VARIATION, ENERGY CONSUMPTION, LATENCY, AND FAILURE AWARE DESIGN FOR MEMRISTIVE ANNs

A Dissertation
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Zhiheng Liao

In Partial Fulfillment of the Requirements
for the Degree of
DOCTOR OF PHILOSOPHY

Major Department:
Electrical and Computer Engineering

June 2021

Fargo, North Dakota

North Dakota State University
Graduate School

Title

VARIATION, ENERGY CONSUMPTION, LATENCY, AND FAILURE AWARE DESIGN FOR MEMRISTIVE ANNs

By

Zhiheng Liao

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

DOCTOR OF PHILOSOPHY

SUPERVISORY COMMITTEE:

Dr. Qifeng Zhang
Chair

Dr. Dali Sun

Dr. Sumitha George

Dr. Umamaheswara Rao Tida

Dr. Ying Huang

Approved:

July 16, 2021	Dr. Benjamin Braaten
Date	Department Chair

ABSTRACT

As a novel non-volatile device, the memristive crossbar array has delivered many promises in giving low computation complexity, high energy efficiency, and high density for neuromorphic computing. However, the intrinsic variability of switching behavior, energy consumption, and stuck at fault are still major obstacles to their implementation. Here we report our investigations of a model that experimentally demonstrates the natural stochasticity of cycle-to-cycle variations. In addition, we propose three techniques to mitigate the adverse impact of cycle-to-cycle variations, optimize energy consumption, reduce system latency, and improve fault tolerance. The relationship of the level of conductance and cycle-to-cycle variation was studied, and experiment results show an optimal number of the levels to mitigate cycle-to-cycle variations in the system. Additionally, the system compresses the number of pulses when the conductance is updated by the pulse stimulus to reduce cycle-to-cycle variations, resulting in a great energy and latency reduction. What's more, the fault tolerance of the memristor-based system has been improved by a novel weight mapping method. This work paves the way of adopting memristors for more efficient applications in the era of edge computing and the Internet of Things.

ACKNOWLEDGMENTS

All praises and thanks to this world for giving me courage and strength to pursue my Ph.D.

My heartiest and sincere appreciation and gratitude to my adviser Dr. Jinhui Wang, who always encouraged me, and persistently conveyed the spirit and guidance required for the research. At the same time, without the help from Dr. Scott C. Smith, Dr. Sudarshan K. Srinivasan, and Dr. Qifeng Zhang, this disquisition would not have been possible.

Special thanks to my committee members, Dr. Ying Huang, Dr. Dali Sun, Dr. Sumitha George, and Dr. Umamaheswara Rao Tida, for their support, guidance and helpful recommendations. Thanks to the Electrical and Computer Engineering staff members Jeffrey Erickson, Laura D. Dallman, and Anne Campbell for all the unconditional help and favor.

Special thanks to the Center for Computationally Assisted Science and Technology (CCAST) staff members Dr. Khang Hoang and Nicholas Dusek for all the unconditional help and favor.

Portions of this work were conducted in the Core Research Facility at North Dakota State University. Special thanks to Dr. Yong Wang, Aaron Reinholz, Gregory Strommen, and Frederik Haring who assisted with fabrication processes and measurements/testing.

I owe my heartiest thanks to all my friends and colleagues, who always helped me in the time of need.

Finally, I would like to thank my family. Their continuous support is always a source of motivation and encouragement for me.

DEDICATION

To this world.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

1T1M	One-Transistor One-Memristor
AdaGrad	Adaptive Gradient
Adam	Adaptive Moment Estimation
ADMM	Alternating Direction Method of Multipliers
AI	Artificial Intelligence
ALU	Arithmetic Logic Unit
ANNs	Artificial neural networks
CMOS	Complementary Metal-Oxide-Semiconductor
G_{max}	Maximum Conductance
G_{min}	Minimum Conductance
IoT	Internet of Things
LTD	Long-Term Depression
LTP	Long-Term Potentiation
MLP	Multilayer Perceptron
MNIST	Modified National Institute of Standards and Technology
ReRAM	Resistive Random-Access Memory
RMSProp	Root Mean Square Prop
SA0	Stuck-Off Fault
SA1	Stuck-On Fault
SGD	Stochastic Gradient Descent
VLSI	Very Large-Scale Integration

1. INTRODUCTION

1.1. Background

The internet of things (IoT) system is a network of devices, sensors, and other items of various functionalities that interact and exchange data electronically [1]. In recent years, there has been significant progress in edge devices and wireless sensor networks, creating unprecedented opportunities to deploy deep learning and artificial intelligence (AI) technologies in IoT, while significantly adding calculation burdens in edges [2]. However, edges consisting of mobile devices and embedded systems usually have limited resources and power, especially when they are used for real-time applications, so resource and power deficiency will result in recognition and prediction accuracy loss in a learning system and malfunctions in IoT [1-6].

IoT is receiving a great attention due to its potential strength and ability to be integrated into any complex systems and it is becoming a great tool to acquire data from environment to the cloud. Data that are acquired from wireless sensor nodes could be predicted using Artificial Neural Networks (ANNs) models. ANNs that are also called neural networks, are computing systems that originated from the neural network structure in biology, which constitute animal brains. The working principle of the calculation system of artificial neural networks imitates the working principle of biological neural networks. Thus, the scientific and technological terms of the computing system of artificial neural networks have therefore inherited the terms of biological neural networks.

The history of neural learning in biology begins with Hebbian theory in 1949. The Hebbian theory is a neuroscience theory that claims that the increase in synaptic efficacy stems from the repeated and continuous stimulation of presynaptic cells to postsynaptic cells. It attempts to explain synaptic plasticity, the adaptation of brain neurons during learning. The biological learning process is a complex one because the basic kinds of connections between neurons are synapses including both chemical and electrical synapses. One principle of this is excitations at the postsynaptic membrane will sum up in the cell body. An action excitation will occur that travels down the axon to the terminal endings to transmit a signal to other neurons. It is noted that different synapses have different responses when receiving excitations and this property is called synaptic plasticity. These are often divided into short-term plasticity and long-term plasticity. Long-term synaptic plasticity is often contended to be the most likely memory substrate. The induction of long-term changes in synaptic efficacy, by long-term potentiation (LTP) or depression (LTD).

In my research, terms LTP and LTD are adopted to present such a learning process. This excitation travels in a way of a forward propagation and in some cells, but neural backpropagation does occur through the dendritic branching and may have important effects on synaptic plasticity and computation. ANNs imitate the biological neural network and developed various Learning paradigms, such as supervised learning, unsupervised learning, reinforcement learning, and self-learning. ANNs also developed various networks' structures, such as multilayer perceptron neural networks, convolutional neural network, and recurrent neural network. Basically, the components of ANNs include neurons, connections, weights, learning rates, cost function, forward propagation function, and backpropagation function.

For training an ANN model, optimization algorithm is utilized for achieving a solution (i.e. a set of weights). In the context of an optimization algorithm, the function used to evaluate a candidate solution is referred to as the objective function. Typically, with neural networks, we seek to minimize the error. As such, the objective function is often referred to as a cost function or a loss function and the value calculated by the loss function is referred to as simply "loss". A loss function is a way to measure how bad the performance of the current model is given the current input and expected output by feedforward process; because it is based on a training set it is, in fact, an empirical loss function [7]. The loss function has an important job in that it must faithfully distill all aspects of the model down into a single number in such a way that improvements in that number are a sign of a better model [8].

After defining the loss function, we want to adjust the parameters so that the loss is minimized. In fitting a neural network, backpropagation [9] computes the gradient of the loss function for the weights of the network for a single input-output example, and does so efficiently, unlike a naive direct computation of the gradient for each weight individually. This efficiency makes it feasible to use gradient methods for training multilayer networks, updating weights to minimize loss; gradient descent [10], or variants such as stochastic gradient descent (SGD) [11], are commonly used. The backpropagation algorithm works by computing the gradient of the loss function for each weight by the chain rule [12], computing the gradient one layer at a time, iterating backward from the last layer to avoid redundant calculations of intermediate terms in the chain rule.

After having above some basic concepts of ANNs, the workflow of ANNs is described as following. ANNs transform inputs to desired outputs by feedforward neural networks that comprise many layers. A simple neural network with one hidden layer is shown in Figure 1. Each node in the network is a neuron that takes a weighted sum of the outputs of the prior layer, and then transmits the sum to the next layer. Finally, the output of network is prediction for a certain task. The main work of a training ANN is to learn the feature that is represented by weight from a large volume of training data. During the training process, weight is updated in each iteration by weight change that is calculated based on an algorithm and backpropagation process. The model is trained when the prediction accuracy achieves a requirement by updating the weights.

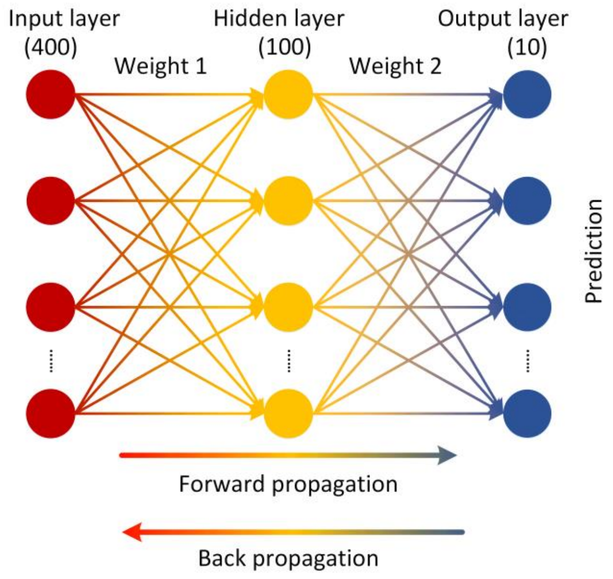


Figure 1. 3-layer fully connected structure of a neural network. The input layer, hidden layer, and output layer are composed of different neurons. Between two neuron layers, there are Weight 1 and Weight 2 that represent the strength of the connection.

For implementing ANNs by hardware, we choose memristor in this research. The conventional complementary metal-oxide-semiconductor (CMOS) transistor technology plateaus the process scaling [13, 14], which cannot provide satisfactory solutions for the emerging edge computing with designated learning systems [6]. Memristors are theoretically postulated by Chua in 1971 [15] and later are physically fabricated by Hewlett-Packard in 2008 [16]. The memristor-based crossbar arrays with storage and computing capability show great potential in neural network and machine learning applications [16-21]. They are characterized by low computational complexity [22], low power consumption [23], fast switching speed [24], high endurance [25], excellent scalability [26], and CMOS compatibility [27], which are especially appropriate for edge computing in IoT. Memristor enables in-memory computing and then implements ANNs [16, 17].

Memristive crossbar arrays conduct in-situ dot products and learns the feature of data by updating each memristor's conductance [18], as shown in Figure 2.

During tunning conductance process, the switching process in transition metal oxides is believed to be driven by the nanoscale motion of donor-type defects such as oxygen vacancies, which may tune the resistance of the oxide layer or the resistance of the electrode/oxide interface, which can be realized by input pulse [28-30]. Every row gets an input voltage signal which is a vector. Each conductance of a memristor in every cross point composes the matrix. Every column transmits an output current which is the sum of the product by the input signal and conductance in the same column. Due to the efficient implementations on vector-matrix multiplication operations, memristive crossbar array is suitable and efficient hardware for ANNs in edges. For instance, the input data in an ANN is converted to reading pulses for inputting a memristive crossbar array. The output current in each column is the weighted sum for the value of one neuron.

Thus, this reading process is a feedforward process in an ANN. At the same time, one reading operation can be done for the whole array, which accelerates the calculation of the weighted sum in an ANN. For backpropagation, the signal is input from columns and output from rows and then the loss function is obtained, which significantly improves the speed of calculation. Those properties determine the analog memristive crossbar array is suitable for implementing ANNs. The conductance of a memristor with multilevel as shown in Figure 2, is increased by supplying a positive pulse until the conductance reaches the maximum. This increasing process is LTP. Conversely, LTD is the process of decreasing the conductance by supplying a negative pulse until the conductance gets to the minimum. Simultaneously, the memristor can store the information even when the power supply is turned off, because of its non-volatile property. The memristor, therefore, is a device that combines learning, storage, and computing, making it an essential part of the hardware for ANNs, especially for edges with limited resources, but a real-time response in IoT systems.

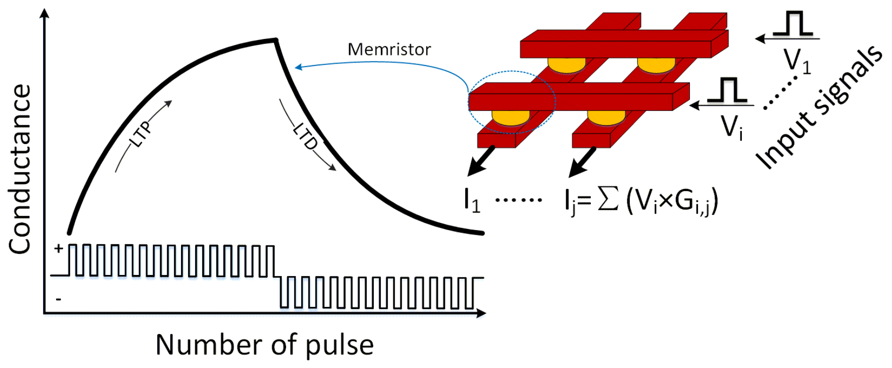


Figure 2. Hardware implementation of neural networks using memristor crossbar. V_i, G_{ij}, and I_j represent the input signal in ith row, the conductance of the memristor in jth column and ith row, and the output current that represent the dot product result of V and G, respectively. The conductance of the memristor is regulated with the number of pulses. The LTP/LTD is triggered by positive/negative pulses.

However, because of the inherent material properties, the intrinsic variation in switching conductance is a major challenge for some applications such as non-volatile memory [31]. Since the switching mechanism of the memristor conductance is prompted by the applied voltage, a memristor switches its conductance from one to another when the voltage of a pulse is larger than a threshold voltage for at least the minimum switching time [32]. Cycle-to-cycle variation is a deviation between target conductance and updated conductance when the same updating signal in different updating cycles is applied in a memristor, even when the initial conductance is the same [33], as shown in Figure 3. For instance, for some given updating pulses, a memristor starting at conductance A and target conductance is B, may end up between C and D, as shown in insert Figure 3. The relative error with desired value is shown in Figure 3. The maximum variation is 9.3% with desired value in this testing result. It significantly affects the accuracy of conductance in updating process and then affects the system prediction accuracy.

Memristors exhibit cycle-to-cycle variation because of the shape of the conductive filament, the oxygen vacancy distribution at and around the filament, and the changing location of the active filament between one cycle to the next. These mechanisms originate from the coexistence of multiple subfilaments and that the active, current-carrying filament may change from cycle to cycle [33]. Thus, cycle-to-cycle variation is a type of inherent randomness associated with the randomness in internal atomic configurations [34-36]. One of the major obstacles to the implementation of redox-based memristive memory or logic technology is the large cycle-to-cycle variation [33]. The memristor-based crossbar arrays suffer from serious cycle-to-cycle variation especially when arrays are used in the neuromorphic computing system that the conductance of the memristor needs to be updated innumerable times during the training and testing process [34, 37-40].

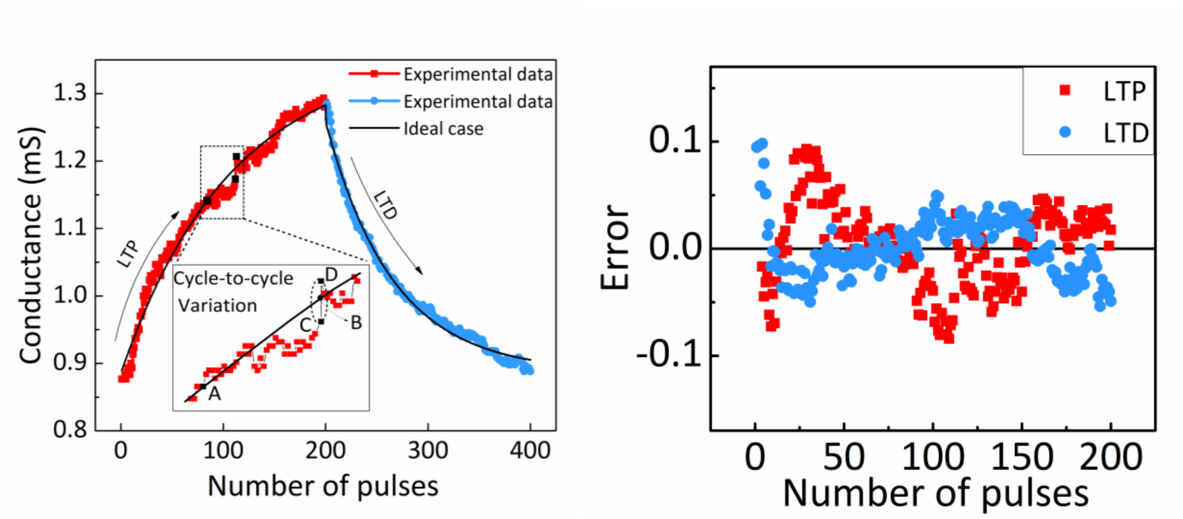


Figure 3. Cycle-to-cycle variation and relative error with the desired value of the memristive device.

Additionally, similar to CMOS circuits, the high-performance functionality of a memristive crossbar array that is utilized in neural networks translates into high power densities, high operating temperatures, and low reliability [41-43]. If the memristive crossbar array runs at the edge for an IoT system, overheat will reduce the effectiveness and lifespan of components. Furthermore, learning algorithms, usually routinely considering accuracy nowadays, rarely pay attention to energy efficiency and latency. In fact, uneven weight updates for different memristors caused by algorithms inevitably leads to local overlarge energy consumption. Also, the writing process of a memristor consumes much more energy than the reading process. Therefore, most of the energy during the training process of a memristive crossbar array comes from the writing for the weight update [44]. Consequently, the energy consumption becomes a significant contributor to the decline system reliability, deteriorates memristors' retention and endurance, and causes severe timing uncertainty [45-47].

In addition to the energy consumption issue, stuck-at-fault is also a barrier that prevents the memristor from becoming a practical technology and using it in real hardware implementations is the reliability issue [48, 49]. The stuck-at-fault defect will lead the memristor's conductance to be fixed to a maximum or minimum conductance, regardless of its programmed, as known as the stuck-on fault (SA1) or stuck-off fault (SA0). The unstable fabrication and test factors will affect the defects, the fabrication yield, and the forming process of the shape of the conductive filament, which causes the device failure [48-50]. For the implementation of a neuromorphic computing system, when mapping the well trained ANNs model to the memristor crossbars, the stuck-at-fault defects result in a mismatch between target weights and the actual values mapped onto the crossbar and eventually lead to an undesirable drop in prediction accuracy of ANNs [51-53].

We studied the memristor's characteristics in-depth by fabricating, testing, modeling, and simulating to improve the performance of a system, including mitigating the effect of variation, reducing energy consumption, lowering system latency, and ameliorating reliability. With the proposed model and techniques, the improvement of our work against the state-of-the-art is shown in this dissertation.

1.2. Research Challenges

This research focuses on the design and optimization of a memristor-based neuromorphic computing system to reduce the effect of variation of memristor and improve the performance of the system. Traditional optimization techniques usually come with significant implementation costs that two of the most common costs are the silicon area overhead and performance penalty.

Therefore, the first goal is to come up with some novel variation-immune techniques with simplified additional logic in order to minimize the area overhead. The second goal is to propose some novel methods to improve the performance of the system. The third goal is to perform simulations to make sure the effectiveness.

1.3. Statement and Contributions

The main focus of this dissertation is on modeling and mitigating the effect of cycle-to-cycle variation, improving the energy efficiency and system latency, enhancing fault tolerance for the memristor-based ANNs, thereby enhancing the performance of the edge computing in IoT systems. A model of cycle-to-cycle variation and an overview of the pulse distribution of the memristor-based ANNs and their relation to the energy consumption, system latency, and fault tolerance are provided. Specifically, this paper makes the following contributions:

1) A model for the cycle-to-cycle variation of memristor: Cycle-to-cycle variation is modeled according to the testing data that was obtained by testing in-house fabricated memristor arrays. The model gives the relationship within one update process between the total cycle-to-cycle variation and the number of updating pulses.

2) Two cost-friendly methods to mitigate the effect of cycle-to-cycle variation of the memristor in ANNs: According to the model of cycle-to-cycle variation of memristor, two methods are proposed. The proposed methods will maximally mitigate the effect of cycle-to-cycle variation by optimizing the way to use memristor.
3) A mechanism for smoothing the learning process, thereby avoiding extra energy consumption and writing latency in edge computing: With the proposed method, the weight updating fluctuation is reduced. Therefore, the energy consumption and the writing latency that is decided by the number of pulses are drastically reduced by the compressing mechanism. Additionally, the timing regularity of the system is improved.
4) A method for fault tolerance of memristor-based ANNs: By a novel weight mapping method, the weight can be mapped to suitable conductance of memristor, which improves the fault tolerance of memristor-based ANNs.
5) Thorough evaluation: We evaluate the proposed methods based on the standard image classification tasks [54] and the hardware-based online learning simulator, NeuroSim+ [55], enabling a system under different failure rates, different network architecture, nonlinearity, and various variations.

1.4. Organization

This dissertation is organized into 8 chapters. Chapter 2 introduces the prior art work and contains an in-depth discussion of the fundamental terminology and concepts, which will provide the foundation for the rest of the dissertation. Modeling cycle-to-cycle variation and mitigating the effect of the cycle-to-cycle variation are introduced and followed by improving system performance. Techniques introduced are compared in this work.

Chapter 3 introduces a complete flow to fabricate and test memristor. Based on the CMOS cleanroom facility, memristor crossbar arrays are fabricated on a 4-inch wafer, which includes baking, coating, photolithography, development, deposition, lift-off, and etching processes. Measurement data is then extracted from the test results using a semiconductor parameter analyzer. The data of this result is a cornerstone of cycle-to-cycle variation in this dissertation.

Chapters 4 presents an analysis of test data and modeling the cycle-to-cycle variation. According to previous work, the cycle-to-cycle variation is a true random value and confirms normal distribution. By analyzing test data, the relationship within one update process between the total cycle-to-cycle variation and the number of updating pulses is studied. A model for presents this relationship is proposed.

Chapters 5 introduces two techniques for mitigating the effect of cycle-to-cycle variation of memristor that includes level scaling method and pulse regulating method. The optimal number of levels is obtained via the level scaling method to reduce the effect of cycle-to-cycle variation. Additionally, the pulse regulating method is proposed according to the model of cycle-to-cycle variation to minimizing the effect of cycle-to-cycle variation by reducing the number of updating pulses during one update process.

In Chapter 6, the pulse regulating method for reducing energy consumption and system latency is discussed. By reducing the number of updating pulses during one update process, the pulse regulating method decreases the total number of updating pulses and then saves energy consumption. Because the total number of updating pulses is saved, the time of updating process is saved, thereby the system latency is reduced.

In Chapter 7, the differential mapping method is used to enhance fault tolerance of memristor-based ANNs. Utilizing the difference of conductance between two memristors to represent one weight value in ANNs, the fault tolerance of memristor-based neuromorphic computing system is enhanced according to the property of distribution of weight value.

Chapter 8 summarizes the major conclusions of this dissertation and suggests a direction for future research.

2. PREVIOUS WORK

To optimize the performance of a memristor-based ANNs system, researchers provide different techniques according to the issues that include cycle-to-cycle variation, energy consumption, system latency, and fault tolerance. However, the improvements in the relative topics of those general-purpose design techniques are often achieved with significant design complexity, increased silicon area, power penalty, and system latency.

2.1. Previous Work on Modeling and Mitigating the Effect of the Cycle-to-cycle Variation

There exist several works on cycle-to-cycle variation modeling. According to the accumulation of the previous conductance changes, the distribution of memristor conductance after the nth pulse has been modeled [56, 57]. The cycle-to-cycle variation is modeled based on sigmoid function [34, 58]. Although those methods can model the given experimental data, they are not suitable for our data. Although cycle-to-cycle variation is considered in simulations [59], the amount of cycle-to-cycle variation is expressed in terms of the percentage of the entire conductance range and cannot present the relationship within one update process between the total cycle-to-cycle variation and the number of updating pulses, explicitly.

In previous works, researchers also proposed some solutions including three aspects to mitigate such impact of the cycle-to-cycle variation.

From a software-based and algorithm perspective: a conversion algorithm is invented to map arbitrary matrix values appropriately to memristor conductance to reduce computational errors [60]. The algorithms of the mutual decision between the conductance of memristor and Boolean functions are used to tolerate a maximum variation [61]. A novel off-device neural network training method is used to improve the performance of the neural network [62]. However, because variation comes from memristor devices - hardware of the neuromorphic computing system, the software-based methods are usually resource-consuming.

From a circuit perspective: the smart programming scheme (read the conductance before writing it) and dummy column technologies to eliminate the off-state current are utilized to improve immunity to cycle-to-cycle variations [63, 64]. The experimental result shows the accuracy is improved to 95% from 70%. In addition, a variation-aware training scheme is used to enhance training robustness [65]. Sophisticated circuits are needed to ensure the quality of conductance switching and either drastically increase the area of circuit and power consumption or bring additional circuit latency.

From a device perspective: instead of using a single memristor, the multiple cells technology using several memristors connected in parallel are applied to improve the variation tolerance [66, 67]. But the multiple cells produce area overhead in the system. In addition, the different materials, such as TiOx as buffer layer [68] and CeO2/Ti/CeO2 tri-layered as active layer [69], are proposed and investigated to improve the resistance of ratio between a high-resistance state and low-resistance state, enhance the endurance of switching, and reduce the variation of the threshold voltage.

Therefore, in this context, it is necessary to propose a new modeling method to model the realistic cycle-to-cycle variation of the memristor. Developing a cycle-to-cycle variation model on memristor is urgent for mitigating that impact so that it is accessible to apply the great potential and advantages of the memristor in practical applications.

2.2. Previous Work on Optimizing Energy Consumption and Latency

Recently, researchers proposed several techniques to improve energy efficiency and system latency from different levels. Dual-element memristors are used to achieve low-power memory design [70]. A memristor-based predictor is designed to reduce energy consumption comparing to the digital counterpart [71]. The hybrid crossbar architecture for improving the performance of energy efficiency and system latency is studied in [72]. In [73], the error-correcting code is proposed to relax the Bit Error Rate requirement of a single memory to improve the write energy consumption and latency for both the CMOS-based and crosspoint-based memristor resistive random-access memory (ReRAM) designs. PRCoder as an algorithm was proposed for different RRAM applications [42]. The cycle-rehabilitate technique was used to alleviate thermal crosstalk [43]. At the same time, increasing the size of the insulator or utilizing new materials with higher thermal conductivity for improving performance were proposed in [74, 75]. A new structure, thermal-house, was presented to optimize thermal management [76]. However, those new algorithms or new material/structure of device-based designs inevitably increase the complexity of peripheral circuits or the difficulty of the manufacture process, even increasing the latency.

