Analog-based Neural Network Implementation

Using Hexagonal Boron Nitride Memristors

by

Sahra T. Afshari

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2023 by the
Graduate Supervisory Committee:

Ivan Sanchez Esqueda, Chair
Hugh J. Barnaby
Jae-sun Seo
Yu Cao

ARIZONA STATE UNIVERSITY

May 2023

ABSTRACT

Resistive random-access memory (RRAM) or memristor, is an emerging technology used in neuromorphic computing to exceed the traditional von Neumann obstacle by merging the processing and memory units. Two-dimensional (2D) materials with non-volatile switching behavior can be used as the switching layer of RRAMs, exhibiting superior behavior compared to conventional oxide-based RRAMs. The use of 2D materials allows scaling the resistive switching layer thickness to sub-nanometer dimensions enabling devices to operate with low switching voltages and high programming speeds, offering large improvements in efficiency and performance as well as ultra-dense integration.

This dissertation presents an extensive study of linear and logistic regression algorithms implemented with 1-transistor-1-resistor (1T1R) memristor crossbars arrays. For this task, a simulation platform is used that wraps circuit-level simulations of 1T1R crossbars and physics-based model of RRAM to elucidate the impact of device variability on algorithm accuracy, convergence rate, and precision. Moreover, a smart pulsing strategy is proposed for the practical implementation of synaptic weight updates that can accelerate training in real crossbar architectures.

Next, this dissertation reports on the hardware implementation of analog dot-product operation on arrays of 2D hexagonal boron nitride (h-BN) memristors. This extends beyond previous work that studied isolated device characteristics towards the application of analog neural network accelerators based on 2D memristor arrays. The wafer-level fabrication of the memristor arrays is enabled by large-area transfer of CVD-grown few-layer h-BN films. The dot-product operation shows excellent linearity and

repeatability, with low read energy consumption, with minimal error and deviation over various measurement cycles. Moreover, the successful implementation of a stochastic linear and logistic regression algorithm in 2D h-BN memristor hardware is presented for the classification of noisy images. Additionally, the electrical performance of novel 2D h-BN memristor for SNN applications is extensively investigated. Then, using the experimental behavior of the h-BN memristor as the artificial synapse, an unsupervised spiking neural network (SNN) is simulated for the image classification task. A novel and simple Spike-Timing-Dependent-Plasticity (STDP)-based dropout technique is presented to enhance the recognition task of the h-BN memristor-based SNN.

DEDICATION

To “Women, Life, Freedom”

ACKNOWLEDGMENTS

I want to express my sincere gratitude to my advisor, Dr. Ivan Sanchez Esqueda, for his unwavering support and guidance throughout my PhD journey. His trust, guidance, and mentorship have been instrumental in shaping my research and contributing to my academic growth and success. I would also like to extend my thanks to the members of my committee, Dr. Hugh Barnaby, Dr. Jae-Sun Seo, and Dr. Yu Cao, for their vast expertise, wise counsel, and constant support throughout my doctoral studies. I am particularly grateful to Dr. Barnaby and Dr. Seo for their kindly allowing me to use their equipment for neuromorphic hardware implementation, which was essential to the success of my research. I would like to express my gratitude to my teammates, Jing Xie, Naim Patoary, Guantong Zhou, Fahad Mamun, Mirembe Musisi-Nkambwe, Priyanka Apsangari and Wangxin He, for their invaluable contributions to my research. Jing's efforts in the fabrication and perfection of the 2D h-BN memristor chips have been crucial to the success of my research. I am grateful to Mirembe, Naim, Guantong, and Fahad for their solid support, company, and constructive criticism, as they have assisted me in overcoming obstacles and enhancing my work. I would also like to thank Priyanka and Wangxin for helping me with device testing and characterization. Moreover, I appreciate the efforts of my FURI students, Sritharini Radhakrishnan and Jaafar Al Shamari.

I want to thank my family and friends from the bottom of my heart for their everlasting love and support during my doctoral journey. I am particularly grateful to my amazing sisters and best friends, Zainab and Najmeh, who have always trusted and supported me wholeheartedly. Lastly, I thank Gajanan, my best friend, who has been my constant support system.

TABLE OF CONTENTS

Page

LIST OF TABLES vii

LIST OF FIGURES viii

CHAPTER

1 INTRODUCTION 1

1.1 Background and Motivation 1

1.2 Overview of Artificial Neural Networks 5

1.3 Overview of Neuromorphic Computing 10

1.4 Overview of Memristors as Synapses in Neuromorphic Computing 13

1.5 Goals and Approach 21

2 CIRCUIT-LEVEL IMPLEMENTATION OF REGRESSION ALGORITHMS USING METAL-OXIDE MEMRISTOR ARRAYS 23

2.1 Variability in Oxide-based RRAM 23

2.2 Effects of Variability on Resistive-Switching Characteristic of 1-Transistor-1-Resistor (1T1R) RRAM Cells 26

2.3 Implementation of Linear Regression on Memristor Crossbars 32

2.4 Implementation of Logistic Regression on Memristor Crossbars 38

3 DOT-PRODUCT COMPUTATION AND LOGISTIC REGRESSION WITH 2D HEXAGONAL-BORON NITRIDE (H-BN) MEMRISTOR ARRAYS 45

3.1. H-BN Memristors 45

3.2 Fabrication, Physical and Electrical Behavior of H-BN Memristors 49

3.3 Hardware Implementation of Dot-Product Using H-BN Memristor Array 54

CHAPTER Page

3.4 Hardware Implementation of Stochastic Logistic Regression 57

3.5 Methods 61

4 LINEAR REGRESSION WITH 2D HEXAGONAL-BORON NITRIDE (H-BN) MEMRISTOR ARRAYS 64

4.1 Fabrication of H-BN Memristor Arrays 64

4.2 Resistive-Switching Properties and Multistate Pulse Programmability 64

4.3 Hardware Implementation of Linear Logistic Regression 68

4.4 Methods 70

5 UNSUPERVISED LEARNING IN HEXAGONAL BORON NITRIDE MEMRISTOR-BASED SPIKING NEURAL NETWORKS 74

5.1 Overview of Memristor-based Spiking Neural Networks 74

5.2 Physical and Electrical Characteristics of H-BN Memristor Devices 76

5.3 Implementation of Unsupervised Learning in h-BN Memristor-Based Spiking Neural Network 82

6 CONCLUSION 90

REFERENCES 93

APPENDIX

A GLOSSARY OF NEUROMORPHIC SYSTEMS..... 103

LIST OF TABLES

Table Page

1.1 RSM Metrics Compared to Conventional Memory. 3

1.2 Desirable NVM Metrics for Neuromorphic Computing Applications 18

LIST OF FIGURES

Figure		Page
1.1	Von Neumann vs. Neuromorphic Computing System	2
1.2	Biological Neuron vs. Artificial Neuron	7
1.3	Increase in the Demand for Computing Power	9
1.4	Neuro-Inspired Computing Architecture Solves Memory Bottleneck	12
1.5	Set/Reset Operations in the RRAM Device	15
1.6	In-Memory Computing with RRAM	16
1.7	Comparing Benchmarks in CMOS vs. RRAM Neuromorphic Chips	17
2.1	Variability in Recent Resistive-Switching Devices	25
2.2	Filamentary Operation in RRAM	26
2.3	Schematic of the 1T1R RRAM Cell and Crossbar Array	28
2.4	Variability Simulation in 1T1R RRAM Cell	29
2.5	Electrical Characteristics of 1T1R RRAM Considering Variability	30
2.6	Translation of ΔG for Hardware Implementation of the Gradient-Descent	32
2.7	Implementation of Linear Regression Algorithm on a Memristor Crossbar	34
2.8	Results of Linear Regression Simulation	36
2.9	Implementation of Logistic Regression Algorithm on a Memristor Crossbar	39
2.10	Results for Logistic Regression	41
2.11	Smart Pulsing vs. Single Pulsing	44
3.1	Comparison of Set Energy Consumption vs. Set Switching Energy.	47
3.2	2D h-BN Memristor Array	49
3.3	Fabrication and Physical Characteristics of H-BN Arrays	50

Figure Page

3.4 Electrical Characteristics of H-BN Memristors 52

3.5 Dot Product Computation Using H-BN Arrays 54

3.6 Implementation of Logistic Regression on H-BN Arrays.....57

3.7 Results of Logistic Regression Algorithm on H-BN Array 59

3.8 Electrical Characterization Test Setup Using Keithley 4200.....63

3.9 Logistic Regression Test Setup Using NI.....63

4.1 Hexagonal Boron Nitride (H-BN) Memristor Arrays 65

4.2 Resistive Switching Characteristics of H-BN Memristors.....66

4.3 Multi-State Non-Volatile Pulse Programming of H-BN Memristors.....67

4.4 Implementation of Linear Regression on H-BN Memristor Arrays 68

4.5 Electrical Characterization and Linear Regression Test Setup 71

5.1 Physical Characteristics of H-BN Memristor Arrays Wafer 76

5.2 Resistive Switching and Pulsing Characteristics of H-BN Memristors 77

5.3 Biological vs. Memristor-Based Artificial Neural Network 80

5.4 Implementation of SNN on H-BN Memristor Crossbar 81

5.5 Simplified STDP vs. STDP-Based Weight Dropout.....84

5.6 Results for Simplified STDP vs. STDP-Based Weight Dropout.....87

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

A study presented that the computer consumption used by the largest artificial intelligence (AI) training is doubling every few months compared to the Moore's law scaling, doubling every two years (Sarker). Modern digital computers, following von Neumann architecture, use a lot of power, making them an unsustainable platform for artificial intelligence applications. Due to the physical separation between the processing unit and the memory unit, which causes high power consumption, processors are required to spend the majority of their time transporting data between the two units (Sarker). Therefore, it is important to find alternative solutions.

A solution is brain-inspired computing also known as neuromorphic computing. The brain is highly functional and power-efficient, prompting scientists to look for ways to model data storage and processing after its fundamental principles (Sarker; Montesinos López et al.). The term "neuromorphic" refers to an architecture that draws inspiration from the human brain. It employs a number of methods similar to those found in the brain, such as combining memory and processing units in one location to maximize parallelism or using spike-based data that is resistant to noise found in nature (Montesinos López et al.). Figure 1.1a shows the architecture of von Neumann computing system compared to neuromorphic computing system in Figure 1.1b.

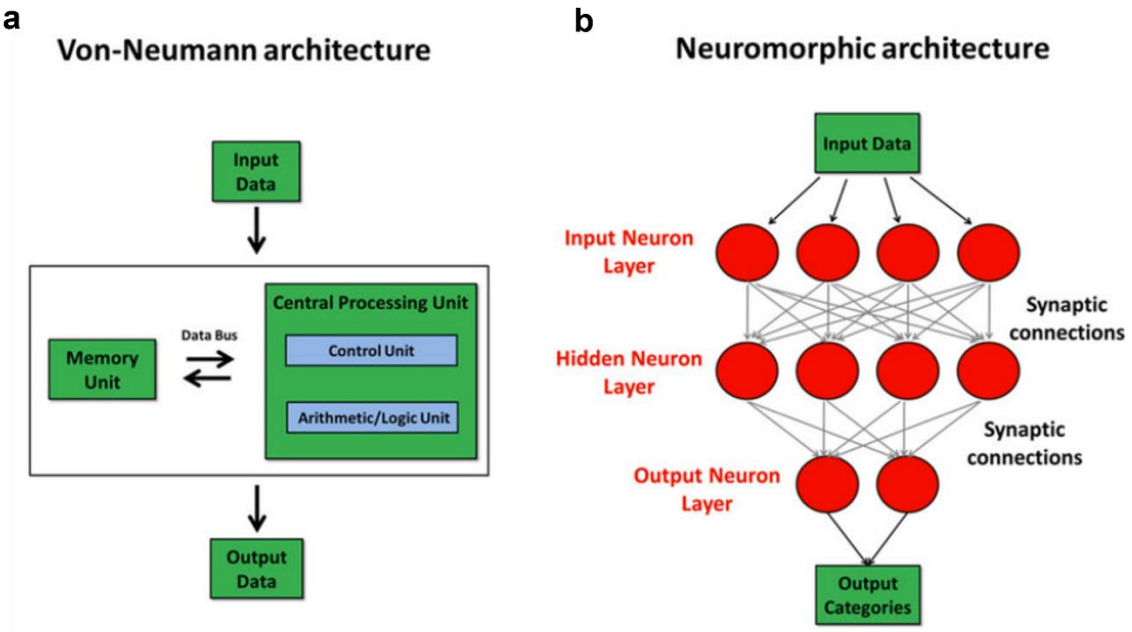


Figure 1.1. (a) von Neumann computing system, (b) neuromorphic computing system (del Valle et al.).

Numerous innovative technologies that draw inspiration from biology can act as synapses in neuromorphic computing (Khan). One of the fundamental technologies for implementing the neuromorphic computer system is the developing analog-type resistive switching memory (RSM) which enables in-memory architecture (Khan). RSM is a two-terminal nonvolatile memory device that has the ability for analog programmability. It stores data in the form of various conductance levels (Khan). These devices can act as synaptic weights to simultaneously store and process input signals (Jo et al.).

RSM can perform vector matrix multiplication (VMM) in a single step by measuring the cumulative output current using Ohm's law and Kirchhoff's law, resulting in high parallelism (Yao et al.). Analog RSM provides many advantages over conventional memory technologies like static random-access memory (SRAM) (Y.-H. Chen et al.) and Flash (Zhao et al.). Although SRAM technology is quick and has a developed manufacturing process due to CMOS scaling down, large-scale SRAM arrays are not

desirable due to their poor area efficiency and high standby power (Burr et al.). Flash technology is a type of nonvolatile memory that exhibits analog programmability (Merrikh-Bayat et al.). Table 1.1 compares the performance metrics of RSM device with Flash and SRAM (B. Li et al.). However, analog RSMs exhibit better switching speed, lower programming voltage, and higher endurance capacity when compared to Flash technology (Lee et al.). Therefore, RSM with such superior properties can benefit neuromorphic computing and significantly improve area efficiency.

Technology	RSM	SRAM	Flash (NOR)
Non-volatility	Yes	No	Yes
Cell size (F^2)	≤ 2	> 100	10
Write energy (pJ/bit)	$\sim 10^{-13}$	$\sim 10^{-15}$	$\sim 10^{-10}$
Read time (ns)	<10	1-3	10
Write time (ns)	<10	1-3	1000
Retention	10 years	5.4 years	10 years
Endurance (cycles)	10^{12}	$> 10^{16}$	10^{5}

Table 1.1 RSM metrics compared to conventional memory.

Compared to conventional digital memory circuits, resistive random-access memory (RRAM or memristor) technology, a subset of RSM, is a great candidate for nonvolatile memory (NVM) due to its low power consumption, excellent scalability, high-speed functionality, CMOS compatibility, and analog programmability (i.e., the ability to retain analog values) (Prakash et al.; Wong et al.). Memristors are typically two-terminal devices made up of two metal layers with an insulating or switching layer (such as an oxide) sandwiched in between (Wong et al.). If we consider RRAM as a binary device for NVM applications, these devices can be programmed into either a low resistance state (LRS) or a high resistance state (HRS) (Wong et al.). Furthermore, programming a continuous range of states is essential for neuromorphic architectures and RRAM analog-based implementations of in-memory computing. RRAM for NVM has also been used to

show multistate storage properties (Patel et al.). The formation and rupture of conductive filaments inside the cell's oxide/switching layer allow for the programming of various resistance states (Patel et al.). The stochastic nature of conductive filament activity introduces variability, programming abruptness, and non-linearity, which may present a significant challenge for machine learning applications using RRAM devices (Zhao et al.). As presented in (Zhao et al.), the basic reliability metrics relate to endurance, retention, noise, and write/read disturbs. Other reliability metrics, referred to as "functional" reliability metrics, include non-linearity, variability, dynamic range, precision, variation, asymmetry, and so on (Zhao et al.). These metrics directly affect the accuracy of neuromorphic computing (Zhao et al.). In the first section of this study, we concentrate on another functional reliability issue, namely, the *variability* in RRAM features, and its effects on gradient descent-based neural network training (convergence rate, accuracy, and precision).

In recent years, researchers have discovered that a number of 2D materials also exhibit memristive phenomena, expanding the category of non-volatile resistive switching materials to include a vast array of ultrathin layered crystalline films (R. Ge et al.; Rehman et al.; Pradhan et al.). These 2D memristors can help alleviate some of the non-idealities of oxide-based RRAM (Kumar et al.). For example, the layered structure of 2D materials could help minimize variation in resistive switching layer thickness to provide a more robust implementation of STDP (Chaudhuri and Chakrabarty). Moreover, compared to oxide-based RRAM, synaptic change occurs in confined and chemically stable defects surrounded by crystalline h-BN (Kumar et al.). Other factors that can further help enhance the energy efficiency of SNN are the low

programming voltages (J. Ge et al.) and fast switching speeds (Zhu, Liang, et al.; Wu et al.) of 2D-material-based memristors. A characteristic 2D insulator known as hexagonal boron nitride (h-BN) has been shown to exhibit resistive switching behavior in multilayer nanosheets (C. Pan et al.). These 2D-based memristors have characteristics such as forming-free operation, high on/off ratio ($>10^6$), quick switching speed (20 ns), and low switching voltage (1 V). In the second section of this research, the first-ever demonstration of machine learning algorithms has been done using novel h-BN 2D-based memristor arrays.

1.2 Overview of Artificial Neural Networks

Artificial intelligence (AI) has a branch called machine learning that allows computers to learn from data on their own and get better over time. To do this, machine learning models are trained on big datasets and given time to discover patterns and connections in the data. Following this learning, the models can use raw data to generate predictions or judgments (Sarker). Artificial neural networks (ANNs) are a type of machine learning technique that is loosely modeled after the structure and function of the human brain. ANNs consist of layers of interconnected nodes, also known as neurons, that process information and learn to recognize patterns in input data (Montesinos López et al.). ANNs are capable of learning complex, non-linear relationships between inputs and outputs, and therefore, can simulate non-linear systems (Waterworth and Lees). For example, in drug discovery, the relationship between a molecule's structure and its activity against a particular target may be highly non-linear, making it difficult to predict using conventional

linear models. ANNs are able to capture this non-linearity, enabling more accurate predictions (Jiménez-Luna et al.).

The applications of ANNs are vast and diverse, and span a variety of industries and domains. ANNs are used extensively in image recognition applications such as facial recognition and object detection (Khan). ANNs are also used for speech recognition applications, such as voice-activated assistants like Siri and Alexa. In a study by Yasar et al., an ANN-based approach was employed for the classification of Parkinson's disease using speech signals (Yasar et al.). Moreover, ANNs are used for natural language processing applications such as sentiment analysis and machine translation. An ANN-based methodology was utilized in a study to analyze the sentiment of internet reviews (Niharika and Malhotra). In addition, ANNs are employed for financial forecasting applications such as stock market prediction and fraud detection. In a study published in ACM Southeast Conference, an ANN-based stock price prediction and trading system was developed which achieve comparable results against the Buy and Hold approach (Sezer et al.). Moreover, ANN is widely used in healthcare applications such as disease diagnosis and drug discovery. Wang et al. demonstrated that ANN models can increase the efficacy and precision of tumor image segmentation (Wang et al.). In a review study, ANN-based pharmaceutical research such as drug modeling, analytical data analysis, protein structure and function, dosage optimization and manufacturing have been discussed (Sutariya et al.). These are only a few examples of the numerous uses for ANNs in various industries. Future applications of ANNs are likely to be much more inventive as technology develops.

Artificial neural networks (ANNs) learn through a process called training, which involves adjusting the weights and biases of the neurons in the network based on a set of

input-output pairs. There are several types of learning algorithms used in ANNs, but one of the most common is called backpropagation (Nigrin). In backpropagation, the network is first fed a set of inputs, and the output is calculated using the current weights and biases of the neurons. The output is then compared to the desired output, and an error signal is calculated. The error signal represents the difference between the predicted output and the desired output (Nigrin). The error signal is then propagated back through the network, and the weights and biases of the neurons are adjusted based on the magnitude and direction of the error signal. This process is repeated for many input-output pairs, gradually adjusting the weights and biases of the neurons until the network is able to accurately predict outputs for a given set of inputs. The learning process in ANNs is often iterative, meaning that the network is trained using a subset of the available data, and then tested on a separate subset of data to evaluate its performance. If the network's performance is not satisfactory, the

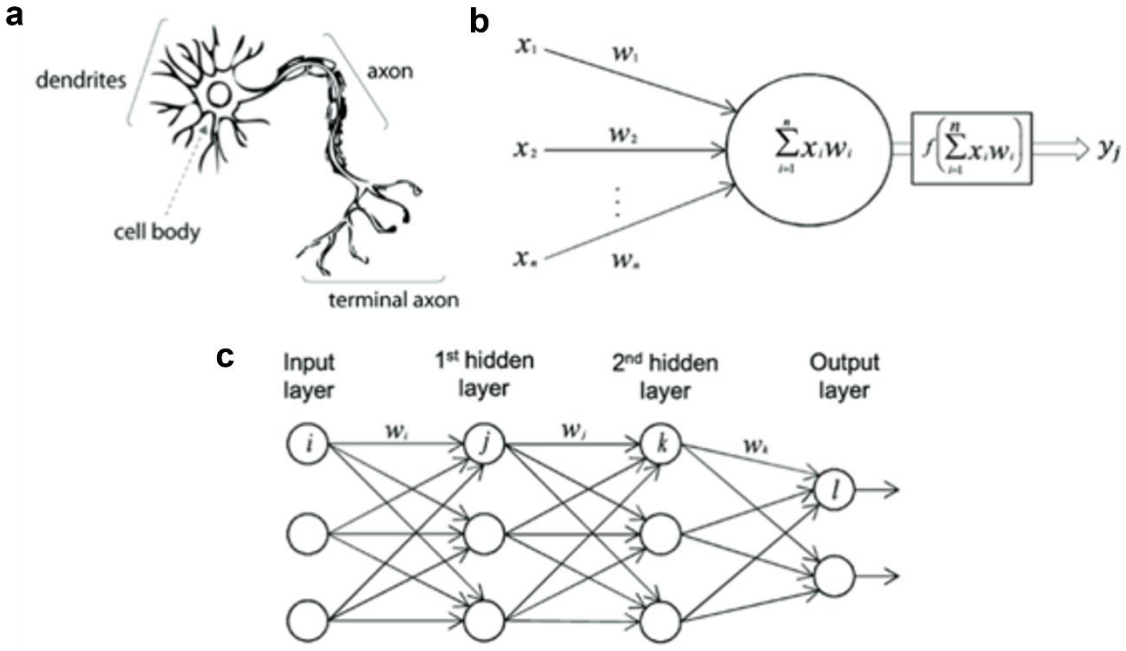


Figure 1.2. A biological neuron compared to an artificial neuron: (a) human neuron, (b) mathematical model for artificial neuron, (c) ANN model (Meng et al.).

weights and biases are further adjusted, and the training process is repeated (Nigrin). It is important to note that the success of an ANN's learning depends on a number of factors, including the size and complexity of the network, the quality and quantity of the training data, and the choice of learning algorithm. But with careful design and training, ANNs can be highly effective at learning from data and making accurate predictions (Schmidhuber).

An artificial neuron, also known as a perceptron, is a basic unit of computation in an artificial neural network allowing them to perform complex computations and learn from data. It is a mathematical function that takes inputs, performs a set of calculations on those inputs, and produces an output. The inputs to an artificial neuron are typically real-valued numbers, and each input is assigned a weight that reflects its importance in the calculation. The neuron also has a bias term, which is a constant value that is added to the weighted sum of the inputs before being passed through an activation function (Schmidhuber; Rosenblatt). The activation function is a non-linear function that determines the output of the neuron. There are several types of activation functions used in ANNs, such as the sigmoid, ReLU, and tanh functions. The output of an artificial neuron is then passed on to the next layer of neurons in the network, or it may be the final output of the network if it is a single-layer perceptron. Figure 1.2a and 1.2b show a biological neuron in comparison to an artificial neuron. Figure 1.2c shows an artificial neural network with two hidden layers (Meng et al.).

Deep learning algorithms based on artificial neural networks require enormous matrix operations and are often trained on large datasets containing millions or billions of data points. This means that the matrix operations involved in training the model need to be performed many times, which can be very computationally intensive. As a result, training deep learning models often requires specialized hardware for edge application, such as graphics processing units (GPUs) or tensor processing units (TPUs), that can perform large-scale matrix operations efficiently (Taher; Strubell et al.; Sun and Kist; Hosseininoorbin et al.). Figure 1.3 shows a significant increase in computational power demand over the past 40 years measured in petaFLOPS days (Mehonic and Kenyon). From 1970 until 2012, the demand for computing power doubled every 24 months. However, in more recent times, this doubling has occurred at a much faster rate, approximately every 2 months (Mehonic and Kenyon). The same study demonstrates that training costs have

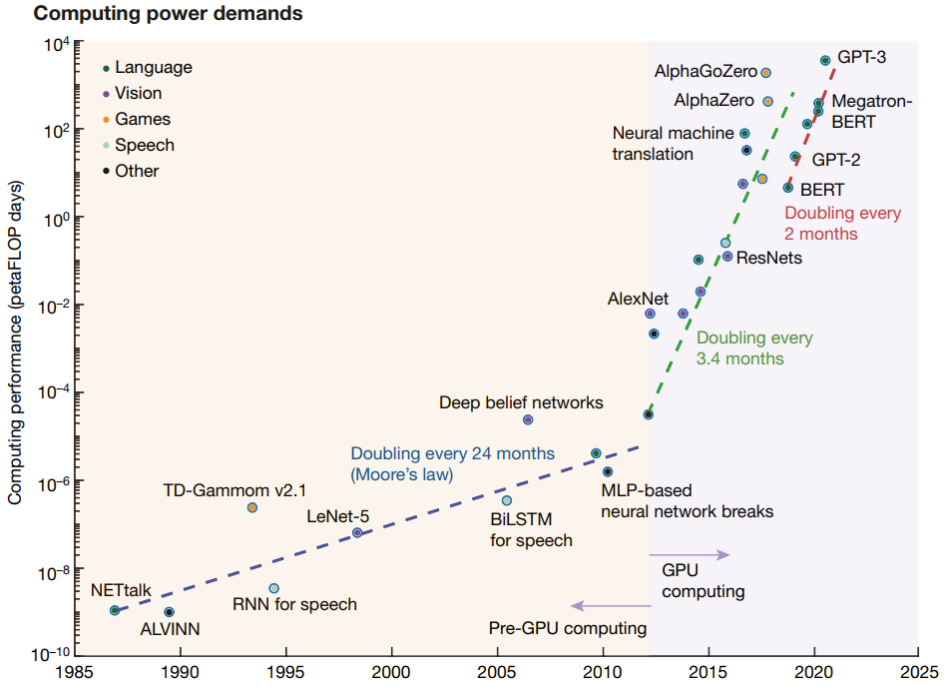


Figure 1.3. Increase in the demand for computing power measured in petaFLOPS days (Mehonic and Kenyon).

climbed exponentially over the past decade and that hardware demand has increased more than 300 times between 2016 and 2021 (Mehonic and Kenyon). With all of them, it is clear that the conventional CMOS technology is not sustainable.

Neuromorphic computing has emerged as a potential solution to the challenges posed by the end of Moore's Law. As the traditional computing paradigm reaches its limits in terms of speed and energy efficiency, neuromorphic computing offers an alternative approach that is modeled after the human brain. Instead of relying on a central processing unit (CPU) and sequential processing of instructions, neuromorphic computing utilizes networks of artificial neurons and synapses that are designed to mimic the behavior of biological neurons. These networks can process information in parallel and adapt to new data, making them well-suited for tasks such as pattern recognition and decision-making (Christensen et al.). Even though neuromorphic computing has the potential to offer significant improvements in energy efficiency and processing speed, there are still many challenges to overcome in terms of hardware design and programming paradigms before neuromorphic computing can become a mainstream technology.

1.3 Overview of Neuromorphic Computing

Despite remarkable advancements in ANNs, biological neural networks continue to outperform ANNs in terms of energy efficiency and capabilities for online learning. The brain consumes only ~20 W despite containing 10^9 neurons and 10^{13} synapses (Sandberg). Since the brain is highly functional and power-efficient, scientists have been looking for ways to store and process data fundamentally differently by modeling the brain. Von Neumann architecture, which is the dominant architecture used

in conventional computers, separates memory and processing units and operates by manipulating data stored in memory using a central processing unit (CPU) through a sequential sequence of instructions. While this approach is highly flexible and suitable for a wide range of computing tasks, it can also be highly inefficient when dealing with certain types of data-intensive and computationally complex tasks, such as deep learning.

