels and improves numerical precision, but in the other hand , increases the possibility of clipping. Conversely, a wider range reduces clipping but increases quantization error. To balance these competing effects, CrossSim provides a calibration procedure in which representative inference data are first collected without quantization

and then used to determine near-optimal quantization ranges for each layer. These calibrated ranges generally provide the highest inference accuracy for a given ADC or activation resolution.

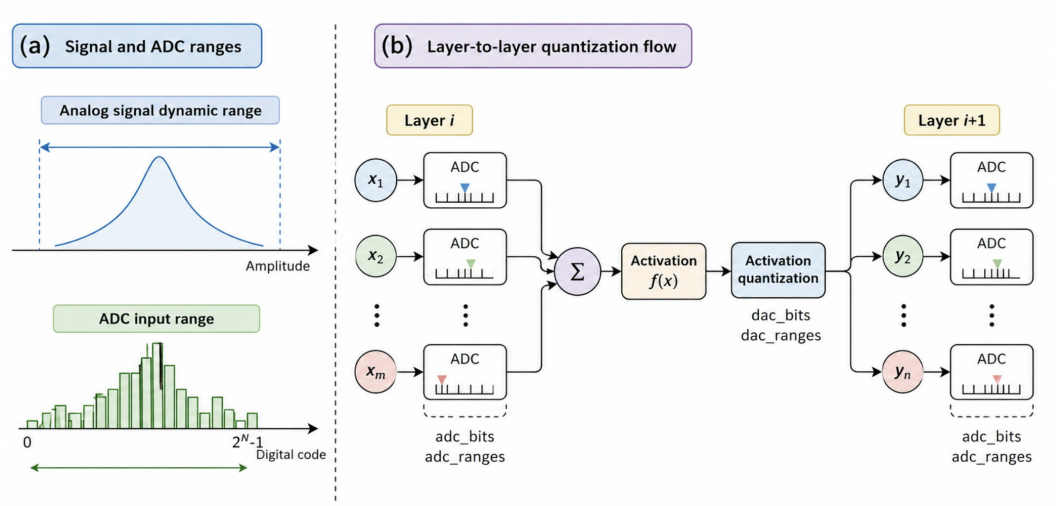


Fig. 4.12 ADC and activation quantization in CrossSim. (a) Analog signals are converted into discrete digital levels according to the ADC resolution and range, introducing quantization and clipping errors. (b) Quantization is applied both to analog MVM outputs through ADCs and to activation values before they are supplied to subsequent neural network layers.

The quantization process implemented in CrossSim is illustrated in Figure 4.12, showing both ADC conversion of analog MVM outputs and activation quantization between successive neural network layers.

Chapter 5

Silicon Nanowire Devices as Synaptic Elements for Neuromorphic Computing

5.1 Motivation for Synaptic Computing

As discussed in Section 3.2.6, oxide engineering enables silicon nanowire memristive devices to exhibit a range of switching behaviors that are potentially useful for neuromorphic computing applications. The functionality of these devices can vary considerably depending on the operating regime and the underlying material properties. In different conditions, they may support tasks such as synaptic weight storage, neuron-inspired activation, or time-dependent memory behavior. Among these applications, their use as artificial synapses in analog neural network accelerators is particularly important, as it enables computation to be performed directly within memory arrays, reducing the need for data movement between memory and processing units.

In ANNs, synaptic weights represent the learned parameters of the model and must be stored in memory during inference. AIMC architectures exploit programmable conductance states of memristive devices to directly encode these weights within crossbar arrays, allowing MVM operations to be performed efficiently in the

analog domain. However, as discussed in the previous chapter, analog hardware introduces numerous non idealities, including programming variability, conductance drift, read noise, quantization effects, and circuit-level imperfections. These non idealities can significantly affect inference accuracy and must be carefully investigated before a device can be considered suitable for neural network acceleration.

A main challenge comes from the difference between the precision of software trained neural network weights and the limited conductance resolution which is available in physical memory devices. Conventional neural network models are typically trained using high-precision numerical representations, whereas practical memristive devices can only store a finite number of conductance levels. Recent memristor technologies have demonstrated multi-level storage capabilities, enabling several bits of information to be represented within a single device. Nevertheless, the conductance resolution remains a critical factor that directly influences the achievable neural network accuracy [64–66].

For the silicon nanowire memristor investigated in this work, the electrical characterization presented in Section 3.2.3 revealed only two stable conductance states corresponding to the high-resistance state (HRS) and low-resistance state (LRS). Consequently, the device can effectively represent a single binary bit. While this characteristic simplifies device operation and fabrication, it introduces an important limitation for neural network applications. Quantization of all network weights directly to a binary representation would cause substantial loss of information and, consequently, a significant reduction in inference accuracy.

To address this challenge, there are two possible implementation strategies. The first approach used the original high-precision ANN model and represents each weight using bit slicing, as discussed in Section 4.2.4. In this method, a multi-bit weight is decomposed into multiple binary slices, each implemented using devices capable of storing only a single bit. Furthermore, because neural network weights contain both positive and negative values, separate positive and negative arrays are required, as described in Section 4.2.3. For example, representing an 8-bit signed weight requires eight binary slices for positive values and eight binary slices for negative values, resulting in a total of sixteen physical crossbar arrays per neural network layer. This approach allows high-precision weight representation but it has hardware complexity, larger area requirements, additional ADC and DAC resources, and higher system cost.

An alternative strategy is to redesign the neural network architecture itself to match the capabilities of the underlying hardware. Instead of storing high-precision weights, the network can be trained using a constrained weight set which can be consist only of the values {-1,0,+1}. Networks trained under this constraint are commonly referred to as TNNs [67, 68] . Although the reduced weight precision generally introduces some accuracy degradation compared with a full-precision ANN of identical architecture, the hardware implementation becomes substantially more efficient. A positive crossbar array can represent positive weights and zero values, while a corresponding negative array represents negative weights and zero values. Consequently, only two crossbar arrays are required per layer, significantly reducing hardware resources, circuit complexity, and power consumption.

The trade-off between these two approaches forms the central motivation behind this chapter. On one hand, bit-sliced ANN implementations maintain high numerical precision but require high hardware overhead. On the other hand, TNNs simplify the hardware implementation but may sacrifice accuracy. Finding the balance between accuracy and hardware efficiency is important when evaluating binary silicon nanowire memristors for neuromorphic computing applications.

This chapter first introduces the principles of TNNs and discusses the modifications needed for training and backpropagation under ternary weight constraints. Then, fully connected ANN and TNN models are trained and evaluated digitally using the MNIST dataset to establish baseline performance. Subsequently, CrossSim is employed to simulate the hardware implementation of both approaches within analog crossbar arrays, enabling direct comparison between ideal digital inference and ideal analog inference.

Following this baseline evaluation, a physics-informed model of the silicon nanowire memristor is incorporated into the simulation framework using experimentally measured device characteristics. Device level non idealities, including programming variability and conductance drift, are introduced based on measured data, while system level effects such as ADC quantization, DAC quantization, and interconnect parasitic resistance are also considered. The influence of each non ideality on neural network accuracy is investigated to identify the dominant sources of performance degradation.

Finally, device-aware training techniques are employed to recover hardware-induced errors. By incorporating realistic device characteristics directly into the

training process, the neural network can adapt to the underlying hardware constraints and recover a portion of the lost accuracy. This provides a comprehensive evaluation of the feasibility of using binary silicon nanowire memristors as synaptic elements in analog neural network accelerators and establishes a complete algorithm–hardware co-design framework spanning device physics, circuit implementation, and neural network performance.

5.2 Digital Neural Network Baseline

Before investigating hardware implementations, a fully digital neural network was established as a reference baseline. This baseline serves two purposes. First, it provides the maximum achievable accuracy under ideal software conditions. Second, it enables a detailed examination of the learned weight distributions that will later be mapped onto memristive crossbar arrays.

A fully connected ANN was implemented using Python [69] and the PyTorch framework [70]. The network architecture consisted of three linear layers with dimensions 784–256–128–10, corresponding to the flattened MNIST input image [71], two hidden layers, and the ten output classes. The MNIST dataset contains grayscale handwritten digit images with a resolution of 28×28 pixels and associated class labels ranging from 0 to 9.

To ensure reproducibility, the random seed was fixed at 42 throughout all experiments. The network was trained using a batch size of 16, a learning rate of 10^{-4}, and a total of 100 training epochs. Figure 5.1 presents the overall ANN architecture and training workflow employed throughout this chapter.

The trained network achieved accuracy of approximately 98% on the MNIST dataset. This accuracy provides a baseline against which the effects of quantization and hardware non-idealities are evaluated in the following sections. Consequently, any loss in accuracy observed during later simulations can be attributed primarily to hardware non-idealities rather than to the underlying network model.

To better understand the trained model, the distributions of the learned weights and biases were analyzed for each fully connected layer. Figure 5.2 presents the corresponding histograms. The first hidden layer (fc1) contains approximately

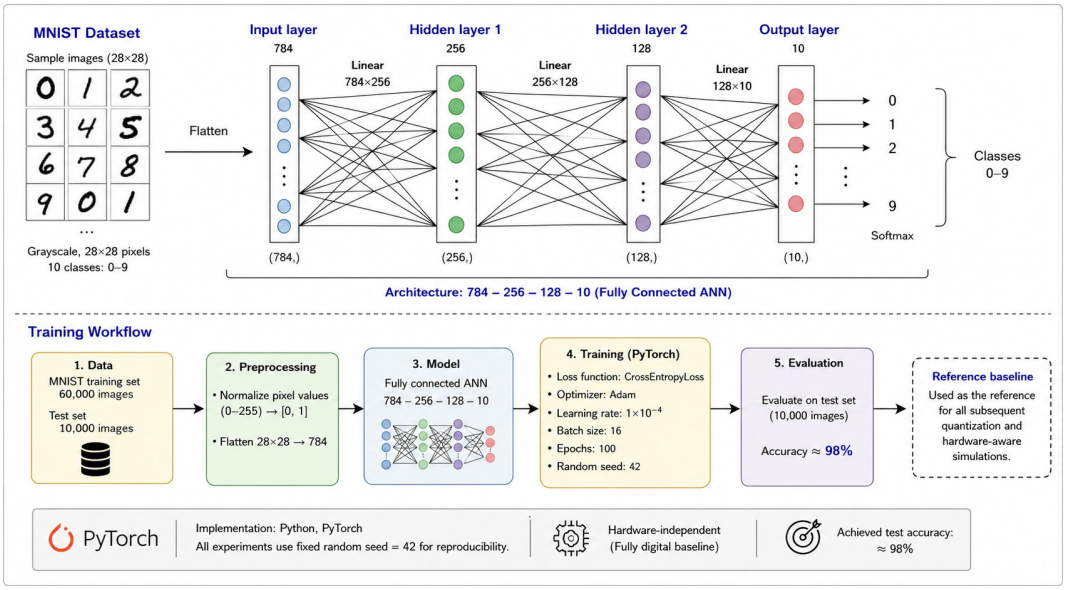


Fig. 5.1 Overview of the MNIST-based fully connected ANN implemented in PyTorch, showing the dataset, network architecture (784–256–128–10), training workflow, and the resulting digital baseline accuracy used for subsequent hardware-aware evaluations.

200,704 trainable weights, while the second and third layers contain 32,768 and 1,280 weights, respectively.

Several important observations can be made from these distributions. First, the weights of the first two layers exhibit approximately symmetric bell-shaped distributions centered near zero, resembling Gaussian distributions. Such behavior is commonly observed in trained neural networks and indicates that most weights possess relatively small magnitudes, while only a limited number contribute large positive or negative values. Second, the spread of the distributions increases progressively from fc1 to fc3. The majority of fc1 weights lie approximately within the range of -0.2 to 0.2, whereas fc2 weights extend roughly between -0.4 and 0.4. The output layer (fc3) exhibits the broadest distribution, with weights extending approximately from -0.6 to 0.5. This increase in weight magnitude suggests that deeper layers rely on stronger feature discrimination to perform final classification decisions.

The bias distributions exhibit a similar tendency to remain centered near zero, although they are generally less symmetric due to the substantially smaller number of parameters. Because biases contain only hundreds of parameters rather than

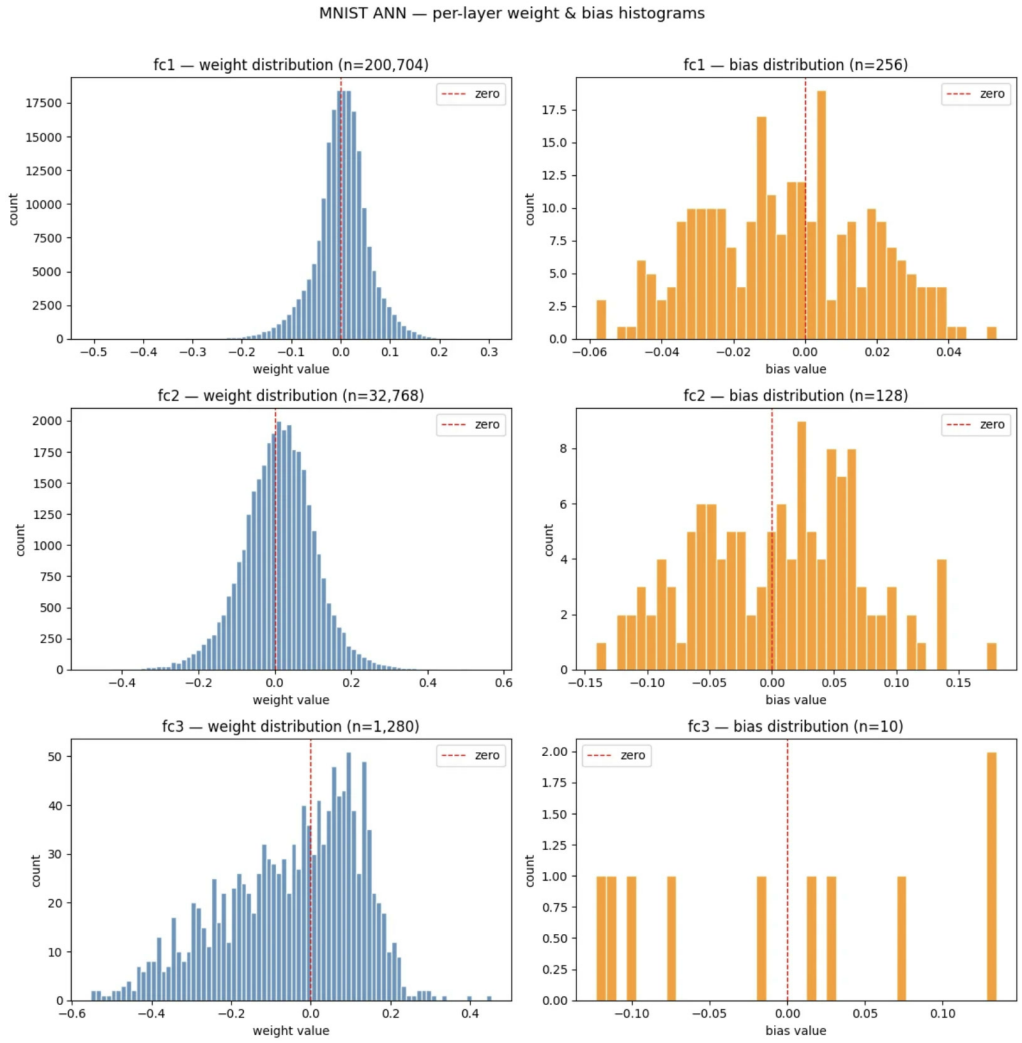


Fig. 5.2 Weight and bias distributions of the trained MNIST ANN. Histograms are shown for all three fully connected layers (fc1, fc2, and fc3). The dashed red line indicates the zero value.

thousands, statistical fluctuations become more visible, particularly in the output layer where only ten bias values are present.

Additional statistical information is summarized in Figure 5.3. Figure 5.3a presents the average absolute weight magnitude and average absolute bias magnitude for each layer. Both quantities increase progressively from fc1 to fc3, showing that later layers use stronger parameter values than earlier layers. Figure 5.3b shows the standard deviation of the weight distributions, which also increases across the network depth. This observation is consistent with the widening distributions observed in the histogram analysis.

Finally, Figure 5.3c visualizes representative weight matrices using heatmap representations. Positive and negative weights are clearly visible throughout the network, demonstrating that the learned parameters are distributed continuously over a wide numerical range rather than being concentrated at a small number of discrete levels. This characteristic is particularly important for neuromorphic hardware implementation because the underlying silicon nanowire memristor investigated in this work provides only two stable conductance states. Consequently, a direct mapping of these full-precision weights to the device is not possible without additional quantization or encoding strategies, motivating the approaches investigated in the following sections.

5.3 Ternary Neural Networks (TNNs)

TNNs are a type of quantized neural network in which the weights are restricted to three possible values, typically $\{-1, 0, +1\}$. By limiting the weight representation in this way, the network requires less memory and significantly less computational complexity compared with conventional full-precision models.

TNNs are appropriate for memristive crossbar systems when there is limitation on the number of conductance states which is reliable and accessible is often limited. Since only three weight levels need to be represented, the mapping between neural network weights and physical device states becomes considerably simpler, making ternary networks a practical option for low-resolution analog hardware.

From a neuromorphic hardware perspective, ternary weights offer an effective compromise between binary and full-precision neural networks. Binary neural

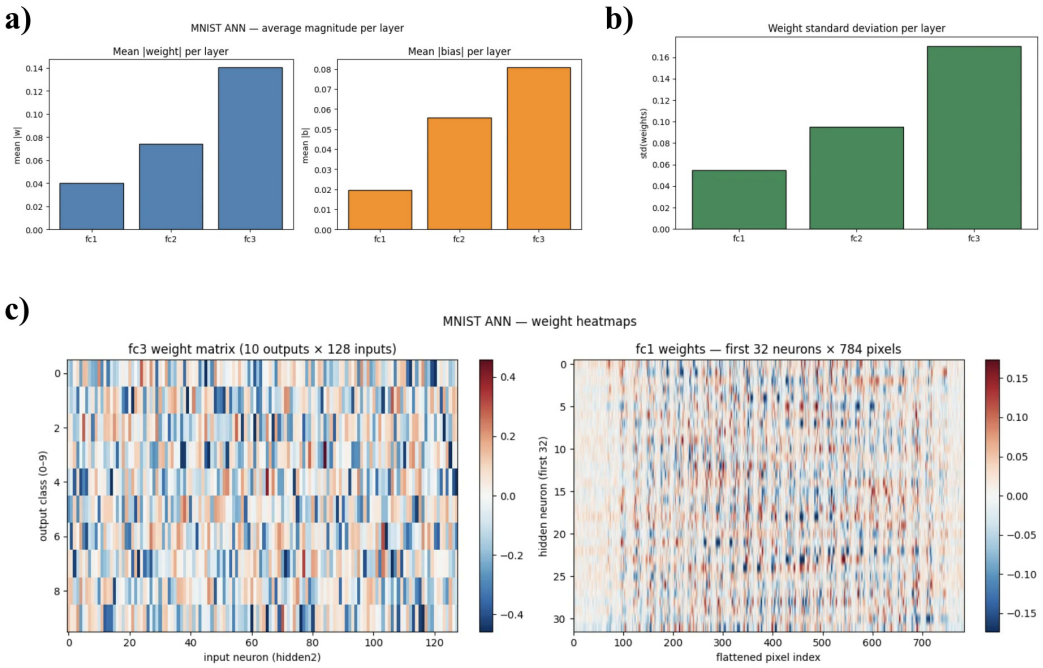


Fig. 5.3 Statistical analysis of the trained MNIST ANN. (a) Mean absolute weight and bias magnitude for each fully connected layer. (b) Standard deviation of the weight distributions. (c) Heatmap visualizations of representative weight matrices.

networks has maximum hardware simplicity but often suffer from significant information loss. Ternary networks introduce an additional zero state, enabling sparse weight representations while preserving a compact hardware mapping. For the silicon nanowire device investigated in this work, which exhibits two stable conductance states, the positive and negative ternary branches can be implemented using differential crossbar representation, while the zero state corresponds to the absence of an effective programmed contribution.

The ternary quantization function used in this work is defined as

$$w_t = \begin{cases} +1, & w > \Delta, \\ 0, & |w| \leq \Delta, \\ -1, & w < -\Delta, \end{cases} \tag{5.1}$$

where w is the latent full-precision weight, w_t is the ternary weight used during the forward pass, and Δ is the ternarization threshold. In this work, $\Delta = 0.05$ was used unless otherwise stated.

Although this quantization function is suitable for inference, it creates a major difficulty during training. The ternary function is discontinuous and piecewise constant; therefore, its derivative is zero almost everywhere and undefined at the threshold points:

$$\frac{\partial w_t}{\partial w} = 0 \quad \text{almost everywhere.} \tag{5.2}$$

As a result, direct backpropagation through the ternary function prevents meaningful weight updates. To address this limitation, surrogate-gradient methods are used. In this approach, the exact ternary function is applied during the forward pass, while an approximate derivative is used during the backward pass:

$$\frac{\partial \mathscr{L}}{\partial w} \approx \frac{\partial \mathscr{L}}{\partial w_t} \cdot g(w), \tag{5.3}$$

where $\mathscr{L}$ is the loss function and $g(w)$ is the selected surrogate-gradient function.

Surrogate-Gradient Formulations

Seven surrogate-gradient approaches were investigated in this work.

Plain Straight-Through Estimator (STE)

The Plain STE directly passes the upstream gradient through the ternary quantization function:

$$g_{\text{Plain}}(w) = 1. \tag{5.4}$$

Therefore,

$$\frac{\partial \mathscr{L}}{\partial w} \approx \frac{\partial \mathscr{L}}{\partial w_t}. \tag{5.5}$$

Clipped STE

The Clipped STE allows gradient propagation only when the latent weight lies within a predefined interval:

$$g_{\text{Clipped}}(w) = \begin{cases} 1, & |w| \leq 1, \\ 0, & |w| > 1. \end{cases} \tag{5.6}$$

This prevents very large latent weights from receiving further updates through the quantization function.

Scaled STE

The Scaled STE multiplies the upstream gradient by a constant factor s:

$$g_{\text{Scaled}}(w) = s. \tag{5.7}$$

Thus,

$$\frac{\partial \mathscr{L}}{\partial w} \approx s\frac{\partial \mathscr{L}}{\partial w_t}. \tag{5.8}$$

Different values of s were evaluated to study the effect of surrogate-gradient magnitude on training stability.

Bi-Real-Style Triangular Surrogate

The Bi-Real-style surrogate uses a triangular gradient profile:

$$g_{\mathrm{Tri}}(w) = \max(2 - 2|w|, 0). \tag{5.9}$$

This formulation gives the largest gradient near zero and gradually suppresses the gradient as the latent weight moves away from the central region.

Tanh/IR-Net-Style Surrogate

The Tanh/IR-Net-style surrogate uses the derivative of a scaled hyperbolic tangent function:

$$g_{\mathrm{Tanh}}(w) = \beta \left(1 - \tanh^2(\beta w)\right), \tag{5.10}$$

where β controls the sharpness of the surrogate approximation. Larger values of β produce a narrower and sharper gradient region.

Threshold-Window Surrogate

The threshold-window surrogate propagates gradients only near the positive and negative ternary transition thresholds:

$$g_{\mathrm{TW}}(w) = \begin{cases} 1, & |w - \Delta| \le r \quad \text{or} \quad |w + \Delta| \le r, \\ 0, & \text{otherwise}, \end{cases} \tag{5.11}$$

where r is the window width.

Smooth Threshold-Window Surrogate

The smooth threshold-window surrogate replaces the hard window with a continuous triangular profile around the ternary thresholds:

$$g_{\mathrm{STW}}(w) = \max\left[\max\left(1 - \frac{|w - \Delta|}{r}, 0\right), \max\left(1 - \frac{|w + \Delta|}{r}, 0\right)\right]. \tag{5.12}$$

This approach concentrates gradient flow near the ternary switching boundaries while avoiding abrupt discontinuities in the backward pass.

5.3.1 Optimization of TNN Backpropagation Surrogates

To evaluate the training behavior of different surrogate-gradient strategies, a custom `TernaryLinear` layer was implemented in PyTorch by extending the standard `nn.Linear` module. During the forward pass, the latent full-precision weights were converted into ternary weights using Equation 5.1. During the backward pass, surrogate-gradient function was used for derivative of the ternary quantizer.

The TNN architecture used for this study consisted of three fully connected ternary layers with dimensions 784–256–128–10. The MNIST dataset was used for training and evaluation. All models were trained using a batch size of 16, a learning rate of 10^{-4}, and 100 training epochs. The tested surrogate-gradient configurations are summarized in Table 5.1.

Table 5.1 Surrogate-gradient configurations evaluated for TNN training.

Surrogate Family	**Configuration**
Plain STE	Default formulation
Clipped STE	Default formulation
Bi-Real surrogate	Default formulation
Scaled STE	$s \in \{0.3, 0.5, 0.6, 0.65, 0.7, 0.75, 0.8, 0.9\}$
Tanh/IR-Net surrogate	$\beta \in \{1, 2, 3\}$
Threshold-window surrogate	$r \in \{0.3, 0.5, 0.7\}$
Smooth threshold-window surrogate	$r \in \{0.3, 0.5, 0.6, 0.7, 0.8, 0.9\}$

Figure 5.4 presents the comparative training behavior of the evaluated surrogate-gradient methods. The complete training-loss curves for all individual configurations are provided in Appendix A (Figure A.1 and A.2). Additional parameter studies, including the effect of the scaling factor in Scaled STE, the effect of β in the Tanh/IR-Net surrogate, the effect of window width in the threshold-window methods, and the comparison between hard and smooth threshold windows, are provided in Figure A.3.

Based on the initial evaluation, three candidate methods were selected for more detailed comparison: Scaled STE with $s = 0.7$, Scaled STE with $s = 0.75$, and the Smooth Threshold-Window surrogate with $r = 0.7$. These candidates were selected because they achieved the strongest overall performance among the evaluated configurations while representing different surrogate-gradient behaviors. Their detailed comparison is shown in Figure 5.5, with additional supporting results provided in Figure A.4.

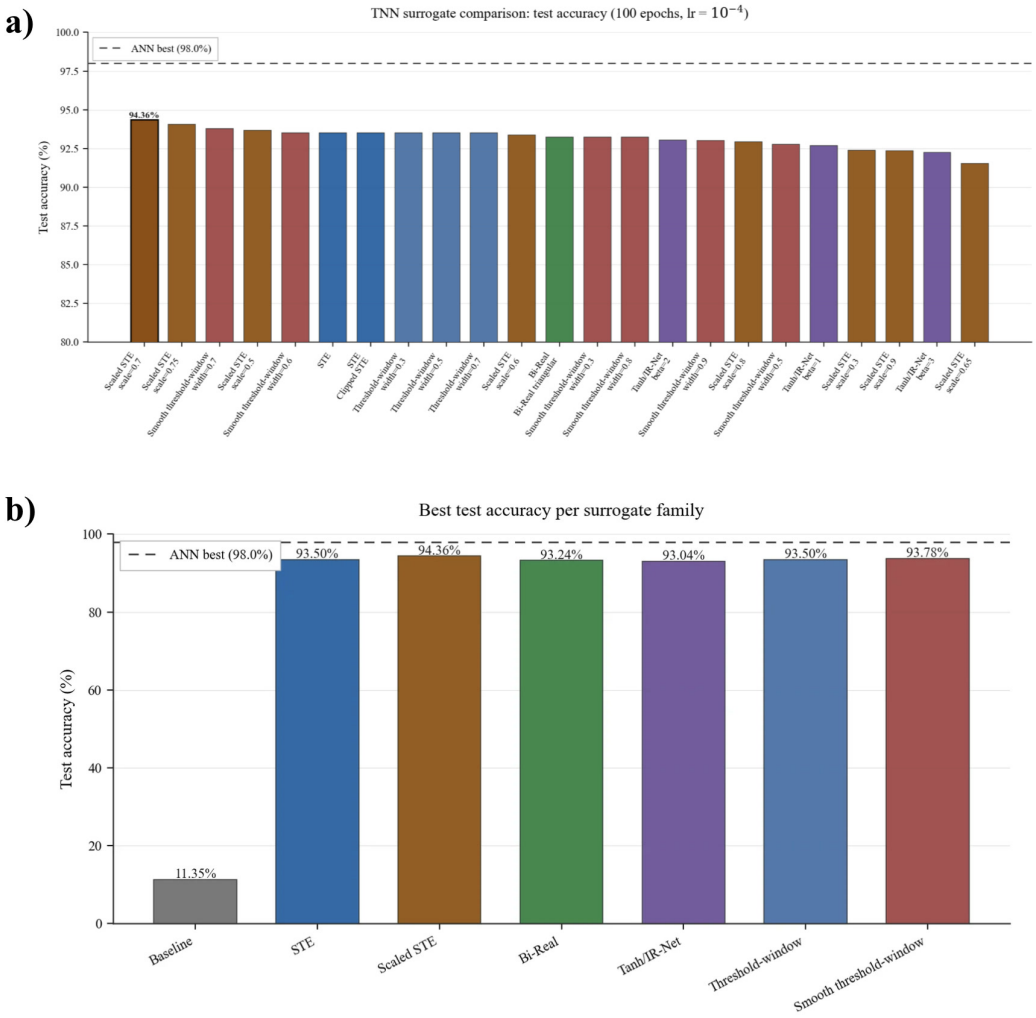


Fig. 5.4 Comparison of TNN training performance using different surrogate-gradient families.

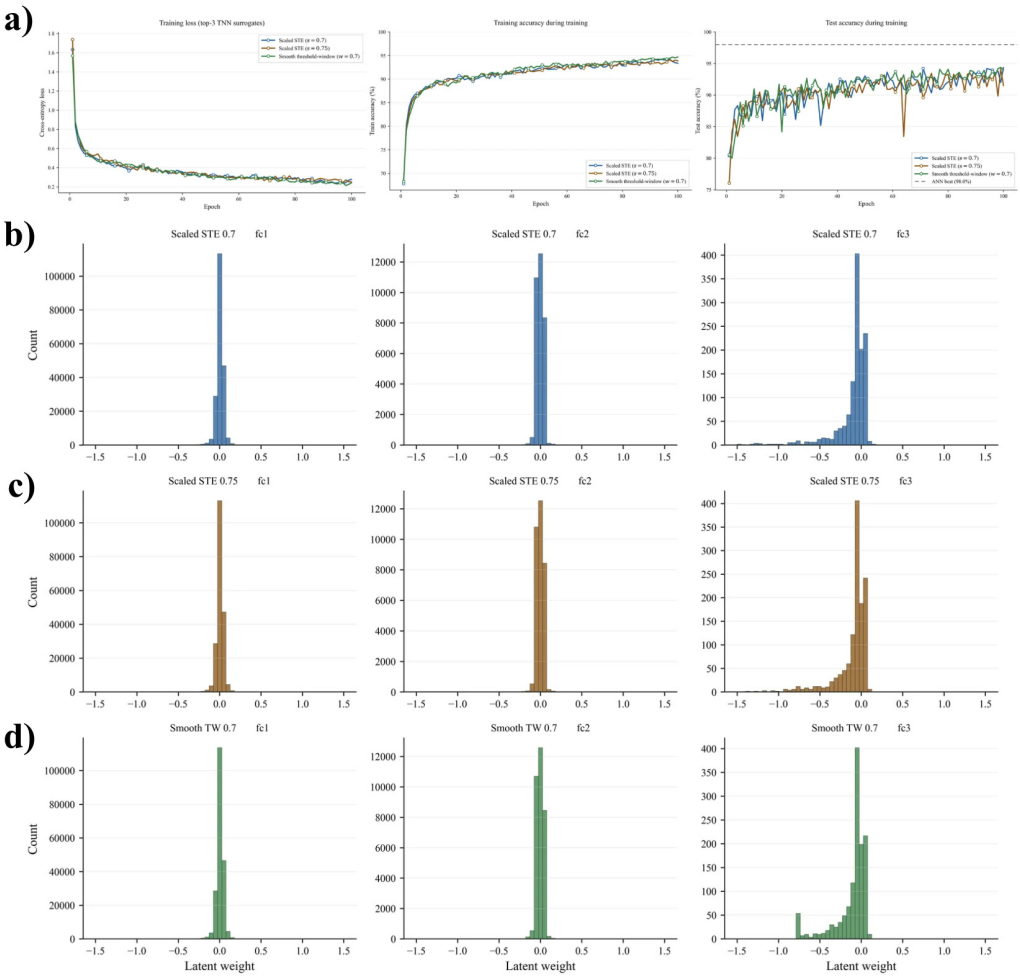


Fig. 5.5 Detailed comparison of the three selected TNN surrogate-gradient candidates: Scaled STE with $s = 0.7$, Scaled STE with $s = 0.75$, and Smooth Threshold-Window surrogate with $r = 0.7$.

The comparative evaluation of the three candidate surrogate-gradient methods revealed that all approaches successfully trained the TNN and achieved final test accuracies above 91%. However, clear differences were observed in convergence stability, optimization behavior, and latent weight organization.

Among these methods, the Smooth Threshold-Window surrogate with $r = 0.7$ has the most balanced performance. Although the Scaled STE with $s = 0.7$ reached slightly higher accuracy, the Smooth Threshold-Window method has smoother learning dynamics, lower oscillation in the training curves, and more stable test accuracy across epochs. In contrast, the Scaled STE with $s = 0.75$ showed larger fluctuations and less stable optimization behavior.

The latent weight distributions showed that the Smooth Threshold-Window surrogate produced more concentrated and structured weight distributions around the ternary transition regions and also keep sufficient spread for effective quantization. This behavior is also more physically meaningful because gradient flow is concentrated near the switching thresholds, similar to the threshold-dependent conductance transitions observed in memristive devices. Therefore, the Smooth Threshold-Window surrogate with $r = 0.7$ was selected as the preferred training strategy for the final TNN model.

5.3.2 Digital TNN and ANN Comparison

Using the selected surrogate-gradient configuration, namely the Smooth Threshold-Window surrogate with a ternarization threshold of $\Delta = 0.05$ and a window width of $r = 0.7$, the final TNN model was trained on the MNIST dataset using a batch size of 16, a learning rate of 10^{-4}, and 100 training epochs. This model serves as the digital ternary baseline for all subsequent hardware-aware simulations and CrossSim evaluations.

Figure 5.6 compares the performance of the optimized TNN against the previously established full-precision ANN baseline. The training-loss curves shown in Figure 5.6a demonstrate successful convergence for both networks. As expected, the ANN achieves a lower final cross-entropy loss because its weights remain unconstrained and can exploit the full numerical precision available during optimization. In contrast, the TNN must learn under the additional restriction that all inference weights are limited to $\{-1, 0, +1\}$ and it can have a higher final training loss.