2.3. Previous Work on Improving Fault Tolerance of Memristive ANNs

Some techniques have been proposed to model and detect the faults in memristor crossbar array including fault model with testing scheme [77], a marching algorithm to cover defect [50], and analyzing impacts of stuck-at faults on the accuracy of a sparse coding network [78]. Various schemes to tolerate faults in memristor crossbar array have also been proposed in some prior works. At hardware level, isolating faulty memristors is proposed by switching off the access transistors [79]. However, for the large memristor crossbar array that is implemented in complicated neural networks, tremendous routing and area overheads are inevitable utilizing controlling individual access transistors. redundant columns of memristors are utilized as a substitution for some columns of memristors that have more failures [49]. However, this method not only introduces nontrivial area overhead but also increases the design complexity of peripheral circuits. Unlike hardware-level schemes, fault aware network retraining with weight mapping/remapping are commonly composed in software level optimizations. For example, a fault-aware neural network training method is proposed or the order of the crossbar rows and columns is permuted to improve fault tolerance [49, 51, 62, 65]. However, the bi-partite algorithm only utilizes zero value weights in sparse networks to benefit SA0 faults. Moreover, besides SA0 faults, there are also many SA1 faults that is much more than SA0 in the memristor crossbar array [50], which cannot be accommodated well. Moreover, retraining and remapping schemes, on one hand, increase computing and interaction costs, furthermore, on the other hand, they are unpractical in some scenarios especially at the edge of the IoT [14, 80-82].

These methods require an individual optimization process for each ReRAM crossbar, and may also introduce complex control circuits, resulting in additional hardware overhead. Although these methods are effective in mitigating the accuracy drop caused by the stuck-at-fault defects, even [52] can restore 99% of the accuracy drop, but for mass-produced IoT products, applying optimization for each product (IoT device) will bring a huge time cost, and is not realistic. Thus, it is desirable to have a universal approach to improve the DNN fault tolerance.

3. MEMRISTOR FABRICATION AND TEST

The mechanism of multilevel conductance is introduced in this chapter. Also, the details of fabrication and test are included, which shows the fabricated memristor device possesses a multilevel conductance characteristic.

In terms of memristor, changes of the resistance are caused by external electrical stimuli cause within an oxide layer that is sandwiched between two metal electrodes [33]. The switching process in transition metal oxides is believed to be driven by the nanoscale motion of donor-type defects such as oxygen vacancies, which may tune the resistance of the oxide layer or the resistance of the electrode/oxide interface [28-30]. The stochastic nature of the resulting conductive filament has been suspected phenomenologically to result from an interplay of the thermodynamic stability of the filament [83], generation/recombination effects of oxygen vacancies [84], and especially the shape and oxygen vacancy distribution of the conductive filament [85-87]. It has therefore been attempted to achieve higher switching uniformity employing filament precursors [88] or preferred sites for oxygen vacancy enrichment [89, 90].

3.1. Memristor Fabrication

In this work, 4 inch Si wafers are used that have 100 nm thermally grown SiO_2 on top as the substrates. For the 40μm x 40μm memristive device, the bottom electrodes were patterned by ultraviolet photolithography. After that, a 100 nm thick Al bottom electrode was deposited in a Kurt Leaker CMS-18 Sputterer, followed by a lift-off process in acetone. A 100 nm-TiO_2/100 nm-TiO_{2-x} active layer was prepared by sputtering of a Ti target under an oxygen atmosphere (power for TiO_2/TiO_{2-x}: 262 W). Top electrodes were defined by a photolithography step, deposition of a 100 nm Al using sputtering (650 W), and lift-off. The exposure of bottom electrodes was done by etching the active layer through HF.

Before showing all of steps for fabricating in this research, some concepts are introduced as following.

- HMDS: The Hexamethyldisilazane promotes good photoresist-to-wafer adhesion because it ensures the wafer surface is hydrophobic. After HMDS treatment the silicon surface oxide becomes silated, leaving a non-polar surface. It creates a bridge of organic to inorganic molecules between the surface of the silicon wafer and photoresist. Without it, the photoresist does not form a secure bond with the wafer.
- Coating: Photoresist (light-sensitive solution) is coated and formed into a film over the SiO_2 film.
- Softbake: Prebake, also known as softbake or preexpose bake, is the physical process of conversion of a liquid-cast resist into a solid film.
- Exposure: Lithography in the MEMS context is typically the transfer of a pattern to a photosensitive material by selective exposure to a radiation source such as light.
- Development: In a developer, the wafers are uniformly covered with a developing solution to develop the mask patterns.
- Rinse/Dry: The excess develop solution and water are removed by this step.
- Deposition: The specified materials is deposited on the surface of wafer.
- Lift Off: The lift-off process is a method of creating structures (patterning) of a target material on the surface of a substrate using a sacrificial material (e.g. photoresist).
- Etching: Etching is traditionally the process of using strong acid or mordant to cut into the unprotected parts of a metal surface to create a design in the metal. In this work, etching technique is utilized to expose the pad of bottom metal. Before etching, coating, softbake, exposure, and development steps are needed to obtain the pattern for etching.

For one layer material fabrication, the process steps typically include from coating step to lift off step. In this work, repeating three times of those steps for fabricating three layers including bottom metal (Al), active layer (TiO_2/TiO_{2-x}), and top metal (Al). Table 1 shows every step for fabricating memristive crossbar array. Figure 4 shows the cross-section of main steps. Note that, (e) shows the cross-section (x direction) of the fabricated top metal layer, which process includes coating, exposure, deposition, and lift-off for fabricating top metal. Every procedure for this layer doesn't be shown in figure because mask patten for this layer is along the y direction rather than x direction. At the same time, etching process doesn't be shown in this figure because it has the same cross-section with (e). The size of memristor discussed in this research is 40 μm x 40 μm.

Table 1. Process Steps for Fabricating Memristive Crossbar Array.

Process	Equipment	Comment
HMDS	YES Oven	Creating a bridge of organic to inorganic molecules between the surface of the silicon wafer and photoresist.
Coating	RC8	Photoresist is coated and formed into a film.
Softbake	Flex Oven	The physical process of conversion of a liquid-cast photoresist into a solid film
Exposure	MA8	Transferring a pattern to a photosensitive material by light for bottom metal layer.
Development	Solvent Station	The wafer is uniformly covered with a developing solution to develop the mask patterns.
Rinse/Dry	Solvent sink	Removing the development solution and drying the wafer.
Inspection	MX50	Make sure photoresist is fully cleared.
Metal deposition	PVD1	Bottom metal (Al) is deposited on the surface of wafer by spattering.
Lift Off	Solvent Station	Creating structures (patterning) of bottom metal on the surface of a substrate using photoresist.
Inspection	MX50	Check the pattern of the bottom metal.

Table 1. Process Steps for Fabricating Memristive Crossbar Array (continued).

Process	Equipment	Comment
Metal deposition	PVD1	Active layer (TiO_2/TiO_{2-x}) is deposited on surface of wafer by spattering with Ti and O_2.
Coating	RC8	Photoresist is coated and formed into a film for top metal layer.
Softbake	Flex Oven	The physical process of conversion of a liquid-cast photoresist into a solid film
Exposure	MA8	Transferring a pattern to a photosensitive material by light for top metal layer.
Development	Solvent Station	The wafer is uniformly covered with a developing solution to develop the mask patterns.
Rinse/Dry	Solvent sink	Removing the development solution and drying the wafer.
Inspection	MX50	Make sure photoresist is fully cleared.
Metal deposition	PVD1	Top metal (Al) is deposited on surface of wafer by spattering with Al.
Lift Off	Solvent Station	Creating structures (patterning) of top metal on the surface of a substrate using photoresist.
Inspection	MX50	Check the pattern of the top metal.
Coating	RC8	Photoresist is coated and formed into a film for etching.
Softbake	Flex Oven	The physical process of conversion of a liquid-cast photoresist into a solid film
Exposure	MA8	Transferring a pattern to a photosensitive material by light for etching.
Development	Solvent Station	The wafer is uniformly covered with a developing solution to develop the mask patterns.
Rinse/Dry	Solvent sink	Removing the development solution and drying the wafer.
Inspection	MX50	Make sure photoresist is fully cleared.
Etching	Trion	Etching for exposing the bottom layer pad.

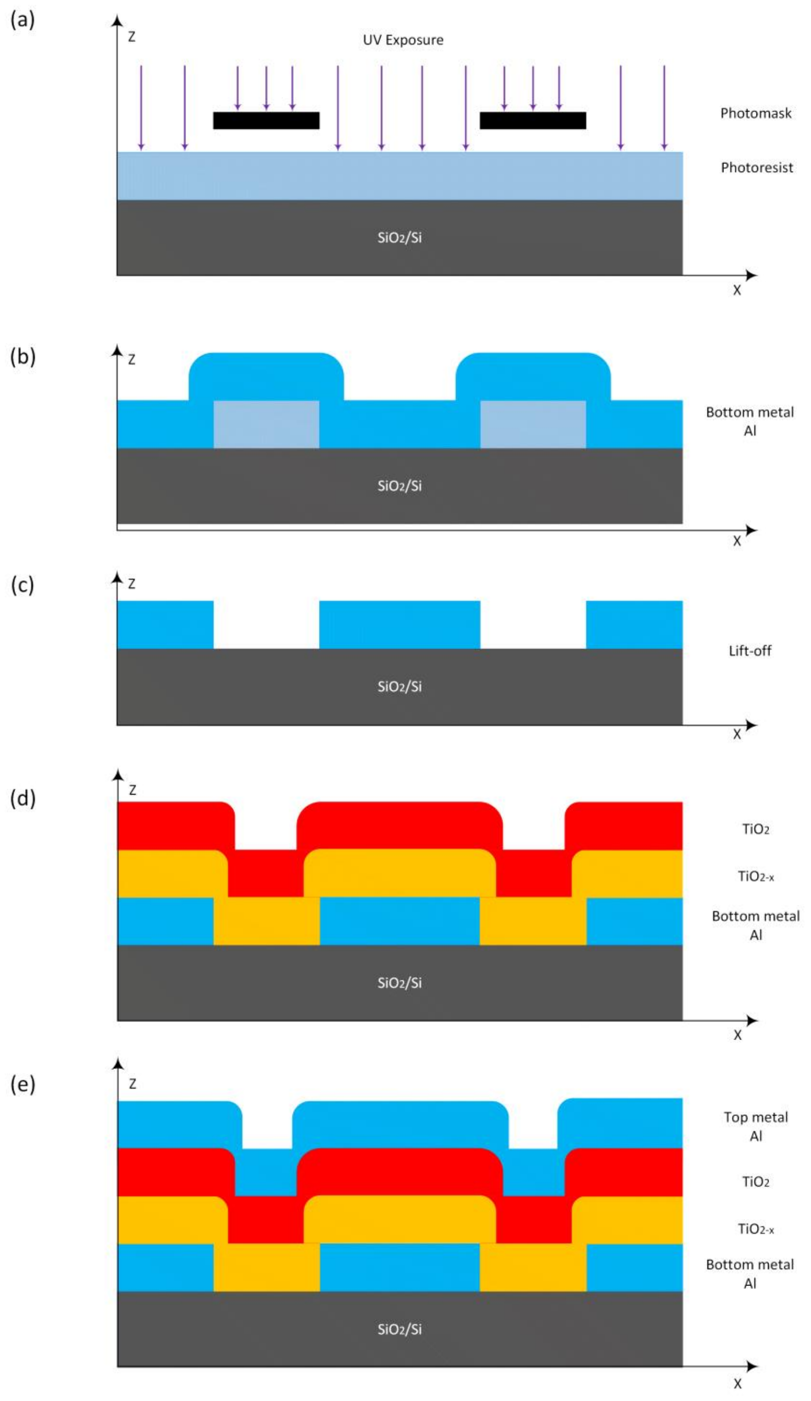
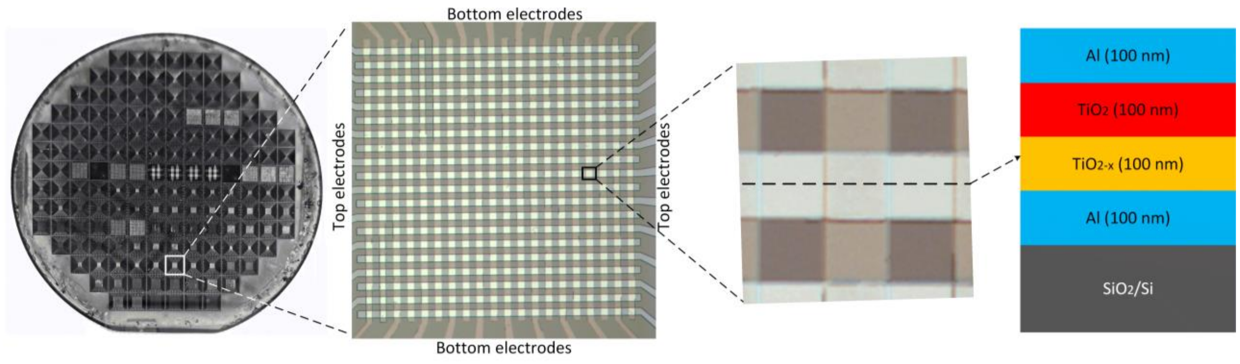


Figure 4. The flowchart detailing various processing steps for fabrication of memristor crossbar array. Note that, (e) shows the cross-section (x direction) of the fabricated top metal layer, which process includes coating, exposure, deposition, and lift-off for fabricating top metal. Every procedure for this layer doesn't be shown in figure because mask patten for this layer is along the y direction rather than x direction. At the same time, etching process doesn't be shown in this figure because it has the same cross-section with (e). The size of memristor discussed in this research is 40 μm x 40 μm.

The optical image and geometry of TiO_2/TiO_{2-x} based memristive crossbar arrays used in this work are schematically shown in Figure 5a. The array is composed of 20 x 20 memristors, as shown in Figure 5b. Physically, a memristor is a 40 μm x 40 μm two-terminal device formed by two aluminous electrodes sandwiching a thin active layer that is TiO_2/TiO_{2-x} material to achieve stable tunable multilevel conductance with a nonlinear current-voltage (IV) relationship, as illustrated in Figure 5c. Figure 5d shows the memristor has an $Al/TiO_2/TiO_{2-x}/Al$ stack in cross-section. In fabrication, a typical memristor presents a very high resistance across its electrodes (an unformed state) and an initial one-time electroforming step is needed [39] for multilevel conductance. This can be done by applying voltage or current sweep across the two electrodes until a soft breakdown of the active layer occurs, generating a conductive filament that changes the conductance [91]. (see next section for more test details)



(a) (b) (c) (d)

Figure 5. Memristive crossbar arrays and device. (a) An optical image of a wafer with memristive crossbar arrays, (b) Close-up of chip image showing crossbar array, (c) Microscope image showing one memristor device, and (d) Cross-sectional schematic diagram of the TiO_2/TiO_{2-x} memristor device structure.

3.2. Memristor Test

The I-V characteristics from positive and negative voltage sweeping were carried out using a Kaysight B1500a semiconductor parameter analyzer in a voltage-sweep and voltage-pulse mode.

The wafer was set on the Micromanipulator probe station and the pads were contacted by gold probe tips with 50 Ω as sensing impedance as shown in Figure 6.

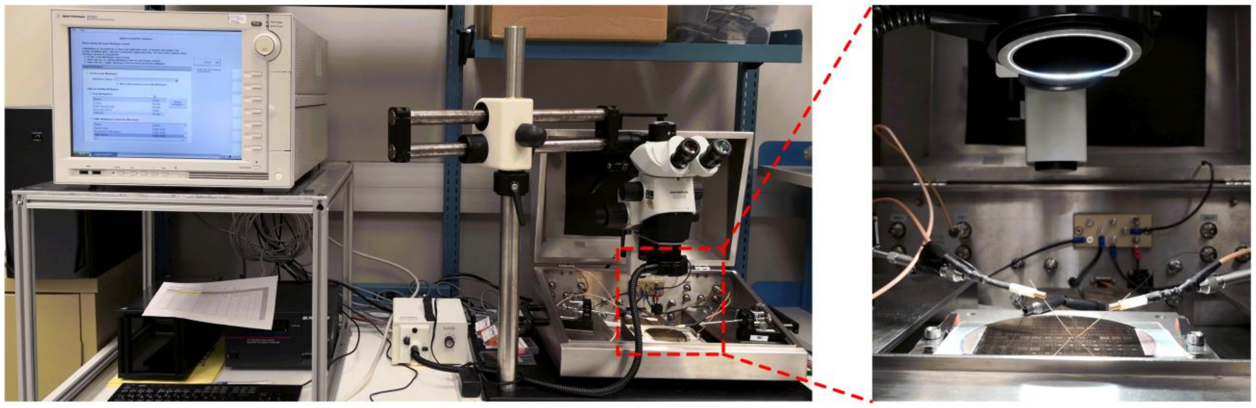


Figure 6. Testing platform.

This TiO_2/TiO_{2-x} memristor displays obvious multilevel conductance. Figure 7 shows the current-voltage response of the memristor when the full range voltage sweeps during different cycles. For further investigating this multilevel property, the positive and negative voltage sweeping is separately applied in the same memristor with ten cycles, as shown in Figure 8 and Figure 9. The conductance of the memristor is changed when the voltage achieves the threshold voltage, which is caused by conducting filament formation across the electrodes [92, 93]. Memristive crossbar arrays carry out the vector-matrix multiplication as shown in Figure 10. Every row of the crossbar array gets input voltage pulses that are the vector. Each conductance of the device in every cross point composes the matrix. Every column of the crossbar array transmits an output current that is the sum of multiplication by the input signal and conductance in each cross point. To update the conductance of a memristor that has multilevel conductance from the minimum to the maximum, a positive pulse signal is applied to increase the conductance, which is LTP. Conversely, LTD is the process of decreasing the conductance by supplying a negative pulse signal until the conductance gets to the minimum. Multilevel memristors effectively utilize such multi-value conductance to learn the features of data and realize a neuromorphic computing system [17, 18, 40].

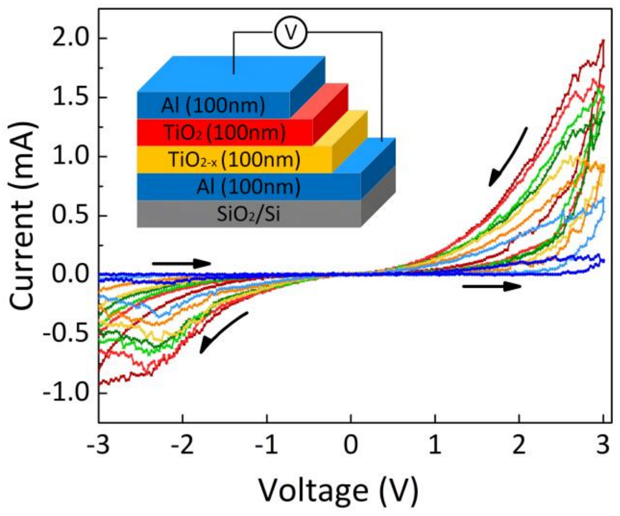


Figure 7. I-V characteristics (current-voltage response) from the full range voltage (-3 V to 3V) sweeps during different cycles in the memristor.

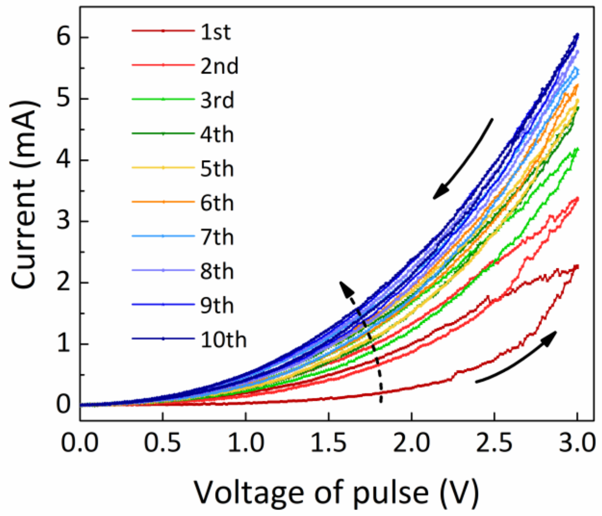


Figure 8. I-V characteristics from consecutive positive voltage pulse sweep showing a continuous increase in conductance. The width of pulses is 1 ms and the step of voltage is 1.5 mV.

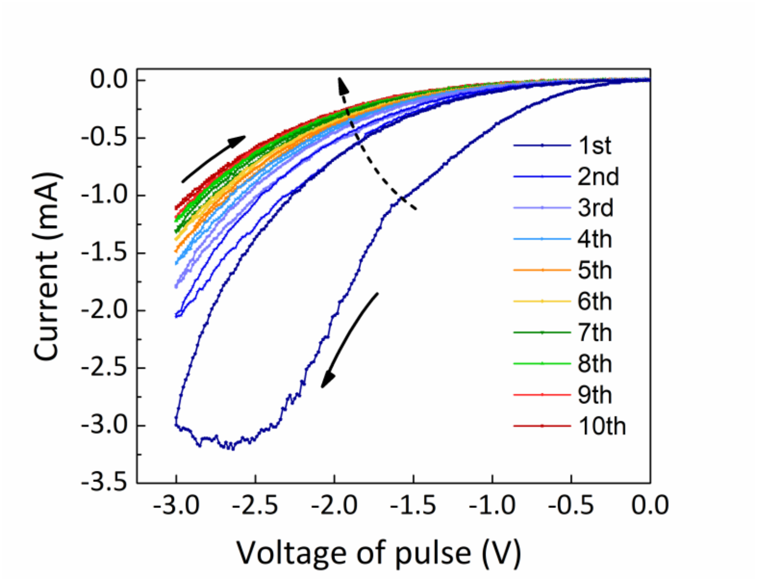


Figure 9. Consecutive negative voltage pulse sweep showing a continuous decrease in conductance. The width of pulses is 1 ms and the step of voltage is 1.5 mV.

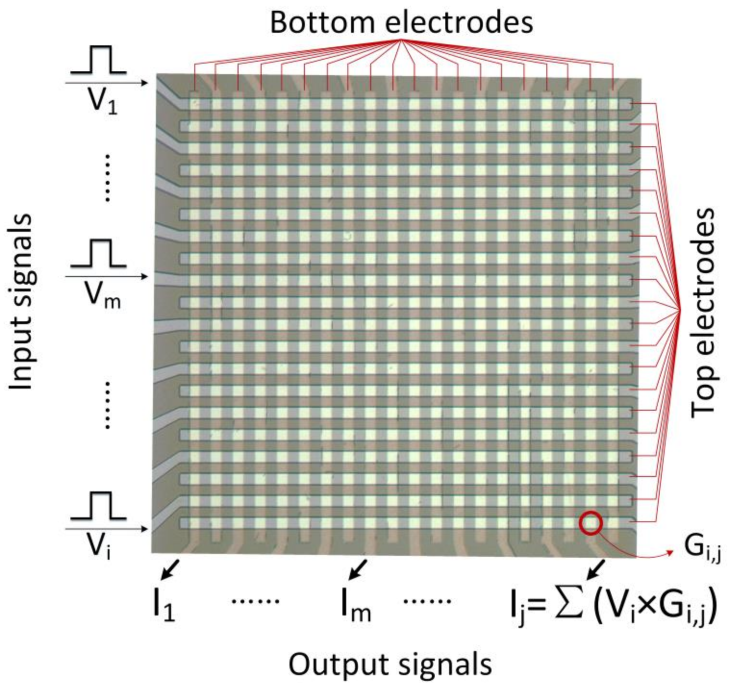


Figure 10. Hardware implementation of the vector-matrix-multiplication using memristor crossbar. V_i, G_{ij}, and I_j represent the input signal in i^{th} row, the conductance of the memristor in j^{th} column and i^{th} row, and the output current that represent the dot product result of V and G, respectively.

In practice, the width of the pulse signal that is used to update the weight cannot be infinitely narrow and limits the accuracy of the conductance updating. Different widths of the pulses change the different amounts of conductance. Therefore, the widths of different pulses that are used for weight update decide the number of the levels as shown in Figure 11. The number of these levels can be expressed qualitatively as equation (1):

$$[(G_{max} - G_{min})/ W_{pulse}] = \text{number of levels} \tag{1}$$

where G_{max} and G_{min} are the maximum conductance and the minimum conductance, W_{pulse} is the width of the updating pulse. Note that although a higher number of the levels gives more precise conductance in the weight update of the memristor, the influence of cycle-to-cycle variation will increase, which is shown in the next chapter.

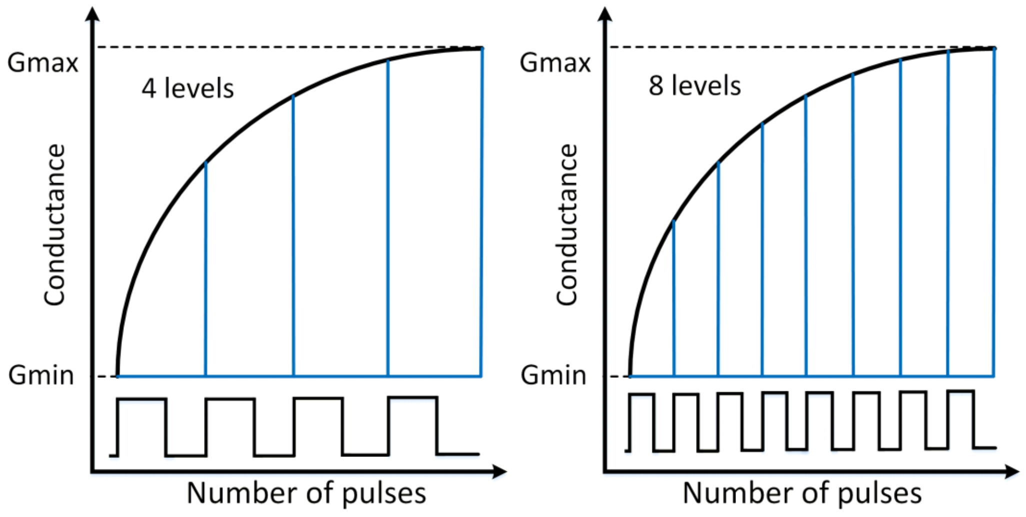


Figure 11. Level of conductance with different widths of the updating pulses.

3.3. Conclusion

At this point, the TiO_2/TiO_{2-x} 40 µm x 40 µm two-terminal memristor crossbar array that is composed of 20 x 20 devices is fabricated. This study on the behavior of fabricated memristor laid a solid foundation for the study of cycle-to-cycle variation and relative techniques. In particular, the fabricated memristor possesses multiple levels that are a typical characteristic of the analog device.

4. CYCLE-TO-CYCLE VARIATION QUANTIZATION

Although existing models for cycle-to-cycle variation can model the given experimental data, they are not suitable for our data. In this chapter, the test data from the previous chapter is analyzed in-depth for modeling the cycle-to-cycle variation and then get a significant conclusion that is a cornerstone for the rest study of this dissertation.

4.1. Test Data Analyzation

Due to the multilevel conductance of the memristor, Figure 12 shows the LTP and LTD processes with different pulse widths. We can fit these LTP and LTD experimental data with exponential formulas [63], as shown in Figure 12 fitting curves and Figure 13 for more details.