Neuromorphic computing, on the other hand, is a field of computer engineering that is inspired by the structure and function of the human brain, and it aims to develop computing systems that can process information in a more efficient and brain-like way. Neuromorphic computing has the potential to address the high computational demands of deep learning and other complex machine learning tasks. This is achieved by mimicking the neural networks and synapses of the brain, which are capable of processing large amounts of data in parallel with high accuracy, computational, and power efficiency. The term "neuromorphic" refers to a hardware architecture that draws inspiration from the human brain. It employs a number of methods similar to those found in the brain, such as combining memory and processing units in one location to maximize parallelism or using spike-based data that is resistant to noise found in nature (Mehonic and Kenyon). Figure 1.4a shows the architecture of the von-Neumann computing system compared to the neuromorphic computing system in Figure 1.4b (Zhang et al.). As observed in Figure 1.4a, the central processing unit (CPU) and memory unit are separated in von-Neumann design, coupled by a bus, and connected to external input and output devices. The CPU itself consists of various units such as the control unit, logic/arithmetic unit, and registers. The

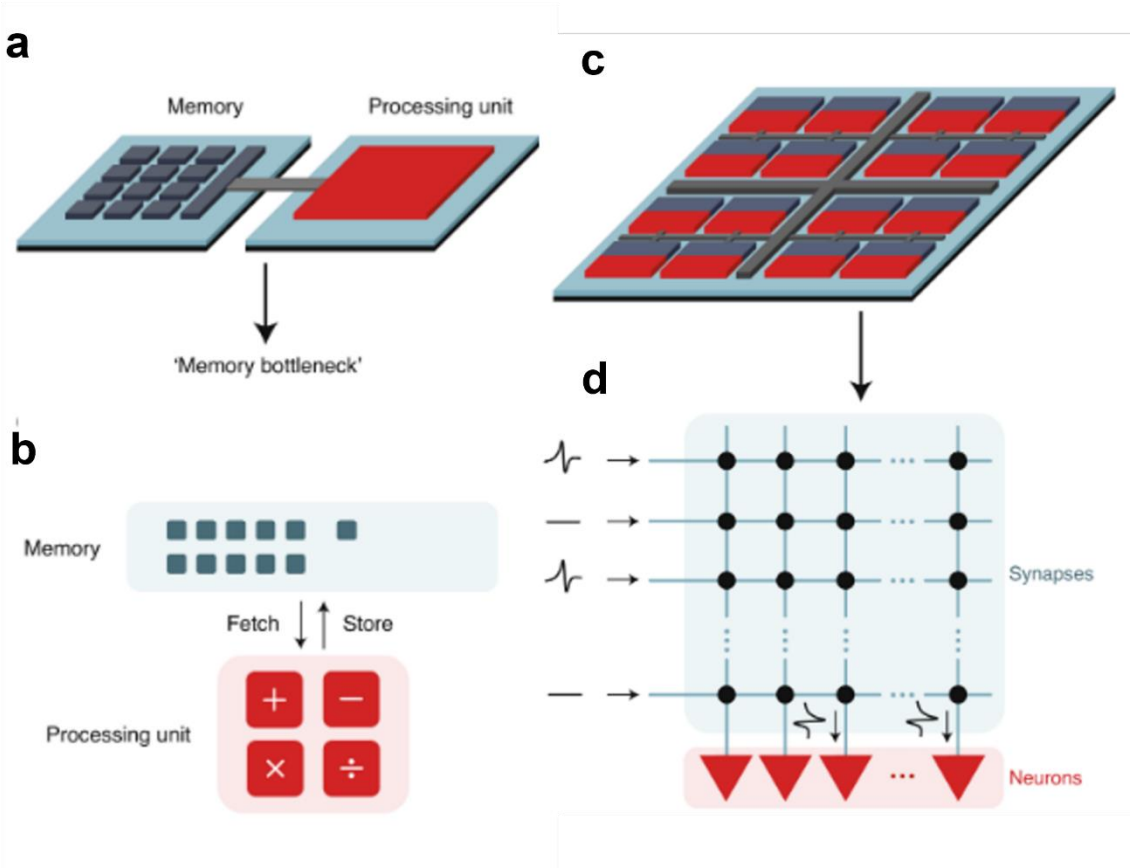


Figure 1.4. (a) Memory bottleneck in von Neumann architecture, (b) computing paradigm in von Neumann architecture when data are fetched from or stored to memory, (c) neuro-inspired architecture, (d) computing paradigm in neuro-inspired computing architecture (Zhang et al.).

CPU fetches data and instruction from the memory unit, and after processing, the results are stored back into the memory unit. The fetch/storage of data between memory unit and CPU is the main bottleneck of power consumption in modern digital computers. On the other hand, in neuromorphic architecture, hardware that mimics brain structure, parallel computation and memory units are governed by neurons and synapses. Synaptic weights are used to connect each nearby input and output neuron, and simple vector matrix multiplication (VMM) computations are carried out. Compared to von-Neumann processors with explicit instructions, programs in the neuromorphic model are defined by neural network structures and their parameters.

Moreover, neuromorphic computing systems can be designed to be more fault-tolerant and adaptable than traditional computing systems. This is because they can reconfigure their connectivity and processing resources in response to changing inputs or environmental conditions, much like the brain adapts to new situations (Davies et al.). Researchers are currently exploring a variety of hardware and software architectures for neuromorphic computing, including spiking neural networks, memristor-based systems, and field-programmable gate arrays (FPGAs) (“Beyond von Neumann”; Indiveri et al.; Merolla et al.).

1.4 Overview of Memristors as Synapses in Neuromorphic Computing

Resistive Random Access Memory (RRAM), also known as memristor, is a type of non-volatile memory that operates based on resistance switching of a thin film between two metal electrodes. The term "memristor" was first coined in 1971 by Leon Chua, a professor at the University of California, Berkeley, to describe a hypothetical fourth fundamental circuit element alongside the resistor, capacitor, and inductor (Chua). Memristor has been the subject of much research and development in recent years because of its promising properties, including high density, low power consumption, and fast switching speeds (Ielmini; F. Pan et al.).

Memristors have been demonstrated to have potential for use in a variety of applications, including analog circuits, digital memory, and neural networks (F. Pan et al.). In recent years, memristors have received increasing attention as a promising technology for brain-inspired computing, due to their ability to emulate the synaptic plasticity and dynamics of biological synapses (F. Pan et al.).

Memristor operates by utilizing a thin film of a material, typically an oxide, sandwiched between two metal electrodes. The oxide film is initially in a high resistance state (HRS) but can be switched to a low resistance state (LRS) by applying a voltage or current pulse. The resistance of the oxide film can then be read by applying a small voltage to the electrodes, which produces a measurable current. Memristors operate by utilizing 'set' and 'reset' processes to change the resistance state of a memory cell. During the "set" operation, a voltage is applied to the RRAM cell which causes the formation of a conductive filament or pathway within the insulating layer, resulting in a low resistance state. This low resistance state is often referred to as the "ON" state. During the "reset" operation, a voltage is applied in the opposite direction, which causes the filament or pathway to break down, returning the RRAM cell to its high resistance state. This high resistance state is often referred to as the "OFF" state. The ability to switch between the ON and OFF states is what allows RRAM to function as a memory device. The resistance values of the ON and OFF states can be used to represent binary values, with the ON state representing a "1" and the OFF state representing a "0" (YingTao Li et al.). Figure 1.5. shows the set /reset operations performed in oxide-based RRAM by formation and rupture of conductive filament (CF) (Ambrosi et al.).

RRAM can be divided into the following two categories based on the nature of the conductive filament: (i) Conductive bridge random access memory (CBRAM), also referred to as electrochemical metallization memory (ECM), which is based on metal ions; and (ii) oxygen vacancies filament-based RRAM (OxRRAM), also known as valence change memory (VCM), which is based on oxygen vacancies filaments (Zahoor et al.).

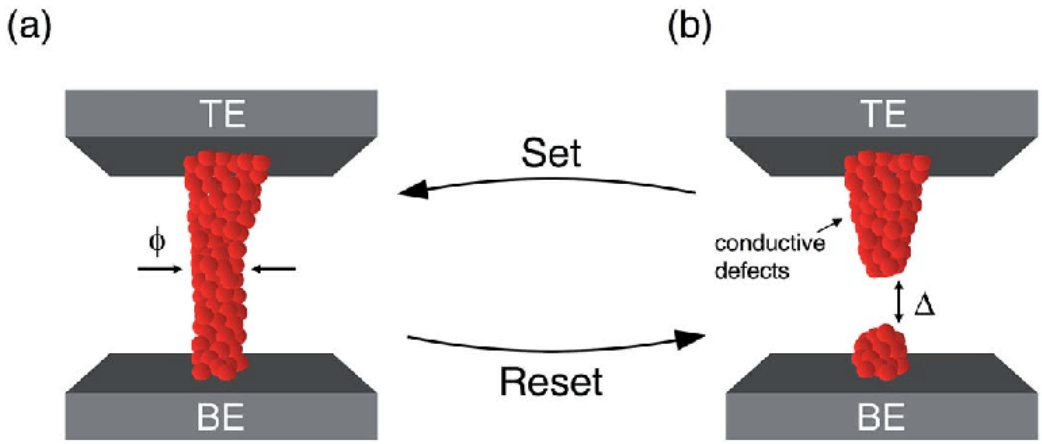


Figure 1.5. Set/reset operations in the RRAM device. By set process, device goes from HRS to LRS by forming conductive filament connecting the top electrode and the bottom electrode. In reset process the conductive filament is ruptured, and device goes to HRS.

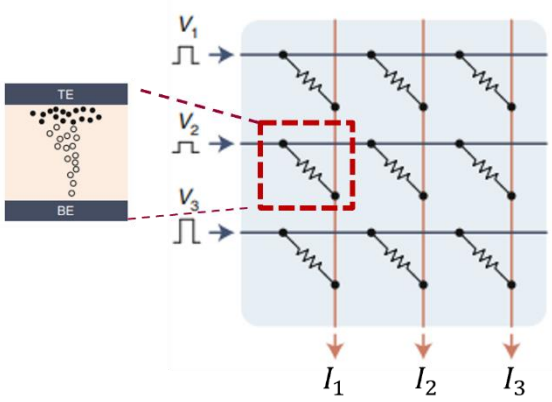
RRAM has several advantages over other non-volatile memory technologies, such as Flash and Static Random Access Memory (SRAM). RRAM has a higher density than Flash, due to its smaller cell size and ability to stack multiple layers of memory cells. RRAM also has a lower power consumption than Flash and SRAM, due to its low operating voltage and ability to operate with low write currents. Additionally, RRAM has a faster switching speed than Flash and SRAM, due to its simpler write and read operations. Moreover, RRAM can perform vector matrix multiplication (VMM) in a single step by measuring the cumulative output current using Ohm's law and Kirchhoff's law, which results in high parallelism as shown in Figure 1.6. Therefore, RRAM with such superior properties can benefit neuromorphic computing.

Recently, RRAM neuromorphic chips have shown improved performance compared to conventional digital neuromorphic computing (Zhang et al.). Four important benchmarking measures are used to assess the performance of the neuromorphic chips (Zhang et al.): (i) Computation density, defined as the chip's efficiency. (ii) Power efficiency, an important factor in overcoming the power consumption gap between

biological brains and neuromorphic systems. (iii) Computation accuracy, which is impacted by imperfections in the hardware, such as thermal noise or reliability problems. Consequently, hardware accuracy is lower than simulation. (iv) The ability to learn: In most conventional chips, learning is carried out in the cloud, and learned parameters are transferred to edge devices. On-chip learning is necessary, nevertheless, for security, quick adaptation, and occasionally customization (Zhang et al.).

Figure 1.7a shows improved computing densities for ANN and spiking neural network (SNN) chips based on RRAM technologies compared to CMOS-based neuromorphic chips. Additionally, in Figure 1.7b, we see improvement for RRAM-based neuromorphic chips in terms of synaptic operation energy (Zhang et al.).

There are also a few application-dependent device metric requirements for RRAMs that affect the learning accuracy of artificial neural networks (Zhang et al.), including the number of analog states (the weight tuning precision), on/off ratio (dynamic range),



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

Figure 1.6. In-memory computing with RRAM. RRAM (i) combines analog computing and data storage at the device level, (ii) use RRAM conductance as analog synaptic weights, and (iii) One-step vector matrix multiplication using Ohm's law and Kirchhoff's law.

linearity (conductance tuning linearity), asymmetry/abruptness (the trajectory of the weight increase/decrease process), endurance, retention, and yield (Zhang et al.). RRAM hardware can store more than just binary data because it can have many analog states. They can therefore be utilized in systems like artificial intelligence and neuromorphic computing. Moreover, they frequently have a high on/off ratio, which means that there is a substantial difference in the resistance of the device between its on and off states. They are therefore advantageous for uses like sensing and communications that demand high signal-to-noise ratios. The ability of RRAM devices to provide a linear output in response to a linear input is referred to as linearity. For applications such as digital-to-analog converters, RRAM devices are helpful because they often display strong linearity. The ability of RRAM devices to switch smoothly between their on and off states is referred to as their asymmetry or abruptness. The ideal transition between these states for RRAM devices should be sharp with little hysteresis. This is important for applications such as memory and logic circuits. The number of times an RRAM device may be switched between the on and off states without degrading or failing is referred to as its endurance. RRAM devices are appropriate

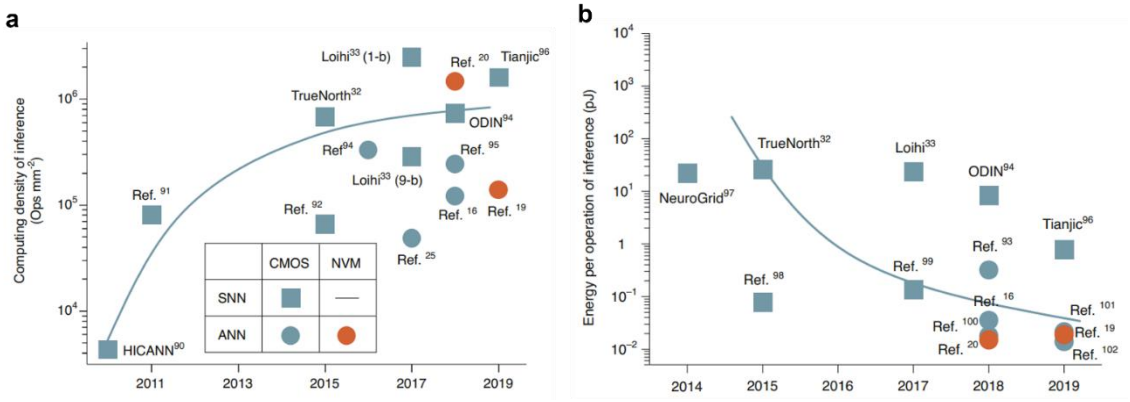


Figure 1.7. Comparing benchmarks in CMOS-based vs. RRAM-based neuromorphic computing. (a) Benchmarking computing density, (b) benchmarking synaptic operation energy (Zhang et al.).

for use in non-volatile memory applications because they typically have good endurance. The ability of RRAM devices to maintain their on or off state over time, even when they are not being actively switched, is referred to as retention. RRAM devices often have strong retention, which is why they are advantageous for uses like data storage. The percentage of functional, standard-compliant devices on a wafer is referred to as the yield of RRAM devices. RRAM devices often have high yields, which are crucial for commercialization and mass production.

In terms of power efficiency, on-state resistance and write voltage are two additional metrics. The system's on-state resistance is a critical measure to assess its energy effectiveness. The crossbar array's current is inversely proportional to its resistance. In terms of write voltage, we require write voltages between 0.5 and 1 V, which can significantly lower the write energy consumption. Table 1.2 shows the desirable metrics for RRAM devices.

Parameters	Targets
Asymmetry/Non-linearity	0/0
Precision	64
On/off ratio	>10
Retention	>10 years
Endurance	$>10^5$ cycles
On-state Resistance (R_{ON})	100 kΩ -10 MΩ
Write Voltage	0.5 V - 1V

Table 1.2 Desirable NVM metrics for neuromorphic computing applications

However, conventional oxide-based memristor technologies present challenges such as a limited conductance range (Gokmen and Vlasov), asymmetric potentiation and depression characteristics, nonlinearity, and variability. As stated before, these non-idealities can affect neuromorphic system performance and efficiency (Christensen et al.; Degraeve et al.; Afshari et al.). Memristors based on

two-dimensional (2D) materials can help alleviate some of the non-idealities of oxide-based RRAM towards more efficient and better performing of neuromorphic computing. 2D materials have attracted significant interest for the downscaling of CMOS (complementary metal-oxide-semiconductor) (Huyghebaert et al.), as well as for beyond-CMOS electronic applications (Robinson). Their atomic scale thicknesses and pristine (i.e., dangling-bond free) surfaces could enable ultra-dense integration for next-generation integrated electronic systems (Lemme et al.). Consequently, many studies have evolved from the demonstration of isolated devices (e.g., field effect transistors or FETs) based on exfoliated flakes towards large-area methods for fabrication of integrated circuits with 2D materials (Quellmalz et al.). While early device demonstrations focused predominantly on FET applications, recent studies have proposed memory and neuromorphic devices based on the non-volatile resistive-switching (NVRS) behavior observed in various 2D materials such as transition metal dichalcogenides (TMD) (R. Ge et al.), hexagonal boron nitride (h-BN) (Nikam et al.), black phosphorus (Rehman et al.), and graphene (Pradhan et al.). These devices are generally configured in vertical two-terminal structures, where the resistive switching layer is sandwiched between top and bottom metal electrodes. The use of 2D materials has enabled the demonstration of devices with atomically thin resistive switching layers having potential advantages. For example, the layered structure of 2D materials could help minimize variation in resistive switching layer thickness to provide a more robust implementation of synaptic operations (Chaudhuri and Chakrabarty). Moreover, compared to oxide-based RRAM, synaptic plasticity (long-term potentiation and depression) can be better controlled in CVD-grown h-BN memristors as filament formation/dissolution

occurs in confined and chemically stable defects surrounded by crystalline h-BN (Kumar et al.). Other factors that can further help enhance the energy efficiency of neuromorphic computing are the low programming voltages (J. Ge et al.) and fast switching speeds (Zhu, Liang, et al.) of 2D-material-based memristors.

Chemical vapor deposition (CVD)-grown h-BN has attracted much attention for use as the resistive switching layer due to its compatibility with large-area wafer-scale fabrication, and arrays of h-BN memristors have been reported (S. Chen et al.). In CVD-grown h-BN devices, the resistive switching process is attributed to the formation and rupture of conductive paths via penetration of metal ions into defects at h-BN grain boundaries. Initial studies of h-BN memristors reported on their non-volatile resistive switching behavior observed as transitions or hysteresis in measurements of DC current–voltage characteristics (Wu et al.). Previous work (S. Chen et al.) has also shown the programming of multiple resistive states in h-BN memristors by the application of consecutive voltage pulses. Pulsed programming is required for practical memory and neuromorphic computing applications. Moreover, the pulsed programming of multiple conductive states is critical for the implementation of synaptic plasticity (i.e., long-term potentiation and depression) in neuromorphic hardware, as well as for the analog-based implementation of machine learning functions in memristor arrays (Musisi-Nkambwe et al.; Huh et al.). For example, most analog-based implementations of neural networks and/or machine learning hardware based on memristor crossbars rely on dot-product (i.e., multiply-accumulate) operations (Mahmoodi et al.; Hu et al.). Here, the accumulated currents at the outputs of the array result from the product of input voltage signals (input vector) and the conductance of the memristors in the array (column vectors) (Xie et al.).

In this work, we present the wafer-scale fabrication of memristor arrays using on CVD-grown h-BN resistive switching layers, and their multi-state analog programmability. We focus on the experimental demonstration of dot-product operation on h-BN memristor arrays and on the hardware implementation of multi-variable stochastic linear and logistic regression. This work extends beyond existing demonstrations of NVRS behavior in isolated h-BN memristors and paves the way for more sophisticated demonstrations of machine learning applications based on 2D materials.

1.5 Goals and Approach

This dissertation is divided into six chapters containing several topics related to simulations and hardware implementation of neuromorphic computing using memristors as synapses. The main focus of this dissertation is to provide a comprehensive study of hardware implementation of ANN-based machine learning algorithms using novel 2D materials. Chapter 2 presents an extensive study of linear and logistic regression algorithms implemented with 1T1R memristor crossbars arrays. Using a sophisticated simulation platform that wraps circuit-level simulations of 1T1R crossbars and physics-based models of RRAM (memristors), we elucidate the impact of device variability on algorithm accuracy, convergence rate and precision. Moreover, a smart pulsing strategy is proposed for practical implementation of synaptic weight updates that can accelerate training in real crossbar architectures. In chapter 3, we report on the hardware implementation of analog dot-product operation on arrays of 2D hexagonal boron nitride (h-BN) memristors. This extends beyond previous work that studied isolated device characteristics towards the application of analog neural network accelerators based on 2D memristor arrays. The

wafer-level fabrication of the memristor arrays is enabled by large-area transfer of CVD-grown few-layer (8 layers) h-BN films. The dot-product operation shows excellent linearity and repeatability, with low read energy consumption (~200 aJ to 20 fJ per operation), with minimal error and deviation over various measurement cycles. Moreover, we present the implementation of a stochastic logistic regression algorithm in 2D h-BN memristor hardware for the classification of noisy images. In chapter 4, we demonstrate the hardware implementation of a linear regression algorithm on h-BN memristor arrays. Chapter 5 investigates the electrical performance of 2D hexagonal Boron Nitride (h-BN) memristors towards their implementation in spiking neural networks (SNN). Based on experimental behavior of the h-BN memristors as artificial synapses, we simulate the implementation of unsupervised learning in spiking neural network (SNN) for image classification on the Modified National Institute of Standards and Technology (MNIST) dataset. Additionally, we propose a simple Spike-Timing-Dependent-Plasticity (STDP)-based dropout technique to enhance the recognition rate in h-BN memristor-based SNN. Finally, chapter 6 provides conclusions and summarizes the main contributions of this work.

CHAPTER 2

CIRCUIT-LEVEL IMPLEMENTATION OF REGRESSION ALGORITHMS USING METAL-OXIDE MEMRISTOR ARRAYS

2.1 Variability in Oxide-based RRAM

RRAM operation for NVM applications typically involves programming (and reading) cells into two distinct (binary) states, a low resistance state (LRS) or high resistance state (HRS). Multistate storage has also been demonstrated using RRAM for NVM (Patel et al.). Additionally, RRAM analog-based implementations of in-memory computing and neuromorphic architectures rely on the ability to program a continuous range of states (Yin et al.). The programming of different resistive states is achieved via the formation and rupture of conductive filaments inside the oxide/switching layer of the cell. RRAM is considered a great candidate for training and inference applications (Yu et al.), but the stochastic essence of conductive filament activity (Ielmini), introduces variability, programming abruptness, and non-linearity that may present significant challenge for the implementation of RRAM-based in-memory computing applications and machine-learning (ML). In (Zhao et al.), reliability concerns for RRAM were identified and metrics were discussed based on the impact on distinguishability of states and computing accuracy. As presented in (Zhao et al.), the basic reliability metrics relate to endurance, retention, noise, and write/read disturbs. Other "functional" reliability metrics include non-linearity, variability, dynamic range, precision, variation, asymmetry, etc. These reliability metrics refer to functional properties of RRAM that can have a severe impact on computing accuracy when degraded. In this chapter, we focus on another

functional reliability concern, i.e., variability in RRAM characteristics, and its impact on neural network training (convergence rate, accuracy, precision) based on gradient descent algorithms. In 2011, Chen et al presented a collection of results on the variability of LRS and HRS in different RRAM and CBRAM technologies (Chen and Lin). A similar collection of LRS and HRS variability from recent RRAM and CBRAM published results is shown in Figure 2.1 (Fey; Mahadevaiah et al.; Milo et al.; Hong et al.; Belmonte et al.; Guy et al.; Radhakrishnan et al.; Goux et al.). These indicate that large variation is prevalent for newer generations of RRAM devices as expected due to the stochastic nature of the resistive switching mechanisms. Thus, it is crucial to study the impact of device variability on in-memory computing circuits to gain insight on the viability of RRAM implementations. This chapter analyzes the effects of RRAM device variability on accuracy and precision of gradient descent-based ML algorithms (linear and logistic regression) using crossbar architectures. The algorithm-level analysis presented in this work uses Spice (Synopsys HSpice) circuit-level simulations that incorporate a compact memristor model previously developed and verified with experimental data (Chen and Yu). The primary goal of this work is to investigate the impact of device variability on the performance of gradient-descent-based machine learning algorithms. Therefore, device-to-device and cycle-to-cycle variations are introduced into key model parameters. The approach involves randomly sampling the model parameters from an experimentally verified distribution. For the ML algorithm analysis, a modified gradient-descent approach is used to train the crossbar array, similar to that presented in previous work by Nair et al (Nair and Dudek). In that previous work, a single programming voltage pulse of fixed amplitude and width is used to adjust the memristor conductance (i.e., the synaptic weights)

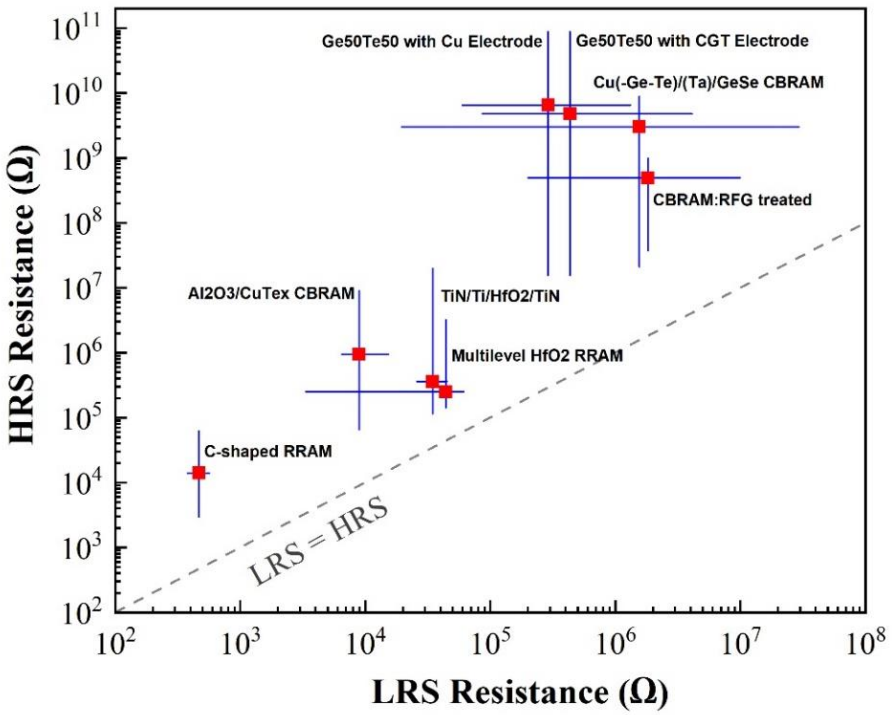


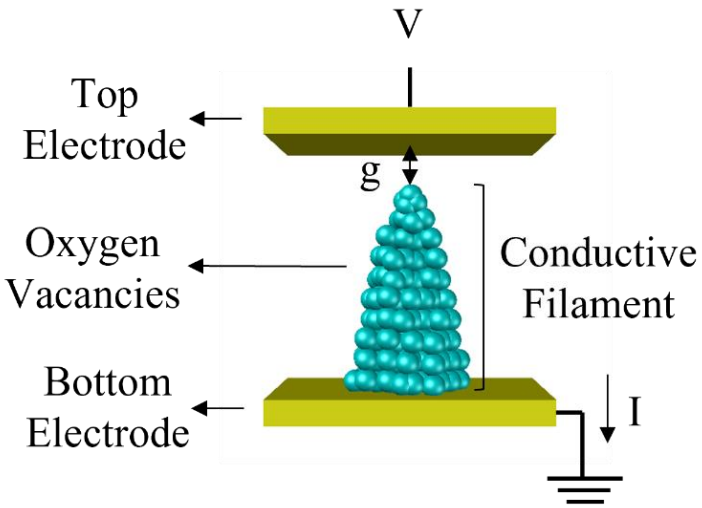
Figure 2.1. Variability of resistive-switching characteristics in recent metal-oxide RRAM and CBRAM technologies discussed in previous research. Intersection for each device represents the mean value of HRS and LRS.

independent of the magnitude of the required update. The polarity of the pulse (positive vs. negative amplitude) is selected based on the sign of the update as determined by the algorithm. In this work, we extend the approach by allowing a discrete number of programming pulses to update memristors in accordance with the necessary update. Based on this new approach, we study the convergence rate and performance of gradient-descent ML algorithms in the presence of large variation in memristor devices. The results of our ML algorithm analysis provide insight on convergence rate, accuracy, and precision of pattern classification experiments on RRAM crossbars and the effects of memristor variability. The chapter is structured as follows: Section 2.2 identifies and analyzes the effects of variability on the resistive-switching characteristic of 1-transistor-1-resistor

(1T1R) RRAM cells and describes the simulation approach for crossbar arrays. Section 2.3 and 2.4 present the implementation of linear and logistic regression on memristor crossbars and establishes the impact of device variability on algorithm performance. Mainly, despite large RRAM cell variability, the crossbar implementation of regression algorithms achieves convergence (as indicated by clear improvements in accuracy with training), but with noticeable degradation on precision (fluctuation in the accuracy of trained arrays).