The classification accuracies are showed in Figure 5.6b. The full-precision ANN achieves approximately 98.0% test accuracy, whereas the optimized TNN achieves 93.78%. Consequently, the transition from continuous-valued weights to ternary weights introduces an accuracy reduction of approximately 4.2 percentage points. This loss is expected because ternary quantization significantly reduces the representational capacity of the network. Nevertheless, the TNN preserves the majority of the classification performance while offering a substantially simpler hardware implementation.

Additional insights into the parameter distributions are provided in Figure 5.6c–d. Figure 5.6c shows the average absolute weight magnitude of the full-precision ANN, illustrating that the later layers generally contain larger weight values and therefore contribute more strongly to the final classification decision. Figure 5.6d shows the ternary composition of each TNN layer. A dominant fraction of weights converges to the zero state, particularly in the first and second layers, indicating that the trained TNN naturally develops sparse connectivity. This sparsity is advantageous for neuromorphic hardware because inactive connections do not contribute to computation and can potentially reduce programming requirements and power consumption.

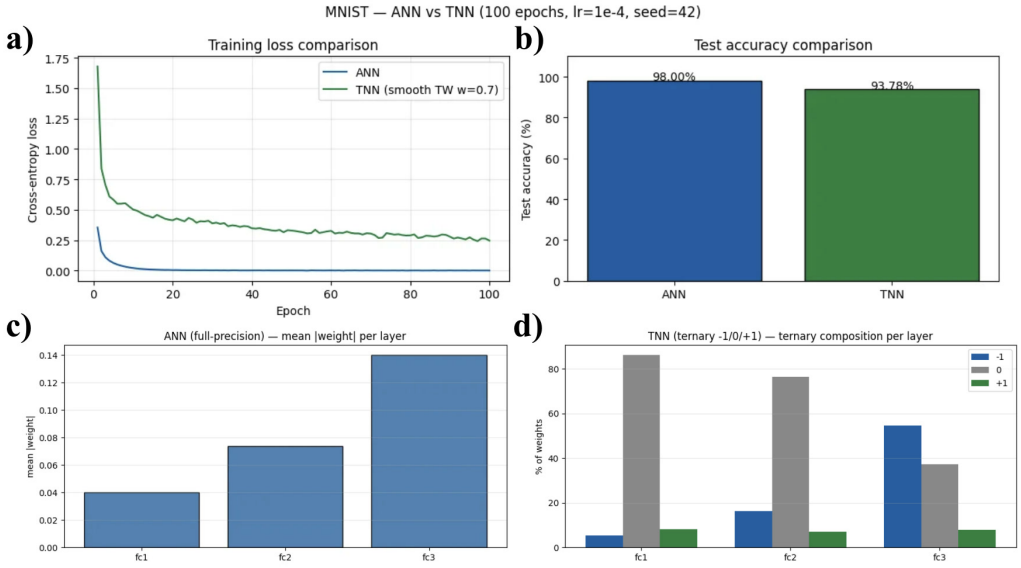


Fig. 5.6 Comparison between the trained ANN and TNN models. (a) Training-loss evolution over 100 epochs. (b) Final test accuracy comparison. (c) Mean absolute weight magnitude for the full-precision ANN. (d) Ternary weight composition percentages.

To further analyze the learned ternary representation, Figure 5.7 presents the weight and bias distributions of the trained TNN. Unlike the ANN discussed previously, whose weights occupy a continuous numerical range, the TNN weight distributions contain only the three discrete values -1, 0, and $+1$. The first and second layers are dominated by the zero state, confirming the sparse nature of the learned representation. A larger number of positive and negative weights can be observed in the output layer. This is expected, as the final layer is responsible for converting the learned internal representations into classification decisions and therefore relies on more strongly differentiated connections between features and output classes.

The bias values retain a continuous distribution because they are not quantized during training or inference. This allows the network to preserve a degree of fine-grained adjustment through the bias terms, despite the use of discrete weight values. It should also be noted that the ternary weights are derived from latent full-precision parameters that remain continuous during optimization and are only converted to ternary values for the forward computation. These latent weight distributions and their dependence on the ternarization threshold are discussed in greater detail in Figure B.1 in Appendix B.

A more intuitive visualization of the ternary structure is presented in Figure 5.8. Figure 5.8a illustrates the active and inactive connections within the first hidden layer. White regions correspond to zero-valued weights, whereas dark regions correspond to active synaptic connections. The visualization clearly demonstrates that a large fraction of the network remains inactive after ternary quantization.

Figure 5.8b breaks down the weights of a representative neuron into positive, negative, and zero-valued groups. Regions associated with positive weights tend to reinforce the neuron's activation, while those linked to negative weights act in the opposite direction. Areas corresponding to zero-valued weights remain inactive and do not participate in the computation. The resulting pattern provides an intuitive view of how ternary networks extract features while maintaining a sparse representation based on only three discrete weight states.

The differences between full-precision and ternary representations become even more visible in Figure 5.9, which compares the weight matrices of all network layers. Figure 5.9a shows the ANN weight matrices. The continuous color gradients indicate that weights are in a broad range of numerical values distributed around

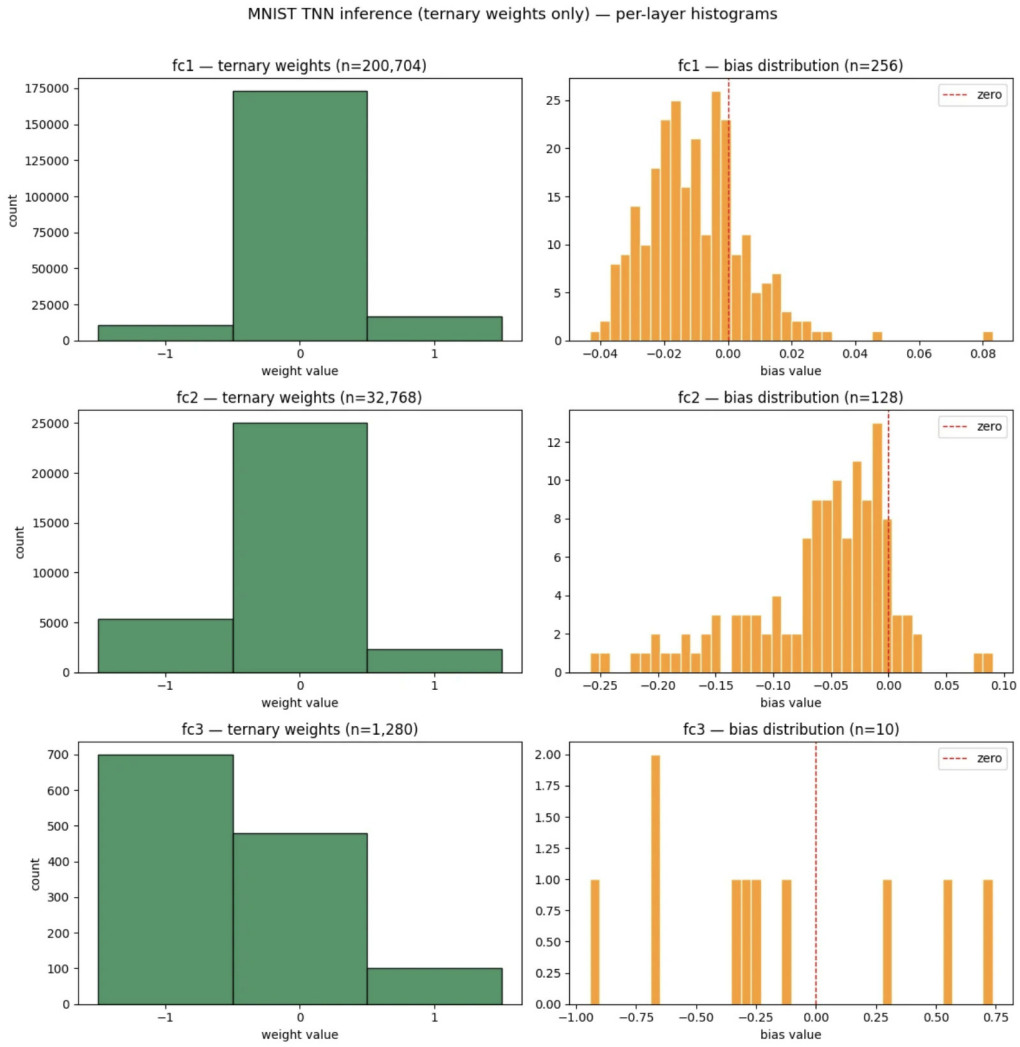


Fig. 5.7 Layer-wise weight and bias distributions of the trained TNN.

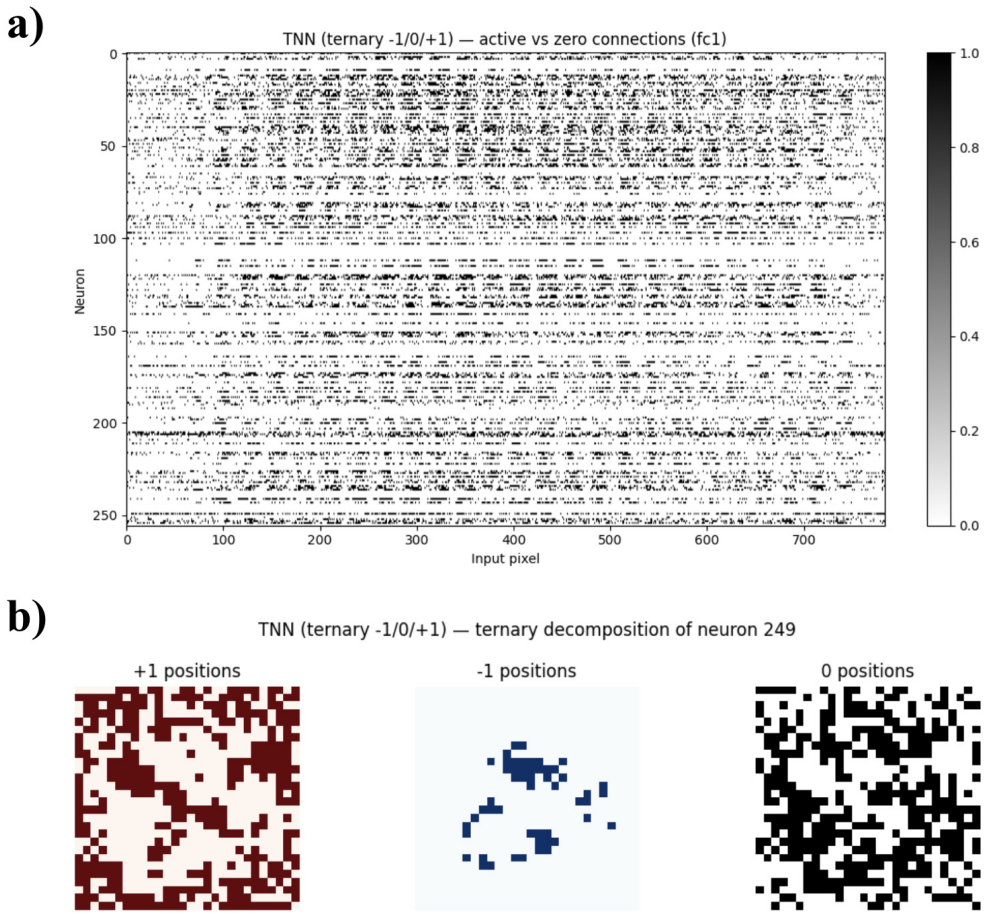


Fig. 5.8 Visualization of ternary connectivity patterns. (a) Active versus inactive connections in the first hidden layer. (b) Decomposition of a representative neuron into positive ($+1$), negative (-1), and zero-valued weight locations.

zero. The learned parameters vary smoothly throughout the network, enabling high representational flexibility and contributing to the high classification accuracy of that.

In contrast, Figure 5.9b shows the corresponding TNN weight matrices. Here, the color scale contains only three possible states corresponding to -1, 0, and $+1$. The absence of intermediate values confirms that the network performs inference using strictly ternary weights. This behavior is particularly visible in the output layer, where the discrete structure becomes highly pronounced.

This comparison clearly illustrates the fundamental trade-off between ANN and TNN architectures, in which the ANN benefits from continuous-valued weight representations and achieves higher accuracy and TNN sacrifices some representational flexibility for high simplified hardware implementation.

From the perspective of the silicon nanowire memristor investigated in this thesis, the TNN representation is especially attractive because it naturally aligns with devices that possess only a small number of stable conductance states. Consequently, ternary networks provide a practical pathway toward efficient implementation on binary or near-binary memristive hardware.

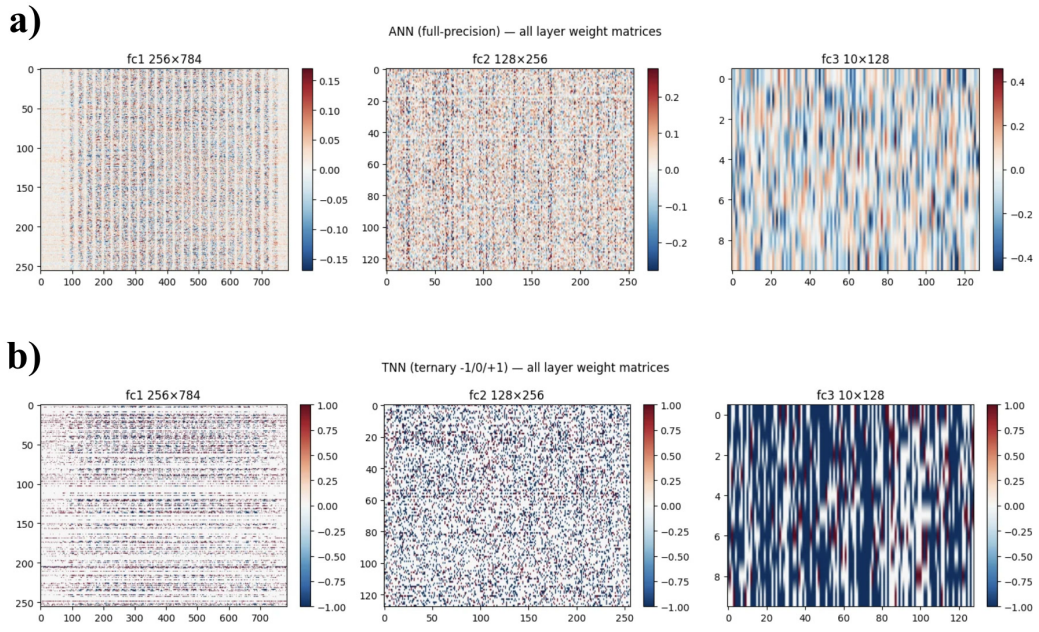


Fig. 5.9 Comparison of ANN and TNN weight matrices. (a) Full-precision ANN weight matrices. (b) TNN weight matrices.

Figure 5.10 shows examples of the neurons and connectivity patterns learned by the ternary network. Although the weights are limited to only three possible values, the network still develops structured feature detectors that capture relevant information from the MNIST images. The sparse nature of these learned connections reflects the constrained representation used by the model and helps explain why ternary networks can be implemented efficiently on neuromorphic hardware.

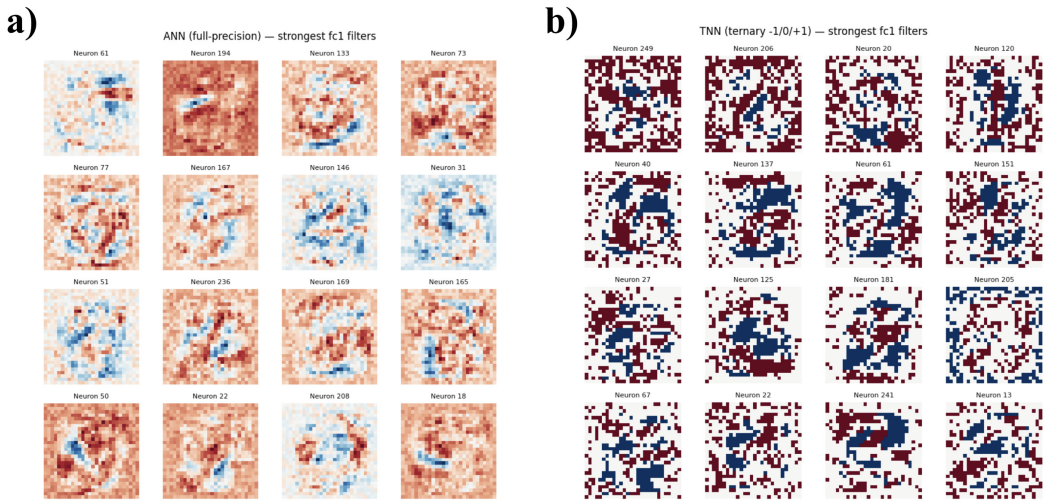


Fig. 5.10 Representative feature detectors learned by the full-precision and TNN.

5.3.3 CrossSim Implementation of ANN and TNN

As discussed previously, the silicon nanowire memristor investigated in this work exhibits only two stable conductance states corresponding to the LRS and HRS. Consequently, direct implementation of arbitrary signed neural network weights is not possible. To overcome this limitation, two alternative strategies were investigated.

The first strategy employs a full-precision ANN implemented using bit slicing. In this approach, each 8-bit weight is sliced into eight binary slices. Furthermore, because memristive crossbar arrays cannot directly represent negative conductance values, each slice is implemented using separate positive and negative arrays. As a result, an 8-bit signed weight requires sixteen physical crossbar arrays per layer (eight positive slices and eight negative slices).

The second strategy employs a TNN. Since TNN weights are restricted to the set $\{-1, 0, +1\}$, only a positive array and a negative array are required for each layer.

This dramatically reduces hardware complexity because only two physical crossbar arrays are needed per layer.

Both implementation strategies were evaluated using the CrossSim framework. In this initial investigation, all hardware non-idealities were disabled in order to establish an ideal baseline. Specifically, programming variability, read noise, conductance drift, ADC quantization, DAC quantization, and parasitic wire resistance were all disabled. The objective of this ideal simulation was twofold. First, it verifies that the CrossSim implementation reproduces the original PyTorch models exactly and therefore provides a reliable foundation for subsequent hardware-aware studies. Second, it allows a direct comparison between the ANN bit-sliced implementation and the TNN implementation before introducing device-level and system-level non-idealities.

The complete implementation flow for both approaches is illustrated in Figure 5.11.

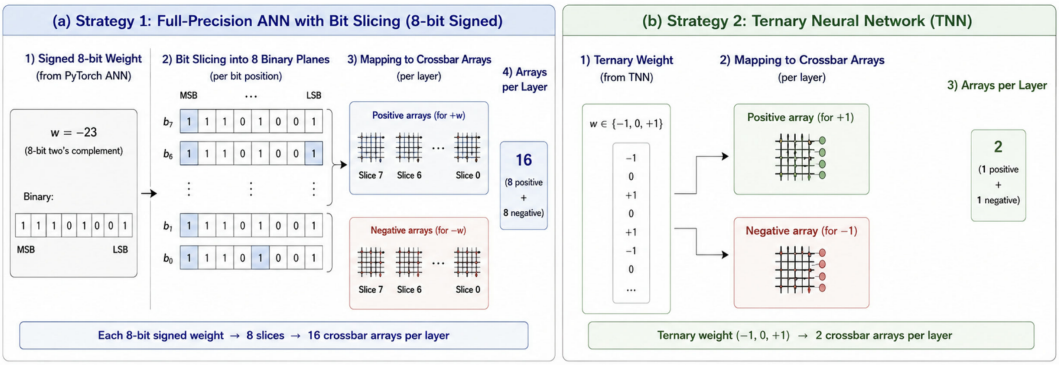


Fig. 5.11 Implementation strategies considered for mapping neural networks onto binary silicon nanowire memristors. The ANN approach employs 8-bit weight quantization and bit slicing, requiring eight positive and eight negative slices per layer. The TNN approach directly maps ternary weights using only positive and negative differential arrays.

Figure 5.12 compares the classification accuracy obtained using the original PyTorch implementations and the corresponding ideal CrossSim implementations. The full-precision ANN achieved 98.00% test accuracy in PyTorch and 98.01% in CrossSim, while the TNN achieved 93.78% in both environments. The negligible differences confirm that the CrossSim implementations reproduce the digital baselines accurately. Therefore, any future accuracy degradation observed after introducing hardware non-idealities can be attributed to the non-idealities themselves rather

Table 5.2 Ideal CrossSim configuration parameters used for ANN and TNN implementations.

Parameter	**Bit-Sliced ANN**	**TNN**
Core style	Bit-Sliced	Balanced
Weight precision	8-bit	Ternary $(-1, 0, +1)$
Number of slices	8	1
Positive/Negative arrays	Yes	Yes
Rows per array	1024	1024
Columns per array	1024	1024
Cell bits	1	1
R_{min}	$1.41 \times 10^9\ \Omega$	$1.41 \times 10^9\ \Omega$
R_{max}	$4.43 \times 10^9\ \Omega$	$4.43 \times 10^9\ \Omega$
V_{read}	1.0 V	1.0 V
Programming error	Disabled	Disabled
Read noise	Disabled	Disabled
Drift error	Disabled	Disabled
ADC quantization	Disabled	Disabled
DAC quantization	Disabled	Disabled
Parasitic resistance	Disabled	Disabled

than implementation inconsistencies between the software and hardware simulation environments.

Additional verification is provided through the confusion matrices shown in Figure 5.13. Figures 5.13a and 5.13b compare the ANN predictions obtained from PyTorch and ideal CrossSim, while Figures 5.13c and 5.13d perform the same comparison for the TNN. The near-identical structure of the confusion matrices confirms that the class-wise prediction behavior is preserved exactly after mapping the networks into the CrossSim framework.

To further verify the correctness of the weight mapping process, the digital weight matrices were compared directly with the conductance matrices reconstructed from CrossSim. Figure 5.14 presents this analysis for the ANN implementation. For each layer, the left column shows the original digital weight matrix, the middle column shows the reconstructed CrossSim weight matrix, and the right column shows the element-wise difference between the two representations. The difference matrices are uniformly zero across all layers, demonstrating that the bit-sliced mapping process preserves the original ANN weights exactly.

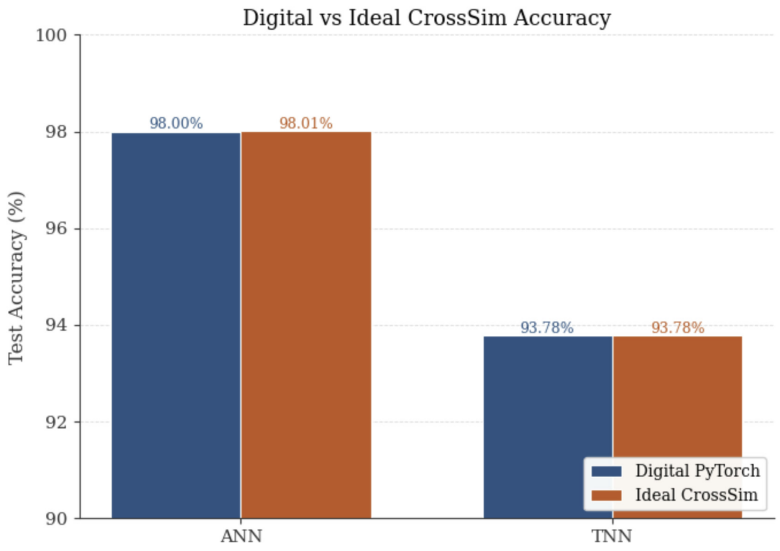


Fig. 5.12 Comparison of test accuracy obtained from the original PyTorch implementations and their corresponding ideal CrossSim implementations.

An equivalent validation for the TNN implementation is presented in Figure 5.15. Again, the difference matrices are identically zero, confirming that the balanced positive-negative mapping strategy reproduces the ternary weight matrices without error. These results provide strong evidence that both ANN and TNN weight representations are correctly implemented within the CrossSim framework.

Figure 5.16 illustrates the conductance representation of the ANN bit-sliced implementation using the third network layer as an example. Each row corresponds to one of the eight binary slices used to represent an 8-bit weight. For every slice, separate positive and negative arrays are required. The effective weight contribution is obtained by subtracting the negative array from the positive array.

This visualization highlights the substantial hardware overhead associated with full-precision ANN implementation on a binary memristive device. Although accurate representation of continuous-valued weights is achieved, each neural network layer requires sixteen physical crossbar arrays, in addition to the associated peripheral circuitry such as ADCs, DACs, and routing resources. Similar visualizations for the first and second layers are provided in Figures B.2 and B.3 respectively.

The corresponding conductance mapping for the TNN implementation is shown in Figure 5.17. Unlike the bit-sliced ANN, no slicing operation is required because

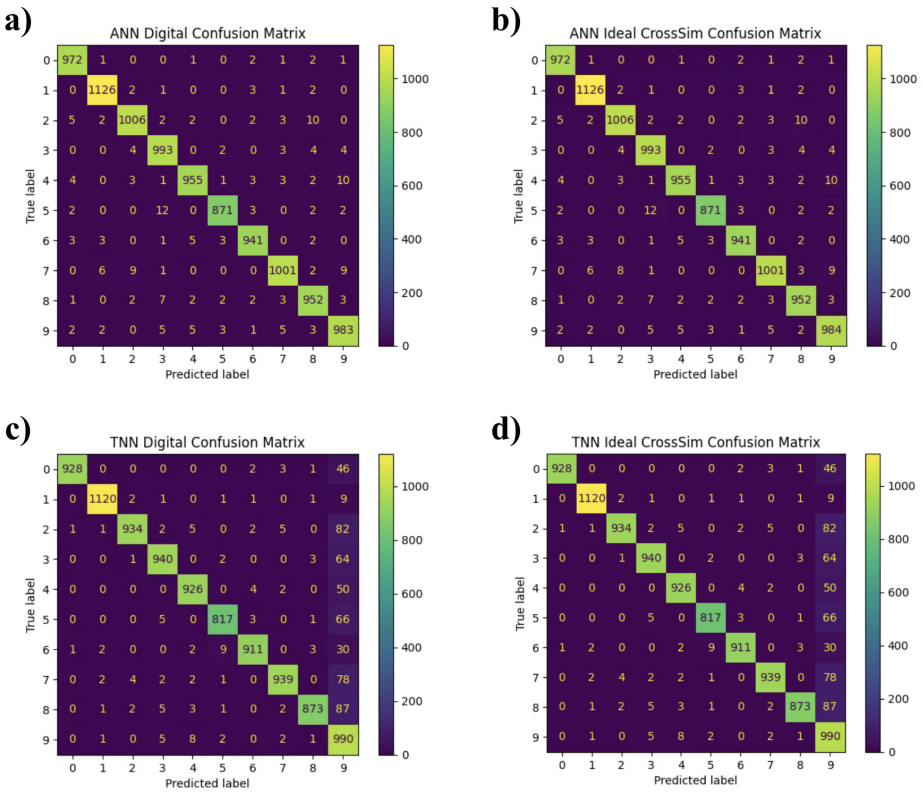


Fig. 5.13 Confusion matrices for digital and ideal CrossSim implementations. (a) ANN implemented in PyTorch. (b) ANN implemented in ideal CrossSim. (c) TNN implemented in PyTorch. (d) TNN implemented in ideal CrossSim.

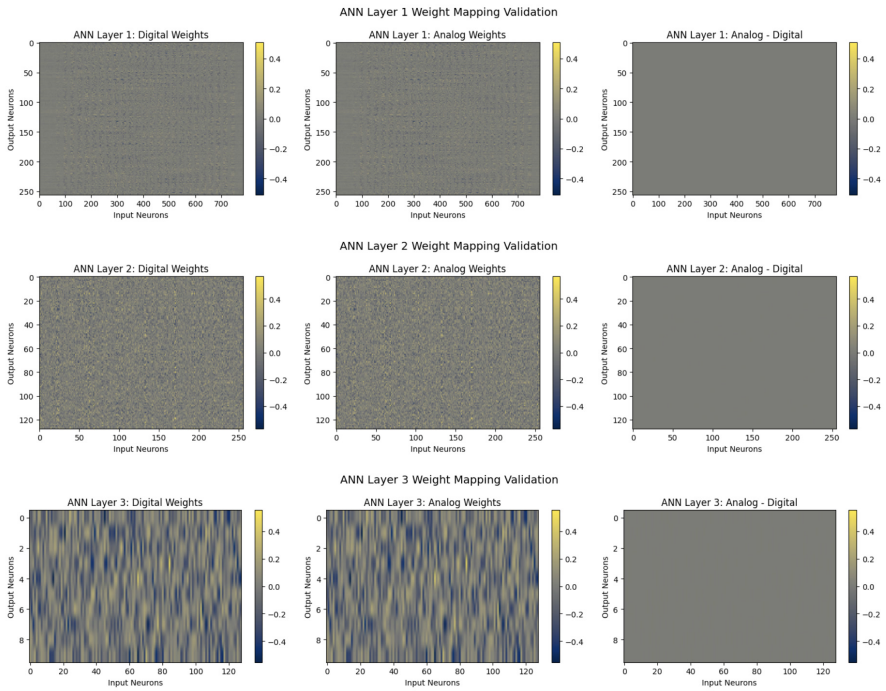


Fig. 5.14 Validation of ANN weight mapping between PyTorch and ideal CrossSim. For each layer, the left column shows the original digital weights, the middle column shows the reconstructed CrossSim weights, and the right column shows the element-wise difference.

the weights are already ternary. Each layer therefore requires only two physical arrays: a positive array and a negative array.

The positive array stores all $+1$ weights, while the negative array stores all -1 weights. Zero-valued weights are represented by the absence of programmed conductance in both arrays. The effective weight matrix is obtained through differential subtraction of the positive and negative currents. As shown most clearly in the third layer, each individual array contains only two conductance states corresponding directly to the experimentally measured HRS and LRS states of the silicon nanowire device. Consequently, ternary weights can be represented using only two binary conductance states while requiring only two physical arrays per layer.

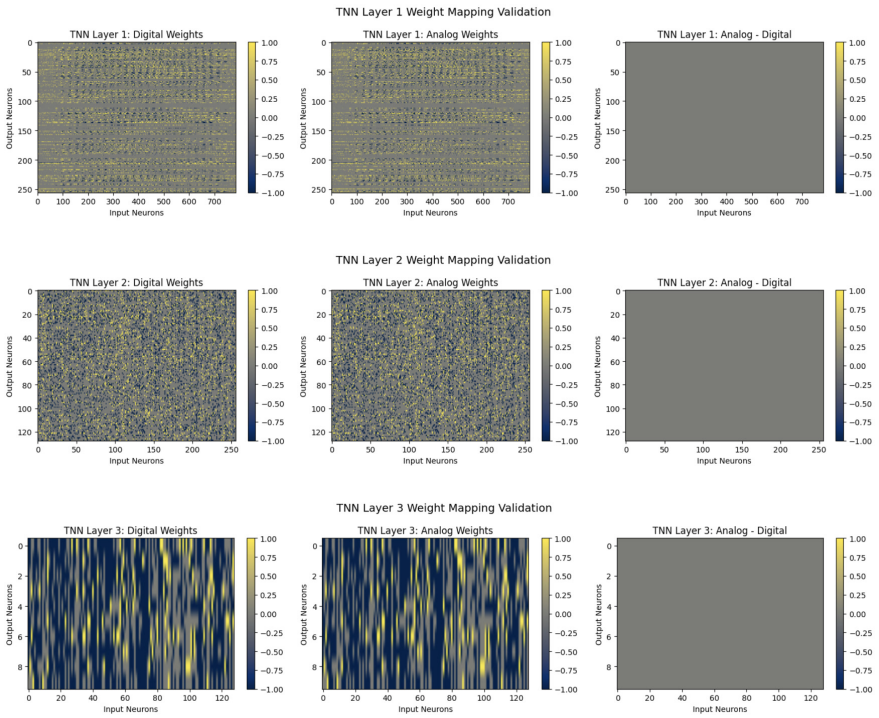


Fig. 5.15 Validation of TNN weight mapping between PyTorch and ideal CrossSim. For each layer, the left column shows the original ternary weights, the middle column shows the reconstructed CrossSim weights, and the right column shows the difference matrix.

5.3.4 Hardware Trade-Off Analysis and Selection of ANN versus TNN

Following the successful implementation and validation of both ANN and TNN architectures within the CrossSim framework, the next step is to evaluate the trade-offs associated with each approach. Although both implementations are functionally correct and achieve high classification accuracy, they differ substantially in hardware complexity, memory requirements, and scalability when implemented using binary silicon nanowire memristors. Therefore, selecting the most appropriate architecture requires balancing classification performance against implementation cost.