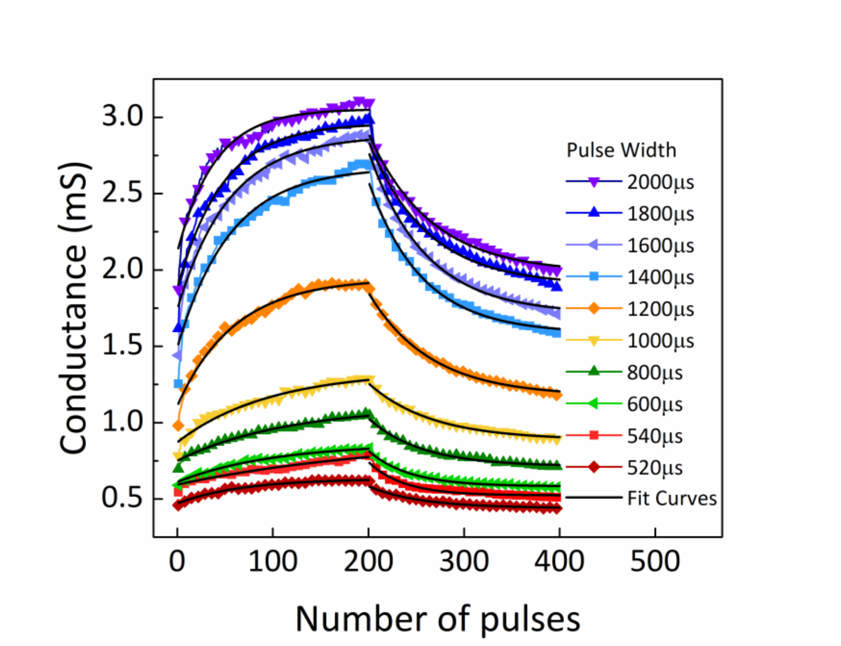
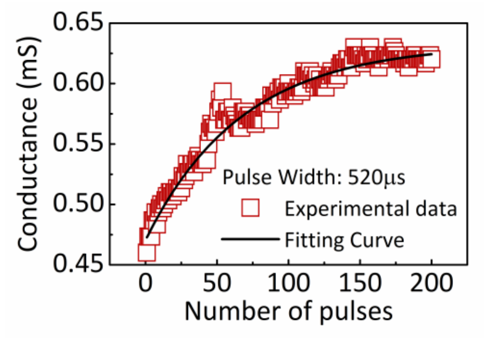
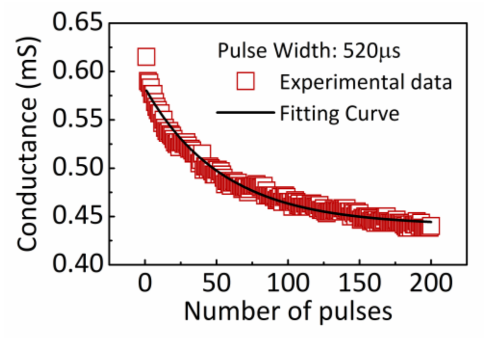


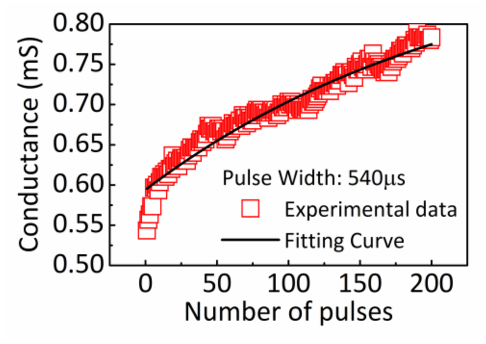
Figure 12. The LTP and LTD process with different pulse widths from 520 μs to 2000 μs. The black curves are fitted by the exponential formula.



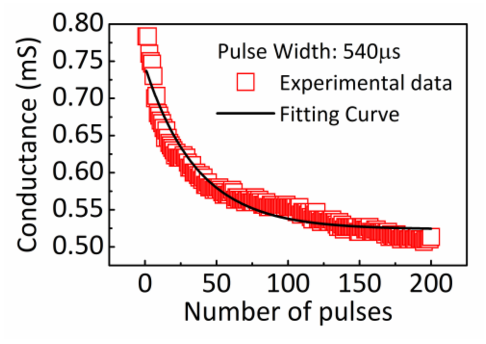
(a)



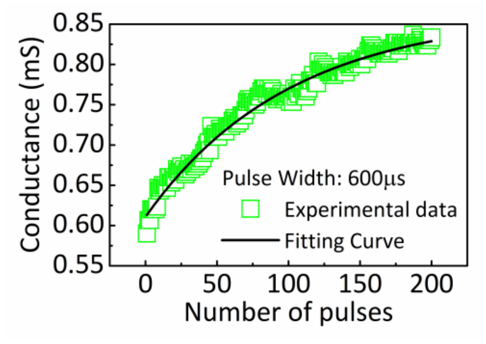
(b)



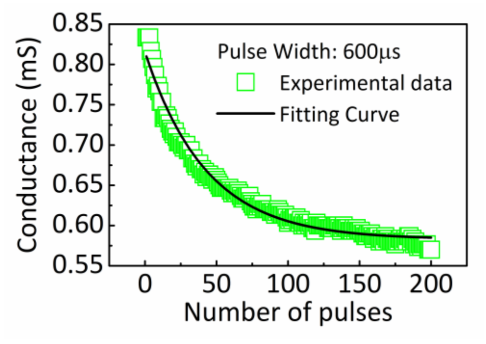
(c)



(d)

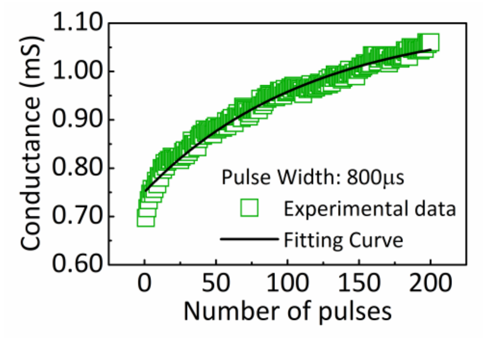


(e)

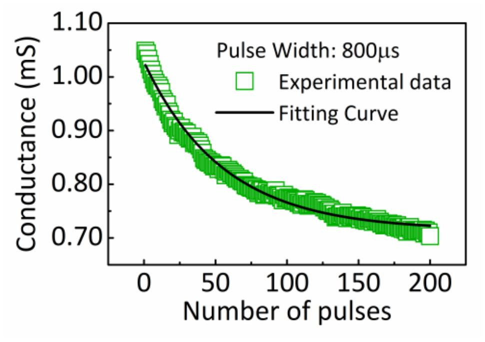


(f)

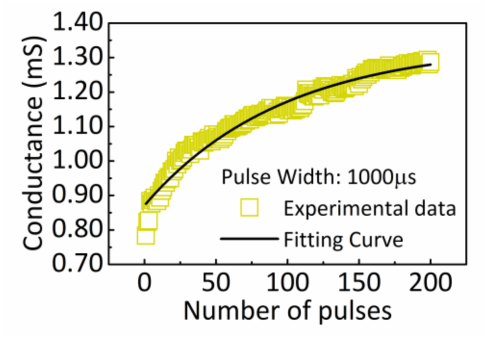
Figure 13. Exponential formula [63] fitting for the LTP/LTD with different width of pulses. From (a) to (t), the width of pulses is 520 μs, 540 μs, 600 μs, 800 μs, 1000 μs, 1200 μs, 1400 μs, 1600 μs, 1800 μs, and 2000 μs, respectively. One pulse width has LTP and LTD processes that are separated. The squire dots are experimental data and solid lines are fitting curves.



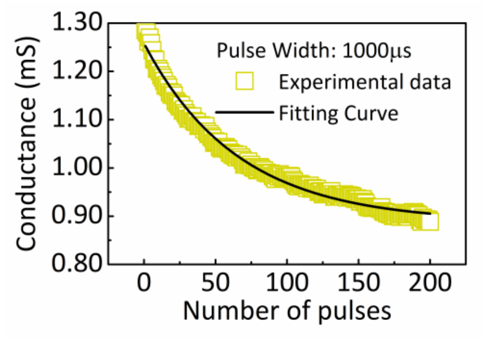
(g)



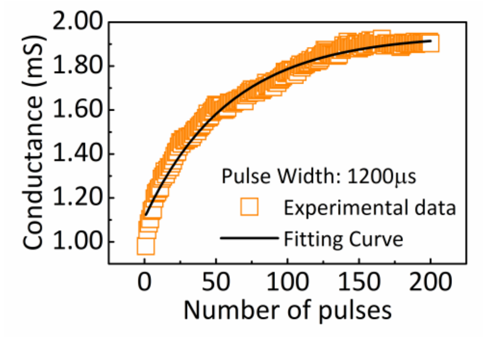
(h)



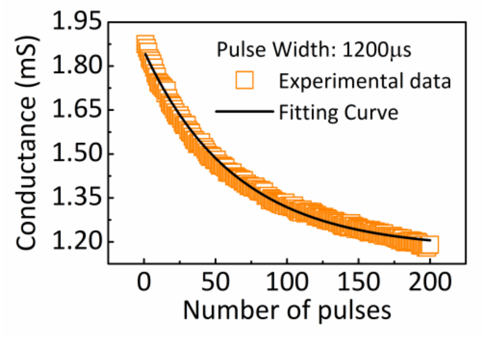
(i)



(j)

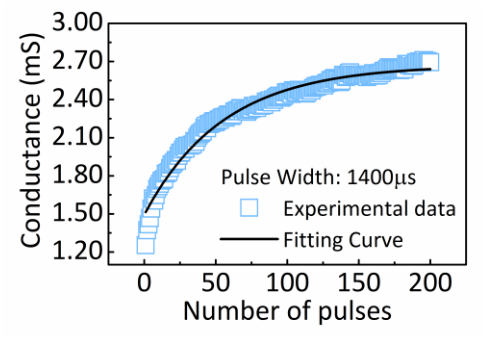


(k)

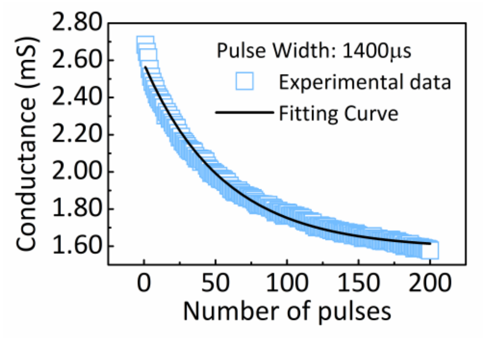


(l)

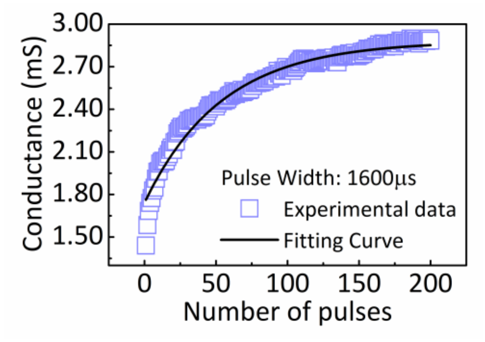
Figure 13. Exponential formula [63] fitting for the LTP/LTD with different width of pulses (continued). From (a) to (t), the width of pulses is 520 μs, 540 μs, 600 μs, 800 μs, 1000 μs, 1200 μs, 1400 μs, 1600 μs, 1800 μs, and 2000 μs, respectively. One pulse width has LTP and LTD processes that are separated. The squire dots are experimental data and solid lines are fitting curves.



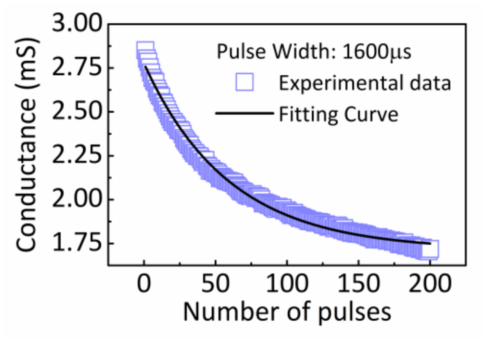
(m)



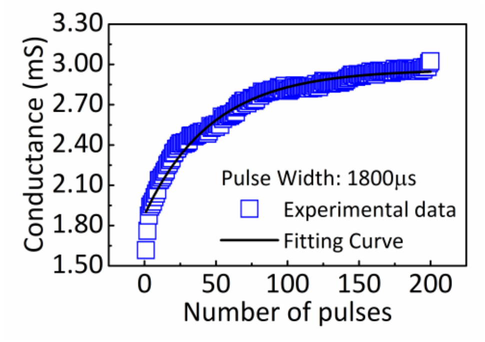
(n)



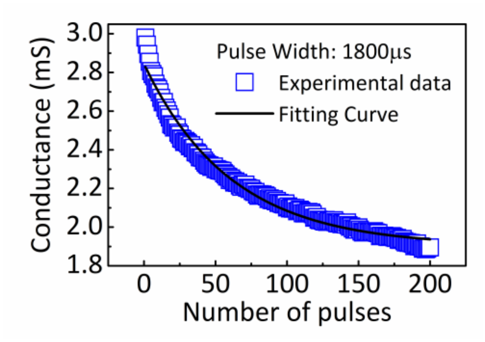
(o)



(p)

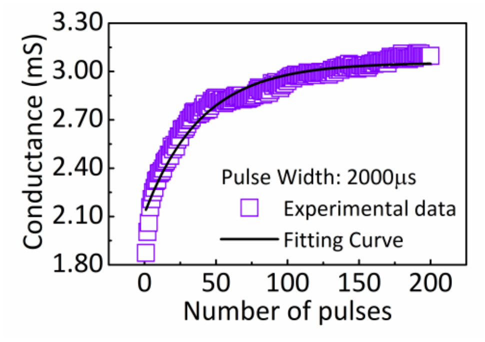


(q)

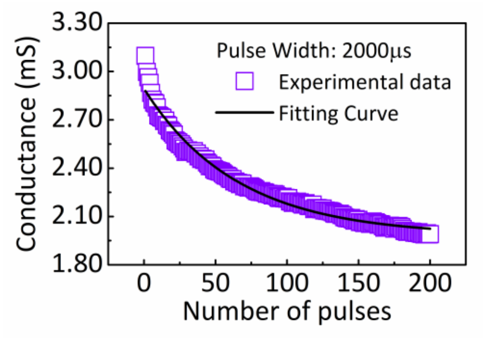


(r)

Figure 13. Exponential formula [63] fitting for the LTP/LTD with different width of pulses (continued). From (a) to (t), the width of pulses is 520 µs, 540 µs, 600 µs, 800 µs, 1000 µs, 1200 µs, 1400 µs, 1600 µs, 1800 µs, and 2000 µs, respectively. One pulse width has LTP and LTD processes that are separated. The squire dots are experimental data and solid lines are fitting curves.



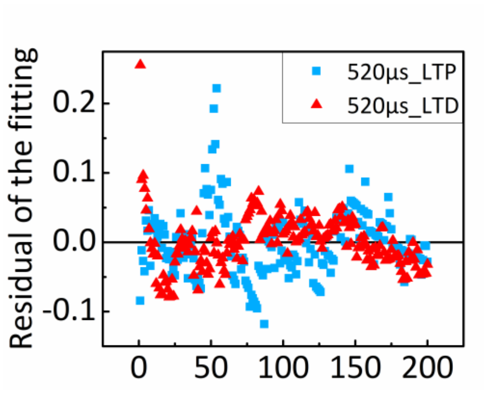
(s)



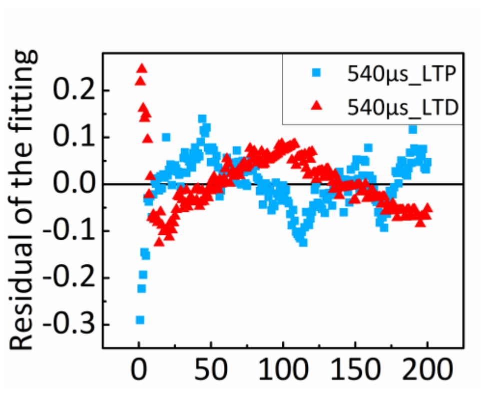
(t)

Figure 13. Exponential formula [63] fitting for the LTP/LTD with different width of pulses (continued). From (a) to (t), the width of pulses is 520 µs, 540 µs, 600 µs, 800 µs, 1000 µs, 1200 µs, 1400 µs, 1600 µs, 1800 µs, and 2000 µs, respectively. One pulse width has LTP and LTD processes that are separated. The squire dots are experimental data and solid lines are fitting curves.

Residual analysis is done after exponential formula fitting as shown in Figure 14. We normalized the residual data by dividing the difference of maximum conductance and minimum conductance corresponding width of pulses.

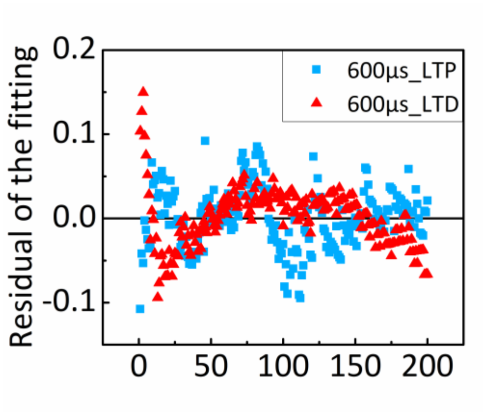


(a)

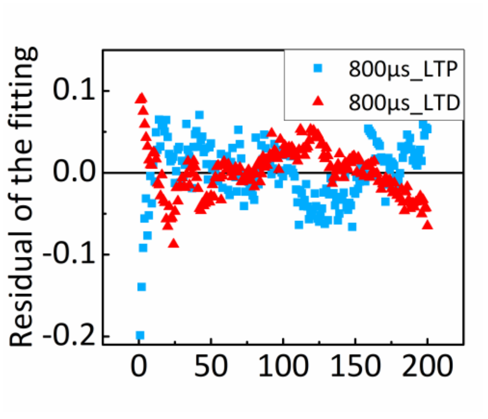


(b)

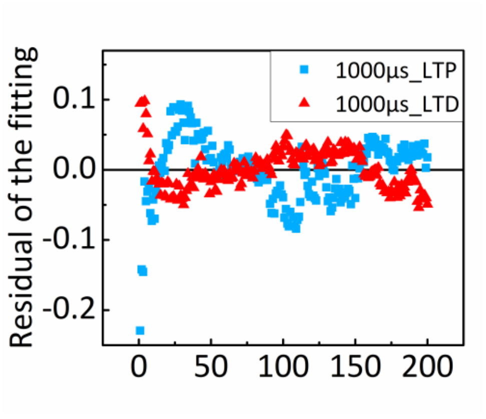
Figure 14. Residual data after exponential formula fitting. Every single point of residual was plotted with the order of the number of points as an X-axis. The squire points are the residual value of the LTP and the triangle points are the residual value of the LTD with different widths of pulses. From (a) to (j), the width of pulses is 520 µs, 540 µs, 600 µs, 800 µs, 1000 µs, 1200 µs, 1400 µs, 1600 µs, 1800 µs, and 2000 µs, respectively.



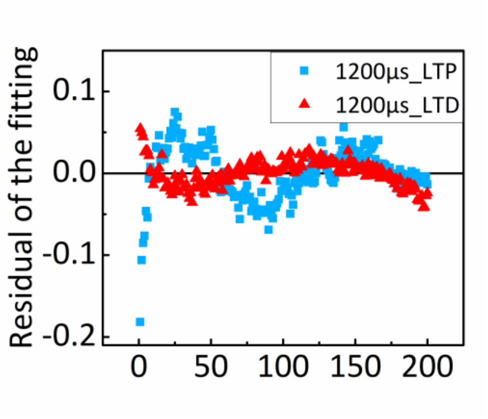
(c)



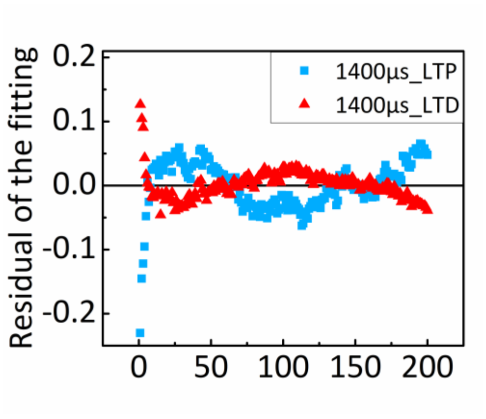
(d)



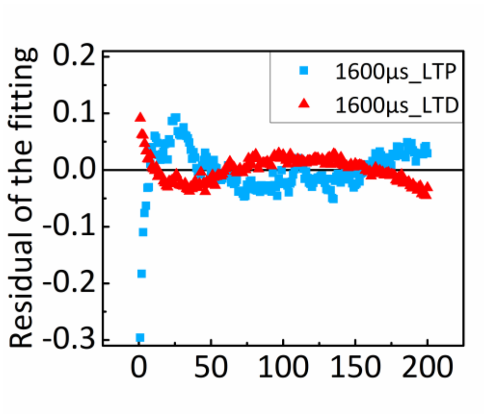
(e)



(f)

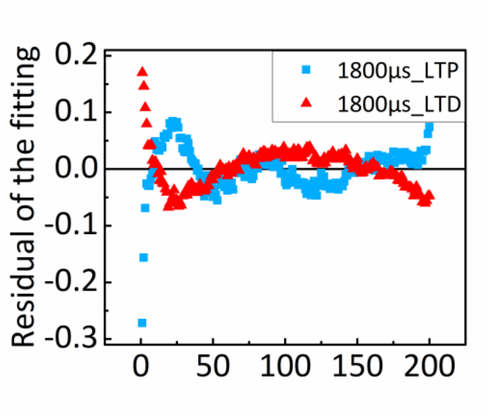


(g)

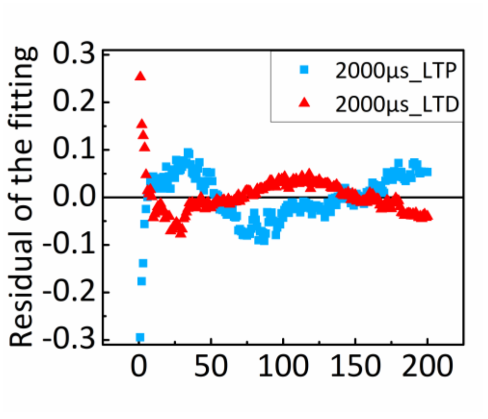


(h)

Figure 14. Residual data after exponential formula fitting (continued). Every single point of residual was plotted with the order of the number of points as an X-axis. The squire points are the residual value of the LTP and the triangle points are the residual value of the LTD with different widths of pulses. From (a) to (j), the width of pulses is 520 μs, 540 μs, 600 μs, 800 μs, 1000 μs, 1200 μs, 1400 μs, 1600 μs, 1800 μs, and 2000 μs, respectively.



(i)



(j)

Figure 14. Residual data after exponential formula fitting (continued). Every single point of residual was plotted with the order of the number of points as an X-axis. The squire points are the residual value of the LTP and the triangle points are the residual value of the LTD with different widths of pulses. From (a) to (j), the width of pulses is 520 µs, 540 µs, 600 µs, 800 µs, 1000 µs, 1200 µs, 1400 µs, 1600 µs, 1800 µs, and 2000 µs, respectively.

Because the fitted curve and stochastic behavior of the cycle-to-cycle variation can be approximated with a normal distribution [35, 36, 94], the residual data is normal distribution fitted after normalization, as illustrated in Figure 15 and Figure 16 for more details.

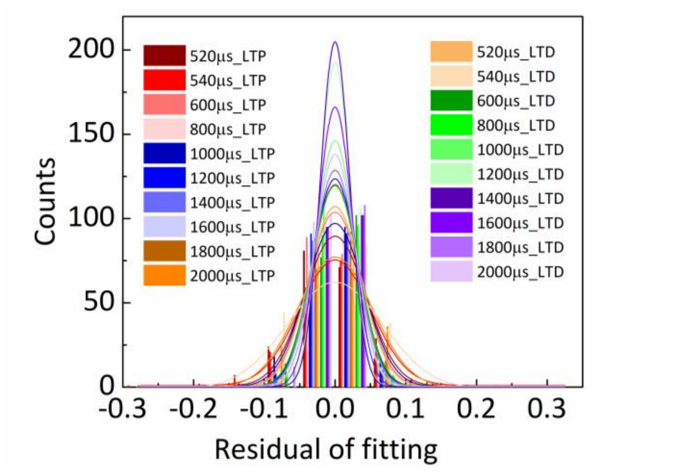
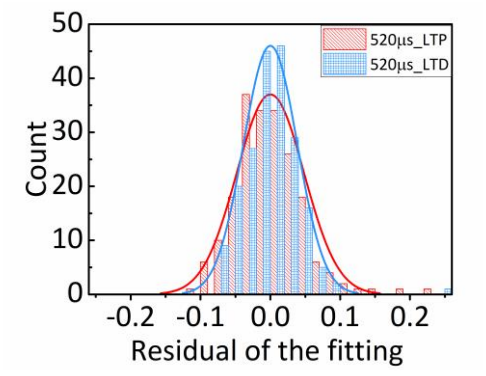
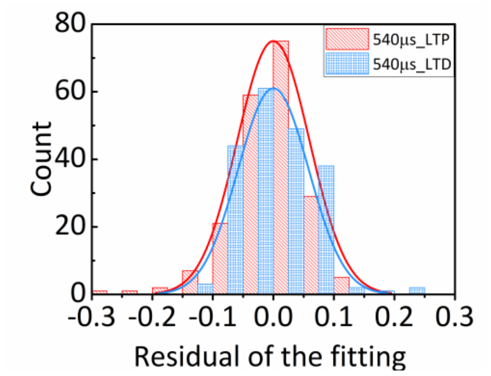


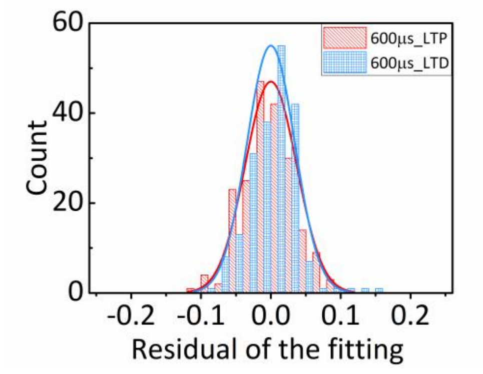
Figure 15. Residual analysis of fitted normal distribution data after normalization of the deviation from Figure 14 in different pulse widths.