2.2 Effects of Variability on Resistive-Switching Characteristic of 1-Transistor-1-Resistor (1T1R) RRAM Cells

In this work, we use a compact model for HfO_x-based RRAM devices (Chen and Yu). The bipolar switching characteristics achieved in the model are based on fundamental physics related to filamentary operation and have been experimentally verified with HfO_x



$$\frac{dg}{dt} = f(V)$$

$$I = f(g, V)$$

Figure 2.2. Filamentary operation and top-level mathematical representation of the physics-based RRAM model used in this work.

devices (Yang Yin Chen et al.; Y. Y. Chen et al.; Fantini, Goux, Degraeve, D.J. Wouters, et al.). A key parameter in the model that captures the internal state of the RRAM cell is the gap (g), specified as the distance between the top electrode and conductive filament as illustrated in Figure 2.2. The memristor conductance is directly related to this parameter. The dynamic process of resistive switching and current flow are modeled by the two general memristor equations shown in Figure 2.2. To model RRAM variation, the model fitting parameters I_0, v_0 and γ_0 (related to filamentary formation/dissolution and conductance) are allowed a dispersion 3σ/μ of 30%, 10% and 10%, respectively. These values were extracted to fit experimental HRS and LRS distributions in TiN/Hf/HfOx/TiN-based RRAM devices (cf. Figure 5 in (Chen and Yu)). As described in (Chen and Yu), dispersion in all three parameters should be included to account for the actual (experimental) variability in RRAM characteristics and measured distribution in LRS an HRS. The RRAM model is implemented in Verilog-A and circuit-level simulations are conducted using Synopsys HSpice. For simulating 1T1R cells we use a 65 nm n-type CMOS transistor model based on the Predictive Technology Model (PTM) from Arizona State University (*Arizona State University, Predictive Technology Model (Ptm).*). Figure 2.3a shows the schematic of the 1T1R cell, indicating the pulsing approach to increase or decrease the conductance of the memristor (i.e., set/reset the memristor). The n-MOS transistor acts as a selector device and the gate voltage is used to modulate or limit the amount of current that flows through the cell. The 1T1R configuration helps eliminate sneak path currents and improves analog programmability by reducing abrupt changes in

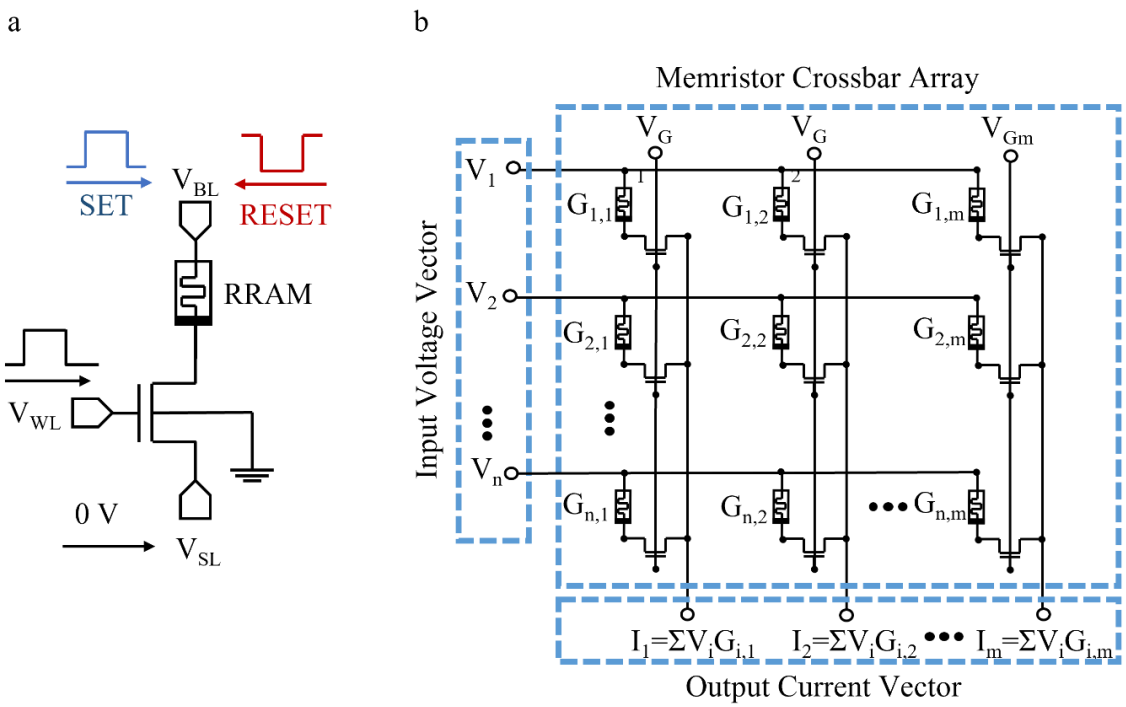


Figure 2.3. (a), (b) Schematic of the 1T1R RRAM cell and crossbar array simulated with Synopsys HSpice.

conductance from set/reset pulses (Chen and Yu). Finally, Figure 2.3b is a schematic of the RRAM 1T1R crossbar arrays simulated in this work.

The impact of RRAM variability on the 1T1R cell resistive switching properties is summarized in Figure 2.4. Figures 2.4b-d show the effects of model parameter dispersion individually (I_0, v_0 and γ_0), and Figure 2.4a shows the combined effects on the resistive switching current-voltage (I-V) characteristics.

Figure 2.5a plots the conductance-voltage (G-V) characteristics including dispersion in all three model parameters. Figure 2.5b reveals the impact of variability on the pulsed characteristics (change in conductance with consecutive pulses). In Figure 2.5b, 100 cycles are shown, each cycle consisting of 100 positive and 100 negative consecutive pulses. A different visualization for the impact of variability on the resistive switching

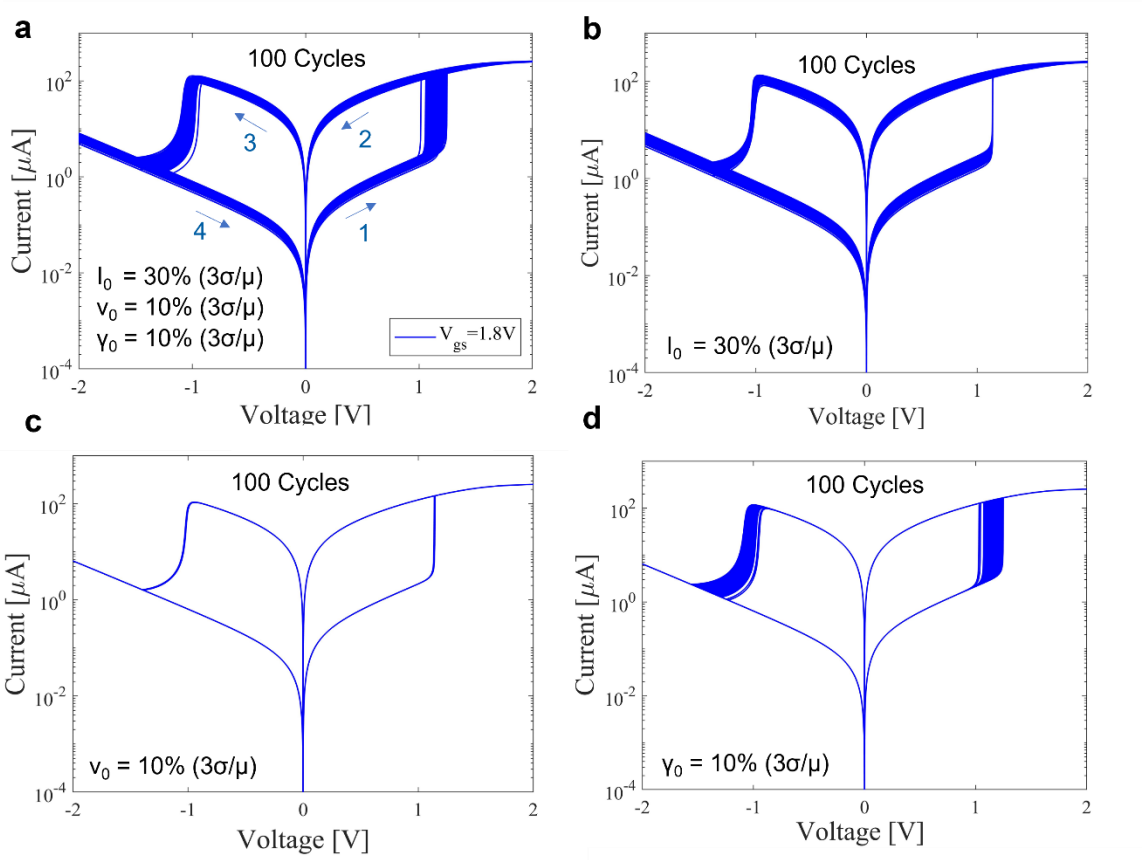


Figure 2.4. (a) I-V characteristics of 1T1R RRAM cell considering joint effects of dispersion in model parameters, (b-d) simulation of DC resistive-switching I-V characteristics of 1T1R RRAM cell considering individual effects of dispersion in model.

properties is provided in Figure 2.5c. This plot shows contours for the cumulative distribution function (CDF) of change in conductance (ΔG) as a function of conductance (G). It provides a graphical representation of the non-linear and abrupt response of ΔG resulting from the programming pulses (only shown for positive pulses). At low levels of G (starting with a weak filament), the CDF shows that most pulses will result in large changes in conductance (abrupt). As G increases, the distribution shifts to smaller changes in conductance (less abrupt) and distribution is narrower (less variation for ΔG).

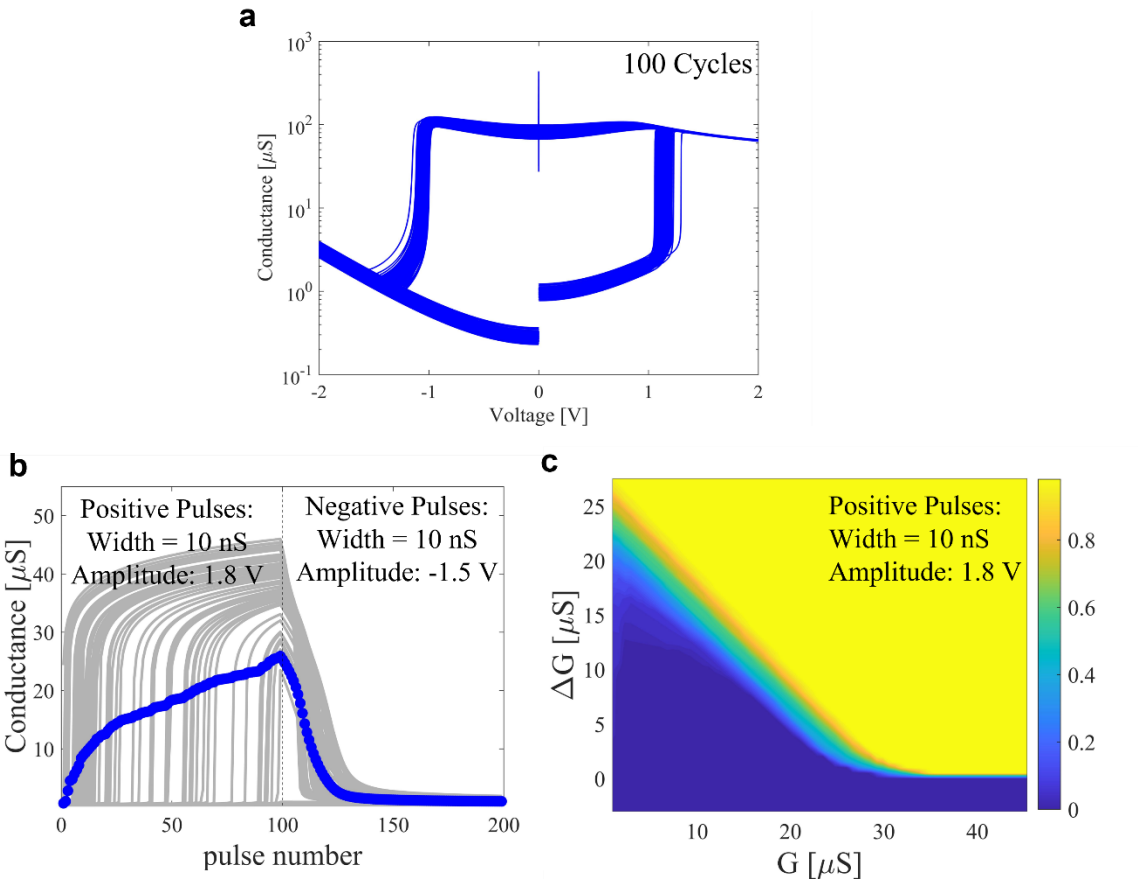


Figure 2.5. (a) G-V characteristics, (b) pulse-programming of memristor conductance (multiple cycles and average), average is solid blue line with circles, (c) contour plot of the CDF for change in conductance vs. conductance.

The implementation and analysis of regression algorithms presented in this work uses MATLAB scripts that organize and execute Synopsys HSpice circuit-level simulations of RRAM 1T1R crossbars. In this simulation platform, the initialization of RRAM devices as well as the functions of the peripheral circuits (e.g., normalization of inputs and outputs, calculations of prediction/classification error, activation functions, etc.) are conducted in MATLAB software. However, crossbar functions including vector matrix multiplications (VMM) and pulsed programming of RRAM 1T1R cells are directly implemented with HSpice circuit simulations using the described compact model. A

detailed explanation of the work is provided in the supplementary material of (Afshari et al.).

Smart Pulsing Strategy for Weight Updates

This work presents a new weight update strategy for accelerated training in ML algorithms. The proposed strategy selects the number of programming pulses for each memristor at each training step not only based on the sign of the required update, but also on its magnitude. For practicality, the number of pulses is discretized to three different ranges of required weight update (see Figure 2.6). For example, a large conductance update requirement leads to more consecutive pulses compared to a smaller update requirement. This leads to larger weight (conductance) changes during early training steps, and smaller changes in later steps to help fine tune and maximize accuracy as the training advances. We note that this technique does not affect the frequency of updates, as an update is still done at every training step. In the next section, we demonstrate how this strategy results in higher convergence rate, as well as improved precision and accuracy for the crossbar implementation of multi-variable linear and logistic regression algorithms compared to existing techniques based on fixed update pulsing methods.

Discussion About Peripheral Circuits

Pulse updates can be generated by a simple CML (current-mode logic) driver circuit where the circuit is tuned to ensure enough drive voltage capability for loads presented in terms of crossbar size (crossbar interconnect resistance and memristors). The write voltages should be verified to have enough margin above the memristor write threshold to

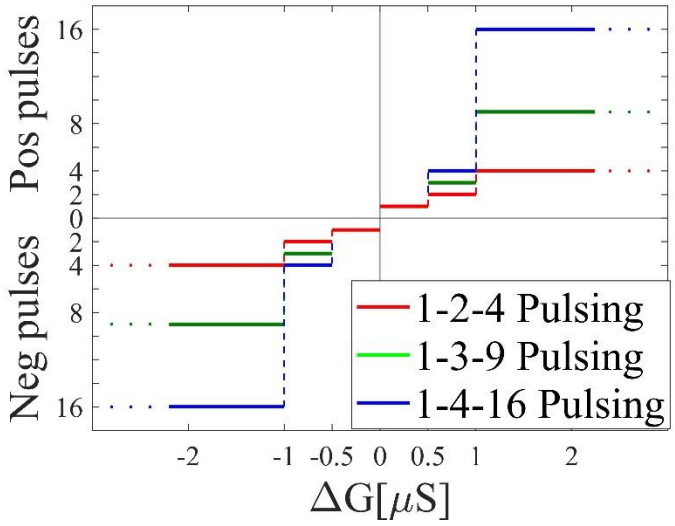


Figure 2.6. Translation of ΔG into number of positive or negative voltage pulses for realistic hardware implementation of the gradient-descent.

effectively drive the furthest memristor in the write path. More specifically, for our proposed smart pulse update strategy, a configurable ring oscillator can be used to ensure a specified number of similar spaced pulses as discussed in (Seo et al.). Read currents are accumulated at the end of crossbar and need to be sensed prior to digital conversion and further processing. The choices of voltage versus current mode sensing circuits are described in (Musisi-Nkambwe et al.). In this solution, a current mode sensing mechanism is preferred where a reference current is generated to compare against the accumulated output current. This choice, while more area intensive, allows for trackability of device variation mirrored in the reference crossbar array. A detailed scheme of the current-mode sensing circuit is described in (Chang et al.).

2.3 Implementation of Linear Regression on Memristor Crossbars

This section presents the implementation of stochastic multivariable linear regression on a 3×1 1T1R RRAM crossbar array. This is a type of regression algorithm

with multiple independent variables ($x_0, x_1, \ldots x_n$) combined into a linear prediction function of the dependent variable (y). The term stochastic comes from the stochastic gradient descent optimization approach where a single sample or subset of the data is randomly selected to update the model parameters during each training step. In practice, we present one data sample at a time to our crossbar array. The model prediction (h) is given by the dot product of the input variables ($x_0, x_1, \ldots x_n$) and the model parameters which are stored as the memristor conductances ($G_0, G_1, \ldots G_n$). Mathematically, the prediction h is given by:

$$h = x^T G \,, x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}, G = \begin{bmatrix} G_0 \\ G_1 \\ G_2 \end{bmatrix}. \qquad [1]$$

Here, x is the normalized 3×1 input vector and G is the 3×1 vector of the memristor conductances. Figure 2.7a is a flowchart for the crossbar implementation of stochastic multivariable linear regression. Figure 2.7b shows the schematic of the 3×1 1T1R RRAM crossbar array as implemented in the simulation. The smart pulsing strategy used in this demonstration is illustrated in Figure 2.6. This discretized approach would be a practical implementation of gradient-descent on a real memristor crossbar. When training, the initial steps will typically require larger updates in conductance (ΔG) because the error (δ) is initially large, prompting a larger number of pulses. As the training advances and the error is reduced, the required update is also reduced leading to a smaller number of applied pulses. In Figure 2.6, three different versions of the pulsing strategy are shown. These correspond to different sets of programming pulses (positive or negative) used to update conductance based on the value of ΔG_i. For example, in the pulsing strategy labeled 1-2-4,

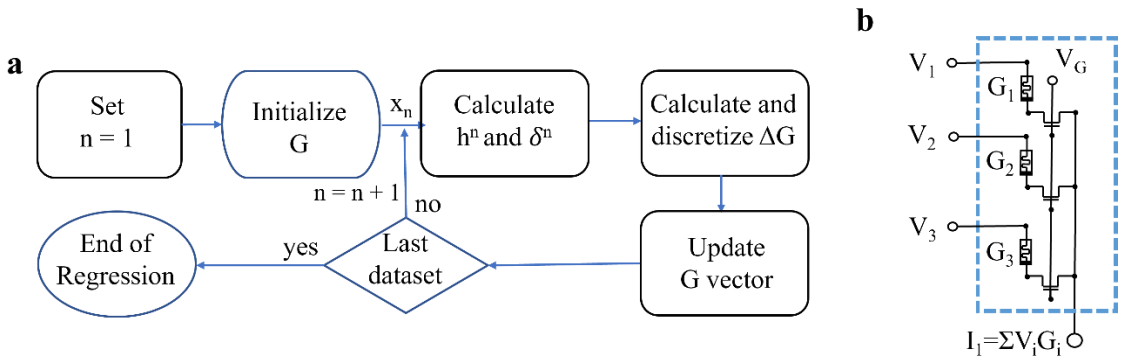


Figure 2.7. (a) Flow chart of implementation of linear regression algorithm on a memristor crossbar, (b) schematic of the 3×1 1T1R crossbar array implemented in Synopsys HSpice.

one, two or four programming pulses are applied depending on whether the required update in conductance is between 0 and 0.5 µS, 0.5 and 1 µS, or above 1 µS. Our demonstration of multivariable linear regression is based on an artificial data set for the price of a pizza as a function of two independent variables, x_1 and x_2, where x_1 represents the number of ingredients, and x_2 represents the size of the pizza. Note that the same approach can be easily extended to N independent variables on an $(N + 1)$ crossbar array. In the hardware implementation, the input variables are presented as voltage signals ($x_i \rightarrow v_i$) on each row of the crossbar (see Figure 2.7b), and the prediction is represented by the current flowing on the crossbar array as given by Kirchhoff's law: $h \rightarrow I = \sum v_i G_i$. To ensure the accuracy of the prediction in this hardware implementation, the amplitude of the input voltage signals is normalized to a range between 0 and 0.25 V. This range results in good linearity (i.e., current is directly proportional to voltage, or equivalently conductance is independent of voltage) as shown in Figure 2.5c. In the optimization process that occurs during training, a cost function J proportional to the mean square error is minimized through the update of the conductance values. The error is determined by the difference in the predicted and actual values as $\delta^n = h^n - y^n$, where the superscript indicates the n th data sample (also

n th training step). At each training step, each device requires a conductance update given by $\Delta G_i = -\alpha \delta^n v_i^n$. Here, α is a learning rate. In practice, it is not feasible (or required) to perfectly update the conductances by exactly ΔG_i. The goal is to minimize the error (or cost function). Therefore, the approach is to use a discrete number of programming pulses (positive or negative) to approximate the change in conductance state of each memristor according to the value of ΔG_i. This approach is referred to as the smart pulsing strategy.

In our demonstration, a dataset of size 1000 is artificially generated to be used as training of the crossbar array network. The conductance values are randomly initialized within a range from 10 to 60 μS. The learning rate α is initially set to 1, and for improved convergence is reduced by 3% after each training step. Each iteration corresponds to presenting a single sample from the dataset followed by the adjustment of the conductance for each memristor based on the calculated ΔG_i. Figure 2.8 summarizes the results of the memristor crossbar implementation of the stochastic multivariable linear regression algorithm (without variation). In Figure 2.8a, the blue dots are the dataset corresponding to price of pizza plotted as a function of two independent variables, x_1, number of ingredients, and x_2, size of the pizza. The algorithm is conducted five different times and for each case the initial and final conductance states are recorded. The red mesh surfaces represent the model prediction based on the initial (random) state of memristor conductances in the crossbar. The green mesh surfaces represent the prediction after 1000 training steps (i.e., after all data samples have been presented to the array). The different final predictions for each case result from the different random initial states along with random shuffling in the sampling process. The results show a significant improvement in the model prediction of the data set after training as indicated by the green mesh surfaces

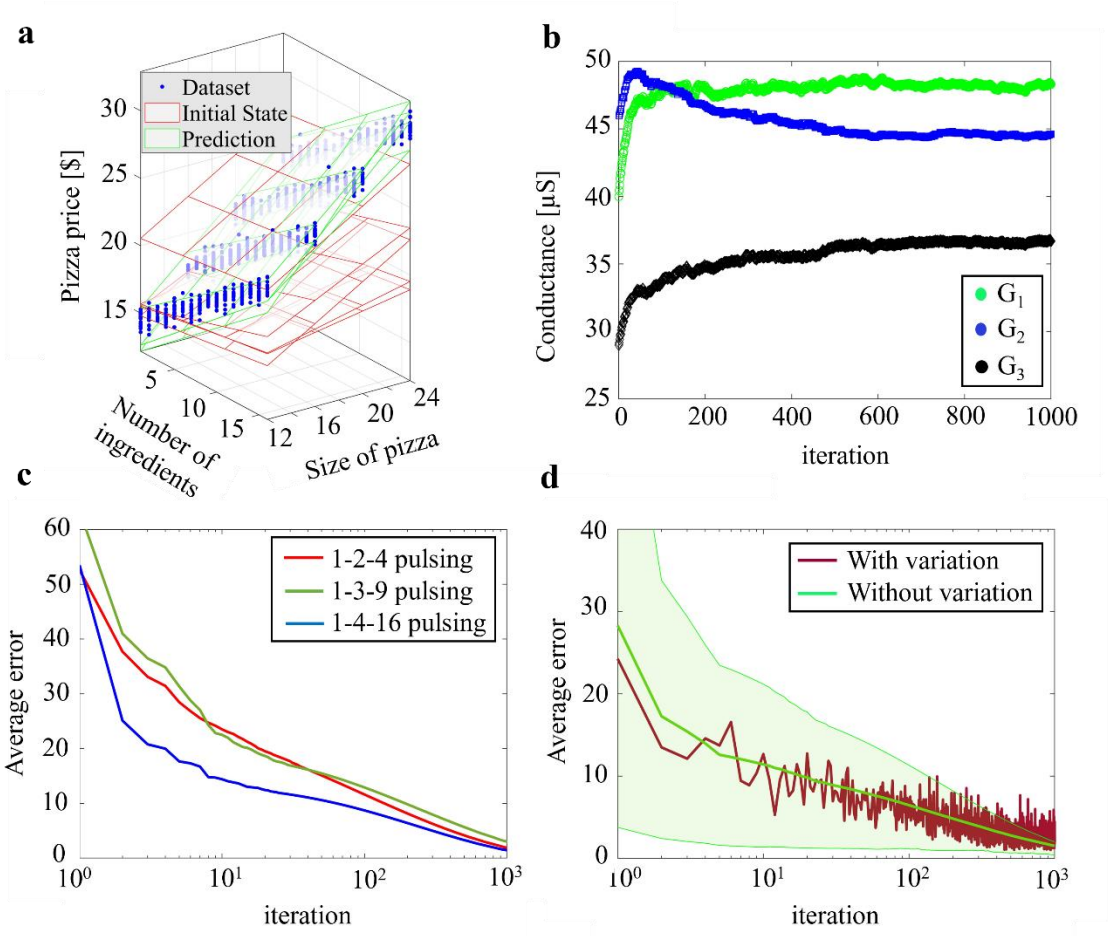


Figure 2.8. (a) Results of linear regression algorithm for 5 simulations, (b) conductance evolution for each memristor vs. iteration, (c) comparison of convergence rate for three different cases, (d) convergence rate (with and without variation).

overlapping the data points. Figure 2.8b plots the evolution of conductance for each memristor in the array as a function of the algorithm iteration step during training. It indicates larger updates in conductance during the initial steps and a settling as convergence is achieved. Figure 2.8c compares the convergence as indicated by the prediction mean squared error (MSE) as a function of iteration number for the three different versions of the pulsing strategy. It is clear from the slope of MSE vs. iteration number that the pulsing strategy with larger number of pulses (i.e., 1-4-16) has a faster initial convergence rate (can reach lower MSE with fewer iterations during the initial

training steps). However, as training advances, the convergence rate slows down and eventually all three pulsing strategies achieve small MSE. We note that a fast initial convergence rate may be desirable for specific training applications. The proposed pulsing approach can achieve a fast initial convergence rate without compromising the high prediction accuracy of the fully trained crossbar array. Finally, we examine the impact of variability on the stochastic multivariable linear regression algorithm memristor crossbar implementation. Figure 2.8d plots the prediction mean squared error (MSE) as a function of iteration number for a pulsing strategy of 1-4-16, with and without memristor variation. For the case of no variation (shown in green), we include the average MSE vs. iteration from 10 simulations (solid line) and the range between maximum and minimum MSE (shaded green region). For the case with variation, we only show the average MSE vs. iteration number (solid red line). While the convergence is still good even with memristor variability, we note the following effects: 1) The results indicate that the convergence is slower (error is reduced at a slower rate with training), 2) The accuracy is degraded (average error after training is slightly larger than what was obtained when neglecting variation), 3) The most significant issue appears to be an impact on precision. The results in Figure 2.8d show noticeable fluctuation in average error when variation is included. We interpret these fluctuations as an impact on the algorithm precision resulting from variability in the programming of memristor conductance states. It should be noted that even with these detrimental effects of memristor variability, the prediction error is still converging (i.e., error reduces with training) to about 3-5%. This is a promising result for memristor crossbars implementations of regression algorithms that appears to indicate some level of immunity to device variability at the algorithm-level. The same simulation

was repeated to compare the individual effects of dispersion in model parameters I_0, and γ_0 on the algorithm performance (not shown). We discover that the observed impact on precision is due mainly to dispersion in I_0, correlating with variation in conductance, and not to dispersion in γ_0 which mostly correlates to dispersion in set/reset voltages (see Figure 2.4a-d). This observation is reasonable as the algorithm implementation is based on pulsed programming where the amplitudes of the applied voltage pulses (+1.8 V/-1.5 V) have sufficient margins above/below the set/reset thresholds.

2.4 Implementation of Logistic Regression on Memristor Crossbars

This section describes the implementation of stochastic logistic regression in a memristor crossbar for classification of 5×5-pixel binary images that represent characters 'S', 'M', 'R', and 'T'. Figure 2.9a is a flowchart describing the logistic regression implementation. The data set is artificially generated and includes "noisy" samples or images where two of the binary pixels have been flipped (see Figure 2.9b). Separate data were generated for training and to test the classification accuracy at fixed training intervals (i.e., after a fixed amount of training images have been presented to the network). Figure 2.9c is a graphical representation of the neural network that is being implemented by the memristor crossbar for this classification task. The crossbar schematic is shown in Figure 2.9d. Here, each synaptic connection is implemented by a memristor differential pair. The effective conductance for each differential pair is given by: $G_{ij} = G_{ij}^{+} - G_{ij}^{-}$. This enables negative weights to be implemented with the crossbar array (all conductances are positive). To perform the classification of the 5×5 images, a 25×8 memristor crossbar is simulated.

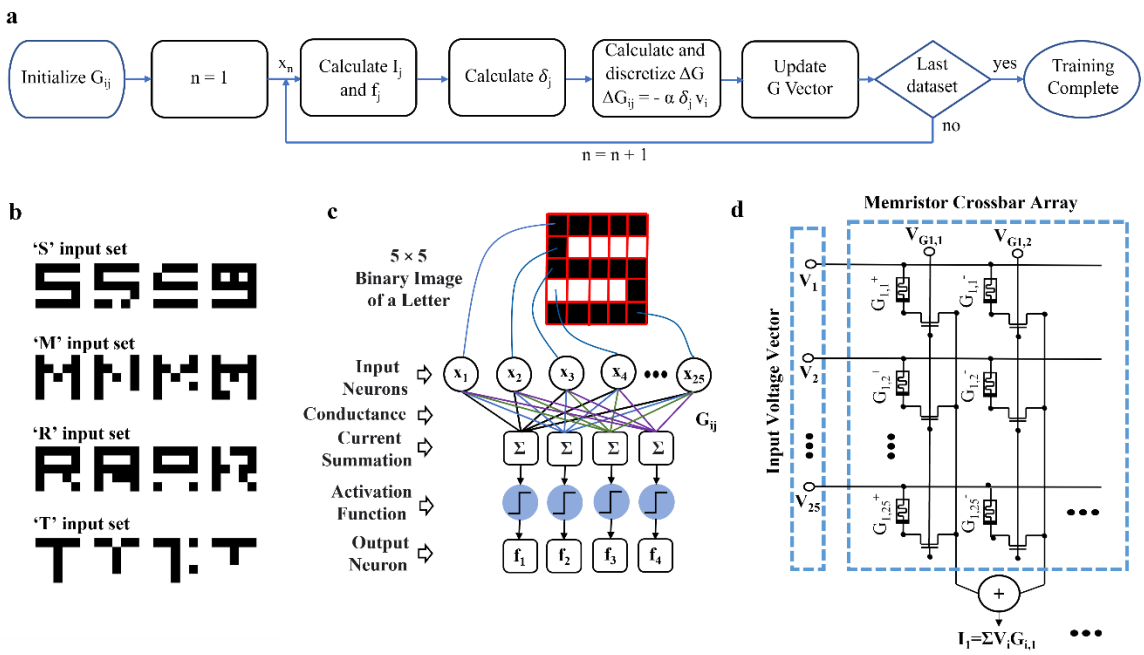


Figure 2.9. Image classification experiment: (a) flow chart for logistic regression algorithm, (b) input binary images, (c) representation of the neural network implementation for image classification, (d) partial schematic of the 25 × 8 memristor crossbar.