The overall comparison is summarized in Figure 5.18. Figure 5.18a shows the classification accuracy of both approaches. The full-precision ANN achieved a test accuracy of approximately 98.0%, whereas the optimized TNN achieved 93.78%. Consequently, adopting the TNN architecture results in an accuracy reduction of approximately 4.22 percentage points. While this reduction is not negligible, it must

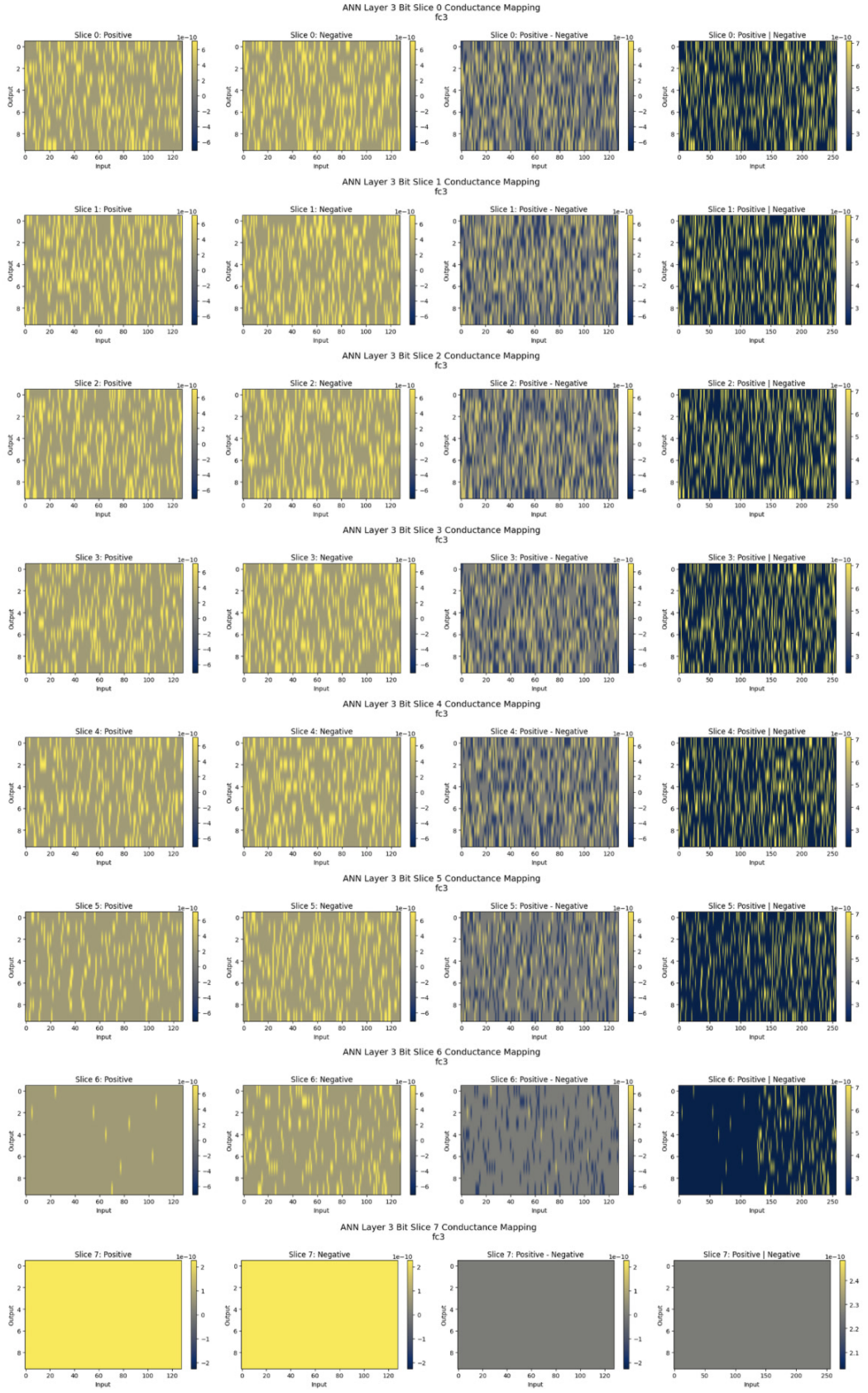


Fig. 5.16 Bit-sliced conductance mapping of the third ANN layer. Each row corresponds to one of the eight binary slices used to represent an 8-bit weight. For every slice, separate positive and negative conductance arrays are required. The effective weight representation is obtained through differential subtraction of the positive and negative arrays.

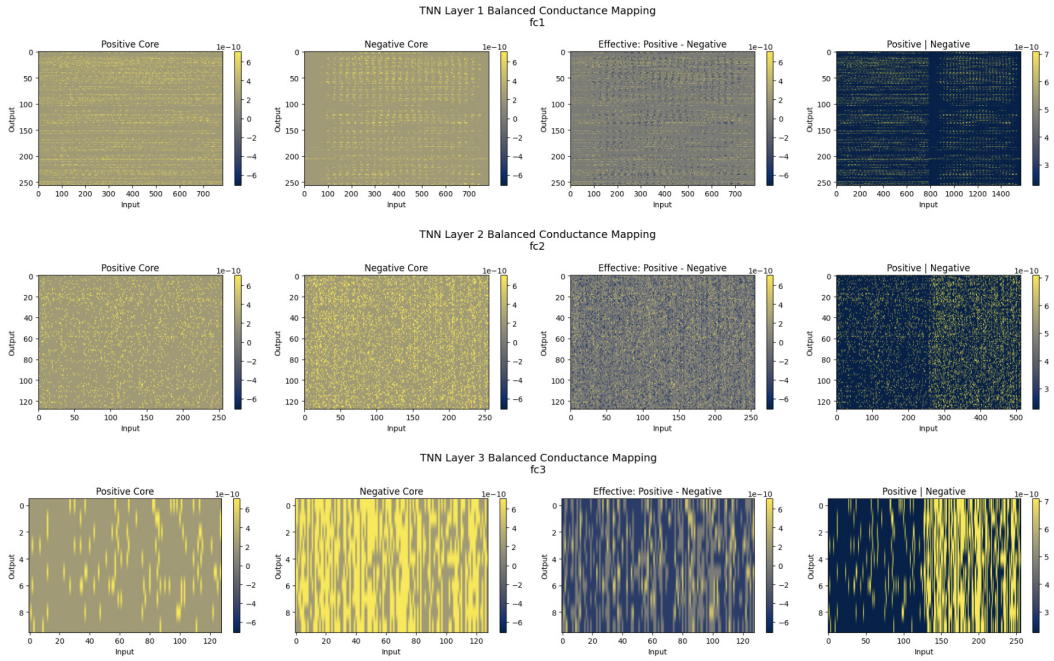


Fig. 5.17 Balanced conductance mapping of the TNN layers. For each layer, the positive array stores $+1$ weights and the negative array stores -1 weights. The effective ternary weight matrix is obtained by subtracting the negative conductance matrix from the positive conductance matrix.

be evaluated in the context of the substantial hardware savings obtained through ternary weight representation.

The most significant difference arises from the number of crossbar arrays required to implement each network. As shown in Figure 5.18b, the ANN implementation relies on 8-bit weight representation through bit slicing. Because each slice requires both positive and negative arrays, a total of sixteen crossbar arrays are required per neural network layer. In contrast, the TNN implementation requires only a single positive array and a single negative array per layer, resulting in only two crossbar arrays per layer. This corresponds to an eight-fold reduction in the number of physical arrays required for computation.

Figure 5.18c further compares the number of memristive devices required for each implementation. For the first layer, the ANN implementation requires approximately 3.21 million memristors, whereas the TNN implementation requires only approximately 401 thousand devices. For the second layer, the ANN requires approximately 520 thousand memristors compared with approximately 66 thousand for

the TNN. Even more notably, the second ANN layer alone requires more memristors than the entire first TNN layer. For the third layer, the ANN implementation requires approximately 20 thousand memristors, while the TNN requires only approximately 3 thousand devices.

When all layers are considered together, the total number of memristors required for the ANN implementation reaches approximately 3,757,032 devices. In contrast, the TNN implementation requires only approximately 469,504 devices. This corresponds almost exactly to the expected eight-fold reduction resulting from the elimination of bit slicing.

However, the reduction in hardware complexity extends far beyond the number of memristive devices. Each crossbar array needs additional peripheral circuitry including ADCs, DACs, control logic, routing resources, and interconnect wiring. Therefore, reducing the number of crossbar arrays directly reduces the amount of supporting circuitry required. For the three-layer network considered in this work, the ANN implementation requires a total of 48 crossbar arrays (3 layers $\times$ 16 arrays per layer), whereas the TNN implementation requires only 6 crossbar arrays (3 layers $\times$ 2 arrays per layer). Consequently, the number of required peripheral blocks is also dramatically reduced. For example, assuming shared peripheral resources, the ANN implementation requires approximately 24 ADC units, whereas the TNN implementation requires only 3 ADC units, as illustrated in Figure 5.18d.

From an energy-efficiency perspective, fewer crossbar arrays imply fewer memory accesses, reduced interconnect activity, lower peripheral-circuit utilization, and a smaller overall chip area. These reductions directly translate to lower power consumption and also improved scalability for large neuromorphic systems. Furthermore, the TNN implementation maps naturally onto the binary conductance properties of the silicon nanowire memristor and it needs only the experimentally observed HRS and LRS states. In contrast, the ANN implementation achieves high precision only through the additional complexity of bit slicing and multiple differential arrays.

Therefore, despite the approximately 4% reduction in classification accuracy, the TNN architecture offers a substantially more practical solution for implementation using the silicon nanowire devices investigated in this work. The dramatic reductions in crossbar count, memristor count, peripheral circuitry, area requirements, and expected energy consumption outweigh the modest loss in accuracy for many neuromorphic computing applications. Consequently, the TNN architecture was selected

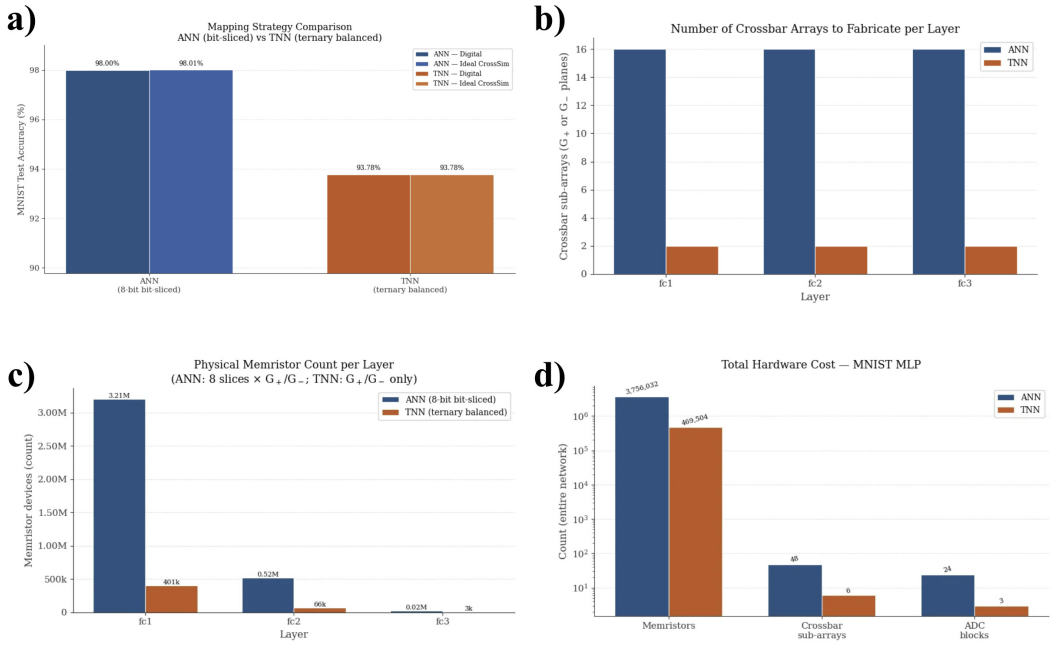


Fig. 5.18 Hardware trade-off comparison between ANN and TNN implementations. (a) Classification accuracy comparison. (b) Number of crossbar arrays required per layer. (c) Number of memristive devices required for each layer and for the complete network. (d) Estimated peripheral-circuit requirements.

as the primary candidate for the subsequent hardware-aware analysis presented in the remainder of this thesis, where device-level and system-level non-idealities are introduced and evaluated.

5.4 Impact of Real Silicon Nanowire Device Characteristics on TNN Performance

The previous sections demonstrated that TNNs provide a significantly more hardware-efficient implementation strategy than bit-sliced ANNs when implemented using binary silicon nanowire memristors. However, the results obtained thus far correspond only to ideal CrossSim simulations, where all device-level and system-level non-idealities were disabled. Under these ideal assumptions, the TNN achieved a classification accuracy of 93.78%.

In practice, the conductance values stored in the memory array do not always match the intended values exactly. A variety of non-ideal effects can introduce deviations during programming, storage, and readout. As discussed in Chapter 4, these effects arise from both the device and the surrounding hardware. Examples at the device level include programming variability, read noise, and conductance drift, while system-level factors such as ADC and DAC quantization and parasitic interconnect resistance can further influence the computation. Collectively, these non-idealities alter the effective neural network weights and can ultimately affect inference accuracy.

Therefore, the objective of this section is to incorporate experimentally measured silicon nanowire device characteristics into the CrossSim framework and evaluate their impact on neural network performance. The analysis begins with device-level effects, specifically programming variability and conductance drift, before considering higher-level circuit non-idealities in subsequent sections.

5.4.1 Experimental Device Modeling

To accurately model the behavior of the silicon nanowire memristor within CrossSim, experimentally measured electrical data were used to extract the parameters

governing device variability. The available dataset consists of two complementary measurements.

The first dataset contains fifty consecutive current–voltage (I–V) sweeps performed on the same device over the voltage range of -4 V to $+4$ V. These measurements provide information regarding cycle-to-cycle variability and switching repeatability.

The second dataset consists of repeated resistance-state measurements acquired from three independent devices. For each device, both the LRS and HRS were programmed and measured five times using a read voltage of $V_{\text{read}} = 1$ V. These measurements provide information regarding both device-to-device and cycle-to-cycle programming variability.

Using these experimental measurements, statistical models were extracted for programming variability and conductance drift. These models were subsequently integrated into CrossSim through a custom device implementation developed specifically for this work, referred to as the *APM SiNW SBFET* model (Silicon Nanowire Schottky Barrier Field-Effect Transistor).

The objective of this custom model is to reproduce the experimentally observed conductance variations during neural network weight programming while maintaining compatibility with the CrossSim simulation framework.

Programming Error Modeling

Programming variability represents the deviation between the target conductance requested by the neural network and the actual conductance achieved after programming the physical device. Because neural network weights are stored as conductance values within the crossbar array, such deviations directly perturb the stored weights and may degrade inference accuracy.

The experimentally measured HRS and LRS data were first analyzed to determine the nominal device operating range. The average resistance values extracted from the measurements yielded

$$R_{\text{min}} = 1.410360 \times 10^{9}\ \Omega$$

and

$$R_{\max} = 4.429246 \times 10^9\ \Omega$$

Using a read voltage of $V_{\text{read}} = 1$ V, the corresponding conductance values are

$$G_{\text{LRS}} = \frac{1}{R_{\min}} = 0.709\ \text{nS}$$

and

$$G_{\text{HRS}} = \frac{1}{R_{\max}} = 0.225\ \text{nS}$$

These values define the nominal conductance states used throughout the hardware simulations.

Figure 5.19a illustrates the experimentally measured HRS and LRS distributions together with the extracted target conductance values. Figure 5.19b presents the fifty repeated I–V sweeps used to characterize cycle-to-cycle variability. By evaluating the conductance at the read voltage of 1 V for all measured cycles, distributions of actual programmed conductance values were obtained for both states, as shown in Figure 5.19c.

The difference between the target conductance and the experimentally observed conductance distributions defines the programming error. Figures 5.19d–f summarize the resulting programming-error statistics, including relative conductance error, absolute conductance error, and error distributions referenced to the target conductance values.

These experimentally derived error distributions were incorporated directly into the custom APM SiNW SBFET CrossSim model. Consequently, whenever CrossSim programs a conductance value corresponding to a neural network weight, the programmed value is perturbed according to the experimentally measured programming-error statistics rather than assuming ideal programming behavior.

Validation of the Device Model

To verify the correctness of the custom programming-error implementation, the resulting device behavior was compared against the reference SONOS device model

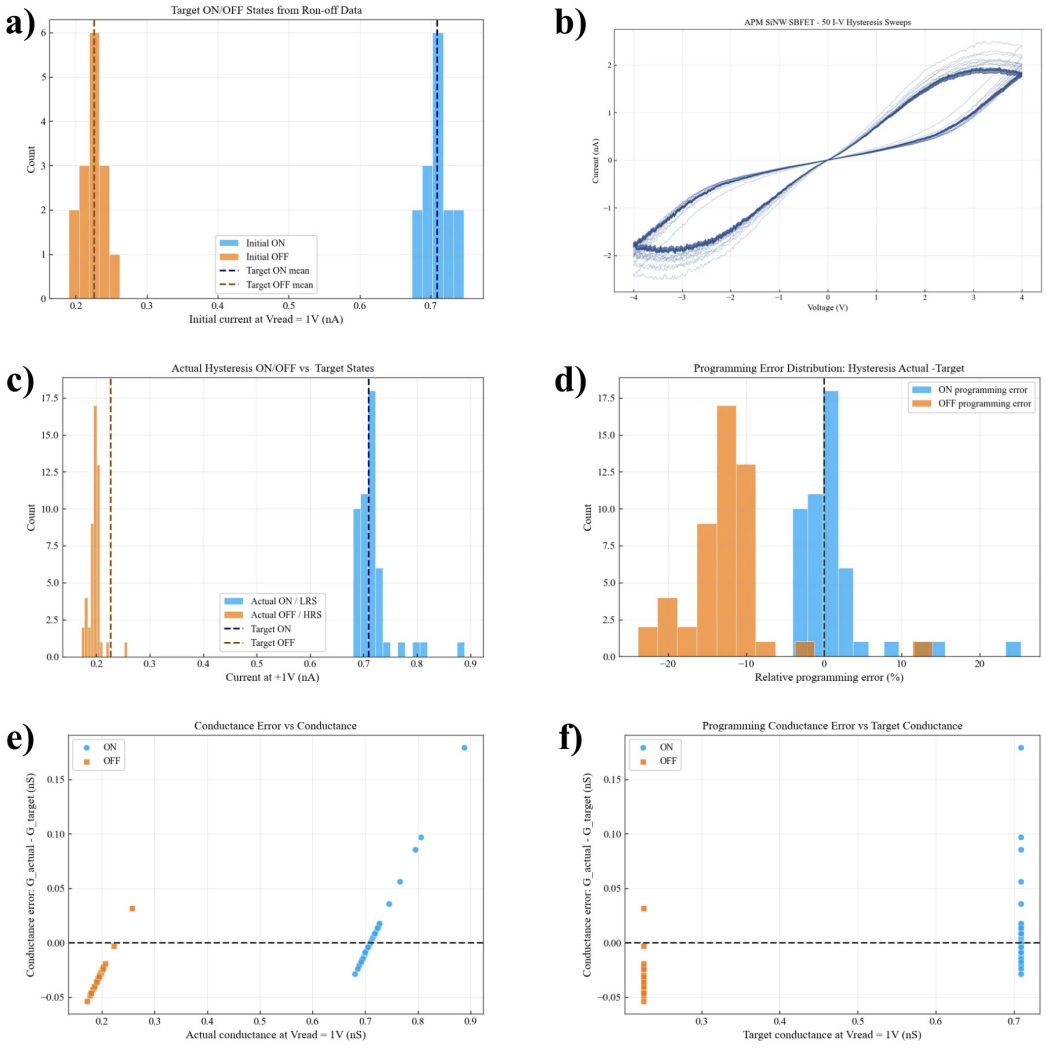


Fig. 5.19 Experimental characterization data used for programming-error extraction and device modeling of the silicon nanowire memristor. (a) Measured HRS and LRS distributions together with the extracted target conductance states. (b) Fifty repeated current–voltage sweeps used to characterize cycle-to-cycle variability. (c) Conductance distributions extracted at $V_{\text{read}} = 1$ V showing the actual programmed states. (d) Relative programming-error distributions. (e) Absolute conductance-error distributions. (f) Conductance error referenced to the target conductance values. These experimentally derived statistics were incorporated directly into the custom APM SiNW SBFET CrossSim device model.

supplied with CrossSim. The SONOS model serves as a well-established benchmark within the simulator and provides a useful reference for validating custom device implementations.

A detailed comparison between the SONOS model and the proposed APM SiNW SBFET model is presented in Figure B.4. The results confirm that the custom implementation correctly reproduces the intended programming-error behavior while preserving compatibility with the CrossSim device framework.

Having verified the device model, the impact of experimentally measured programming variability on neural network inference accuracy can now be investigated.

Impact of Programming Variability on TNN Accuracy

To evaluate the influence of programming variability, the TNN configuration established in the previous sections was reused without modification. All simulation parameters remained unchanged except that programming variability was enabled using the experimentally derived APM SiNW SBFET device model.

The resulting accuracy comparison is presented in Figure 5.20a. The original digital TNN achieved a classification accuracy of 93.78%, while the ideal CrossSim implementation without programming variability also achieved 93.78%. When programming variability was enabled, the classification accuracy decreased to 92.56%.

Consequently, experimentally measured programming variability introduces an accuracy degradation of approximately 1.22 percentage points. This reduction originates from the mismatch between the target conductance values requested by the neural network and the actual conductance values programmed into the physical device.

To better understand the origin of this degradation, Figure 5.20b compares the stored weight matrices before and after programming variability is applied. For each neural network layer, the left column shows the ideal programmed weights, the middle column shows the weights after applying experimentally measured programming error, and the right column presents the element-wise difference between the two representations.

Unlike the ideal mapping validation presented previously, the difference matrices are no longer identically zero. The visible deviations indicate that the conductance

perturbations introduced during programming modify the effective neural network weights. Although individual perturbations appear relatively small, their cumulative effect across thousands of synaptic connections results in the observed reduction in classification accuracy.

These results demonstrate that programming variability constitutes a non-negligible source of error for silicon nanowire-based neuromorphic systems and therefore must be considered during hardware-aware neural network design and optimization.

Drift Error Modeling

In addition to programming variability, the temporal stability of the programmed conductance states must also be considered. Although the silicon nanowire device exhibits non-volatile behavior, the stored conductance values are not perfectly constant over time. Instead, the conductance continues to evolve gradually after programming as the device relaxes toward a new equilibrium state. This behavior, commonly known as conductance drift, comes from the physical processes that govern charge transport and the dynamics of interface states inside device. As much as the conductance changes with time, the effective neural network weights which are represented by the memory cells also change and it introduce additional errors during inference.

To characterize experimentally this behavior, drift measurements were performed on three independent devices (D1, D2, and D3). For each device, five repeated measurements were acquired for both the ON state (LRS) and OFF state (HRS) and totally it has been fifteen ON-state drift curves and fifteen OFF-state drift curves. During each experiment, the device current was monitored continuously for 100 seconds after programming. These measurements provide information regarding both device-to-device and cycle-to-cycle drift variability.

Figure 5.21 summarizes the drift behavior measured in the ON state. The raw current data are presented in Figure 5.21a. While the overall behavior is similar across different devices and measurement cycles, some variation in the absolute current levels can be observed. To facilitate comparison, the data were normalized to their respective initial current values, as shown in Figure 5.21b.

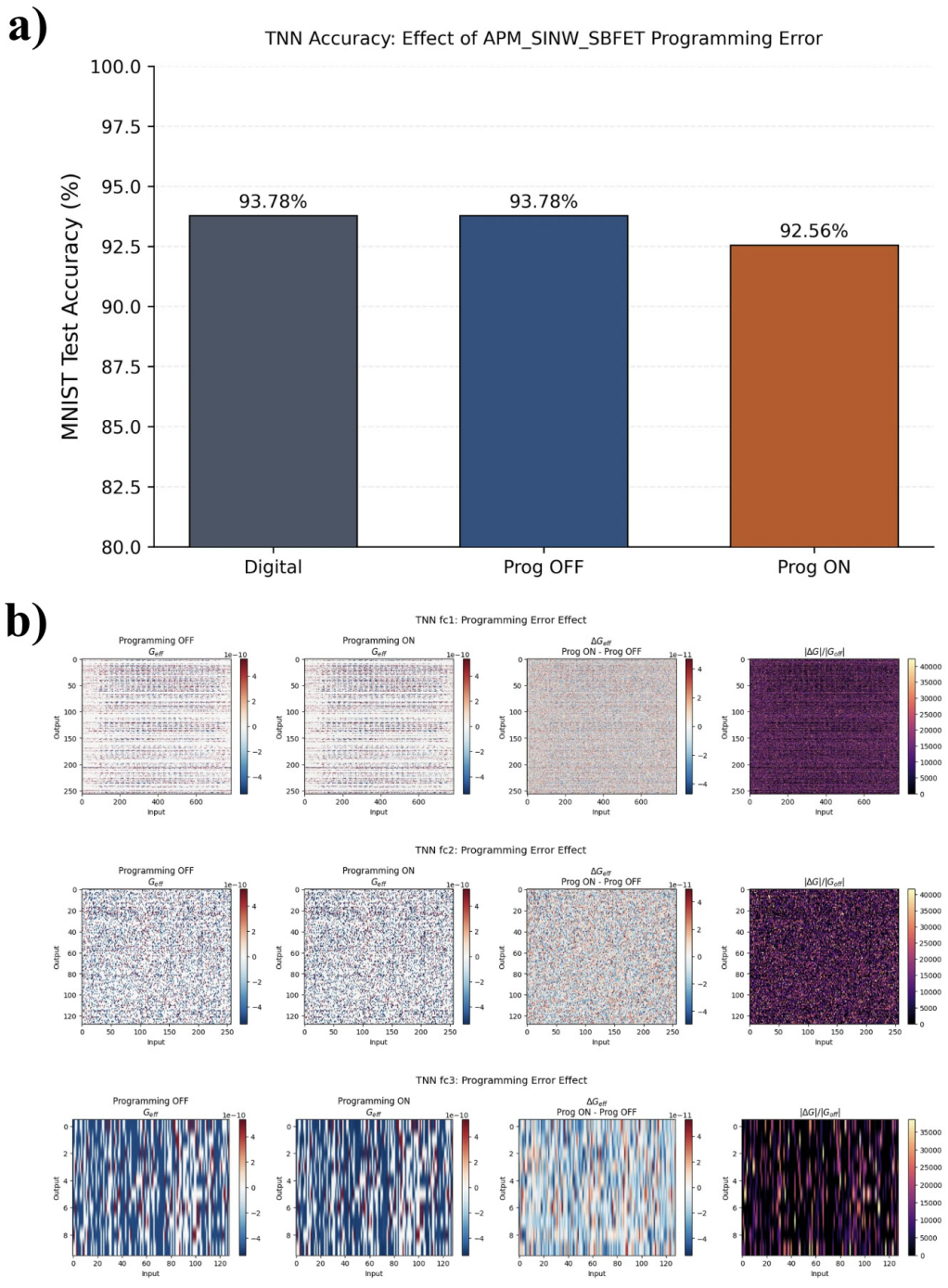


Fig. 5.20 Impact of experimentally measured programming variability on TNN inference. (a) Classification accuracy comparison between the digital TNN baseline, ideal CrossSim implementation (Programming OFF), and CrossSim implementation with experimentally modeled programming variability enabled (Programming ON). (b) Weight-matrix comparison for each network layer showing the ideal programmed weights, the programmed weights after applying experimental programming variability, and the corresponding element-wise difference matrices.

After normalization, the drift behavior becomes more consistent and the underlying temporal trend is easier to identify. The normalized data follow a characteristic evolution with time that can be described using a power-law model. The resulting fits are shown in Figure 5.21c together with the corresponding experimental measurements.

The ON-state drift was modeled using the power-law expression

$$I_{\mathrm{ON}}(t) = I_0 \left(1 + \frac{t}{t_0}\right)^a \tag{5.13}$$

where I_0 is the initial current, $t_0 = 1$ s is a normalization constant, and a is the drift exponent. Fitting all fifteen ON-state measurements yielded

$$a_{\mathrm{mean}} = 0.0258$$

with a standard deviation of

$$\sigma_a = 0.0029$$

Across all measured devices, the fitted exponent is in range of 0.0201 and 0.0329, and it shows relatively consistent ON-state drift behavior.

For OFF state, the story is different. As shown in Figure 5.22a, the OFF-state current initially decreases and then exhibits a slow recovery trend. Furthermore, larger variability is observed between devices, particularly for Device 3. After normalization (Figure 5.22b), it becomes clear that a simple power-law model is insufficient to capture the observed behavior. For this reason, a two-term exponential model was adopted, and the fitted results are shown in Figure 5.22c.

The OFF-state drift was modeled as

$$I_{\mathrm{OFF}}(t) = I_0 \left[1 + A\left(1 - e^{-t/\tau_1}\right) + B\left(1 - e^{-t/\tau_2}\right)\right] \tag{5.14}$$

where A and B specify the amplitudes of the two drift components and τ_1 and τ_2 represent the corresponding time constants. Fitting the fifteen OFF-state measurements produced a family of parameter sets rather than a single deterministic curve. The average fitted parameters were

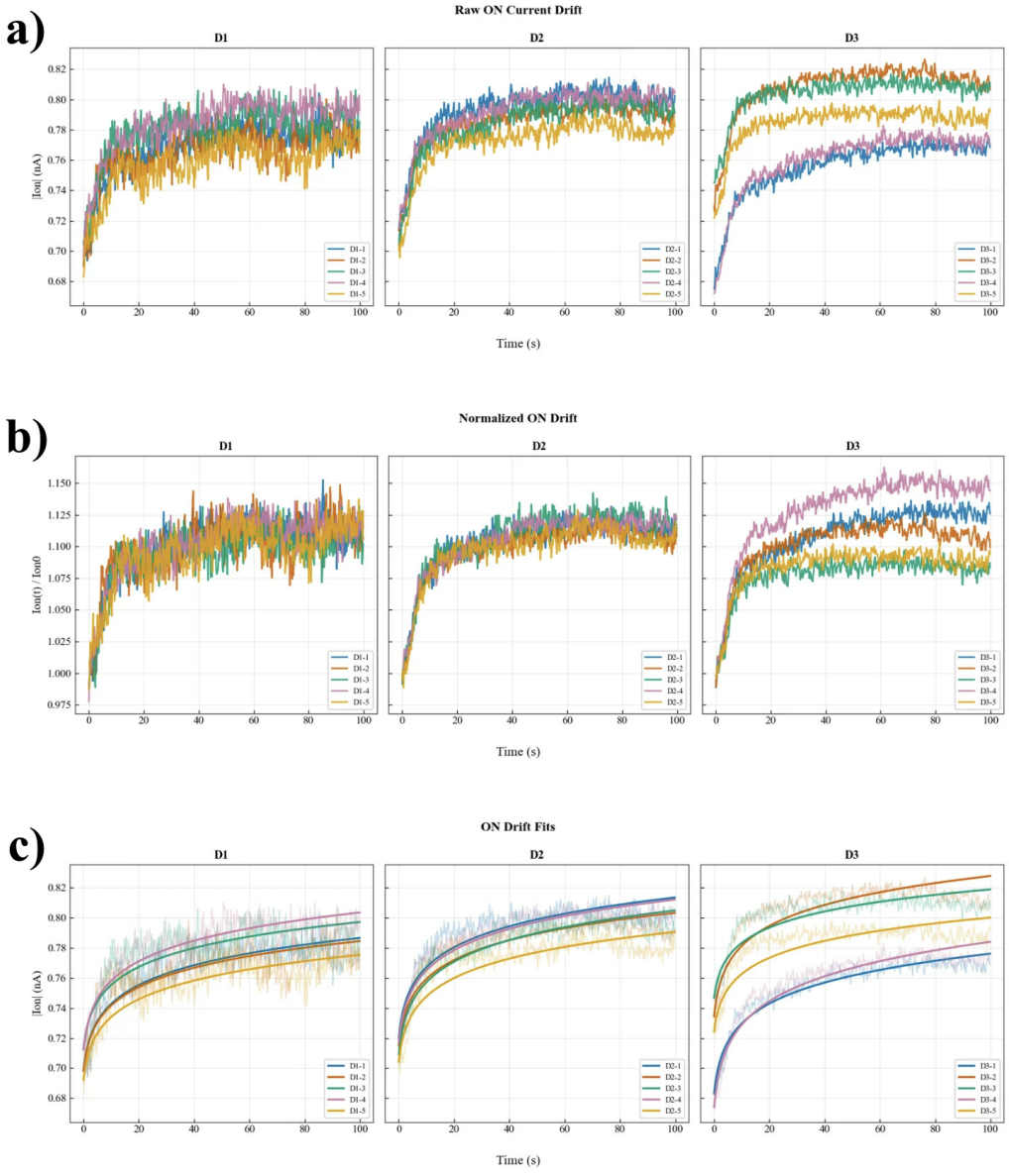


Fig. 5.21 ON-state (LRS) drift characterization. (a) Raw measured current drift for three devices with five repeated measurements each over a period of 100 seconds. (b) Normalized ON-state drift curves. (c) Power-law fits applied to the normalized ON-state measurements.

$$A = -0.429, \qquad \tau_1 = 16.5 \text{ s},$$

and

$$B = 0.631, \qquad \tau_2 = 196.9 \text{ s}.$$

The average fitting errors remained low, with mean normalized Root Mean Squared Error (RMSE) values of approximately 1% for the ON state and approximately 2% for the OFF state, indicating that the selected models provide an accurate representation of the measured drift behavior.

To incorporate these effects into CrossSim, the experimentally extracted drift statistics were embedded directly into the custom APM SiNW SBFET device model. For the ON state, the drift exponent a is sampled from the experimentally fitted distribution. For the OFF state, the model randomly selects one of the experimentally fitted parameter sets (A, τ_1, B, τ_2) obtained from the measured devices. This approach keeps both device-to-device and cycle-to-cycle variability and also maintains consistency with the experimental data.

The resulting CrossSim drift model is compared with the measured experimental data in Figure 5.23. Figure 5.23a compares the ON-state model against the measured normalized drift curves. The gray curves correspond to the experimental measurements, while the blue curves represent randomly sampled model realizations and the dark blue curve corresponds to the empirical mean model. Similarly, Figure 5.23b compares the OFF-state model against the measured data. The close agreement between the simulated and measured drift trajectories demonstrates that the proposed drift model accurately captures the statistical behavior of the silicon nanowire device and is therefore suitable for hardware-aware neural network simulations.