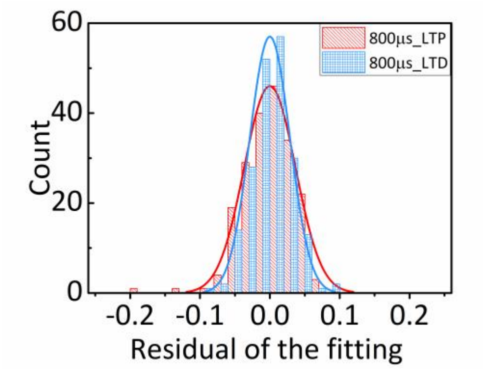
(a)



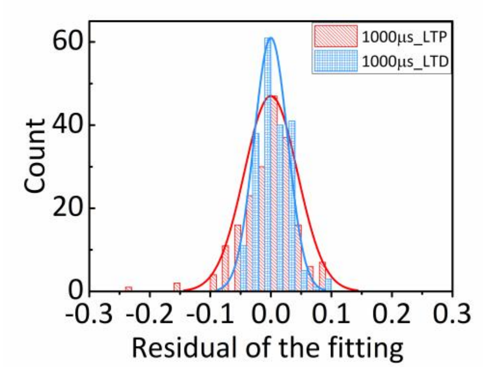
(b)



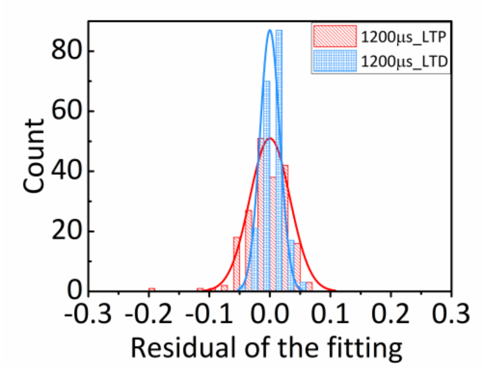
(c)



(d)

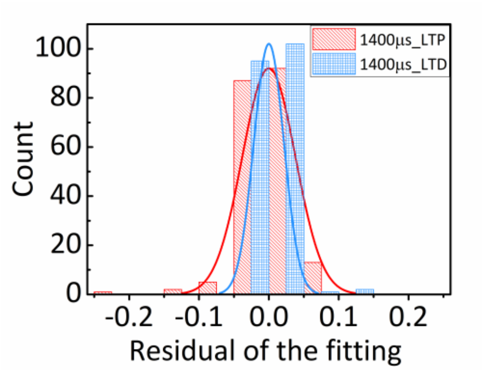


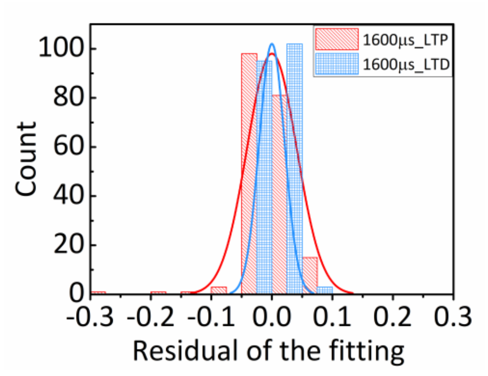
(e)



(f)

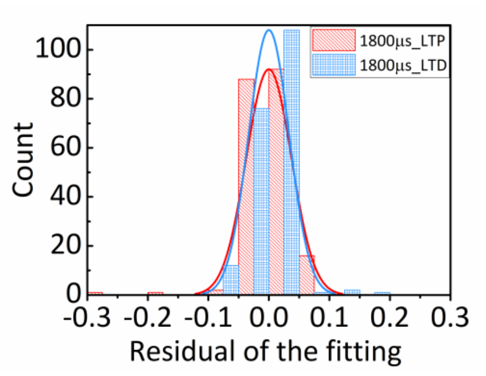
Figure 16. Normal distribution analysis for the residual data after exponential formula fitting. The red and blue lines are normal distribution fitting curves that are LTP and LTD, respectively. In this way, the relative parameters, such as standard deviation, can be obtained. From (a) to (j), the width of pulses is 520 μs, 540 μs, 600 μs, 800 μs, 1000 μs, 1200 μs, 1400 μs, 1600 μs, 1800 μs, and 2000 μs, respectively.

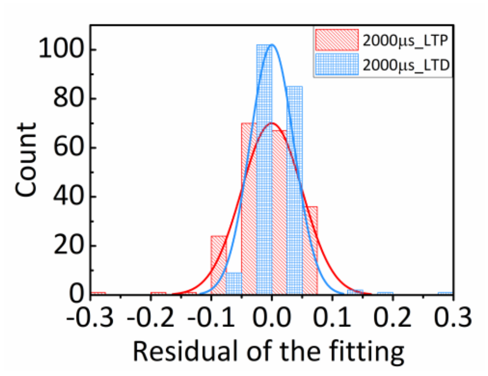




(g)

(h)





(i)

(j)

Figure 16. Normal distribution analysis for the residual data after exponential formula fitting (continued). The red and blue lines are normal distribution fitting curves that are LTP and LTD, respectively. In this way, the relative parameters, such as standard deviation, can be obtained. From (a) to (j), the width of pulses is 520 µs, 540 µs, 600 µs, 800 µs, 1000 µs, 1200 µs, 1400 µs, 1600 µs, 1800 µs, and 2000 µs, respectively.

4.2. Model for Cycle-to-cycle Variation

Because stochastic behavior of the cycle-to-cycle variation can be approximated as Gaussian distribution, the value of cycle-to-cycle variation corresponding to one pulse can be defined as

$$\psi = N(0,\ \delta) \tag{2}$$

where ψ is cycle-to-cycle variation that is generated for one pulse update process, $N(0,\ \delta)$ is Gaussian noise, and δ is the standard deviation. The value of cycle-to-cycle variation corresponding to two pulses can be calculated as

$$\psi_{1+2} = N_1(0,\ \delta_1) +\ N_2(0,\ \delta_2) \tag{3}$$

Because different cycle-to-cycle variation obey Gaussian distributions and that are independent and identically distributed corresponding to different one pulse, $\delta_1 = \delta_2 = \delta$, the equation (2) can be converted as [95]

$$\psi_{1+2} = N(0,\ \delta_1 +\ \delta_2) = N(0,\ 2\delta) = N(0,\ \delta) * \sqrt{2} \tag{4}$$

Therefore, the total cycle-to-cycle variation that is generated for one memristor at one update process with n pulses can be calculated as

$$\psi_{total} = N(0,\ \delta) *\ \sqrt{n} \tag{5}$$

$$\delta = \alpha * (G_{max} - G_{min}) \tag{6}$$

where ψ_{total} is the total cycle-to-cycle variation that is generated for one memristor at one update process, $N(0,\ \delta)$ is Gaussian noise, n is the number of pulses for this memristor at this update process, α is coefficient that is a percentage of difference of the maximum conductance and the minimum conductance, G_{max} is the maximum conductance, and G_{min} is the minimum conductance.

According to the normal distribution, the coefficient, α, can be obtained and shown in Table 2. Note that, this α value is determined by structure and material of device. The values with different type of memristors can be obtained by this testing and analyzing method.

Table 2. The Coefficient Values (α) with Different Width of Pulses.

Width of pulse (μs)	α (LTP)	α (LTD)
520	0.04354	0.03712
540	0.0506	0.06754
600	0.03846	0.03279
800	0.03714	0.03109
1000	0.04077	0.02746
1200	0.03191	0.02173
1400	0.03278	0.02064
1600	0.03277	0.02444
1800	0.0328	0.03045
2000	0.05307	0.02825
Average	0.03577	

4.3. Conclusion

After Gaussian distribution fitting, we get a distribution of α values with different pulse widths such that the average is 0.03577, as shown in Figure 17. The lines are linear fitting for α values of LTP and LTD. Both slopes are negative. Therefore, it can be concluded that increasing the pulse width does not increase the cycle-to-cycle variation when using the same number of pulses to tune conductance. This conclusion provides an experimental and theoretical basis for subsequent research on mitigating the effect of the cycle-to-cycle variation.

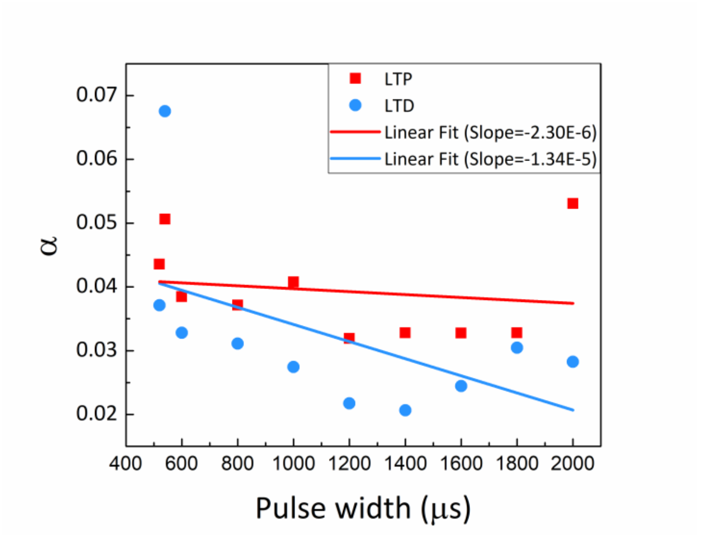


Figure 17. Extraction of the coefficient that is between 0 to 1 from the standard deviation of the Gaussian distribution fitting from Figure 16 in different pulse widths.

5. MITIGATING THE EFFECT OF THE CYCLE-TO-CYCLE VARIATION

We already knew that cycle-to-cycle variation is a type of inherent random mechanism associated with the randomness in internal atomic configurations. One of the major obstacles to the implementation of redox-based multilevel memristive memory or logic technology is the large cycle-to-cycle variation. In this chapter, two techniques are proposed to mitigate the effect of cycle-to-cycle variation.

5.1. Level Scaling Method

Theoretically, the prediction accuracy is higher if the system utilizes a higher number of conductance levels [63, 96]. However, this relationship is broken by the cycle-to-cycle variation - sometimes a higher number of the levels yields diminishing accuracy. The number of levels of memristor conductance is set in the circuit parameter configuration step, to determine how many levels can be obtained between the maximum and minimum conductance, inclusively. The specified number of levels will determine the width of the pulse output from the pulse generator hardware. A larger number of levels corresponds to narrower pulses, which theoretically would allow the system to achieve higher weight precision for a given value of conductance; however, a larger number of levels also introduces more cycle-to-cycle variation when the system updates conductance because more pulses are required to update conductance for the same delta-weight, compared to a system with fewer conductance levels. Therefore, the level scaling method is applied, as described next section, to appropriately select the number of levels to achieve maximum accuracy.

5.2. Evaluation Methodology and Results

In order to evaluate the memristor-based crossbar arrays in the different number of the levels and to find optimal conductance levels under the cycle-to-cycle variation, the multilayer perceptron platform (MLP platform) is used to emulate the learning classification scenario with the Modified National Institute of Standards and Technology (MNIST) handwritten dataset [63]. We adopt the hardware platform, NeuroSim+ [55, 63], to perform handwriting recognition as shown in Figure 18 for the hardware implementation and Figure 19 for the processing flow chart.

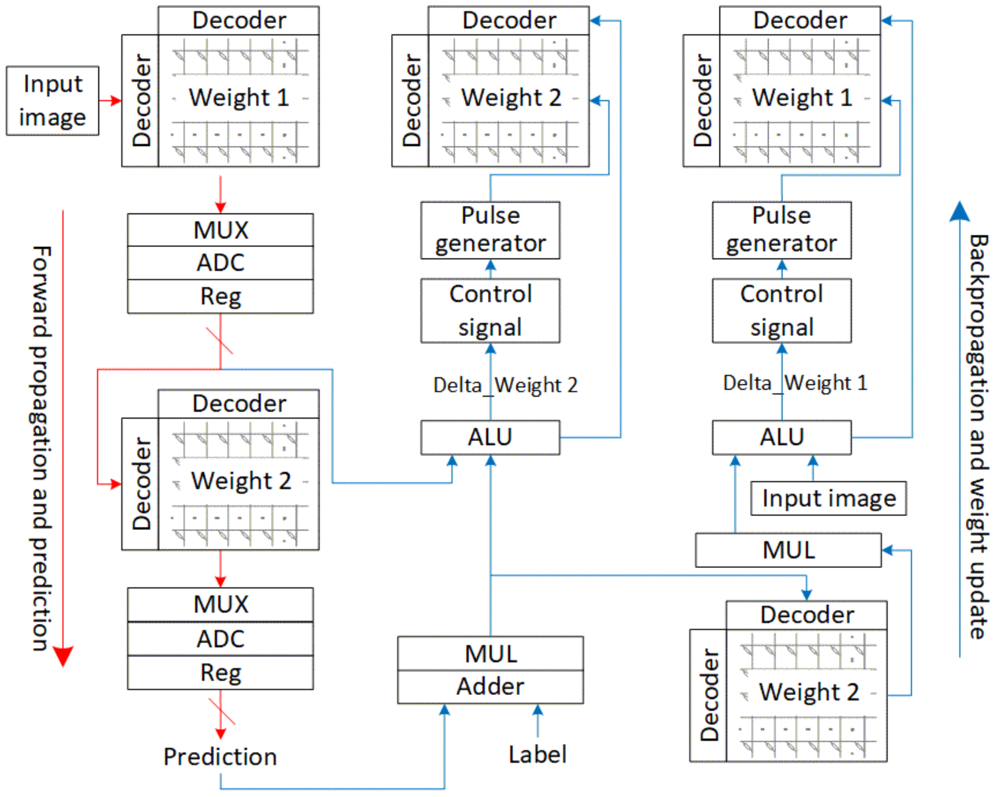


Figure 18. Hardware implementation of the multilayer perceptron platform.

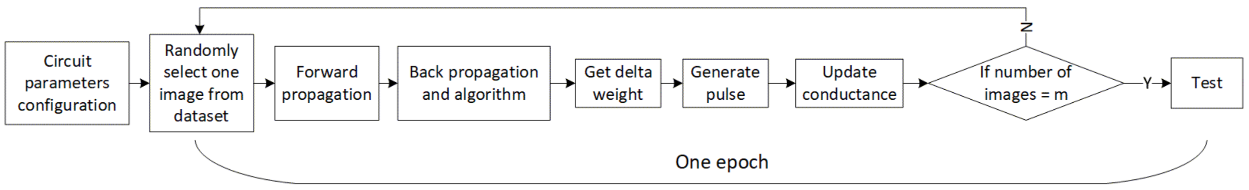


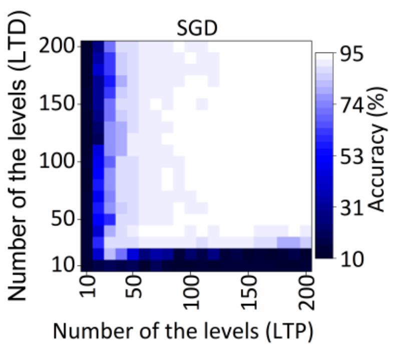
Figure 19. Working flow of the multilayer perceptron platform. An optimized number of the level value and measured cycle-to-cycle variation are set at the circuit parameters configuration step. m equals 8000.

The crossbar array architecture with memristors had been proposed for on-chip implementation of weighted sum and weight update in the training process of learning algorithms

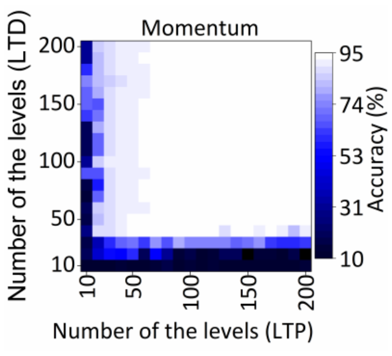
[59]. This platform contains a three-layer with 400 neurons for the input layer, 100 neurons for the hidden layer, and 10 neurons for the output layer. The perceptron neural network is simulated that basis on memristive crossbar arrays that refers to a special subset of the memristor that can tune the conductance by voltage pulse stimulus. The desired weight update for each layer is calculated in software [7], then applied to the crossbar by the system as illustrated in Figure 18 for the hardware implementation block diagram, and in Figure 19 for the processing flow chart. For the level scaling method, each evaluation trains 125 epochs, and every epoch randomly selects 8000 images from 60000 training images. A different set of 10000 images are included in the testing dataset. Note that, the networks will continually learn the feature of input data after the last epoch since this platform is an online learning network [55]. In this platform, parameters of memristor come from the measurement results of our fabricated devices. In summary, this MLP platform that combines from device level to algorithm level system in neural networks is a standalone functional platform that is able to evaluate the learning accuracy and device-level performance during the learning process.

The Stochastic Gradient Descent (SGD) algorithm is one possible solution to accelerate the gradient descent process to use approximate methods that goes through the data in samples composed of random examples drawn from the original dataset [7, 11], which is the major algorithm used in this simulator. In fact, SGD is a rough approximation, producing a non-smooth convergence. Because of that, variants were proposed to compensate, such as the Momentum [97], Adaptive Gradient (AdaGrad) [98], Root Mean Square Prop (RMSProp) [99], and Adaptive Moment Estimation (Adam) [100], which are included in this simulator.

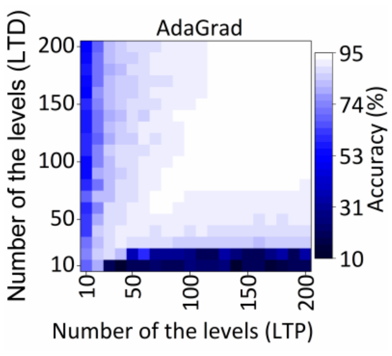
To study the relationship between the number of the levels and cycle-to-cycle variation, the different number of the level for the LTP and LTD is set. The ideal circumstances (α = 0) with the number of the levels from 10 to 200 and step 10 are set with five algorithms as shown in Figure 20. When the cycle-to-cycle variation is not involved, with the increasing number of the level, the accuracy goes up to the high area (bright area) from the low area (dark area), where the highest accuracy appears at the number of the levels = 200 at LTP and LTD (upper right corner). It can be concluded that increasing the number of levels does increase the recognition accuracy without the cycle-to-cycle variation. In bright areas of the figures, the recognition accuracies are around 90% in the lower-left corner and higher than 93% in the upper right corner.



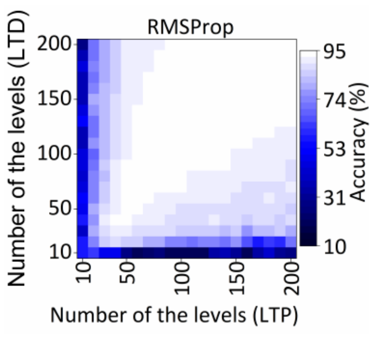
(a)



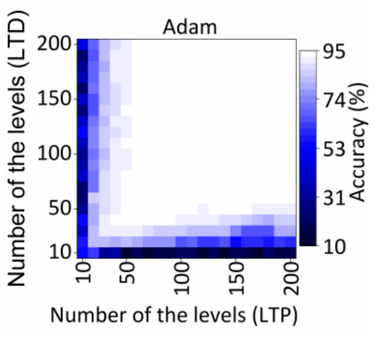
(b)



(c)



(d)



(e)

Figure 20. Recognition accuracy without cycle-to-cycle variation (α = 0) with different LTP and LTD number of the levels (from 10 to 200, step is 10) in 5 algorithms. The x-axis is number of LTP levels and y-axis is number of LTD levels. The color from dark to bright represents the system prediction accuracy from low to high. From (a) to (e), the corresponding algorithm is SGD, Momentum, AdaGrad, RMSProp, and Adam, respectively.

Figure 21 depicts the realistic case that includes a cycle-to-cycle variation for the 5 weight updating algorithms, using the average α value of 0.03577, calculated in Chapter 4. The added cycle-to-cycle variation reduces accuracy, and the bright areas where accuracies are higher than 88% are much smaller than in the ideal case. The highest accuracies occur at LTP/LTD levels of 50/40 for SGD, 60/50 for Momentum, 60/50 for AdaGrad, 50/50 for RMSProp, and 50/40 for Adam, resulting in maximum accuracies of 89.2%, 91.4%, 90.0%, 91.0%, and 91.7%, respectively. Therefore, when cycle-to-cycle variation is considered, the best performance of memristor-based neuromorphic computing systems occurs when the number of LTP and LTD levels are far fewer than the maximum. This level scaling method is used to optimize the number of levels so that the system achieves maximum recognition accuracy.

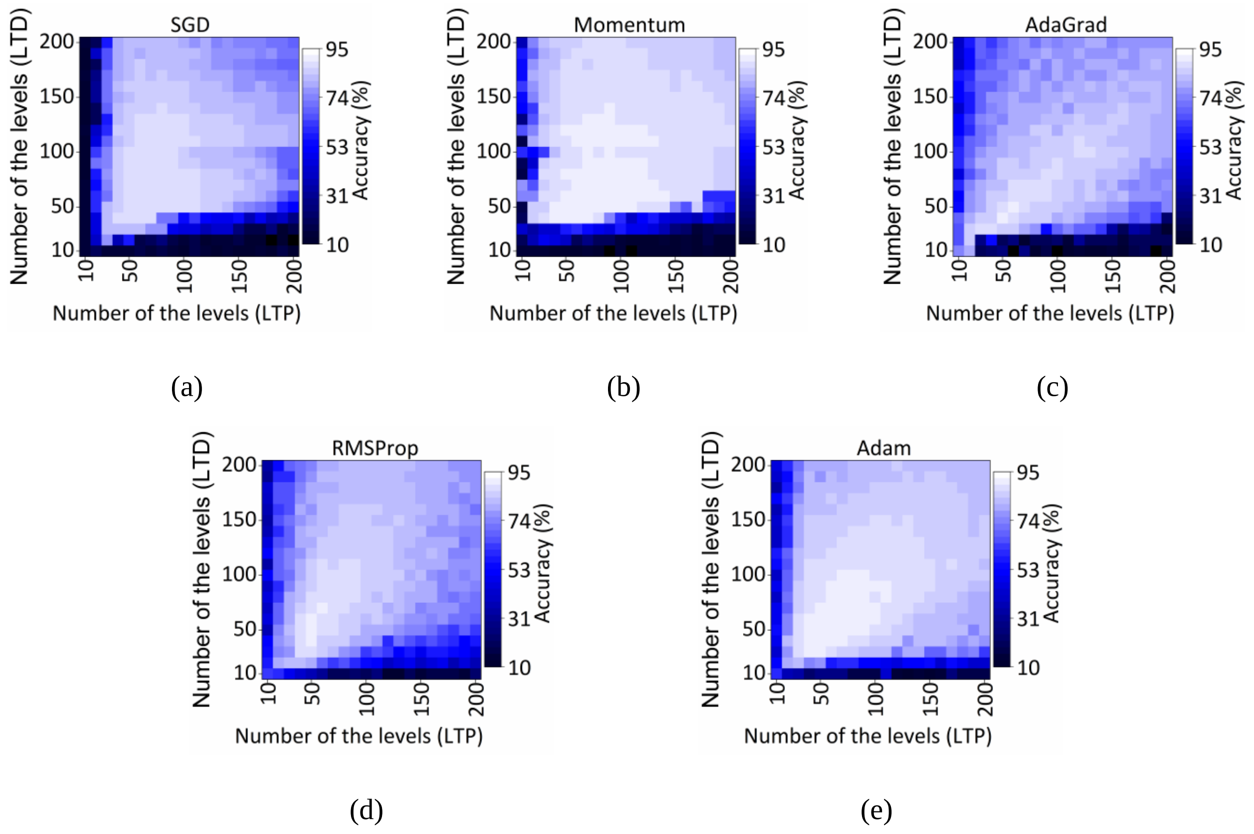
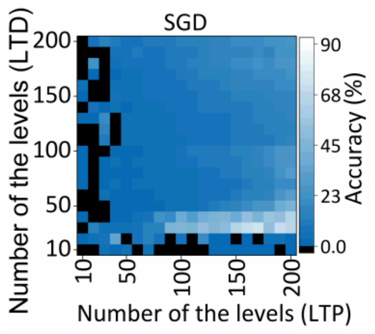


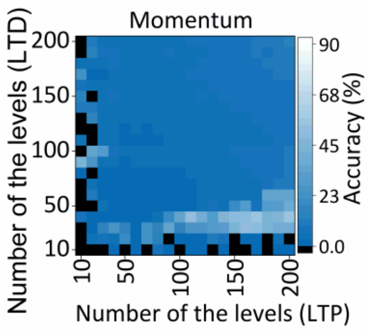
Figure 21. Recognition accuracy with different LTP and LTD number of the level values (from 10 to 200, step is 10) in 5 algorithms under measured cycle-to-cycle variation (α = 0.03577). The x-axis is number of LTP levels and y-axis is number of LTD levels. The color from dark to bright represents the system prediction accuracy from low to high. From (a) to (e), the corresponding algorithm is SGD, Momentum, AdaGrad, RMSProp, and Adam, respectively.

For a given memristive crossbar array, the distribution of cycle-to-cycle variation can be modeled by equation (5). At the same time, according to Figure 11, a lower number of the levels means larger conductance change between two consecutive pulses, and the system uses wider and fewer updating pulses for the same weight change that is calculated through any machine learning algorithm. According to experiment results and equation (5), the wider pulse does not increase the cycle-to-cycle variation and fewer updating pulses correspond to a smaller n, which reduces the cycle-to-cycle variation. Therefore, level scaling is an effective method to mitigate the effect of cycle-to-cycle variation. Note that, an extremely low number of levels will influence the accuracy of the conductance, which means some desired values of conductance cannot be achieved as shown in Figure 11, so reducing the precision of the system. This influence also is reflected by the low recognition accuracy as shown in the low (dark) number of the levels area of Figure 20. Thereby, for multilevel memristive crossbar arrays that are used in machine learning systems, the highest recognition accuracy of the system occurs when the memristor uses an optimized number of levels rather than the highest number of levels.

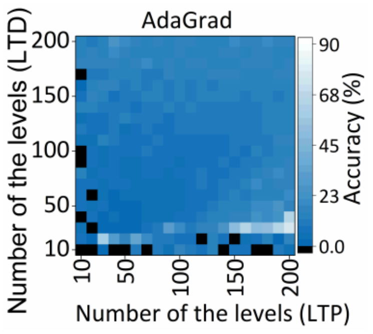
In a further comparison of Figure 20 and Figure 21, some accuracies with cycle-to-cycle variations and with certain LTP/LTD levels are higher than those without cycle-to-cycle variation as shown in Figure 22.



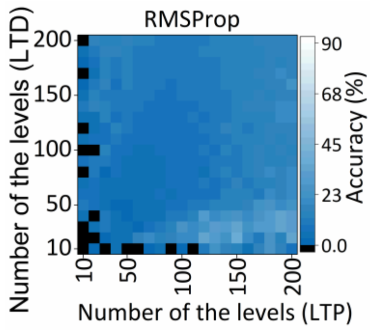
(a)



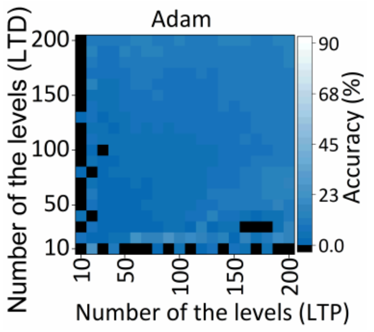
(b)



(c)



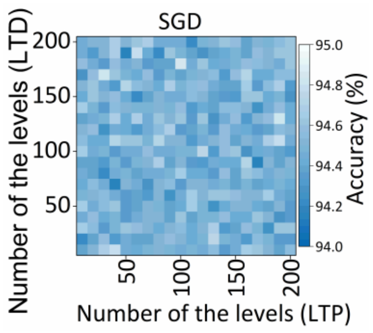
(d)



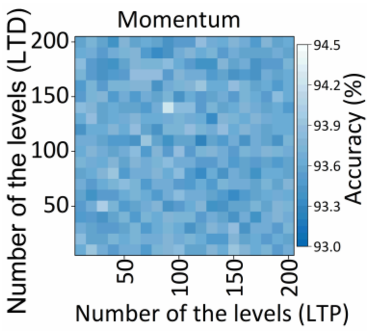
(e)

Figure 22. The difference of accuracy between ideal case (α=0) and that with cycle-to-cycle variation (α=0.03577). The black squares represent the negative values that mean the recognition accuracy with cycle-to-cycle variation is higher than that without the cycle-to-cycle variation. The x-axis is number of LTP levels and y-axis is number of LTD levels. The color from dark to bright represents the system prediction accuracy from low to high. From (a) to (e), the corresponding algorithm is SGD, Momentum, AdaGrad, RMSProp, and Adam, respectively.

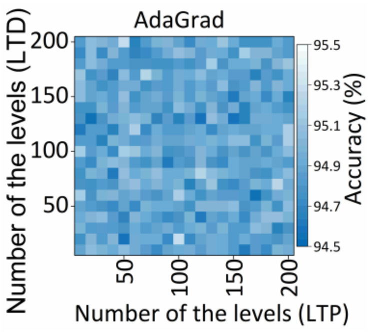
This is because only an integer number of pulses are generated in the circuit. As for the mechanism of the updating process, the amount of conductance that is increased or decreased will be calculated by the algorithm, then the accurate number of pulses is gotten accordingly. However, at the circuit level (hardware), only an integer number of pulses is available to update the conductance. Hence, the truncation function for an integer number of pulses is employed for hardware implementation. The updated weight gets the deviation by an integer number of pulses. But when the cycle-to-cycle variation is involved in every weight update, in some cases, they make the updated weight achieve closer to the accurate weight that the algorithm requires, and then the system gets even higher recognition accuracy. To proof this hypothesis, the simulations without the truncation function are done and as shown in Figure 23, Figure 24, and Figure 25.