During training, images that correspond to different characters (S, M, R, or T) are randomly presented to the array, so all 4 neurons are simultaneously trained to recognize their assigned character. As discussed in the previous section, the linear range for the I-V characteristics of the memristors falls between the range of -0.25 to +0.25 V. Thus, during the "read" operation, each pixel from the binary image is mapped to a crossbar input voltage signal of 0.1 V for white pixels and -0.1 V for black pixels. The output current on each neuron is essentially a dot product of the input voltage vector and the effective conductance vector from the corresponding column pair. Mathematically, the output currents are given by $I_j = \sum_{i=1}^{25} v_i G_{ij}$ where G_{ij} are the adjustable effective conductance and vi are the input voltages. The output current is normalized and then goes through the

sigmoid activation function which returns the value of $f_j = \frac{1}{1+e^{-I'_j}}$. Here, I'_j is the normalized output current of each column. In this normalization, the original current (I_j) is simply divided by a constant factor and presented to logistic function as I'_j . The sigmoid function gives an output ranging between 0 and 1. In this implementation, the classification error (δ_j) is calculated for each neuron as: $\delta_j = f_j - y_j$, where y_j is determined by the label in the training data set (equals 1 for the neuron that corresponds to the training image and zero for other neurons).

Similar to the linear regression demonstration, a smart pulsing strategy is used where different number of pulses are applied at each iteration based on the required conductance update given by $\Delta G_{ij} = -\alpha \delta_j v_i$. For ΔG_{ij} greater than ±0.01 μS, five positive/negative pulses are applied, for ΔG_{ij} between ±0.005 and ±0.01, two positive/negative pulses are applied and for ΔG_{ij} smaller than ±0.005 μS, a single pulse is applied. This pulsing strategy is illustrated in Figure 2.10a. In this demonstration, the programming pulses have amplitudes of +1.4 V and –1.35 V, and widths of 20 ns and 10 ns respectively, and the learning rate, α, is constant with the value of 0.5. A single image from the dataset is presented to the network during each training step, followed by an adjustment of the effective conductance through the application of consecutive voltage pulses determined based on ΔG_{ij}. For example, if the effective conductance (ΔG_{ij}) needs to be increased, positive pulses are applied to the positive memristor (G_{ij}^{+}) and negative

pulses are applied to the negative memristor (G_{ij}^{-}) in the differential pair. This increases the

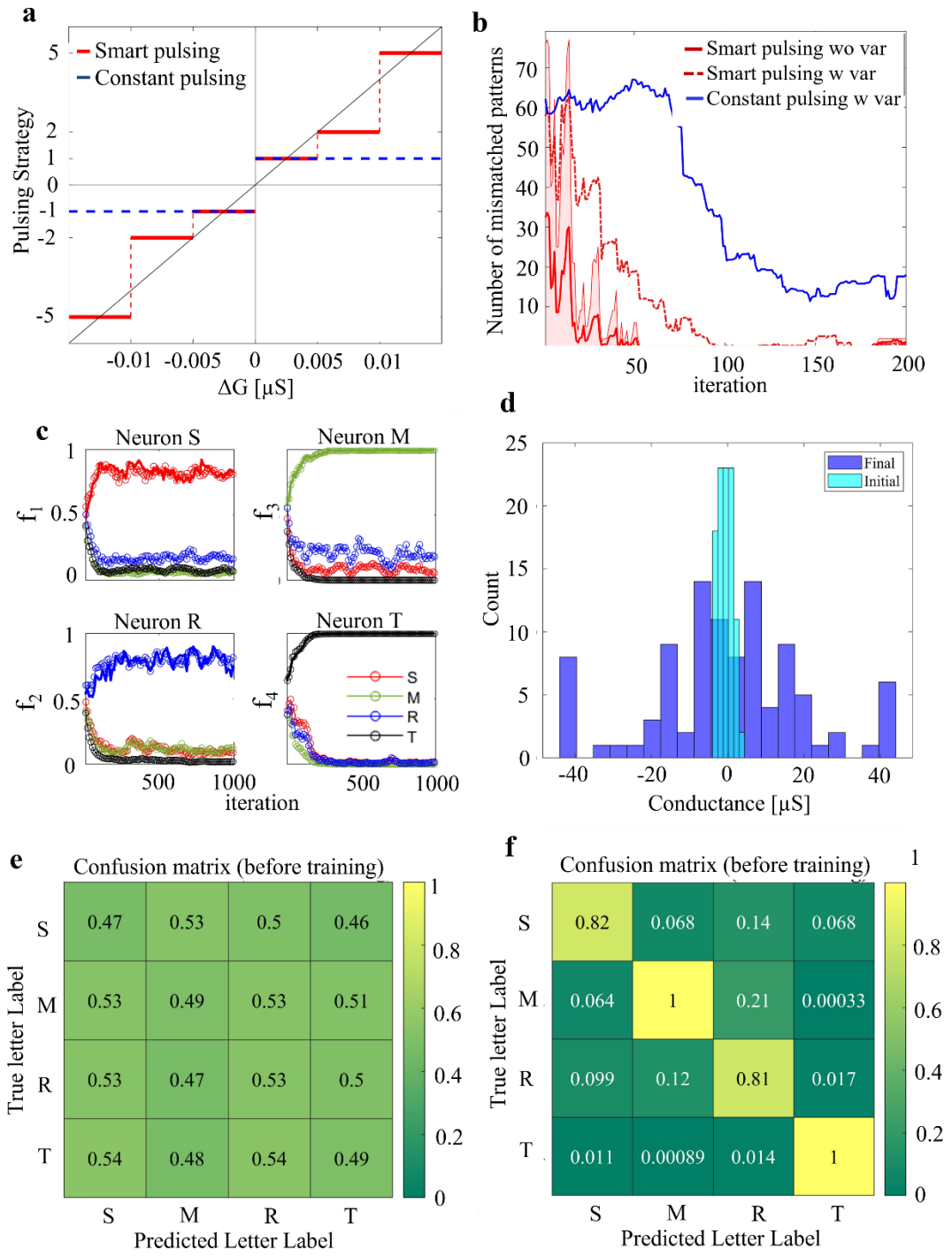


Figure 2.10. Results for logistic regression: (a) pulsing strategy, (b) mismatched patterns for "S", (c) evolution of convergence for output neurons, (d) histogram for the distribution, (e),(f) confusion matrix before and after training.

effective conductance. Similarly, if the effective conductance needs to be decreased, negative pulses are applied to the positive memristor and positive pulses are applied to the negative memristor in the differential pair. The accuracy of the prediction is evaluated at fixed training intervals using a separate dataset that consists of 400 5x5 binary images (100 noisy images for each character). Figure 2.10 summarizes the results of the classification algorithm. We first compare the smart pulsing strategy against the constant pulse update approach described in (Nair and Dudek) and implemented in a real crossbar in (Prezioso et al.). The constant pulsing approach is indicated by the dashed blue line in Figure 2.10a, where a single pulse is applied independent of the value of ΔG_{ij}. Figure 2.10b shows the number of mismatched patterns for character "S" in the evaluation set as a function of first 200 training steps. With increasing training steps, the percentage of mismatched patterns decreases. Red lines indicate the smart pulsing strategy proposed in this paper whereas blue shows the results for the constant pulsing method (Nair and Dudek; Prezioso et al.). The solid red line corresponds to the average mismatch from 5 different trials with different initial states and without variation. The shaded region indicates the range of maximum and minimum mismatch from all 5 individual runs. The red dotted line is the average mismatch with variation. As observed, it takes longer for the case with variation to reach almost perfect classification (zero mismatched patterns). The blue line is the average mismatch including memristor variability for the constant pulsing approach. Compared to the smart pulsing strategy, convergence rate and accuracy are reduced. For the smart pulsing strategy, Figure 2.10c shows the evolution of convergence based on the average output of the sigmoid function (f_j) at each of the neurons and for each of the different characters in the evaluation data set (100 noisy images for each character). For example, for the neuron

assigned to character ‘S’ (labeled “Neuron S” in Figure 2.10c), f_j converges to a value close to 1 for images corresponding to the character ‘S’ and to values close to 0 for others. In Figure 2.10c, the results shown with open circles are the average of 5 different trials (each up to 1000 training steps) without memristor variation. For comparison, solid lines plot the case with variation (only shown for results from images that match the assigned character to each neuron). These results indicate that memristor variation appears to have minimal impact on classification accuracy but affects precision by introducing more fluctuations as a function of training step (consistent with results from linear regression). Figures 2.10e and 2.10f show the confusion matrix corresponding to f_j values for each neuron before and after training. Before training, the f_j values for each neuron are randomly distributed around 0.5 based on the initial random effective conductances (see Figure 2.10d for distribution of initial and final effective conductance). The final values of f_j, after the training is complete (1000 steps), are shown in Figure 2.10f. The results are in accordance with Figure 2.10c, where corresponding neurons converge towards 1 and the non-corresponding neurons approach 0. Our results indicate that with memristor variability, which is the realistic case for actual physical crossbars, more iterations are necessary to converge to a desirable classification accuracy. For more complex patterns, this gap may be large. From Figure 2.10c, it can be concluded that because of the nature of logistic regression, where the output current (weighted sum of inputs) goes through the logistic function (in this case), the variation does not have outstanding impact in the learning process. It is important to point out that in some cases, small levels of device variation (noise) can help achieve improvements in accuracy as it may act as a form of regularization to prevent overfitting to the training set. This has been demonstrated in (Y.-

C. Chen et al) for MNIST datasets where small levels of variability improved accuracy but was ultimately degraded for larger levels of variation. Another technique to prevent overfitting and overreliance on individual devices is dropout regularization and is commonly used in multi-layer neural networks and was recently proposed to alleviate stuck-at-faults in memristor crossbar implementations (Xu et al.). Moreover, the work in (Lillicrap et al.) pointed out in the context of spiking neural networks that noise symmetrically distributed about a mean of zero will integrate out when trained across many samples. Another well-known source of noise that is neglected in the present analysis results from quantization of bit-line currents as typical implementations of the logistic function use digital circuits (Seo et al.).

Additionally, we compared the smart pulsing approach with the constant pulse updating strategy that was outlined in (Mbarek et al.). The dashed line in Figure 2.11a and b represents the single pulsing method, where a single pulse is applied regardless of the conductance update value. The smart pulsing approach suggested in this study is indicated by solid lines. Convergence rate and accuracy for a single pulsing method are lower than they are for a smart pulsing strategy.

CHAPTER 3

DOT-PRODUCT COMPUTATION AND LOGISTIC REGRESSION WITH 2D HEXAGONAL-BORON NITRIDE (H-BN) MEMRISTOR ARRAYS

3.1. H-BN Memristors

Since the discovery of graphene, two-dimensional (2D) materials have been the focus of intense research and have shown great potential to advance the capabilities of future integrated electronic systems. Recent studies have proposed the possibility of adding new functionality through the hybrid integration of 2D materials with complementary metal oxide semiconductor (CMOS) technologies (Lemme et al.; Zhu, Wen, et al.). Here, neuromorphic computing is recognized as one of the main applications of next-generation electronic systems enabled by 2D materials integration (Lemme et al.). This unconventional computing paradigm aims at the implementation of artificial neural networks using compute-in-memory hardware to achieve energy-efficient data processing for machine learning and artificial intelligence (AI) applications. It requires devices that can emulate bio-inspired functions (e.g., artificial synapses and neurons) and memristors have emerged as a primary choice (Yibo Li et al.). Memristors are electronic devices with variable resistance states that depend on their past and recent experience with external stimuli. Conventional memristor technologies are constructed from bulk materials and their resistive switching behavior can be achieved through various mechanisms (e.g., ionic transport, filamentary, phase change, charge trapping, etc.). Filamentary metal-oxide resistive random-access memory (i.e., RRAM) is a widely studied technology due to its nonvolatility, high switching speed, low switching energy, and small energy footprint.

Recently, several studies have reported the non-volatile resistive switching (NVRS) behavior of two-dimensional (2D) materials down to the single atomic layers (Wu et al.; J. Ge et al.). A variety of 2D materials were shown to exhibit NVRS properties including transition metal dichalcogenides (TMD) (R. Ge et al.), hexagonal boron nitride (h-BN) (Nikam et al.; S. Chen et al.,), black phosphorus (Rehman et al.), graphene (Pradhan et al.), etc. CVD-grown h-BN has attracted significant interest due to its compatibility with high-density wafer-scale integration (S. Chen et al.). In CVD-grown h-BN memristors, the NVRS behavior is attributed to the formation and dissolution of conductive nanofilaments that result from the penetration and removal of metal ions (from an adjacent electrode) into defects at grain boundaries in the h-BN film (Kumar et al.).

Two-dimensional h-BN memristors have demonstrated superior properties compared to their bulk counterparts (e.g., metal-oxide memristors), making them ideal candidates for future neuromorphic chips for artificial intelligence applications. For example, they can extend the vertical scaling limit of oxide-based RRAM as the NVRS behavior endures even in atomically thin h-BN monolayers (Wu et al.; Kumar et al.). Moreover, the layered structure of h-BN may help alleviate programming errors and variability (e.g., stuck-at issues) associated with non-uniformity in the thickness of the resistive switching medium in bulk technologies (Chaudhuri and Chakrabarty). Additionally, 2D h-BN memristors were shown to provide better analog control of conductance programmability (e.g., long-term potentiation/depression of artificial synapses) over a wide range of operating currents when compared to metal-oxide RRAM where programmability is limited to high currents. This is attributed to filament formation happening in native defects surrounded by stable crystalline 2D layered h-BN (Kumar et

al.). In fact, the superior chemical stability of h-BN memristors is expected to also alleviate oxidation reaction to filaments and prevent the redundant formation of undesired paths, thus helping improve endurance, which has been an issue with oxide-based RRAM.

Despite their great potential for neuromorphic hardware, few reports of dot-product computation using 2D memristors have appeared in the literature, even though it is crucial for most analog-based implementations of neural network accelerators. Previous work reported on dot-product computation in h-BN memristors (two devices in parallel) and its application towards hardware implementation of linear regression algorithms (Xie et al.). In this chapter, we report a more extensive dot-product computation with larger arrays based on a wafer-scale process for 2D h-BN memristors. In addition to dot-product, our analysis elucidates the NVRS characteristics of wafer-scale CVD-grown h-BN memristors, including on/off ratio, low-voltage operation, endurance, retention, pulsed analog programmability. Figure 3.1 presents a comparative analysis of the set switching energy versus set switching time (set delay) of the fabricated device in our study, with similar

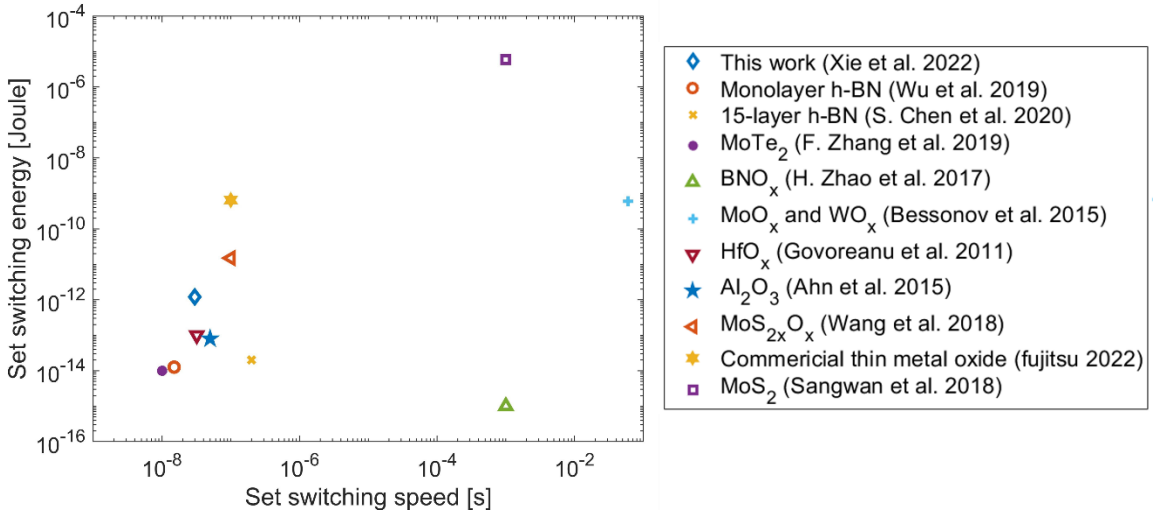


Figure 3.1. Comparison of energy consumption vs. switching time in our CVD-grown h-BN device compared to counterparts and conventional oxide-based memristors.

resistive switching materials and conventional oxide-based technology as reported in the relevant literature. The ability to store information is a crucial factor in integrated circuits that operate at high clock speeds, with switching time being a critical parameter in this regard. The energy used during the training process is directly affected by the set/reset switching energy, making it necessary to strive for lower switching energy levels (~1 pJ). By reducing the ventilation requirements and enabling the use of wearable and self-powered technologies, smaller switching energies can significantly enhance the practicality of these devices (Lanza et al.). Our device shows promising results in terms of power/energy efficiency. We examine the dot-product computation with respect to its accuracy, variability, and energy efficiency. This analysis, the first of its kind for 2D memristor technology, represents significant progress towards the practical implementation of neuromorphic hardware using 2D materials. Finally, we demonstrate the implementation of a logistic regression learning algorithm to classify noisy images using our 2D h-BN memristor hardware with near-ideal performance and accuracy (by comparisons with simulations).

3.2 Fabrication, Physical Characteristics and Electrical Behavior of H-BN Memristor Devices

Arrays of 2D memristors with a metal-insulator-metal (MIM) structure are fabricated on Si/SiO_2 wafers using CVD-grown few-layer h-BN films. A photograph of a

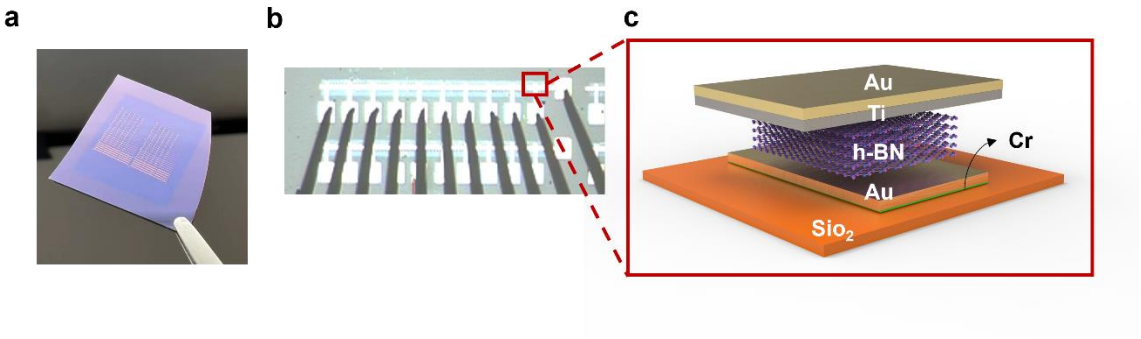


Figure 3.2. (a) Photograph of a typical 2D h-BN memristor array wafer, (b) micrograph of h-BN memristor arrays under test, (c) cross-sectional schematic of the few-layer h-BN memristor arrays with Ti/Au top electrode and Au bottom electrodes.

typical Si/SiO_2 wafer with the h-BN memristor arrays is shown in Figure 3.2a. This work reports on devices with Au bottom electrodes, and Ti top electrodes (capped with Au). A micrograph of Au/h-BN/Ti memristor arrays under test is provided in Figure 3.2b, and details of a single device cross section are depicted in Figure 3.2c. Each array shares a common bottom electrode (BE) while the top electrodes (TE) are distinct. The fabrication steps are illustrated in Figure 3.3a and include the patterning and deposition/lift-off of the shared bottom electrodes (steps *i*, *ii*, *iii*), followed by transfer of the few-layer CVD-grown h-BN film (~5 nm in thickness) and patterning of the active regions by dry-etching (steps *iv*, *v*, *vi*). Finally, the top electrodes are prepared by photolithography, e-beam evaporation, and lift-off (steps *vii*, *viii*, *ix*). See section 3.5 for more details on the fabrication process.

Figure 3.3b is a micrograph of two different fully fabricated h-BN memristor arrays (1×3 and 1×10). We note that each fabricated sample contains over 200 h-BN memristor arrays, and the majority of devices demonstrate reasonable resistive-switching behavior yielding >90% working devices. This is consistent with previous work that used similar methods for wafer-scale integration and processing of 2D memristive crossbars (reported 98% yield) (S. Chen et al.). A critical step in the fabrication of the arrays to achieve good resistive-switching behavior and high yield is the transfer of the CVD-grown h-BN film. Thus, to verify the quality of the h-BN film, we conducted Raman spectroscopy at various randomly selected locations immediately after the transfer step. Figure 3.3c shows Raman spectra revealing peak positions at 1370 cm^{-1} for all different locations, consistent with previously published results on few-layer h-BN films (Basu et al.). Additional verification of the h-

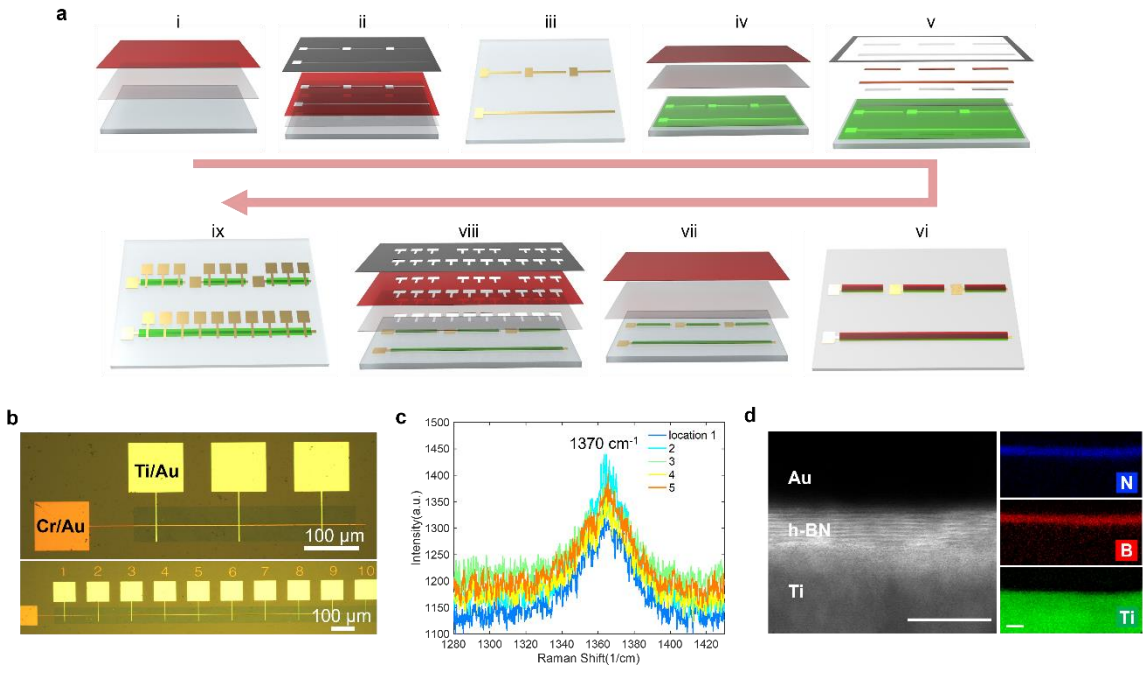


Figure 3.3. (a) CVD-grown fabrication steps for h-BN memristor arrays on Si/SiO_2 wafers, (b) micrograph of two different h-BN memristor arrays size 1×3 and 1×10, (c) Raman spectra, (d) TEM cross-sectional image of Au/h-BN/Ti/Au memristors.

BN memristor structure is achieved via cross-sectional TEM imaging (Figure 3.3d) revealing the layered nature of the h-BN film. The dark and blurry regions in the TEM image may reveal lattice disorder and native defects along grain boundaries known to be responsible for conductive nanofilament formation and resistive switching behavior (C. Pan et al.). Compositional analysis using electron energy loss spectroscopy (EELS) confirms the regions of N, B and Ti within the stack as highlighted in Figure 3.3d (right panels).

A comprehensive analysis of the resistive switching behavior of h-BN memristors is provided in Figure 3.4. Dual voltage sweep measurements are used to observe hysteresis in the current-voltage (I-V) characteristics associated with transitions between a high resistance state (HRS) and a low resistance state (LRS). In these measurements, a DC voltage across the top and bottom electrodes is swept (starting from zero) up to a positive value (e.g., 1.5 V), then back to a negative value (e.g., –1 V), and back to zero, all while measuring the current through the memristor. The results from 30 cycles of dual voltage sweeps are plotted in Figure 3.4a for an h-BN memristor with 3 μm × 3 μm active area (area of overlap between top and bottom electrodes). The number labels indicate the sweep direction, each light gray line is data from a single cycle, and the blue line with circles is the average from all 30 cycles. As shown, a compliance (limit) is applied to the current at a value of 100 μA to control the programming of the LRS by limiting the "strength" of the conductive path being formed. The measurements indicate repeatable results with little cycle-to-cycle variation and low set and reset voltages (approximately +/- 0.5 V for

set/reset). Figure 3.4b are cumulative distribution plots of the high and low resistance states (HRS and LRS) from all I-V measurement cycles extracted at a read voltage of V_{read} = 0.1 V. The cumulative distribution plots reveal an on/off ratio exceeding an order of magnitude (i.e., >10× ratio). Figure 3.4c shows the extracted HRS and LRS from the same device as a function of cycle number, showing good cycle-to-cycle repeatability.

Finally, Figure 3.4d shows the results from a room temperature retention test, where the current in an h-BN memristor is sampled for up to 10,000 seconds immediately after programming to HRS and LRS. The results from the retention test show negligible drift in the programmed state of the h-BN memristors. This could suggest another potential

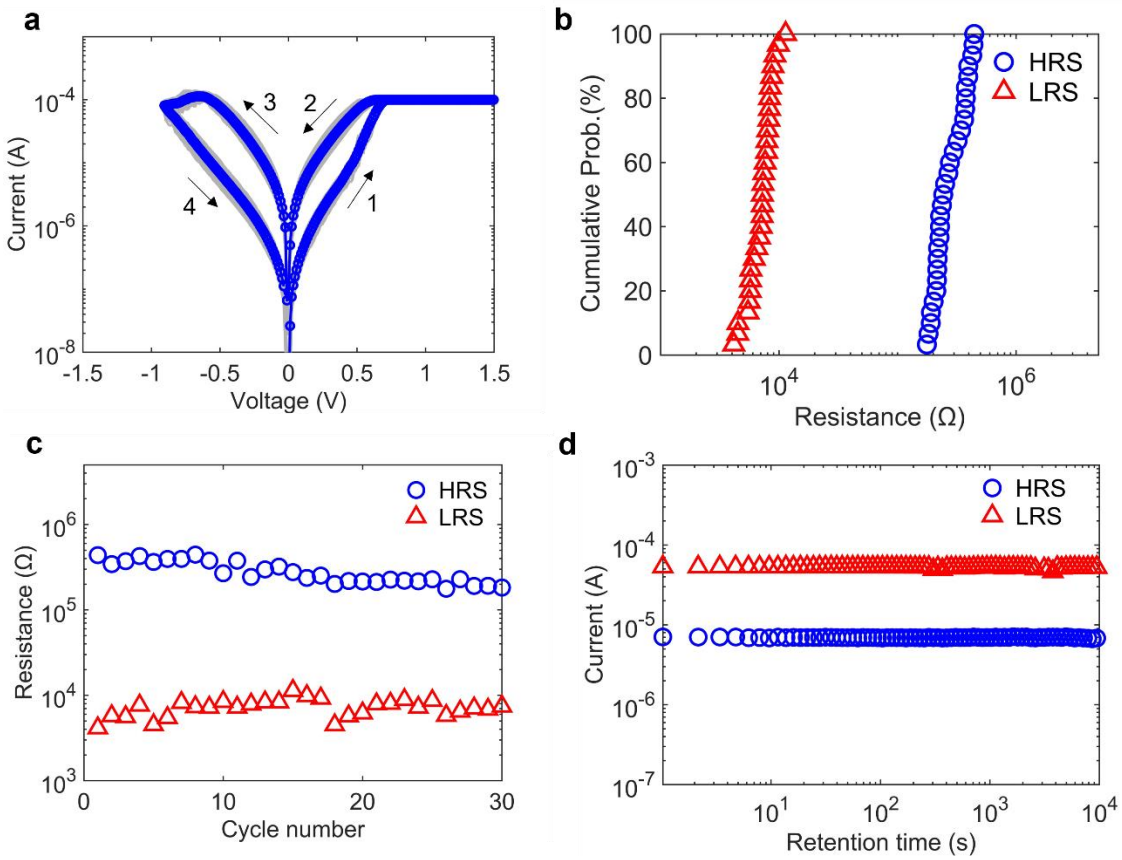


Figure 3.4. (a) I-V characteristics, (b) cumulative distribution plot of HRS and LRS, (c) the resistance corresponding to HRS and LRS as a function of cycle number, (d) room temperature retention.

advantage of 2D h-BN memristors over conventional (bulk) oxide-based RRAM which suffers from the retention-induced conductive drift that can lead to significant degradation in inference accuracy in neuromorphic computing systems (Baroni et al.).