Impact of Drift Error on TNN Accuracy

After implementing the experimentally derived drift model within the custom APM SiNW SBFET device class, the drift mechanism was enabled in CrossSim and evaluated using the previously selected TNN architecture. Unlike programming error, which is introduced only once during the weight programming stage, drift continuously modifies the stored conductance values as a function of time. Consequently,

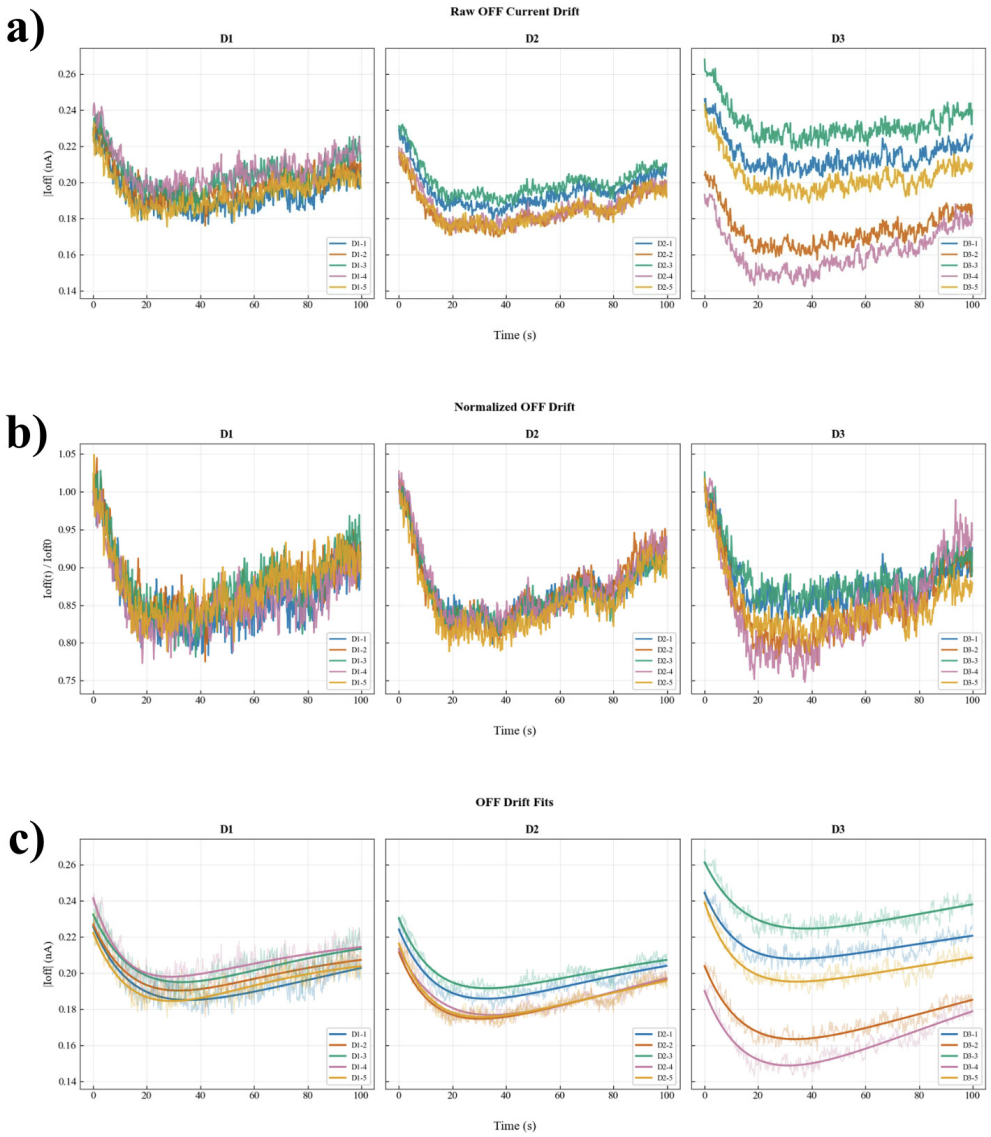


Fig. 5.22 OFF-state (HRS) drift characterization. (a) Raw measured current drift for three devices with five repeated measurements each over a period of 100 seconds. (b) Normalized OFF-state drift curves. (c) Double-exponential fits applied to the normalized OFF-state measurements.

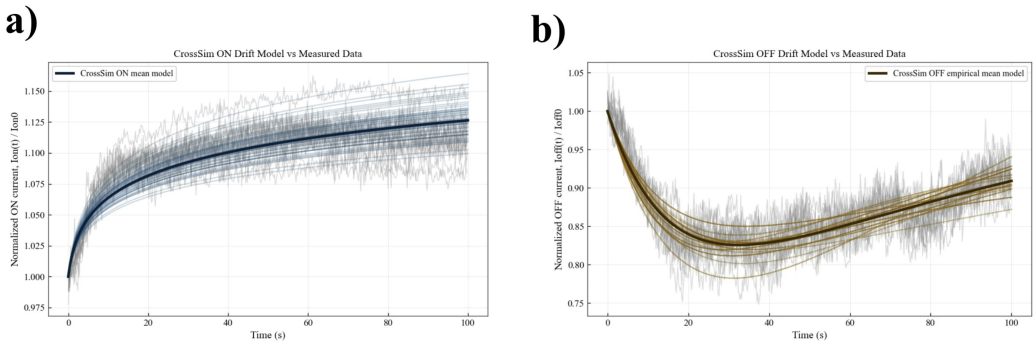


Fig. 5.23 Validation of the final CrossSim drift model against experimental measurements. (a) ON-state drift model comparison showing experimental measurements (gray), randomly sampled CrossSim realizations (blue), and the empirical mean model (dark blue). (b) OFF-state drift model comparison showing experimental measurements (gray), randomly sampled CrossSim realizations (brown), and the empirical mean model (dark brown).

the effective neural network weights evolve after programming, potentially affecting inference accuracy when the network is used at different retention times.

To investigate this effect, the TNN was evaluated at different time points after programming from $t = 0$ s to $t = 100$ s. The resulting accuracy trend is shown in Figure 5.24a. Interestingly, the classification accuracy does not show a steady decline as the retention time increases. Instead, small variations are observed around the baseline performance. This behavior is consistent with the stochastic nature of the drift model, where conductance changes are introduced as random perturbations rather than as a purely deterministic shift.

As drift changes the programmed conductance values, some weights may move away from their optimal values while others may, by chance, move slightly closer. Consequently, short-term fluctuations in accuracy can occur without a clear monotonic trend. From a practical perspective, the more significant effect of drift is not an immediate loss of accuracy but the gradual reduction in the stability and reproducibility of the stored network parameters over time.

The relatively small accuracy variation observed over the first 100 seconds indicates that the silicon nanowire device exhibits moderate short-term drift. For the evaluated TNN, the test accuracy remains within a narrow range between approximately 92.6% and 93.4%. This result suggests that short-term conductance drift alone does not severely degrade network performance. Nevertheless, the conduc-

tance states are no longer identical to the originally programmed values, and larger deviations may accumulate over longer retention times.

To gain further insight into the effect of conductance drift, Figures 5.24b–d show the relative weight changes in the three neural network layers over time. In each case, the initial weight distribution at $t = 0$ s is used as the reference and compared with the corresponding weights after 10, 25, 50, 75, and 100 seconds. As expected, the difference map at $t = 0$ s is entirely black which means there is no deviation from the original programmed state.

As the retention time increases, localized changes begin to appear across the weight maps and it is sign of gradual drift in the stored conductance values. These variations are distributed throughout the arrays rather than being confined to a specific region and it means that the effect is widespread across the network. The figures illustrate how the effective weights continue to evolve after programming, even though the overall network accuracy remains relatively stable over the time interval considered.

The effect can be observed in all three layers, although it is most noticeable in the third layer shown in Figure 5.24d. Since this layer is responsible for producing the final classification outputs, even small changes in the stored conductance values can influence the output logits and potentially affect the network prediction. While the drift-induced variations remain relatively modest over the time period examined here, they nevertheless indicate that the programmed conductance states continue to evolve after programming rather than remaining perfectly stable.

Overall, the results demonstrate that the proposed drift model successfully captures the temporal evolution of the silicon nanowire device and enables realistic hardware-aware simulations in CrossSim. While the short-term drift observed in this work causes only minor accuracy variations, the methodology established here provides a framework for evaluating longer retention times and future device generations. Additional visualizations, including the drift-induced conductance changes of the positive and negative cores, conductance distributions, and layer-by-layer drift statistics at 10 s, 25 s, 50 s, 75 s, and 100 s, are provided in Figures B.5, B.6, and B.7.

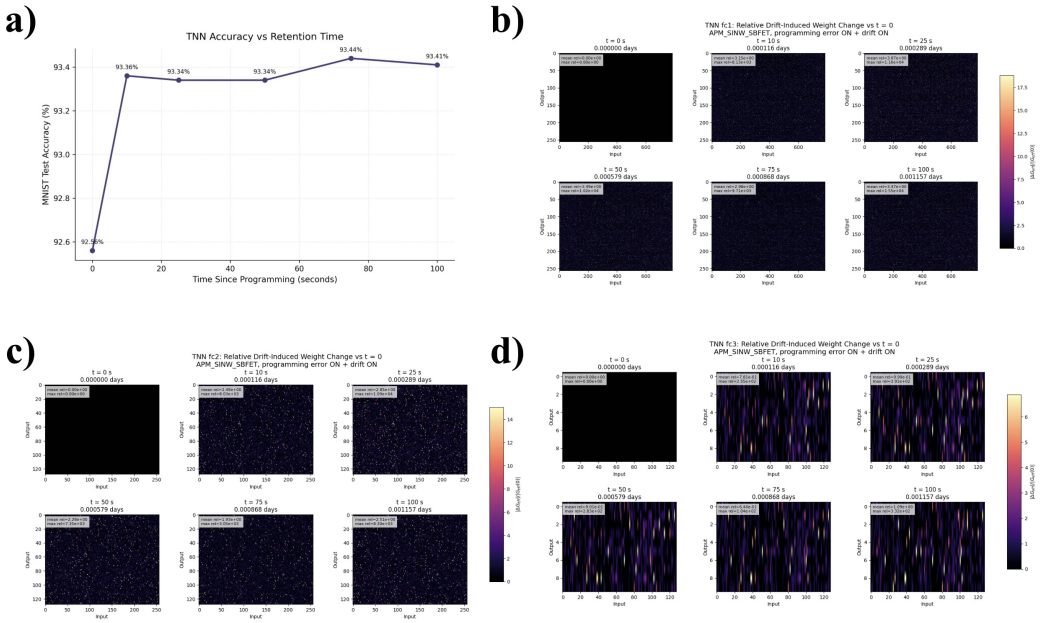


Fig. 5.24 Impact of conductance drift on TNN performance. (a) TNN classification accuracy evaluated at different retention times after programming. Relative drift-induced weight changes in (b) Layer 1 , (c) Layer 2 and (d) Layer 3 at different retention times.

5.4.2 System-Level Peripheral Non-Idealities

The previous sections investigated the impact of device-level non-idealities originating from the silicon nanowire device itself, including programming variability and conductance drift. However, practical neuromorphic accelerators also contain peripheral circuitry whose imperfections may significantly affect inference accuracy. In addition to the memristive crossbar arrays, analog neural network accelerators require DACs, ADCs, sensing circuitry, and interconnect networks. These components introduce additional quantization and signal degradation mechanisms that must be considered for realistic hardware evaluation.

Consequently, this section discussed on two important system-level non idealities, first peripheral quantization errors which are introduced by ADCs and DACs, and then voltage drops caused by parasitic wire resistances inside crossbar array.

ADC/DAC Quantization Effects

As discussed in previous chapter, DACs are responsible for converting digital input activations into analog voltages that can be applied to the crossbar rows, while ADCs convert the analog output currents generated by MVM back into digital values for subsequent neural network processing. Since both operations involve finite resolution quantization, they inevitably introduce information loss and quantization noise.

To better understand how the data converters influence network performance, an ADC/DAC optimization study was carried out using the selected TNN architecture together with the experimentally calibrated silicon nanowire device model. The programming-variability model introduced in the previous section was kept active throughout all simulations so that the results would reflect realistic device behavior. Different combinations of DAC resolution, ADC resolution, input range, and ADC operating range were then evaluated to determine how these parameters affect inference accuracy.

Specifically, DAC resolutions of 4 bits and 6 bits were chosed. For each DAC configuration, ADC resolutions of 2, 3, and 4 bits were considered. Input ranges of ± 4, ± 8, ± 16, and ± 32 were considered together with ADC ranges of ± 1, ± 2, and ± 4.

Figures 5.25a and 5.25b summarize the classification accuracy obtained using 4-bit and 6-bit DAC resolutions, respectively. A clear trend emerges from the results: the 4-bit DAC configuration is unable to maintain satisfactory inference performance, with the highest accuracy observed in this case remaining below 50%.

In the other hand, the 6-bit DAC configuration improves performance across all evaluated operating points. Furthermore, the results indicate that increasing ADC resolution from 2 bits to 4 bits produces a significant improvement in classification accuracy. Consequently, the 4-bit ADC configuration was selected for further investigation.

Among configurations, three high-performing candidates were selected using a 6-bit DAC, a 4-bit ADC, and an ADC range of ± 1, while varying only the input range. These candidate configurations are summarized in Figure 5.25c.

The best overall result was obtained using a 6-bit DAC, a 4-bit ADC, an input range of ± 32, and an ADC range of ± 1. Under these conditions, the TNN achieved a classification accuracy of 81.14%.

The corresponding accuracy degradation relative to the programming-error-only baseline of 92.56% is shown in Figure 5.25d. The optimized peripheral configuration introduces an accuracy loss of approximately 11.42 percentage points, demonstrating that peripheral quantization represents one of the dominant sources of performance degradation in the evaluated system.

To further illustrate this effect, Figure 5.25e compares the confusion matrices obtained from three different operating conditions: the digital TNN baseline, the ideal CrossSim implementation without ADC/DAC quantization, and the optimized quantized configuration. The increase in off-diagonal entries clearly indicates that quantization noise introduces additional classification errors across multiple digit classes.

It should be noted that the ADC and DAC models employed in this work are based on the generic quantizer models provided by CrossSim. More advanced converter architectures, including Successive Approximation Register (SAR) ADCs and SignMagnitude ADCs, may provide different accuracy–efficiency trade-offs and could be investigated in future work. The methodology required to incorporate such converter models and compare them against the generic quantizer approach is discussed in Figure C.1 in Appendix C.

Parasitic Resistance and Voltage-Drop Effects

In addition to peripheral quantization, practical crossbar arrays contain finite-resistance row and column interconnects. These parasitic resistances introduce voltage drops along the crossbar structure, causing deviations between the applied voltages and the voltages experienced by individual memory cells. Such effects are commonly referred to as IR-drop or voltage-drop errors.

To evaluate this phenomenon, the previously established TNN configuration was simulated while introducing parasitic row and column resistances within CrossSim. The baseline configuration corresponds to ideal interconnects with zero parasitic resistance and achieved an accuracy of 92.56%.

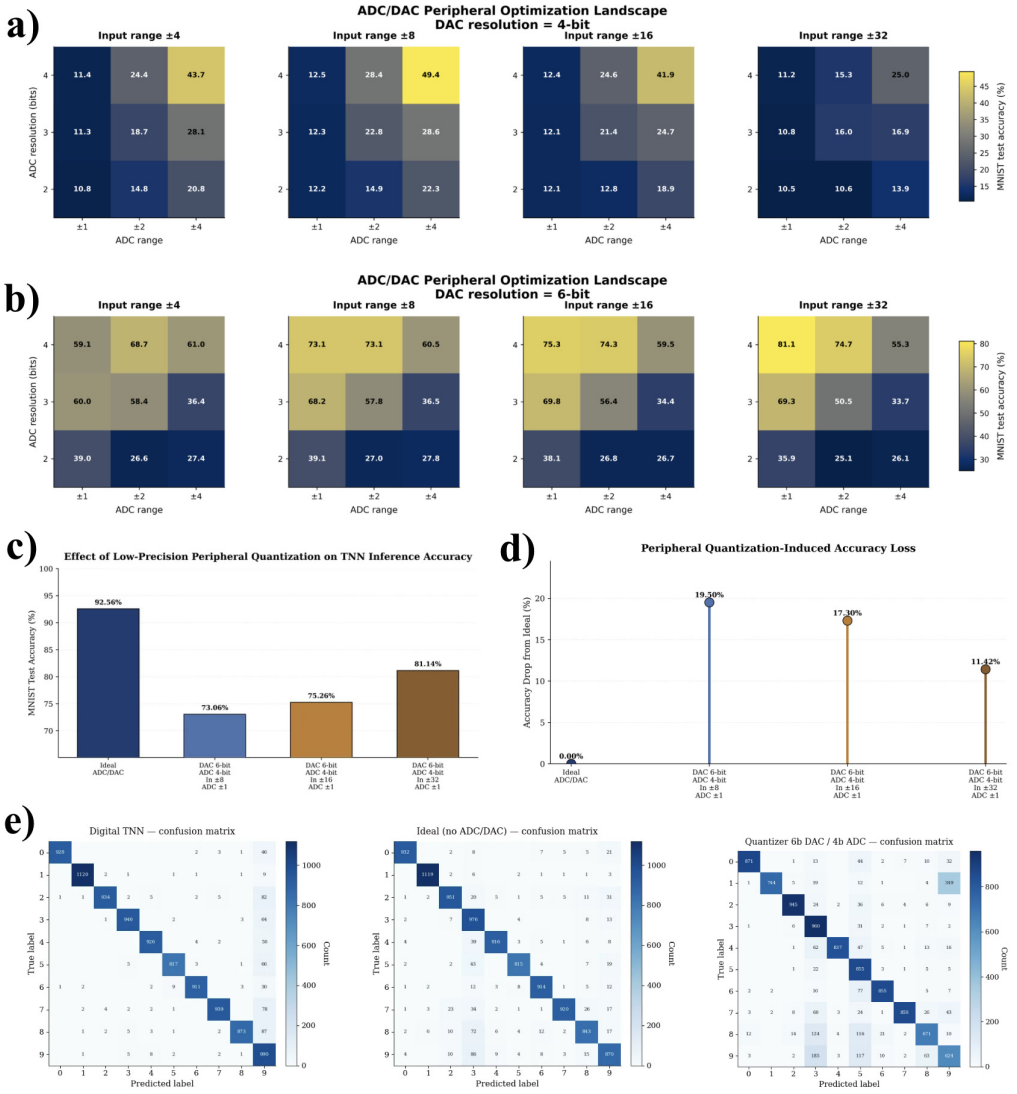


Fig. 5.25 ADC/DAC optimization study for the TNN implementation. (a) Classification accuracy obtained using a 4-bit DAC under different ADC resolutions, ADC ranges, and input ranges. (b) Classification accuracy obtained using a 6-bit DAC. (c) Comparison of the best-performing ADC/DAC candidates. (d) Accuracy degradation relative to the programming-error-only baseline. (e) Confusion matrices comparing the digital TNN baseline, ideal CrossSim implementation without quantization, and the optimized ADC/DAC configuration.

Four parasitic resistance values were evaluated:

$$R_p = 0.01\ \Omega, \quad 0.1\ \Omega, \quad 1\ \Omega, \quad 10\ \Omega.$$

The resulting classification accuracies are shown in Figure 5.26a. Interestingly, no measurable degradation in classification accuracy was observed across the entire investigated resistance range. The network accuracy remained essentially unchanged at approximately 92.56%.

This behavior can be explained by the extremely high resistance values of the silicon nanowire devices used in this work. The extracted conductance values are on the order of nanosiemens, corresponding to gigaohm-scale device resistances. Consequently, the currents flowing through the crossbar are extremely small, resulting in negligible voltage drops across the investigated interconnect resistances.

Although no observable accuracy degradation occurs, parasitic resistance still influences electrical operation. Figure 5.26b presents the estimated worst-case IR-drop as a function of parasitic resistance. As expected, the voltage drop increases approximately linearly with resistance. Therefore, while parasitic resistance does not significantly affect inference accuracy in the present system, it remains relevant from an electrical design and energy-efficiency perspective.

Because the investigated parasitic resistance values produced no measurable impact on classification accuracy while significantly increasing simulation time, ideal interconnects ($R_p = 0$) were assumed for all subsequent simulations presented in this thesis.

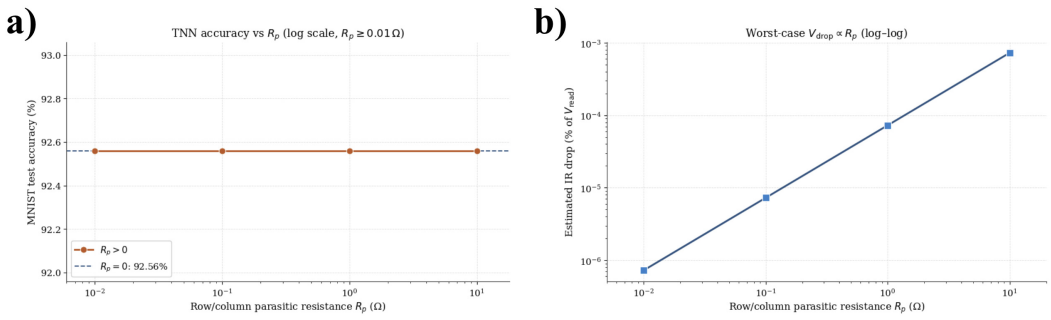


Fig. 5.26 Impact of parasitic wire resistance on TNN performance. (a) Classification accuracy as a function of row/column parasitic resistance.(b) Estimated worst-case IR-drop as a function of parasitic resistance.

5.4.3 Overall Impact of Non-Idealities

The results presented throughout this chapter provide a complete hardware-aware evaluation workflow, beginning from a digitally trained neural network and progressing toward a realistic implementation using experimentally characterized silicon nanowire devices.

The starting point was the digital TNN implemented and trained in PyTorch. After 100 training epochs, the network achieved a classification accuracy of 93.78% on the MNIST test dataset. The trained network was subsequently modeled in CrossSim platform and evaluated under ideal analog conditions, where all device-level and system-level non-idealities were disabled. Under these ideal conditions, the CrossSim implementation achieved the same classification accuracy of 93.78%, confirming that the digital PyTorch model and the CrossSim implementation were fully consistent and that the hardware mapping process introduced no unintended errors.

The first source of degradation was introduced by programming variability. Using the experimentally extracted cycle-to-cycle and device-to-device statistics obtained from the silicon nanowire measurements, a custom programming-error model was developed and integrated into CrossSim.The classification accuracy decreased from 93.78% to 92.56% once this non ideality was enabled.

In the next stage peripheral quantization effects introduced by the ADC and DAC circuits are considered. After evaluating a wide range of converter resolutions and operating ranges, the optimal configuration was identified as a 6-bit DAC, a 4-bit ADC, an input range of ± 32, and an ADC range of ± 1. Even with this optimized configuration, peripheral quantization remained the dominant source of performance degradation. The classification accuracy decreased from 92.56% to 81.14%, corresponding to an additional loss of approximately 11.42 percentage points.

Finally, parasitic row and column resistances were investigated to evaluate voltage-drop effects within the crossbar array. Owing to the extremely high resistance values of the silicon nanowire devices, the resulting currents were very small, leading to negligible IR-drop across the investigated interconnect resistances. Consequently, no measurable reduction in classification accuracy was observed, and the final accuracy remained unchanged at 81.14%.

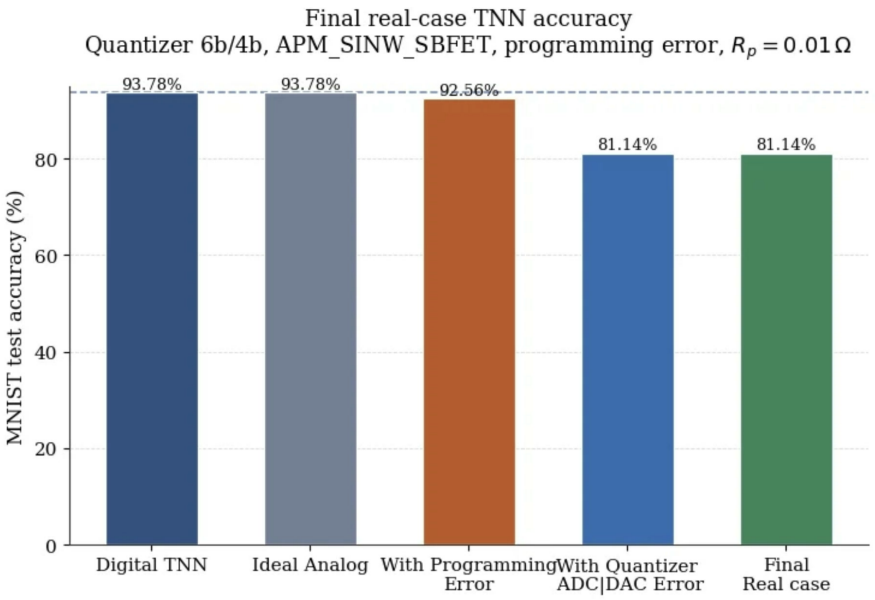


Fig. 5.27 Cumulative impact of device-level and system-level non-idealities on TNN inference accuracy. Starting from the digital TNN baseline, progressively more realistic hardware effects are introduced, including ideal CrossSim mapping, experimentally extracted programming variability, optimized ADC/DAC quantization, and parasitic interconnect resistance.

The cumulative impact of all considered non-idealities is summarized in Figure 5.27. The figure illustrates the progressive reduction in classification accuracy as increasingly realistic hardware effects are introduced into the simulation flow. Starting from the digital TNN baseline, the incorporation of experimentally measured programming variability, optimized ADC/DAC quantization, and parasitic interconnect effects ultimately results in a final hardware-aware inference accuracy of 81.14%.

These results demonstrate the importance of evaluating both device-level and system-level non-idealities when designing neuromorphic hardware. Although the silicon nanowire device itself introduces relatively modest degradation through programming variability, peripheral quantization effects represent a substantially larger challenge and dominate the overall accuracy loss observed in the present implementation.

5.5 Recovery of Accuracy Degradation in Silicon Nanowire Neuromorphic Circuits

As demonstrated in the previous sections, the digital TNN achieved an accuracy of 93.78%, while the inclusion of experimentally characterized device-level and system-level non idealities reduced the final hardware inference accuracy to 81.14%. Although this accuracy loss is primarily caused by programming variability and ADC/DAC quantization effects, it does not necessarily represent the ultimate performance limit of the silicon nanowire neuromorphic system.

To solve this challenge, two recovery strategies are presented in this work. The first approach is using silicon nanowire device itself as an analog nonlinear activation element and it cause to reduce the number of ADC/DAC conversions and therefore minimize quantization induced errors. The second approach is device-aware training, where hardware non idealities are incorporated directly into the training process, allowing the neural network to adapt its parameters to the physical behavior of the underlying hardware. The following sections evaluate these two approaches and their potential for recovering the accuracy lost due to hardware imperfections.

5.5.1 Using Silicon Nanowire Devices as Analog Neuromorphic Activation Functions

The previous sections demonstrated that peripheral ADC/DAC quantization is the dominant source of accuracy degradation in the present silicon nanowire implementation. Although programming variability reduced the TNN accuracy by only approximately 1.22 percentage points, the optimized ADC/DAC configuration introduced an additional loss exceeding 11 percentage points. This observation motivates the exploration of alternative architectures that reduce or eliminate intermediate analog-to-digital and digital-to-analog conversions.

One potential approach is to perform the nonlinear activation operation directly in the analog domain. Instead of converting the crossbar output current into a digital value, applying a digital activation function, and subsequently converting the result back into an analog signal, the activation operation could be implemented using the intrinsic nonlinear electrical characteristics of the silicon nanowire device itself.

Figure 5.28 illustrates the proposed concept. Input signals are first processed by a memristive crossbar array that performs analog MVM. The resulting output currents are then applied to a silicon nanowire device whose current-voltage characteristic exhibits a nonlinear transfer function. The activated analog current can subsequently be injected directly into the next crossbar layer without requiring intermediate ADC/DAC conversions.

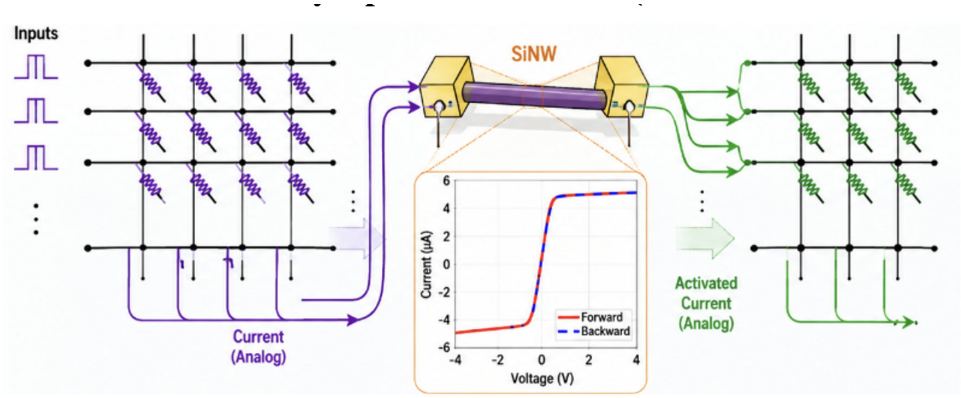


Fig. 5.28 Conceptual illustration of using a silicon nanowire device as an analog activation function between successive crossbar layers. The nonlinear current-voltage characteristic of the device can directly transform analog crossbar outputs into activated analog signals, potentially reducing the need for intermediate ADC/DAC conversions.

One interesting feature of current–voltage curves is the nonlinear response observed around zero bias. Similar nonlinear behavior is widely used in ANNs through activation functions, which help distinguish between weak and strong input signals. Although the device was not specifically designed for this purpose, the shape of its electrical response suggests that it could potentially be used as an analog activation element within neuromorphic computing architectures.

The main advantage of this approach is the elimination of multiple ADC and DAC stages between neural network layers. Since peripheral quantization was identified as the largest contributor to accuracy degradation, replacing digital activation blocks with device-level analog nonlinearities could significantly improve both computational efficiency and inference accuracy while simultaneously reducing energy consumption and circuit complexity.

At present, however, this concept remains a future research direction. A complete evaluation would require detailed modeling of the activation behavior, including cycle-to-cycle variability, device-to-device variability, temperature dependence, long-

term stability, and circuit-level integration. Furthermore, the nonlinear transfer characteristics must be incorporated into CrossSim or a similar hardware-aware simulation framework to quantify the achievable accuracy improvements. Nevertheless, the results suggest that silicon nanowire devices may serve not only as memory elements but also as active computational components for future neuromorphic systems.

5.5.2 Device-Aware Training for Accuracy Recovery

Although hardware non-idealities reduced the inference accuracy from 93.78% to 81.14%, this loss is not necessarily permanent. A widely adopted strategy in neuromorphic computing is device-aware training, where the neural network is retrained while incorporating hardware imperfections directly into the forward propagation process.

In conventional digital training, the forward pass assumes ideal arithmetic operations and ideal weights. Consequently, the optimization process never encounters the distortions that will later be present in the physical hardware. During inference on analog hardware, programming variability, conductance drift, ADC/DAC quantization, and other non-idealities perturb the network parameters and activations, causing a mismatch between training and deployment conditions.

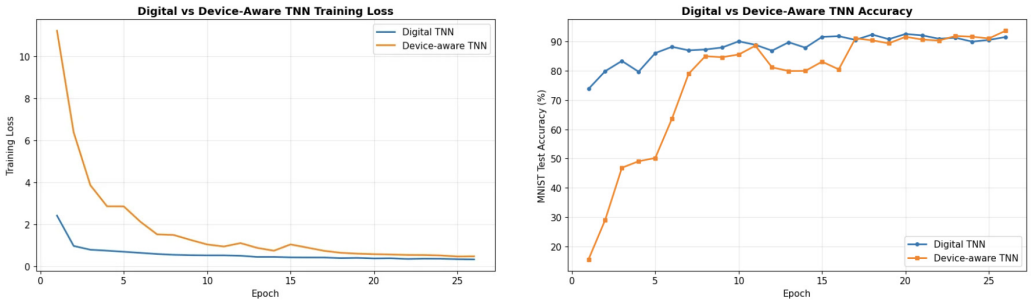


Fig. 5.29 Training evolution of the digital TNN and the device-aware TNN.

Device-aware training addresses this problem by embedding the hardware model inside the training loop. During the forward pass, CrossSim replaces the ideal computations with hardware-aware computations that include experimentally characterized programming errors and the optimized ADC/DAC quantization model.

The resulting non-ideal outputs are then used to compute the loss function. During backpropagation, the network updates its parameters to compensate for these hardware-induced distortions. As training progresses, the network gradually learns weight configurations that are more robust to the underlying device and circuit imperfections.

For this study, a device-aware TNN was trained using a batch size of 64, a learning rate of 10^{-4}, a ternarization threshold of $\Delta = 0.05$, a smooth threshold-window width of $r = 0.7$, and 26 training epochs. The training and validation curves are presented in Figure 5.29. Compared with the ideal digital TNN, the device-aware model initially exhibits significantly lower accuracy because the hardware non-idealities are already present during the forward pass. However, as training proceeds, the network progressively adapts to these distortions and recovers a substantial portion of the lost accuracy.

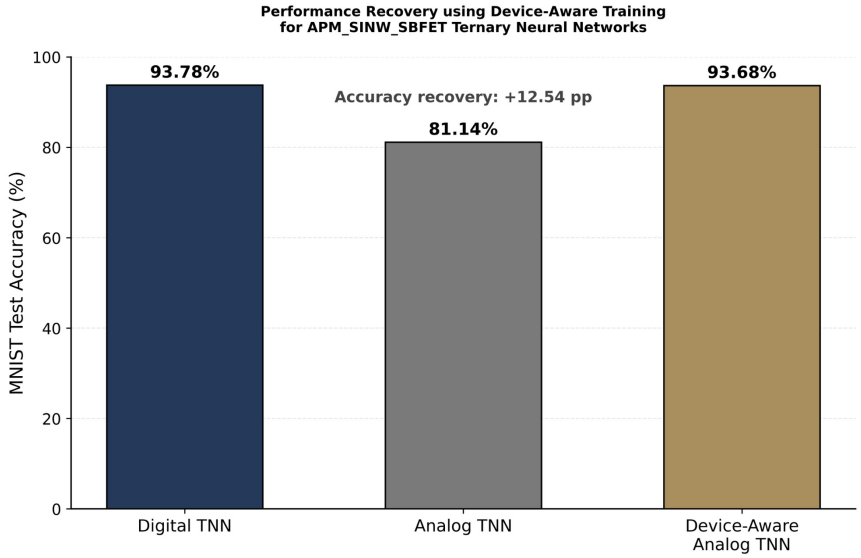


Fig. 5.30 Performance recovery using Device-aware training for APM SINW SBFET TNNs.

The final results demonstrate a remarkable recovery in performance. The hardware-aware inference accuracy increased from 81.14% to 93.96%, corresponding to an improvement of approximately 12.82 percentage points. This recovered accuracy is even slightly higher than the original digital TNN baseline of 93.78%, indicating that the network successfully learned weight distributions that compensate for the hardware imperfections introduced by the silicon nanowire implementation.

The overall recovery process is summarized in Figure 5.30. These results highlight one of the most important advantages of device-aware training: rather than attempting to eliminate every hardware non-ideality through improved fabrication or circuit design, the neural network itself can learn to adapt to the imperfections of the physical hardware.