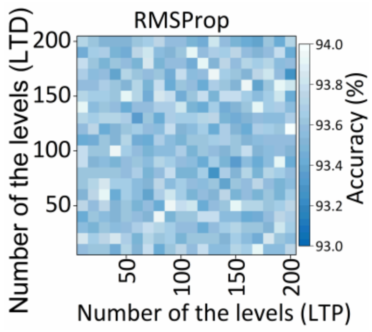
(a)



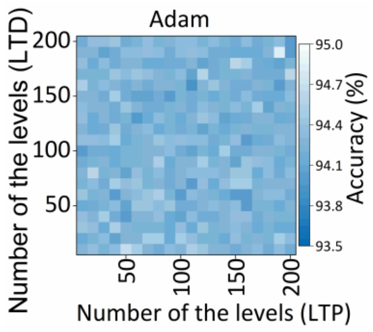
(b)



(c)

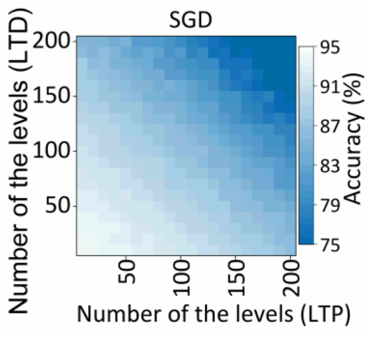


(d)

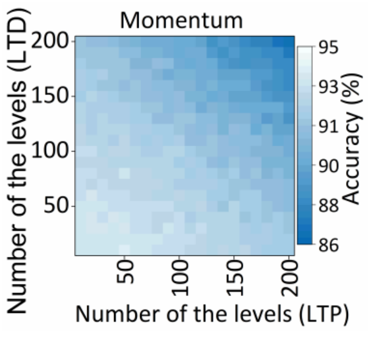


(e)

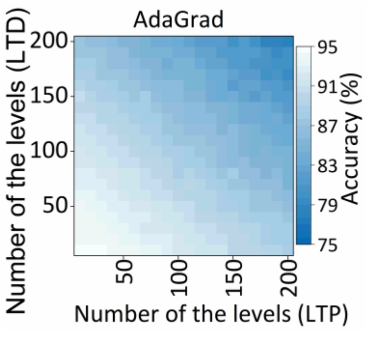
Figure 23. Recognition accuracy without cycle-to-cycle variation (α = 0) with different LTP and LTD number of levels values (from 10 to 200, step is 10) in 5 algorithms. The x-axis is number of LTP levels and y-axis is number of LTD levels. The color from dark to bright represents the system prediction accuracy from low to high. From (a) to (e), the corresponding algorithm is SGD, Momentum, AdaGrad, RMSProp, and Adam, respectively.



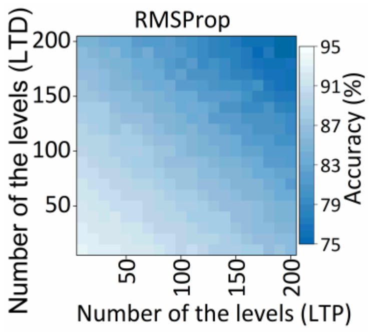
(a)



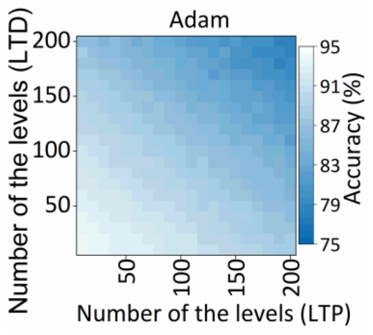
(b)



(c)

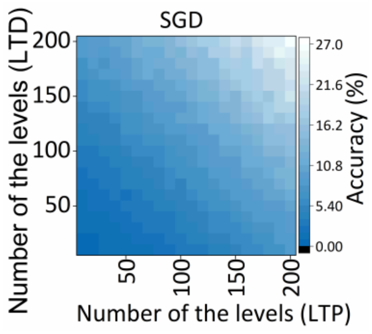


(d)

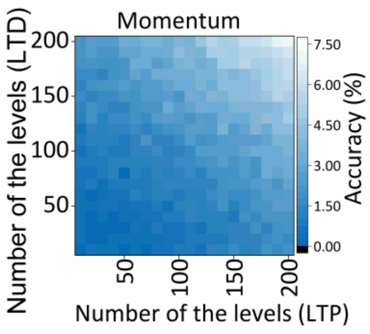


(e)

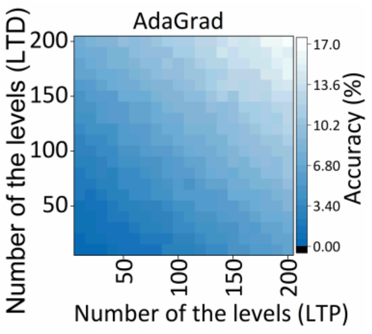
Figure 24. Recognition accuracy with a different number of the levels (from 10 to 200, step is 10) in 5 algorithms under measured cycle-to-cycle variation (α = 0.03577). The x-axis is number of LTP levels and y-axis is number of LTD levels. The color from dark to bright represents the system prediction accuracy from low to high. From (a) to (e), the corresponding algorithm is SGD, Momentum, AdaGrad, RMSProp, and Adam, respectively.



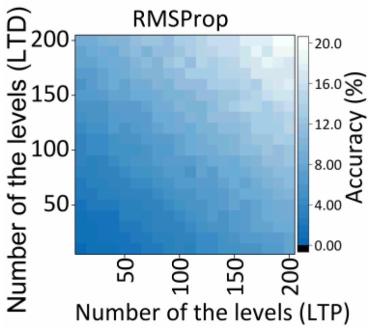
(a)



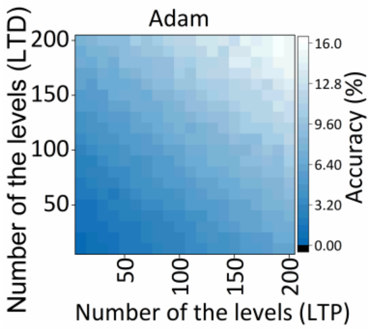
(b)



(c)



(d)



(e)

Figure 25. The difference of accuracy between ideal case (α=0) and that with cycle-to-cycle variation (α=0.03577). The black squares represent the negative values. The x-axis is number of LTP levels and y-axis is number of LTD levels. The color from dark to bright represents the system prediction accuracy from low to high. From (a) to (e), the corresponding algorithm is SGD, Momentum, AdaGrad, RMSProp, and Adam, respectively.

In Figure 23, the truncation function for converting delta-weight to an integer number of pulses is disabled, which means the number of pulses can be a decimal number. Therefore, the number of the levels doesn't limit the accuracy of weight that the system desires to reach. Thus, all of the predictions are higher than 93% and there is no difference between a low number of the levels and a high number of the levels. In Figure 24, when cycle-to-cycle variation is involved, although the number of update pulses can be decimal and the desired value of weight still can be reached, according to equation (5), a higher number of the levels introduces more cycle-to-cycle variation even when the system has the same delta-weight that needs to be updated. Therefore, the recognition of accuracy is higher when the system has a lower number of levels than that with a higher number of levels. In Figure 25, note that, all of the difference in recognition accuracies are positive, which means the accuracy in the ideal case is higher than that with cycle-to-cycle variations.

Thereby, with the truncation function, the cycle-to-cycle variation compensates for the decrease of the accuracy by truncating the number of update pulses. That is why some accuracies with cycle-to-cycle variation and with certain LTP/LTD levels in Figure 21 are higher than those without cycle-to-cycle variation in Figure 20.

5.3. Pulse Regulating Method

As for the conventional method to update the conductance of a memristor, according to the value of weight change that is calculated by the algorithm, the control circuit will generate corresponding signals to control the pulse generator and update time for producing positive/negative pulses and tuning the conductance of the memristor. Note that a memristor has the characteristics of finite conductance states, specific switching time, and fixed threshold voltage. Thus, some small delta-weights cannot be converted to pulses. Therefore, the precision of conductance during the update process is limited.

The traditional system originally has writing pulses whose widths are appropriate, on the one side, to regulate the conductance of a memristor, on the other side, not to damage the device. Simultaneously, each writing pulse is identical during different update processes. During the training process of ANNs, the weight change that is calculated by the algorithm is converted to positive/negative pulses that are n in equation (5) to update the conductance of the memristor. According to parameter, n, in equation (5), drastically change in conductance by positive or negative pulses for LTP or LTD process causes more cycle-to-cycle variations in corresponding memristors. To mitigate the effect of cycle-to-cycle variation, we propose a universal pulse regulating method in this work. The proposed pulse regulating method, instead of updating the weight by the number of the pulses that are directly converted from the value of weight change in each iteration, only applies one pulse and keeps the original width of the writing pulse, as shown in Figure 26. The decoder gets a signal from an arithmetic logic unit (ALU) for selecting one row to update. At the same time, the registers get the values of weight change that are calculated by an ALU. Then these values are transmitted to multiplexers as control signals. Multiplexers select one writing pulse that comes from a pulse generator as output when control signals are enabled. The enabled signal means the corresponding memristor needs to be updated. In the other words, the system will check if the current memristor needs to be written according to the result of the algorithm calculation. If the conductance of this memristor needs to be updated, then the system will generate one pulse that is an identical pulse with the traditional system to update the conductance (increasing or decreasing) no matter how big the delta weight is. For instance, in the evaluation platform (next section), one writing pulse with a certain width and amplitude is enough to change a conductance of a memristor. Thus, pulse regulating compresses all updating numbers of the pulses to the minimum, so that the update time in different iterations are the same.

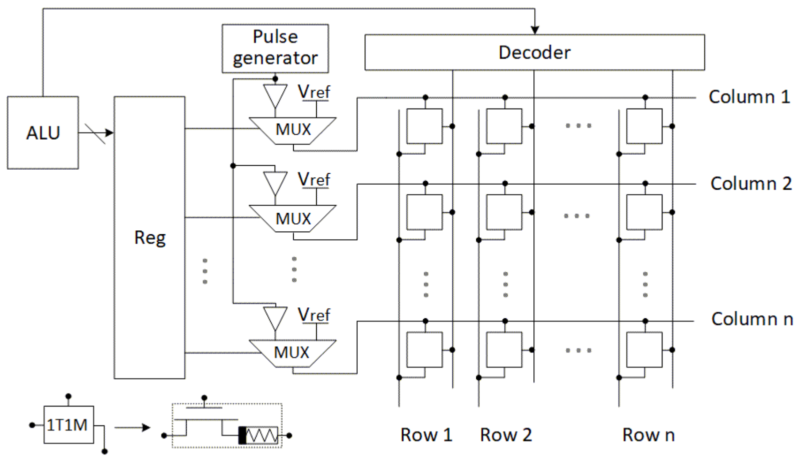


Figure 26. Circuit level design of the pulse regulating method. Each cell includes one selection transistor forming the one-transistor one-memristor (1T1M) array to avoid sneak path current problems.

5.4. Evaluation Methodology and Results

In order to verify the effectiveness of the proposed pulse regulating method in the edge AI, the MLP simulator is used to emulate the learning classification scenario with the MNIST handwritten dataset [59]. The memristive crossbar has energy efficiency and area superiority compared with CMOS synapses in the very large scale integrated (VLSI) circuit in online learning and can be necessary to overcome the effect of device variability and alternate current paths [38]. The crossbar array architecture with memristors had been proposed for on-chip implementation of weighted sum and weight update in the training process of learning algorithms [59]. We adopt the online learning hardware platform, NeuroSim+ [55], that needs to constantly writes the crossbar array to perform handwriting recognition. The networks in this simulator contain a three-layer with 400 neurons, 100 neurons, and 10 neurons, respectively, and base on memristors that can tune the conductance by voltage pulse stimulus. Since the edges of the images are not the most informative, one handwritten digit is cropped into 20 x 20. The recognitions of networks are ten digits. Thus, the input layer is 400 neurons and the output layer is 10 neurons. Note that, the availability of the pulse regulating method is not constrained by the number of hidden layers and the dimensions of each hidden layer and therefore can be adapt to any architecture and dimension in a given network and realize performance improvement. Parameters that are set in the simulator come from the results of the real memristor measurement [101].

In this platform, each simulation trains up to 100 epochs, and every epoch randomly selects 500 images from 60000 training images. The testing dataset has 10000 images and the system runs test after each training epoch. The networks will continually learn the feature of input data after 100 epochs since this simulator is an online learning network [59]. The metrics are evaluated with 100 epochs in this work. As for the flow of the training and testing, firstly, at the beginning of the training process, all weights are initialized to simulate the conductance of untrained memristors. Secondly, the system randomly selects one image from the dataset and follows the ANN algorithms to process forward propagation and backpropagation. Thirdly, the system gets delta-weight that will be converted to the number of pulses to be applied for weight updating. Fourthly, the pulse regulating method is applied to compress the number of pulses to one in this system. Using one pulse that is processed by the pulse regulating method to update the corresponding conductance of a memristor. Fifthly, the second to fourth steps are repeated until the system trains 500 images and the system runs the test process. Finally, the above procedures except for the first step will repeat 100 times.

Specifically, the hardware implementation block diagram of the pulse regulating method for one image training is shown in Figure 27. The system follows ANN algorithms by forwarding propagation to get recognition results, which includes vector-matrix multiplication operations. The "label" data of the training dataset is involved by backpropagation to get delta-weight for each memristor that is the value of weight needs to be updated, as shown in Figure 27. Then the pulse regulating method is implemented to compress the number of pulses. In our case, the system compresses the number of pulses to one. Therefore, when the delta-weight is larger than that corresponding to one pulse, the multiplexer will generate the signal that selects only one pulse to update the memristor, as shown in Figure 26. The basic peripheral and internal circuits that are included in this platform such as MUX, Adder, and MUL, and so on are explained in [55] and [59].

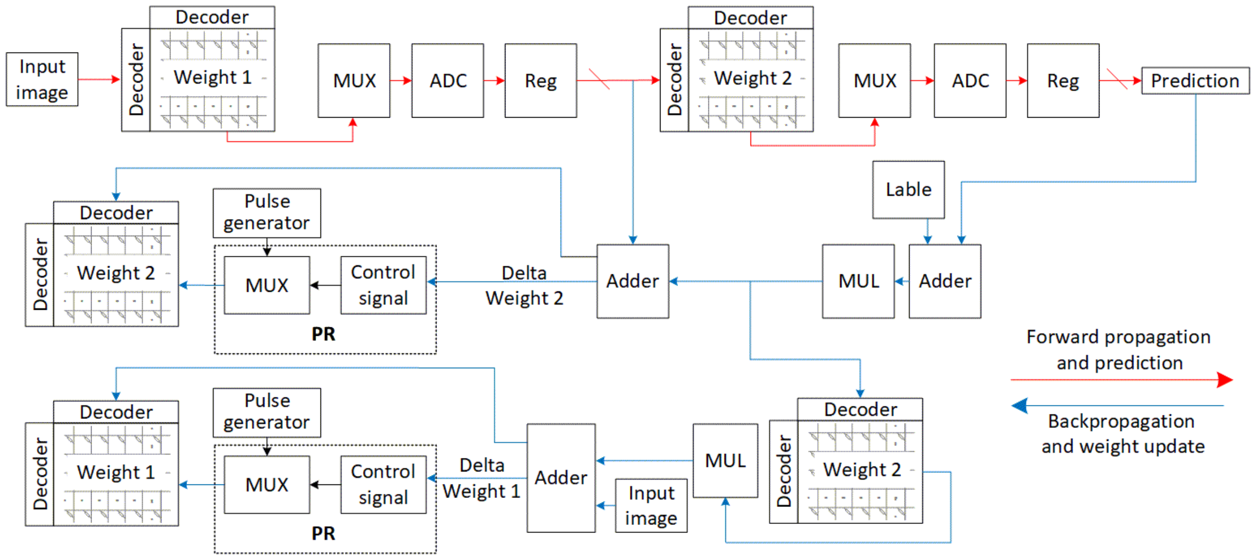
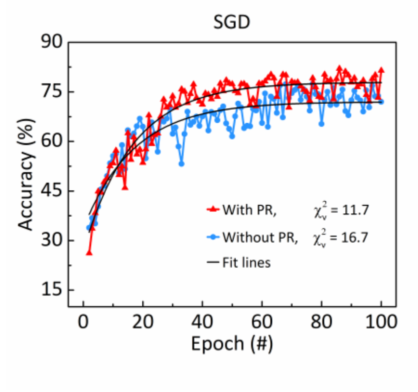
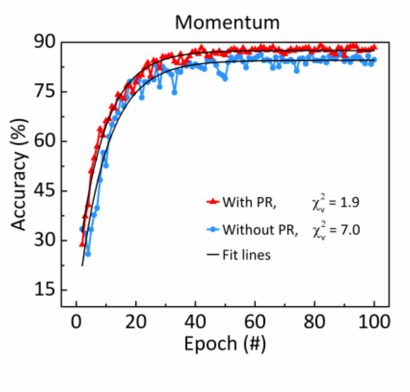
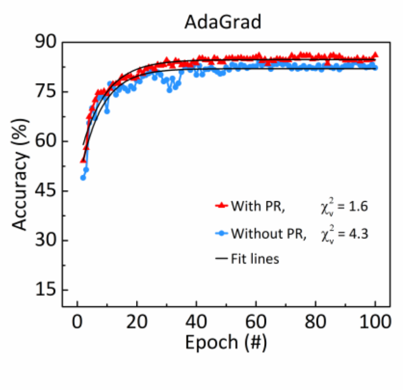


Figure 27. Hardware implementation block diagram of the pulse regulating method. The red path is forward propagation and prediction. The blue path is backpropagation and weight update. PR represents the pulse regulating method.

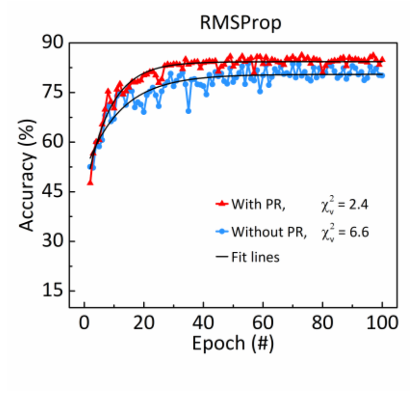
In this platform, SGD and its' variants are utilized, such as the Momentum, AdaGrad, RMSProp, and Adam. The pulse regulating is suitable for all five algorithms that the accuracies are higher than that without the pulse regulating method as shown in Figure 28. The negative impact of the pulse regulating method is reducing the learning speed, which only exits at the several beginning learning epochs and is reflected by the red curves below the blue curves in Figure 28. Although the learning speed is reduced by the pulse regulating method at the several beginning learning epochs, all recognition accuracies of five algorithms have significant improvement with the pulse regulating method after 100 epochs. In addition, the pulse regulating method effectively produces a smoother convergence of the training process, which reduces the excessive fluctuation of the recognition accuracy. The regressions are carried out by the exponential model to fit the experimental data without and with the pulse regulating method. The Reduced Chi-Sqr values that are represented as χ_{υ}^{2} with the pulse regulating method are smaller (closer to 1) than that without the pulse regulating method as shown in Figure 28, which demonstrates that the fluctuation of the recognition accuracy is reduced by the pulse regulating method [102].

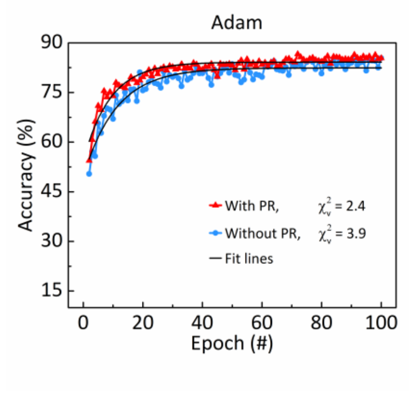






(a) (b) (c)





(d) (e)

Figure 28. Pulse regulating method for mitigating the effect of the cycle-to-cycle variation. Recognition accuracy with/without the pulse regulating method in 5 algorithms (from a to e). χ_v^2 is the Reduced Chi-Sqr values with analyzing data. PR represents the pulse regulating method. Each curve includes 100 epochs, and each epoch includes 500 images. The x-axis is number of training epoch and y-axis is the system prediction accuracy. From (a) to (e), the corresponding algorithm is SGD, Momentum, AdaGrad, RMSProp, and Adam, respectively.

Therefore, for every update, the cycle-to-cycle variation is limited with one pulse's impact, which minimizes the cycle-to-cycle variation for the system. Note that, the recognition accuracies have significant improvement with the pulse regulating method as shown in Figure 28. The reasons are two aspects that include the pulse regulating method minimizes the cycle-to-cycle variation and each update step uses at most one pulse to tune conductance. One pulse to tune conductance means smaller steps is achieved in the direction of convergence, while a big step will make the learning jump over the minimum point of weight [103].

5.5. Conclusion

We proposed and evaluated the level scaling method and the pulse regulating method, which are simple and feasible universal methods to effectively mitigate the impact of cycle-to-cycle variation. Under cycle-to-cycle variation, the recognition accuracy in the maximum number of the levels is not optimal for the real device. As for different materials-based multilevel memristors, using the same analysis method, the level scaling method can be used to optimize the neuromorphic computing system by selecting appropriately the number of the levels. Similarly, the pulse regulating method mitigates the impact of cycle-to-cycle variation by compressing the number of updating pulses to one. Furthermore, both methods can be implemented at edge computing, which paves the way for the adoption of memristors for more efficient applications for the era of the IoT.

6. ENERGY CONSUMPTION AND LATENCY IMPROVEMENT

In the training process of ANNs, the drastic update occurs by these positive or negative pulses at the beginning stage, which causes more energy consumption in corresponding memristors. Because specific features are various for different given training data, and only corresponding conductance of memristors will be updated to record features in one iteration, inevitably, this will lead to uneven pulse distribution in a crossbar array. Additionally, the maximum number of pulses determines the writing latency of the update stage in one iteration as a critical path. Note that, such maximum number for updating weight in different iterations are different due to the presence of the different training data and status of the current conductance. In this chapter, we utilize the pulse regulating method to optimize energy consumption and system latency by saving the number of updating pulses.

6.1. Pulse Regulating Method

One multiplexer is used to compress the number of updating pulses to one whenever a conductance of a memristor needs to tune as shown in Figure 26. The decoder gets a signal from an ALU for selecting one row to update. At the same time, the registers get the values of delta-weight that are calculated by an ALU. Then these values are transmitted to multiplexers as control signals. Multiplexers select one writing pulse that comes from a pulse generator as output when control signals are enabled. The enabled signal means that the corresponding memristor needs to be updated and that the corresponding delta-weight value is greater than or equals to the weight change of one pulse. In this way, the pulse regulating method directly affects every weight update and minimizes the number of pulses, and then optimizes the performance of the system.

6.2. Evaluation Methodology and Results

The same platform as mentioned in the previous chapter is utilized for evaluating the effectiveness with 125 epochs and 8000 images each epoch. In the simulation, the reading voltage is 0.5 V and the reading pulse's width is 5 ns. For the writing process, the voltage of the LTP and LTD is 3.2 V, -2.8 V, respectively. The pulse width of writing is 300 μs for both LTP and LTD [55, 101]. Reading and writing energy are determined by both the operations of the periphery circuit and the reading/writing within the crossbar array. In terms of the reading energy, it includes the operation energy of the periphery circuit - decoder, multiplexer, register, and analog-to-digital converter, etc., and the energy within the crossbar array - word lines, bit lines, and memristors. As for writing energy, it includes the operation energy of the periphery circuit – decoder, pre-charger, etc., and the energy within the crossbar array - word lines, bit lines, and memristors that are selected to update [55].

As for the training process, the latency of the memristor-based crossbar array includes reading and writing latency that is determined by both the operations of the periphery circuit and the reading/writing within the crossbar array. In terms of the reading latency, it includes the operation latency of the periphery circuit - decoder, multiplexer, register, and analog-to-digital converter, etc., and the latency within the crossbar array that is the width of the reading pulse. As for writing latency, it includes the operation latency of the periphery - decoder and pre-charger circuit, etc., and the latency within the crossbar array - both in the LTP and LTD that is calculated by multiplication of the number of update pulses and width of pulses. Note that, for each row, the latency of the writing is determined by both the latency of the maximum LTP process and the maximum LTD process, as shown in equation (7).

$$L = w_{pulse} * \sum_{1}^{n}(max\ (p_1, p_2, \dots p_m) + max\ (d_1, d_2, \dots d_m)) \tag{7}$$

where L is the latency of the writing within the whole crossbar array at a certain iteration, w_{pulse} is the width of writing pulse, n is the number of rows, p is a latency of the LTP process for one memristor, d is a latency of the LTD process in one memristor, and m is the number of the corresponding columns [55].

6.2.1. Energy Consumption

As for the memristive crossbar array, reading energy and writing energy constitute the total energy consumption, as listed in Table 3. The reading energy for different algorithms is determined by the size of the crossbar array and the number of total iterations. According to the process of a vector-matrix multiplication in a given memristive crossbar array including 41000 memristors in our experiments, the reading energy is always the same - 0.4 nJ for each iteration. However, the writing energy is much larger than reading energy, this is because 1) voltage of reading pulse is lower than the voltage of writing pulse [44], as mentioned in Section 6.2; and 2) the number of the pulses for the reading is usually much less than the writing. As shown in Figure 29, the writing energy for the crossbar array changes with the number of epochs without and with the pulse regulating method. It demonstrates the system consumes less writing energy with the pulse regulating method than that without the pulse regulating method, and the energy-saving is increased with the increased number of epochs. Furthermore, the details of numerical writing energy are shown in Figure 30. The red and blue bars represent the writing energy after 125 epochs without and with the pulse regulating method with five algorithms, respectively. Because less pulse is used in weight updating following the pulse regulating method, the writing energy saving is from 7.7% to 26.9% with five algorithms. Furthermore, the AdaGrad consumes the least writing energy in five algorithms that is respectively 3.9 mJ and 3.6 mJ without and with the pulse regulating method, which realizes 7.7% writing energy saving. Meanwhile, as listed in Table 3, the total energy of the AdaGrad is respectively 4.3 mJ and 4.0 mJ without and with the pulse regulating method, which realizes 7.0% total energy saving. Additionally, the RMSProp is the most energy-saving among the five algorithms. It consumes respectively 17.1 mJ and 12.5 mJ writing energy without and with the pulse regulating method, which realizes 26.9% writing energy saving. The total energy is respectively 17.5 mJ and 12.9 mJ without and with the pulse regulating method, which realizes 26.3% total energy saving. Such energy-saving makes the proposed pulse regulating method especially suitable for the edge AI in IoT systems with serious energy constrain.