In Figure 3.5a, we explore the analog programmability of the h-BN memristors using 100 ns pulses. We use voltage amplitudes of 0.9 V for positive pulses and -1.1 V for negative pulses. We apply 15 consecutive positive pulses followed by 15 consecutive negative pulses and measure the current after each pulse using V_{read} = 0.1 V. The measured data is plotted for 200 cycles (a total of 6,000 pulses). Here, the gray lines correspond to the individual 30-pulse cycles, and the red line with circles is the average for all 200 cycles. The results indicate good monotonic behavior with pulse polarity (i.e., current increases with positive pulses and decreases with negative pulses). Moreover, the h-BN memristors show good endurance to pulse programming as evidenced by consistent resistive switching behavior even after 6,000 pulses. We note that the dynamic range in Figure 3.5a (range of programmed currents) is small (~2.5×) due to the relatively small amplitudes of the programming pulses (+0.9 V and –1.1 V). We can estimate the programming energy (energy used in changing conductance with a single programming pulse) assuming a current of approximately 45 μA (estimated current for V_{pos} = 0.9 V instead of V_{read} = 0.1 V) and using t_{set} = 30 ns to obtain $E_{set} = (V_{pos})(I_{set})(t_{set})$ ~ 1.2 pJ/pulse. In this calculation we use t_{set} = 30 ns instead of 100 ns (test instrument limitation), as determined by extrapolation of transient measurements indicating that approximately 20-30 ns is

sufficient to switch the h-BN memristors. Note that this programming energy is higher than the energy used in reading the device in a dot-product operation as will be described next.

3.3 Hardware Implementation of Dot-Product Using H-BN Memristor Array

Having verified the NVRS and analog pulse programmability of individual devices, we then test an array of h-BN memristors. Figure 3.5b is a schematic of the array, where we illustrate the voltages applied to each top electrode, and the total current through the shared bottom electrode given by the dot-product of voltages (v_i) and the corresponding

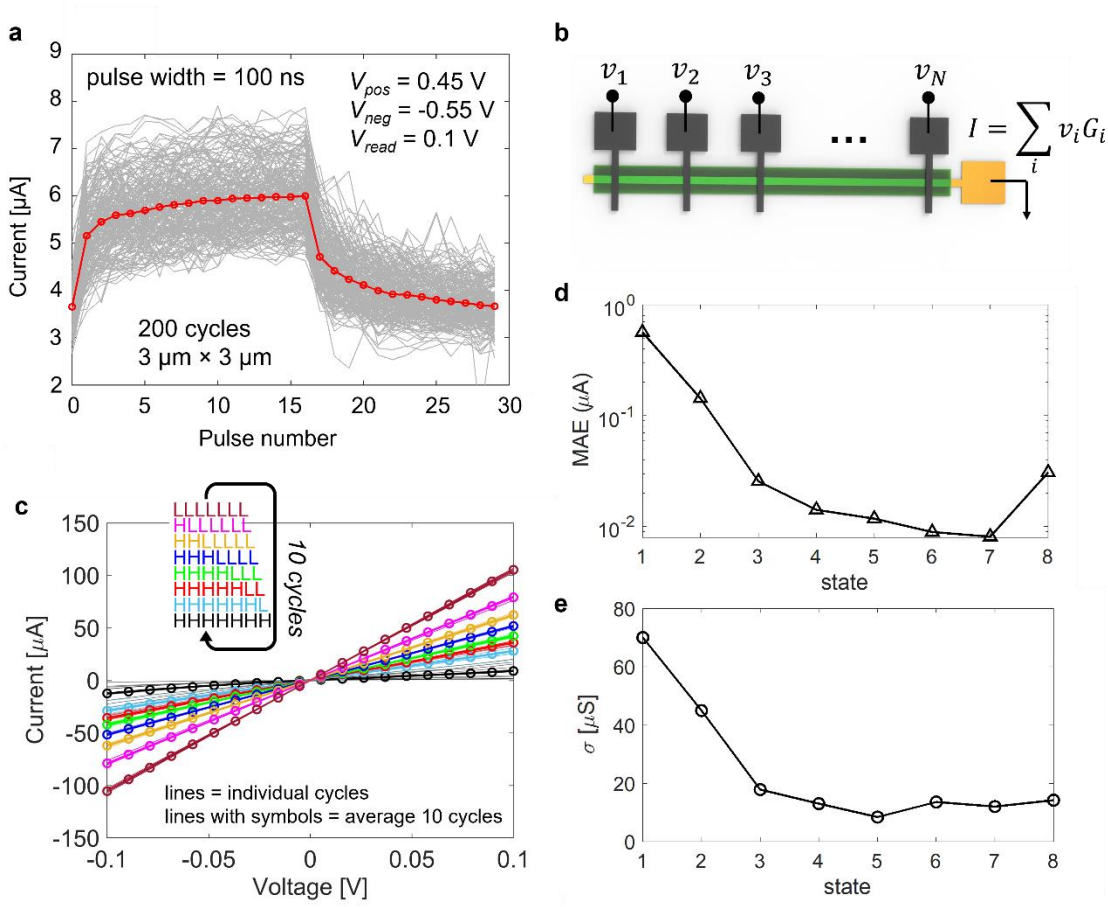


Figure 3.5. (a) Pulse programming of a single h-BN memristor, (b) schematic of the h-BN memristor array illustrating dot-product operation, (c) dot product computation in hardware, (d),(e) MAE and standard deviation in the dot-product computation.

memristor conductances (G_i) as $I = \sum_i v_i G_i$. For the dot-product test, we sequentially program individual devices from HRS to LRS (7 devices in a row). We conduct a voltage sweep on the top electrodes (all top electrodes at the same voltage) after programming each device. During the sweep, we measure the total current through the shared bottom electrode. Once all devices have been programmed to LRS, we reset all devices to HRS and begin the next cycle (repeated 10 times). The data is shown in Figure 3.5c as current vs. swept voltage. Here, each color represents a different 'state' corresponding to a different number of memristors in LRS. For each state, we plot the individual cycles (lines) as well as the average (thick lines with circles). This is a direct measurement of dot-product implementation on memristor hardware scanning both relevant parameters (i.e., voltages and conductances). We note that the dot-product computations show good linearity and reproducibility (quantitative analysis below). We estimate read energy from the dot-product measurements as $E_{read} = (V_{read})(I_{read})(t_{read})/N$, where N is the number of memristors in parallel. For the worst case (all devices in LRS), the energy is between 200 aJ and 20 fJ per operation (each MAC counted as two operations).

Quantitative Analysis

Non-ideal memristor behavior (e.g., nonlinear I-V characteristics) as well as variability in conductance programming (inherent stochastic nature of filamentary resistive-switching mechanisms) can lead to inaccuracy in the computation of dot-products. This inaccuracy can introduce significant error in the implementation of artificial neural networks (ANN) using memristor hardware. To quantify accuracy in dot product computation, we calculate the mean absolute error (MAE) as well as the standard deviation

in our implementation using h-BN memristor arrays. To obtain MAE, we perform a linear fit to the average currents (symbols) in Figure 3.5c for each state (using a least-squares method). This linear fit represents a perfectly linear or "exact" dot-product implementation for each corresponding state. We then compare the experimental values (all cycles) against the exact calculation to obtain MAE over the entire voltage range (from –0.1 V up to +0.1 V). MAE is plotted in Figure 3.5d for each different state (average over all cycles). As shown, the error is largest for state = '1' which corresponds to all devices in HRS. However, this MAE is relatively small (<1 μA) compared to the range of current (up to ~100 μA) and drops significantly (down to ~10 nA) with more devices in LRS. We attribute the small error in the dot-product computation to good linearity in the h-BN memristor I-V characteristics over this read voltage range (from –0.1 V to +0.1 V). We also quantify cycle-to-cycle variability based on extractions of standard deviation (σ) in the *effective state conductance* from the dot-product data (i.e., the slope from the I-V characteristics in Figure 3.5c). The standard deviation is plotted in Figure 3.5e for each different state, where the largest deviation of ~70 μS happens for state = '1' which corresponds to all devices in HRS. We note that this is small compared to the full range of conductance (200-1000 μS) in the dot-product implementation.

3.4 Hardware Implementation of Stochastic Logistic Regression Using H-BN Memristor Array

As a demonstration of dot-product computation in a machine learning algorithm, we present the implementation of gradient-descent-based stochastic logistic regression for image classification. Logistic regression is widely employed for object categorization and pattern identification. Here, we carry out the hardware-level computation of dot-product (including the corresponding weight updates) on an array of h-BN memristors. Gradient descent is an iterative optimization approach for minimization of a cost function associated with classification error. In *stochastic* gradient descent (an online version of this technique that processes data one observation at a time), the weight updates (pulse-based adjustments in conductance) are exerted on the h-BN memristor array at every iteration in the training process. Since memristive crossbar arrays are unable to achieve the steepest gradient

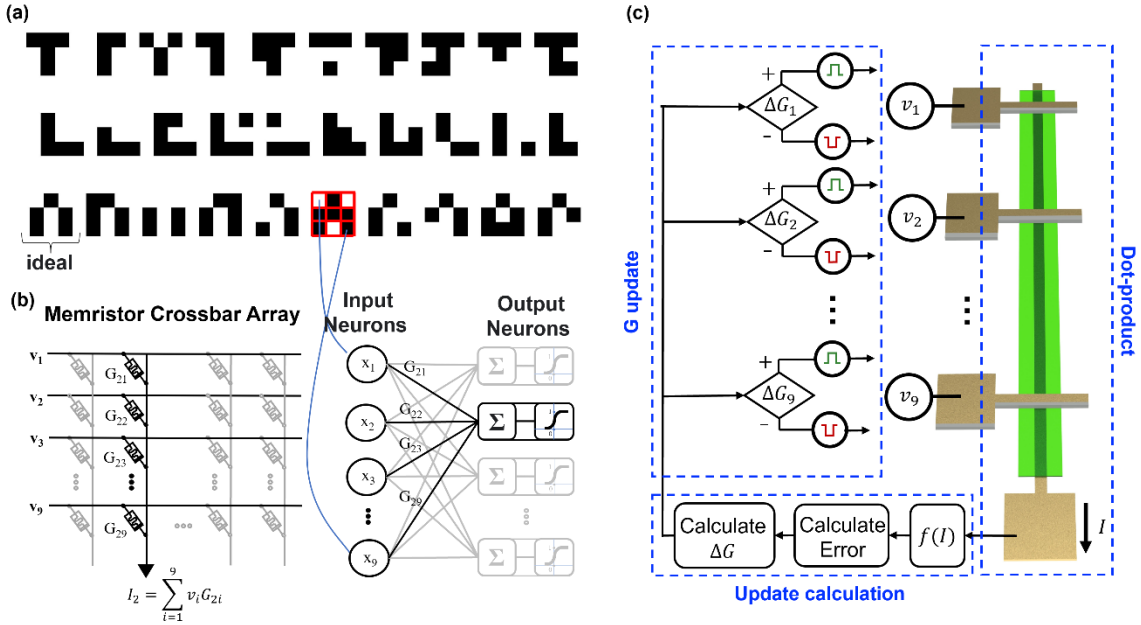


Figure 3.6. (a) Training images, (b) graph illustration of logistic regression on h-BN memristor arrays, (c) a flowchart representing one iteration step in the training process for stochastic logistic regression (see text).

descent in an effective manner due to device limitations, we use a modified (hardware-compatible) gradient descent rule to train the h-BN memristor array. In this approach, a single pulse, or a set of consecutive programming pulses (with fixed amplitude and width) are applied to update the conductance of each memristor in the array as determined by the magnitude and polarity required weight updates given by the gradient descent optimization algorithm. In this demonstration, we use a dataset of size 500 containing 3×3-pixel noisy binary images of characters "T", "L" and "n" (training images). We train a 9×1 h-BN memristor array to discern images of character "T" from the other characters in a separate dataset (test images). We note that the training and test images are independently generated with one randomly flipped pixel. Figure 3.6a shows a subset of the images illustrating the ideal characters (1st image for each row) as well as some noisy samples (1 modified pixel).

The training process includes two consecutive steps during each iteration: a feedforward integration mode and a feedback update mode. In feedforward integration mode, vector-matrix multiplication (a collection of dot-products) is performed to achieve a hypothesis based on the accumulated output currents. In this hardware implementation, each binary picture pixel is translated to a crossbar input voltage equal to +0.1 V for white pixels and –0.1 V for black pixels, as shown in Figure 3.6b. Importantly, these input voltages are within the range in the I-V characteristics of h-BN memristors showing good linearity in dot-product computation (see Figure 3.5c). Then, the image-dependent array of voltages is applied as inputs to the memristor array to obtain an output current given by $I_j = \sum_{i=1}^{9} v_i G_i$ (i.e., the dot-product of input voltages and conductance "weights"). We then apply the logistic activation function $f_j = \frac{1}{1+e^{-I_j}}$ to the (normalized) current to obtain an output bounded between 0 and 1. This output represents the likelihood that the input image

corresponds to a specific category (i.e., corresponds to a specific character like "T"). Each training image contains a "label" that indicates if it corresponds to a given category. In this example we are training the memristor array to recognize character "T" from the rest, so the training images have label y_j = 1 for character "T" and y_j = 0 for all other characters. At each training step, the classification error is calculated as $\delta_j = f_j - y_j$. In feedback update mode, the conductance update (weight update) for each memristor is calculated as $\Delta G_{ij} = -\delta_j v_i$. Here we use a simplified, hardware-compatible update rule where a single programming pulse of polarity determined by the sign of ΔG_{ij} is applied to change the

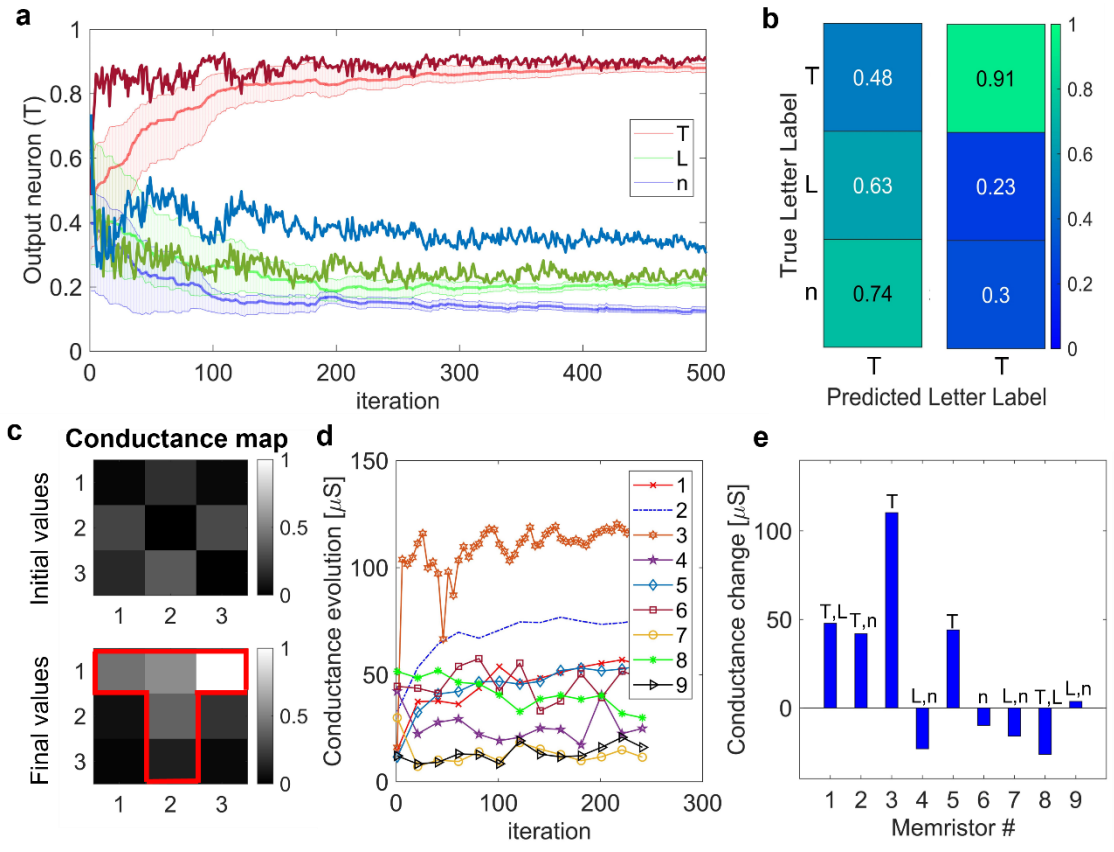


Figure 3.7. (a) Convergence of a logistic regression algorithm, (b) confusion matrix before and after training, (c) conductance maps before and after training, (d) evolution of experimental conductance vs. iteration, (e) change in conductance during training.

conductance of each memristor in the array. We use programming pulses with fixed widths and amplitudes of 30 ns, and +2.5 V/-2.6 V respectively. Figure 3.6c provides a flow diagram illustration of one iteration in our implementation of stochastic logistic regression on h-BN memristor array hardware.

Figure 3.7 summarizes the results of the classification algorithm implemented on the arrays of h-BN memristors. At predetermined training intervals, the classification accuracy is assessed using the test images. Figure 3.7a plots the output of the logistic function (f_j) as a function of training step (iteration). This is actually the average for all test images (100 test images for each character). The different lines with symbols correspond to the measured output for each different character (maroon for "T", blue for "n", and green for "L"). We see that for images of character "T" the value approaches 1, while for "L" and "n" it approaches 0, meaning an accurate classification as this array was trained to classify character "T". Also shown in Figure 3.7a are simulation results for the hardware implementation of stochastic multivariable logistic regression. The shaded regions indicate the range of the simulation results (min to max) from 10 different runs using random initial conductance values, and the solid line is the average. A small learning rate is factored into the conductance updates obtained from gradient descent to ensure a gradual change in conductance and improve convergence. In our simulations, we bound conductance values to G_{min} and G_{max} values obtained experimentaly from pulse testing of the h-BN memristors. The simulation represents an ideal situation where conductance updates are perfectly controlled (no variability) for all memristors in the array, and the dot product is linear and without cycle-to-cycle variability. The comparison between simulations and experiments indicates that our hardware implementation achieves similar

performance and accuracy to the ideal case. We note more abrupt changes in the experimental data, likely due to abruptness in changes of memristor conductance, but this appears to have little impact on the final accuracy. The rest of the results in Figure 3.7b-e are only for experimental results on the h-BN memristor arrays. Figure 3.7b shows the confusion matrix for the f_j values before and after 500 training steps. Clearly, the classification improves significantly with training, in accordance with Figure 3.7a. Figure 3.7c shows a conductance map of the h-BN memristor before and after the hardware-implemented training. It is observed that after 500 training steps the pattern 'T' becomes noticeable, while before training the conductance pattern was random. A reasonable explanation for the full brightness of pixel 3 is the maximum percentage of reinforcement applied to this pixel during training since it is only present in pattern "T" (not in "L" or "n"). In fact, this is evident in the evolution of conductance for the full array over the course of 250 training steps plotted in Figure 3.7b. Another visualization of the learning process is the change in conductance during training for each h-BN memristor as plotted in Figure 3.7e. As shown, devices that correspond to pixels that are in the ideal "T" pattern are more strongly reinforced (positive change in conductance) compared to the pixels that are not.

3.5 Methods

Ti/h-BN/Au memristor array fabrication: The Au/h-BN/Ti/Au (from bottom to top) memristor arrays were fabricated on a 90 nm SiO2/Si wafer. First, the shared bottom electrodes (5 nm Cr/20 nm Au) with 3 μm width were patterned on the substrate via photolithography and e-beam evaporation. Then, the CVD-grown few-layer h-BN film on copper from Six Carbon Technologies (Shenzhen) was transferred onto the prepared

SiO2/Si substrate by wet transfer method. Subsequently, few-layer h-BN film was patterned to expose the 100 μm by 100 μm bottom electrodes pads using photolithography and oxygen plasma process. Finally, the top electrodes with 30 nm Ti and 30 nm Au were patterned with the same electrode width and the same methods as that of the bottom electrodes.

Electrical characterization: A Keithley 4200 semiconductor characterization system (SCS) was used for the electrical characterization on a Cascade semiautomatic probe station. Source measure units (SMUs) were used to measure I-V characteristics. Pulse measure units (PMU, model 4225) were used in conjunction with SMUs to read currents during pulse programming experiments. Keithley 4225-RPM remote amplifier/switch is employed to switch between PMU and SMU for pulse measurements. Figures 3.7 include photographs of the experimental setup.

Logistic regression test: The experimental demonstration of logistic regression was performed utilizing a nine-slot National Instrument (NI) PXle-1078 chassis. A customized probe card and test board together with a modular script developed in the graphical development environment NI LabVIEW was used to interface the h-BN memristor arrays and to implement the algorithm on hardware. For dot product (feedforward mode), the input voltages applied to the memristor array used the PXl-6738, a 32-channel analog card that sources analog signals, while an SMU (PXI-4140) channel is used to read currents. In feedback update mode, positive and negative programming pulses are utilized to update the conductance value of the memristors using the PXI arbitrary waveform generator (PXI-5413) while SMUs are utilized to measure resistance (conductance) at each iteration. Figures 3.8 include photographs of the experimental setup.

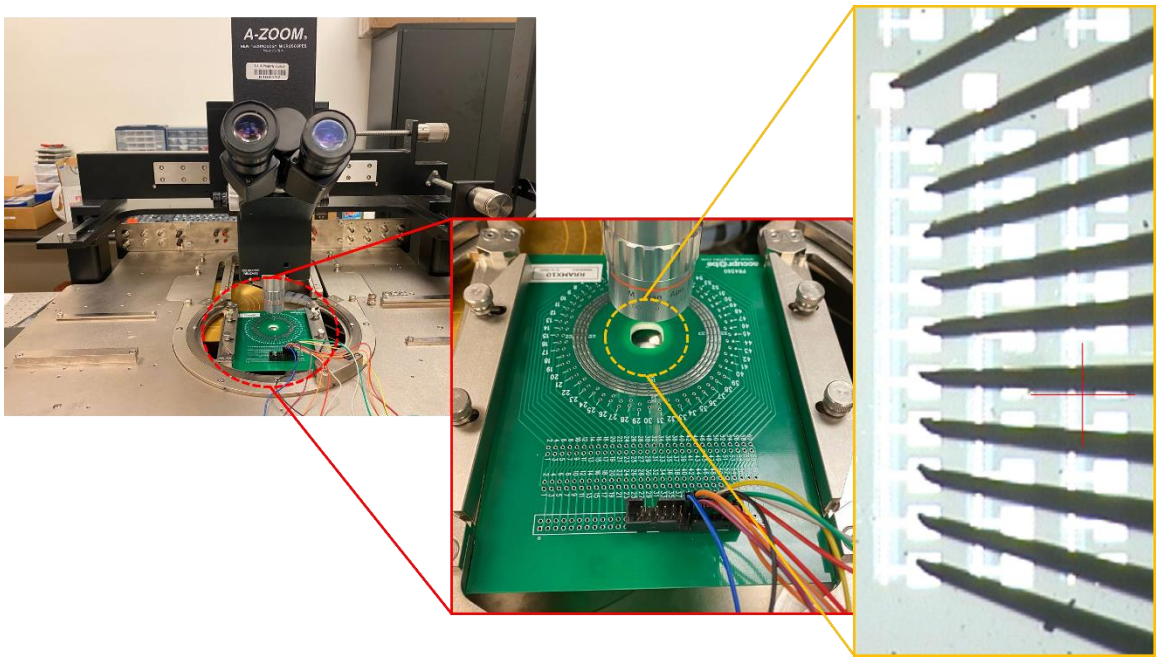


Figure 3.8. Customized probe card on a Cascade semi-automatic probe station contacting memristor array. Right panel shows a micrograph of probed h-BN memristor array.

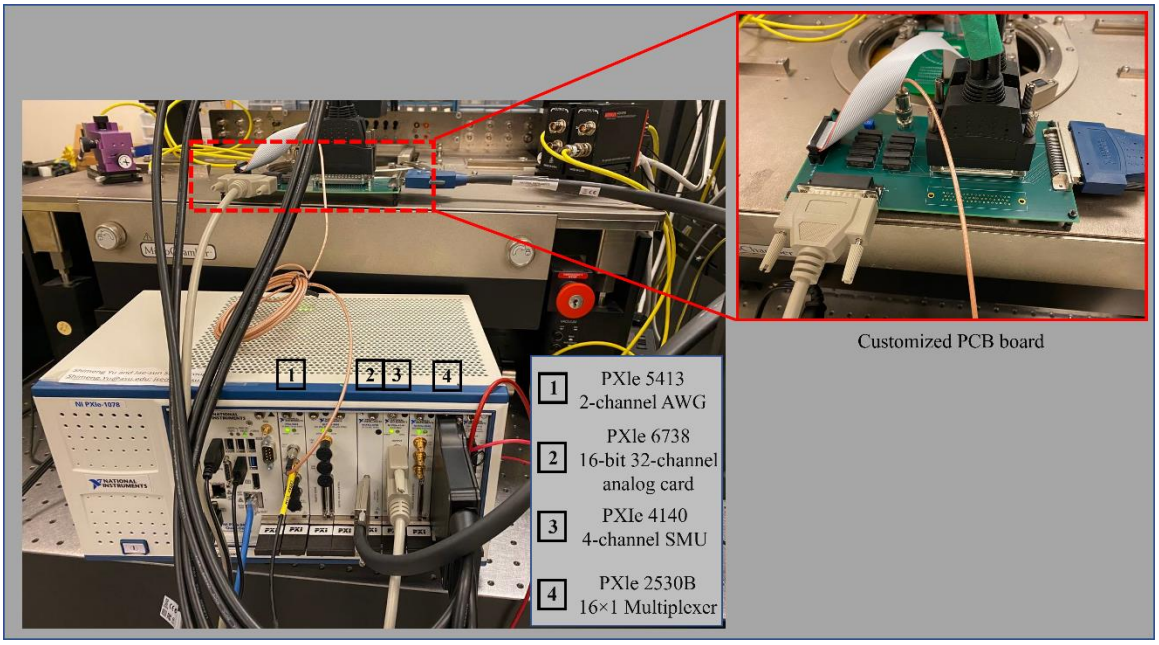


Figure 3.9. Photo of the test setup showing nine-slot National Instrument (NI) PXIe-1078 chassis connected to a customized PCB board interfacing between the NI device and probe card. Right panel shows the customized PCB interface.

CHAPTER 4

LINEAR REGRESSION WITH 2D HEXAGONAL-BORON NITRIDE (H-BN) MEMRISTOR ARRAYS

4.1 Fabrication of H-BN Memristor Arrays

Multilayer CVD-grown h-BN was transferred from copper onto a 90 nm SiO_2/Si substrate patterned with Au bottom electrodes. The h-BN film was then shaped using standard photolithographic and etching techniques to expose the bottom electrodes. Subsequently, we prepared top electrodes through patterning and Ti deposition using e-beam evaporation and lift-off. Figure 4.1a shows a schematic of the fabricated Au/h-BN/Ti memristors arrays where the Au bottom electrode (BE) is shared across various devices each having an independent Ti top electrode (TE) (1 × 3 and 1 × 10 arrays are shown). Figure 4.1b illustrates the cross-section of Au/h-BN/Ti memristor. Figure 4.1c is a photograph of the memristor arrays on a ~2 cm by 2 cm SiO_2/Si wafer. A micrograph of the fabricated h-BN memristor arrays shown in Figure 4.1d corroborates the dimensions of the 100 μm × 100 μm squared pads and the narrow and long electrodes with 3 μm × 3 μm active areas.

