MITIGATING NONLINEAR EFFECT AND PRESERVING PRIVACY FOR MEMRISTOR BASED ON-CHIP NEURAL NETWORK

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

By

Jingyan Fu

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

Major Department:
Electrical and Computer Engineering

December 2021

Fargo, North Dakota

North Dakota State University
Graduate School

Title

MITIGATING NONLINEAR EFFECT AND PRESERVING PRIVACY FOR MEMRISTOR BASED ON-CHIP NEURAL NETWORK

By

Jingyan Fu

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. Danling Wang
Chair

Dr. Qifeng Zhang

Dr. Sumitha George

Dr. Mijia Yang

Approved:

December 13, 2021
Date

Dr. Benjamin Braaten
Department Chair

ABSTRACT

Memristors offer advantages as a hardware solution for neuromorphic computing, however, their non-ideal property makes the weight update difficult and reduces the accuracy of a neural network. Also, a large amount of personal data has raised great concern about the privacy preservation of neural networks. Thus, the performance of memristor-based neural networks gets worse when considering non-ideal property and introducing a privacy preservation mechanism. This dissertation focuses on improving the performance of a memristor-based privacy-preserving neural network.

A piecewise linear (PL) method is proposed to mitigate the nonlinear effect of memristors by calculating the weight update parameters along a piecewise line, which reduces errors in the weight update process. It mitigates the nonlinearity impact without reading the precise conductance of the memristor in each updating step, thereby avoiding complex peripheral circuits. What's more., the PL method is proved to be an effective technique that can prevent accuracy loss and increase privacy preservation space for privacy-preserving ANN. Also, we propose a Noise Distribution Normalization (NDN) method to add Gaussian distributed noise through hardware implementation, thereby achieving differential privacy in edge AI. Instead of using traditional algorithmic noise-insertion methods, we take advantage of inherent cycle-to-cycle variations of memristors during the weight-update process as the noise source, which does not incur extra software or hardware overhead.

ACKNOWLEDGMENTS

Thanks to all the people that supported my effort on this dissertation.

I would like to express my deepest gratitude to my advisor Dr. Jinhui Wang, who read my numerous revisions, helped make some sense of the confusion, and persistently encouraged me. and conveyed the spirit and guidance required for the research. Also, I am deeply indebted to my advisors Dr. Danling Wang, Dr. Scott C. Smith, and Dr. Sudarshan K. Srinivasan. Without their invaluable guidance this project would not have been possible. I would also like to thank Dr. Qifeng Zhang, Dr. Mijia Yang, Dr. Sumitha George, who shared me expertise and valuable guidance.

I am extremely grateful to Dr. Khang Hoang. I am extremely thankful and indebted to him for supporting, sharing and encouragement. I am also grateful to all staff at Center for Computationally Assisted Science and Technology (CCAST). I would also like to thank Fred Haring in R2 at Center for Nanoscale Science and Engineering of North Dakota State University.

DEDICATION

To my family.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

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

SGD.....Stochastic Gradient Descent

VLSI.....Very Large-Scale Integration

1. INTRODUCTION

1.1. Background

Device scaling on the CMOS technology that effectively benefits the cost, speed, and power efficiency of Integrated Circuits (ICs), has driven the boom of the computing systems over the past several decades. Unfortunately, the increasing physical restrictions such as thermodynamic limits including quantum tunneling [1] make COMS device scaling reach a plateau, which suppresses the further progress of computing systems in large data centers or the Cloud, and also obstruct the fast deployment of sensors and actuators for the Internet of Things (IoT) [2]. Meanwhile, neuromorphic computing using bio-inspired learning algorithms has emerged and achieved tremendous success. However, due to the separated computation unit and memories [3], conventional von Neumann-based computing platform results in insufficient bandwidth and overlarge power consumption, particularly for the fast training and/or classification tasks in neuromorphic computing, such as real-time online recognition. Therefore, new synapse devices and technologies with remarkable performance and less cost are urgently needed.

Memristors are one of such promising devices. A memristor is an analog device to exploit the multilevel conductance states, to regulate the flow of electrical current, and to store the amount of charge that has previously flowed through it [4, 5]. In a memristor-based neuromorphic computing platform, memristors are connected as neuron devices that implement the function of the Sum of Product (SOP). Weight update depends on the transition between different conductance states in memristors, which is typically triggered by voltage pulses. Specifically, according to the programmed voltage pulses, the conductance of memristors positively and negatively changes, enabling weight increase and decrease, respectively [3]. As a result, it can not only act as the non-volatile memory, but realize the in-memory calculation which brings high potential to implement

neural networks in hardware. Different from traditional CMOS-based hardware, such as Field Programmable Gate Array (FPGA), Application Specific Integrated Circuit (ASIC), Graphics Processing Unit (GPU), and Tensor Processing Unit (TPU), the memristor-based in-memory computing architecture breaks the memory wall that results from the Von Neumann architecture, thereby achieving at least a 7× reduction of active power for implementing a basic neuron function [6]. Also, memristors have better CMOS process compatibility as compared with emerging technologies such as quantum computing, molecular computing, quantum dots, and spin-wave devices. However, memristors are not flawless and intrinsically suffering from their variability issues. The physical mechanism of the conductance modulation in most prospective synaptic devices is typically an ionic reconfiguration process based on electro/thermo-dynamics [7, 8]. This atomic-level random process is responsible for unavoidable variations including nonlinearity, device-to-device, cycle-to-cycle, and ON/OFF conductance variations [9]. The nonlinearity property makes it challenging to determine a proper width or amplitude of input signals for achieving the desired conductance of memristors. The cycle-to-cycle variations result in the different updated conductance when the same updating signal in different updating cycles is applied to a memristor, even when the initial conductance is the same.

Recent years have witnessed significant progress in mobile devices and wireless sensor networks, creating unprecedented opportunities to deploy deep learning for smart IoT applications. However, when deep learning algorithms work on the edge for human-related applications, IoT devices will collect data, which in some cases may contain quite personal and high-value user information, therefore raising significant concerns regarding information privacy. It is thus possible for the private data to be misused, or to be hacked by outside attacks. Therefore, some sort of privacy protection method is required to guarantee a strong notion of privacy, while

preserving learning accuracy. Differential privacy [10] is one of the most popular technologies for privacy-preserving deep learning, realized by introducing perturbations. Differential privacy (DP) provides a mathematical constraint for the privacy loss associated with any data release from a statistical database. However, differentially private learning systems not only need to train complex models, but also must perform an additional computation for noise insertion as a protective mechanism to datasets, models, and algorithms [11]. Such a high-cost training process undoubtedly challenges traditional CMOS based hardware technology, and hinders development of deep learning, especially for power-sensitive and resource-limited edge computing in IoT systems. Therefore, instead of traditional CMOS based hardware technology, a memristor based differentially private learning system, with its low computation and storage cost, is an excellent candidate for edge AI.

Thus, this dissertation focuses on improving the performance of memristor based privacy preserving neural network.

1.2. Research Objective

Memristor-based neuromorphic computing is such a flexible and attractive technology to meet the increasing needs of data processing [4, 5]. A memristor is a device with only three layers structure that can not only realize desirable device properties such as sub-10 nm feature sizes [12], sub-nanosecond switching speed [13, 14], long write-erase endurance [15], and nanoamperes programming energy [16], but also exploit multilevel conductance states [5]. However, memristors are not flawless, and intrinsically suffer from variability issues. The physical mechanism of the conductance modulation in most prospective synaptic devices is typically an ionic reconfiguration process based on electro/thermo-dynamics [7, 8]. This atomic-level random process is responsible for unavoidable variations including nonlinearity, device-to-device, cycle-to-cycle, and ON/OFF

conductance variations [9]. These non-ideal properties extremely degrade the performance of a memristor-based circuit. For instance, the nonlinearity makes it challenging to determine a proper width or amplitude of input signals for achieving the desired conductance of memristors. It is reported that the linear conductance change is the major requirement of memristor-based neuromorphic computing to realize high accuracy for online learning [3]. Four state-of-the-art memristors in literature – Ag:a-Si [17], TaO_x/TiO_2 [18], PCMO [19], and $AlOx/HfO_2$ [20] are all characterized by device nonlinearity. As discussed in [9, 17], the accuracy of recognition based on Ag:a-Si with nonlinearity decreases over 20% compared to without nonlinearity. Therefore, firstly, my research objective is to propose new techniques that can mitigate the negative impact of memristor's variations, especially the nonlinearity variation.