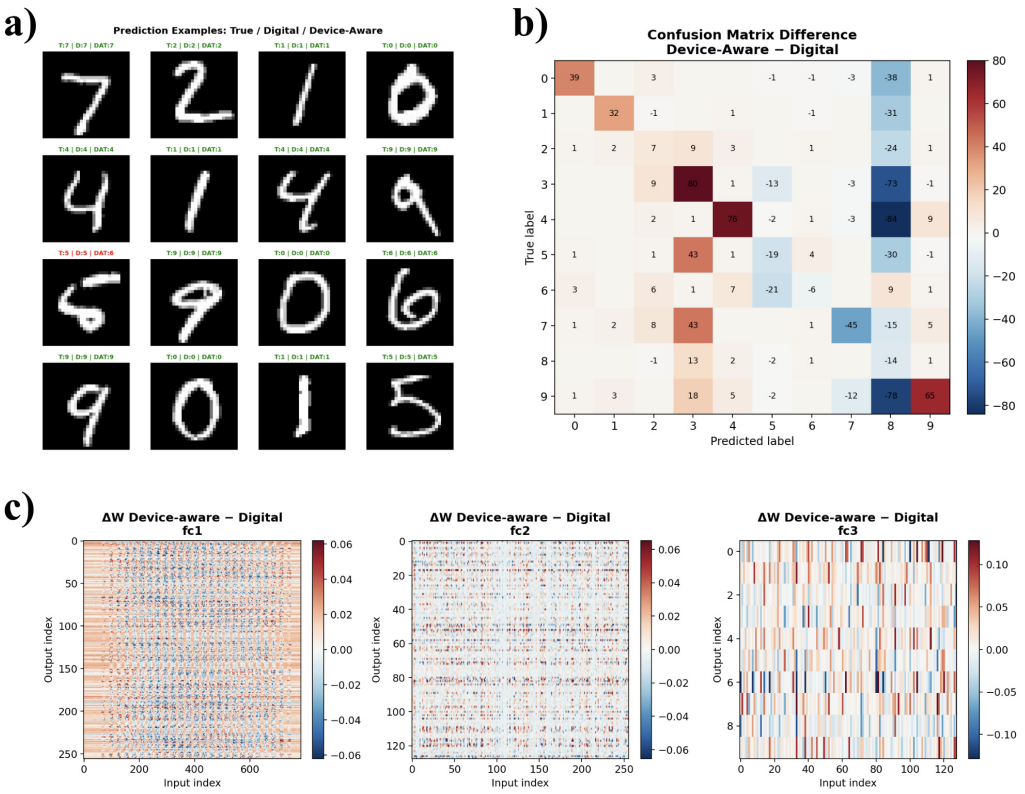


Fig. 5.31 a) Classification performance comparison between the digital TNN and the device-aware trained TNN. b)confusion matrix difference of device-aware and digital network, c)Heat map of difference between device-aware and digital network.

Further insight is provided by the representative inference examples shown in Figure 5.31(a). Several samples that are incorrectly classified after direct hardware mapping are correctly recognized following device-aware training, demonstrating that the network successfully learns to compensate for the distortions introduced by the analog hardware model. These qualitative examples illustrate that incorporating realistic device characteristics during optimization enables the network to recover many of the prediction errors caused by hardware non-idealities while preserving the overall classification behavior of the original digital model.

This improvement is further quantified by the confusion matrix difference presented in Figure 5.31(b). Positive values along the main diagonal indicate an increase in correctly classified samples for multiple digit classes, whereas negative values in several off-diagonal entries correspond to a reduction in misclassifications after device-aware retraining. Rather than improving only a few isolated categories, the recovered performance is distributed across the entire classification task, demonstrating that the learned compensation generalizes well to different classes.

Additional insight is obtained from the weight-difference heat maps shown in Figure 5.31(c). The parameter updates are distributed throughout all three fully connected layers, although the largest changes are observed in the earlier layers responsible for feature extraction. This behavior indicates that the network compensates for hardware imperfections by reorganizing its internal feature representations instead of relying solely on adjustments in the final classification layer. Consequently, the learned representations become more compatible with the characteristics of the physical analog hardware.

Figure 5.32 shows the weight distributions obtained after device-aware training for each of the three network layers. The left column corresponds to the latent (shadow) weights that are updated during optimization. The middle column shows the ideal ternary weights obtained after thresholding, while the right column presents the corresponding analog weights after mapping to the silicon nanowire hardware model.

Compared with the original digital TNN, the device-aware network exhibits noticeably different weight distributions. These changes arise because the optimization process is no longer performed solely for an ideal software implementation. Instead, the training procedure explicitly incorporates the measured characteristics of the silicon nanowire memristive devices, including quantization effects and programming variability. As a result, the learned weights become better aligned with the physical behavior of the target hardware, allowing the mapped network to preserve its decision boundaries despite the presence of analog non-idealities.

The results presented in Figures 5.31 and 5.32 demonstrate that hardware-aware training is one of the most effective approaches for mitigating the accuracy degradation introduced by analog hardware imperfections. Rather than treating device variability as an unavoidable limitation, the training algorithm directly incorporates realistic hardware models into the optimization process, enabling the network

to learn parameters that are inherently robust to the characteristics of the physical devices. Consequently, much of the performance loss observed after direct digital-to-hardware mapping can be recovered without requiring additional hardware calibration or complex compensation circuitry.

Overall, these results highlight the importance of algorithm–hardware co-design for practical neuromorphic computing systems. As analog in-memory computing technologies continue to evolve, completely eliminating device non-idealities through fabrication improvements alone becomes increasingly challenging. Hardware-aware training instead provides a scalable and efficient solution by allowing the neural network to adapt to the hardware during learning rather than after deployment. The substantial recovery in classification accuracy achieved in this work confirms that incorporating realistic device models during training is essential for translating high digital performance into reliable and energy-efficient neuromorphic hardware implementations.

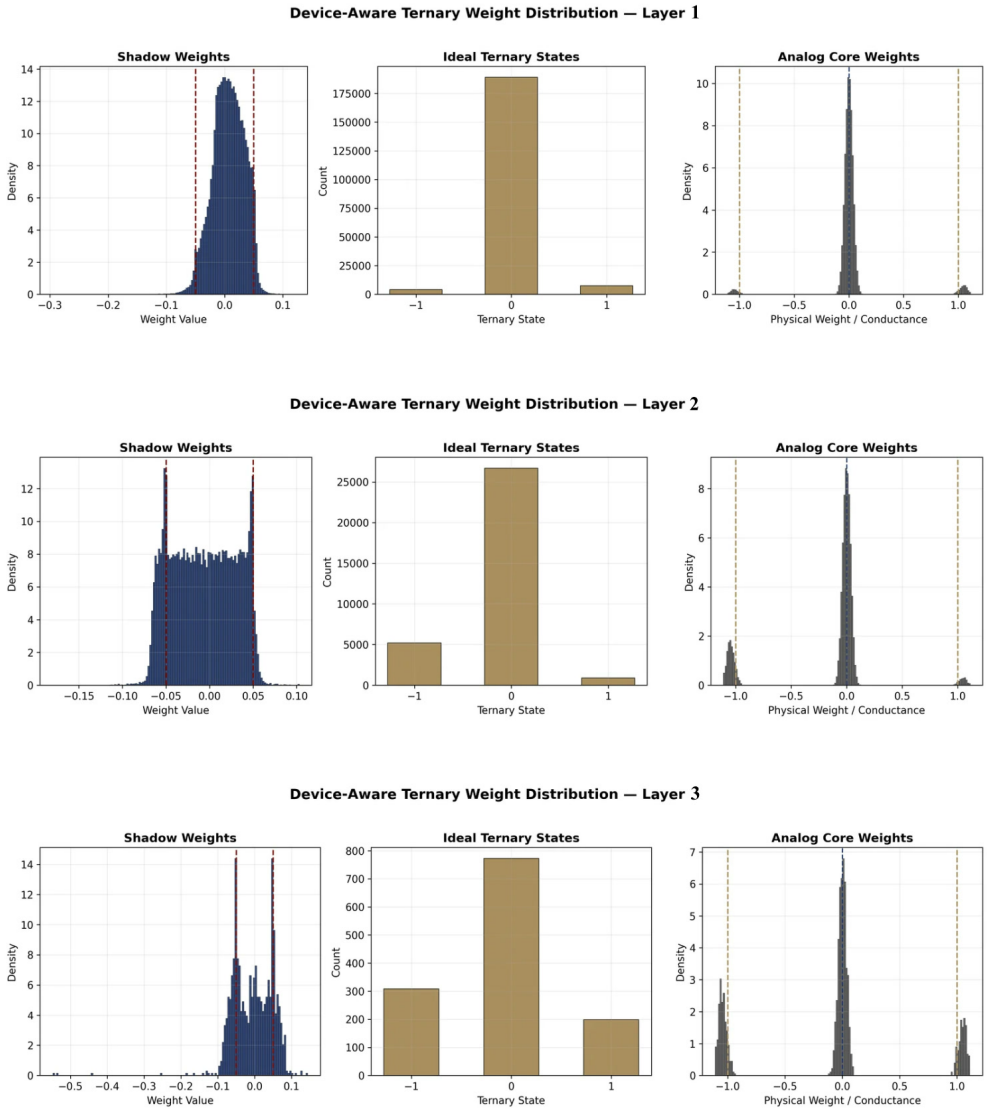


Fig. 5.32 Weight distributions of the device-aware trained TNN for all three network layers. The left column shows the latent shadow weights optimized during training, the middle column shows the corresponding ideal ternary states after thresholding, and the right column shows the physical analog weights after mapping to the silicon nanowire hardware model.

Chapter 6

Toward Silicon Nanowire Neuromorphic Biosensing Systems

6.1 Silicon Nanowire Memristive Biosensors for Neuromorphic Signal Generation

6.1.1 Motivation for Pulse-Based Neuromorphic Biosensing

Neuromorphic computing systems process information using discrete temporal events rather than continuously sampled signals. This event-driven mode of operation offers significant advantages in terms of energy efficiency, computational sparsity, and compatibility with biological information processing mechanisms. This has motivated the development of sensing devices that can produce spike-like signals directly at the sensor level. By generating outputs that are already compatible with neuromorphic hardware, these devices can decrease the need for extensive signal processing before computation.

Among emerging nanoelectronic devices, volatile memristors are particularly interesting because they can naturally generate temporary conductive states in response to electrical stimulation. Traditional sensing systems usually require several processing steps, including signal acquisition, analog-to-digital conversion, and digital analysis. Volatile memristive devices, however, can respond to physical or

biochemical events by directly producing temporal pulse streams, creating a closer link between sensing and information processing.

When combined with capacitive elements and operated under constant bias conditions, volatile memristors can exhibit self-sustained oscillatory behavior, generating sequences of electrical spikes without the need for external clocking circuitry [72, 73]. Such pulse-generating characteristics make them highly attractive candidates for neuromorphic sensing systems, where information is represented through spike timing and firing frequency rather than signal amplitude.

In this work, silicon nanowire memristive devices are investigated as a platform capable of bridging biosensing and neuromorphic computation. The fundamental concept is to exploit the interaction between biochemical species and the electrical dynamics of the memristive device, allowing molecular information to be encoded directly into pulse-domain signals suitable for subsequent processing by SNNs.

6.1.2 Memristive Biosensor Structure and Operating Principle

The silicon nanowire memristive biosensor employed in this work is based on a silicon nanowire channel contacted by nickel electrodes and fabricated on a silicon dioxide substrate. As illustrated in Fig. 6.1(a), the nanowire surface can be functionalized with biological recognition elements, such as antibodies, enabling selective binding of target biomolecules [74, 75].

When target molecules are captured on the nanowire surface, local charge redistribution occurs near the sensing channel. These electrostatic changes influence the electrical properties of the device and modify its memristive switching dynamics. As a result, the biochemical interaction is translated into measurable electrical variations.

Previous experimental investigations demonstrated that the electrical behavior of the functionalized device can be represented by an equivalent circuit consisting of a volatile memristive element connected in parallel with an effective capacitance, as shown in Fig. 6.1(b) [57]. The effective capacitance represents the combined contribution of device parasitics, interface effects, and biomolecular interactions occurring at the sensing surface.

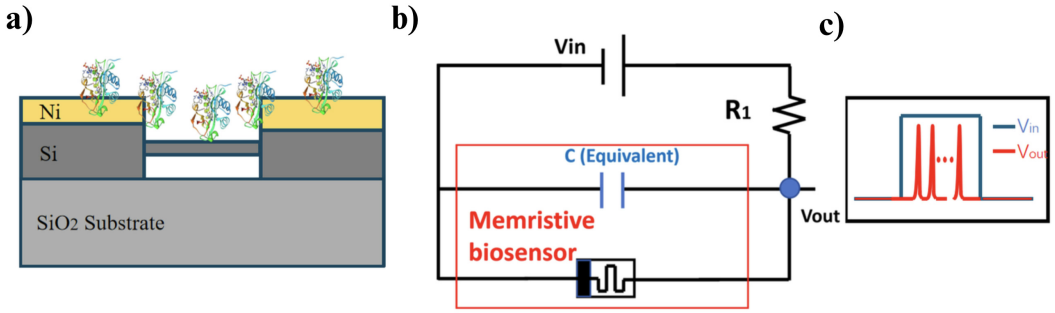


Fig. 6.1 Silicon nanowire memristive biosensor and pulse-generation mechanism. (a) Functionalized SiNW biosensor for selective biomarker sensing. (b) Equivalent electrical circuit composed of a volatile memristor and an effective capacitance. (c) Pulse-domain output generated through self-sustained oscillatory dynamics under DC excitation.

Under an appropriate Direct Current(DC) bias, the interaction between the memristive switching mechanism and the parallel capacitance produces self-sustained oscillatory dynamics. Instead of generating a steady-state electrical output, the device periodically transitions between conductive and non-conductive states, resulting in a train of electrical pulses. A representative pulse output is illustrated in Fig. 6.1(c).

The ability to directly generate spike-like electrical signals is particularly important from a neuromorphic perspective. Rather than measuring continuous analog quantities and subsequently converting them into spike representations, the sensing device itself performs the signal encoding process. Consequently, the generated pulse stream can be directly interfaced with SNNs, enabling a compact and energy-efficient sensing-to-processing architecture.

6.1.3 Biomarkers and PSA-Based Sensing

Biomarkers are measurable biological molecules that provide information about physiological or pathological conditions within the body. They are widely used in medical diagnostics, disease monitoring, and risk assessment. However, in many clinical applications, the concentration of a single biomarker is often insufficient to accurately characterize complex disease states. Instead, diagnostic decisions frequently rely on the relationships among multiple biomarkers, where the relative balance between different molecular indicators provides more meaningful information than their individual concentrations alone [76–79].

A representative example is prostate cancer screening, where prostate-specific antigen (PSA) is commonly used as a diagnostic biomarker. In clinical practice, both total PSA (tPSA) and free PSA (fPSA) are measured, and the ratio between these quantities serves as an important indicator for estimating cancer risk [14, 15]. Lower free-to-total PSA ratios are generally associated with a higher probability of prostate cancer, whereas larger ratios indicate a lower risk category. The commonly used risk classification ranges are summarized in Table 6.1.

Table 6.1 Free-to-total PSA ratio, corresponding prostate cancer risk categories, and approximate probabilities [14, 15].

Free-to-Total PSA Ratio	**Risk Category**	**Approximate Prostate Cancer Probability**
$< 10\%$	High Risk	Approximately 55%
$10\% - 15\%$	Moderately High Risk	Approximately $28\% - 35\%$
$15\% - 25\%$	Intermediate Risk	Approximately $16\% - 20\%$
$> 25\%$	Low Risk	Approximately $5\% - 9\%$

The silicon nanowire memristive biosensor investigated in this work provides an alternative sensing paradigm. Following biofunctionalization of the nanowire surface with selective biological receptors, target biomolecules influence the internal electrical dynamics of the device, enabling biochemical information to be represented directly through pulse timing and firing frequency rather than conventional analog voltage levels. This pulse-based representation naturally creates an interface between biosensing devices and neuromorphic computing systems.

6.1.4 Pulse Generation Model

Building upon the operating principle introduced in Fig. 6.1, the oscillatory behavior of the silicon nanowire memristive biosensor can be described mathematically using the equivalent memristor–capacitor model. Previous studies have shown that biomolecular interactions involving PSA modify the effective capacitance of the sensing structure, thereby altering the pulse-generation dynamics and firing frequency [80, 57].

Under DC excitation, the charging and discharging dynamics of the equivalent circuit determine the duration of each oscillation cycle. The corresponding charging and discharging times are given by

$$t_{\uparrow} = T_{\text{off}} \ln\left(\frac{V_{\infty}^{\text{off}} - V_{\text{hold}}}{V_{\infty}^{\text{off}} - V_{\text{th}}}\right), \qquad t_{\downarrow} = T_{\text{on}} \ln\left(\frac{V_{\text{th}} - V_{\infty}^{\text{on}}}{V_{\text{hold}} - V_{\infty}^{\text{on}}}\right), \tag{6.1}$$

where V_{th} and V_{hold} denote the threshold and holding voltages of the memristive device, respectively. The parameters T_{on} and T_{off} represent the effective RC time constants associated with the conductive and non-conductive states of the device.

$$V_{\infty}^{\text{off}} = \frac{V_s R_{\text{off}}}{R + R_{\text{off}}}, \qquad V_{\infty}^{\text{on}} = \frac{V_s R_{\text{on}}}{R + R_{\text{on}}}, \tag{6.2}$$

where V_s is the applied bias voltage, R is the series resistance, and R_{on} and R_{off} are the low-resistance and high-resistance states of the memristive device, respectively.

The pulse firing frequency is obtained as the inverse of the oscillation period,

$$f = \frac{1}{t_{\uparrow} + t_{\downarrow}}. \tag{6.3}$$

Experimental studies have further shown that PSA concentration affects the effective capacitance of the biosensor and therefore influences the pulse generation frequency. Based on previously extracted device parameters, the pulse event rate can be approximated by the empirical log-linear relationship

$$\log_{10}(f) \approx 0.60 \log_{10}([\text{PSA}]) + 6.63, \tag{6.4}$$

where [PSA] denotes the PSA concentration. This relationship indicates that increasing biomarker concentration leads to a corresponding increase in pulse firing frequency.

In the present work, Eq. (6.4) is used to generate representative synthetic pulse streams that emulate experimentally observed device behavior. These pulse trains provide the event-based inputs used in the subsequent neuromorphic processing framework.

6.1.5 Overall Neuromorphic Front-End Architecture

The overall sensing-to-decision workflow is illustrated in Fig. 6.2. Biomolecular interactions occurring at the functionalized silicon nanowire surface modify the electrical dynamics of the volatile memristive device, producing pulse streams whose firing characteristics depend on biomarker concentration. These pulse trains are then applied directly to the SNN, where temporal integration and classification are performed to determine the final prostate cancer risk category.

Unlike conventional biosensing systems that rely on analog signal acquisition, digitization, and feature extraction, the proposed architecture keeps information in the temporal domain with sensing and processing chain. Consequently, pulse generation, signal representation, and classification are integrated within a unified neuromorphic framework.

The proposed front-end demonstrates the feasibility of combining silicon nanowire memristive biosensors with neuromorphic computing systems, providing a compact sensing-to-computing pathway for future bio-integrated platforms capable of near-sensor intelligence and real-time biomedical decision support.

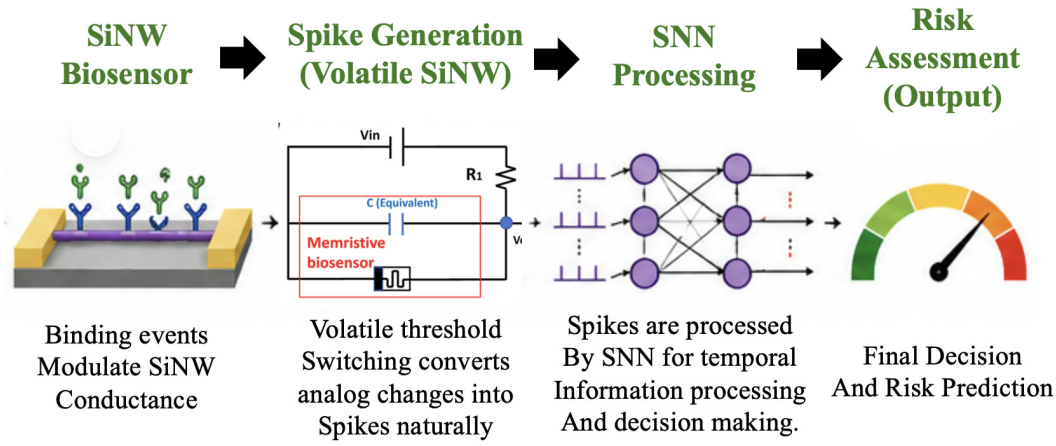


Fig. 6.2 Detailed sensing-to-decision workflow of the proposed neuromorphic biosensing system. Biomolecular binding events modulate the electrical dynamics of the silicon nanowire biosensor, resulting in spike generation through volatile memristive switching. The resulting pulse streams are processed by a SNN, which performs temporal integration and classification to generate the final risk assessment.

Because the sensor output is represented as a sequence of spikes, an appropriate computational framework is needed to interpret this temporal information. SNNs

are particularly suitable in this context, since they operate directly on discrete spike events and retain the timing information carried by the input signals during inference. The following section therefore introduces the fundamental concepts underlying SNNs before presenting the proposed classification architecture.

6.2 Spiking Neural Network (SNN) Fundamentals

6.2.1 Introduction to SNNs

SNNs are the third generation of ANNs and are inspired by the communication mechanisms of biological nervous systems [81, 82]. Unlike conventional ANNs, which exchange continuous-valued activations, neurons in SNNs communicate through discrete spike events. As a result, information processing is inherently event-driven and evolves over time.

The incorporation of temporal dynamics allows SNNs to represent and process information using sequences of spikes rather than static numerical values. This characteristic makes SNNs particularly attractive for neuromorphic computing systems, where computation and communication are performed using biologically inspired principles.

6.2.2 Neuron Dynamics and Information Encoding

As illustrated in Fig. 6.3(a), an SNN neuron receives spike-based inputs, integrates synaptic currents, and generates an output spike when its membrane potential exceeds a firing threshold. Following spike generation, the membrane potential is reset and the integration process begins again.

Unlike conventional neural networks, where information is represented solely by activation magnitude, SNNs can encode information through firing rate, spike timing, spike latency, or temporal firing patterns. Consequently, both the occurrence and timing of spikes contribute to information representation and processing.

Because neurons remain largely inactive in the absence of input events, SNNs naturally support sparse computation and can achieve high energy efficiency when implemented on neuromorphic hardware platforms [83].

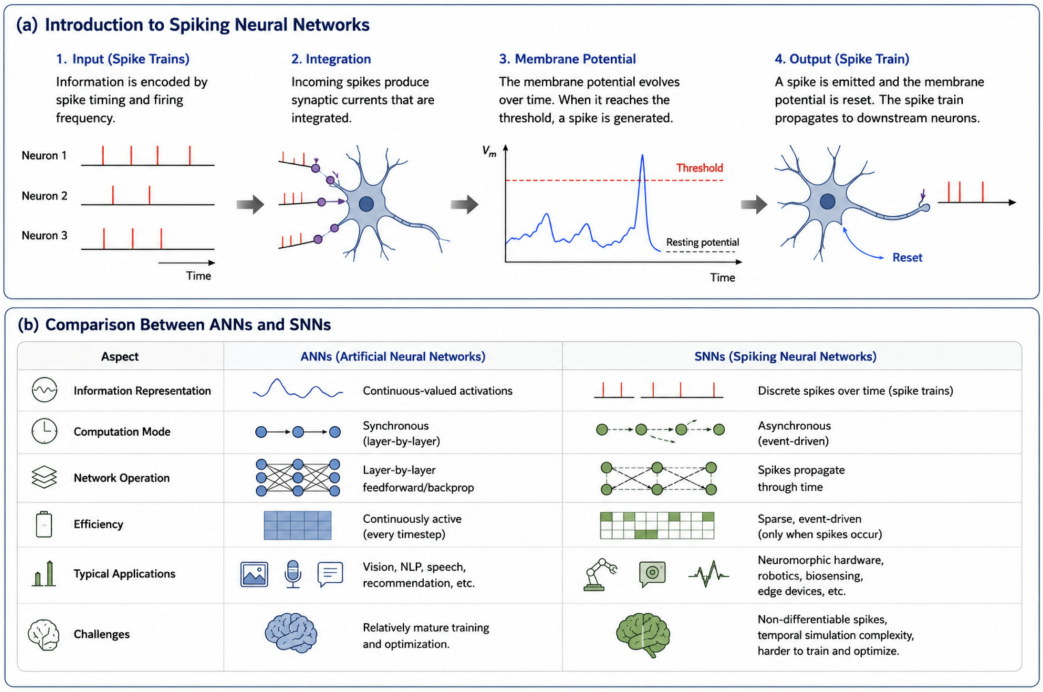


Fig. 6.3 Fundamental concepts of SNNs. (a) Illustration of spike-based information processing, including spike-train inputs, temporal integration, membrane potential evolution, threshold crossing, and output spike generation. (b) Comparison between conventional ANNs and SNNs in terms of information representation, computation model, network operation, efficiency, applications, and training challenges.

The differences between conventional ANNs and SNNs are summarized in Fig. 6.3(b). Unlike ANNs, which operate using continuous-valued activations and synchronous layer-by-layer computations, SNNs employ event-driven processing based on discrete spike events and explicitly incorporate temporal dynamics into the computation process.

6.2.3 Applications of SNNs

The ability of SNNs to process temporal information has led to their application in a wide range of domains, including computer vision, speech recognition, robotics, autonomous systems, and event-based sensing [84]. In many of these applications, information arrives as asynchronous events rather than continuously sampled signals, making spike-based processing particularly effective.

Biomedical sensing represents another promising application area. Physiological and biochemical processes often exhibit temporal dynamics that can be naturally represented through spike trains [85].

6.2.4 Learning Paradigms in SNNs

Training SNNs presents unique challenges because the spike-generation process is non-differentiable, preventing the direct application of conventional backpropagation algorithms. To address this limitation, several learning approaches have been developed.

Supervised surrogate-gradient learning is currently the most widely adopted training strategy because it approximates gradients during backpropagation while preserving spike-based neuron dynamics [86]. Alternative approaches include biologically inspired learning rules such as STDP, which adjusts synaptic weights according to the relative timing of pre- and post-synaptic spikes [87], as well as ANN-to-SNN conversion techniques that transform a trained ANN into an equivalent spiking implementation [88].

In this work, surrogate-gradient learning is employed because it enables efficient end-to-end supervised optimization while maintaining the temporal processing characteristics of SNNs.

6.2.5 Neuron Model Used in This Work

A variety of neuron models have been developed for SNNs, ranging from biologically detailed formulations such as the Hodgkin–Huxley model [89] to computationally efficient integrate-and-fire models [90]. For practical neuromorphic applications, simplified neuron models are often preferred because they significantly reduce computational complexity while preserving the essential characteristics of spike-based information processing.

In this work, the Leaky Integrate-and-Fire (LIF) neuron model is employed [91]. The LIF model is widely used in neuromorphic computing because of its simplicity, computational efficiency, and ability to capture the fundamental temporal behavior of biological neurons.

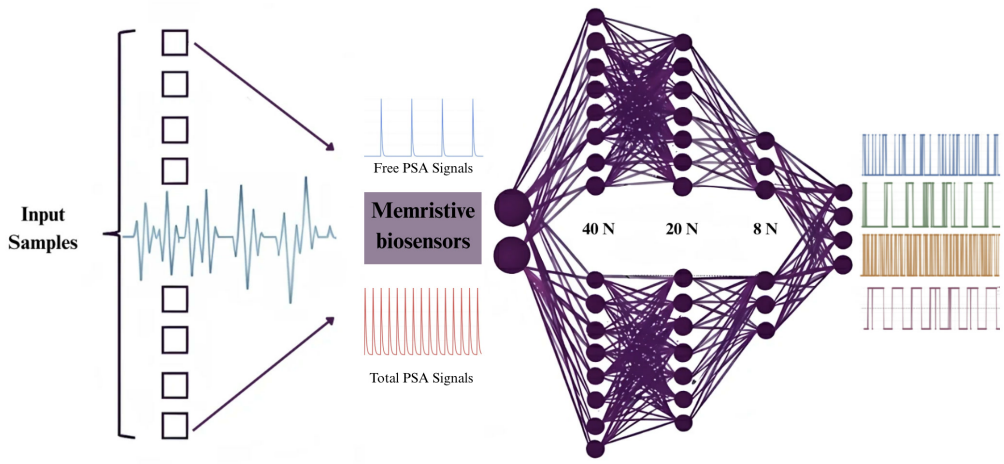


Fig. 6.4 Architecture of the proposed SNN. Pulse streams generated from fPSA and tPSA measurements are transformed into synaptic currents and processed through multiple hidden layers of LIF neurons for prostate cancer risk classification [13].

The membrane potential is described by

$$\tau_m \frac{dV(t)}{dt} = -(V(t) - V_{rest}) + RI(t), \tag{6.5}$$

where $V(t)$ is the membrane potential, V_{rest} is the resting potential, $I(t)$ is the input current, R is the membrane resistance, and τ_m denotes the membrane time constant. When the membrane potential reaches a predefined threshold, an output spike is generated and the membrane potential is reset. The resulting dynamics provide a computationally efficient mechanism for processing temporal spike sequences.

The concepts introduced in this section form the foundation for the neuromorphic classifier developed in this work. The following section presents the proposed SNN architecture used for pulse-based prostate cancer risk classification.

6.3 SNN for Pulse-Based Risk Classification

6.3.1 Proposed SNN Architecture

The pulse streams generated by the silicon nanowire memristive biosensors are processed using the SNN shown in Fig. 6.4. The network receives two input channels corresponding to fPSA and tPSA pulse streams and performs temporal feature extraction for prostate cancer risk classification.

Prior to entering the network, the spike trains undergo a synaptic filtering stage that converts discrete pulse events into continuous synaptic currents. This operation approximates temporal integration in biological synapses and facilitates information propagation through the network.

The proposed architecture consists of three fully connected hidden layers containing 40, 20, and 8 neurons, respectively. The output layer contains four neurons corresponding to the four prostate cancer risk categories defined in Table 6.1. LIF neurons are employed throughout the network because they provide an effective balance between biological realism and computational efficiency.

The network was implemented in Python using the PyTorch deep-learning framework [70]. Spiking neuron models and surrogate-gradient learning were implemented using the SNNtorch library [92].

6.3.2 Synaptic Filtering and Temporal Processing

The pulse streams generated by the memristive biosensors consist of discrete spike events. To model synaptic current dynamics, each spike train is passed through a first-order low-pass filter that produces exponentially decaying current responses between consecutive spikes. This filtering stage is introduced to approximate the continuous-time synaptic integration process and to provide a suitable input representation for the downstream spiking neurons.

The synaptic current $x(n)$ at discrete time step n is described by

$$x(n) = a\,x(n-1) + s(n), \tag{6.6}$$

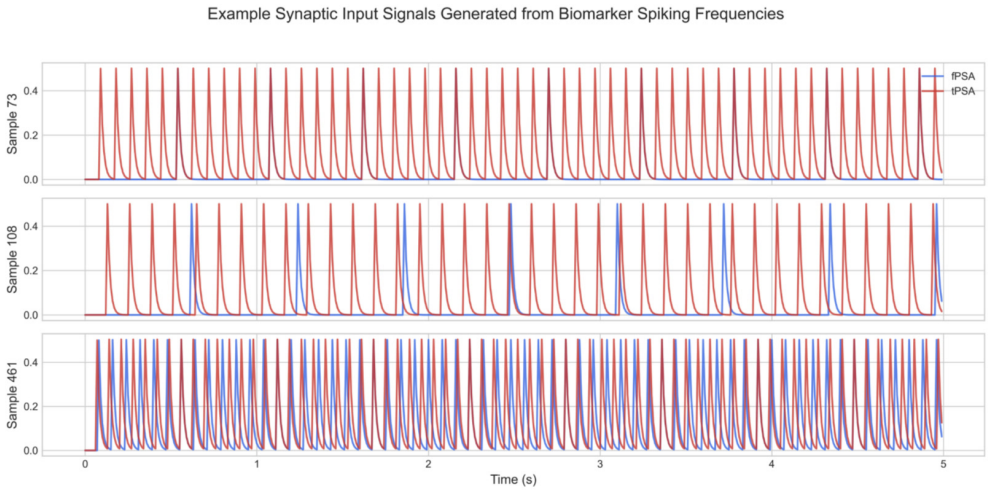


Fig. 6.5 Representative filtered synaptic current signals obtained from pulse-encoded PSA measurements. The blue traces correspond to fPSA inputs and the red traces correspond to tPSA inputs. Variations in pulse frequency result in distinct temporal current patterns that are subsequently processed by the SNN [13].

where a denotes the decay coefficient and $s(n)$ represents the input spike event at time step n.

This filtering operation preserves the temporal information contained within the original pulse stream while generating a continuous synaptic current suitable for neuronal integration. Representative filtered signals are shown in Fig. 6.5. Different biomarker concentrations result in different pulse frequencies and therefore distinct temporal current patterns.

Once the pulse streams have passed through the synaptic filters, the resulting currents are supplied to the LIF neurons. These neurons integrate the incoming activity over time and emit spikes when sufficient input has accumulated. As the spike signals move through successive network layers, meaningful temporal patterns begin to emerge from the pulse-encoded biomarker data. The network then uses these learned patterns to distinguish between the different prostate cancer risk categories.

6.3.3 Network Training and Decision Decoding

The proposed SNN is trained in a supervised manner to associate pulse-encoded PSA measurements with the corresponding prostate cancer risk categories.

For each input sample, spiking activity is observed over a fixed temporal window of duration T. Let r_k denote the average firing rate of the k-th output neuron during the observation interval. A rate-based loss function is employed to align the output activity with the target class,

$$L = \sum_k (r_k - y_k)^2, \tag{6.7}$$

where y_k represents the one-hot encoded target vector associated with the correct risk category.

Training is performed using backpropagation through time (BPTT) together with the surrogate-gradient framework introduced in the previous section. This enables end-to-end optimization of network parameters over the complete temporal observation window while maintaining spike-based computation during inference.

During inference, output spikes are accumulated throughout the observation window and a rate-based decoding strategy is employed. The predicted risk category is determined by selecting the output neuron exhibiting the highest average firing rate. Through this approach, classification is performed directly on pulse-domain information without reconstructing continuous biomarker concentrations or explicitly calculating biomarker ratios.

6.3.4 Dataset and Evaluation Protocol

A synthetic dataset consisting of 500 samples was generated for system-level evaluation. Each sample contains two pulse-encoded biomarker channels corresponding to fPSA and tPSA measurements and is assigned to one of four clinically defined risk categories based on the free-to-total PSA ratio presented in Table 6.1. The dataset was designed to provide balanced representation across all risk categories, with each class containing an equal number of samples.