4.2 Resistive-Switching Properties and Multistate Non-Volatile Pulse Programmability

Individual h-BN memristors from the arrays were measured electrically to evaluate their resistive-switching properties (see Methods for details on electrical characterization). Current-voltage (*I-V*) characteristics were obtained by sweeping a voltage across the top and bottom electrodes while measuring current. Figure 4.2a plots 100 consecutive cycles

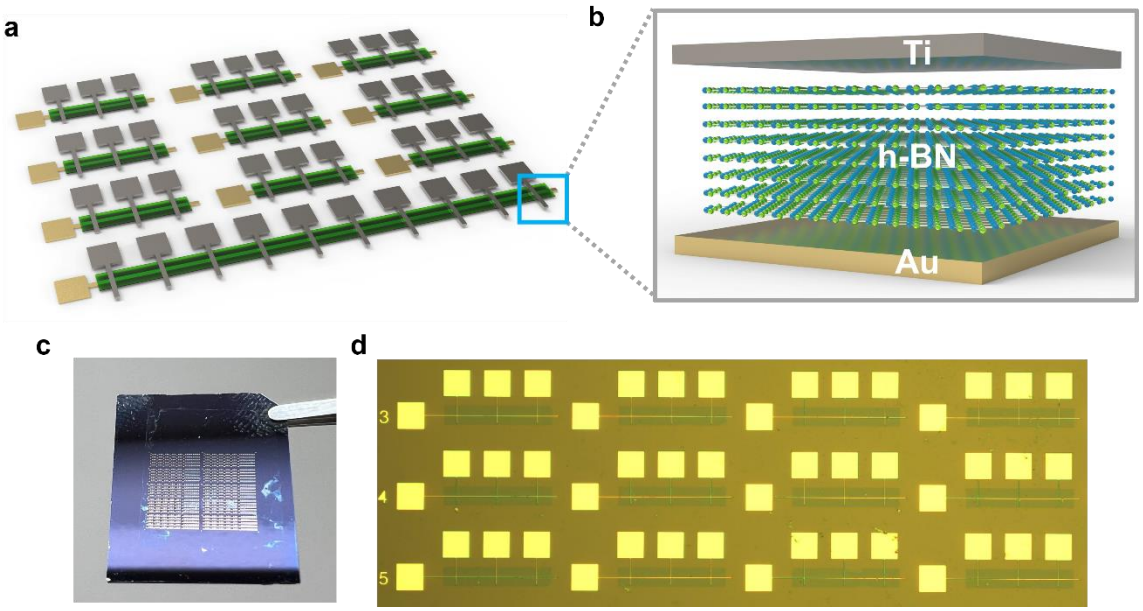


Figure 4.1. (a) Schematic of the Au/h-BN/Ti memristor arrays, (b) cross-sectional schematic of single memristor, (c) photograph of Au/h-BN/Ti memristor arrays on 90 nm SiO_2/Si wafer under ambient light, (d) micrograph of arrays with 3 µm × 3 µm active

of *I-V* measurements on an Au/h-BN/Ti memristor with a 3µm × 3 µm active area. A compliance of 0.1 mA was activated for positive applied voltages. The numbered labels indicate the sweeping process during the *I-V* measurement. As shown, clear transitions occur between resistive states, evidence of a forming-free bipolar resistive-switching (RS) operation with low cycle-to-cycle resistance variability and low set and reset voltages (approximately 1 V and -1 V). The cumulative distribution plot of the resistive states extracted at a read voltage of 0.1 V from all 100 cycles is shown in Figure 4.2b. Two distinct states labelled as HRS (high resistance states) and LRS (low resistance state) are easily observed as their distributions are separated by approximately two orders of magnitude. Another illustration of the HRS and LRS distributions is provided in Figure 4.2c where the resistances are plotted as a function of the cycle number. A histogram of the set and reset

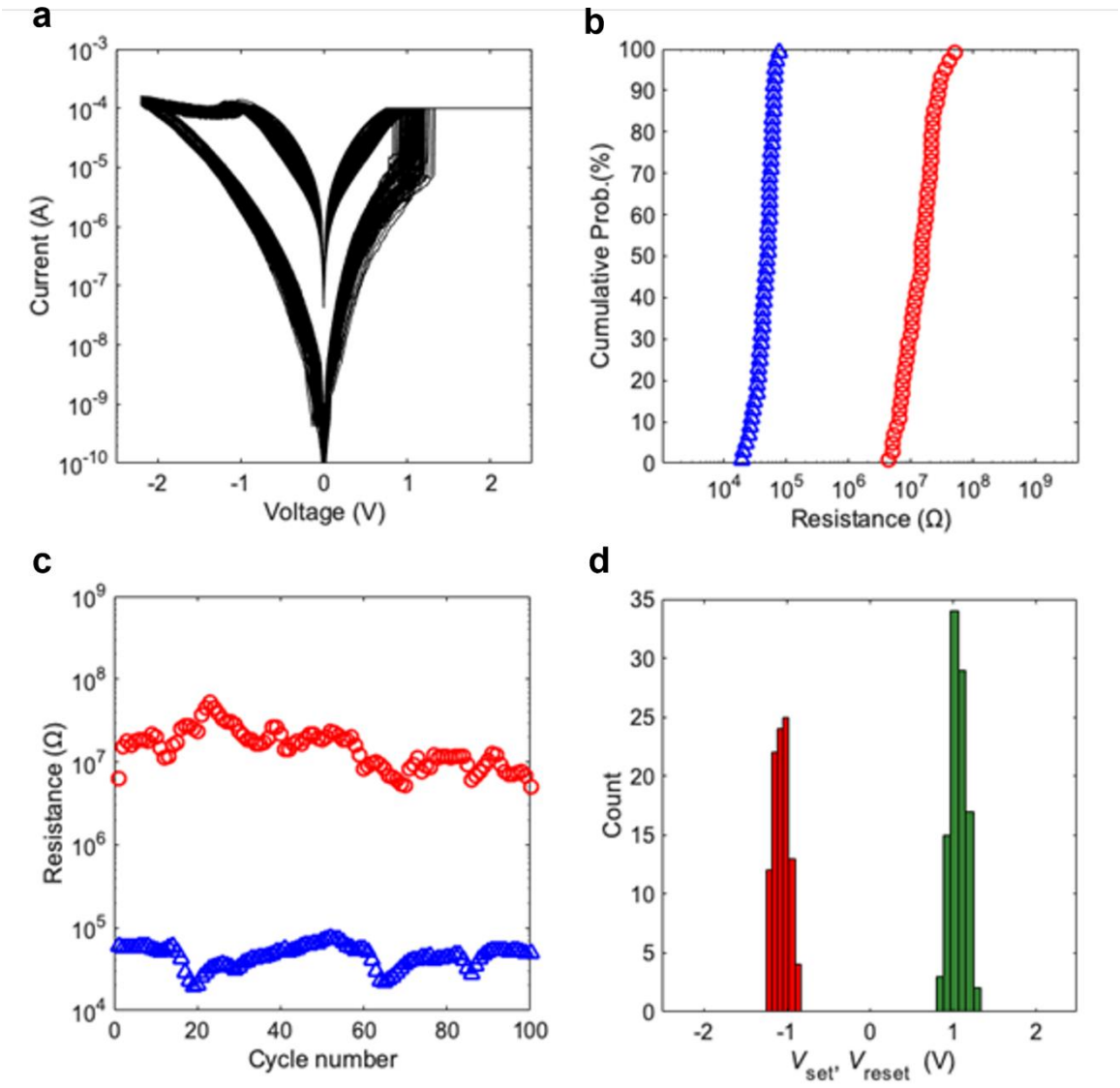


Figure 4.2. (a) I–V characteristics, (b) cumulative probability distribution of the HRS and LRS (read at 0.1 V), (c) resistance vs. cycle number plot, (d) histogram of set and reset voltages.

voltages corresponding to transitions between HRS and LRS is shown in Figure 4.2d. All results indicate a stable and reliable RS bipolar operation.

Moreover, achieving multiple conductive states through the application of programming pulses is critical for the implementation of neuromorphic hardware and for the analog-based implementation of machine learning functions in memristor arrays. We

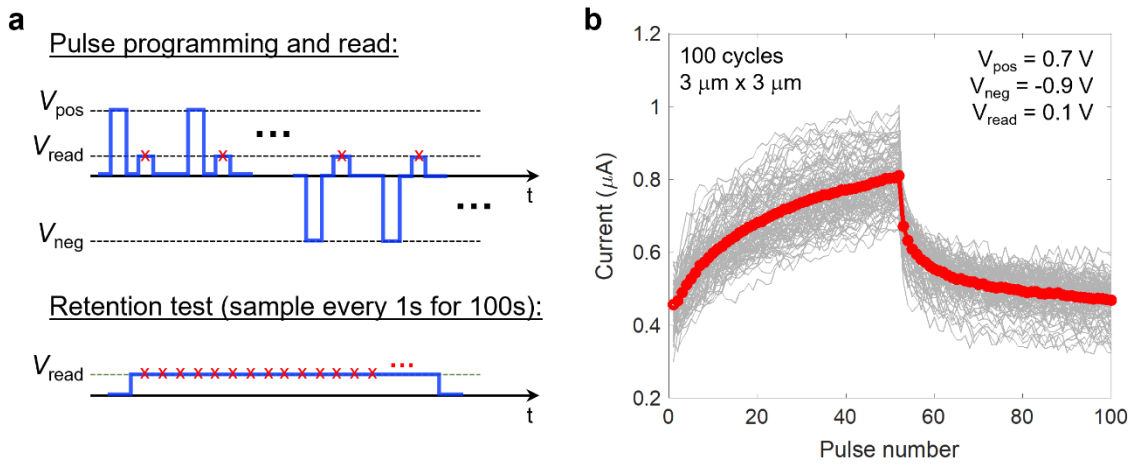


Figure 4.3. (a) Diagram of the pulsed measurements and retention test, (b) 100 cycles of pulse programming for Au/h-BN/Ti memristor with 3 μm × 3 μm active area.

investigate the multistate pulse programmability of the Au/h-BN/Ti memristors by applying a sequence of positive/negative voltage pulses (pulse width is 500 ns, amplitudes indicated in Figure 4.3b). After each pulse a small read of 0.1 V is applied to read the current (conductive state) of the device (see Figure 4.3a top panel). The results are shown in Figure 4.3b, where 100 cycles of 50 positive pulses followed by 50 negative pulses were applied. The gray lines are the results from each individual cycle and the solid red line with circles is the average from all 100 cycles. The results show a gradual change in conductance (from ~ 4 to 10 μS) indicating good analog (i.e., multistate) programmability. Due to the fast-switching behavior (nanoseconds), a low energy consumption per programming pulse of $E_{\text{pulse}} = (I)(V)(t_{\text{pulse}}) \approx 125$ fJ is achieved. We note that this can be further reduced to aJ/pulse by applying a low compliance current as previously reported on h-BN memristors (S. Chen et al.).

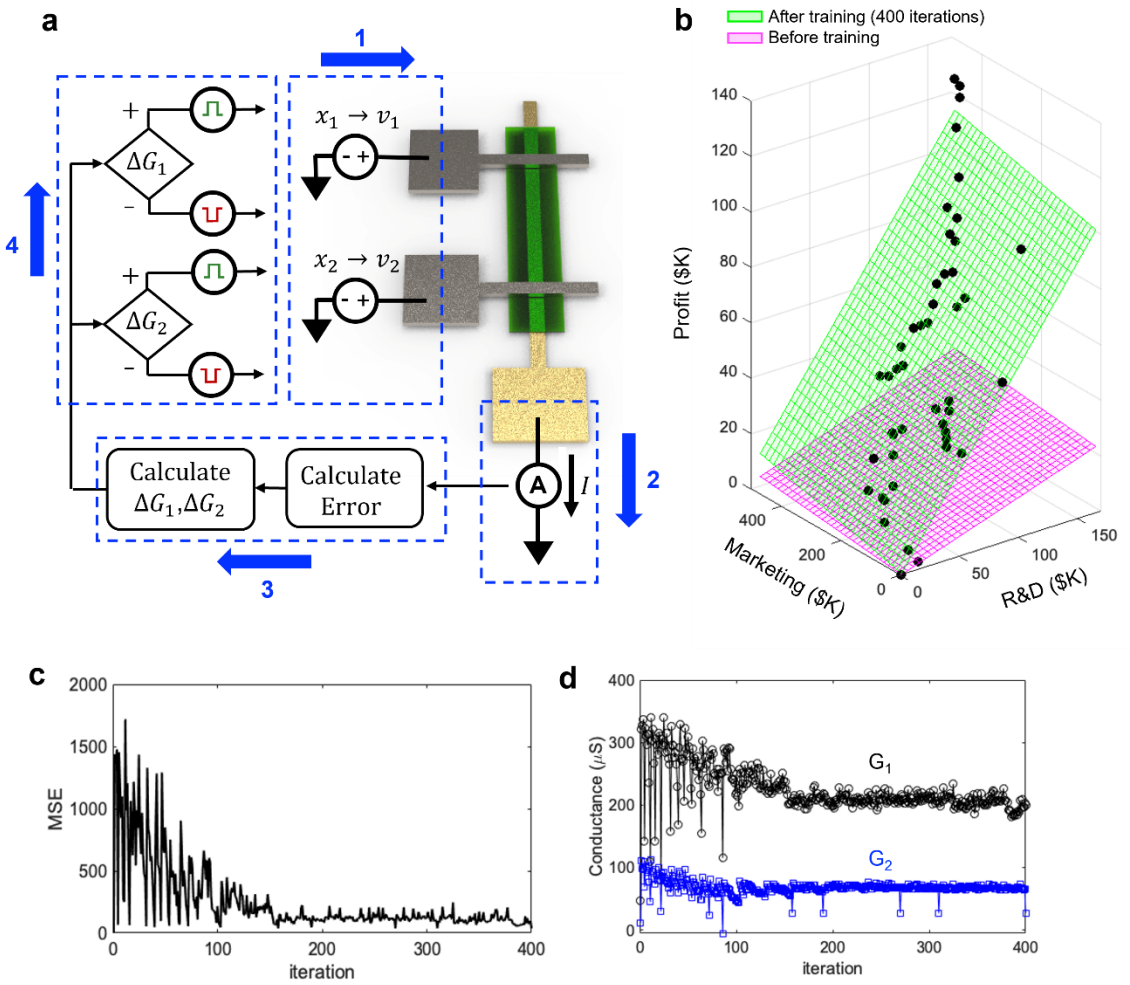


Figure 4.4. (a) Flow diagram for stochastic multivariable linear regression, (b) model prediction fit to training data training, (c) evolution of MSE with training, (d) evolution of model parameters vs. iterations.

4.3 Hardware Implementation of Linear Logistic Regression Using H-BN Memristor Array

We now demonstrate the implementation of stochastic multivariable linear regression on the h-BN memristor arrays. In this implementation we use the h-BN memristor arrays to predict the profit of a startup company given its investment in marketing and in research and development (R&D). Our model is trained using a dataset from 50 startup companies available online (Farhan). In this implementation, the memristor

conductances (G_1 and G_2) are the model parameters. The training process is illustrated in Figure 4.4a. For each training step a single sample from the dataset is randomly selected (the sample includes profit, marketing, and R&D in $K). The input variables (marketing and R&D) are translated (normalized) to voltages between 0 and 0.15 V. These voltages are applied to the h-BN memristors (v_1 and v_2). This is important for the implementation of linear regression as the prediction (h) is determined from the output current of the h-BN memristor arrays given by the dot product as

$$I = v^{\mathrm{T}}G\ , v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, G = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} \qquad (1)$$

The prediction is then compared against the training sample (profit) from which we determine the error and the required update for each of the model parameters (ΔG_1 and ΔG_2) (see Methods for details of the implementation). Here we use a hardware-compatible approach to update the model parameters whereby a single programming pulse is applied to each memristor, and the polarity of the pulse is determined by the sign of ΔG_1 and ΔG_2. This programming pulse will slightly adjust the conductances to ultimately minimize the error in the prediction. To achieve good convergence, stochastic regression algorithms typically limit the parameter updates with a learning rate that is gradually reduced with training number. In our experiments the learning rate is implemented by gradually reducing the amplitude of the programming pulses. We reduce the amplitude of the programming pulses by 0.1% after each iteration (starting with ±1 V, the pulse amplitude will be reduced to ±0.67 V after 400 training steps). The width of positive and negative programming pulses is kept fixed at 500 ns throughout the training process.

Figures 4.4b-d show the results of the stochastic linear regression implementation. In Figure 4.4b we plot the training data (black dots) as well as the model prediction before (magenta plane) and after 400 training steps (green plane). As shown, the trained model clearly predicts the profit of startup companies based on their investments in marketing and R&D much better than the before training. A more quantitative result is shown in Figure 4.4c where we plot the mean squared error (MSE) as a function of the training step (i.e., iteration) as given by $\text{MSE} = (1/N)\sum_i \delta_i^2$ where N is the sample size (50 in this case) and $\delta_i = h_i - y_i$ is the error in the prediction. As shown, the MSE reduces with training indicating good convergence of the algorithm. Figure 4.4d shows the change in conductances G_1 and G_2 (the model parameters) during the training process. We see larger updates and fluctuations in the conductances during the initial training steps, and eventually converge to the optimal values for the model parameters.

4.4 Methods

h-BN memristor and memristor arrays fabrication: The Au/h-BN/Ti memristor arrays were fabricated on a 90 nm SiO2/Si wafer. First, the bottom electrodes (5 nm Cr/35 nm Au) with 3, 20, and 50 μm width were patterned on the substrate via photolithography and e-beam evaporation methods. Second, CVD-grown multilayer h-BN on copper from Graphene Supermarket was transferred onto the prepared SiO2/Si substrate by wet transfer method. Third, h-BN film was patterned to expose the 100 μm by 100 μm bottom electrodes pads using oxygen plasma. Finally, the top electrodes (70 nm Ti) were patterned with the same electrode width and the same methods as that of the bottom electrodes. The top electrodes are exposed to air and a thin surface layer may be oxidized over time. This oxidized layer can be easily penetrated with probe needles during measurements, and its

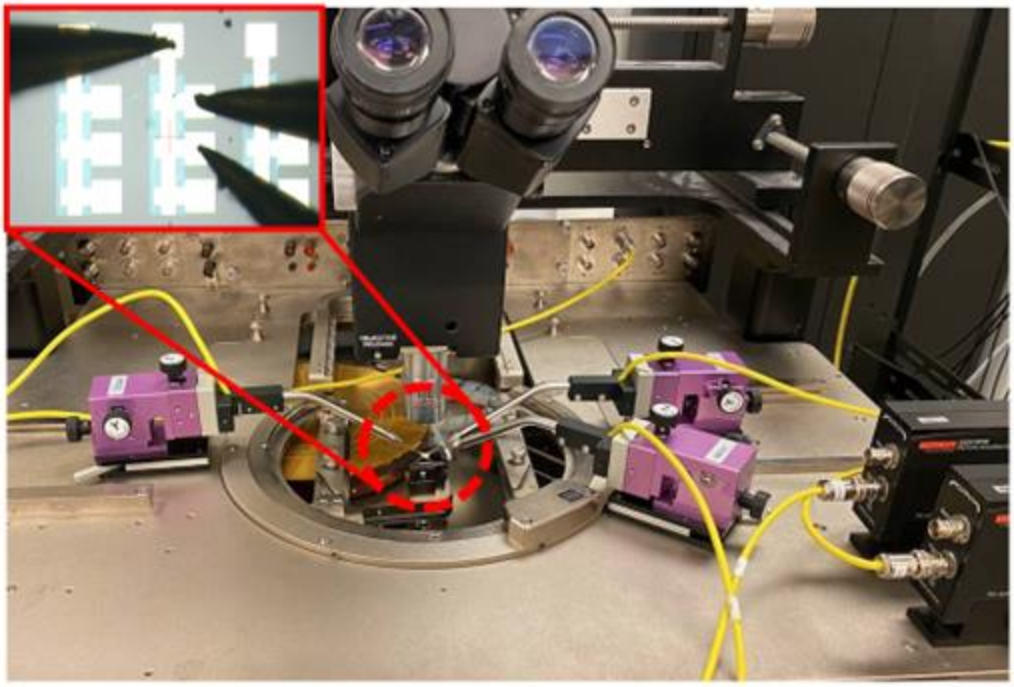


Figure 4.5. Probe connections made on the h-BN memristor array pads shown on the inset are routed with triaxial cables through the Keithley remote amplifier/switch (RPM) where we can automatically connect pulse or source/measure units (PMU or SMU).

impact on the resistive switching behavior has been ruled out by comparing against devices with Au-capped top electrodes that show very similar characteristics.

Electrical characterization: The electrical characterization was conducted on a Cascade semi-automatic probe station using a Keithley 4200 semiconductor characterization system. The DC *I-V* measurements were performed using source measure units (SMUs), while the pulse programming experiments used a combination of pulse measure units (PMU, model 4225) for programming pulses and SMUs for reading currents. In the pulse programming experiments we switched between PMU and SMU automatically using a Keithley remote amplifier/switch (4225-RPM). Figure 4.5 shows the experimental setup.

Linear regression test: Our implementation of multivariable stochastic linear regression on the Au/h-BN/Ti memristor arrays was trained using a dataset available online (Farhan). The experimental demonstration was done with a Keithley 4200 SCS using a custom test script developed in the Keithley user library tool (KULT) and executed in the Keithley interactive testing environment (KITE). The input parameters to the test script are the minimum and maximum conductance values for each memristor (predetermined based on pulse measurements, used to normalize output currents from the arrays), the initial values for the programming pulse amplitudes, the constant value for the width of the programming pulses, and the number of iterations. The test script loads the training data and normalizes the independent variables (in this case marketing and R&D investments in thousands of dollars) to voltages between 0 and 0.15 V. We also subtract a constant offset (y-intercept) from the dependent variable (profit) so that the model is based only on two regression coefficients (model parameters represented by the memristor conductances). The script then goes into a loop where it randomly selects a sample for the data set and apply the read voltages (v_1 and v_2) that correspond to the independent variables of that sample. The current $I = v_1 G_1 + v_2 G_2$ is read at the output of the h-BN memristor arrays (shared bottom electrode) and is translated from Amps to dollars to be compared against the training sample. This read operation is conducted with the Keithley SMUs. We then calculate the error (δ) in the prediction as well as the required update for each model parameter (i.e., ΔG_1 and ΔG_2). From the minimization of the cost function (i.e., $\delta^2/2$) the updates are calculated as $\Delta G = -\delta v$. Here we propose a simplified hardware-compatible regression approach where the memristor conductances (i.e., the model parameters) are updated through the application of a single programming pulse, and the polarity of the

pulse is determined by the sign of the corresponding ΔG. The programming pulses are applied using Keithley's 4225 PMU (pulse width is fixed to 500 ns). Gradient descent algorithms typically use learning rate decay to improve convergence, where model parameter updates are weighted by a learning rate (α) that is reduced gradually as training advances. In our hardware demonstration we introduce learning rate decay by gradually reducing the amplitude of the programming pulses (we have reduced the amplitude of the programming pulses by 0.1% after each iteration).

CHAPTER 5

UNSUPERVISED LEARNING IN HEXAGONAL BORON NITRIDE MEMRISTOR-BASED SPIKING NEURAL NETWORKS

5.1 Overview of Memristor-based Spiking Neural Networks

Artificial neural networks (ANN) offer an approximate simulation of the human brain and can be realized with highly interconnected processing units in neuromorphic computing hardware. Despite remarkable advancements in neuromorphic computing hardware, biological neural networks continue to outperform ANNs in terms of energy efficiency and capabilities for online learning. To better emulate biological neural networks and to bridge the gap between neuroscience and machine learning, spiking neural networks (SNN) exploit event-based spikes for data transfer and processing. SNNs employ processing units and biologically plausible learning models (e.g., spike-timing-dependent-plasticity or STDP) that closely mimic the human brain. SNN-based neuromorphic computing systems are noteworthy and promising solution to improve energy efficiency as demonstrated with TrueNorth, a neuromorphic CMOS integrated circuit produced by IBM (Merolla et al.), and Loihi, a neuromorphic processor with on-chip learning from Intel (Davies et al.). While not a direct comparison, TrueNorth can produce 400 billion SOPS (synaptic operations per second) per watt for networks with high spike rates and a high number of active synapses, compared to one of the most energy-efficient supercomputers at the time that only managed 4.5 billion FLOPS (floating-point operations per second) per watt (Merolla et al.). Another comparison

can be made between Loihi and a 1.67-GHz Atom CPU to solve L1-minimization. Results showed that Loihi is 2.58x, 8.08x and 48.74x more energy efficient depending on the number of unknowns (Davies et al.).

Here, we fabricate stable CVD-grown Au/Ti/h-BN/Au memristors and further study their experimental properties in anticipation for their use in SNN. We develop a recursive mathematical model which follows the experimental pulsing behavior of the h-BN memristor to simulate an energy-efficient, CMOS-compatible, and hardware-friendly SNN for pattern classification. In comparison to the classification performance of 77.2% of the network trained using double-precision floating-point network parameters with 50 output neurons (Boybat et al.), we get classification accuracy of 67.5% while considering 40 output neurons. We then propose a novel STDP-based weight dropout technique to improve classification accuracy. Previous studies have shown the feasibility of SNN implementation utilizing non-2D material memristors as synaptic devices (Sanchez Esqueda et al.; Guo et al.). A recent study used an empirically extracted STDP learning rule to examine the viability of Au/Ti/h-BN/Au memristors as synapses in a SNN (Roldan et al.). Compared to this previous work, our method achieves similar classification accuracy with fewer leaky integrate and fire (LIF) output neurons. In addition, our proposed implementation complies with the experimental potentiation/depression characteristics of h-BN memristor resulting from pulses of fixed amplitude and same width. This makes our method hardware friendly as it eliminates the need for complicated pulses with different shapes/width. Finally, we show that the proposed STDP-based weight dropout strategy considerably enhances the outcomes by 12%

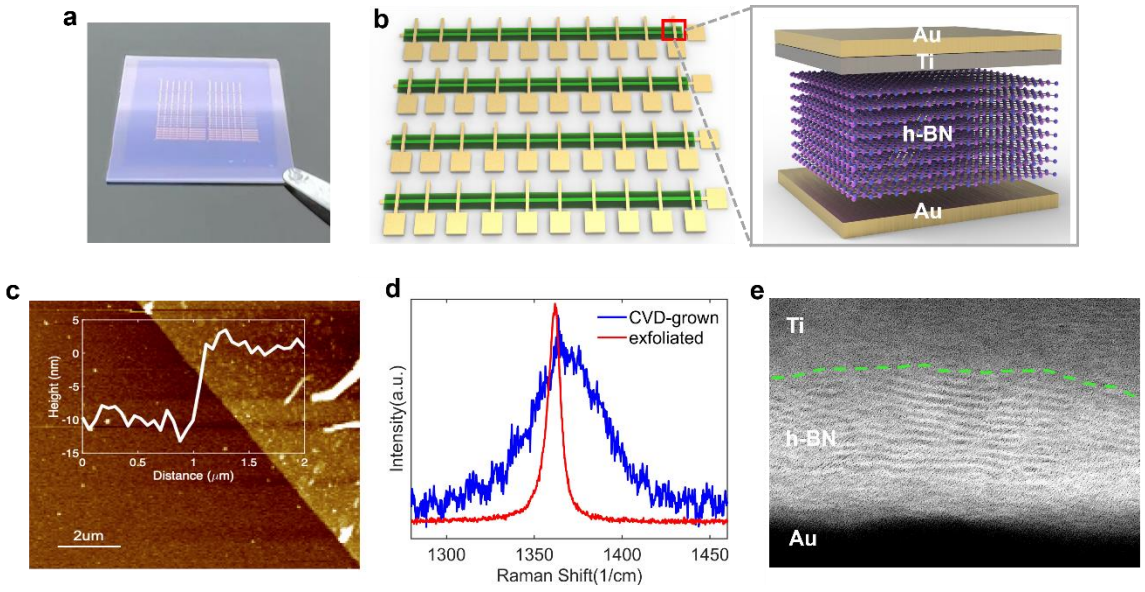


Figure 5.1. (a) The memristor arrays wafer, (b) schematic of the memristor arrays (on left) and graphical design of a single Au/h-BN/Ti memristor (on right, (c) atomic force microscope of the h-BN memristor, (d) Raman spectrum measurements, (e) cross-sectional TEM image.

reaching testing accuracy of ~80% for 40 output neurons, making our SNN more computationally efficient as well as more hardware- and power-friendly. Our study demonstrates the viability of training an adaptable SNN with memristors based on 2D materials.

5.2 Physical and Electrical Characteristics of H-BN Memristor Devices

This work uses 2D h-BN memristors with non-volatile and multi-state resistive switching characteristics. The detailed fabrication methods for these devices are found elsewhere (Xie et al.). Here, we investigate the physical and electrical properties of the device towards their application as artificial synapses in SNN. Figure 5.1a shows a picture of 2D h-BN memristor arrays fabricated on a SiO_2/Si wafer. A schematic of the memristor arrays is shown on the left side of

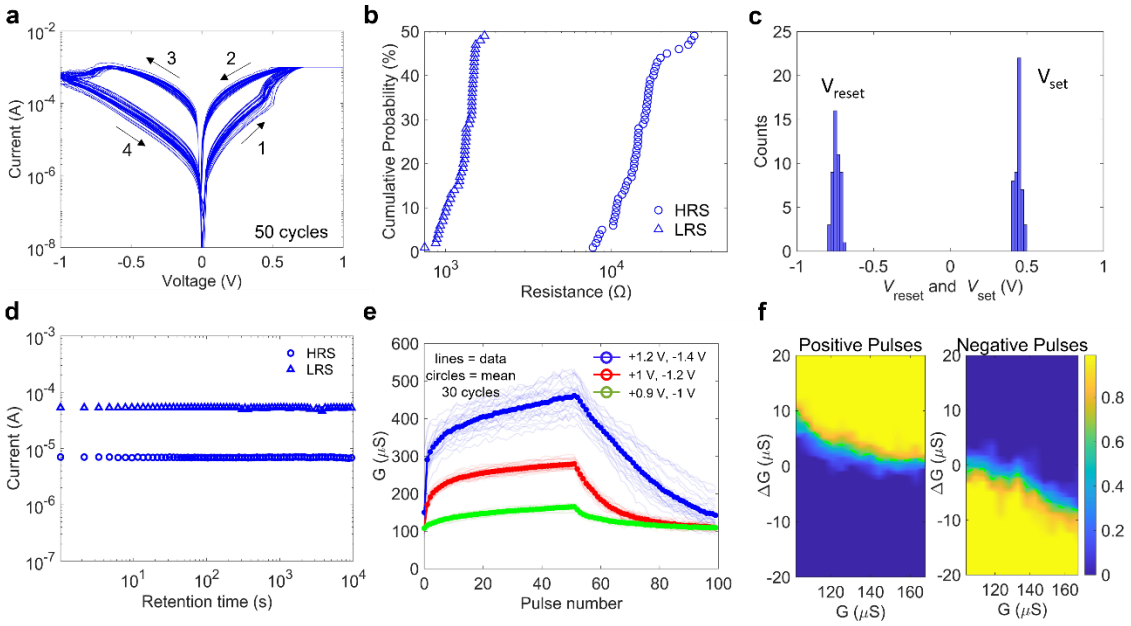


Figure 5.2. (a) I-V characteristics, (b) HRS and LRS cumulative probability, (c) histogram of set and reset voltages, (d) retention tests, (e) pulse measurement, (f) CDF plot for the green case in (e).

Figure 5.1b. The Au bottom electrode is shared by 10 devices in a row, each device having separate Ti/Au top electrode. The right side of Figure 5.1b depicts the design of a single Au/Ti/h-BN/Au memristor. The top electrode (TE) is Ti (30 nm) capped with Au (30 nm) to prevent oxidization. The resistive-switching medium is multilayer h-BN, and the bottom electrode (BE) is Au. The thickness of the multilayer h-BN is identified by atomic force microscope (AFM) as indicated in Figure 5.1c and is approximately 10 nm. The CVD-grown multilayer h-BN film is further characterized by its Raman spectrum plotted in Figure 5.1d (blue line), which shows a peak position around 1368 cm^{-1} and full width at half maximum (FWHM) of approximately 45 cm^{-1}. For comparison, an exfoliated h-BN sample is also characterized by its Raman spectrum and plotted in Figure 5.1d (red line). For the exfoliated h-BN sample, the approximate peak position and FWHM are respectively

1362 cm^{-1} and 10 cm^{-1}. CVD-grown samples reveal a broadened FWHM compared to exfoliated crystalline flakes because of random defects generated during the CVD process.