To build a memristor-based neuromorphic computing system, another non-negligible point is privacy protection because usually the network is built from a large amount of private data training. Effective privacy protection technologies of memristor-based neuromorphic system are urgently needed A privacy preserving method is to introduce a randomized noise mechanism for differential privacy technology to quantify the protection ability. However, noisy and distorted data would lead to a degradation of recognition accuracy in ANN. Accordingly, solutions to balance privacy preserving and recognition accuracy are indeed needed. Therefore, secondly, my research objective is applying our proposed techniques on a memristor based privacy preserving neural network to counteract recognition degradation due to noise injection.

Apart from the nonlinearity of memristors, which is mentioned above, the other non-ideal properties of memristors also degrade the performance of a memristor-based neuromorphic computing circuit. These properties include device-to-device variation, cycle-to-cycle variation, maximum conductance variation, and minimum conductance variation. Although these non-ideal

properties influence learning precision of a memristor-based system, such variations can also be considered as inherent resources for noise generation. Also, the privacy preserving method needs to introduce such randomized noise. Therefore, a promising way is to utilize variations of memristor. By this method, it is possible to add Gaussian noise distribution to a system without adding computational complexity and introducing extra hardware. Therefore, thirdly, my objective is to convert the negative impact of memristor's cycle-to-cycle variation into a positive impact by utilizing the memristor for privacy-preserving neural networks.

1.3. Contributions

This dissertation focuses on improving the performance of memristor based privacy preserving neural network.

First, a piecewise linear (PL) method is proposed to mitigate the nonlinear effect of memristors by calculating the weight update parameters along a piecewise line, which reduces errors in the weight update process. It mitigates the nonlinearity impact without reading the precise conductance of the memristor in each updating step, thereby avoiding complex peripheral circuits. This method makes the following contributions:

- A PL method is proposed to effectively mitigate the impact of memristor's nonlinearity property. Without reading precise conductance of memristors, the PL method makes weight update process along a piecewise line and the detailed working flow is presented.
- To explore the PL method and its configuration, the relations among the recognition accuracy, number of segments, split selection strategies, and nonlinearity behaviors are investigated through the Modified National Institute of Standards and Technology (MNIST) database [21] based on simulation platform.

- Based on various configuration models provided by the PL method, the tradeoff analysis is conducted to reduce the cost by selecting an appropriate model.
- By thorough evaluation, the effectiveness of the PL method is verified even with various variations including device-to-device variation, cycle-to-cycle variation, maximum or minimum conductance variation, and ON/OFF ratio variation.

What's more, the PL method is proved to be an effective technique that can prevent accuracy loss and increase privacy preservation space for privacy-preserving ANN. This investigation makes the following contributions:

- A method for mitigating the nonlinearity impact in memristor-based privacy-preserving ANN. To mitigate the impact of memristor's nonlinearity property, an effective, hardware-based PL method with low circuit overhead is proposed, which makes the neuromorphic system become more accurate and applicable for ANN application.
- A mechanism enhancing the immunity of memristor-based privacy-preserving ANN to nonlinearity property of memristor device. By applying the PL method under eight groups of private perturbations that follow differential privacy theory, the recognition accuracy of ANN is proved to get negligible degradation or even get increase than before.
- Thorough evaluation. We evaluate the proposed method on standard image classification tasks [21] and conduct over 1,500 simulations that include 4 models, 8 groups of privacy perturbations, and 49 nonlinearity cases.
- The tradeoff analysis. The PL method provides a variety of configuration models, and we discuss how to reduce the cost by selecting the appropriate model while meeting the privacy and accuracy requirements based on the actual device.

Also, we propose a Noise Distribution Normalization (NDN) method to add Gaussian distributed noise through hardware implementation, thereby achieving differential privacy in edge AI. Instead of using traditional algorithmic noise-insertion methods, we take advantage of inherent cycle-to-cycle variations of memristors during the weight-update process as the noise source, which does not incur extra software or hardware overhead. This method makes the following contributions:

- A hardware solution that breaks the limitations of traditional software-based noise-adding mechanisms of DP. A memristor-based hardware solution is proposed for differentially private learning systems that does not require additional circuitry. In this paper, the positive and negative pulse pair (PN) method is used to generate adjustable Gaussian noise, satisfying the DP constraint. The proposed method transforms non-beneficial cycle-to-cycle variations into a valuable measure for privacy protection.
- Methods that address Differentially Private Stochastic Gradient Descent (DP-SGD) by hardware implementation. The Clipping method is proposed to avoid the L2 norm calculation of gradient matrices. A combination of the PN method and clipping method, called Noise Distribution Normalization (NDN) method, is proposed to implement the DP mechanism.
- Privacy analysis and performance evaluation. Privacy analysis is conducted to verify the effectiveness of proposed methods. Furthermore, to illustrate the performance of each method, a comprehensive suite of simulations has been conducted.

1.4. Organization

This dissertation is organized into 7 chapters. Chapter 2 introduces related work on mitigating nonlinear effect of memristive synaptic device, related work on improving performance of memristor based neuromorphic hardware for privacy preserving neural network, and related

work on improving performance of memristor based variation enabled differentially private learning systems for edge computing in IoT.

Chapter 3 introduces the preliminaries about memristor and its property, differential privacy, and neural network, which will provide the foundation for the rest of the dissertation.

Chapter 4 introduces a new technique that uses pulses programming method to update the weight, following the nonlinear curve of memristors, thereby enhancing the accuracy of the learning algorithms. It mitigates the nonlinearity impact without reading the precise conductance of the memristor in each updating step, thereby avoiding complex peripheral circuits. The effectiveness of the proposed PL method with respectively 2-segment, 3-segment, and 4-segment models in two split selection strategies is investigated and the impact of various variations are considered.

Chapter 5 applies the proposed method in Chapter 3 to enable privacy-preserving ANN without accuracy degradation. The effectiveness of the proposed PL method with respectively 2-segment, 3-segment, and 4-segment models is investigated. The results show that under different nonlinearity and different perturbation noise required by differential privacy theory, the PL method can increase the recognition accuracy of MNIST handwriting digits by 39.67% on average, which provides more space and margin for privacy-preserving technology.

Chapter 6 introduces a Noise Distribution Normalization (NDN) method to add Gaussian distributed noise through hardware implementation, thereby achieving differential privacy in edge AI. Instead of using traditional algorithmic noise-insertion methods, we take advantage of inherent cycle-to-cycle variations of memristors during the weight-update process as the noise source, which does not incur extra software or hardware overhead. In one case study, the proposed method realizes ultra-low-cost DP-SGD (Differentially Private Stochastic Gradient Descent) for edge AI

in IoT systems, achieving a 3.5% to 15.5% average recognition accuracy improvement under different noise levels, as compared with a baseline mechanism.

Chapter 7 summarizes the major conclusions of this dissertation and suggests a direction for future research.

2. RELATED WORK

To optimize the performance of a memristor-based privacy preserving neural network, researchers propose various techniques addressing the issues that include nonlinear effect of memristive synaptic device, privacy preserving neural network, and cycle-to-cycle variation effect of memristive synaptic device. This chapter includes related work on these topics.

2.1. Related Work on Mitigating Nonlinear Effect of Memristive Synaptic Device

For analog applications of the memristor technology, the intrinsic nonlinearity property causes different conductance changes even with the same voltage pulse. Such nonlinear characteristic is also one of the most undesirable non-ideal factors for neuromorphic computing [9, 22-25]. To mitigate the impact of nonlinearity property, investigations have generally proceeded in the following three aspects.

The first aspect is to change the device structure by manufactural modification in order to eliminate nonlinearity property. For example, a thermal enhanced layer is added to confine heat in the switching layer to the HfO_x RRAM device [26]. Another solution is to introduce an ion-diffusion limiting layer for the TiN/TaO_x RRAM device [27]. A more effective way is to utilize a charge trap layer to a gated Schottky diode in order to cancel the nonlinearity factor [28]. Although those devices can achieve relatively better linear property, they may ignore the other important features designated for the neuromorphic computing, such as the ON/OFF ratio and endurance characteristics. For instance, the reconfigurable gated Schottky diode has better linearity, but with low ON/OFF ratio [18, 28], which limits its application as an analog synapse. Moreover, nonlinearity is widespread and intrinsic in almost all memristors. Due to the implementation cost consideration, it is unfeasible nor impractical to create a new manufacturing method to mitigate

nonlinearities for each kind of memristor. As a result, novel effective and feasible approaches are necessary to conduct a synaptic operation.