Table 3. Total Energy and Saving Percentage of Neural Network with/without Pulse Regulating Method with Five Algorithms

	Algorithm	Without PR (mJ)	With PR (mJ)	Energy Saved (%)
Total Energy	SGD	6.6	5.8	12.1
	Momentum	6.7	5.7	14.9
	AdaGrad	4.3	4.0	7.0
	RMSProp	17.5	12.9	26.3
	Adam	12.2	9.8	19.7

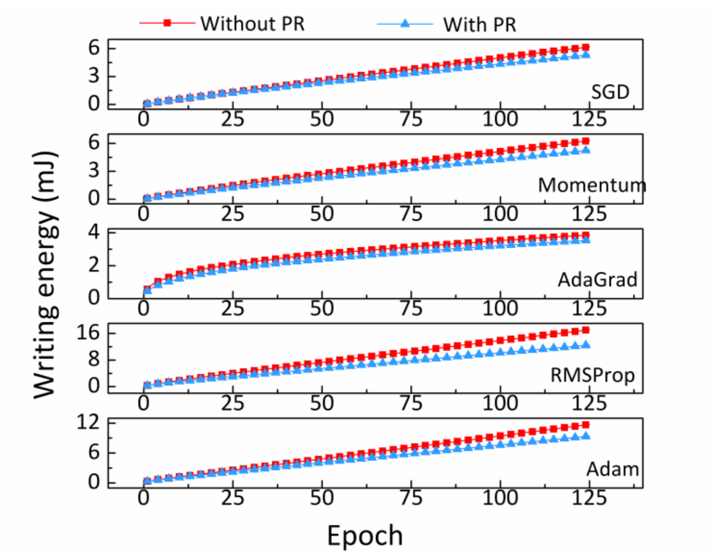


Figure 29. Writing energy as a function of epoch with five algorithms. Red and blue lines represent without and with the pulse regulating method.

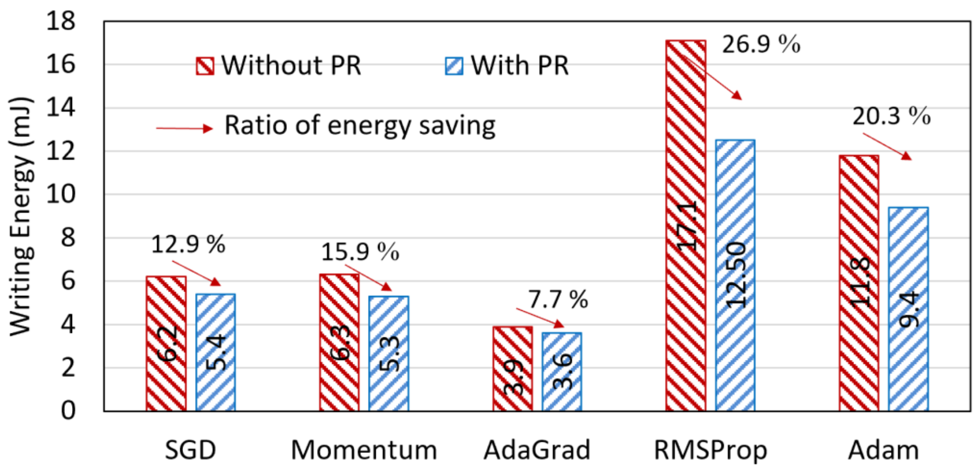


Figure 30. Writing energy with five algorithms. The red and blue bars represent the writing energy after 125 epochs without and with the pulse regulating method.

Table 4 shows the recognition accuracy of the five algorithms after the first image, first epoch, and 125th epoch training, respectively. All accuracy is higher than 92.3% after 125 epochs training. The difference without and with the pulse regulating method is smaller than 1.0%. Additionally, the accuracy is limited by the number of bits of input data and hardware-based constraint that includes ADC precision and circuit noise in this platform [55]. Therefore, the pulse regulating method does not hurt recognition accuracy much.

Table 4. Recognition Accuracy of Neural Network with Different Training Stage (%)

Algorithm	1st image without/with PR	1st epoch without/with PR	125th epoch without/with PR	Fluctuation (after 125 epoch)
SGD	14.8 / 14.8	70.4 / 71.8	92.7 / 92.8	+0.1
Momentum	14.8 / 14.8	76.9 / 72.4	93.1 / 93.7	+0.6
AdaGrad	12.9 / 14.7	70.1 / 84.0	93.3 / 92.3	-1.0
RMSProp	11.3 / 14.7	79.8 / 83.0	93.6 / 94.5	+0.9
Adam	12.5 / 14.8	83.4 / 83.4	94.2 / 94.7	+0.5

In addition, the pulse regulating method effectively produces a smoother convergence of the training process, which reduces the excessive fluctuation of the recognition accuracy. Taking SGD as an example, Figure 31 shows the recognition accuracy with increasing epochs. The regressions are carried out by the Nelder model to fit the experimental data of the simulation without and with the pulse regulating method. The Reduced Chi-Sqr with the pulse regulating method, 0.4, is higher than that without one, 0.2, which demonstrates that the fluctuation of the recognition accuracy is reduced by the pulse regulating method [102]. Therefore, a smooth convergence of the training process is another reason that the total writing energy is lower than that without the pulse regulating method. Indeed, the pulse regulating method theoretically impacts the learning speed, but this impact only occurs at the beginning of the first epoch, which can be neglected comparing 125 epochs, as shown in the inserted figure in Figure 31. Only before the cross point - 1250, the accuracy without the pulse regulating method is higher than that with the pulse regulating method. Therefore, the drawback of the pulse regulating method can be neglected.

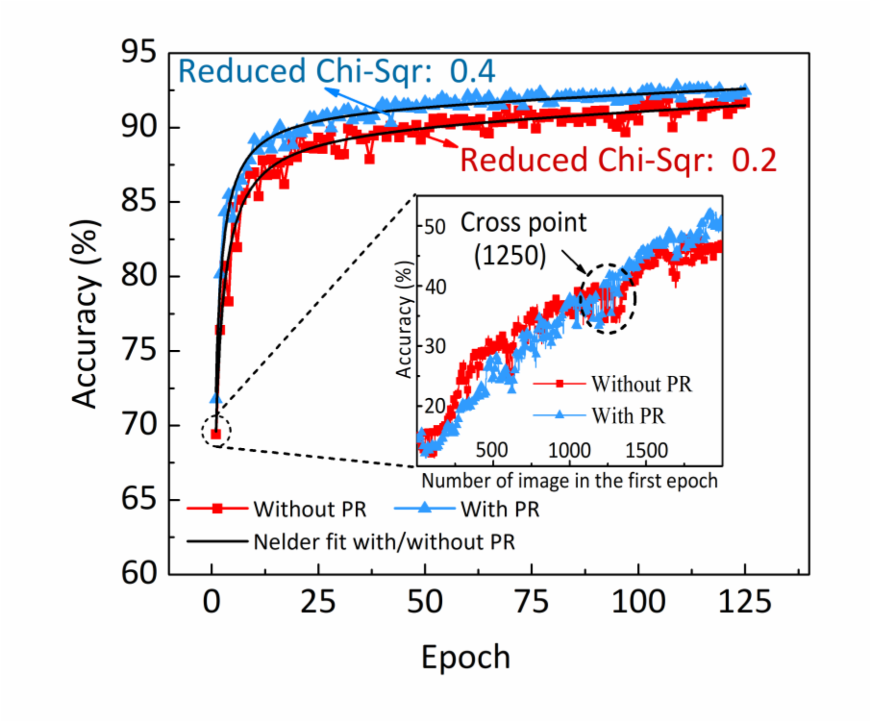


Figure 31. Nelder model fit without and with the pulse regulating method.

Finally, we calculate the sum of the number of the updating pulses required for 125 epochs, taking SGD as an example, without and with the pulse regulating method as shown in Table 5.

Note that the power (energy/latency) with the pulse regulating method is higher than that without the pulse regulating method. The reason is that the saved latency with the pulse regulating method is more significant than the saved energy with the pulse regulating method. However, the total number of the pulses is saved 46.6% and 37.8% for weight 1 and weight 2 layer, respectively. The fewer pulses are utilized, the less energy is consumed. Those results further prove that the proposed method can effectively reduce energy consumption during the training process in ANN.

Table 5. Number of the Pulses and Power Consumption for Weight Update

Layer	Pulse number without PR	Pulse number with PR	Pulse saved (%)
Weight 1	70250859	37498805	46.6%
Weight 2	16541627	10283640	37.8%
Power (nW)	Without PR 354	With PR 586	

Following discussion is about the energy consumption of the peripheral circuit. As the weight increase and decrease need different programming voltage polarities (LTP and LTD), the weight update process requires 2 steps with different voltage bias schemes. In weight update, the selected memristors will be on the same row, and programming pulses or biases (if no update) are provided, allowing the selected memristors to be tuned differently in parallel. To perform weight updates for the entire array, a row-by-row operation is necessary. The pulse generator in the system with the pulse regulating method is the same as the system with the normal method. In the system with the normal method, each row needs registers and counter to record and control the updating process since the time of updating process in the different training iterations is probably different [7, 55, 104]. In the system with the pulse regulating method, those two components are not needed because the selected row only uses one pulse to update the conductance of the memristor. Instead, one multiplexer is added to the system. Therefore, the energy of the peripheral circuit with the pulse regulating method is not increased compared with the system that uses the normal method.

6.2.2. Pulse Distribution

The energy of the pulses in the reading and writing process will generate thermal power. The more pulses are generated for updating conductance, the more heat the crossbar array generates. Since the reading pulse is evenly distributed, now we only analyze the writing pulse distribution. Additionally, the accuracy of the recognition at the beginning stage is very low since the weight is randomly initialized before training. Therefore, the weight change is larger at the beginning stage than later, which can be reflected by the difference of accuracies, as shown in Table 4 after training of the first image and first epoch without and with the pulse regulating method in five algorithms. All the accuracies are lower than 15.0% after training of the first image and higher than 70.0% after training of the first epoch. The increment of the accuracy is more than 55.0%. Thereby, it needs more pulses at the beginning stage of the training for large weight updating.

Because of more writing pulses at the beginning stage of the training, we extracted the weight update's pulse distribution at the 1st and 1000th iteration at the first epoch with the AdaGrad algorithm as an example. Each iteration will update weight 1 and 2 layers. Figures 32 (a) and (c) represent the weight 1 layer with 400 input and 100 output for the 1st and 1000th

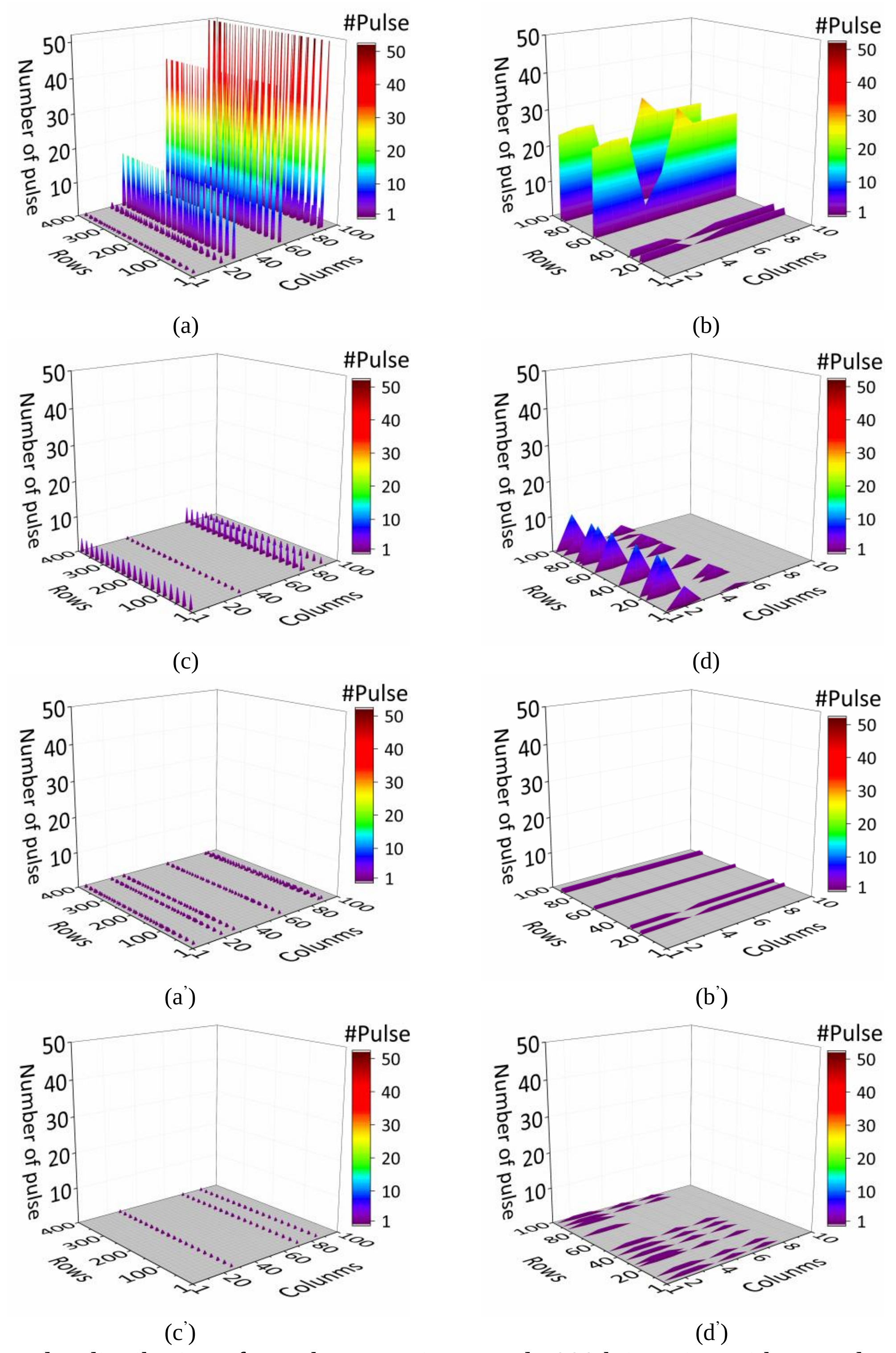


Figure 32. Pulse distribution of crossbar array in 1st and 1000th iteration without and with the pulse regulating method. (a) and (c) represent the weight 1 layer with 400 input and 100 output for the 1st and 1000th iteration, and Figures 32 (b) and (d) represent the weight 2 layer with 100 input and 10 output for the 1st and 1000th iteration. The Z-axis is the number of pulses for weight update that includes LTP and LTD.

iteration, and Figures 32 (b) and (d) represent the weight 2 layer with 100 input and 10 output for the 1st and 1000th iteration. The Z-axis is the number of pulses for weight update that includes LTP and LTD. As shown in Figures 32 (a) and (b), for the 1st iteration without the pulse regulating method, in the weight 1 layer, the maximum number of the pulses is 52 that is in the 91st-column covering the related 125 rows; in the weight 2 layer, the maximum number of the pulses is 30 that is in the 60th-row 6th-column and the 91st-row 6th-column. Similarly, as shown in Figures 32 (c) and (d), for the 1000th iteration without the pulse regulating method, in the weight 1 layer, the maximum number of the pulses is 5 that is in the 75th-column covering the related 90 rows; in the weight 2 layer, the maximum number of the pulses is 10 that is in the 2nd-column covering 7 rows. With increasing the iterations, the weight is closer to the global minimum. Therefore, the number of pulses decreases with an increasing number of iterations. It is concluded that without the pulse regulating method, extremely uneven heat distribution is caused by pulses' uneven distribution. Figures 32 (a'), (b'), (c'), and (d') show the distribution of the pulses with the pulse regulating method at the same update stage. All of the numbers of the pulses are compressed to one. Note that, the position of the pulses that are used to update at 1,000th iteration in Figures 32 (c) and (c') are different, and pulse distribution in Figure 32 (c') cannot be obtained directly by compressing all pulses in Figure 32 (c) to one. The reason is that after weight updating based on the first image without and with the pulse regulating method, the following weight updating between both of that is totally different since the current pulse distribution in Figure 32 (c) and Figure 32 (c') only bases on the present image and current weight. It is the same reason for different pulses distribution in Figures 32 (d) and (d'). For further analysis, Table 6 shows the mean and standard deviation of those number of update pulses without and with the pulse regulating method. With the pulse regulating method, the mean of the number of pulses decreases by 18.8, 9.5, 1.6, and 4.5 for weight layers at 1st and 1,000th iteration, respectively. All the standard deviation of the number of update pulses with the pulse regulating meth65od is 0, but that is 20.0, 11.2, 1.6, and 3.6 without the pulse regulating method, respectively. Thus, it is verified that even pulse distribution is achieved using the pulse regulating method in the ANN system.

Table 6. Statistics of Number of the Pulses with Different Iteration[a]

	1st iteration without/with PR	1000th iteration without/with PR
M of weight 1	19.8 / 1.0	2.6 / 1.0
M of weight 2	10.5 / 1.0	5.5 / 1.0
SD of weight 1	20.0 / 0	1.6 / 0
SD of weight 2	11.2 / 0	3.6 / 0

[a] *M represents mean. SD represents standard deviation.*

6.2.3. System Latency

For a given ANN structure in edges, every iteration has stable reading latency since the process of a vector-matrix multiplication is executed using a parallel reading strategy. However, the system updates its weight row by row, which indicates a parallel writing strategy cannot be implemented for all rows at the same time. In the learning process, the system will randomly take one image and feed-forward according to machine learning algorithms [7]. The machine learning algorithm will calculate the weight change according to backpropagation [9, 55, 104]. In this way, the weight change can be either a positive value or negative value, which corresponds to LTP and LTD process. Each row's writing latency is determined by the maximum number of writing pulses as a critical path. Thereby, the main latency for the crossbar array is writing latency that strongly depends on the maximum update pulses of each row. For example, as shown in Figure 33, suppose the writing latency is four pulses without the pulse regulating method for the selected row, but it is only one pulse with the pulse regulating method, reducing the latency of the pulses by 75.0%. In some extreme cases, suppose the one change is from the minimum conductance to the maximum conductance, which has 100 levels (default in simulator [55]), theoretically, the maximum number of the needed writing pulses without the pulse regulating method is 100. However, with the pulse regulating method, the maximum number of the writing pulses is still one, since the number of the writing pulses is compressed to one, reducing the latency of the pulses by up to 99.0%. Therefore, with the pulse regulating method, equation (7) can be improved to:

$$L = w_{pulse} * ((\,NO.of\ p\,) + (\,NO.of\ d\,)) \tag{8}$$

where L is the latency of the writing within the whole crossbar array at a certain iteration, w_{pulse} is the width of writing pulse, $NO.of\ p$ and $NO.of\ d$ are the total number of rows that are selected for the LTP and LTD process. Latency schematic diagram of the writing process is reduced by the pulse regulating method as shown in Figure 34. In each iteration, the system will read memristor that is forward propagation, then calculate delta-weights that include backpropagation, and finally write memristor. When applying the pulse regulating method, the time of writing will be decreased to a minimum by compressing to one pulse.

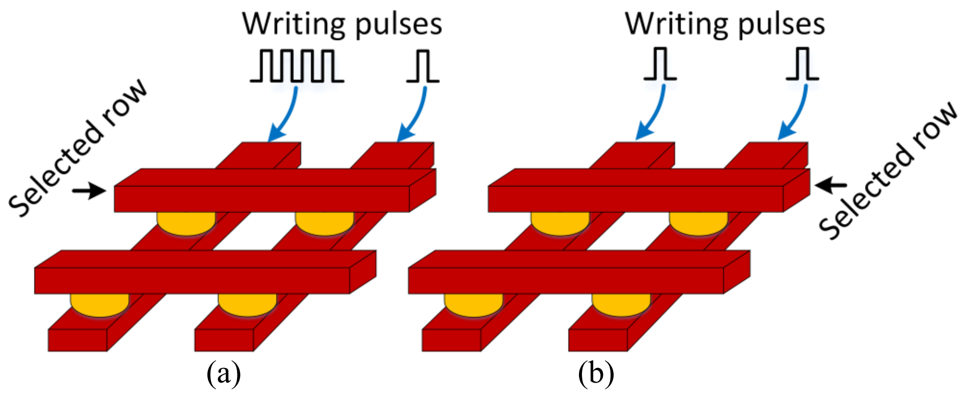


Figure 33. Weight update by pulse signal in a selected row. (a) without the pulse regulating method, (b) with the pulse regulating method.

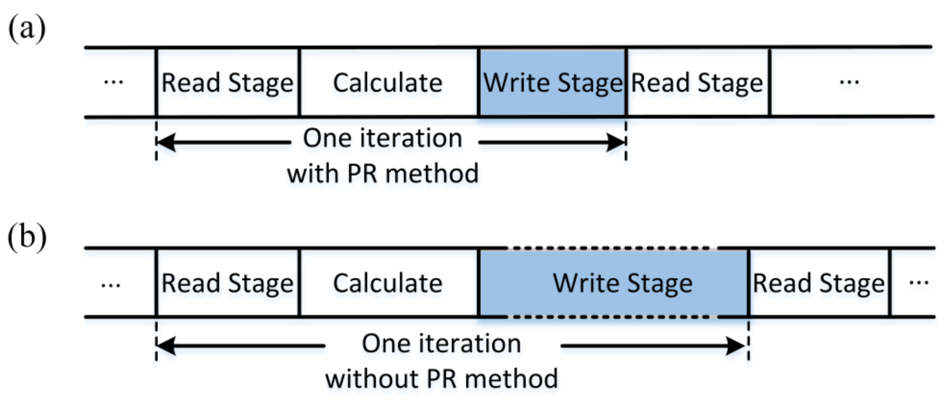


Figure 34. Latency schematic diagram. (a) The latency of the write stage is reduced by the pulse regulating method compared to the original system. (b) Latency diagram of the original system. The latency of write stage may different, which depends on the max number of pulses for LTP and LTD processes.

Figure 35 shows the total writing latency that is normalized after 125 epochs without and with the pulse regulating method. The total writing latency is decreased by 30.0%-50.0% for five algorithms, respectively. Therefore, the pulse regulating method extremely effective in reducing writing latency, which is preferred by the edge AI with real-time requirements. Additionally, because of the pulse regulating method, every iteration has the same number of the writing pules, and then the timing regularity of the system and the reliability of the system is greatly improved.

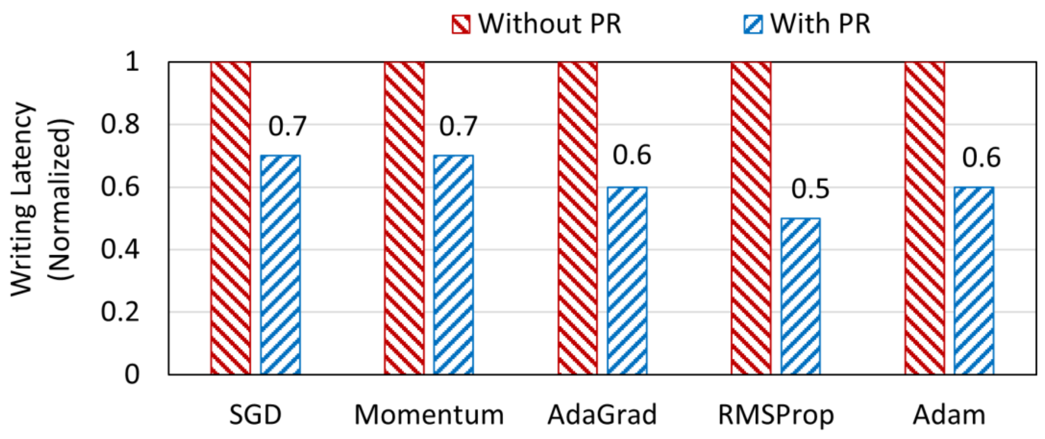


Figure 35. Writing latency of crossbar array without and with the pulse regulating method.

For the system with the pulse regulating method, the system reduces the number of pulses to one at every single update. Some weights need to be updated several times to reach a certain value during the entire training process according to a global minimum of the loss function that is calculated by the algorithm. Thus, multiple pulses and updates are implemented at the entire train process rather than within one update. Such necessary multiple updates consume indispensable energy. Thus, energy saving is limited. But as for latency, because the pulse regulating method reduces the number of pulses to one at every single update, although sometimes several updates are needed for a memristor, each update can be parallel with the update for the other memristor in the same row, which is actually all memristors in a row shared the update time without additional latency. Therefore, the latency reduction is more notable.

6.2.4. Nonlinear Property of Memristors

Ideally, when LTP or LTD occurs, the change in the conductance of an ideal synapse device is proportional to the number of writing pulses. However, in reality, such change mismatches the writing pulses due to the nonlinearity of memristors [20, 105, 106]. In our simulation, the actual conductance curve is labeled with a nonlinearity value from +3 to -3 [20, 105], which represents the extent to the curve deviates from the ideal linear device.

Taking the AdaGrad algorithm as an example, Table 7 shows the total writing energy without and with the pulse regulating method, under the significant nonlinear property. The recognition accuracy does not have significant fluctuation. The accuracy recovery is done by the piecewise linear method [20] that regains accuracy over 90% under 3/-3 circumstances. All of the total writing energy with the pulse regulating method is lower than without the pulse regulating method. Energy saved is from 7.7% to 13.0%, respectively. Thus, the pulse regulating method is proved to effectively reduce writing energy even with the nonlinear property of a memristor.

Table 7. Writing Energy and Recognition Accuracy of Neural Network with Nonlinearity

NL[a] (LTP/LTD)	Writing energy without PR (mJ)	Writing energy with PR (mJ)	Energy saved (%)
0 / 0	3.9	3.6	7.7
1 / -1	4.6	4.0	13.0
2 / -2	4.6	4.2	8.7
3 / -3	4.7	4.1	12.8
NL (LTP/LTD)	Recognition accuracy without PR (%)	Recognition accuracy with PR (%)	
0 / 0	93.3	92.3	
1 / -1	92.2	92.0	
2 / -2	89.1	88.3	
3 / -3	84.6	86.7	

[a] *NL represents the value of nonlinearity.*

6.2.5. Variations of Memristors

Because of the physical limitations of a memristor, minimum conductance variation (Gmin), maximum conductance variation (Gmax), ON/OFF ratio variation (Gmax/Gmin), cycle-to-cycle variation (CtoC), and device-to-device variation (DtoD) [18] exist in the application of memristor-based hardware implementation, as shown in Figure 36. To explore the effectiveness of the pulse regulating method, we take the AdaGrad algorithm as an example and investigate these variations following standard/Gaussian distribution N (μ, σ) into consideration. In our experiments, minimum conductance subjects to N (Gmin, σ×Gmin), and maximum conductance subjects to N (Gmax, σ×Gmax). Device-to-device variation subjects to N (NL, σ) distribution. A cycle-to-cycle variation that subjects to N (0, σ×(Gmax-Gmin)2) represents conductance deviations in each weight update [105].

Above, NL, Gmax, and Gmin are fixed parameters for each simulation. ON/OFF ratios are configured as 17 in variation 1 and 15 in variation 2. For Variations 1 and 2 in Table 8, we set σ of the minimum conductance, maximum conductance, device-to-device, and cycle-to-cycle variation as 5.0%, 5.0%, 0.5, 1.0%, and 15.0%, 15.0%, 1.4, 2.5%, respectively [20, 105]. Table 8 shows the result of simulations under different circumstances.

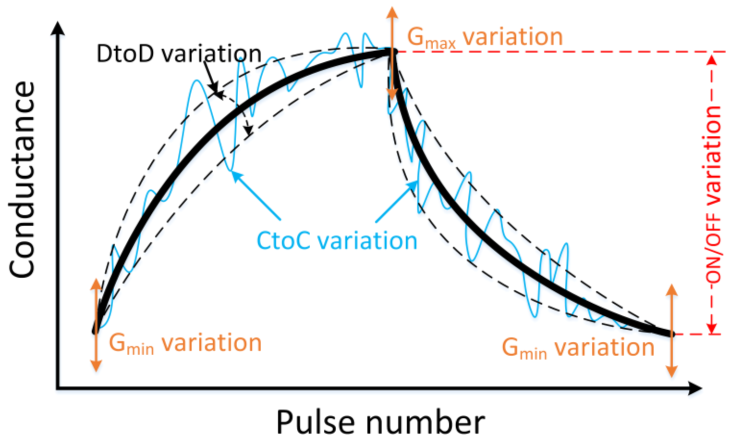


Figure 36. Variations of a memristor. DtoD, CtoC, Gmax, and Gmin represent device to deice, cycle to cycle, maximum conductance, and minimum conductance, respectively [55].