A cross-sectional TEM image is shown in Figure 5.1e identifying the multilayered h-BN structure (approximately 15 to 20 layers) and revealing defects (blurred darker regions) associated with filamentary formation of conductive paths. As proposed earlier. The CVD process promotes defects within the h-BN lattice (Wu et al.) providing potential percolation paths for metallic filament penetration. This penetration can be initiated a priori during the top electrode (TE) contact deposition process (Mao et al.) (the extent being dependent on the deposition rate) as metal atoms bombard the h-BN surface. However, the primary method for filamentary behavior is by gradual migration of active metal ions (in this work Ti) into the h-BN lattice (Mao et al.). With the application of a positive bias to the TE, metal ions can cross the metal-dielectric interface and penetrate the h-BN lattice towards the bottom electrode along boron-vacancy-rich grain boundaries (percolation paths). As the filament forms, the conductive gap between the electrodes is reduced increasing the electric field and further increasing current density (Yu and Philip Wong). We note that a second resistive switching mechanism is proposed for devices with two electrochemically inter electrodes (e.g., Pt/h-BN/Pt or Au/h-BN/Au, etc.) where to existence of vacancies (B, N, or multi-vacancies) lead to the formation of interlayer bridges (bonds) that modifying the electronic properties and conductance of the switching medium around the site of the defects (Mao et al.; Ducry et al.).

Figure 5.2a plots the experimental current-voltage (I-V) measurements of a device with a 3 μm × 3 μm active area over 50 consecutive cycles. The data shows a transition in resistance from HRS (high resistance state) to LRS (low resistance state) and the arrows label the direction of the voltage sweep and I-V characteristics. These I-V measurements were obtained by applying a sweeping voltage on the top electrode (bottom electrode grounded) while measuring current to reveal the resistive-switching effect (hysteresis in I-V curves). Here, a compliance of 10^{-3} and 10^{-2} A were activated for positive and negative applied voltages, respectively. Figure 5.2b plots the cumulative distribution of resistance (HRS and LRS) at a read voltage of 0.1 V for all 50 cycles. Figure 5.2c shows histograms of the set and reset voltages (V_{set} and V_{reset}) corresponding to transitions between HRS and LRS as extracted from the 50 cycles of DC I-V measurements. Figure 5.2d illustrates the retention (non-volatile) properties of the h-BN memristors by measurements of current as a function of time up to 10^4 seconds with negligible drift in HRS and LRS.

In addition to DC I-V, we perform pulsed voltage experiments to capture gradual changes in conductance and verify the feasibility of using h-BN memristors to emulate synaptic functions (i.e., long-term potentiation and depression). Figure 5.2e shows the pulsed programming of the h-BN memristor. By delivering a succession of positive/negative voltage pulses, we reveal analog conductance characteristics compatible with the emulation of synaptic plasticity (i.e., changes in the strength of neuron connections). We used 50 positive pulses followed by 50 negative pulses with fixed width of 100 ns and varying pulse amplitudes of +1.2 and -1.4 V for case 1 (blue line), +1 and -1.2 V for case 2 (red line), and +0.9 and -1 V

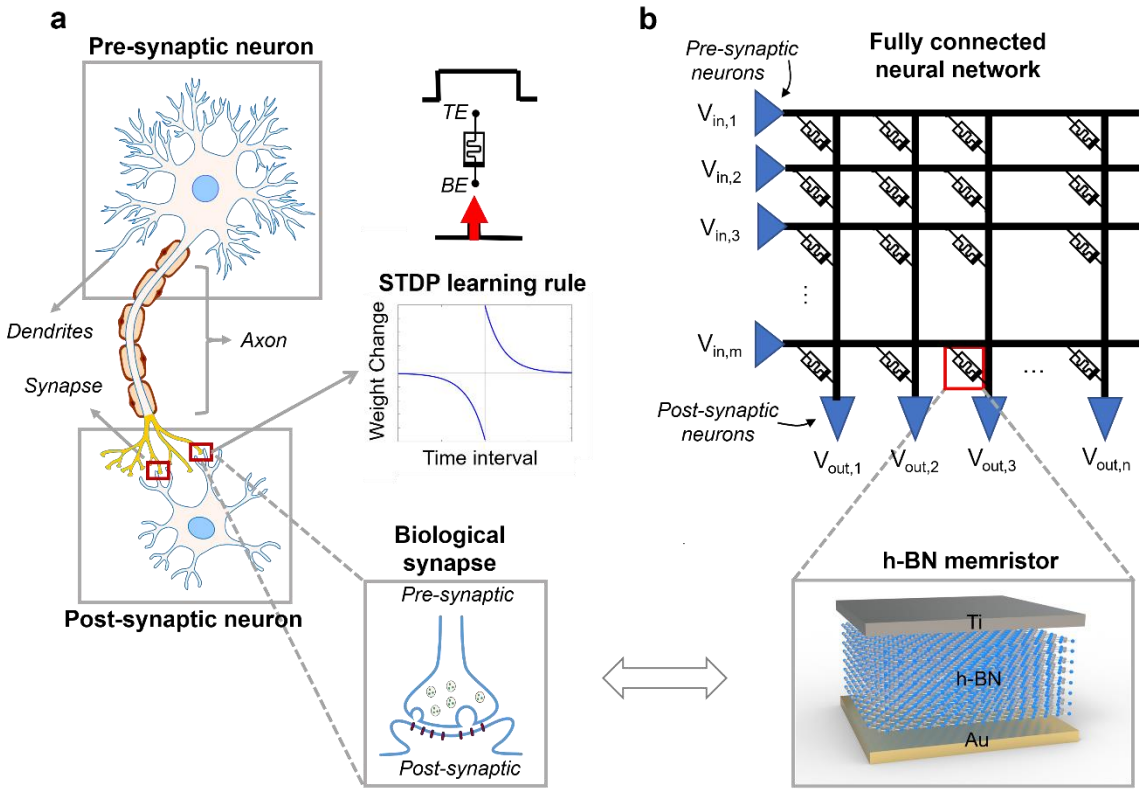


Figure 5.3. (a) An illustration of a biological neurons consisting of pre-synaptic and post-synaptic neurons, axon, and biological synapse, (b) a fully connected memristor-based artificial neural network utilizing h-BN memristor device as an artificial synapse.

for case 3 (green line) over 30 consecutive cycles each. After each pulse, a voltage of 0.1 V is applied to read current and obtain conductance. The results reveal a gradual change in conductance with each programming pulse, indicating applicability of h-BN devices as artificial synapses. Moreover, increasing the pulse amplitudes achieves larges update in conductance (ΔG) suggesting advanced synaptic functionality (e.g., tunable synaptic plasticity).

By observing the distribution of conductance updates (ΔG) as a function of conductance (G) form multiple programming/erase cycles we can better identify the variability and linearity of the pulse update scheme. This is shown in Figure 5.2f as

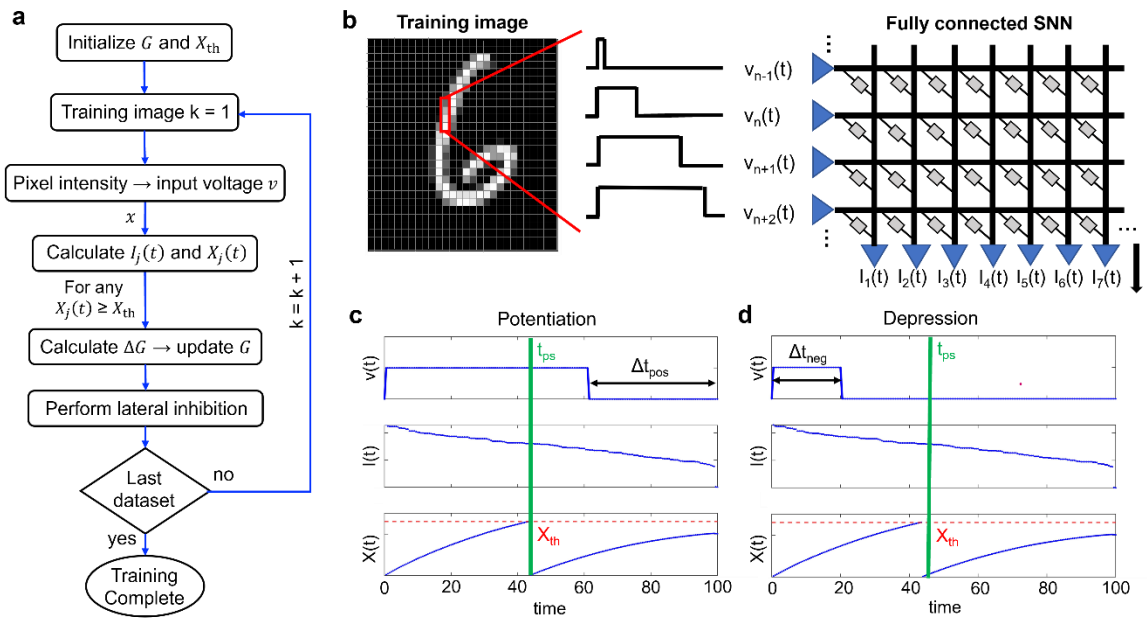


Figure 5.4. (a) Flow chart of implementation of SNN on a h-BN memristor crossbar, (b) demonstration of two-layer SNN, (c),(d) output current accumulation and charge integration for an input voltage larger and smaller than t_{ps}.

contours of the cumulative distribution function (CDF) of ΔG versus G over 30 cycles (for positive and negative pulses). Note that the CDF plots correspond to the data in Figure 5.2e (green) for pulse amplitudes of +0.9 and -1 V. In Figure 5.2f, the green dashed line traces the midpoint in the distribution (i.e., the value of 0.5). For a perfectly linear device, the midpoint line should remain constant as a function of G, and the transition through the midpoint would be abrupt in the absence of variation. Here we observe a minor change in the distribution midpoint with G indicating good linearity for both positive (potentiation) and negative (depression) pulses. Moreover, we verify that cycle-to-cycle variability is small, as illustrated by a short range in the distribution of ΔG transitioning through the midpoint (abrupt change in contour plot).

5.3 Implementation of Unsupervised Learning in h-BN Memristor-Based Spiking Neural Network

This section describes the implementation of our SNN model based on experimental data from individual h-BN memristors. The SNN architecture consists of two fully connected layers: 784 input neurons and 40 output neurons. For the output neurons, we consider a commonly used spiking neuron model: the leaky integrate-and-fire (LIF) model. In a circuit implementation of the LIF model, an RC circuit with a threshold acts as integrator of synaptic signal inputs (Datta Sahoo). The accumulated (integrated) signal is compared against a threshold reference and will activate an output spike production circuit if the threshold is achieved. Figure 5.4a shows a flowchart for the simulated crossbar implementation of the SNN. The simulation conducts unsupervised learning to classify the Modified National Institute of Standards and Technology (MNIST) handwritten digit dataset. This dataset consists of 60,000 training images of handwritten digits and 10,000 separate testing images. In our implementation, SNN training is implemented through feedforward/feedback modes. Both make use of experimental h-BN memristor data (DC I-V and pulsed data) to simulate accumulated currents (based on Ohm's law and Kirchhoff's law) and to update synaptic weights (conductance updates).

Phase 1: Feedforward mode (current and charge integration)

Initially, the pixel intensities of two-dimensional grey-scale input training images (28-by-28 pixels) are translated to one-dimensional temporal voltages (784 input voltage pulses). Each voltage pulse has fixed amplitude of V_r = 0.1 V and

different widths (t_{width}) ranging between 0 and 100 ms corresponding to pixel intensity. A black pixel corresponds to a voltage pulse with minimum t_{width} (minimum intensity), a white pixel corresponds to $t_{width} =$ 100 ms (maximum intensity), and any other pixel intensity translates to pulse widths between 0 and 100 ms. The input voltages, $v_i(t)$, are applied to the h-BN memristor crossbar and the accumulated currents at the bottom electrodes are calculated at every time step. Figure 5.4b depicts this procedure graphically. The post-synaptic currents at each column (indexed with j) are obtained based on Kirchhoff's law as $I_j(t) = \sum_i v_i(t)G_{ij}$, where G_{ij} are the adjustable h-BN memristor conductances and $v_i(t)$ are the input voltages at each row in the crossbar. Mathematically, the output current vector results from the multiplication of the input voltage vector and the matrix of memristor conductances (vector-matrix multiplication or VMM). In the crossbar architecture, VMM can be computed with a single read operation (parallel computation). The SNN simulation follows with the calculation of accumulated charge at the output LIF neurons based on

$$\tau_{RC}\frac{dX_j(t)}{dt} - X_j(t) = I_j(t). \quad (1)$$

In (1), $I_j(t)$ is the current in neuron j at time t and $X_j(t)$ is the accumulated charge. Here, τ_{RC} is the time constant associated with the LIF circuit. Reaching a predetermined threshold (X_{th} = 10 mC) at any of the output neuron will trigger the

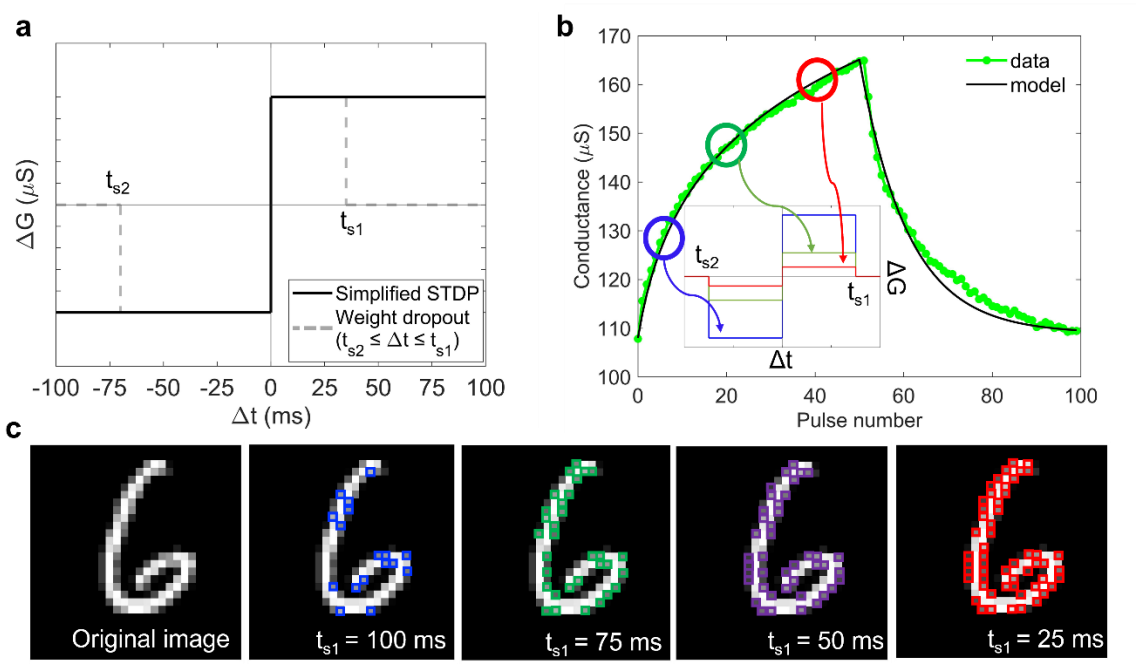


Figure 5.5. (a) Simplified STDP vs. STDP-based weight dropout, (b) mathematical model fit to experimental pulsed h-BN memristor data, (c) updates performed during learning for STDP-based weight dropout rule with different positive time filters.

firing of a post-synaptic spike. In Figures 5.4c, d, t_{ps} denotes the triggering time of the post-synaptic spike. Through lateral inhibition pathways, the output spike propagates among other LIF output neurons to prevent them from firing at the same time. In our implementation, following a post-synaptic spike, the charge at every output neuron is reset to an initial condition (X_i = 0) and held there for a fixed time (t_{inh} = 10 ms) except for the neuron that recently fired which can immediately return to accumulating charge. This competitive learning model where neurons can inhibit each other is known as winner-takes-all (WTA) (Datta Sahoo). WTA is thought to be a basic component of cognitive tasks including attention and object recognition (Maass). All synaptic connections to the neuron that fired will be adjusted. In hardware, this means updating the conductance of memristors from a specific

column connected to the neuron that fired. We must consider two cases of synaptic plasticity: 1) strengthening the connection (potentiation) to inputs that contribute to the firing (inputs that were active at the time of the post-synaptic spike, $t_{width} > t_{ps}$); 2) weakening the connection (depression) for inputs that contribute less (inputs that were inactive at the time of the post-synaptic spike, $t_{width} < t_{ps}$). Figures 5c, d respectively illustrate examples of potentiation and depression with plots of input voltage (top), output current at the post-synaptic neuron (middle), and accumulated charge (bottom) during a 100 ms timeframe (single training step). Synaptic weight update is discussed next.

Phase 2: Feedback mode (synapse update)

We use a simplified learning rule to update synapse weights which follows the experimental pulsing behavior of h-BN memristor. In our simplified hardware-friendly STDP implementation, ΔG will be either positive or negative based on the temporal correlation of corresponding input voltage pulse widths (t_{width}) and post-synaptic spikes (t_{ps}), and the magnitude will be modeled to simulate h-BN memristor pulsed characteristics (see Figure 5.5a). In other words, a single or set of consecutive positive (negative) voltage pulses are applied to memristors that require potentiation (depression). The change in h-BN memristor conductance follows an experimentally verified recursive model given by

$$\Delta G = a_p + b_p e^{-c_p \frac{G - G_{min}}{G_{max} - G_{min}}} \qquad \textit{Potentiation} \qquad (2a)$$

$$\Delta G = a_d + b_d e^{-c_d \frac{G_{max} - G}{G_{max} - G_{min}}}, \qquad \textit{Depression} \qquad (2b)$$

where $a_p, a_d, b_p, b_d, c_p, c_d$ are fitting parameters and G_{max}, G_{min} correspond to the maximum and minimum experimental conductances respectively. Figure 5.5b shows the model fit to experimental data with fitting parameters values of 10^{-10}, 10^{-4}, 5×10^{-6}, -10^{-4}, 2.5, 0.05 for a_p, a_d, b_p, b_d, c_p and c_d respectively. The conductance is bounded to $G_{min}\ and\ G_{max}$ which are measured at 108 and 165 µS for the h-BN memristor with a 3 × 3 µm active area. The inset in Figure 5.5b emphasizes how the simplified STDP approach introduces non-ideal (non-linear) h-BN memristor behavior into our SNN simulation. For example, when G is in the lower end, the change in conductance with the application of a single pulse (positive or negative) is larger compared to when G is in towards the upper end. In addition, the simulation incorporates homeostatic regulation to maintain similar firing rates for all neurons by making small adjustments to the firing thresholds as given by $\Delta X_{th} = \gamma(f_r - 1/N)$. Here, γ is a threshold update fitting factor (set to 5 µC), f_r is the firing rate, and N is the number of output neurons. By adjusting γ, recognition and convergence rate changes. It is important to adjust X_{th} and δ to maintain reasonable firing rates and to avoid overfitting in the learning SNN unsupervised learning process.

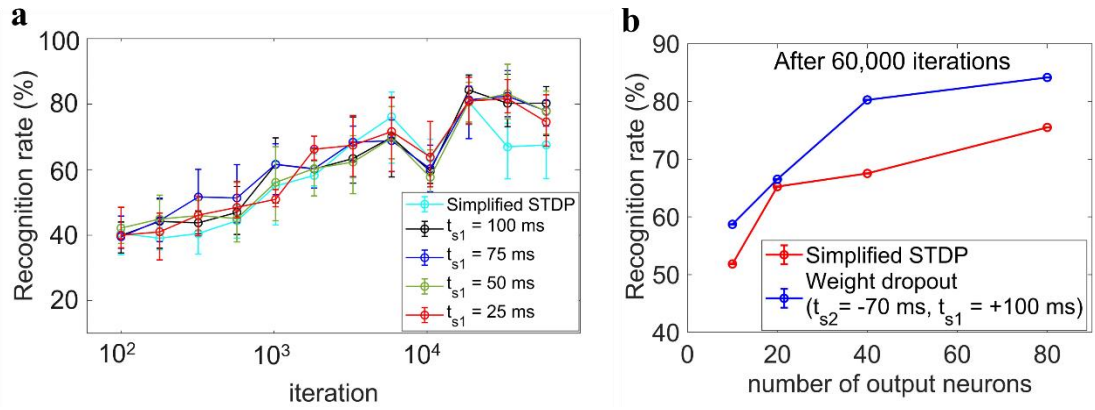


Figure 5.6. (a) Comparison of recognition rate as a function of iteration for simplified STDP vs. STDP-based weight dropout with 4 different positive time, (b) comparison of recognition rate vs. output neurons for simplified STDP vs. STDP-based weight dropout.

To improve the effectiveness of neural transmissions, excess neurons and synaptic connections are removed through a process known as synaptic weight dropout. Synapses connecting neurons with high spiking correlation are preserved, while synapses with poor or uncorrelated spiking activity are pruned. Weight dropout also mitigates overfitting in neural networks trained with large size data sets by preventing unwanted specialization towards details and noise in the training data and allowing better generalization (Faghihi et al.).

We demonstrate an STDP-compatible technique to prune (remove) insignificant weights for an improved network performance in terms of classification accuracy. This technique applies a time filter on the temporal correlation between input voltage pulse widths and post-synaptic spikes (i.e., Δt), to limit the number of conductances that will be updated in the feedback phase. The process is as follows. First, Δt is calculated as

$$\Delta t_{neg} = t_{width}\ (ms) - 0\ (ms) \ldots \quad \text{for } t_{width} < t_{ps} \quad \text{(4a)}$$

$$\Delta t_{pos} = 100\ (ms) - t_{width}\ (ms), \ldots \text{for } t_{width} \geq t_{ps} \quad (4b)$$

where t_{width} denotes the input DC voltage width, t_{ps} denotes the post-spike time (see Figure 5.4c, d). Next, the calculated Δt is normalized to fall within the STDP range (+/- 100 ms). As shown in Figure 5.5a, we define t_{s1} as the positive (potentiation) time filter and t_{s2} as the negative (depression) time filter. All the synapses with Δt between t_{s2} and t_{s1} are subject to a conductance update determined by the experimentally verified recursive model in equation (2). We have performed simulations for t_{s2} = -70 ms and t_{s1} = 100, 75, 50 and 25 ms. In Figure 5.5c, the same grayscale MNIST image (out of 60,000 training images) is shown with colored pixel outlines indicating the synapses that were dropped (not updated) with various values of the time filters t_{s2} = -70 ms and t_{s1} = 100, 75, 50 and 25 ms). Evidently, the number of pruned (dropped) conductances is largest for t_{s2} = -70 ms and t_{s1} = 25 ms. Also, the figure labeled "original image" represents the case without weight dropout. This implies that in simplified STDP learning approach no weight is eliminated, and during each iteration all the corresponding synapses are updated.

Figure 5.6 summarizes the results from the SNN simulations using simplified STDP as well as STDP-based weight dropout learning schemes. In Figure 5.6a, the recognition rate for arrays with 40 output neurons is plotted as a function of training number for both cases (different time filters shown for weight dropout). For each case, the simulation is conducted five times, and the mean value is plotted with the length of error bars indicating the standard deviation. Since the labels of training images are not known to the network (training is unsupervised), recognition rate is based on the spiking activity for all 10,000 test images of MNIST dataset. Each

neuron is assigned to the handwritten digit for which it spiked the most, and the ratio of spikes on the assigned digit to the total number of spikes is calculated as the recognition rate. The recognition rate shown in Figure 5.6a is the average of all the handwritten digits in MNIST dataset. As observed in Figure 5.6a, the final recognition rate (after 60,000 training steps) for the STDP-based weight dropout rule of t_{s2} = -70 ms and t_{s1} = 100 ms (in solid black line) has improved by 12% for 40 output neurons reaching 80.2% compared to the simplified STDP method recognition rate of 67.5%. Figure 5.6b shows the results of the simulations for 10, 20, 40 and 80 output neurons when trained with recursive model-based simplified STDP (red line) and STDP-based weight dropout with time filters of t_{s2}= -70 ms and t_{s1} = 100 ms (blue line). Plots are the average over five simulation cycles for each case. The results shown for each number of output neurons are the recognition rate after training with one single epoch (i.e., 60,000 training images). Improved recognition rate is observed when the proposed STDP-based weight dropout technique is applied. The improvement is attributed to alleviating overfitting in the SNN, as the improvement appears more significantly towards the end of the training epoch.

CHAPTER 6

CONCLUSION

This dissertation offers an outline of neuromorphic computing as a remedy for high power consumption in modern digital computers and as an efficient platform for AI applications that are integrated into our daily lives. The dissertation then gives an overview of memristor, a sort of analog resistive switching memory with excellent properties, a promising candidate for artificial synapses in neuromorphic computing. Additionally, 2D devices exhibit resistive switching properties that can be reduced to a single layer and exhibit even superior properties, such as improved energy efficiency (see Figure 3.1).

Chapter 2 presents the circuit-level analysis of the 1T1R crossbar implementation of the linear and logistic regression algorithms using a compact model for memristors that was physics-based, variation-aware, and experimentally proven. The analysis includes the impact of device variability on convergence, as well as on prediction/classification accuracy and precision. The algorithm implementations are based on crossbar vector matrix multiplication, which is the core operation of typical neuromorphic computing platforms. This chapter also offers an enhanced gradient-descent strategy that works with real-world hardware. With this method, a faster initial convergence rate can be attained without sacrificing the excellent prediction accuracy. The findings of this study suggest that our suggested smart pulsing technique can be modified to accelerate training in real crossbar architectures. The following was the result of our analysis of how memristor variability affects algorithm performance: Memristor variability does not seem to significantly influence prediction accuracy in linear regression (can still attain high accuracy), but convergence rate and precision are noticeably degraded. Variations in the

prediction error as a function of training steps show degraded precision. Similar to logistic regression, slower convergence rates and fluctuations in error as a function of algorithm iteration (effect on precision) were noted. However, classification accuracy was not significantly impacted by memristor variability in logistic regression. Additionally, we have contrasted our suggested pulsing strategy with earlier approaches that applied a single positive or negative pulse depending on the sign of the needed update at each iteration. The suggested method classifies noisy binary images more accurately and more quickly even when memristor fluctuation is present. The results of this study are crucial for understanding how device variability affects algorithm performance and memristor crossbar viability for prediction and classification tasks.

In chapter 3, we performed hardware-level implementation and simulations of ML Algorithms Using Novel 2D Material, h-BN memristor. We demonstrate the implementation of dot-product operations on h-BN memristor arrays showing excellent linearity and reproducibility. Then, we demonstrate a hardware-compatible implementation of stochastic logistic regression on h-BN memristor arrays for image classification. The experimental results show classification accuracy and algorithm performance comparable to arrays with ideal memristive behavior (from simulations). Exceptional resistive switching characteristics, dot-product performance, and implementation of logistic regression in h-BN memristor arrays indicate a significant step towards the integration of 2D materials for next-generation neuromorphic computing systems.

Next, in chapter 4, we performed hardware-level implementation of multivariable stochastic linear regression on the h-BN memristor arrays using a dataset available online.

We proposed a simplified hardware-compatible stochastic linear regression approach where the memristor conductances (i.e., the model parameters) are updated through the application of a single programming pulse, and the polarity of the pulse is determined by the sign of the corresponding ΔG. The results showed that the model is trained to perfectly fit the training data.

Moreover, we have reported the synaptic characteristics of 2D Au/h-BN/Ti memristors for spiking neural network neuromorphic application in chapter 5. The devices exhibit advanced synaptic functionality such as a larger dynamic range with increased pulse amplitude, good linearity for both potentiation and depression, and small cycle-to-cycle variability. Simulation results for MNIST pattern classification based on Au/h-BN/Ti memristive SNN hardware following the experimental pulsing behavior of memristor reaches satisfactory recognition rate of 67.5% for 40 output neurons. The recognition rate improved as we increased the number of output neurons. We then proposed a STDP-based pruning technique to improve the recognition rate to 80% for 40 output neurons by improving the overfitting issue observed in our system. Our work is a step towards the deployment of real 2D materials in SNN hardware for training and inference applications.

REFERENCES

Afshari, Sahra, et al. “Analyzing the Impact of Memristor Variability on Crossbar Implementation of Regression Algorithms With Smart Weight Update Pulsing Techniques.” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 5, May 2022, pp. 2025–34, https://doi.org/10.1109/TCSI.2022.3144240.

Ambrosi, Elia, et al. “Impact of Oxide and Electrode Materials on the Switching Characteristics of Oxide ReRAM Devices.” *Faraday Discussions*, vol. 213, 2019, pp. 87–98, https://doi.org/10.1039/C8FD00106E.

Arizona State University, Predictive Technology Model (Ptm).

Baroni, Andrea, et al. “Low Conductance State Drift Characterization and Mitigation in Resistive Switching Memories (RRAM) for Artificial Neural Networks.” *IEEE Transactions on Device and Materials Reliability*, vol. 22, no. 3, Sept. 2022, pp. 340–47, https://doi.org/10.1109/TDMR.2022.3182133.

Basu, Nilanjan, et al. “Large Area Few-Layer Hexagonal Boron Nitride as a Raman Enhancement Material.” *Nanomaterials*, vol. 11, no. 3, Mar. 2021, p. 622, https://doi.org/10.3390/nano11030622.

Belmonte, Attilio, et al. “Voltage-Controlled Reverse Filament Growth Boosts Resistive Switching Memory.” *Nano Research*, vol. 11, no. 8, Aug. 2018, pp. 4017–25, https://doi.org/10.1007/s12274-018-1983-2.