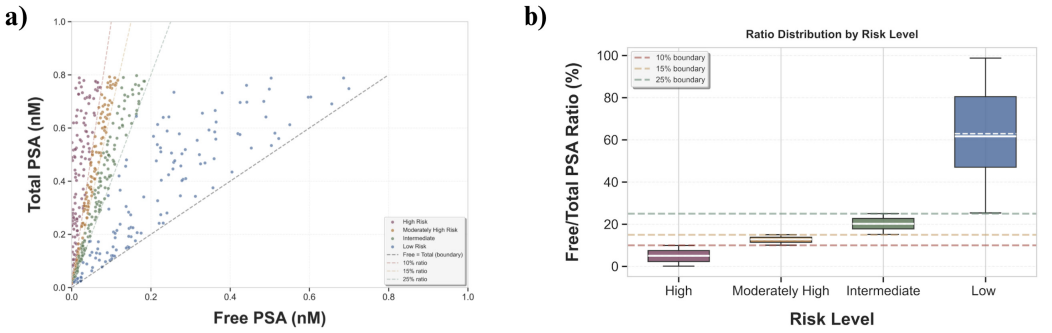


Fig. 6.6 Distribution of the synthetic dataset used for system-level evaluation. (a) Distribution of fPSA and tPSA concentrations. (b) Distribution of free-to-total PSA ratios across the four prostate cancer risk categories [13].

The distribution of the generated biomarker concentrations and free-to-total PSA ratios is illustrated in Fig. 6.6. The ratio values are distributed within clinically established threshold ranges, with partial overlap occurring near class boundaries. These distributions provide a statistical representation of the input space encountered by the proposed neuromorphic classification system.

The dataset was randomly split into training and testing sets, with 80% of the samples used for training and the remaining 20% reserved for testing. The training set was used to optimize the network parameters, while the test set was kept separate and used only to evaluate the final performance of the model.

Classification performance was evaluated using overall accuracy, per-class accuracy, and confusion matrix analysis. Overall accuracy quantifies the proportion of correctly classified samples across the entire test set, whereas per-class accuracy provides insight into the classification performance within individual risk categories. Confusion matrices were additionally employed to visualize misclassification patterns and to examine the relationship between predicted and ground-truth labels.

6.3.5 Training Performance and Classification Results

The convergence behavior of the proposed SNN was evaluated by monitoring the training and testing loss together with the corresponding classification accuracy over 210 training epochs. The resulting learning curves are presented in Fig. 6.7.

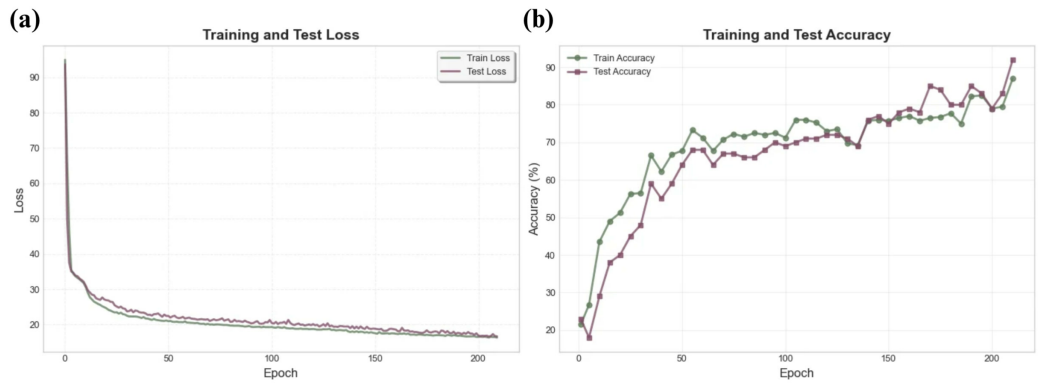


Fig. 6.7 Training and testing performance of the proposed SNN over 210 epochs. (a) Evolution of training and testing loss. (b) Evolution of training and testing accuracy [13].

As shown in Fig. 6.7(a), both the training and testing loss decrease rapidly during the initial training phase and continue to decline gradually throughout the remaining epochs. The close agreement between the two curves indicates stable optimization and suggests that the network does not exhibit significant overfitting during training. By the end of the training process, the loss values converge to a relatively stable level, demonstrating effective learning of the underlying relationships between pulse-encoded biomarker signals and prostate cancer risk categories.

The corresponding accuracy curves are shown in Fig. 6.7(b). At the beginning of training, both training and testing accuracy are close to chance level, as expected for a four-class classification problem. As learning progresses, accuracy increases steadily for both datasets. At convergence, the training accuracy reaches 87.0%, while the testing accuracy reaches 92.0%. The similarity of the two curves throughout the training process further confirms the robustness and generalization capability of the proposed SNN architecture.

The classification behavior of the trained network is further illustrated through the confusion matrices shown in Fig. 6.8. Most predictions are concentrated along the main diagonal, indicating that the majority of samples are assigned to their correct risk categories. The observed classification errors occur primarily between neighboring clinical risk levels, particularly between the intermediate-risk and moderately high-risk categories. This behavior is expected because samples located near decision boundaries often exhibit similar biomarker characteristics and therefore produce comparable pulse-generation patterns.

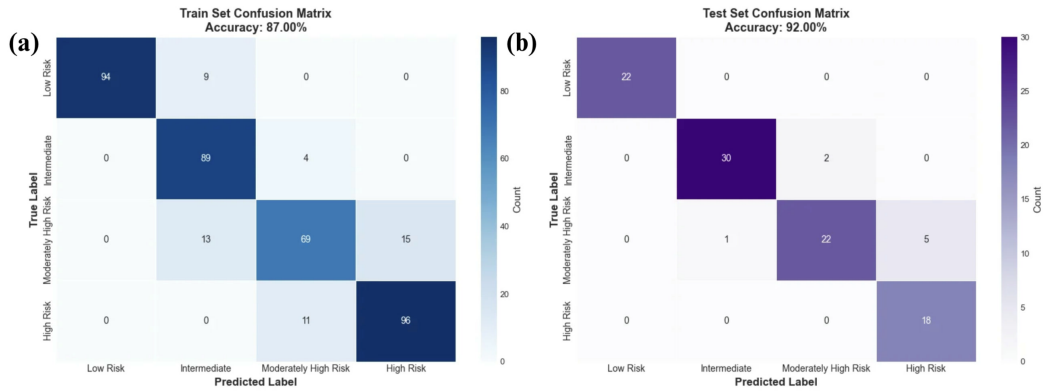


Fig. 6.8 Confusion matrices obtained using the proposed SNN for prostate cancer risk classification. (a) Training dataset. (b) Testing dataset [13].

Table 6.2 Classification performance of the proposed SNN on the testing dataset [13].

Class	**Precision**	**Recall**	**F1-score**	**Support**
Low Risk	1.000	1.000	1.000	22
Intermediate Risk	0.968	0.938	0.952	32
Moderately High Risk	0.917	0.786	0.846	28
High Risk	0.783	1.000	0.878	18
Accuracy	–	–	0.920	100
Macro Average	0.917	0.931	0.919	100
Weighted Average	0.927	0.920	0.920	100

For the testing dataset, all low-risk and high-risk samples are correctly identified. The majority of the remaining classification errors occur between the intermediate-risk and moderately high-risk groups, indicating that the proposed network successfully distinguishes the clinically important low- and high-risk extremes while maintaining strong performance for the intermediate categories.

A quantitative summary of the classification performance is provided in Table 6.2. The low-risk category achieves perfect precision, recall, and F1-score values of 1.000, indicating flawless classification on the testing dataset. The intermediate-risk category also demonstrates strong performance, achieving an F1-score of 0.952. The moderately high-risk category exhibits the lowest recall (0.786) and F1-score (0.846), which is consistent with the confusion patterns observed in Fig. 6.8. The

high-risk category achieves a recall of 1.000, indicating that all high-risk samples are successfully identified.

Overall, the proposed neuromorphic classification framework achieves a testing accuracy of 92.0%. The macro-averaged and weighted-averaged F1-scores are 0.919 and 0.920, respectively, demonstrating balanced classification performance across all four prostate cancer risk categories. These results confirm that pulse-encoded biomarker information generated directly by the silicon nanowire memristive biosensor can be effectively processed by the proposed SNN to perform accurate prostate cancer risk stratification without reconstructing continuous-valued biomarker concentrations.

6.3.6 Temporal and Layer-Wise Representation Analysis

To better understand how temporal information is processed throughout the proposed SNN, the membrane potentials and spike outputs of neurons across different layers were analyzed using a representative sample from the high-risk category. The resulting neuron responses are shown in Figs. 6.9, 6.10, and 6.11.

Figure 6.9(a) presents representative neurons from the first hidden layer. The neurons in this layer exhibit a wide variety of firing patterns in response to the input pulse streams. Some remain largely silent, others generate only a few spikes, and some respond much more actively throughout the simulation. Such differences indicate that the neurons are not all responding to the same aspects of the input. Instead, they appear to become sensitive to different temporal features contained within the pulse-encoded biomarker signals. The membrane potentials and spike responses for all 40 neurons in the first hidden layer are included in Figures D.1, D.2 in Appendix D for reference.

As the signals propagate to the second hidden layer, shown in Fig. 6.9(b), neuronal activity becomes more structured and concentrated. Compared with the first hidden layer, fewer neurons remain inactive, with only a single neuron exhibiting no spiking activity throughout the observation interval. The increased firing activity suggests that temporal features extracted by the first layer are progressively consolidated and transformed into more discriminative internal representations. The complete responses of all 20 neurons in the second hidden layer are provided in Figure D.3.

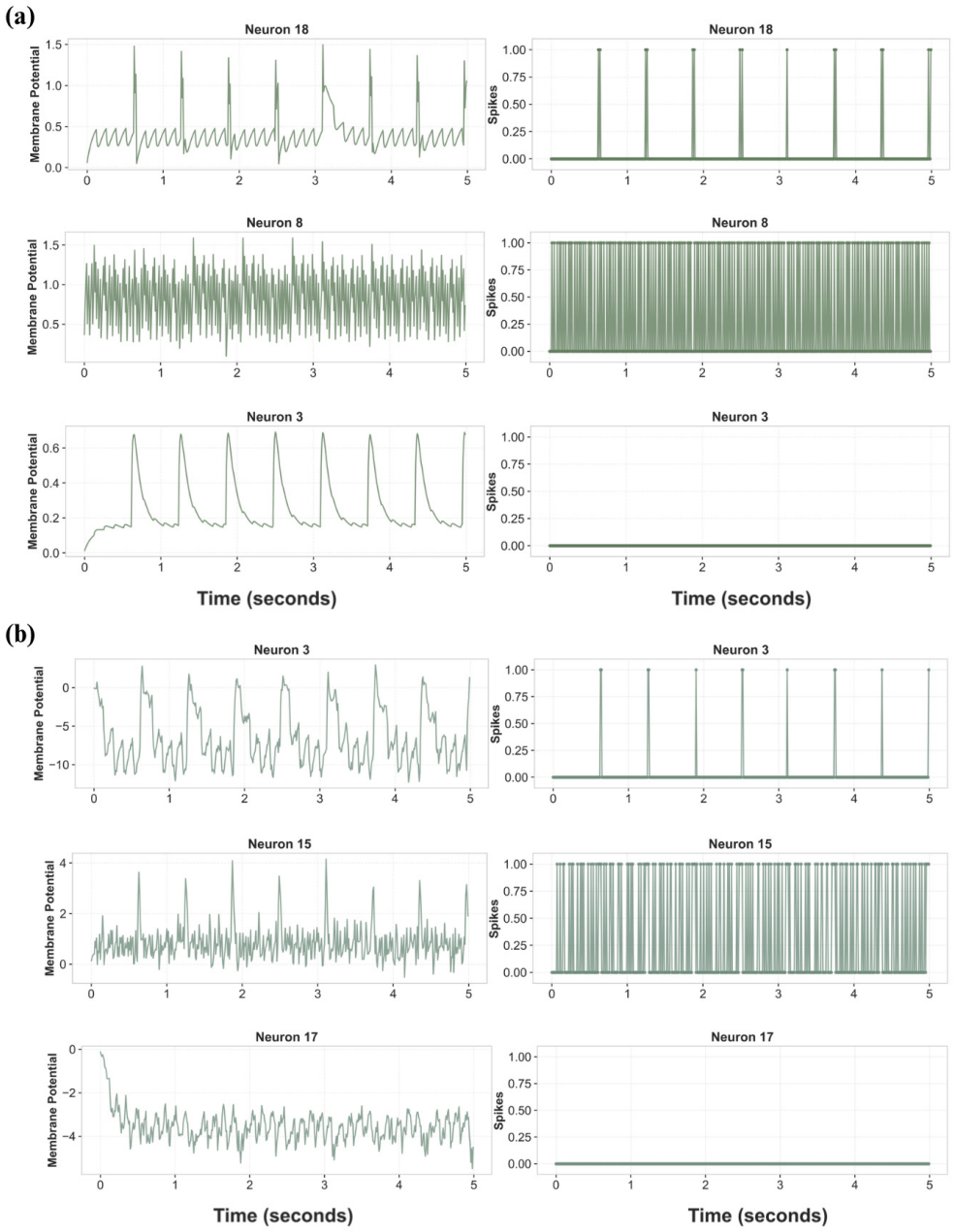


Fig. 6.9 Representative membrane potentials and spike outputs from neurons in the first and second hidden layers for a high-risk sample [13].

After processing through the first two hidden layers, the signals reach the third hidden layer consisting of eight neurons. This layer serves as a feature integration stage in which the most relevant temporal information extracted from previous layers

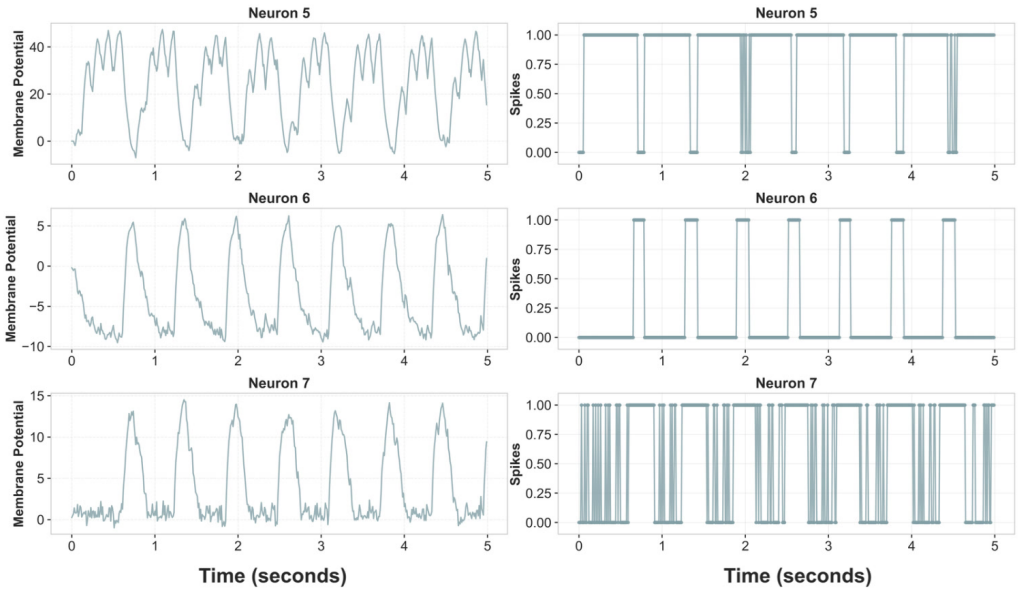


Fig. 6.10 Representative membrane potentials and spike outputs from neurons in the third hidden layer for a high-risk sample [13].

is combined before classification. Representative neuron responses are shown in Fig. 6.10.

Different neurons exhibit noticeably different firing behaviors. For instance, neuron 6 shows a relatively regular spiking pattern with consistent firing intervals, while neuron 5 remains active throughout much of the simulation and produces spikes at a higher rate. In contrast, neuron 7 responds more intermittently, generating short bursts of activity when the synaptic input becomes sufficiently strong. These variations suggest that the neurons are capturing different aspects of the temporal information contained in the input signals. Together, they provide complementary spike-based representations that are passed to the output layer for the final classification decision. The responses of all eight neurons in the third hidden layer are included in Figure D.4.

The final output layer aggregates the spiking activity generated by the preceding hidden layers to produce class-specific responses. Figure 6.11 shows the output-layer activity for the same high-risk sample. The neuron corresponding to the high-risk category exhibits the strongest firing activity, whereas the remaining output neurons remain comparatively less active. Since classification is performed using a rate-based

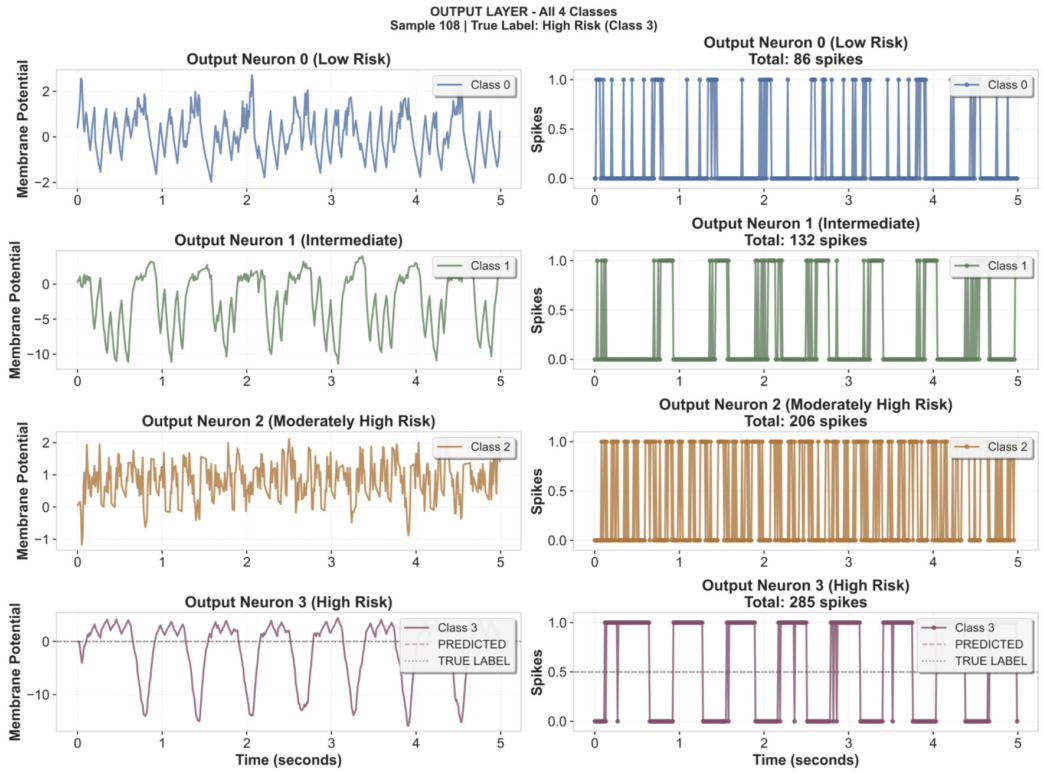


Fig. 6.11 Output-layer membrane potentials and spike responses for a representative high-risk sample [13].

decoding strategy, the predicted class is determined by the neuron with the highest firing rate. In this example, the predicted label correctly matches the ground-truth high-risk category.

Overall, the layer-wise analysis demonstrates how the proposed SNN progressively transforms pulse-encoded biomarker signals into increasingly compact and discriminative representations. Early layers perform initial temporal feature extraction, intermediate layers consolidate relevant information, and the final layers generate decision-oriented representations that enable accurate prostate cancer risk classification.

6.4 Chapter Summary

This chapter presented a neuromorphic front-end that directly exploits the pulse-generating behavior of silicon nanowire memristive biosensors for PSA-based prostate cancer risk stratification. Unlike conventional biosensing systems that rely on analog signal acquisition, digitization, and subsequent feature extraction, the proposed framework operates directly on pulse-domain information generated at the sensor level. As a result, biomarker information remains in the temporal domain throughout the sensing and classification process.

The experimental results showed that relevant risk information can be extracted from stochastic and partially overlapping pulse streams via hierarchical temporal processing in a SNN. The proposed system achieved an overall classification accuracy of 92%, with balanced performance across all four prostate cancer risk categories. The layer-wise representation analysis also showed how temporal information is progressively transformed into increasingly discriminative internal representations before reaching the final decision stage.

Overall, the results confirm the feasibility of combining pulse generating memristive biosensors with neuromorphic computing systems for biomedical risk assessment. Rather than reconstructing continuous-valued biomarker concentrations or explicitly calculating diagnostic ratios, the proposed approach performs classification directly on event-driven signals. The work presented in this chapter has been published in [13].

Chapter 7

Conclusions, Limitations, and Future Work

7.1 Conclusion

This thesis investigated the potential of silicon nanowire memristive devices for neuromorphic computing and neuromorphic biosensing applications through a comprehensive framework spanning device fabrication, electrical characterization, hardware-aware neural network modeling, and spike-based biosignal processing.

The first part of this work focused on the fabrication and characterization of silicon nanowire memristive devices. With oxide engineering and electrical characterization, the devices were shown to exhibit reproducible resistive switching behavior with two stable conductance states corresponding to the HRS and LRS. The results also suggest that oxide engineering can transform a single silicon nanowire device into multiple neuromorphic building blocks. Depending on the oxide characteristics and operating regime, the device may function as a synaptic memory element, a nonlinear activation component, a pulse-generating neurospiker, or a neuron-like computational unit. Experimental investigations further demonstrated the influence of fabrication conditions and oxide treatment on device performance, providing insight into the physical mechanisms governing the observed switching behavior.

Following device characterization, the suitability of the silicon nanowire memristor as a synaptic element for AIMC was investigated. Since the device has only two

stable conductance states, direct implementation of full precision neural network weights was not practical. To address this limitation, two hardware mapping strategies were investigated: a bit-sliced ANN implementation and a TNN implementation. A comprehensive comparison demonstrated that although the TNN sacrifices approximately 4% classification accuracy relative to the full-precision ANN, it decreases hardware complexity by approximately a factor of eight in terms of crossbar arrays, memristive devices, and peripheral circuitry. Consequently, the TNN architecture was identified as the most practical implementation strategy for the investigated silicon nanowire technology.

To evaluate realistic hardware performance, a complete hardware-aware simulation framework was established using CrossSim. Experimentally measured programming variability and conductance drift characteristics were extracted from silicon nanowire devices and incorporated into a custom CrossSim device model. This enabled direct investigation of how measured device-level non-idealities propagate through neural network inference.

The results illustrated that programming variability introduced only a small accuracy reduction from 93.78% to 92.56%. Conductance drift over the time interval caused only minor fluctuations in network performance. In contrast, ADC/DAC quantization was considered as the dominant source of accuracy degradation. After optimization of converter parameters, the final hardware-aware implementation achieved an inference accuracy of 81.14%. These findings demonstrate that peripheral circuit limitations currently represent a more significant challenge than the intrinsic variability of the silicon nanowire memristive device itself.

To recover this performance degradation, device-aware training was employed. By incorporating hardware non idealities directly into the training process, the neural network adapted to the underlying hardware constraints and recovered nearly all of the lost accuracy. Specifically, the hardware-aware TNN accuracy increased from 81.14% to 93.68%, approaching the original digital TNN accuracy of 93.78%. This represents an accuracy recovery of 12.54 percentage points and demonstrates the effectiveness of algorithm–hardware co-design in compensating for analog hardware non-idealities in practical neuromorphic computing systems.

Beyond synaptic computing, this thesis also explored the use of silicon nanowire memristive devices for neuromorphic biosensing applications. The volatile switching dynamics of silicon nanowire memristors were investigated as a potential mechanism

for generating pulse-based biosensor signals. Taking PSA sensing as a case study, a neuromorphic sensing framework was developed in which biomarker concentrations are represented as temporal spike streams. These spike sequences provide a direct interface to SNNs, enabling biomolecular information to be processed in the temporal domain.

A multi-layer SNN built from leaky integrate-and-fire neurons was designed and trained for prostate cancer risk classification. The network reached an accuracy of 92%, showing that relevant biomedical information can be extracted directly from the temporal structure of the spike trains. Since the signals are processed in their native pulse form, there is no need to reconstruct continuous-valued sensor outputs before inference. These results show that sensing and computation can be combined within the same event-driven neuromorphic system.

Overall, this thesis demonstrates that silicon nanowire memristive devices possess significant potential for future neuromorphic technologies. It brings together device-level studies, hardware-aware neural network design, and neuromorphic biosensing within a single research framework. The results show how silicon nanowire memristive devices can be connected to higher-level computational architectures, including AIMC, TNNs, and SNNs. By linking these different layers of the system, the work provides a foundation for future neuromorphic platforms in which sensing, memory, and computation operate together in an efficient and tightly integrated manner.

7.2 Limitations and Assumptions

Although this work combines experimental characterization with hardware-aware simulations, a number of assumptions and limitations remain and should be taken into account when evaluating the results.

First, TNN architecture investigated in this study was not exhaustively optimized. The selected network configuration was intended to provide a fair comparison with the conventional ANN while keeping practical implementation constraints. Although the TNN achieved approximately 94% classification accuracy, further optimization of network topology, hyperparameters, quantization strategies, and training procedures could potentially improve performance and provide a more representative evaluation of the maximum achievable accuracy.

The programming-error model developed for the silicon nanowire memristor was derived from experimentally measured device characteristics. However, the model primarily captures the variability observed in the final programmed conductance states. A more rigorous characterization would require analyzing the complete programming dynamics, including cycle-to-cycle variations, pulse-to-pulse fluctuations, and intermediate conductance transitions during SET and RESET operations. Such measurements would enable a more accurate representation of programming variability within the simulation framework.

In addition, the experimental characterization was conducted on a relatively limited number of devices and switching cycles. Although this was sufficient to demonstrate the feasibility of the proposed approach but more comprehensive assessment would require testing a larger number of devices produced across different fabrication batches and evaluated under a broader range of operating conditions. Such an expanded dataset would provide a deeper understanding of device-to-device variability and help assess the reproducibility of the fabrication process more reliably.

Read noise was not explicitly characterized in this work. The main focus of the study was on programming variability and system-level accuracy degradation. Although read noise is generally expected to have a smaller impact than programming errors for this device but future studies should incorporate read-noise models to provide a more complete device model.

For conductance drift, the analysis focused on the temporal evolution of the HRS and LRS over relatively short time intervals. Long-term retention measurements extending over hours, days, or longer periods were beyond the scope of this work. Consequently, the presented drift model should be considered an initial approximation rather than a complete long-term reliability assessment.

The ADC and DAC models employed in CrossSim are generic behavioral models intended to represent quantization and conversion effects. While suitable for evaluating system-level trends, they do not correspond to a specific hardware implementation. Accurate hardware prediction would require detailed modeling of the exact converter architectures used in a physical neuromorphic accelerator. Nevertheless, the results clearly demonstrate that ADC/DAC quantization constitutes one of the dominant sources of accuracy degradation in analog neural network inference.

Furthermore, the ANN and TNN architectures evaluated in this study employed networks of similar size to facilitate direct comparison. Alternative network topolo-

gies, including smaller and more hardware-efficient architectures, were not explored and may provide more favorable trade-offs between accuracy and implementation complexity.

The CrossSim study focused mainly on inference accuracy and how it is affected by device and hardware non idealities. Other factors that are important when evaluating practical hardware systems, such as energy consumption, latency, throughput, silicon area, and overall hardware efficiency, were beyond the scope of this work. As a result, the comparison between the ANN and TNN implementations should be viewed mainly in terms of classification performance rather than as a comprehensive assessment of hardware efficiency.

For the neuromorphic biosensing framework, the SNN was trained using 500 synthetic signal samples generated to mimic biosensor outputs. Real patient derived datasets were not available at the time of this study because the biosensing device has not yet been experimentally realized. Consequently, the synthetic dataset was intended to demonstrate the feasibility of integrating biosensing and neuromorphic processing rather than to establish clinical performance.

Finally, the SNN implemented in this work was based on the LIF neuron model, which was chosen because it is widely used as starting point for evaluattion. The results show that the overall approach is viable; however, the network design was not exhaustively optimized. Future improvements could come from exploring alternative neuron models, modifying the network architecture, or adopting more advanced training methods. Such investigations may lead to improved performance while also helping to reduce the hardware resources required for implementation.

7.3 Future Perspectives

Several promising research directions emerge from the findings of this work.

For the memristor-based neural network accelerator, an important next step is the experimental implementation of a hardware prototype using the silicon nanowire memristor technology investigated in this study. Such a prototype would enable direct comparison between simulated and measured inference accuracy and would provide valuable insights into practical implementation challenges. In addition, device-aware

trained weights should be deployed onto physical hardware to evaluate the extent to which simulation-based accuracy recovery can be reproduced experimentally.

Further work is also needed to improve the fabrication process and reduce variability between devices. Achieving more consistent switching behavior would make the overall system more predictable and reliable. As neuromorphic circuits scale to larger numbers of devices, better control of these variations is expected to play an important role in maintaining accurate and stable operation

The results obtained in this work indicate that ADC/DAC quantization error is one of the dominant factors that cause accuracy degradation. While device-aware training provides a partial solution, it does not remove the dependence on digital conversion stages. Consequently, an important research direction involves exploring analog alternatives to conventional activation and conversion circuits. The nonlinear property found in the silicon nanowire device open door for implementing activation functions directly at the device level. Future investigations should examine the reproducibility and controllability of these nonlinear responses and determine whether functions such as ReLU, sigmoid, or hyperbolic tangent activation can be approximated using the intrinsic device physics. Successful implementation of such analog activation functions could significantly reduce conversion overhead and improve overall system efficiency.

A key future objective for the neuromorphic biosensing platform is the realization of the proposed pulse-generating biosensor. Once fabricated, the device could be tested using real biomarker measurements and patient samples and it allows the SNN models to be trained and validated on experimental data. Moving from synthetic datasets to real world measurements is important step toward practical use of this device in biomedical applications.

From the neuromorphic computing perspective, future work should investigate ternary spiking neural networks (T-SNNs), where synaptic weights are constrained to the set $\{-1, 0, +1\}$. Since the silicon nanowire memristor naturally exhibits two stable conductance states, ternary weight representations offer an attractive pathway toward efficient hardware implementation. The resulting architecture could combine the benefits of event-driven computation with highly compact memristive memory structures.

Another promising direction is the development of silicon nanowire devices that simultaneously perform synaptic storage and neuronal computation. In particular,

future research may investigate whether the nonlinear dynamic behavior of the device can be utilized to emulate neuron models such as the LIF neuron. Integrating synaptic and neuronal functionality within the same physical device could significantly reduce circuit complexity and improve hardware efficiency.

Looking further ahead, the broader goal of this research is to develop a fully integrated neuromorphic biosensing platform in which sensing, memory, and computation are brought together within a single system. A platform like this could combine pulse generating biosensors, memristive synaptic elements, hardware-based spiking neurons, and neuromorphic processing circuitry into a unified architecture. By converting biological information directly into spike-based representations and processing it within the same framework and this approach has the potential to enable a new generation of intelligent biomedical sensing systems that are both energy efficient and well suited for real-time operation.

Data Availability Statement

The datasets, source codes, simulation frameworks, and supporting materials developed and used throughout this thesis are publicly available through the GitHub repositories listed below.

Silicon Nanowire Devices as Synaptic Elements for Neuromorphic Computing (Chapter 5)

All experimental datasets, including device characterization data, programming error measurements, drift error measurements, and associated analysis scripts used in Chapter 5 are openly available. Furthermore, all machine learning, hardware-aware simulation, and optimization codes developed in this work can be accessed through the following repository:

https://github.com/APMaii/Algorithm-Hardware-Neuromorphic-CoDesign

The repository contains:

- Baseline digital ANN training and evaluation codes.
- Baseline digital TNN implementations.
- TNN optimization frameworks.
- CrossSim-based ideal ANN and TNN simulations.
- Silicon nanowire device modeling, including programming error and drift error models.

- Hardware-aware CrossSim simulations incorporating experimentally characterized device behavior.
- Real-device-aware TNN optimization frameworks.
- ADC/DAC optimization methodologies.
- Final device-aware TNN simulation workflows.
- Device-aware training loops and path-model implementations.
- All supporting datasets and processing scripts required to reproduce the reported results.

In addition, this thesis introduces novel device models and TNN layers integrated directly into the CrossSim framework. The enhanced simulation package developed during this work, termed *CrossSim-TNN+*, is available at:

https://github.com/APMaii/cross-sim-TNN/tree/main

CrossSim-TNN+ extends the original CrossSim hardware-aware simulation framework by incorporating:

- TNN layers.
- Custom APM silicon nanowire device models.
- Model conversion and deployment utilities.
- Extended analog in-memory computing simulation capabilities.
- Hardware-aware training and inference support for neuromorphic systems.

Toward Silicon Nanowire Neuromorphic Biosensing Systems (Chapter 6)

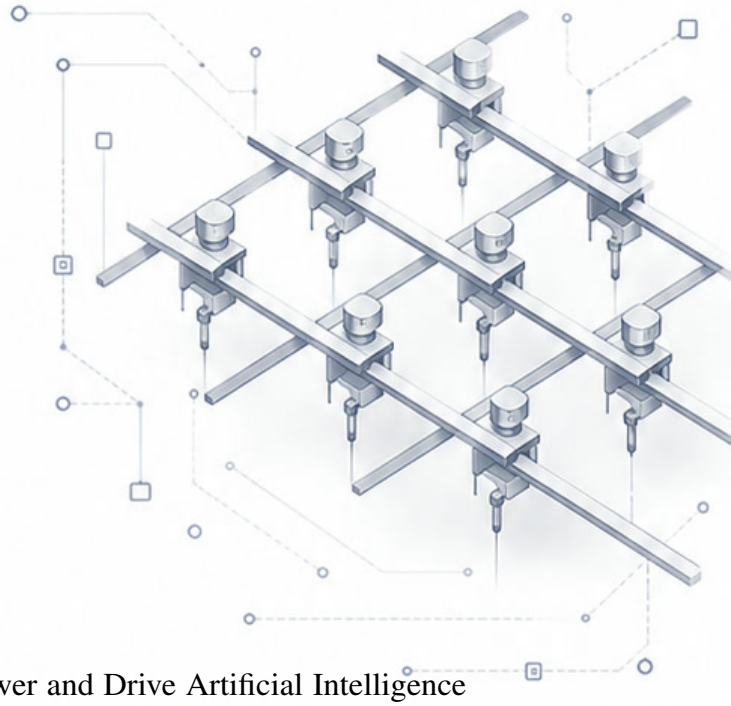
All datasets, signal synthesis tools, neuromorphic preprocessing algorithms, spiking encoders, and SNN implementations used in the chapter entitled *Toward Silicon Nanowire Neuromorphic Biosensing Systems* are publicly available at:

https://github.com/APMaii/neuromorphic-biosignal-prescreening

This repository contains:

- Experimental and synthetic biosignal datasets.
- Signal synthesis and augmentation tools.
- Spike encoding implementations.
- Spiking Neural Network models developed using SNNTorch.
- Training, validation, and evaluation scripts.
- Data preprocessing and analysis pipelines required for reproducing the reported results.