Table 8. Experimental Results of Neural Network with Variations

	Variation 1 without/with PR	Variation 2 without/with PR
Writing Energy (mJ)	3.9 / 3.5	3.8 / 3.2
Writing Latency (normalized)	1 / 0.7	1 / 0.6

In Table 8, for different circumstances without the pulse regulating method, the total writing energy is 3.9 mJ and 3.8 mJ. After utilizing the pulse regulating method, the total writing energy is saved by 10.4% and 15.3%, respectively. Additionally, writing latency is reduced by 30.0% and 40.0%. Thus, even with many variations, the pulse regulating method is still efficient to reduce energy consumption and writing latency of the crossbar array.

6.2.6. Failure Rate, Endurance, and Aging

A typical manufacturing process typically seeks a failure rate of <10% [50]. To evaluate the influence of failure rate in a crossbar array for the edge AI, 5%, 10%, and 15% of the fault in the crossbar array are simulated, as shown in Figure 37 [107]. The random positions of the fault memristors are chosen in the crossbar array. The ratio of the stuck at 0 and 1 is 1 : 5.2 [50]. Taking the SGD algorithm as an example, Table 9 shows the accuracy under the influence of the failure rate in this network. The neural network with the pulse regulating method still has accuracy improvement as compared to that without the pulse regulating method according to results of the mean and standard deviation that are obtained from 500 random cases of each failure rate. When the failure rate is 15%, the accuracies of the neural network without and with the pulse regulating method both reduce by about 3%.

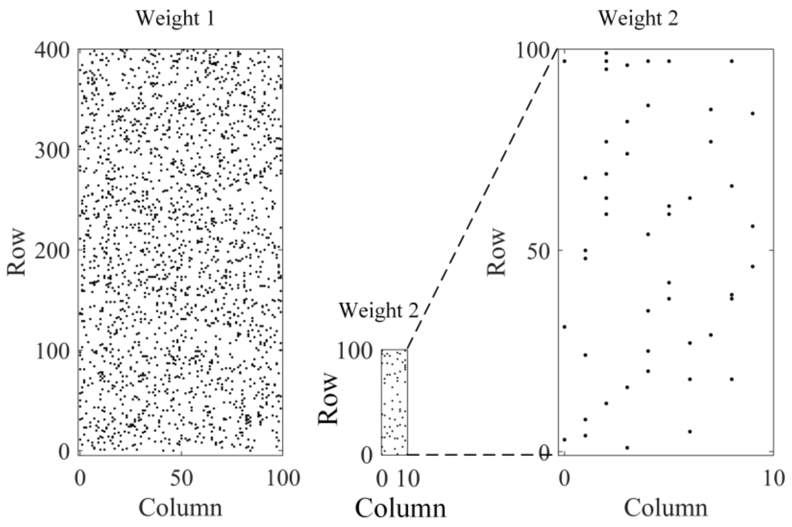


Figure 37. Random positions of the failure memristors in crossbar array with 5% failure rate.

Table 9. Recognition Accuracy and Standard Deviation with Different Failure Rates

Failure rate	Mean[a]		Standard deviation[a]	
	Without PR	With PR	Without PR	With PR
5%	91.7%	92.0%	0.0050	0.0039
10%	91.0%	91.4%	0.0058	0.0044
15%	88.5%	89.6%	0.0071	0.0047

[a] *Mean and Standard deviation are obtained including 500 random cases of each failure rate.*

Memristors can only be programmed reliably for a given number of times. Afterward, the conductance tunability of the memristor deviates from the initial state, which is called aging, and it limits the lifetime of memristor-based crossbars in the edge computing system [108]. The conductance is assumed to drift towards different final states, or randomly drift, based on different various drift rates, which are equivalent to conductance drift different amounts over 10 years, respectively [109]. Taking the SGD algorithm as an example, Figure 38 shows the accuracies under the influence of aging in this network. The precision, recall, and F1 score are shown in Table 10. The restoration of accuracy can be completed by retaining and remapping method [49]. The pulse regulating method is still effective by comparing to the accuracy that is without the pulse regulating method.

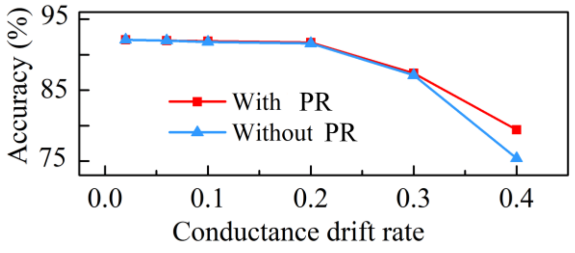


Figure 38. Recognition accuracy with different conductance drift ratios that are 0.02, 0.06, 0.10, 0.20, 0.30, and 0.40.

Table 10. Classification Report[a]

Class	Without PR method			With PR method		
	P	R	F1	P	R	F1
0	0.742	0.956	0.836	0.812	0.945	0.874
1	0.628	0.990	0.768	0.753	0.990	0.856
2	0.727	0.806	0.765	0.740	0.850	0.791
3	0.805	0.784	0.794	0.781	0.665	0.719
4	0.630	0.764	0.690	0.811	0.797	0.804
5	0.840	0.577	0.684	0.789	0.609	0.687
6	0.770	0.863	0.814	0.923	0.864	0.893
7	0.890	0.714	0.792	0.832	0.791	0.811
8	0.887	0.507	0.645	0.843	0.633	0.723
9	0.810	0.449	0.578	0.757	0.798	0.777
Micro _avg	0.773	0.741	0.737	0.804	0.794	0.793

[a] *The result with 0.4 conductance drift ratios. P, R, and F1 represent precision, recall, and F1-score, respectively.*

In addition, the endurance of a memristor is one limitation for high-frequency writing in an ANN system [101, 104]. The pulse regulating method extremely saves the number of writing pulses, as shown in Table 5. Therefore, this method is still effective with failure and aging circumstances and benefits the cycling endurance performance of a memristor.

6.2.7. Different Architecture and Database

To verify the pulse regulating method in different architecture, different hidden layers are simulated as shown in Figure 39. As expected, the recognition accuracy for using the pulse regulating method is higher than that without the pulse regulating method. However, the leakage power is increasing when increasing the neurons of the hidden layer as shown in Figure 39. The leakage power with the pulse regulating method is a little higher (<10%) than that without the pulse regulating method because multiplexors are added. Thus, the pulse regulating method is effective with different architectures.

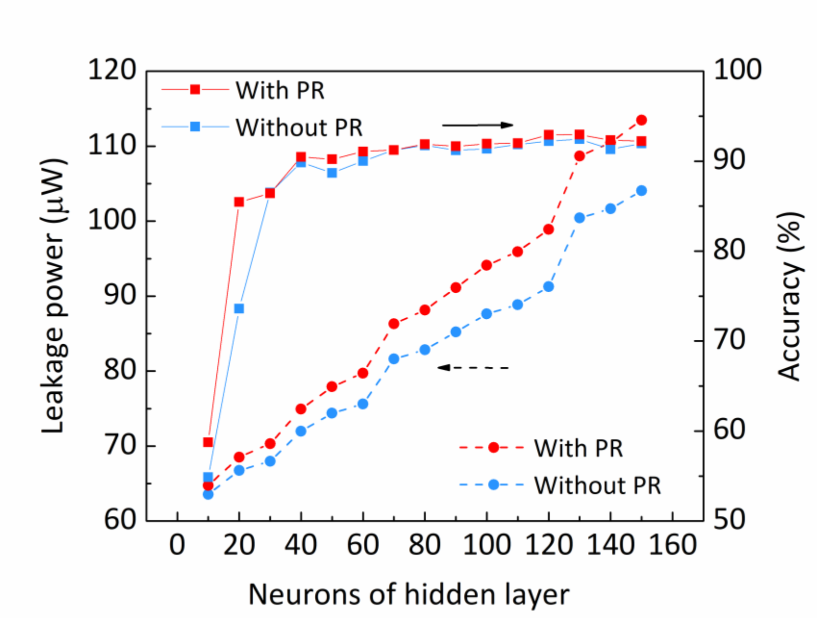


Figure 39. Leakage power with the different number of neurons of the hidden layer. Recognition accuracy with the different number of neurons of the hidden layer.

Furthermore, the pulse regulating method in VGG-8 with memristive crossbar array architecture and CIFAR-10 database are evaluated as shown in Table 11. The accuracy difference without and with the pulse regulating method is smaller than 1.0%. Therefore, the pulse regulating method does not hurt recognition accuracy much. Additionally, the accuracy is limited by the hardware-based setting that includes variations and nonlinearity in this platform [104]. As expected, the pulse regulating method respectively reduces latency by 46.00% and energy consumption by 16.67% in the system.

Table 11. Experimental Results of Neural Network with VGG-8 and CIFAR-10

	Accuracy	Latency[a]	Energy
With PR method	90.3%	0.54	0.25 J
Without PR method	91.1%	1.00	0.30 J
Difference	0.88%	46.00%	16.67%

[a] *Latency values are normalized.*

6.2.8. Pulse Regulating Method and Other Works

PRCoder is an algorithm proposed for different RRAM applications [42]. The cycle-rehabilitate technique was used to alleviate thermal crosstalk [43]. At the same time, increasing the size of the insulator or utilizing new materials with higher thermal conductivity for improving performance were proposed in [74, 75]. A new structure, thermal-house, was presented to optimize thermal management [76]. However, those new algorithms or new material/structure of device-based solutions inevitably increase the complexity of peripheral circuits or the difficulty of the manufacturing process, even increasing the latency.

The pulse regulating method is a simple and feasible method for edge computing in IoT systems. As shown in Table 12, as compared with the state-of-art, the pulse regulating method does not need to use special structure and material. Simultaneously, it does not need to add an extra algorithm to alleviate uneven pulse distribution. What’s more, with the pulse regulating method, the energy consumption is effectively reduced by 7.5%, and the writing latency is averagely reduced 38.0%.

Table 12. Comparison of the State-of-art

	[105]	This work
Energy consumption	6.7 mJ	6.2 mJ
Write latency (Normalized)	1	0.62

	[42]	[43]	[74]	[75]	[76]	This work
Without new material or structure	√	√	×	×	×	√
Without adding the extra algorithm	×	×	√	√	√	√

6.3. Conclusion

In this chapter, we utilize the pulse regulating method to reduce energy consumption, decrease writing latency, and improve the timing regularity of memristor-based smart edge computing in IoT systems. The pulse regulating method is verified to be effective based on devices to algorithms architectures. Instead of modifying the traditional algorithm-based technology, the pulse regulating method that only needs to add a multiplexer circuit before every writing operation in the weight-updating process, to optimize the performance of the system, especially at several beginning iterations that have a dramatic change of weight, and effectively reduces the writing latency by reorganizing the update timing. ANNs in five algorithms with nonlinearity from (0/0) to (3/-3), different failure rates (5%, 10%, and 15%), two variations conditions, different architectures and datasets, and aging effect have been evaluated to investigate the effectiveness of the pulse regulating method in the edge computing. Note that, some accuracies with pulse regulating method are not higher than that without pulse regulating method because the cycle-to-cycle variation is not involved. The results indicate that it saves the writing energy of the crossbar array by 7.7%-26.9% and reduces the writing latency by 30.0%-50.0%. It concludes: 1) The proposed pulse regulating method enables low energy consumption and even pulse distribution to reduce the heat resulted from intensive pulses. 2) Because the number of pulses for weight update is compressed to one, the pulse regulating method effectively reduces the writing latency and improves timing regularity. 3) The pulse regulating method is still effective under different nonlinearity, failure rates, aged devices, architectures, datasets, and variation circumstances in memristor-based ANN for paving the way for the further development of edge computing in IoT systems.

7. FAULT TOLERANCE IMPROVEMENT

Popular ANN weight initialization techniques [110, 111] consist of an effective layer-wise scaling of random weight values sampled from a Gaussian distribution. Assuming that weights follow a Gaussian distribution at time t = 0, owing to the central limit theorem weights will also converge towards a Gaussian distribution [112], which illustrates the weight distribution of a trained model in a VGG 8 [104, 113] is a Gaussian distribution as shown in Figure 40 and Figure 41. It concludes that the most of weight values equal to or are close to 0.

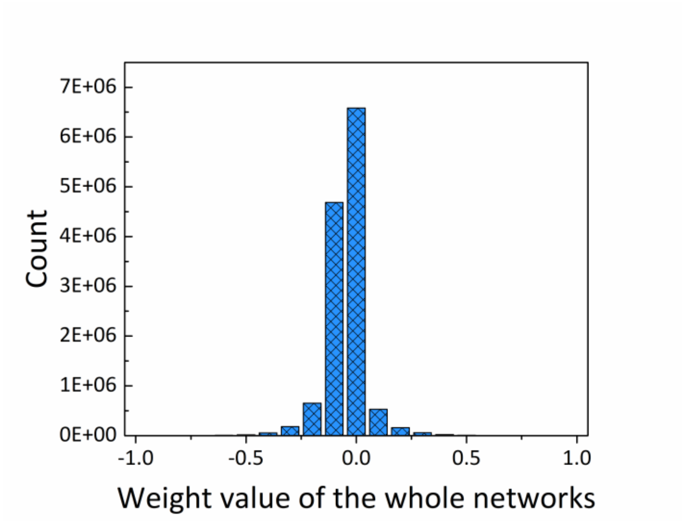
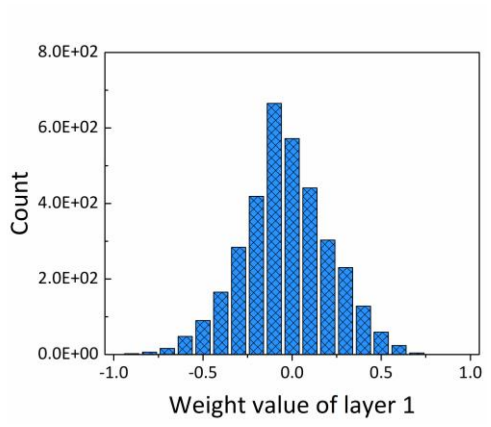
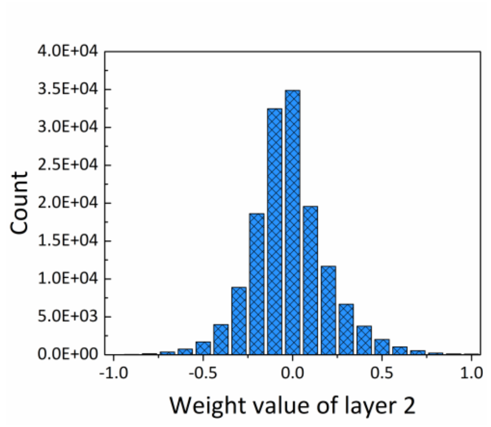


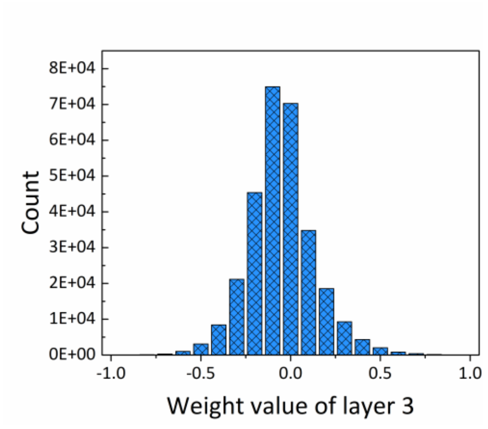
Figure 40. Weight distribution for the whole networks.



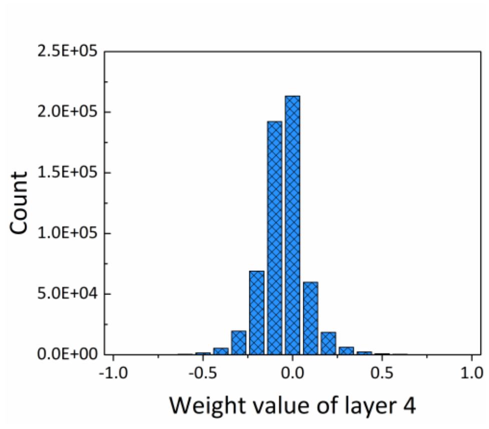
(a)



(b)

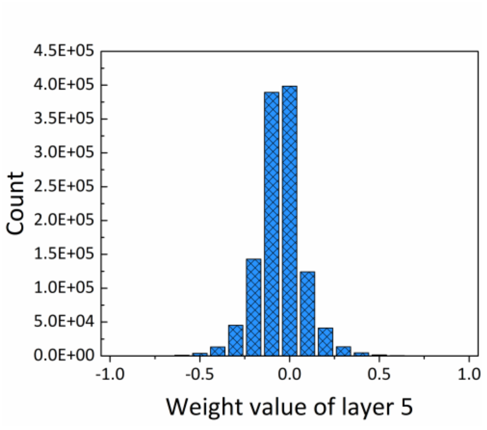


(c)

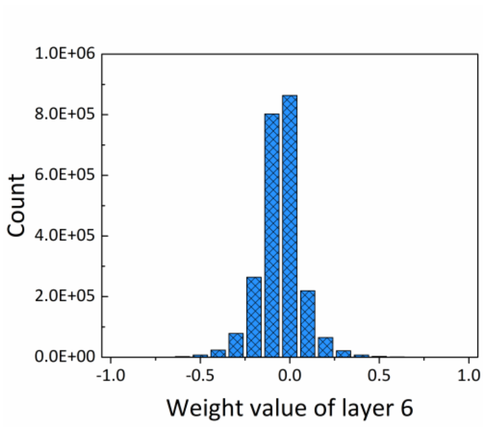


(d)

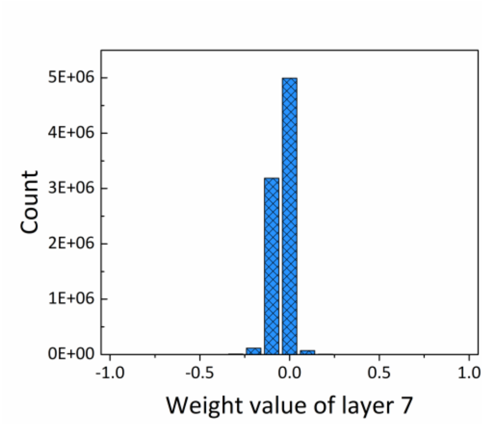
Figure 41. Weight distribution for each layer of the networks. (a) to (h) represent 8 layers' weight distributions.



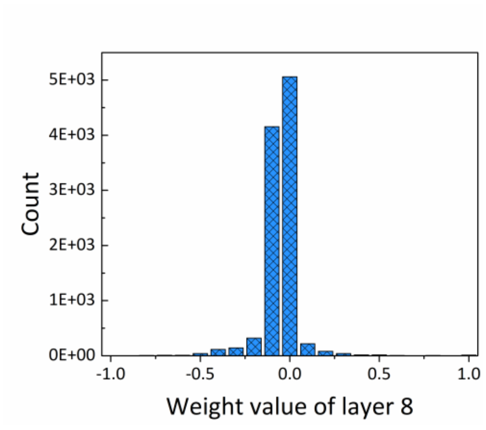
(e)



(f)



(g)



(h)

Figure 41. Weight distribution for each layer of the networks (continued). (a) to (h) represent 8 layers' weight distributions.

At the same time, the performance of hardware implementation for memristive ANNs needs to consider both SA0 and SA1 fault. Moreover, according to [50], the failure rate of the SA1 fault can be 5.2x higher than the failure rate of the SA0 faults. In this case, the SA1 fault becomes the major issue leading to the accuracy drop rather than the SA0 fault. To solve this problem, we propose a differential mapping technique to improve the network fault tolerance for the zero-rich networks.

7.1. Differential Mapping Method

The defect model reported in [50] is utilized in this work. Note that the value of ANN weights can be either positive or negative, but the weights stored in memristors are represented by the conductance value that can only be positive values. Thus, there are two main mapping methods to make sure the results can be calculated correctly. The general way is to decompose each weight into a positive portion and a negative portion, then map them to two separate memristor columns. Another way is to add an offset to the original weights when mapping them to the crossbars. This will shift all negative weights to the positive range. By doing so, with a certain cost of offset circuitry, half of the crossbars can be saved compared to the two-column mapping method. Both mapping methods suffer severe accuracy loss compared to the ideal case, where the offset method shows much worse tolerance to the defect [114]. Therefore, in this chapter, we only study a two-column mapping scheme.

The traditional mapping scheme decomposes the weight into a positive magnitude portion w+ and a negative magnitude portion w-, and uses two memristor cells to represent positive and negative portions separately. Different from the traditional mapping scheme that decomposes the weight into a positive magnitude portion w+ and a negative magnitude portion w- and sums them to reconstruct the original weight value during the computation, the proposed differential mapping technique represents the weight value by using the difference between two memristor cells.

Given a weight value w scaled to the range [-1,1], then it will be mapped as:

$$w\_a = \begin{cases} 1 & w \geq 0, \\ 1 - \ |w| & w < 0, \end{cases} \qquad w\_b = \begin{cases} 1 - \ w & w > 0, \\ 1 & w \leq 0, \end{cases} \tag{9}$$

where the w_a and w_b are the two memristor cells (with normalized range to [0,1]) used to represent the weight. During the computations, the original weight value is obtained by using w_a – w_b.

Differential mapping scheme ensures a greater number of 1s to be mapped on the memristor-based crossbars. There is always at least one 1 to be mapped on one of the two memristors. At the same time, both memristors will be 1 when the weight value is zero or close to 1 when the weight value is close to 0. In this way, the differential mapping technique can improve the performance significantly, since the failure rate of SA1 fault is higher than SA0 fault and will lead to a more significant accuracy drop, although the differential mapping scheme only improves the tolerance to the SA1 fault.

Figure 42 shows the hardware implementation of the proposed differential mapping method. Compared with traditional hardware implementation [115] that weight values are calculated by two memristor columns and an operational amplifier-based subtractor, the differential mapping method still utilizes a similar arithmetic circuit without increasing hardware cost [116]. When performing differential mapping, the w_a and w_b values are mapped to the memristors in the G+ and G- columns, respectively. When reading the weight value, the current sum of the G+ and G- columns is subtracted by the arithmetic circuit to obtain an effective activation value.

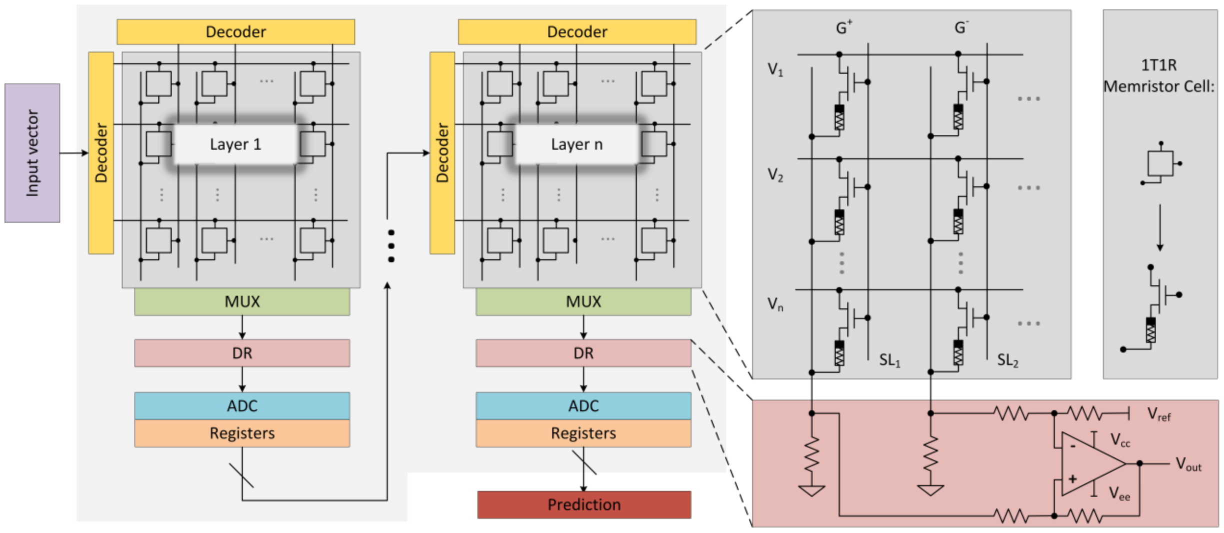


Figure 42. Hardware implementation of the proposed differential mapping scheme.

7.2. Evaluation Methodology and Results

We evaluate the differential mapping method on the CIFAR10 dataset and ImageNet dataset using for an image classification task. The memristor failure model is adopted from [50] with the ratio of SA0 and SA1 fault is 1:5.2. The alternating direction method of multipliers (ADMM)-based pruning algorithm [117] is used for weight pruning during the hierarchical progressive pruning process. All model training, pruning, and accuracy evaluations are conducted on a GPU server using PyTorch. Each accuracy result is obtained by averaging the results of 100 runs.

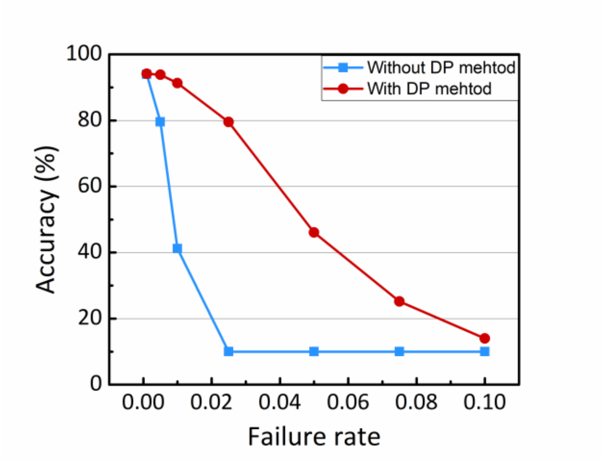


Figure 43. The recognition accuracy on the CIFAR10 dataset under different failure rates.

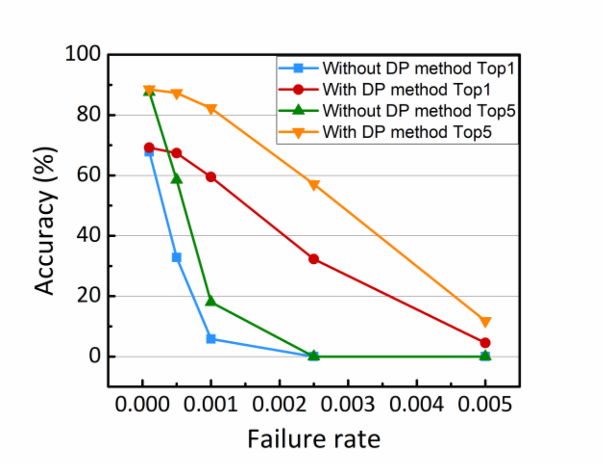


Figure 44. The recognition accuracy on the ImageNet under different failure rates.

7.3. Conclusion

We first evaluate the model fault tolerance via model accuracy on the image classification task. As shown in Figure 43 and Figure 44, we compare the model fault tolerance results optimized by differential mapping method to the original model with traditional two-column mapping. On the CIFAR10 dataset, the original model accuracy is 94.1% without introducing the stuck-at faults. Under failure rate of 0.001, differential mapping method only has 0.2% accuracy drop on average, where the accuracy drop of the traditional mapping is 1.2%. It can be observed that a severe accuracy drop occurs to the traditional mapping under a failure rate of 0.005, where the differential mapping method can preserve a high accuracy under a failure rate of 0.01. For the ImageNet, since the classification task on ImageNet is harder than CIFAR10, the network is more sensitive to the stuck-at faults. As we can see, the differential mapping method clearly provides a better fault tolerance than the traditional two-column mapping.