“Beyond von Neumann.” *Nature Nanotechnology*, vol. 15, no. 7, July 2020, pp. 507–507, https://doi.org/10.1038/s41565-020-0738-x.

Boybat, Irem, et al. “Neuromorphic Computing with Multi-Memristive Synapses.” *Nature Communications*, vol. 9, no. 1, June 2018, p. 2514, https://doi.org/10.1038/s41467-018-04933-y.

Burr, Geoffrey W., et al. “Emerging Materials in Neuromorphic Computing: Guest Editorial.” *APL Materials*, vol. 8, no. 1, Jan. 2020, p. 010401, https://doi.org/10.1063/1.5143659.

Chang, Meng-Fan, et al. “Challenges and Circuit Techniques for Energy-Efficient On-Chip Nonvolatile Memory Using Memristive Devices.” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 5, no. 2, June 2015, pp. 183–93, https://doi.org/10.1109/JETCAS.2015.2426531.

Chaudhuri, Arjun, and Krishnendu Chakrabarty. “Analysis of Process Variations, Defects, and Design-Induced Coupling in Memristors.” *2018 IEEE International Test Conference (ITC)*, IEEE, 2018, pp. 1–10, https://doi.org/10.1109/TEST.2018.8624819.

Chen, An, and Ming-Ren Lin. “Variability of Resistive Switching Memories and Its Impact on Crossbar Array Performance.” *2011 International Reliability Physics Symposium*, IEEE, 2011, p. MY.7.1-MY.7.4, https://doi.org/10.1109/IRPS.2011.5784590.

Chen, Pai-Yu, and Shimeng Yu. “Compact Modeling of RRAM Devices and Its Applications in 1T1R and 1S1R Array Design.” *IEEE Transactions on Electron Devices*, vol. 62, no. 12, Dec. 2015, pp. 4022–28, https://doi.org/10.1109/TED.2015.2492421.

Chen, Shaochuan, et al. “Wafer-Scale Integration of Two-Dimensional Materials in High-Density Memristive Crossbar Arrays for Artificial Neural Networks.” *Nature Electronics*, vol. 3, no. 10, Oct. 2020, pp. 638–45, https://doi.org/10.1038/s41928-020-00473-w.

Chen, Yang Yin, et al. “Improvement of Data Retention in HfO<inf>2</Inf>/Hf 1T1R RRAM Cell under Low Operating Current.” *2013 IEEE International Electron Devices Meeting*, IEEE, 2013, pp. 10.1.1-10.1.4, https://doi.org/10.1109/IEDM.2013.6724598.

Chen, Ying-Chen, et al. “Analog Synaptic Behaviors in Carbon-Based Self-Selective RRAM for In-Memory Supervised Learning.” *2021 IEEE 71st Electronic Components and Technology Conference (ECTC)*, IEEE, 2021, pp. 1645–51, https://doi.org/10.1109/ECTC32696.2021.00261.

Chen, Yu-Hsin, et al. “Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks.” *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, Jan. 2017, pp. 127–38, https://doi.org/10.1109/JSSC.2016.2616357.

Christensen, Dennis V, et al. “2022 Roadmap on Neuromorphic Computing and Engineering.” *Neuromorphic Computing and Engineering*, vol. 2, no. 2, June 2022, p. 022501, https://doi.org/10.1088/2634-4386/ac4a83.

Chua, L. “Memristor-The Missing Circuit Element.” *IEEE Transactions on Circuit Theory*, vol. 18, no. 5, 1971, pp. 507–19, https://doi.org/10.1109/TCT.1971.1083337.

Datta Sahoo, Bibhu. “Ring Oscillator Based Sub-1V Leaky Integrate-and-Fire Neuron Circuit.” *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2017, pp. 1–4, https://doi.org/10.1109/ISCAS.2017.8050980.

Davies, Mike, et al. “Loihi: A Neuromorphic Manycore Processor with On-Chip Learning.” *IEEE Micro*, vol. 38, no. 1, Jan. 2018, pp. 82–99, https://doi.org/10.1109/MM.2018.112130359.

Degraeve, R., et al. “Causes and Consequences of the Stochastic Aspect of Filamentary RRAM.” *Microelectronic Engineering*, vol. 147, Nov. 2015, pp. 171–75, https://doi.org/10.1016/j.mee.2015.04.025.

del Valle, Javier, et al. “Challenges in Materials and Devices for Resistive-Switching-Based Neuromorphic Computing.” *Journal of Applied Physics*, vol. 124, no. 21, Dec. 2018, p. 211101, https://doi.org/10.1063/1.5047800.

Ducry, Fabian, et al. “An Ab Initio Study on Resistance Switching in Hexagonal Boron Nitride.” *Npj 2D Materials and Applications*, vol. 6, no. 1, Sept. 2022, p. 58, https://doi.org/10.1038/s41699-022-00340-6.

Faghihi, Faramarz, et al. “A Synaptic Pruning-Based Spiking Neural Network for Hand-Written Digits Classification.” *Frontiers in Artificial Intelligence*, vol. 5, Feb. 2022, https://doi.org/10.3389/frai.2022.680165.

Fantini, A., et al. “Intrinsic Switching Variability in HfO<inf>2</Inf> RRAM.” *2013 5th IEEE International Memory Workshop*, IEEE, 2013, pp. 30–33, https://doi.org/10.1109/IMW.2013.6582090.

Farhan. “50 Startups.” *Https://Www.Kaggle.Com/Datasets/Farhanmd29/50-Startups* , 2022.

Fey, Dietmar. “Memristors Divide to Conquer Device Variability.” *Nature Electronics*, vol. 1, no. 8, Aug. 2018, pp. 438–39, https://doi.org/10.1038/s41928-018-0123-z.

Ge, Jun, et al. “A Sub-500 MV Monolayer Hexagonal Boron Nitride Based Memory Device.” *Materials & Design*, vol. 198, Jan. 2021, p. 109366, https://doi.org/10.1016/j.matdes.2020.109366.

Ge, Ruijing, et al. “Atomristor: Nonvolatile Resistance Switching in Atomic Sheets of Transition Metal Dichalcogenides.” *Nano Letters*, vol. 18, no. 1, Jan. 2018, pp. 434–41, https://doi.org/10.1021/acs.nanolett.7b04342.

Gokmen, Tayfun, and Yurii Vlasov. “Acceleration of Deep Neural Network Training with Resistive Cross-Point Devices: Design Considerations.” *Frontiers in Neuroscience*, vol. 10, July 2016, https://doi.org/10.3389/fnins.2016.00333.

Goux, Ludovic, et al. “Key Material Parameters Driving CBRAM Device Performances.” *Faraday Discussions*, vol. 213, 2019, pp. 67–85, https://doi.org/10.1039/C8FD00115D.

Guo, Yilong, et al. “Unsupervised Learning on Resistive Memory Array Based Spiking Neural Networks.” *Frontiers in Neuroscience*, vol. 13, Aug. 2019, https://doi.org/10.3389/fnins.2019.00812.

Guy, J., et al. “Guidance to Reliability Improvement in CBRAM Using Advanced KMC Modelling.” *2017 IEEE International Reliability Physics Symposium (IRPS)*, IEEE, 2017, p. PM-2.1-PM-2.5, https://doi.org/10.1109/IRPS.2017.7936384.

Hong, XiaoLiang, et al. “A Novel Geometry of ECM-Based RRAM with Improved Variability.” *Journal of Physics D: Applied Physics*, May 2018, https://doi.org/10.1088/1361-6463/aac2b4.

Hosseininoorbin, Seyedehfaezeh, et al. *Exploring Deep Neural Networks on Edge TPU*. Oct. 2021.

Hu, Miao, et al. “Dot-Product Engine for Neuromorphic Computing.” *Proceedings of the 53rd Annual Design Automation Conference*, ACM, 2016, pp. 1–6, https://doi.org/10.1145/2897937.2898010.

Huh, Woong, et al. “Memristors Based on 2D Materials as an Artificial Synapse for Neuromorphic Electronics.” *Advanced Materials*, vol. 32, no. 51, Dec. 2020, p. 2002092, https://doi.org/10.1002/adma.202002092.

Huyghebaert, C., et al. “2D Materials: Roadmap to CMOS Integration.” *2018 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2018, pp. 22.1.1-22.1.4, https://doi.org/10.1109/IEDM.2018.8614679.

Ielmini, Daniele. “Resistive Switching Memories Based on Metal Oxides: Mechanisms, Reliability and Scaling.” *Semiconductor Science and Technology*, vol. 31, no. 6, June 2016, p. 063002, https://doi.org/10.1088/0268-1242/31/6/063002.

Indiveri, Giacomo, et al. “Neuromorphic Silicon Neuron Circuits.” *Frontiers in Neuroscience*, vol. 5, 2011, https://doi.org/10.3389/fnins.2011.00073.

Jiménez-Luna, José, et al. “Drug Discovery with Explainable Artificial Intelligence.” *Nature Machine Intelligence*, vol. 2, no. 10, Oct. 2020, pp. 573–84, https://doi.org/10.1038/s42256-020-00236-4.

Jo, Sung Hyun, et al. “Nanoscale Memristor Device as Synapse in Neuromorphic Systems.” *Nano Letters*, vol. 10, no. 4, Apr. 2010, pp. 1297–301, https://doi.org/10.1021/nl904092h.

Khan, Amjad Rehman. “Facial Emotion Recognition Using Conventional Machine Learning and Deep Learning Methods: Current Achievements, Analysis and Remaining Challenges.” *Information*, vol. 13, no. 6, May 2022, p. 268, https://doi.org/10.3390/info13060268.

Kumar, Pratik, et al. “Hybrid Architecture Based on Two-Dimensional Memristor Crossbar Array and CMOS Integrated Circuit for Edge Computing.” *Npj 2D Materials and Applications*, vol. 6, no. 1, Jan. 2022, p. 8, https://doi.org/10.1038/s41699-021-00284-3.

Lanza, Mario, et al. “Resistive Switching Crossbar Arrays Based on Layered Materials.” *Advanced Materials*, vol. 35, no. 9, Mar. 2023, p. 2205402, https://doi.org/10.1002/adma.202205402.

Lee, Myoung-Jae, et al. “A Fast, High-Endurance and Scalable Non-Volatile Memory Device Made from Asymmetric Ta2O5−x/TaO2−x Bilayer Structures.” *Nature Materials*, vol. 10, no. 8, Aug. 2011, pp. 625–30, https://doi.org/10.1038/nmat3070.

Lemme, Max C., et al. “2D Materials for Future Heterogeneous Electronics.” *Nature Communications*, vol. 13, no. 1, Mar. 2022, p. 1392, https://doi.org/10.1038/s41467-022-29001-4.

Li, Bing, et al. “An Overview of In-Memory Processing with Emerging Non-Volatile Memory for Data-Intensive Applications.” *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, ACM, 2019, pp. 381–86, https://doi.org/10.1145/3299874.3319452.

Li, Yibo, et al. “Review of Memristor Devices in Neuromorphic Computing: Materials Sciences and Device Challenges.” *Journal of Physics D: Applied Physics*, vol. 51, no. 50, Dec. 2018, p. 503002, https://doi.org/10.1088/1361-6463/aade3f.

Li, YingTao, et al. “An Overview of Resistive Random Access Memory Devices.” *Chinese Science Bulletin*, vol. 56, no. 28–29, Oct. 2011, pp. 3072–78, https://doi.org/10.1007/s11434-011-4671-0.

Lillicrap, Timothy P., et al. “Backpropagation and the Brain.” *Nature Reviews Neuroscience*, vol. 21, no. 6, June 2020, pp. 335–46, https://doi.org/10.1038/s41583-020-0277-3.

Maass, Wolfgang. “On the Computational Power of Winner-Take-All.” *Neural Computation*, vol. 12, no. 11, Nov. 2000, pp. 2519–35, https://doi.org/10.1162/089976600300014827.

Mahadevaiah, M. K., et al. “Reliability of CMOS Integrated Memristive HfO2 Arrays with Respect to Neuromorphic Computing.” *2019 IEEE International Reliability Physics Symposium (IRPS)*, IEEE, 2019, pp. 1–4, https://doi.org/10.1109/IRPS.2019.8720552.

Mahmoodi, M. R., et al. “Versatile Stochastic Dot Product Circuits Based on Nonvolatile Memories for High Performance Neurocomputing and Neurooptimization.” *Nature Communications*, vol. 10, no. 1, Nov. 2019, p. 5113, https://doi.org/10.1038/s41467-019-13103-7.

Mao, Jing-Yu, et al. “A van Der Waals Integrated Damage-Free Memristor Based on Layered 2D Hexagonal Boron Nitride.” *Small*, vol. 18, no. 12, Mar. 2022, p. 2106253, https://doi.org/10.1002/smll.202106253.

Mbarek, Khaoula, et al. “On the Design and Analysis of a Compact Array with 1T1R RRAM Memory Element.” *Analog Integrated Circuits and Signal Processing*, vol. 102, no. 1, Jan. 2020, pp. 27–37, https://doi.org/10.1007/s10470-019-01488-w.

Mehonic, A., and A. J. Kenyon. “Brain-Inspired Computing Needs a Master Plan.” *Nature*, vol. 604, no. 7905, Apr. 2022, pp. 255–60, https://doi.org/10.1038/s41586-021-04362-w.

Meng, Zhenzhu, et al. “Using a Data Driven Approach to Predict Waves Generated by Gravity Driven Mass Flows.” *Water*, vol. 12, no. 2, Feb. 2020, p. 600, https://doi.org/10.3390/w12020600.

Merolla, Paul A., et al. “A Million Spiking-Neuron Integrated Circuit with a Scalable Communication Network and Interface.” *Science*, vol. 345, no. 6197, Aug. 2014, pp. 668–73, https://doi.org/10.1126/science.1254642.

Merrikh-Bayat, Farnood, et al. “High-Performance Mixed-Signal Neurocomputing With Nanoscale Floating-Gate Memory Cell Arrays.” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, Oct. 2018, pp. 4782–90, https://doi.org/10.1109/TNNLS.2017.2778940.

Milo, V., et al. “Multilevel HfO_2 -Based RRAM Devices for Low-Power Neuromorphic Networks.” *APL Materials*, vol. 7, no. 8, Aug. 2019, p. 081120, https://doi.org/10.1063/1.5108650.

Montesinos López, Osval Antonio, et al. *Multivariate Statistical Machine Learning Methods for Genomic Prediction*. Springer International Publishing, 2022, https://doi.org/10.1007/978-3-030-89010-0.

Musisi-Nkambwe, Mirembe, et al. “The Viability of Analog-Based Accelerators for Neuromorphic Computing: A Survey.” *Neuromorphic Computing and Engineering*, vol. 1, no. 1, Sept. 2021, p. 012001, https://doi.org/10.1088/2634-4386/ac0242.

Nair, Manu V, and Piotr Dudek. “Gradient-Descent-Based Learning in Memristive Crossbar Arrays.” *2015 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2015, pp. 1–7, https://doi.org/10.1109/IJCNN.2015.7280658.

Nigrin, Albert. *Neural Networks for Pattern Recognition*. The MIT Press, 1993, https://doi.org/10.7551/mitpress/4923.001.0001.

Niharika*, and Sona Malhotra. “Sentiment Analysis Using Artificial Neural Network.” *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 5, Jan. 2020, pp. 3267–73, https://doi.org/10.35940/ijrte.E6450.018520.

Nikam, Revannath Dnyandeo, et al. “Single-Atom Quantum-Point Contact Switch Using Atomically Thin Hexagonal Boron Nitride.” *Small*, vol. 17, no. 7, Feb. 2021, p. 2006760, https://doi.org/10.1002/smll.202006760.

Pan, Chengbin, et al. “Coexistence of Grain-Boundaries-Assisted Bipolar and Threshold Resistive Switching in Multilayer Hexagonal Boron Nitride.” *Advanced Functional Materials*, vol. 27, no. 10, Mar. 2017, p. 1604811, https://doi.org/10.1002/adfm.201604811.

Pan, F., et al. “Recent Progress in Resistive Random Access Memories: Materials, Switching Mechanisms, and Performance.” *Materials Science and Engineering: R: Reports*, vol. 83, Sept. 2014, pp. 1–59, https://doi.org/10.1016/j.mser.2014.06.002.

Patel, Ravi, et al. “Multistate Register Based on Resistive RAM.” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 9, Sept. 2015, pp. 1750–59, https://doi.org/10.1109/TVLSI.2014.2347926.

Pradhan, Sangram K., et al. “Resistive Switching Behavior of Reduced Graphene Oxide Memory Cells for Low Power Nonvolatile Device Application.” *Scientific Reports*, vol. 6, no. 1, May 2016, p. 26763, https://doi.org/10.1038/srep26763.

Prakash, Amit, et al. “TaO x -Based Resistive Switching Memories: Prospective and Challenges.” *Nanoscale Research Letters*, vol. 8, no. 1, Dec. 2013, p. 418, https://doi.org/10.1186/1556-276X-8-418.

Prezioso, M., et al. “Training and Operation of an Integrated Neuromorphic Network Based on Metal-Oxide Memristors.” *Nature*, vol. 521, no. 7550, May 2015, pp. 61–64, https://doi.org/10.1038/nature14441.

Quellmalz, Arne, et al. “Large-Area Integration of Two-Dimensional Materials and Their Heterostructures by Wafer Bonding.” *Nature Communications*, vol. 12, no. 1, Feb. 2021, p. 917, https://doi.org/10.1038/s41467-021-21136-0.

Radhakrishnan, Janaki, et al. “Impacts of Ta Buffer Layer and Cu–Ge–Te Composition on the Reliability of GeSe-Based CBRAM.” *IEEE Transactions on Electron Devices*, vol. 66, no. 12, Dec. 2019, pp. 5133–38, https://doi.org/10.1109/TED.2019.2948894.

Rehman, Shania, et al. “Thickness-Dependent Resistive Switching in Black Phosphorus CBRAM.” *Journal of Materials Chemistry C*, vol. 7, no. 3, 2019, pp. 725–32, https://doi.org/10.1039/C8TC04538K.

Robinson, Joshua A. “Perspective: 2D for beyond CMOS.” *APL Materials*, vol. 6, no. 5, May 2018, p. 058202, https://doi.org/10.1063/1.5022769.

Roldan, Juan B., et al. “Spiking Neural Networks Based on Two-Dimensional Materials.” *Npj 2D Materials and Applications*, vol. 6, no. 1, Sept. 2022, p. 63, https://doi.org/10.1038/s41699-022-00341-5.

Rosenblatt, F. “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.” *Psychological Review*, vol. 65, no. 6, 1958, pp. 386–408, https://doi.org/10.1037/h0042519.

Sanchez Esqueda, Ivan, et al. “Aligned Carbon Nanotube Synaptic Transistors for Large-Scale Neuromorphic Computing.” *ACS Nano*, vol. 12, no. 7, July 2018, pp. 7352–61, https://doi.org/10.1021/acsnano.8b03831.

Sandberg, Anders. *Energetics of the Brain and AI*. Feb. 2016.

Sarker, Iqbal H. “Machine Learning: Algorithms, Real-World Applications and Research Directions.” *SN Computer Science*, vol. 2, no. 3, May 2021, p. 160, https://doi.org/10.1007/s42979-021-00592-x.

Schmidhuber, Jürgen. “Deep Learning in Neural Networks: An Overview.” *Neural Networks*, vol. 61, Jan. 2015, pp. 85–117, https://doi.org/10.1016/j.neunet.2014.09.003.

Seo, Jae-sun, et al. “On-Chip Sparse Learning Acceleration With CMOS and Resistive Synaptic Devices.” *IEEE Transactions on Nanotechnology*, vol. 14, no. 6, Nov. 2015, pp. 969–79, https://doi.org/10.1109/TNANO.2015.2478861.

Sezer, Omer Berat, et al. “An Artificial Neural Network-Based Stock Trading System Using Technical Analysis and Big Data Framework.” *Proceedings of the SouthEast Conference*, ACM, 2017, pp. 223–26, https://doi.org/10.1145/3077286.3077294.

Strubell, Emma, et al. *Energy and Policy Considerations for Deep Learning in NLP*. June 2019.

Sun, Yipeng, and Andreas M. Kist. *Deep Learning on Edge TPUs*. Aug. 2021.

Sutariya, Vijaykumar, et al. “Artificial Neural Network in Drug Delivery and Pharmaceutical Research.” *The Open Bioinformatics Journal*, vol. 7, no. 1, Dec. 2013, pp. 49–62, https://doi.org/10.2174/1875036201307010049.

Taher, Mohamed. “Accelerating Scientific Applications Using GPU’s.” *2009 4th International Design and Test Workshop (IDT)*, IEEE, 2009, pp. 1–6, https://doi.org/10.1109/IDT.2009.5404114.

Wang, Shaohua, et al. “Advances on Tumor Image Segmentation Based on Artificial Neural Network.” *Journal of Biosciences and Medicines*, vol. 08, no. 07, 2020, pp. 55–62, https://doi.org/10.4236/jbm.2020.87006.

Waterworth, G., and M. Lees. “Artificial Neural Networks in the Modelling and Control of Non-Linear Systems.” *IFAC Proceedings Volumes*, vol. 33, no. 1, Feb. 2000, pp. 95–97, https://doi.org/10.1016/S1474-6670(17)35594-5.

Wong, H. S. Philip, et al. “Metal–Oxide RRAM.” *Proceedings of the IEEE*, vol. 100, no. 6, June 2012, pp. 1951–70, https://doi.org/10.1109/JPROC.2012.2190369.

Wu, Xiaohan, et al. “Thinnest Nonvolatile Memory Based on Monolayer H-BN.” *Advanced Materials*, vol. 31, no. 15, Apr. 2019, p. 1806790, https://doi.org/10.1002/adma.201806790.

Xie, Jing, et al. “Hexagonal Boron Nitride (h-BN) Memristor Arrays for Analog-Based Machine Learning Hardware.” *Npj 2D Materials and Applications*, vol. 6, no. 1, July 2022, p. 50, https://doi.org/10.1038/s41699-022-00328-2.

Xu, Qi, et al. “Reliability-Driven Neuromorphic Computing Systems Design.” *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2021, pp. 1586–91, https://doi.org/10.23919/DATE51398.2021.9473929.

Yang Yin Chen, et al. “Balancing SET/RESET Pulse for $>\hbox{10}^{10}$ Endurance in $\hbox{HfO}_{2}\hbox{/Hf}$ 1T1R Bipolar RRAM.” *IEEE Transactions on Electron Devices*, vol. 59, no. 12, Dec. 2012, pp. 3243–49, https://doi.org/10.1109/TED.2012.2218607.

Yao, Peng, et al. “Face Classification Using Electronic Synapses.” *Nature Communications*, vol. 8, no. 1, May 2017, p. 15199, https://doi.org/10.1038/ncomms15199.

Yasar, A., et al. “Classification of Parkinson Disease Data with Artificial Neural Networks.” *IOP Conference Series: Materials Science and Engineering*, vol. 675, no. 1, Nov. 2019, p. 012031, https://doi.org/10.1088/1757-899X/675/1/012031.

Yin, Shihui, et al. “Monolithically Integrated RRAM- and CMOS-Based In-Memory Computing Optimizations for Efficient Deep Learning.” *IEEE Micro*, vol. 39, no. 6, Nov. 2019, pp. 54–63, https://doi.org/10.1109/MM.2019.2943047.

Yu, Shimeng, et al. “RRAM for Compute-in-Memory: From Inference to Training.” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 7, July 2021, pp. 2753–65, https://doi.org/10.1109/TCSI.2021.3072200.

Yu, Shimeng, and H. S. Philip Wong. “Modeling the Switching Dynamics of Programmable-Metallization-Cell (PMC) Memory and Its Application as Synapse Device for a Neuromorphic Computation System.” *2010 International Electron Devices Meeting*, IEEE, 2010, pp. 22.1.1-22.1.4, https://doi.org/10.1109/IEDM.2010.5703410.

Zahoor, Furqan, et al. “Resistive Random Access Memory (RRAM): An Overview of Materials, Switching Mechanism, Performance, Multilevel Cell (Mlc) Storage, Modeling, and Applications.” *Nanoscale Research Letters*, vol. 15, no. 1, Dec. 2020, p. 90, https://doi.org/10.1186/s11671-020-03299-9.

Zhang, Wenqiang, et al. “Neuro-Inspired Computing Chips.” *Nature Electronics*, vol. 3, no. 7, July 2020, pp. 371–82, https://doi.org/10.1038/s41928-020-0435-7.

Zhao, Meiran, et al. “Reliability of Analog Resistive Switching Memory for Neuromorphic Computing.” *Applied Physics Reviews*, vol. 7, no. 1, Mar. 2020, p. 011301, https://doi.org/10.1063/1.5124915.

Zhu, Kaichen, Xianhu Liang, et al. “Graphene–Boron Nitride–Graphene Cross-Point Memristors with Three Stable Resistive States.” *ACS Applied Materials & Interfaces*, vol. 11, no. 41, Oct. 2019, pp. 37999–8005, https://doi.org/10.1021/acsami.9b04412.

Zhu, Kaichen, Chao Wen, et al. “The Development of Integrated Circuits Based on Two-Dimensional Materials.” *Nature Electronics*, vol. 4, no. 11, Nov. 2021, pp. 775–85, https://doi.org/10.1038/s41928-021-00672-z.

APPENDIX A

GLOSSARY OF NEUROMORPHIC SYSTEMS

1. Neuromorphic computing: An emerging field of computing intended to develop computer systems that function more like biological brains. These systems employ electronic components and circuits that mimic the behavior of neurons and synapses in the brain to process information.

2. Machine Learning (ML): A subset of artificial intelligence that involves the use of algorithms and statistical models to enable computer systems to automatically learn from and improve upon data without being explicitly programmed.

3. Deep learning: A subset of machine learning which uses artificial neural networks to help computers learn and make decisions from massive sets of complex data. These neural networks process and extract information from the input data using numerous layers of interconnected nodes. Deep learning models can find intricate patterns and relationships in massive amounts of data that would be challenging for humans to observe.

4. Deep neural network: A type of artificial neural network used in deep learning that consists of multiple layers of interconnected nodes. The output from the previous layer is processed by each layer of nodes, enabling the extraction of increasingly abstract properties from the incoming data.

5. In-memory computing: A type of computing architecture that uses random access memory (RAM) to store and process data instead of the more conventional disk-based storage. Large data sets can be processed and analyzed faster due to the greatly decreased access times provided by in-memory data storage.

6. Moore's law: A prediction made by Gordon Moore, co-founder of Intel, in 1965 that the number of transistors on a microchip would double approximately every two years, while the cost of computing would decrease.

7. Weight: In neuromorphic computing, weight is the measure of how strongly or intensely neurons or artificial synapses are connected in a neural network. As they determine the importance of the input data and the response of the neurons, these weights serve as a representation of the network's ability for processing information.

8. Activation function: In neural networks, activation function is a mathematical function that is applied to a neuron's output to decide whether it should be activated. The network's non-linearity, brought forth by the activation function, enables it to simulate complex relationships between inputs and outputs. In neural networks, activation functions come in a variety of forms, including:

- Sigmoid function: This function, which has the shape of a sigmoid, converts any input into a number between 0 and 1. It is frequently employed in binary classification issues where the output is a probability value.
- ReLU (Rectified Linear Unit) function: It outputs the input value if it is positive and 0 otherwise. Due to its effectiveness and simplicity, ReLU is the activation function that deep neural networks utilize the most frequently.
- Tanh function: Similar to the sigmoid function, the tanh function converts the input to a number between -1 and 1. Often, feedforward neural networks use tanh.
- Softmax function: A generalization of the sigmoid function, the softmax function converts the input to a probability distribution across many classes. It frequently appears in multiclass classification problems.

9. Training/learning: In neural networks, training is the process of adjusting the weights and biases of the network's artificial neurons to increase the predictability of the network.

10. Supervised/unsupervised learning: In supervised learning, the network is trained using a labeled dataset, implying that the right output is given for each input. After then, the network is trained to reduce the discrepancy between expected and actual output. In unsupervised learning, neural networks are trained on unlabeled datasets, implying that the correct output for each input is not provided. Instead, it is left to the network to identify structure and patterns in the input data on its own. Tasks like clustering and anomaly detection frequently involve unsupervised learning.

11. Epoch: A single iteration of the whole training dataset while a neural network is being trained is referred to as an epoch in machine learning. In order to reduce the discrepancy

between the expected output and the actual output, the network processes the full training dataset during each epoch and adjusts the weights and biases of the neurons.

12. Loss/cost function: A loss or cost function in machine learning is a mathematical function that assesses the discrepancy between a neural network's predicted and actual output. By modifying the neural network's weights and biases, the training process aims to reduce the value of loss function.

13. Learning rate: In machine learning, learning rate is a hyperparameter that controls how frequently a neural network's weights and biases are updated during training. How fast or slowly the network modifies its weights and biases in response to the discrepancy between the projected output and the actual output depends on its learning rate.

14. Accelerator: A customized hardware device developed to expedite the training and inference procedures of neural networks in the context of artificial intelligence. Accelerators, as opposed to general-purpose CPUs or GPUs, are made to perform the large computations needed by neural networks more effectively.

15. Inference: The practice of utilizing a trained neural network to generate decisions or predictions based on fresh, unobserved data is known as inference in neural networks.

16. Overfitting: Overfitting is when the network gets too complicated and fits the training data too closely. This leads to poor generalization to new data. In other words, the network has gotten too specialized to that specific dataset as a result of how effectively it has learned the training data.

17. Petaflop/s-days: It is a measure of communication volume (throughput). It represents the total amount of computations that would be performed throughout a complete day on a computer with a throughput of 1 PetaFLOP/s.

18. TOPS (Tera Operations Per Second): A measure of the computational efficiency of a neural network accelerator or processor. The number of operations (such as multiply-

accumulate operations) that a hardware device can complete in one second is measured by TOPS.

19. TOPS/W: A measure of performance and energy efficiency.