All repositories are maintained by the author and provide the materials necessary to reproduce the principal computational results presented in this thesis.



References

[1] How Much Longer Can Computing Power and Drive Artificial Intelligence Progress. Ai and compute. 2022.

[2] Kaixuan Sun, Jingsheng Chen, and Xiaobing Yan. The future of memristors: Materials engineering and neural networks. *Advanced Functional Materials*, 31(8):2006773, 2021.

[3] Conrad D James, James B Aimone, Nadine E Miner, Craig M Vineyard, Fredrick H Rothganger, Kristofor D Carlson, Samuel A Mulder, Timothy J Draelos, Aleksandra Faust, Matthew J Marinella, et al. A historical survey of algorithms and hardware architectures for neural-inspired and neuromorphic computing applications. *Biologically Inspired Cognitive Architectures*, 19:49–64, 2017.

[4] Dhireesha Kudithipudi, Catherine Schuman, Craig M Vineyard, Tej Pandit, Cory Merkel, Rajkumar Kubendran, James B Aimone, Garrick Orchard, Christian Mayr, Ryad Benosman, et al. Neuromorphic computing at scale. *Nature*, 637(8047):801–812, 2025.

[5] Xiaoqian Wei, Xinghou He, Pian Wu, Fangjie Gong, Danqi Wang, Shanlin Wang, Siyu Lu, Jingwen Zhang, Shan Xiang, Tianhan Kai, et al. Recent advances in the design of semiconductor hollow microspheres for enhanced photocatalyticv water splitting. *International Journal of Hydrogen Energy*, 46(55):27974–27996, 2021.

[6] H Kasani. Fabrication of carbon nanoparticle/polymer nanocomposite based thermometer: Communication. *Diyala Journal of Engineering Sciences*, pages 606–621, 2015.

[7] Shaurya Prakash and Junghoon Yeom. *Nanofluidics and microfluidics: systems and applications*. William Andrew, 2014.

[8] Junrui Chen, Kapil Bhardwaj, and Sandro Carrara. Role of the oxide in memristive quasi-1d silicon nanowires. *Nanoscale*, 17(14):8660–8671, 2025.

[9] Ben Feinberg, T. Patrick Xiao, Curtis J. Brinker, Christopher H. Bennett, Matthew J. Marinella, and Sapan Agarwal. Crosssim: accuracy simulation of analog in-memory computing.

[10] T Patrick Xiao, Ben Feinberg, Christopher H Bennett, Vineet Agrawal, Prashant Saxena, Venkatraman Prabhakar, Krishnaswamy Ramkumar, Harsha Medu, Vijay Raghavan, Ramesh Chettuvetty, et al. An accurate, error-tolerant, and energy-efficient neural network inference engine based on sonos analog memory. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 69(4):1480–1493, 2022.

[11] Vinay Joshi, Manuel Le Gallo, Simon Haefeli, Irem Boybat, Sasidharan Rajalekshmi Nandakumar, Christophe Piveteau, Martino Dazzi, Bipin Rajendran, Abu Sebastian, and Evangelos Eleftheriou. Accurate deep neural network inference using computational phase-change memory. *Nature communications*, 11(1):2473, 2020.

[12] Valerio Milo, Francesco Anzalone, Cristian Zambelli, Eduardo Pérez, Mamathamba K Mahadevaiah, Óscar G Ossorio, Piero Olivo, Christian Wenger, and Daniele Ielmini. Optimized programming algorithms for multilevel rram in hardware neural networks. In *2021 IEEE International Reliability Physics Symposium (IRPS)*, pages 1–6. IEEE, 2021.

[13] Junrui Chen, Ali Pilehvar Meibody, and Sandro Carrara. A neuromorphic front-end based on memristive biosensors for risk pre-screening by pulse-encoded signals. *IEEE Sensors Journal*, 2026.

[14] Mark H Kawachi, Robert R Bahnson, Michael Barry, J Erik Busby, Peter R Carroll, H Ballentine Carter, William J Catalona, Michael S Cookson, Jonathan I Epstein, Ruth B Etzioni, et al. Prostate cancer early detection. *Journal of the National Comprehensive Cancer Network*, 8(2):240–262, 2010.

[15] Andrew W Roddam, Michael J Duffy, Freddie C Hamdy, Anthony Milford Ward, Julietta Patnick, Christopher P Price, Janet Rimmer, Cathie Sturgeon, Peter White, Naomi E Allen, et al. Use of prostate-specific antigen (psa) isoforms for the detection of prostate cancer in men with a psa level of 2–10 ng/ml: systematic review and meta-analysis. *European urology*, 48(3):386–399, 2005.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[18] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[19] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 3645–3650, 2019.

[20] Radosvet Desislavov, Fernando Martínez-Plumed, and José Hernández-Orallo. Trends in ai inference energy consumption: Beyond the performance-vs-parameter laws of deep learning. *Sustainable Computing: Informatics and Systems*, 38:100857, 2023.

[21] Rudolf Eigenmann and David J Lilja. Von neumann computers. *Wiley Encyclopedia of Electrical and Electronics Engineering*, 23:387–400, 1998.

[22] II Arikpo, FU Ogban, and IE Eteng. Von neumann architecture and modern computers. *Global Journal of Mathematical Sciences*, 6(2):97–103, 2007.

[23] Giacomo Indiveri and Shih-Chii Liu. Memory and information processing in neuromorphic systems. *Proceedings of the IEEE*, 103(8):1379–1397, 2015.

[24] Doo Seok Jeong, Kyung Min Kim, Sungho Kim, Byung Joon Choi, and Cheol Seong Hwang. Memristors for energy-efficient new computing paradigms. *Advanced Electronic Materials*, 2(9):1600090, 2016.

[25] Naveen Verma, Hongyang Jia, Hossein Valavi, Yinqi Tang, Murat Ozatay, Lung-Yen Chen, Bonan Zhang, and Peter Deaville. In-memory computing: Advances and prospects. *IEEE solid-state circuits magazine*, 11(3):43–55, 2019.

[26] Daniele Ielmini and Giacomo Pedretti. Device and circuit architectures for in-memory computing. *Advanced Intelligent Systems*, 2(7):2000040, 2020.

[27] Catherine D Schuman, Shruti R Kulkarni, Maryam Parsa, J Parker Mitchell, Prasanna Date, and Bill Kay. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, 2(1):10–19, 2022.

[28] Steve Furber. Large-scale neuromorphic computing systems. *Journal of neural engineering*, 13(5):051001, 2016.

[29] Mattias Nilsson, Olov Schelén, Anders Lindgren, Ulf Bodin, Cristina Paniagua, Jerker Delsing, and Fredrik Sandin. Integration of neuromorphic ai in event-driven distributed digitized systems: Concepts and research directions. *Frontiers in Neuroscience*, 17:1074439, 2023.

[30] Daniele Ielmini and Stefano Ambrogio. Emerging neuromorphic devices. *Nanotechnology*, 31(9):092001, 2020.

[31] Navnidhi K Upadhyay, Hao Jiang, Zhongrui Wang, Shiva Asapu, Qiangfei Xia, and J Joshua Yang. Emerging memory devices for neuromorphic computing. *Advanced Materials Technologies*, 4(4):1800589, 2019.

[32] JJ Yang. Strukovdb and stewartdr 2013memristive devices for computingnat. *Nanotechnol*, 8:13–24.

[33] Sung Hyun Jo, Ting Chang, Idongesit Ebong, Bhavitavya B Bhadviya, Pinaki Mazumder, and Wei Lu. Nanoscale memristor device as synapse in neuromorphic systems. *Nano letters*, 10(4):1297–1301, 2010.

[34] Shimeng Yu. Neuro-inspired computing with emerging nonvolatile memorys. *Proceedings of the IEEE*, 106(2):260–285, 2018.

[35] Rainer Waser, Regina Dittmann, Georgi Staikov, and Kristof Szot. Redox-based resistive switching memories–nanoionic mechanisms, prospects, and challenges. *Advanced materials*, 21(25-26):2632–2663, 2009.

[36] Zhongrui Wang, Saumil Joshi, Sergey E Savel'ev, Hao Jiang, Rivu Midya, Peng Lin, Miao Hu, Ning Ge, John Paul Strachan, Zhiyong Li, et al. Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing. *Nature materials*, 16(1):101–108, 2017.

[37] Bryan L Jackson, Bipin Rajendran, Gregory S Corrado, Matthew Breitwisch, Geoffrey W Burr, Roger Cheek, Kailash Gopalakrishnan, Simone Raoux, Charles T Rettner, Alvaro Padilla, et al. Nanoscale electronic synapses using phase change devices. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 9(2):1–20, 2013.

[38] Duygu Kuzum, Rakesh GD Jeyasingh, Byoungil Lee, and H-S Philip Wong. Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing. *Nano letters*, 12(5):2179–2186, 2012.

[39] Abu Sebastian, Manuel Le Gallo, Geoffrey W Burr, Sangbum Kim, Matthew BrightSky, and Evangelos Eleftheriou. Tutorial: Brain-inspired computing using phase-change memory devices. *Journal of Applied Physics*, 124(11), 2018.

[40] TS Böscke, Johannes Müller, Dennis Braeuhaus, Uwe Schröder, and UJAPL Böttger. Ferroelectricity in hafnium oxide thin films. *Applied Physics Letters*, 99(10), 2011.

[41] Asif Islam Khan, Ali Keshavarzi, and Suman Datta. The future of ferroelectric field-effect transistor technology. *Nature Electronics*, 3(10):588–597, 2020.

[42] Alice Mizrahi, Tifenn Hirtzlin, Akio Fukushima, Hitoshi Kubota, Shinji Yuasa, Julie Grollier, and Damien Querlioz. Neural-like computing with populations of superparamagnetic basis functions. *Nature communications*, 9(1):1533, 2018.

[43] Adrien F Vincent, Jérôme Larroque, Nicolas Locatelli, Nesrine Ben Romdhane, Olivier Bichler, Christian Gamrat, Wei Sheng Zhao, Jacques-Olivier Klein, Sylvie Galdin-Retailleau, and Damien Querlioz. Spin-transfer torque magnetic memory as a stochastic memristive synapse for neuromorphic systems. *IEEE transactions on biomedical circuits and systems*, 9(2):166–174, 2015.

[44] Abhronil Sengupta, Yong Shim, and Kaushik Roy. Proposal for an all-spin artificial neural network: Emulating neural and synaptic functionalities through domain wall motion in ferromagnets. *IEEE transactions on biomedical circuits and systems*, 10(6):1152–1160, 2016.

[45] S Jiang, S Nie, Y He, R Liu, C Chen, and Q Wan. Emerging synaptic devices: from two-terminal memristors to multiterminal neuromorphic transistors. *Materials Today Nano*, 8:100059, 2019.

[46] Kyunghyun Kim, Chia-Ling Chen, Quyen Truong, Alex M Shen, Yong Chen, et al. A carbon nanotube synapse with dynamic logic and learning. *Adv. Mater*, 25(12):1693–1698, 2013.

[47] Davide Sacchetto, M Haykel Ben-Jamaa, Sandro Carrara, Giovanni De Micheli, and Yusuf Leblebici. Memristive devices fabricated with silicon nanowire schottky barrier transistors. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 9–12. IEEE, 2010.

[48] V Aglieri, A Zaffora, G Lullo, M Santamaria, F Di Franco, U Lo Cicero, M Mosca, and R Macaluso. Resistive switching in microscale anodic titanium dioxide-based memristors. *Superlattices and Microstructures*, 113:135–142, 2018.

[49] Nauman Malik Muhammad, Navaneethan Duraisamy, Khalid Rahman, Hyun Woo Dang, Jeongdae Jo, and Kyung Hyun Choi. Fabrication of printed memory device having zinc-oxide active nano-layer and investigation of resistive switching. *Current Applied Physics*, 13(1):90–96, 2013.

[50] Xiao-Ying Wei, Ming Hu, Kai-Liang Zhang, Fang Wang, Jin-Shi Zhao, and Yin-Ping Miao. Analysis of the resistive switching behaviors of vanadium oxide thin film. *Chinese Physics B*, 22(3):037201, 2013.

[51] D Kumar, R Aluguri, U Chand, and Tseung-Yuen Tseng. Metal oxide resistive switching memory: Materials, properties and switching mechanisms. *Ceramics International*, 43:S547–S556, 2017.

[52] Fei Zhuge, Kang Li, Bing Fu, Hongliang Zhang, Jun Li, Hao Chen, Lingyan Liang, Junhua Gao, Hongtao Cao, Zhimin Liu, et al. Mechanism for resistive switching in chalcogenide-based electrochemical metallization memory cells. *AIP Advances*, 5(5), 2015.

[53] Hou-Yen Tsao and Yow-Jon Lin. Resistive switching behaviors of au/pentacene/si-nanowire arrays/heavily doped n-type si devices for memory applications. *Applied Physics Letters*, 104(5), 2014.

[54] Razvan Pascu and Cosmin Romanitan. Phase transition of nickel silicide compounds and their electrical properties. *Journal of Materials Science: Materials in Electronics*, 32(12):16811–16823, 2021.

[55] Karl Skucha, Zhiyong Fan, Kanghoon Jeon, Ali Javey, and Bernhard Boser. Palladium/silicon nanowire schottky barrier-based hydrogen sensors. *Sensors and Actuators B: Chemical*, 145(1):232–238, 2010.

[56] Ji Hyun Hur, Myoung-Jae Lee, Chang Bum Lee, Young-Bae Kim, and Chang-Jung Kim. Modeling for bipolar resistive memory switching in transition-metal oxides. *Physical Review B—Condensed Matter and Materials Physics*, 82(15):155321, 2010.

[57] Kapil Bhardwaj, Ata Golparvar, Junrui Chen, Gian Luca Barbruni, and Sandro Carrara. New insights into the i/v hysteretic characteristics of memristive biosensors. *IEEE Sensors Letters*, 7(9):1–4, 2023.

[58] Jinming Zou, Yi Han, and Sung-Sau So. Overview of artificial neural networks. *Artificial neural networks: methods and applications*, pages 14–22, 2009.

[59] Aydin Buluç, Samuel Williams, Leonid Oliker, and James Demmel. Reduced-bandwidth multithreaded algorithms for sparse matrix-vector multiplication. In *2011 IEEE International Parallel & Distributed Processing Symposium*, pages 721–733. IEEE, 2011.

[60] Xingqi Zou, Sheng Xu, Xiaoming Chen, Liang Yan, and Yinhe Han. Breaking the von neumann bottleneck: architecture-level processing-in-memory technology. *Science China Information Sciences*, 64(6):160404, 2021.

[61] Abhairaj Singh, Manuel Le Gallo, Athanasios Vasilopoulos, Jose Luquin, Pritish Narayanan, Geoffrey W Burr, and Abu Sebastian. The design of analogue in-memory computing tiles. *Nature Electronics*, pages 1–14, 2025.

[62] Piergiulio Mannocci, Giacomo Larelli, Marco Bonomi, and Daniele Ielmini. Achieving high precision in analog in-memory computing systems. *npj Unconventional Computing*, 3(1):1, 2026.

[63] Peng Yao, Huaqiang Wu, Bin Gao, Jianshi Tang, Qingtian Zhang, Wenqiang Zhang, J Joshua Yang, and He Qian. Fully hardware-implemented memristor convolutional neural network. *Nature*, 577(7792):641–646, 2020.

[64] Xia Sheng, Catherine E Graves, Suhas Kumar, Xuema Li, Brent Buchanan, Le Zheng, Sity Lam, Can Li, and John Paul Strachan. Low-conductance and multilevel cmos-integrated nanoscale oxide memristors. *Advanced electronic materials*, 5(9):1800876, 2019.

[65] Xiaoxuan Yang, Brady Taylor, Ailong Wu, Yiran Chen, and Leon O Chua. Research progress on memristor: From synapses to computing systems. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 69(5):1845–1857, 2022.

[66] Wei Wang, Yang Li, and Ming Wang. Difficulties and approaches in enabling learning-in-memory using crossbar arrays of memristors. *Neuromorphic Computing and Engineering*, 4(3):032002, 2024.

[67] Fengfu Li, Bin Liu, Xiaoxing Wang, Bo Zhang, and Junchi Yan. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

[68] Taylor Simons and Dah-Jye Lee. A review of binarized neural networks. *Electronics*, 8(6):661, 2019.

[69] Van Rossum. Python tutorial, technical report cs-r9526. *(No Title)*, 1995.

[70] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[71] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[72] Tianda Fu, Xiaomeng Liu, Hongyan Gao, Joy E Ward, Xiaorong Liu, Bing Yin, Zhongrui Wang, Ye Zhuo, David JF Walker, J Joshua Yang, et al. Bioinspired bio-voltage memristors. *Nature communications*, 11(1):1861, 2020.

[73] Rui Yuan, Pek Jun Tiw, Lei Cai, Zhiyu Yang, Chang Liu, Teng Zhang, Chen Ge, Ru Huang, and Yuchao Yang. A neuromorphic physiological signal processing system based on vo2 memristor for next-generation human-machine interface. *Nature Communications*, 14(1):3695, 2023.

[74] Junrui Chen and Sandro Carrara. Capacitive effect in memristive biosensors. In *2024 IEEE BioSensors Conference (BioSensors)*, pages 1–4. IEEE, 2024.

[75] Ioulia Tzouvadaki, N Madaboosi, Irene Taurino, V Chu, JP Conde, Giovanni De Micheli, and Sandro Carrara. Study on the bio-functionalization of memristive nanowires for optimum memristive biosensors. *Journal of Materials Chemistry B*, 4(12):2153–2162, 2016.

[76] William J Catalona, Alan W Partin, Kevin M Slawin, Michael K Brawer, Robert C Flanigan, Anup Patel, Jerome P Richie, Jean B DeKernion, Patrick C Walsh, Peter T Scardino, et al. Use of the percentage of free prostate-specific antigen to enhance differentiation of prostate cancer from benign prostatic disease: a prospective multicenter clinical trial. *Jama*, 279(19):1542–1547, 1998.

[77] Marylyn D Ritchie, Emily R Holzinger, Ruowang Li, Sarah A Pendergrass, and Dokyoon Kim. Methods of integrating data to uncover genotype–phenotype interactions. *Nature Reviews Genetics*, 16(2):85–97, 2015.

[78] Margaret Sullivan Pepe, Ruth Etzioni, Ziding Feng, John D Potter, Mary Lou Thompson, Mark Thornquist, Marcy Winget, and Yutaka Yasui. Phases of biomarker development for early detection of cancer. *Journal of the National Cancer Institute*, 93(14):1054–1061, 2001.

[79] Ian M Thompson, Donna K Pauler, Phyllis J Goodman, Catherine M Tangen, M Scott Lucia, Howard L Parnes, Lori M Minasian, Leslie G Ford, Scott M Lippman, E David Crawford, et al. Prevalence of prostate cancer among men with a prostate-specific antigen level 4.0 ng per milliliter. *New England Journal of Medicine*, 350(22):2239–2246, 2004.

[80] Ioulia Tzouvadaki, Francesca Puppo, Marie-Agnès Doucey, Giovanni De Micheli, and Sandro Carrara. Computational study on the electrical behavior of silicon nanowire memristive biosensors. *IEEE Sensors Journal*, 15(11):6208–6217, 2015.

[81] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

[82] Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[83] Samanwoy Ghosh-Dastidar and Hojjat Adeli. Spiking neural networks. *International journal of neural systems*, 19(04):295–308, 2009.

[84] Kashu Yamazaki, Viet-Khoa Vo-Ho, Darshan Bulsara, and Ngan Le. Spiking neural networks and their applications: A review. *Brain sciences*, 12(7):863, 2022.

[85] Sanaullah, Shamini Koravuna, Ulrich Rückert, and Thorsten Jungeblut. Exploring spiking neural networks: a comprehensive analysis of mathematical models and applications. *Frontiers in Computational Neuroscience*, 17:1215824, 2023.

[86] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[87] Natalia Caporale and Yang Dan. Spike timing–dependent plasticity: a hebbian learning rule. *Annu. Rev. Neurosci.*, 31(1):25–46, 2008.

[88] Yuchen Wang, Hanwen Liu, Malu Zhang, Xiaoling Luo, and Hong Qu. A universal ann-to-snn framework for achieving high accuracy and low latency deep spiking neural networks. *Neural networks*, 174:106244, 2024.

[89] Mark Nelson and John Rinzel. The hodgkin—huxley model. In *The book of GENESIS: exploring realistic neural models with the GEneral NEural SImulation System*, pages 29–49. Springer, 1998.

[90] Jianfeng Feng. Is the integrate-and-fire model good enough?—a review. *Neural networks*, 14(6-7):955–975, 2001.

[91] Kai Sun, Peibo Duan, Levin Kuhlmann, Beilun Wang, and Bin Zhang. Ilif: Temporal inhibitory leaky integrate-and-fire neuron for overactivation in spiking neural networks. *arXiv preprint arXiv:2505.10371*, 2025.

[92] Jason K Eshraghian, Max Ward, Emre O Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 111(9):1016–1054, 2023.

Appendix A

Detailed Training-Loss Evolution of Surrogate Gradient Configurations

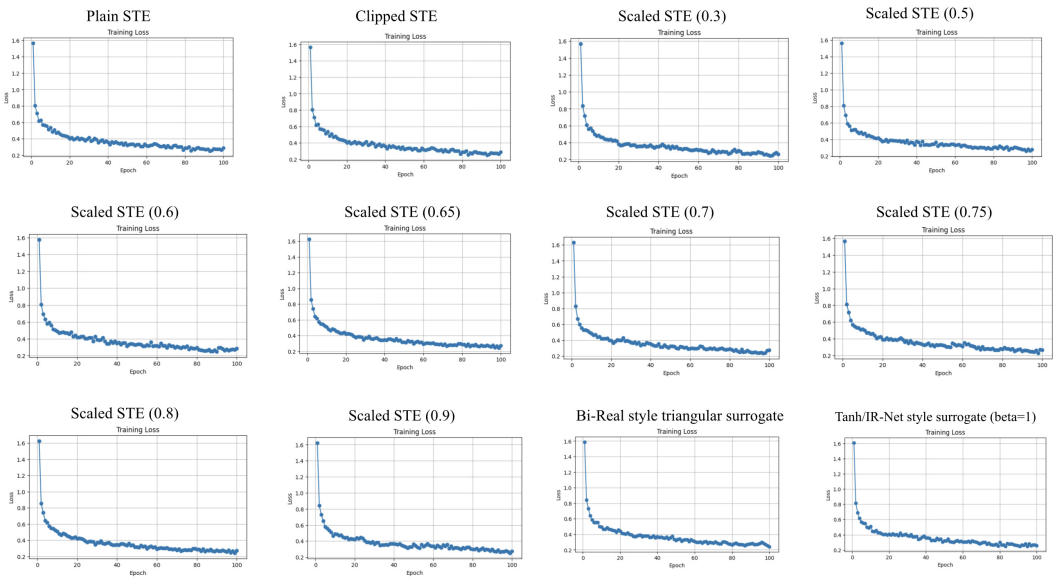


Fig. A.1 Training-loss evolution of the initial surrogate-gradient methods evaluated for TNN training. The figure compares Plain STE, Clipped STE, multiple Scaled STE configurations ($s = 0.3$–0.9), the Bi-Real style triangular surrogate, and the Tanh/IR-Net style surrogate ($\beta = 1$). All methods exhibit stable convergence, while differences in the final loss values and convergence rates provide insight into the influence of surrogate-gradient design on optimization performance.

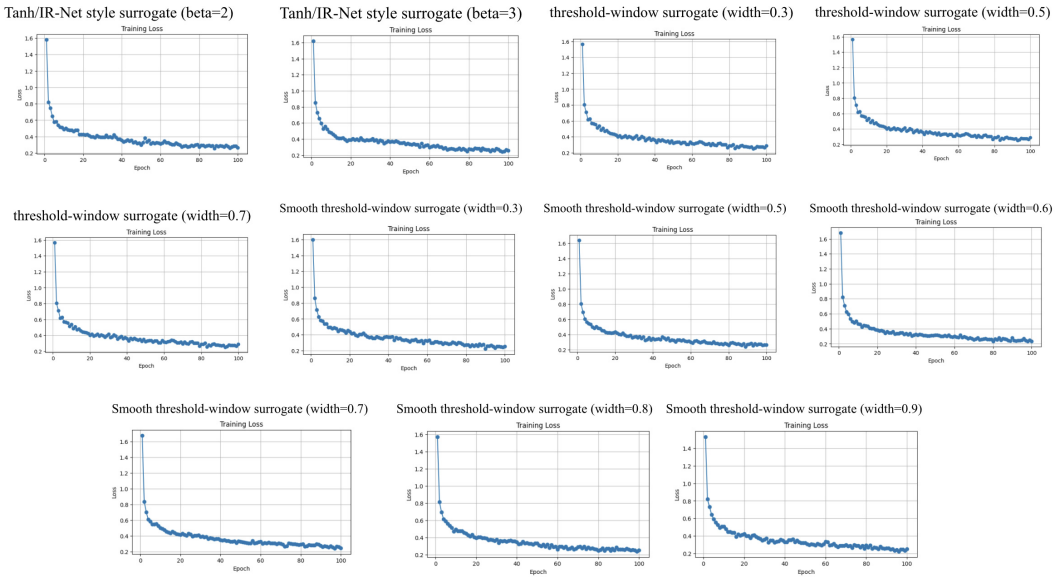


Fig. A.2 Training-loss curves for additional surrogate-gradient configurations evaluated in this work, including Tanh/IR-Net style surrogates ($\beta = 2$ and $\beta = 3$), threshold-window surrogates with different window widths, and smooth threshold-window surrogates with widths ranging from 0.3 to 0.9.

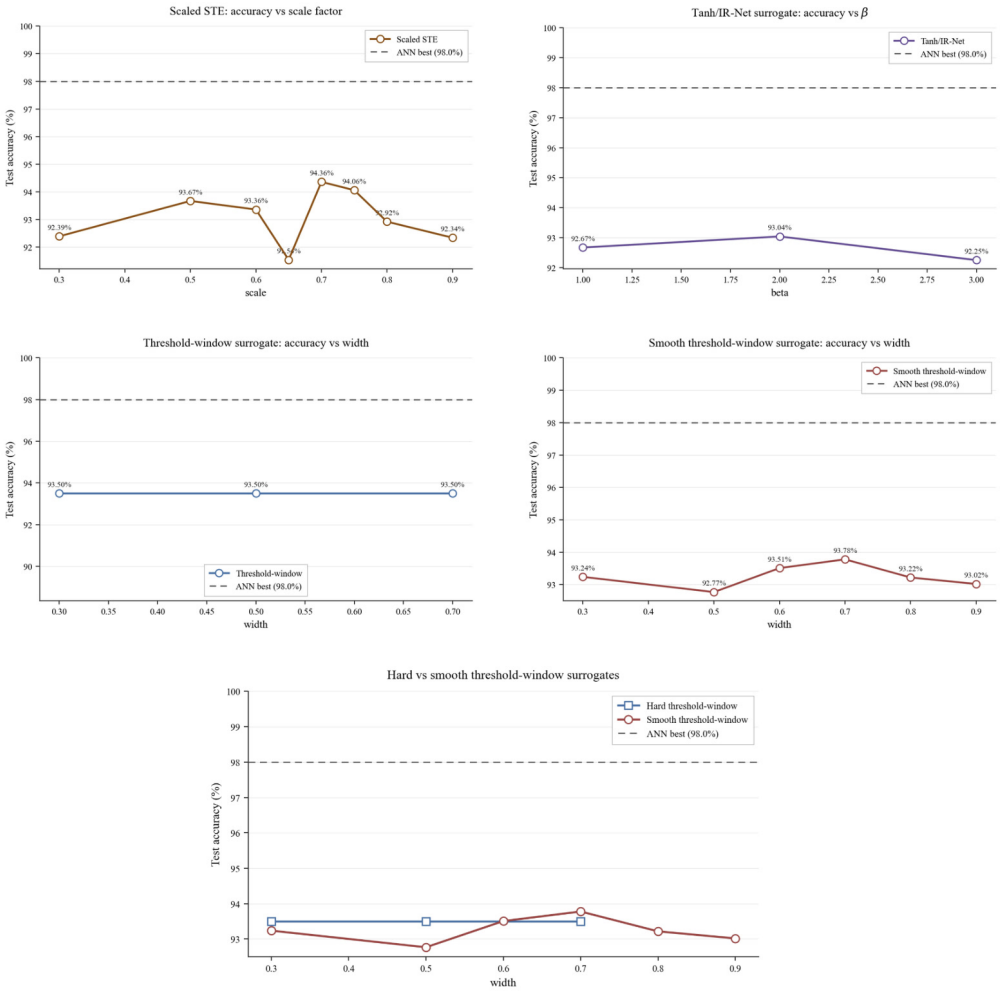


Fig. A.3 Parameter sensitivity analysis of the evaluated surrogate-gradient methods. The figure shows the influence of the scaling factor in Scaled STE, the parameter β in the Tanh/IR-Net surrogate, the window width in the threshold-window surrogate, and the window width in the smooth threshold-window surrogate on the final test accuracy. A direct comparison between hard and smooth threshold-window surrogates is also included. The dashed line indicates the best ANN baseline accuracy of 98.0%.

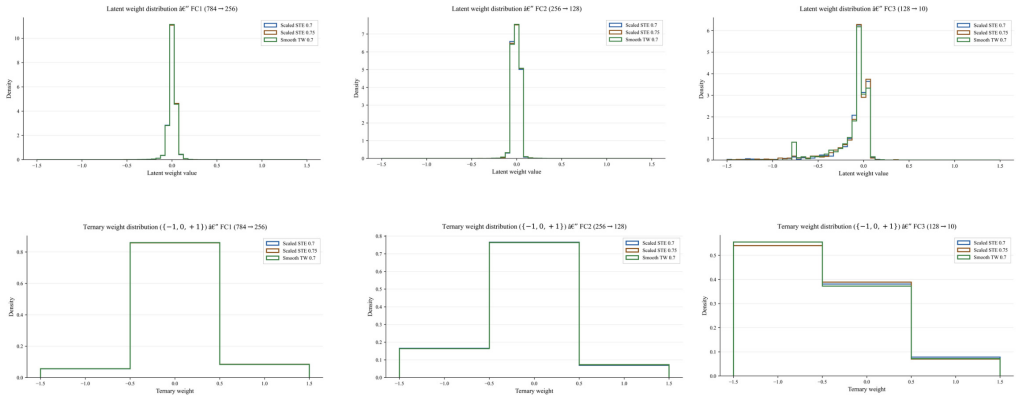


Fig. A.4 Latent full-precision weight distributions (top row) and corresponding ternary weight distributions (bottom row) for the three selected surrogate-gradient methods: Scaled STE ($s = 0.7$), Scaled STE ($s = 0.75$), and Smooth Threshold-Window ($r = 0.7$). Results are shown for FC1, FC2, and FC3. The latent weights remain continuous during optimization, whereas the forward computation uses ternary weights obtained through threshold-based quantization. The distributions demonstrate that all three methods produce highly similar weight representations, with only minor differences observed in the final classification layer.

Appendix B

Latent Weight Distributions and Ternarization Threshold Effects

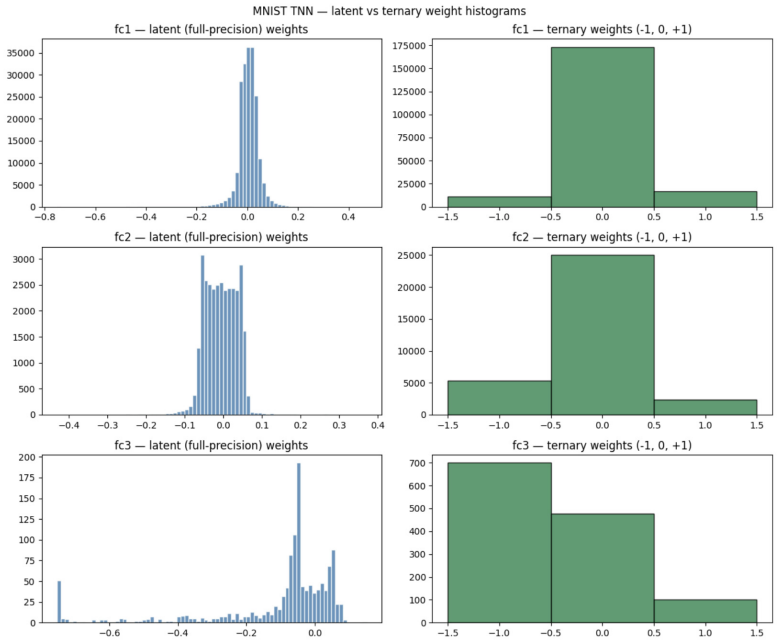


Fig. B.1 Latent full-precision weight distributions (left column) and corresponding ternary weight distributions (right column) for the three fully connected layers of the MNIST TNN. The latent weights remain continuous during optimization and are subsequently mapped to ternary values $\{-1, 0, +1\}$ through threshold-based quantization. The figure illustrates how the underlying latent distributions influence the final ternary weight allocation in each layer.