A more universal aspect is to control the conductance change with the current control, time-domain control, and the flux- and charge-domain control. The current control mechanisms are based on transistor gate voltage and time duration in the configuration of one transistor and one memristor [29, 30]. The flux- and charge-domain control method describes device state as a function of flux or charge and adjusts the device state according to quantization of the flux or charge [31, 32]. In theory, these methods can accurately control the conductance of the memristor. However, the voltage or current inputs required by these methods are too complex to be implemented, since irregular shape pulses are difficult to generate. For example, for the time-domain control [33], the input voltage curve requires an approximation using a 3rd, 4th, 7th, or 9th order function. Since generating that voltage consumes too much time and power, it is often impossible to realize at circuit level. Therefore, an effective solution that is not only suitable for most devices, but also simple enough to realize is desirable.

The third aspect is to utilize simply programmed pulses on memristors. Several programming schemes of the voltage pulse have been developed to achieve controllable conductance modulation. For example, the bipolar-pulse scheme [9, 34] applies a pair of positive and negative pulses with different amplitudes and durations. It partly mitigates nonlinearity at the low conductance stages where usually have large overshoots. However, the nonlinearity at the high conductance stages still exists. Also, in order to obtain precise conductance tuning, in [2, 35, 36], write-and-verify tuning with feedback circuits are used to adjust the device reliably. A linear and symmetric relation is demonstrated but using a much larger digital memory and multiple types of pulses [35]. With these methods, the conductance can be effectively controlled, but it needs to

identify and verify the precise conductance of the device for each weight update. Accordingly, extra processing circuits and a specific pulse generator are necessary, which increases the complexity of the circuit design and leads to area overhead and performance penalty.

In this research, a novel Piecewise Linearity (PL) method is proposed to address the nonlinearity problem for memristors. The proposed PL method, using the same amplitude pulses with simple duration adjustment and without identification and verification of the precise conductance, effectively mitigates the nonlinearity and achieves a high recognition accuracy in neuromorphic computing.

2.2. Related Work on Memristor-Based Neuromorphic Hardware Improvement for Privacy-Preserving Neural Network

With artificial neural networks (ANN) develops rapidly, it powers intelligent products by extracting patterns and building models. Meantime, data privacy greatly impacts our daily life, such as politics, security, businesses, relationships, health, and finances. The privacy problem is not limited to the threats associated with private data exposures or hacking attempts. It is also possible to glean extra information even if the data are anonymized and the ANN models are inaccessible. Privacy-preserving ANN technologies are proposed to make that ANN transform our society positively without risking our sensitive data, which is mainly conducted by cryptographic approaches or differential privacy approaches [37]. Especially, differential privacy, that is more efficient and popular, resists attacks by adding random noise to the input data, to iterations in a certain algorithm, or to the algorithm output. In 2017, the Google security and privacy team released a Private Aggregation via Teacher Ensembles (PATE) framework [38], which scales to learning tasks with large numbers of output classes and un-curated, imbalanced training data with errors, and it was proved as tighter differential-privacy guarantees. In 2018, the ARDEN

framework was proposed to protect the sensitive information via local differentially private and noise training [39]. However, because these technologies are all based on software technologies, the presentence of noise is bound to cause a drop in accuracy and it is impossible to get higher accuracy than that without noise injection. Also, software-based noise injection causes latency problems and computational overhead. Recently, designers proposed to exploit inherent noise with the equivalent error-prone hardware to replace software-based noise to save much power [40], which indicates the hardware can provide a more effective solution to realize privacy-preserving ANN.

In this research, instead of using traditional software method, we propose a method that applies in memristor-based ANN hardware system to improve privacy preserving space for differential privacy technology. The proposed method focuses on mitigating the nonlinearity problem of memristors to enable privacy preserving.

2.3. Related Work on Memristor Based Variation Enabled Differentially Private Learning Systems for Edge Computing in IoT

Most differentially private learning systems for IoT focus on algorithm-based framework improvement and optimization. For example, [41] proposes a framework that uses a protection layer to perform noise injection; and [42] designs a mechanism named LATENT that adds a randomization layer between the convolutional module and the fully connected module to perturb data for machine learning services. In [43], the Google security and privacy team released a Private Aggregation via Teacher Ensembles (PATE) framework that achieved private learning by carefully coordinating the activity of several different machine learning models. Differentially Private Stochastic Gradient Descent (DP-SGD) [44] makes fewer assumptions about the machine learning task than PATE, but it comes at the expense of making modifications to the training

algorithm. Such technologies inevitably need a large cost of noise-generation computing. Furthermore, all the above frameworks are software-based privacy protection technologies that might not be deployable on IoT devices due to resource constraints.

Other researchers have presented novel hardware level solutions. In [40], low-voltage static random-access memory (SRAM) chips are used to add bit failures as training data noise. This method can save energy, but the added noise only followed a uniform distribution, and does not guarantee DP. In [45, 46], in order to generate random numbers with true randomness, dedicated random number generation modules, such as physical unclonable function (PUF) and random number generator, are designed. These modules are accurate but require additional circuitry. In [47], an advanced neuro-morphic system is implemented based on memristor arrays, but noise is inserted in training data to promise strong theoretical privacy guarantees, where the hardware and software are utilized separately for learning and privacy protection.

Different from the above approaches, we propose a method that can achieve differentially private learning in edge AI with high efficiency and low computing cost by taking advantage of memristor-based hardware variations, which does not require any additional hardware or software. In this research, we utilize a 3-layer fully connected network and MNIST database as an example to verify the proposed method. However, our proposed method is generic, and can be applied to other databases and any deep learning models that can be mapped into a memristor-based crossbar array.

3. PRELIMINARIES

This chapter provides preliminaries to this study by first introducing the concept of memristor, followed by discussing non-ideal properties of memristor. Second, besides discussing artificial neural networks (ANN), we also discussed how to build an ANN using memristors. Finally, a differential privacy concept is involved, which will provide the foundation for the rest of the dissertation.

3.1. Artificial Neural Networks (ANN)

3.1.1. Neural Networks

Artificial neural networks (ANNs), or neural networks (NNs), are computing systems inspired by the biological neural networks. An ANN is based on a collection of connected units or nodes called artificial neurons, which model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. To be clear, in this research, such connection is named as synapses. An artificial neuron receives a signal then processes it by propagation function and produce weighted sum. This weighted sum is called activation. This weighted sum is then passed through a (usually nonlinear) activation function to produce the output. The initial inputs are external data, such as images and documents. The ultimate outputs accomplish the task, such as recognizing an object in an image. A given neuron can have multiple input and output connections. Between neurons, the weighted sum is passed to neighboring neurons that is connected by these connections (synapses). Each synapse also has a weight that adjusts as learning proceeds. The learning process of ANN is also called training process. Briefly, learning of ANN is a method by which a machine can extract information from data by sending it through different layers of abstraction. With learning process, we are teaching

an AI to recognize the differences between things like cats and dogs, and to find patterns in large amounts of data.

The three major learning paradigms are supervised learning, unsupervised learning and reinforcement learning. For supervised learning, the training of a neural network from a given example is usually conducted by determining the difference between the processed output of the network (often a prediction) and a target output. This difference is the error. The network then adjusts its weighted associations according to a learning rule and using this error value. Successive adjustments will cause the neural network to produce output which is increasingly similar to the target output. After a sufficient number of these adjustments the training can be terminated based upon certain criteria. For unsupervised learning, take image recognition as an example, developers create algorithms that cluster data by similarities. Instead of trying to determine if a group of pixels is cat or a dog, for example, it simply tries to figure out everything it can be. It output patterns in clusters and is possible to separate the images into dogs, cats, brown animals, white animals and so on. For reinforcement learning, the aim is to weight the network (devise a policy) to perform actions that minimize long-term (expected cumulative) cost.

Some hyperparameters needs to be set before the learning process begins. Examples of hyperparameters include learning rate, the number of hidden layers and batch size. The learning rate defines the size of the corrective steps that the model takes to adjust for errors in each training step. A high learning rate shortens the training time, but with lower ultimate accuracy, while a lower learning rate takes longer, but with the potential for greater accuracy. A cost function is a mathematical formula used to evaluate how production expenses will change at different output levels. In other words, it estimates the total cost of production given a specific quantity produced. Backpropagation is a method used to adjust the connection weights to compensate for each error

found during learning. The error amount is effectively divided among the connections. Technically, backprop calculates the gradient (the derivative) of the cost function associated with a given state with respect to the weights. The weight updates can be done via stochastic gradient descent or other methods.