All the results show that the differential mapping method can tolerate almost an order of magnitude higher failure rate than the traditional two-column method, which demonstrates the effectiveness of the differential mapping method.

8. CONCLUSION AND FUTURE WORK

This chapter summarizes the contributions presented within this dissertation and shows the improvement of state-of-the-art technologies. A direction for future work will also be introduced.

In this research, the TiO_2/TiO_{2-x} 40 μm x 40 μm two-terminal memristor crossbar array that is composed of 20 x 20 devices is fabricated. Based on this fabricated memristor, one of the objectives of this research was to model the total cycle-to-cycle variation that is generated for one memristor at one update process with n pulses. After Gaussian distribution fitting, we get the α values with different pulse widths. Both slopes of the linear fitting for α values of LTP and LTD are negative. Therefore, increasing the pulse width does not increase the cycle-to-cycle variation when using the same number of pulses to tune conductance. This conclusion provides a solid foundation for subsequent research on mitigating the effect of the cycle-to-cycle variation and relative techniques.

The goal of the second phase of this research was to propose and determine the effectiveness of the level scaling method and the pulse regulating method that are simple and feasible universal methods to effectively mitigate the impact of cycle-to-cycle variation. Under cycle-to-cycle variation, the recognition accuracy in the maximum number of the levels is not optimal for the real device. As for the multilevel memristors, the level scaling method can be used to optimize the neuromorphic computing system by selecting appropriately the number of the levels. Similarly, the pulse regulating method mitigates the impact of cycle-to-cycle variation by compressing the number of updating pulses to one.

The third objective of this research was to reduce energy consumption, decrease writing latency, and improve the timing regularity of memristor-based smart edge computing in IoT systems. Instead of modifying the traditional algorithm-based technology, the pulse regulating method that only needs to add a multiplexer circuit before every writing operation in the weight-updating process, to optimize the performance of the system. The results indicate that it saves the writing energy of the crossbar array by 7.7%-26.9% and reduces the writing latency by 30.0%-50.0%. It concludes: 1) The proposed pulse regulating method enables low energy consumption and even pulse distribution to reduce the heat resulted from intensive pulses. 2) Because the number of pulses for weight update is compressed to one, the pulse regulating method effectively reduces the writing latency and improves timing regularity. 3) The pulse regulating method is still effective under different nonlinearity, failure rates, aged devices, architectures, and variation circumstances in memristor-based ANNs.

In addition, the proposed differential mapping method is evaluated for optimizing fault tolerance of memristor-based ANNs system and compared with the traditional two-column mapping. The differential mapping method clearly provides a better fault tolerance than the traditional two-column mapping, which demonstrates differential mapping method can tolerate almost an order of magnitude higher failure rate than the traditional two-column method.

According to the testing results with different pulse width, the changes of conductance vary in different pulse width. A wider pulse changes more conductance compared with a narrower pulse. The mechanism of effect of pulse variation and jitter is the same with the cycle-to-cycle variation because they all influence the formation of subfilaments through the input pulse and then introducing variation. In this way, the effect of pulse variations and jitter can be studied for future work. Simultaneously, even though those proposed techniques show significant improvements for mitigating the impact of the cycle-to-cycle variation, reducing energy consumption and system latency, and enhancing fault tolerance based on simulation, which paves the way for the adoption of memristors for more efficient applications for the era of the IoT, more research needs be done with realistic integrated circuit and tape-out.

REFERENCES

[1] R. H. Weber, and R. Weber, *Internet of things*: Springer, 2010.

[2] A. Fayyazi, M. Ansari, M. Kamal, A. Afzali-Kusha, and M. Pedram, “An ultra low-power memristive neuromorphic circuit for internet of things smart sensors,” *IEEE Internet of Things Journal,* vol. 5, no. 2, pp. 1011-1022, 2018.

[3] Y. Deng, "Deep learning on mobile devices: a review." p. 109930A.

[4] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, L. Jiao, L. Qendro, and F. Kawsar, "Deepx: A software accelerator for low-power deep learning inference on mobile devices." pp. 1-12.

[5] J. Wang, J. Zhang, W. Bao, X. Zhu, B. Cao, and P. S. Yu, "Not just privacy: Improving performance of private deep learning in mobile cloud." pp. 2407-2416.

[6] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Future generation computer systems,* vol. 29, no. 7, pp. 1645-1660, 2013.

[7] M. A. Ponti, L. S. F. Ribeiro, T. S. Nazare, T. Bui, and J. Collomosse, "Everything you wanted to know about deep learning for computer vision but were afraid to ask." pp. 17-41.

[8] R. Reed, and R. J. MarksII, *Neural smithing: supervised learning in feedforward artificial neural networks*: Mit Press, 1999.

[9] R. Hecht-Nielsen, "Theory of the backpropagation neural network," *Neural networks for perception*, pp. 65-93: Elsevier, 1992.

[10] C. Lemaréchal, “Cauchy and the gradient method,” *Doc Math Extra,* vol. 251, no. 254, pp. 10, 2012.

[11] L. Bottou, “Online learning and stochastic approximations,” *On-line learning in neural networks,* vol. 17, no. 9, pp. 142, 1998.

[12] G. F. Simmons, and G. F. Simmons, *Calculus with analytic geometry*: McGraw-Hill New York, 1996.

[13] R. S. Williams, “What's Next?[The end of Moore's law],” *Computing in Science & Engineering,* vol. 19, no. 2, pp. 7-13, 2017.

[14] M. M. Waldrop, “The chips are down for Moore’s law,” *Nature News,* vol. 530, no. 7589, pp. 144, 2016.

[15] L. Chua, “Memristor-the missing circuit element,” *IEEE Transactions on circuit theory,* vol. 18, no. 5, pp. 507-519, 1971.

[16] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, “The missing memristor found,” *nature,* vol. 453, no. 7191, pp. 80-83, 2008.

[17] Z. Wang, S. Joshi, S. Savel’ev, W. Song, R. Midya, Y. Li, M. Rao, P. Yan, S. Asapu, and Y. Zhuo, “Fully memristive neural networks for pattern classification with unsupervised learning,” *Nature Electronics,* vol. 1, no. 2, pp. 137-145, 2018.

[18] M. A. Zidan, J. P. Strachan, and W. D. Lu, “The future of electronics based on memristive systems,” *Nature Electronics,* vol. 1, no. 1, pp. 22-29, 2018.

[19] L. Xia, T. Tang, W. Huangfu, M. Cheng, X. Yin, B. Li, Y. Wang, and H. Yang, "Switched by input: Power efficient structure for RRAM-based convolutional neural network." pp. 1-6.

[20] J. Fu, Z. Liao, N. Gong, and J. Wang, “Mitigating Nonlinear Effect of Memristive Synaptic Device for Neuromorphic Computing,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2019.

[21] J. Fu, Z. Liao, and J. Wang, “Memristor-Based Neuromorphic Hardware Improvement for Privacy-Preserving ANN,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2019.

[22] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, “TEAM: Threshold adaptive memristor model,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 1, pp. 211-221, 2012.

[23] M. D. Pickett, and R. S. Williams, “Sub-100 fJ and sub-nanosecond thermally driven threshold switching in niobium oxide crosspoint nanodevices,” *Nanotechnology*, vol. 23, no. 21, pp. 215202, 2012.

[24] A. C. Torrezan, J. P. Strachan, G. Medeiros-Ribeiro, and R. S. Williams, “Sub-nanosecond switching of a tantalum oxide memristor,” *Nanotechnology*, vol. 22, no. 48, pp. 485203, 2011.

[25] M.-J. Lee, C. B. Lee, D. Lee, S. R. Lee, M. Chang, J. H. Hur, Y.-B. Kim, C.-J. Kim, D. H. Seo, and S. Seo, “A fast, high-endurance and scalable non-volatile memory device made from asymmetric Ta 2 O 5− x/TaO 2− x bilayer structures,” *Nature materials*, vol. 10, no. 8, pp. 625, 2011.

[26] S. Pi, P. Lin, and Q. Xia, “Cross point arrays of 8 nm× 8 nm memristive devices fabricated with nanoimprint lithography,” *Journal of Vacuum Science & Technology B, Nanotechnology and Microelectronics: Materials, Processing, Measurement, and Phenomena*, vol. 31, no. 6, pp. 06FA02, 2013.

[27] Q. Xia, W. Robinett, M. W. Cumbie, N. Banerjee, T. J. Cardinali, J. J. Yang, W. Wu, X. Li, W. M. Tong, and D. B. Strukov, “Memristor− CMOS hybrid integrated circuits for reconfigurable logic,” *Nano letters*, vol. 9, no. 10, pp. 3640-3645, 2009.

[28] R. Waser, and M. Aono, “Nanoionics-based resistive switching memories,” *Nanoscience And Technology: A Collection of Reviews from Nature Journals*, pp. 158-165, 2010.

[29] A. Marchewka, R. Waser, and S. Menzel, "Physical simulation of dynamic resistive switching in metal oxides using a Schottky contact barrier model." pp. 297-300.

[30] C. Baeumer, C. Schmitz, A. Marchewka, D. N. Mueller, R. Valenta, J. Hackl, N. Raab, S. P. Rogers, M. I. Khan, and S. Nemsak, “Quantifying redox-induced Schottky barrier variations in memristive devices via in operando spectromicroscopy with graphene electrodes,” *Nature communications*, vol. 7, no. 1, pp. 1-7, 2016.

[31] X. Guan, S. Yu, and H.-S. P. Wong, “On the switching parameter variation of metal-oxide RRAM—Part I: Physical modeling and simulation methodology,” *IEEE Transactions on electron devices*, vol. 59, no. 4, pp. 1172-1182, 2012.

[32] M. Uddin, M. S. Hasan, and G. S. Rose, "On the Theoretical Analysis of Memristor based True Random Number Generator." pp. 21-26.

[33] C. Baeumer, R. Valenta, C. Schmitz, A. Locatelli, T. O. Menteş, S. P. Rogers, A. Sala, N. Raab, S. Nemsak, and M. Shim, “Subfilamentary networks cause cycle-to-cycle variability in memristive devices,” *ACS nano*, vol. 11, no. 7, pp. 6921-6929, 2017.

[34] J.-H. Lee, D.-H. Lim, H. Jeong, H. Ma, and L. Shi, “Exploring Cycle-to-Cycle and Device-to-Device Variation Tolerance in MLC Storage-Based Neural Network Training,” *IEEE Transactions on Electron Devices*, vol. 66, no. 5, pp. 2172-2178, 2019.

[35] T. Tuma, A. Pantazi, M. Le Gallo, A. Sebastian, and E. Eleftheriou, “Stochastic phase-change neurons,” *Nature nanotechnology,* vol. 11, no. 8, pp. 693, 2016.

[36] H. Jiang, D. Belkin, S. E. Savel’ev, S. Lin, Z. Wang, Y. Li, S. Joshi, R. Midya, C. Li, and M. Rao, “A novel true random number generator based on a stochastic diffusive memristor,” *Nature communications,* vol. 8, no. 1, pp. 1-9, 2017.

[37] R. Zhu, S. Chang, H. Wang, Q. Huang, J. He, and F. Yi, “A versatile and accurate compact model of memristor with equivalent resistor topology,” *IEEE Electron Device Letters,* vol. 38, no. 10, pp. 1367-1370, 2017.

[38] S. Choi, P. Sheridan, and W. D. Lu, “Data clustering using memristor networks,” *Scientific reports,* vol. 5, no. 1, pp. 1-10, 2015.

[39] M. A. Lastras-Montaño, and K.-T. Cheng, “Resistive random-access memory based on ratioed memristors,” *Nature Electronics,* vol. 1, no. 8, pp. 466-472, 2018.

[40] S. Kim, M. Lim, Y. Kim, H.-D. Kim, and S.-J. Choi, “Impact of synaptic device variations on pattern recognition accuracy in a hardware neural network,” *Scientific reports,* vol. 8, no. 1, pp. 2638, 2018.

[41] A. P. James, "Introduction to neuro-memristive systems," *Deep Learning Classifiers with Memristive Networks*, pp. 3-12: Springer, 2020.

[42] Y. Li, H.-H. Shen, C. Li, and F. Zhang, "An efficient parity rearrangement coding scheme for RRAM thermal crosstalk effects." pp. 20-23.

[43] P. Sun, N. Lu, L. Li, Y. Li, H. Wang, H. Lv, Q. Liu, S. Long, S. Liu, and M. Liu, “Thermal crosstalk in 3-dimensional RRAM crossbar array,” *Scientific reports,* vol. 5, no. 1, pp. 1-9, 2015.

[44] Y. V. Pershin, and M. Di Ventra, “Practical approach to programmable analog circuits with memristors,” *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 57, no. 8, pp. 1857-1864, 2010.

[45] T. Ninomiya, S. Muraoka, Z. Wei, R. Yasuhara, K. Katayama, and T. Takagi, “Improvement of Data Retention During Long-Term Use by Suppressing Conductive Filament Expansion in ${\rm TaO} _ {x} $ Bipolar-ReRAM,” *IEEE electron device letters,* vol. 34, no. 6, pp. 762-764, 2013.

[46] B. Chen, J. F. Kang, B. Gao, Y. X. Deng, L. F. Liu, X. Y. Liu, Z. Fang, H. Y. Yu, X. P. Wang, and G. Q. Lo, “Endurance degradation in metal oxide-based resistive memory induced by oxygen ion loss effect,” *IEEE electron device letters,* vol. 34, no. 10, pp. 1292-1294, 2013.

[47] U. Russo, D. Ielmini, C. Cagli, and A. L. Lacaita, “Self-accelerated thermal dissolution model for reset programming in unipolar resistive-switching memory (RRAM) devices,” *IEEE Transactions on Electron Devices,* vol. 56, no. 2, pp. 193-200, 2009.

[48] L. Chen, J. Li, Y. Chen, Q. Deng, J. Shen, X. Liang, and L. Jiang, "Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar." pp. 19-24.

[49] C. Liu, M. Hu, J. P. Strachan, and H. Li, "Rescuing memristor-based neuromorphic design with high defects." pp. 1-6.

[50] C.-Y. Chen, H.-C. Shih, C.-W. Wu, C.-H. Lin, P.-F. Chiu, S.-S. Sheu, and F. T. Chen, “RRAM defect modeling and failure analysis based on march test and a novel squeeze-search scheme,” *IEEE Transactions on Computers,* vol. 64, no. 1, pp. 180-190, 2014.

[51] L. Xia, M. Liu, X. Ning, K. Chakrabarty, and Y. Wang, "Fault-tolerant training with on-line fault detection for RRAM-based neural computing systems." pp. 1-6.

[52] B. Zhang, N. Uysal, D. Fan, and R. Ewetz, “Handling stuck-at-fault defects using matrix transformation for robust inference of dnns,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2448-2460, 2019.
[53] R. E. Pino, H. Li, Y. Chen, M. Hu, and B. Liu, "Statistical memristor modeling and case study in neuromorphic computing." pp. 585-590.
[54] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.
[55] P.-Y. Chen, X. Peng, and S. Yu, "NeuroSim+: An integrated device-to-algorithm framework for benchmarking synaptic devices and array architectures." pp. 6.1. 1-6.1. 4.
[56] S. Nandakumar, M. Le Gallo, I. Boybat, B. Rajendran, A. Sebastian, and E. Eleftheriou, "Mixed-precision architecture based on computational memory for training deep neural networks." pp. 1-5.
[57] S. Nandakumar, I. Boybat, M. Le Gallo, A. Sebastian, B. Rajendran, and E. Eleftheriou, "Supervised learning in spiking neural networks with MLC PCM synapses." pp. 1-2.
[58] N. Gong, T. Idé, S. Kim, I. Boybat, A. Sebastian, V. Narayanan, and T. Ando, “Signal and noise extraction from analog memory elements for neuromorphic computing,” *Nature communications*, vol. 9, no. 1, pp. 1-8, 2018.
[59] P.-Y. Chen, X. Peng, and S. Yu, “NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 37, no. 12, pp. 3067-3080, 2018.
[60] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication." p. 19.
[61] J. Rajendran, H. Maenm, R. Karri, and G. S. Rose, "An approach to tolerate process related variations in memristor-based applications." pp. 18-23.
[62] L. Chen, J. Li, Y. Chen, Q. Deng, J. Shen, X. Liang, and L. Jiang, "Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar." pp. 19-24.
[63] P.-Y. Chen, B. Lin, I.-T. Wang, T.-H. Hou, J. Ye, S. Vrudhula, J.-s. Seo, Y. Cao, and S. Yu, "Mitigating effects of non-ideal synaptic device characteristics for on-chip learning." pp. 194-199.
[64] K. M. Kim, J. J. Yang, J. P. Strachan, E. M. Grafals, N. Ge, N. D. Melendez, Z. Li, and R. S. Williams, “Voltage divider effect for the improvement of variability and endurance of TaO x memristor,” *Scientific reports,* vol. 6, pp. 20085, 2016.
[65] B. Liu, H. Li, Y. Chen, X. Li, Q. Wu, and T. Huang, "Vortex: variation-aware training for memristor x-bar." p. 15.
[66] J. Rajendran, R. Karri, and G. S. Rose, “Improving tolerance to variations in memristor-based applications using parallel memristors,” *IEEE Transactions on Computers,* vol. 64, no. 3, pp. 733-746, 2014.
[67] J. Rajendran, R. Karri, and G. S. Rose, "Parallel memristors: Improving variation tolerance in memristive digital circuits." pp. 2241-2244.
[68] Y. Li, S. Long, H. Lv, Q. Liu, Y. Wang, S. Zhang, W. Lian, M. Wang, K. Zhang, and H. Xie, “Improvement of resistive switching characteristics in ZrO2 film by embedding a thin TiOx layer,” *Nanotechnology,* vol. 22, no. 25, pp. 254028, 2011.
[69] A. M. Rana, T. Akbar, M. Ismail, E. Ahmad, F. Hussain, I. Talib, M. Imran, K. Mehmood, K. Iqbal, and M. Y. Nadeem, “Endurance and cycle-to-cycle uniformity

improvement in tri-layered CeO 2/Ti/CeO 2 resistive switching devices by changing top electrode material," *Scientific reports*, vol. 7, pp. 39539, 2017.

[70] D. Niu, Y. Chen, and Y. Xie, "Low-power dual-element memristor based memory design." pp. 25-30.

[71] J. Wang, Y. Tim, W.-F. Wong, and H. H. Li, "A practical low-power memristor-based analog neural branch predictor." pp. 175-180.

[72] H. Saadeldeen, D. Franklin, G. Long, C. Hill, A. Browne, D. Strukov, T. Sherwood, and F. T. Chong, "Memristors for neural branch prediction: a case study in strict latency and write endurance challenges." pp. 1-10.

[73] D. Niu, Y. Xiao, and Y. Xie, "Low power memristor-based ReRAM design with error correcting code." pp. 79-84.

[74] S. Li, W. Chen, Y. Luo, J. Hu, P. Gao, J. Ye, K. Kang, H. Chen, E. Li, and W.-Y. Yin, "Fully coupled multiphysics simulation of crosstalk effect in bipolar resistive random access memory," *IEEE Transactions on Electron Devices*, vol. 64, no. 9, pp. 3647-3653, 2017.

[75] Y. Luo, W. Chen, M. Cheng, and W.-Y. Yin, "Electrothermal characterization in 3-D resistive random access memory arrays," *IEEE Transactions on Electron Devices*, vol. 63, no. 12, pp. 4720-4728, 2016.

[76] D.-W. Wang, W. Chen, W.-S. Zhao, G.-D. Zhu, K. Kang, P. Gao, J. E. Schutt-Aine, and W.-Y. Yin, "Fully Coupled Electrothermal Simulation of Large RRAM Arrays in the "Thermal-House"," *IEEE Access*, vol. 7, pp. 3897-3908, 2018.

[77] S. Kannan, N. Karimi, R. Karri, and O. Sinanoglu, "Modeling, detection, and diagnosis of faults in multilevel memristor memories," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 5, pp. 822-834, 2015.

[78] P. Sheridan, and W. D. Lu, "Defect consideratons for robust sparse coding using memristor arrays." pp. 137-138.

[79] H. Manem, G. S. Rose, X. He, and W. Wang, "Design considerations for variation tolerant multilevel cmos/nano memristor memory." pp. 287-292.

[80] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, and C. E. Graves, "Analogue signal and image processing with large memristor crossbars," *Nature electronics*, vol. 1, no. 1, pp. 52-59, 2018.

[81] G. Yuan, C. Ding, R. Cai, X. Ma, Z. Zhao, A. Ren, B. Yuan, and Y. Wang, "Memristor crossbar-based ultra-efficient next-generation baseband processors." pp. 1121-1124.

[82] R. Cai, A. Ren, Y. Wang, and B. Yuan, "Memristor-based discrete fourier transform for improving performance and energy efficiency." pp. 643-648.

[83] K. M. Kim, J. J. Yang, J. P. Strachan, E. M. Grafals, N. Ge, N. D. Melendez, Z. Li, and R. S. Williams, "Voltage divider effect for the improvement of variability and endurance of TaO x memristor," *Scientific reports*, vol. 6, no. 1, pp. 1-6, 2016.

[84] S. Yu, X. Guan, and H.-S. P. Wong, "On the switching parameter variation of metal oxide RRAM—Part II: Model corroboration and device design strategy," *IEEE Transactions on Electron Devices*, vol. 59, no. 4, pp. 1183-1188, 2012.

[85] A. Fantini, L. Goux, R. Degraeve, D. Wouters, N. Raghavan, G. Kar, A. Belmonte, Y.-Y. Chen, B. Govoreanu, and M. Jurczak, "Intrinsic switching variability in HfO 2 RRAM." pp. 30-33.

[86] S. Balatti, S. Ambrogio, D. C. Gilmer, and D. Ielmini, “Set variability and failure induced by complementary switching in bipolar RRAM,” *IEEE electron device letters,* vol. 34, no. 7, pp. 861-863, 2013.

[87] C. La Torre, K. Fleck, S. Starschich, E. Linn, R. Waser, and S. Menzel, “Dependence of the SET switching variability on the initial state in HfOx‐based ReRAM,” *physica status solidi (a),* vol. 213, no. 2, pp. 316-319, 2016.

[88] B. J. Choi, A. C. Torrezan, K. J. Norris, F. Miao, J. P. Strachan, M.-X. Zhang, D. A. Ohlberg, N. P. Kobayashi, J. J. Yang, and R. S. Williams, “Electrical performance and scalability of Pt dispersed SiO_2 nanometallic resistance switch,” *Nano letters,* vol. 13, no. 7, pp. 3213-3217, 2013.

[89] Y. Hayakawa, A. Himeno, R. Yasuhara, W. Boullart, E. Vecchio, T. Vandeweyer, T. Witters, D. Crotti, M. Jurczak, and S. Fujii, "Highly reliable TaO x ReRAM with centralized filament for 28-nm embedded application." pp. T14-T15.

[90] S. Cho, C. Yun, S. Tappertzhofen, A. Kursumovic, S. Lee, P. Lu, Q. Jia, M. Fan, J. Jian, and H. Wang, “Self-assembled oxide films with tailored nanoscale ionic and electronic channels for controlled resistive switching,” *Nature communications,* vol. 7, no. 1, pp. 1-10, 2016.

[91] R. Switching, “From Fundamentals of Nanoionic Redox Processes to Memristive Device Applications,” *Wiley-VCH*, 2016.

[92] Y. Yang, P. Gao, S. Gaba, T. Chang, X. Pan, and W. Lu, “Observation of conducting filament growth in nanoscale resistive memories,” *Nature communications,* vol. 3, pp. 732, 2012.

[93] T. Chang, S.-H. Jo, K.-H. Kim, P. Sheridan, S. Gaba, and W. Lu, “Synaptic behaviors and modeling of a metal oxide memristive device,” *Applied physics A,* vol. 102, no. 4, pp. 857-863, 2011.

[94] C. Ye, J. Wu, G. He, J. Zhang, T. Deng, P. He, and H. Wang, “Physical mechanism and performance factors of metal oxide based resistive switching memory: a review,” *Journal of Materials Science & Technology,* vol. 32, no. 1, pp. 1-11, 2016.

[95] D. S. Lemons, and P. Langevin, *An introduction to stochastic processes in physics*: JHU Press, 2002.

[96] T. Tang, L. Xia, B. Li, R. Luo, Y. Chen, Y. Wang, and H. Yang, "Spiking neural network with rram: Can we use it for real-world application?." pp. 860-865.

[97] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *nature,* vol. 323, no. 6088, pp. 533-536, 1986.

[98] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of machine learning research,* vol. 12, no. 7, 2011.

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

[100] D. P. Kingma, and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[101] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, “Nanoscale memristor device as synapse in neuromorphic systems,” *Nano letters,* vol. 10, no. 4, pp. 1297-1301, 2010.

[102] D. W. Hosmer, and S. Lemesbow, “Goodness of fit tests for the multiple logistic regression model,” *Communications in statistics-Theory and Methods,* vol. 9, no. 10, pp. 1043-1069, 1980.

[103] N. Buduma, and N. Locascio, *Fundamentals of deep learning: Designing next-generation machine intelligence algorithms*: " O'Reilly Media, Inc.", 2017.

[104] X. Peng, S. Huang, H. Jiang, A. Lu, and S. Yu, “DNN+ NeuroSim V2. 0: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators for On-chip Training,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* 2020.

[105] J. Fu, Z. Liao, N. Gong, and J. Wang, "Linear Optimization for Memristive Device in Neuromorphic Hardware." pp. 453-458.

[106] O. Krestinskaya, A. Irmanova, and A. P. James, "Memristive non-idealities: Is there any practical implications for designing neural network chips?." pp. 1-5.

[107] J. Edstrom, D. Chen, Y. Gong, J. Wang, and N. Gong, “Data-pattern enabled self-recovery low-power storage system for big video data,” *IEEE Transactions on Big Data,* vol. 5, no. 1, pp. 95-105, 2017.

[108] A. Irmanova, A. Maan, A. James, and L. Chua, “Analog Self-timed Programming circuits for Aging Memristors,” *IEEE Transactions on Circuits and Systems II: Express Briefs,* 2020.

[109] P.-Y. Chen, and S. Yu, "Reliability perspective of resistive synaptic devices on the neuromorphic system performance." pp. 5C. 4-1-5C. 4-4.

[110] X. Glorot, and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks." pp. 249-256.

[111] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." pp. 1026-1034.

[112] G. Franchi, A. Bursuc, E. Aldea, S. Dubuisson, and I. Bloch, "TRADI: Tracking deep neural network weight distributions."

[113] K. Simonyan, and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[114] G. Yuan, Z. Liao, X. Ma, Y. Cai, Z. Kong, X. Shen, J. Fu, Z. Li, C. Zhang, and H. Peng, "Improving DNN Fault Tolerance using Weight Pruning and Differential Crossbar Mapping for ReRAM-based Edge AI." pp. 135-141.

[115] O. Krestinskaya, B. Choubey, and A. James, “Memristive GAN in Analog,” *Scientific reports,* vol. 10, no. 1, pp. 1-14, 2020.

[116] O. Krestinskaya, A. P. James, and L. O. Chua, “Neuromemristive circuits for edge computing: A review,” *IEEE transactions on neural networks and learning systems,* vol. 31, no. 1, pp. 4-23, 2019.

[117] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang, "A systematic dnn weight pruning framework using alternating direction method of multipliers." pp. 184-199.