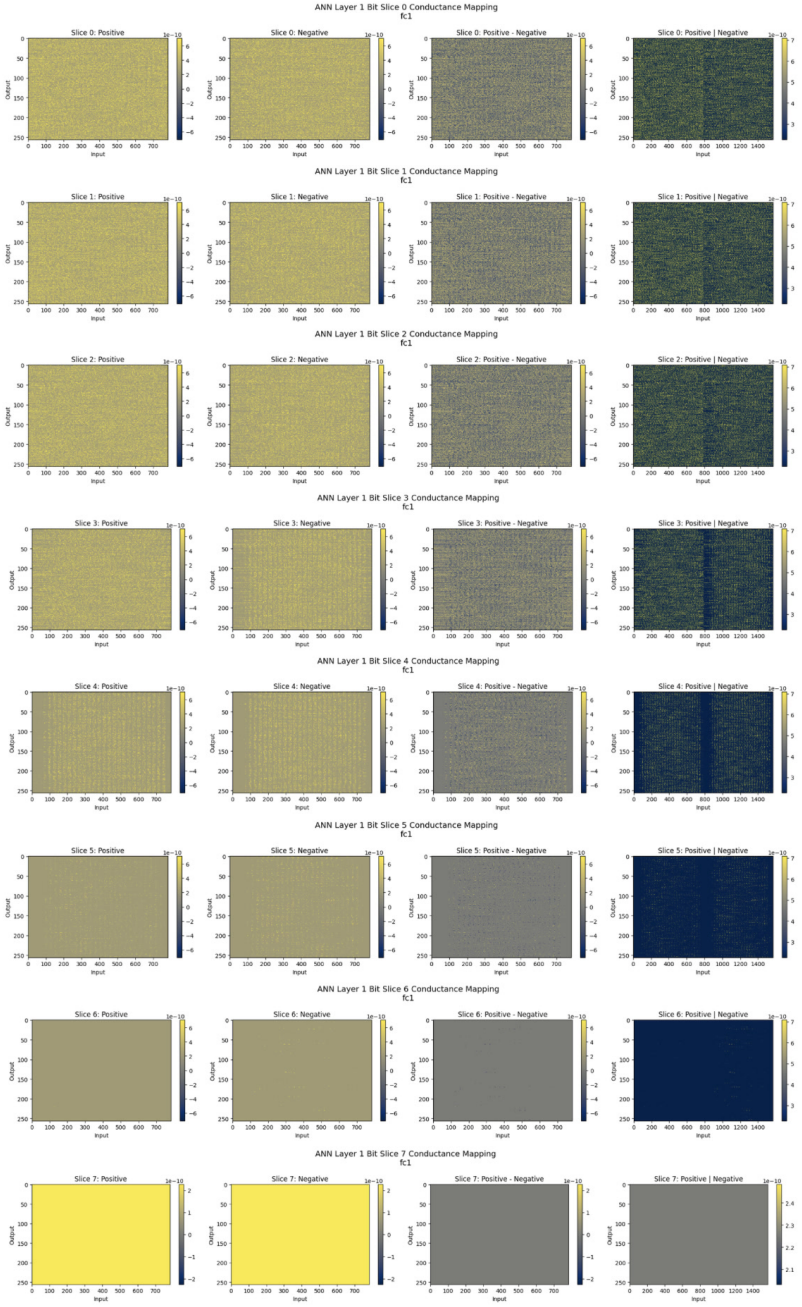


Fig. B.2 Conductance mapping of the first ANN layer (FC1) using the bit-sliced crossbar implementation. For each of the eight weight-bit slices, the positive conductance array, negative conductance array, differential representation (positive minus negative), and the combined crossbar mapping are shown. The figure illustrates the hardware overhead associated with implementing signed 8-bit weights on a binary memristive crossbar, where each bit slice requires separate positive and negative arrays, resulting in sixteen physical crossbar arrays for a single neural-network layer.

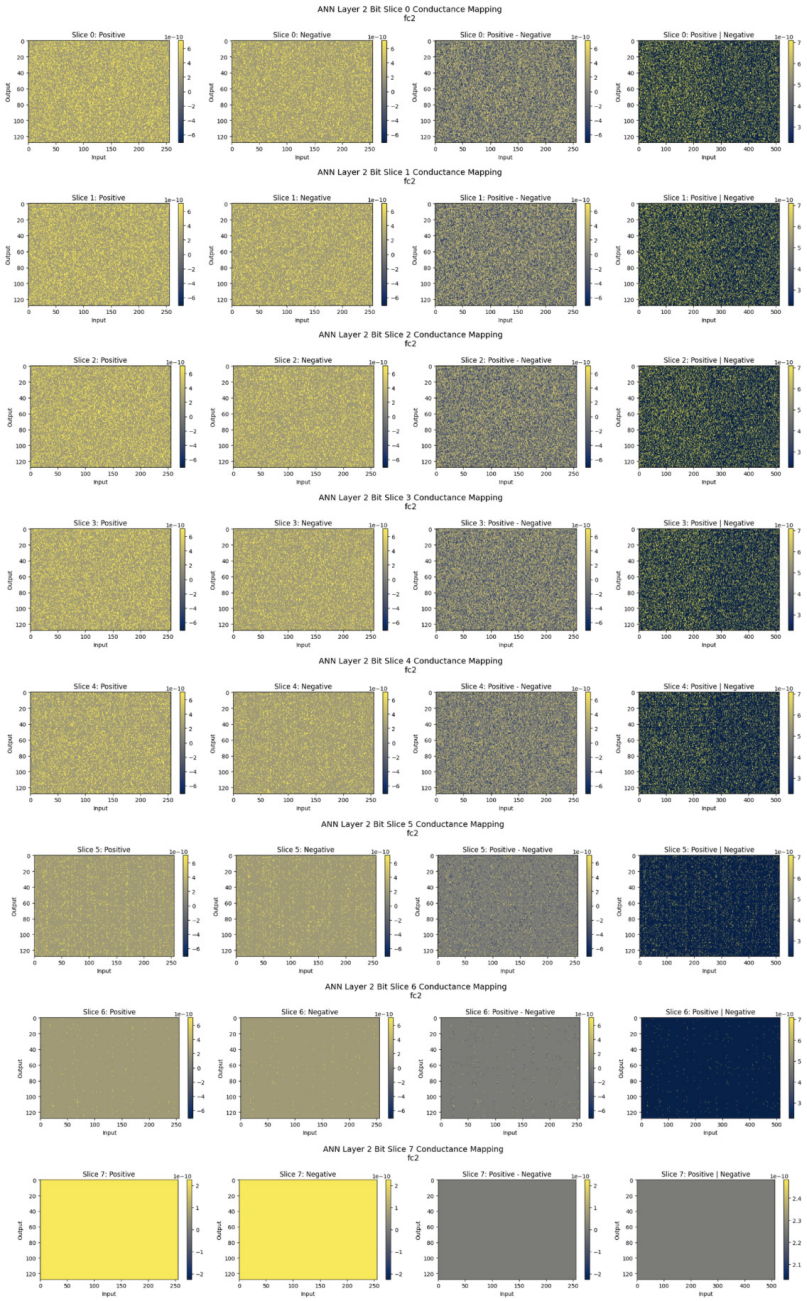


Fig. B.3 Bit-sliced conductance mapping of the second ANN layer (FC2). Each row corresponds to one weight-bit slice and shows the positive array, negative array, differential conductance map, and combined crossbar representation used to implement signed weights on binary memristive hardware. The figure illustrates how the 8-bit signed weights of FC2 are decomposed into individual bit slices and mapped onto separate positive and negative crossbar arrays.

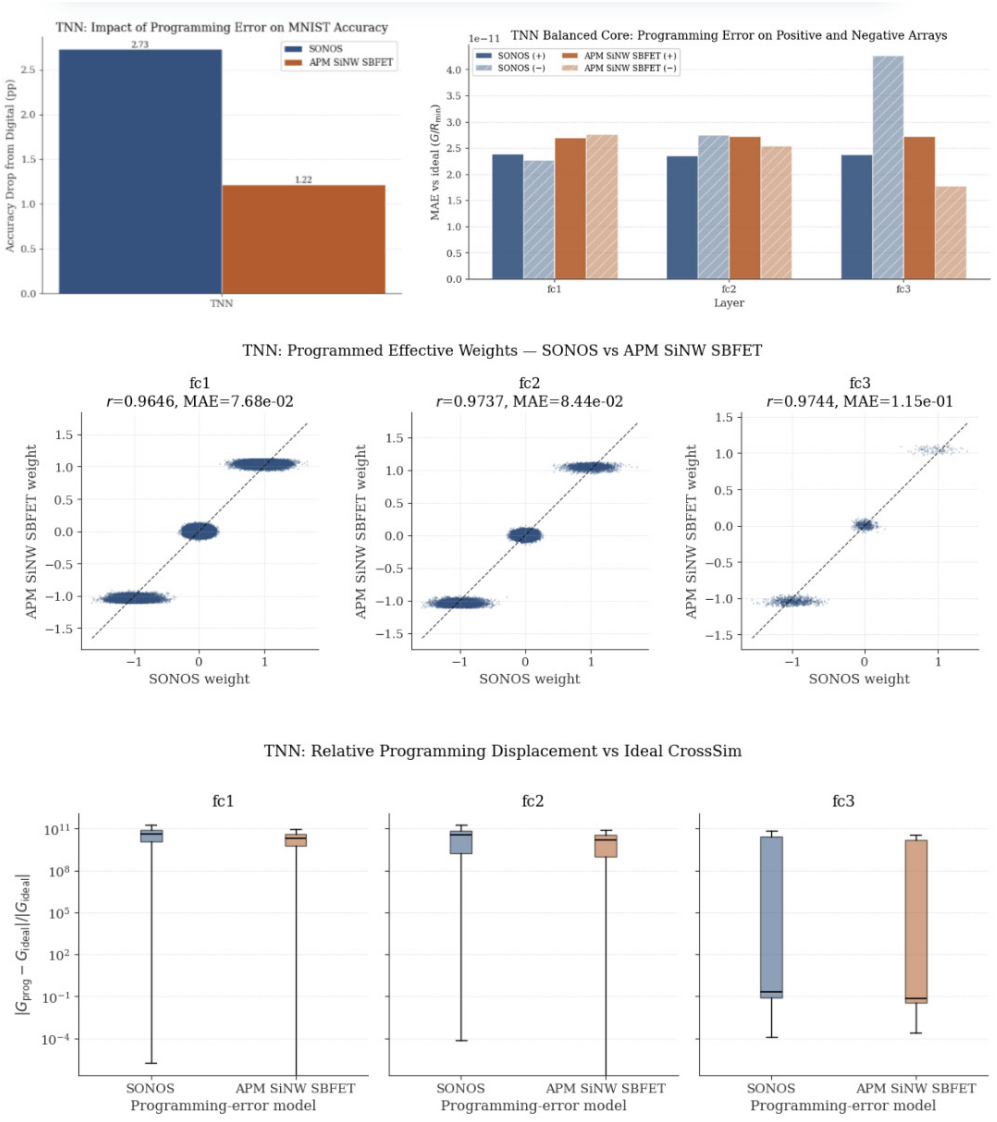


Fig. B.4 Validation of the proposed APM SiNW SBFET programming-error model against the reference SONOS model. The top row compares the resulting MNIST accuracy degradation and the mean absolute programming error of the positive and negative arrays across all network layers. The middle row shows the correlation between programmed effective weights obtained using the SONOS and APM SiNW SBFET models for FC1, FC2, and FC3. The bottom row presents the distributions of relative programming displacement from the ideal conductance values. The results demonstrate that the custom APM SiNW SBFET implementation reproduces the expected programming-error behavior while remaining fully compatible with the CrossSim device framework.

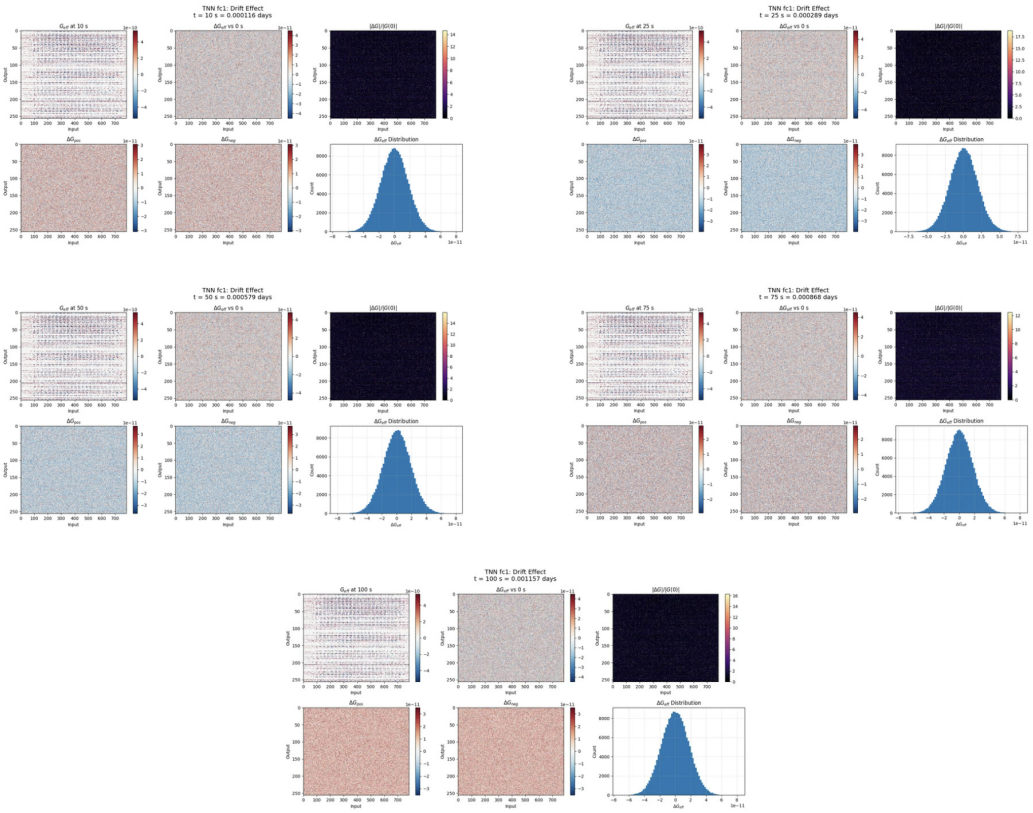


Fig. B.5 Drift-induced conductance variations in the first network layer (FC1) at retention times of 10 s, 25 s, 50 s, 75 s, and 100 s. For each time point, the effective conductance matrix (G_{eff}), conductance change relative to the initial state (ΔG), positive-core conductance variation (ΔG_{pos}), negative-core conductance variation (ΔG_{neg}), and the corresponding distribution of conductance changes are shown. The results illustrate the fluctuations in conductance values caused by the drift model over time.

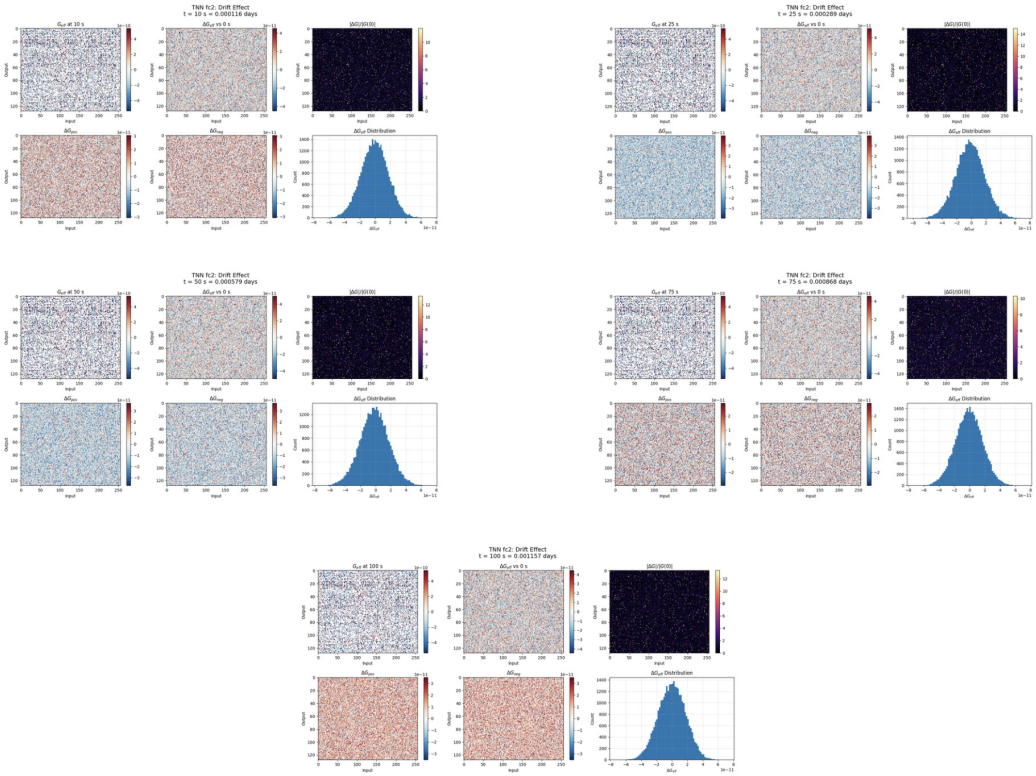


Fig. B.6 Conductance drift visualization for the second network layer (FC2) at different retention times. The figure shows the effective conductance matrix (G_{eff}), conductance changes relative to the initial state (ΔG), positive-core conductance variations (ΔG_{pos}), negative-core conductance variations (ΔG_{neg}), and the corresponding distributions of conductance changes. The results illustrate the conductance fluctuations introduced by the drift model over time.

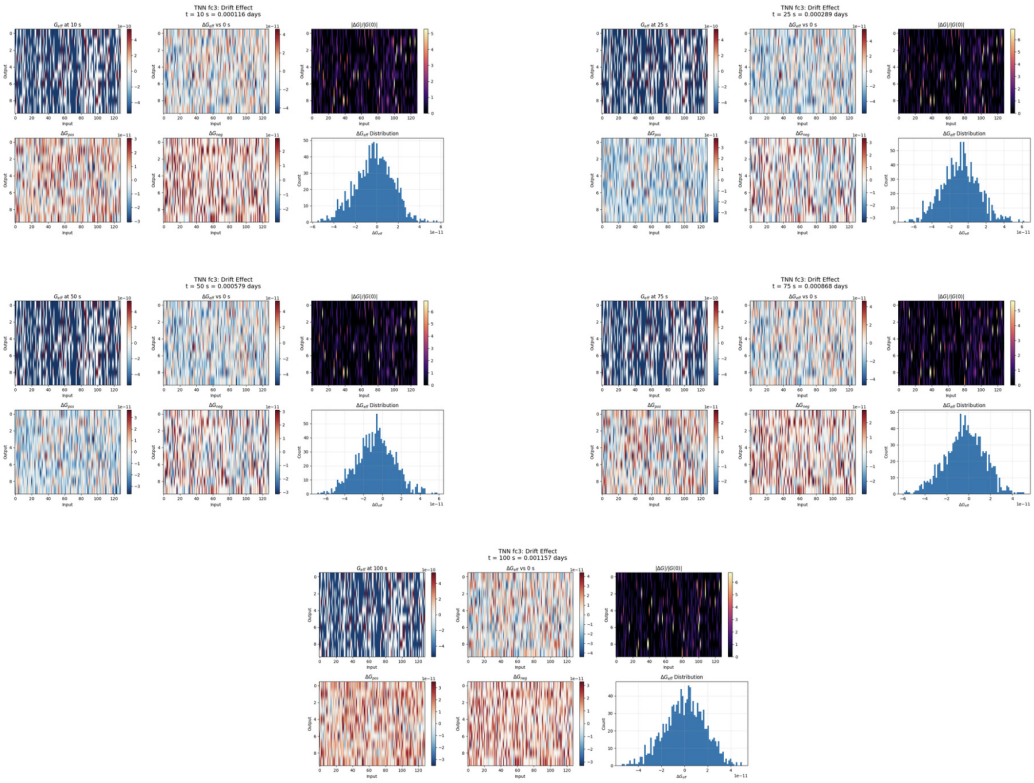


Fig. B.7 Drift-induced conductance variations in the third network layer (FC3) at retention times of 10 s, 25 s, 50 s, 75 s, and 100 s. The figure shows the effective conductance matrix (G_{eff}), conductance changes relative to the initial state (ΔG), positive-core conductance variations (ΔG_{pos}), negative-core conductance variations (ΔG_{neg}), and the corresponding distributions of conductance changes. The results illustrate the conductance fluctuations introduced by the drift model over time.

Appendix C

Alternative ADC/DAC Models

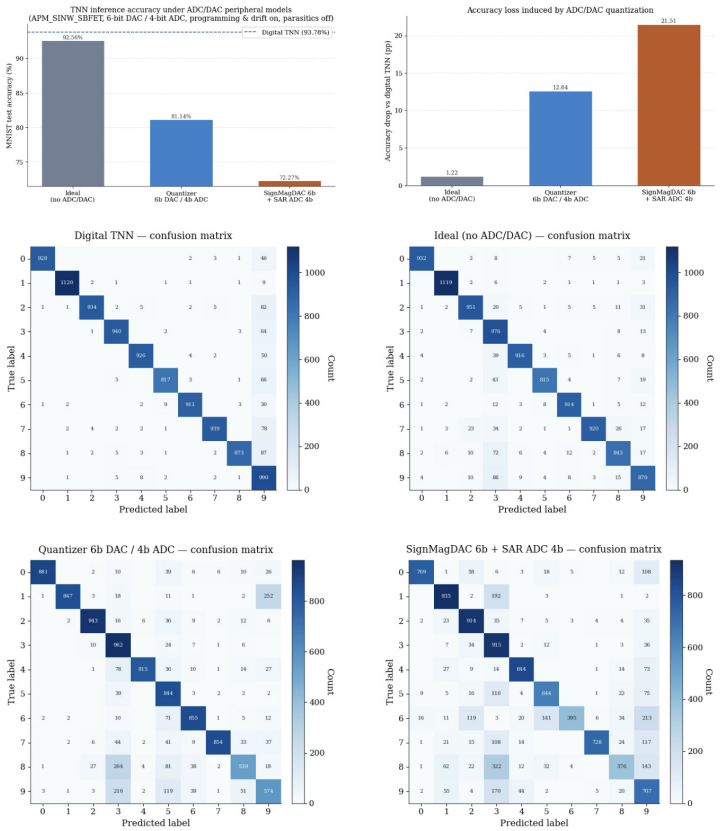


Fig. C.1 Comparison of ADC/DAC models for TNN inference using the APM SiNW SBFET device. Top panels show classification accuracy and loss relative to the digital TNN baseline. Realistic SignMagnitude DAC and SAR ADC implementations cause significantly larger accuracy degradation than the generic CrossSim quantizer model.

Appendix D

Membrane Potentials and Spike Responses

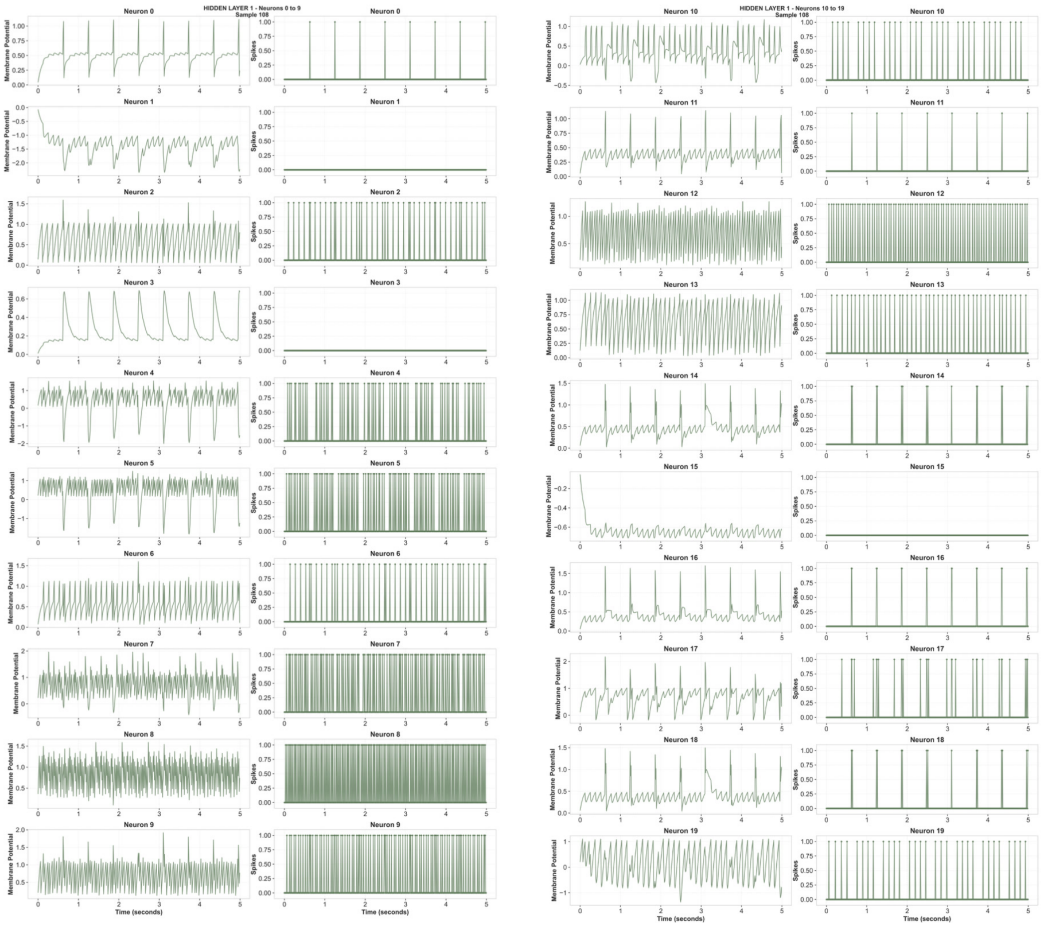


Fig. D.1 Membrane-potential traces and corresponding spike trains for the first 20 neurons of the first hidden layer of the SNN. For each neuron, the left panel shows the membrane potential dynamics and the right panel shows the generated spike activity over the simulation interval.

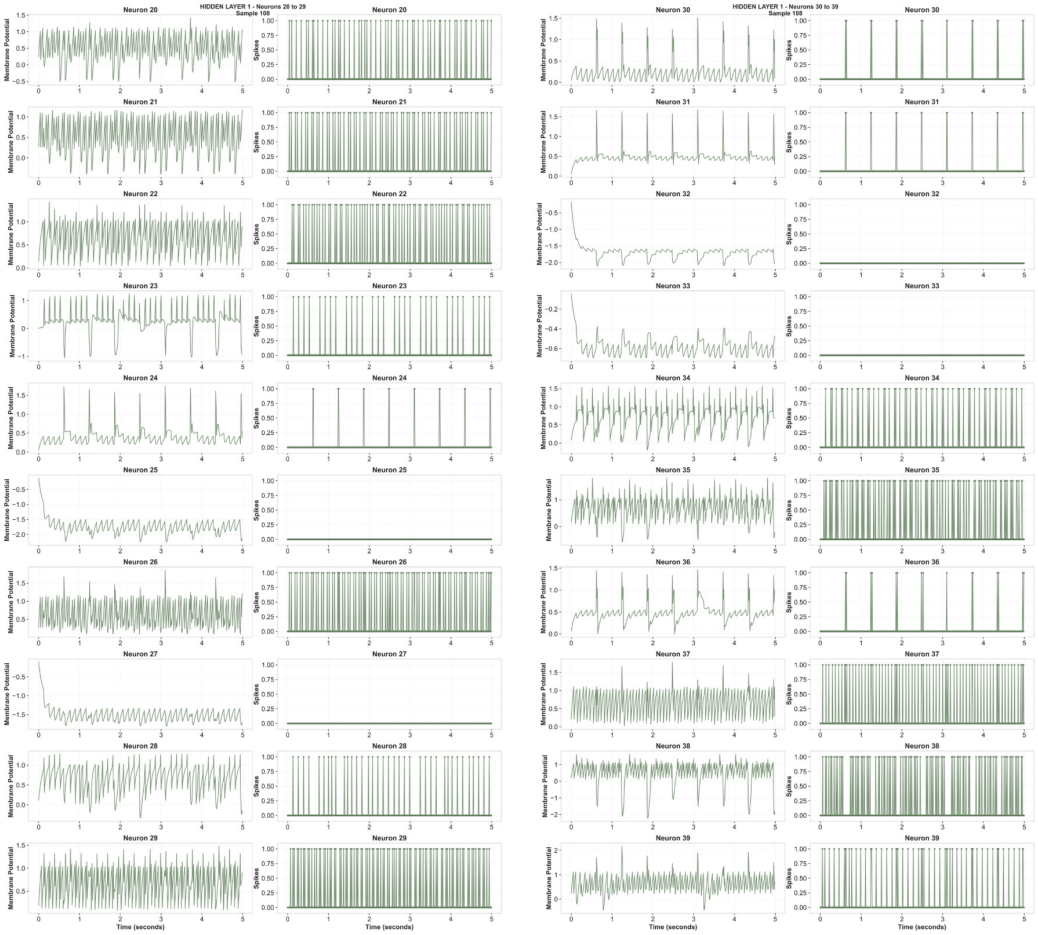


Fig. D.2 Membrane-potential traces and corresponding spike trains for neurons 20–39 of the first hidden layer of the SNN. For each neuron, the left panel shows the membrane potential dynamics and the right panel shows the generated spike activity over the simulation interval.

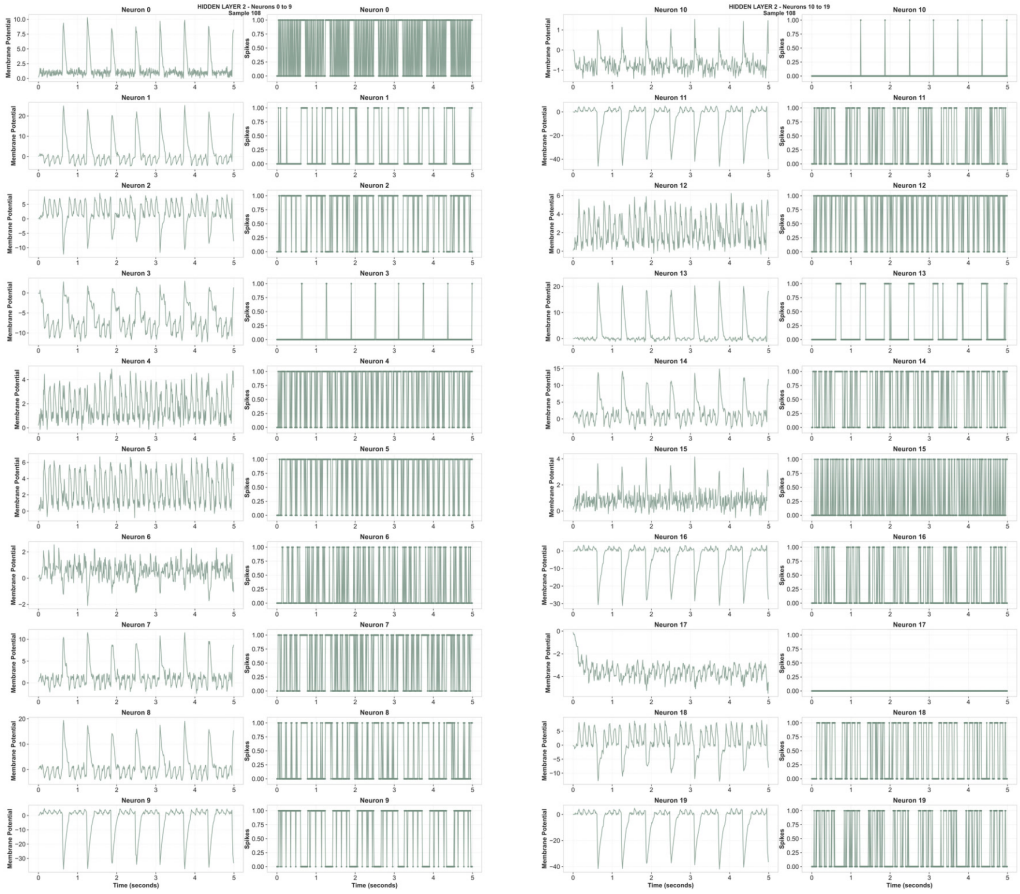


Fig. D.3 Membrane-potential traces and corresponding spike trains for all 20 neurons in the second hidden layer of the SNN. For each neuron, the left panel shows the membrane potential dynamics and the right panel shows the generated spike activity over the simulation interval.

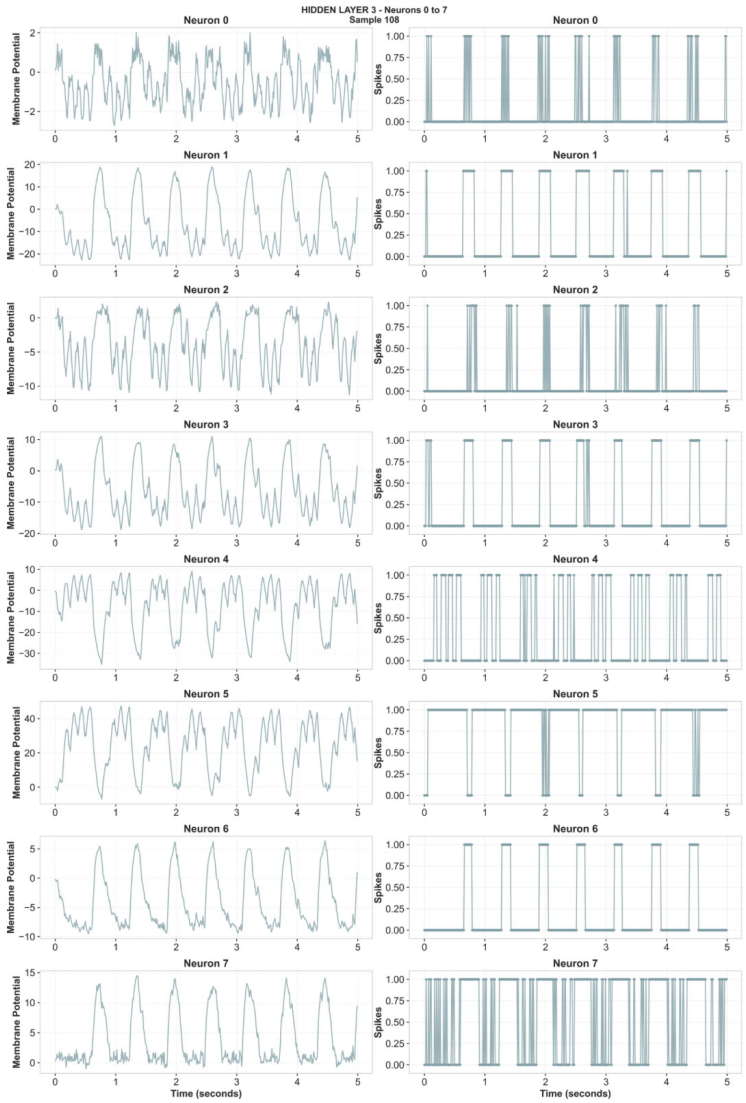


Fig. D.4 Membrane-potential traces and corresponding spike trains for all eight neurons in the third hidden layer of the SNN. For each neuron, the left panel shows the membrane potential dynamics and the right panel shows the generated spike activity over the simulation interval.