In this section, we take a three-layer neural network as an example to illustrate the machine learning process. An artificial neural network is an interconnected group of nodes, inspired by a simplification of neurons in a brain. Here, each circular node represents an artificial neuron, and an arrow represents a connection from the output of one artificial neuron to the input of another.

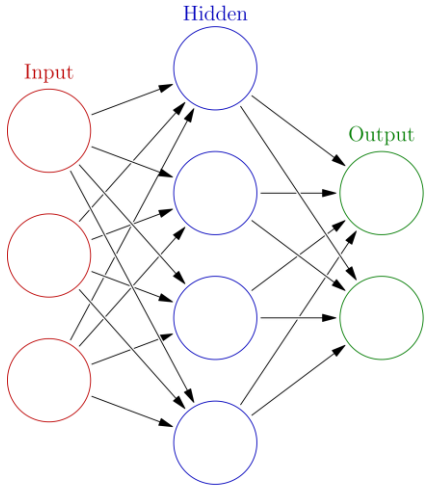


Figure 1. Artificial neural network.

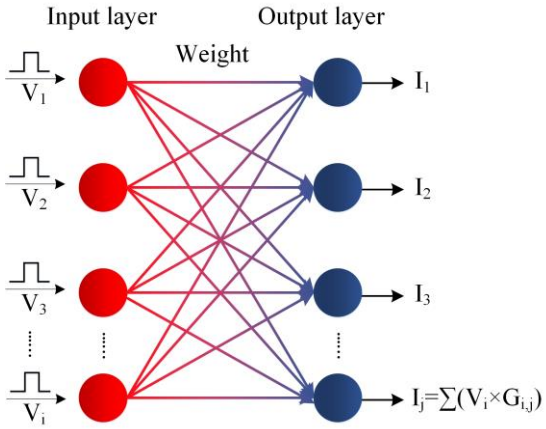


Figure 2. Neural network between two layers. V_i, $G_{i,j}$, and I_j represent the input signal in i^{th} neuron, the weight of the synapses in j^{th} neuron of output layer and i^{th} neuron in input layer, and the output sum that represent the dot product result of V and G, respectively.

3.1.2. SGD Algorithm

Stochastic gradient descent (often abbreviated SGD) is an iterative method for optimizing an objective function. It can be regarded as a stochastic approximation of gradient descent optimization, since it replaces the actual gradient (calculated from the entire data set) by an estimate thereof (calculated from a randomly selected subset of the data). Especially in high-dimensional optimization problems this reduces the computational burden, achieving faster iterations in trade for a lower convergence rate.

$$Q(w) = \frac{1}{n}\sum_{i=1}^{n} Q_i(w) \tag{1}$$

where the parameter w that minimizes $Q(w)$ is to be estimated. Each summand function Q_i is typically associated with the i-th observation in the data set (used for training).

When used to minimize the above function, a standard (or "batch") gradient descent method would perform the following iterations.

$$w = w - \frac{\eta}{n}\sum_{i=1}^{n} \nabla Q_i(w) \tag{2}$$

where η is a step size (sometimes called the learning rate in machine learning).

3.1.3. Long-term Potentiation (LTP) and Long-term Depression (LTD)

Long-term potentiation (LTP) and long-term depression (LTD) are cellular processes involved in learning and memory. In this research, the terms LTD and LTD are used to represent the trend of each synapse weight change. In other words, it represents the trend of each memristor conductance change for each synapse. Concretely, the conductance of the memristor (G) represents the weight of the synapse and it needs to be updated frequently during the data training process as determined by learning algorithms. In such updating process, the conductance can either increase in a process as long-term potentiation (LTP) or decrease in a process as long-term depression (LTD). In other words, the process that the stimulation leads to a persistent increase of the weight is called long-term potentiation of synapses, or LTP for short. The process that the stimulation leads to a persistent decrease of the weight is called long-term depression (LTD), or LTD for short. Ideally, when LTP or LTD occurs, the change in the conductance of an ideal synapse device is proportional to the number of input pulses.

3.2. Memristors

A memristor is an analog device to exploit the multilevel conductance states, to regulate the flow of electrical current, and to store the amount of charge that has previously flowed through it [4, 5]. It can realize desirable device properties such as sub-10 nm feature sizes [7], sub-nanosecond switching speed [8, 9], long write-erase endurance [17], and nanoamperes programming energy [18].

Memristor is name from memory and resistor. The concept of the memristor was first proposed by Chua. Memristor is introduced as the fourth hidden fundamental element, in addition to the resistor, inductor, and capacitor. The theoretical memristor connects the flux φ and charge q, which is shown in the equation: $d\varphi = Md(q)$ [5].

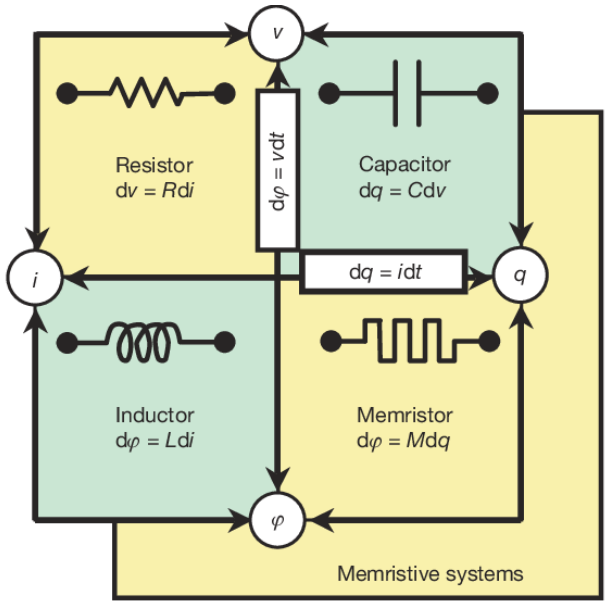


Figure 3. The four fundamental two-terminal circuit elements.

In 2008, Hewlett-Packard researchers presented the prototypical memristor (memory + resistor) devices based on a titanium dioxide insulator layer sandwiched between two metal electrodes (2008). The insulator layer comprises of a stoichiometric titanium oxide layer and a non-stoichiometric titanium sub-oxide layer (TiO_2/TiO_{2-x}). This prototypical device's conductance can be changed by applying a voltage bias across the Pt electrodes. Such ability of encoding the biasing history makes the memristor be a promising storage memory. Different from dynamic random-access memory (DRAM) or static random-access memory (SRAM), memristors can realize non-volatile property, sub-10 nm feature sizes [12], sub-nanosecond switching speed [13, 14], long write-erase endurance [15], and nanoamperes programming energy [16]. However, it is challenged to combine all these properties into a single material system.

Also, different from digital conductance change, the analog conductance tunability of memristor makes in-memory computing be possible. Unlike existing CMOS-based memory technology, which reads volatile capacitance states, memristor technology is a nonvolatile

technology that stores data using nonvolatile resistance states. All these devices exhibit an I-V hysteresis that changes from low resistance states (LRSs) to high resistance states (HRSs) and vice versa. Moreover, the area surrounded by the I-V hysteresis loop is continuously changed by changing the applied electrical signal. In other words, the memristor implements gradual conductance changes that can be utilized as synaptic plasticity.

As a result, it can not only act as the non-volatile memory, but realize the in-memory calculation which brings high potential to implement neural networks in hardware [3]. In-memory or near-memory computing with SRAM, DRAM, or flash memory provides a highly parallel solution to resolve the 'memory wall' of the von-Neumann architecture.

3.2.1. Synaptic Device

Since memristor fabricated by HP in 2008, most memristors are built using simple three-layer structures consisting of two electrodes layer and a switching layer that is generally comprised of dielectrics. The dielectrics used are mainly binary oxide materials such as silicon oxide (SiO2), titanium oxide (TiO_2), copper oxide (CuO), nickel oxide (NiO), zinc oxide (ZnO), hafnium oxide (HfO_2), tantalum oxide (Ta_2O_5) and aluminum oxide (Al_2O_3). Depending on the material and structure of memristors, the resistive switching characteristics can either be abrupt (binary) or gradual (analog). The abrupt resistance change is suitable for storing or processing binary data, and the gradual resistance change is more suitable for storing multiple resistance states for analog computing. There are reports of other switching materials, including heterogeneous materials, such as phase change memory (PCM) that are mainly built out of Chalcogenide materials (prime example is $Ge_2Sb_2Te_5$). The other switching materials are cerium oxide and strontium Titanate (CeO_2 and $SrTiO_3$); organic materials, such as copper-Tetracyanoquinodimethane (Cu-TCNQ); electrolytic materials, such as copper sulfide (Cu_2S) and chalcogenides such as silver–germanium–

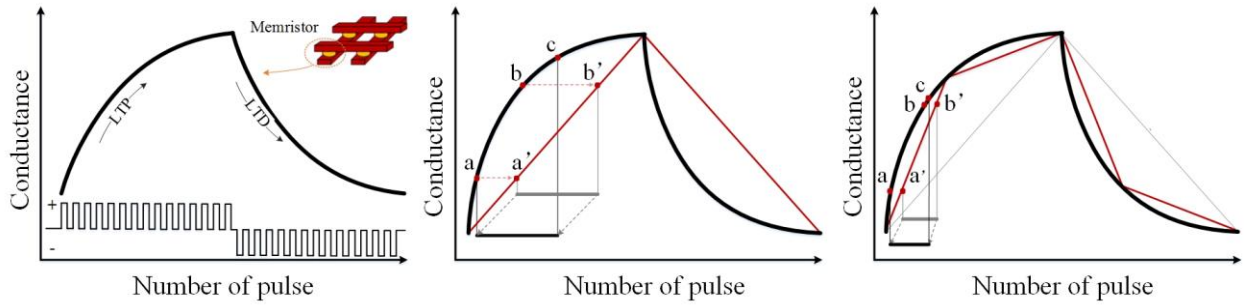
selenium (Ag–Ge–Se). However, because of the unconventional fabrication techniques, CMOS compatibility issues, or relatively unstable physical characteristics, these materials are less commonly used.

In this research, we focus on implementing artificial neural network that generally consists of a large number of synapses. Memristor with gradual resistance change is more suitable for storing multiple resistance states for analog neural computing that implements the function of Sum of Product (SOP). Therefore, in this research, the synaptic devices that we adopted for neuromorphic computing is the memristor with gradual (analog) resistance change. For the rest of this dissertation, all mentioned memristors are refer to a special subset of the resistive memory devices that can continuously tune the conductance into multi-level states.

In a memristor-based neuromorphic computing platform, memristors are connected as neuron devices that implement the function of the Sum of Product (SOP). Weight update depends on the transition between different conductance states in memristors, which is typically triggered by voltage pulses. Specifically, according to the programmed voltage pulses, the conductance of memristors positively and negatively changes, enabling weight increase and decrease, respectively [3]. However, memristors are not flawless and intrinsically suffering from their variability issues. The physical mechanism of the conductance modulation in most prospective synaptic devices is typically an ionic reconfiguration process based on electro/thermo-dynamics [7, 8]. This atomic-level random process is responsible for unavoidable variations including nonlinearity, device-to-device, cycle-to-cycle, and ON/OFF conductance variations [9]. We discuss nonlinear property and cycle-to-cycle variation in the following sub-sections.

3.2.2. Nonlinearity of Memristors

The conductance of the memristor (G) represents the weight of the synapse and it needs to be updated frequently during the data training process as determined by learning algorithms. In such updating process, the conductance can either increase in a process as long-term potentiation (LTP) or decrease in a process as long-term depression (LTD), as shown in Figure 4 (a). Ideally, when LTP or LTD occurs, the change in the conductance of an ideal synapse device is proportional to the number of input pulses. Unfortunately, in reality, such change mismatches the input pulse due to the nonlinearity of memristors. For instance, as demonstrated in Figure 4 (b), the curve (black) represents the conductance of an actual memristor device as a function of the number of input pulses where the pulses have the same duration and the same amplitude. While the line (red) in Figure 4 (b) represents the function of the ideal case. In LTP, as shown in Figure 4 (b), assuming in a weight update process, the device's conductance needs to be updated from point a to b. Usually, the corresponding number of pulses is calculated according to the ideal case (red). But, when these pulses are applied to the actual device, instead of changing from point a to b, the device conductance changes from point a to c. Therefore, the actual change of conductance and the required change are quite different. Similar weight updating error occurs in the LTD process. Consequently, the nonlinearity of the memristor causes the weight change of the synapse device to be inconsistent with the change required by the learning algorithm, thereby reducing the accuracy of ANN's recognition.



(a) (b) (c)

Figure 4. The conductance change (weight updating) curve. (a) Conductance changes with identical input pulses. (b) Weight updating process based on linear line. (c) Weight updating process based on a piecewise line.

We adopt a general conductance change behavior model [9] that is defined by the following equations:

$$G_{LTP} = B\left(1 - e^{\left(-\frac{P}{A}\right)}\right) + G_{min} \tag{3}$$

$$G_{LTD} = -B\left(1 - e^{\left(-\frac{P-P_{max}}{A}\right)}\right) + G_{max} \tag{4}$$

$$B = \frac{G_{max} - G_{min}}{1 - e^{\left(\frac{-P_{max}}{A}\right)}} \tag{5}$$

where G_{max}, G_{min}, and P_{max} are directly extracted from the actual test data [9], which represents the maximum conductance, minimum conductance, and the maximum pulse number required to switch the device between the minimum and maximum conductance states. A and B are the parameter that controls the nonlinear behavior of the weight update. In this model, by adjusting A, the conductance curve is labeled with a nonlinearity value (NL) from +6 to -6, which represents the extent to the curve deviates from the ideal linear device and is illustrated in Figure 5. Here the positive (+) and negative (-) signs are merely to label LTP and LTD, respectively.

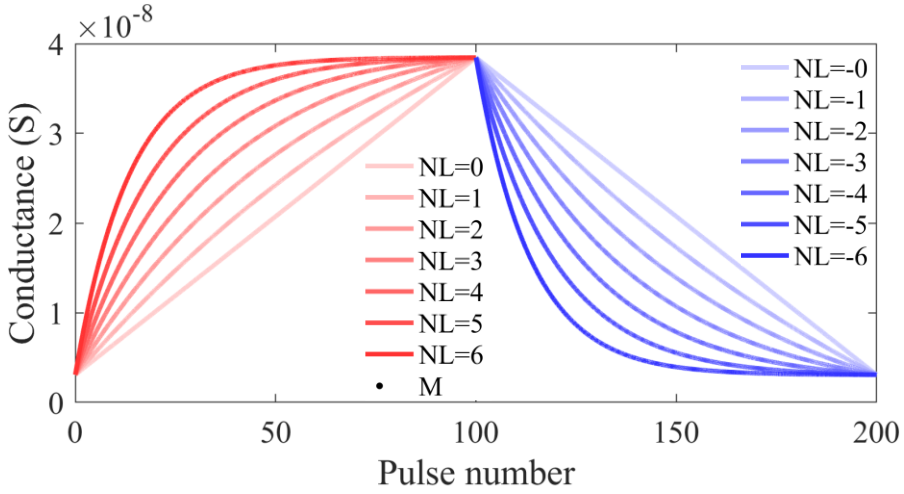


Figure 5. Conductance change curves under various nonlinearity of LTP and LTD.

3.2.3. Cycle-to-cycle Variation of Memristors

Memristors can achieve multiple conductance states. In our learning system, the conductance of each memristor represents the weight of each synapse. As shown in Figure 6, positive and negative input voltage pulses that are larger than the threshold voltage can switch a memristor gradually from Gmin to Gmax or from Gmax to Gmin, where Gmin and Gmax represent minimum conductance and maximum conductance, respectively. Thus, the conductance/weight increase process is called long-term potentiation (LTP) and the conductance/weight decrease process is called long-term depression (LTD) [48].

In the backpropagation phase of the DP-SGD algorithm, the weight update values (Δw) will be translated to a number of LTP or LTD pulses and applied to the synaptic array. The amount of conductance change should be linearly proportional to the number of write pulses; however, this linear change is broken by memristor variations. Among all variations of memristors not attributed to manufacturing process, cycle-to-cycle variation is caused by intrinsically stochastic resistance switching mechanisms [49-52] that can be approximated as a Gaussian or normal distribution [45, 46, 53-57]. It originates from the random formation and disruption of conducting

filaments [51] and the co-existence of multiple sub-filaments, where the active, current-carrying filament may change from cycle to cycle [52, 58]. To prove such randomness introduced by each programming operation, 500 cycles [59] and 5000 cycles [56] of experimental data are collected. Cycle-to-cycle variations result in the different updated conductance when the same updating signal in different updating cycles is applied to a memristor, even when the initial conductance is the same.

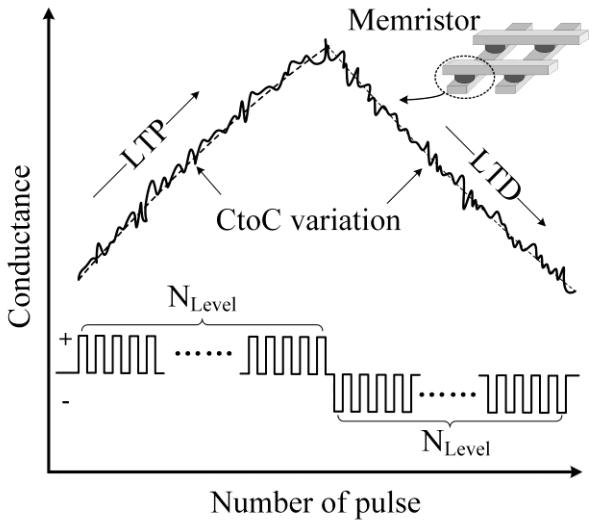


Figure 6. Long-term potentiation process (LTP), long-term decrease process (LTD), and cycle-to-cycle variation of a memristor. N_{Level} represents the number of conductance states of a memristor.

3.2.4. Artificial Neural Networks (ANN) and Memristor

ANN are computing systems vaguely inspired by the biological neural networks that transform inputs to desired outputs by feed-forward networks. As shown in Figure 7, each neuron in the network takes a weighted sum of the outputs of the prior layer, and then transfer the sum to the next layer.

In the hardware implementation, the neural network can be directly mapped into a crossbar from where the inputs are connected into the rows and the outputs are connected into the columns.

Memristor-based crossbar circuit can store the synaptic weight and calculate the desired result (Sum of Product), at the same time, extremely improving the system efficiency. The desirable properties of memristors support the memristor-based crossbar circuit to be a promising substitute technology to traditional ones so that researchers begin realizing device-engineering and array-integration hardware implementation of memristors. Usually, as shown in Figure 7, in the hardware application of the neural network, memristors act as synapses in crossbar structure and locate in each cross point.

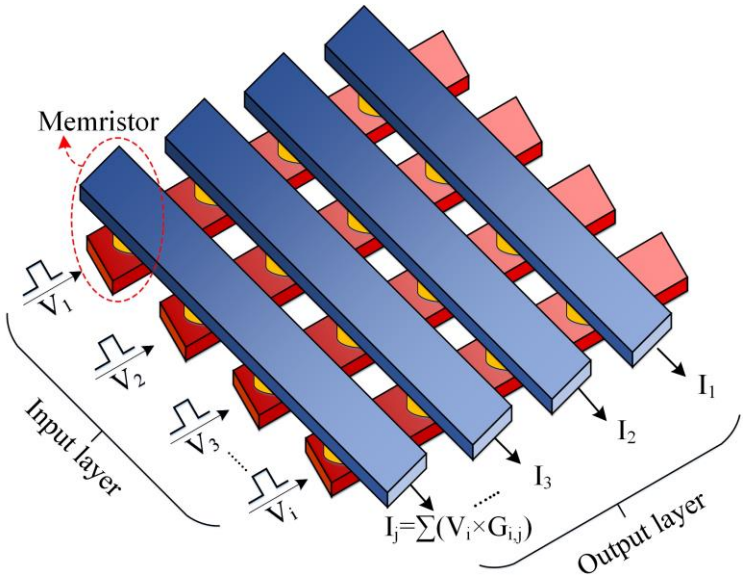


Figure 7. Hardware implementation of neural networks using memristor crossbar. V_i, $G_{i,j}$, and I_j represent the input signal in i^{th} row, the conductance of the memristor in j^{th} column and i^{th} row, and the output current that represent the dot product result of V and G, respectively.

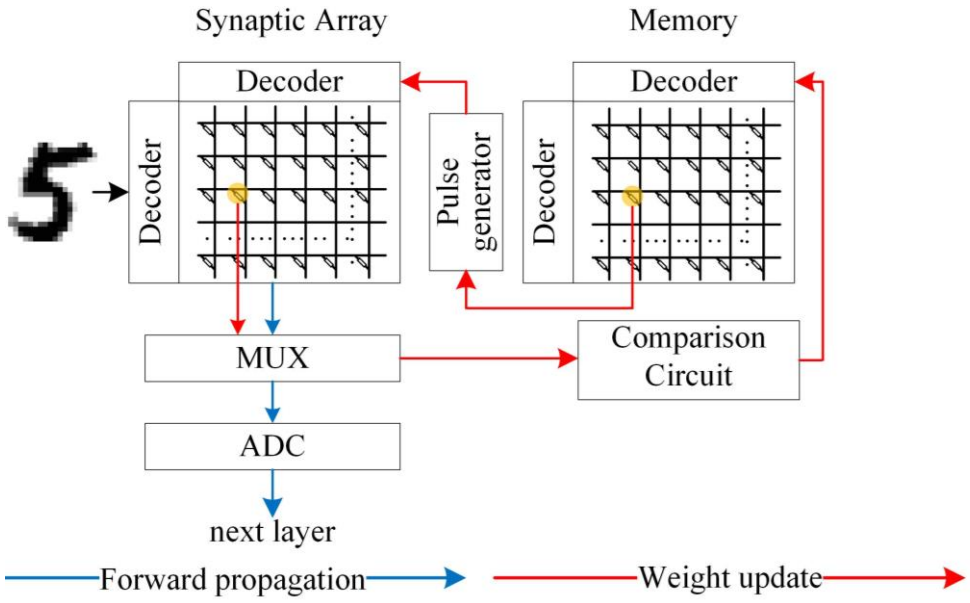


Figure 8. Block diagram of circuit flow of a three-layer ANN.

3.3. Differential Privacy

When deep learning algorithms work on the edge for human-related applications, IoT devices will collect data, which in some cases may contain quite personal and high-value user information, therefore raising significant concerns regarding information privacy. It is thus possible for the private data to be misused, or to be hacked by outside attacks. Therefore, some sort of privacy protection method is required to guarantee a strong notion of privacy, while preserving learning accuracy. A traditional privacy protection method is anonymization or de-identification, which removes attributes in the data and returns a sampled data to protect privacy. However, it is still possible to discern information about the datasets, which is known as linkage attacks, statistical inference attacks, generative adversarial networks (GANs) based attack, and re-identification attacks [60, 61]. For example, if we use deep learning for cancer diagnosis, when we release a learning model, we may unintentionally disseminate information about the training dataset so that a malicious attacker could identify individuals diagnosed with cancer. Another

privacy protection method is via cryptography such as homomorphic encryption, but it requires large computational overhead [62]. Currently, differential privacy [10] is one of the most popular technologies for privacy-preserving deep learning, realized by introducing perturbations. Differential privacy (DP) provides a mathematical constraint for the privacy loss associated with any data release from a statistical database. In edge computing, federated learning is a recent advance in private learning, where the model is trained in a decentralized manner without sharing the raw data. It prevents a third party from storing personal data as well as performing learning tasks on that data. Despite such privacy improvements, local differential privacy is necessary for federated learning because the weight updates uploaded by individual edge devices may still reveal private information. Adding noise to the weights/gradients during training on local data prior to aggregation by an untrusted server provides greater privacy protection to users [63]. Differential Privacy (DP) is a system that publicly shares data set information by describing the group pattern in the data set while concealing personal information in the data set. The idea behind differential privacy is that if the impact of any single replacement in the database is small enough, the query results cannot be used to infer the information of any single individual, thus providing privacy. Another way to describe differential privacy is as a constraint on the algorithm used to publish summary information in the statistical database, which limits the disclosure of private information recorded in the database. For example, some government agencies use differentiated private algorithms to publish demographic information or other statistical summaries, while ensuring the confidentiality of survey responses, and companies use differentiated private algorithms to collect information about user behavior, while controlling and even controlling internal analysts. Roughly speaking, if the observer who sees its output cannot judge whether a particular individual's information is used in the calculation, then the algorithm is differentially private. Differential

privacy is usually discussed in the context of identifying individuals whose information may be in the database. In other words, the basic idea of this model is to add specific noise so that inserting or deleting a record in a dataset does not statistically affect any calculated output. DP provides provable guarantees of privacy, mitigating the risk of exposing sensitive training data in machine learning [64]. Differential privacy promises a powerful standard for privacy guarantees either on algorithms or databases [39, 44, 64]. DP protection technology is recognized as a rigorous and robust protection model.

Firstly, the definition of ε-differential privacy and equation are given below.

Definition [10]: A randomized mechanism A satisfies ε-differential privacy when any adjacent input d and d', and any output S of A is defined by,

$$\Pr[A(d) = S] \leq e^{\varepsilon} \times \Pr[A(d') = S] \tag{6}$$

where d and d' are adjacent inputs that differ in a single entry. In our work, for instance, each training dataset is a set of image-label pairs. The d and d' are two sets that only one image-label pair is present in one set and absent in the other. The parameter ε is the privacy budget, which evaluates the privacy guarantee of the randomized mechanism A. A smaller value of ε means the closer recognition accuracies can be got from adjacent inputs and indicates a stronger privacy guarantee. By this definition, privacy preservation can be calculated and evaluated through ε.

Typically, adding noise calibrated to the global sensitivity is a general method for approximating a function f, denoted as Δf, which is the maximal value of $||f(d) - f(d')||$ among any input pair of d and d' [10, 64]. For instance, the Laplacian mechanism is defined by,

$$Af(d) = f(d) + Lap(\Delta f/\varepsilon) \tag{7}$$

where $Lap(\Delta f/\varepsilon)$ is a random variable sampled from the Laplace distribution with scale Δf/ε.

Secondly, the definition of (ε, δ)-differential privacy is given below [9, 10]. A randomized mechanism, A, satisfies (ε, δ)-differential privacy when any adjacent input datasets, d and d', and any output, S, of A satisfy the following equation.

$$\Pr[A(d) = S] \leq e^{\varepsilon} \times \Pr[A(d') = S] + \delta \quad (8)$$

In our study, each training dataset is a set of image-label pairs. Given a negligibly small probability, δ, parameter ε is the privacy budget, which measures the privacy bound of the randomized mechanism, A, for adjacent datasets. A smaller value of ε means higher indistinguishability, thus a stronger privacy guarantee. By this definition, privacy preservation can be calculated and evaluated through ε, given δ.

3.4. Differentially Private SGD Algorithm (DP-SGD)

DP-SGD is a modification of the Stochastic Gradient De-scent (SGD) algorithm that is popular and serves the basis for many optimizers in machine learning [44]. Models trained with DP-SGD have provable privacy guarantees in terms of DP. Instead of working only on final parameters from the training process, DP-SGD controls the influence of training data during the training process. Figure 8 outlines the principles for training a model with weight parameters, θ, by minimizing the loss function.

At each step of the DP-SGD [44], it computes the gradient for a random subset of examples, clips each gradient, computes the average, adds noise in order to protect privacy, and takes a step in the opposite direction of this average noisy gradient. Two operations are needed to ensure that stochastic gradient de-scent is a differentially private algorithm. The first is to clip the gradient computed on each training image to limit how much each training image can impact model parameters. The algorithm clips each gradient by a clipping threshold, C. In this paper, we use L2 norm of gt(xi) to represent $||gt(xi)||_2$, which is explained in Figure 8. The second is to sample and

add random noise to randomize the algorithm. Thus, it is statistically im-possible to identify whether a particular sample is included in the training set.

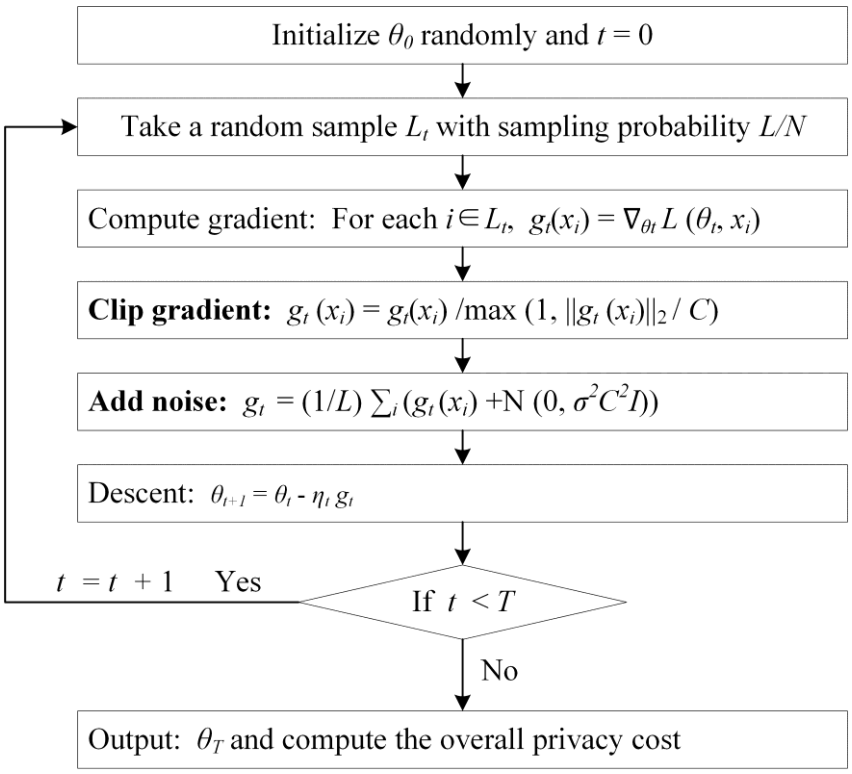


Figure 9. Outline of DP-SGD. Symbols and parameters: input dataset, weights θ, loss function $\mathcal{L}(\theta)$, gradient g, learning rate η_t, noise scale σ, group size L, gradient norm bound C, total weight update step T, the square root of the largest eigenvalue of the matrix $g_t(x_i)^* g_t(x_i)$, $\|g_t(x_i)\|_2$, where $g_t(x_i)^*$ denotes the conjugate transpose of $\|g_t(x_i)\|_2$.

In this paper, we achieve the algorithmic essence of DP-SGD using a memristor-based neural network and our proposed methods. We realize a privacy-preserving memristor-based learning system without introducing extra computational processing or noise generation units.

4. MITIGATING NONLINEAR EFFECT OF MEMRISTIVE SYNAPTIC DEVICE FOR NEUROMORPHIC COMPUTING

Memristors offer advantages as a hardware solution for neuromorphic computing, however, their nonlinear device property makes the weight update inaccurately and reduces the recognition accuracy of a neural network. In this paper, a piecewise linear (PL) method is proposed to mitigate the nonlinear effect of memristors by calculating the weight update parameters along a piecewise line, which reduces errors in the weight update process. It mitigates the nonlinearity impact without reading the precise conductance of the memristor in each updating step, thereby avoiding complex peripheral circuits. The effectiveness of the proposed PL method with respectively 2-segment, 3-segment, and 4-segment models in two split selection strategies is investigated and the impact of various variations are considered. The results show that under different nonlinearity, the PL method can increase the recognition accuracy of MNIST handwriting digits to 87.87%-95.05%, as compared to 10.77%-73.18% of the cases without the PL method.

4.1. Introduction

Device scaling on the CMOS technology that effectively benefits the cost, speed, and power efficiency of Integrated Circuits (ICs), has driven the boom of the computing systems over the past several decades. Unfortunately, the increasing physical restrictions such as thermodynamic limits including quantum tunneling [1] make COMS device scaling reach a plateau, which suppresses the further progress of computing systems in large data centers or the Cloud, and also obstruct the fast deployment of sensors and actuators for the Internet of Things (IoT) [2]. Meanwhile, neuromorphic computing using bio-inspired learning algorithms has emerged and achieved tremendous success. However, due to the separated computation unit and memories [3], conventional von Neumann-based computing platform results in insufficient

bandwidth and overlarge power consumption, particularly for the fast training and/or classification tasks in neuromorphic computing, such as real-time online recognition. Therefore, new synapse devices and technologies with remarkable performance and less cost are urgently needed. Memristors are one of such promising devices. A memristor is an analog device to exploit the multilevel conductance states, to regulate the flow of electrical current, and to store the amount of charge that has previously flowed through it [4, 5]. As a result, it can not only act as the non-volatile memory, but realize the in-memory calculation which brings high potential to implement neural networks in hardware [3].

In a memristor-based neuromorphic computing platform, memristors are connected as neuron devices that implement the function of the Sum of Product (SOP). Weight update depends on the transition between different conductance states in memristors, which is typically triggered by voltage pulses. Specifically, according to the programmed voltage pulses, the conductance of memristors positively and negatively changes, enabling weight increase and decrease, respectively [9]. However, memristors are not flawless and intrinsically suffering from their variability issues. The physical mechanism of the conductance modulation in most prospective synaptic devices is typically an ionic reconfiguration process based on electro/thermo-dynamics [7, 8]. This atomic-level random process is responsible for unavoidable variations including nonlinearity, device-to-device, cycle-to-cycle, and ON/OFF conductance variations [3]. Especially, the nonlinearity makes it challenging to determine a proper width or amplitude of input signals for achieving the desired conductance of memristors. It is reported that the linear conductance change is the major requirement of a memristor-based neuromorphic computing to realize high accuracy for the online learning [9]. However, four state-of-the-art memristors in literature – Ag:a-Si [17], TaO_x/TiO_2 [18], PCMO [19], and AlO_x/HfO_2 [20] are all characterized by device nonlinearity. For example,

the accuracy of recognition based on Ag:a-Si with nonlinearity decreases over 20% than that without nonlinearity [3, 17].

Therefore, a new technique is proposed to use pulses programming method to update the weight following the nonlinear curve of memristors, thereby enhancing the accuracy of the learning algorithms.

4.2. Methodology

To mitigate the impact of nonlinear property of memristors on neuromorphic computing, a Piecewise Linear (PL) method is proposed in this paper. As shown in Figure 3 (a), instead of calculating the number of required pulses along the ideal curve (green), the PL method performs the calculation along a polyline (pink), which fits the actual device property better than the ideal curve (green). Thus, the error that incurs from the nonlinearity of memristors is reduced. For example, Figure 9 (a) and Figure 9 (c) present the same conductance change curve without and with the PL method. After the same number of pulses are applied to points a and b, the conductance difference between the actual points d and c in Figure 9 (c) is reduced than that in Figure 9 (a).

In Figure 3 (a), the polyline is composed of two lines, and we call it a 2-segment PL model. Similarly, based on the number of lines in the polyline, this method can be applied in the 3-segment (Figure 9 (d)) and a 4-segment PL model. From a qualitative point of view, the 3-segment and 4-segment PL models result in further error reduction as compared to the 2-segment model.

Next, we analyze the implementation process of the PL method. To make the change of conductance along with a piecewise line, instead of using the pulses with the same duration, the PL method uses different durations of pulses. First, we select split points to generate a polyline, then assume that the slope of the ideal conductance curve is k_0 and the polyline slopes are $\{k_1, k_2, \ldots, k_n\}$. Then, we apply n types of pulses that are defined by the original pulse and these slopes,

which have the same amplitude. When the conductance change is required, one of the n types of pulses is selected. Specifically, the PL method scales the duration of pulses to k0/ki times of the original duration to compensate the conductance change caused by the nonlinear effect. In this way, the larger/smaller the line's slope is, the shorter/longer the duration is. However, if a large weight update that crosses the turning point, the PL method will cause deviations using a single pulse type. We will discuss this case in section V. B. To implement the PL method, a set of memory is also needed to store the segment information, which is used to select a slope before each weight update. The configurations of the PL method and its detailed working flow are presented in the section IV based on digits pattern recognition task.

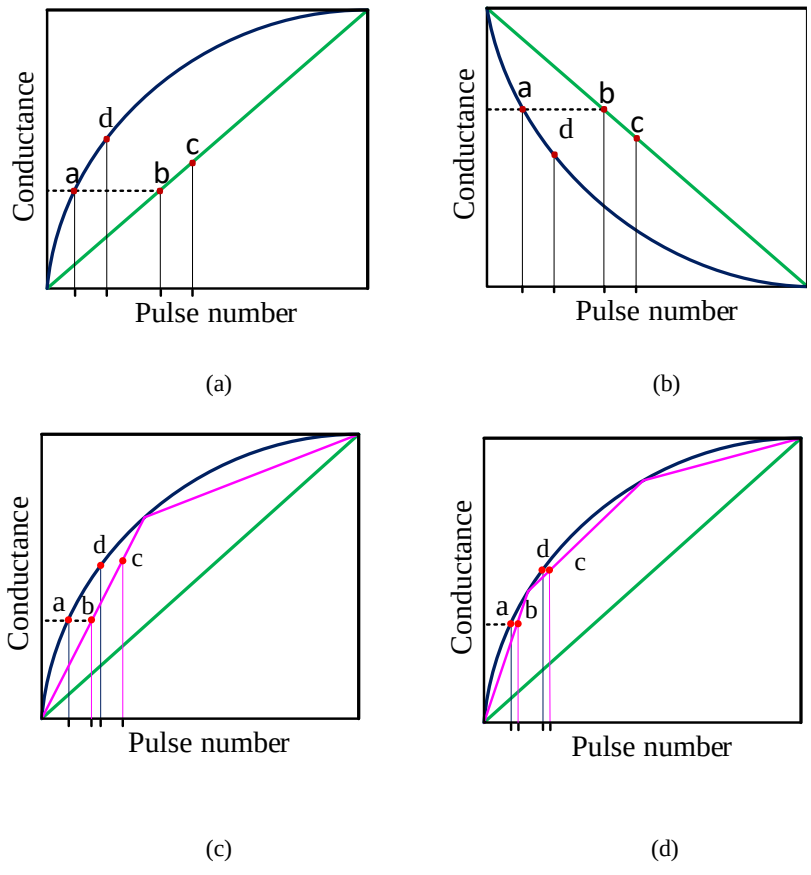


Figure 10. Conductance change diagram. (a) LTP, (b) LTD, (c)2-segment model, (d) 3-segment model.

4.3. Application on Digits Image Recognition

To verify the effectiveness of the proposed PL method, we adopt the neural network hardware platform, NeuroSim+ [9, 48], to perform handwriting recognition using the Stochastic Gradient Descent (SGD) machine learning algorithm based on the MNIST database. The neural network of this simulator includes three layers with 400 neurons, 100 neurons (hidden layers), and 10 neurons, respectively. Each simulation trains up to 125 epochs. Each epoch selects 8,000 images randomly from 60,000 training images and takes 10,000 images as a testing dataset. The result data that shown in this paper is an average result of 10 simulations which is defined as a group.

As discussed in Section III, the proposed PL method needs to select the split points to generate a polyline. Also, the PL method can select to be used in only one process (LTP or LTD) or both process (LTP & LTD). These configurations determine the total segment number, the required types of pulses, and the size of memory. The more segments, the less weight deviation that is caused by the nonlinearity but with higher circuit cost. In order to investigate the tradeoff among these configurations, we conduct two split selection strategies and three models including 2-segment, 3-segment, and 4-segment models, and apply the PL method in LTD or LTP as well as LTD & LTP. In addition, further simulations are conducted to verify the effectiveness of the PL method under memristor's variations, based on 4-segment PL model.

4.3.1. Split Selection Strategies and 2/3/4-segment Models

Split selection strategies include middle strategy and slope strategy. With middle strategy, the split points are selected to divide the conductance range into *n* equal segments. With slope strategy, the split points are selected to minimize the average difference between the actual curve and the polyline. The detailed illustration and points choosing method are shown in the Appendix.

When the PL method needs $log_2(n)$-bit to store the conductance range information apart from the slope strategy in LTD & LTP that needs $2log_2(n)$-bit. This difference is because when using the PL method in LTD & LTP with slope strategy, the conductance at split points in LTP usually differs from the points in LTD. But they are equal when using middle strategy, thus, the memory can be shared between LTP and LTD. Additionally, the 2-segment, 3-segment, and 4-segment models are developed to verify the PL method with different split points number, which are also illustrated in the Appendix.

4.3.2. Working Flow of the PL Method

Based on MNIST handwritten database and SGD algorithm, the hardware working flow of the PL method with different configurations is shown in Figures 10, 11, and 12. First, the PL method provides various configurations that are needed to be determined before the hardware implementation. We choose segment model and split strategy of the PL method based on nonlinearity property of memristor devices that are used as synapses device. Accordingly, circuit parameters including pulses types and memory space are determined. The detailed configuration steps are 1) We investigate the NL value of the curve that represents conductance changes of the memristor to be applied with pulses; 2) A group of split points $\{M_1, M_2, \ldots, M_{n-1}\}$ in this updating curve are selected to identify n segments and n lines. The slope values of these lines $\{k_1, k_2, \ldots, k_n\}$ are obtained based on Equation 9 and the pulses types are determined based on these slopes; 3) A $log_2(n)$-bit memory is needed to store the segment information of each device.

$$k_i = (G_i - G_{i-1})/(P_i - P_{i-1}) \tag{9}$$

where $0 < i < n$; k_i, represent the slope of the line end with the points M_i and M_{i-1}; G_i and P_i represent the conductance and the pulse number corresponding to M_i, respectively. Here, M_i represents the

split point in the set of $\{M_1, M_2, \ldots, M_{n-1}\}$, which is shown in the first step and it can also be the initial end of the updating curve M_0 and the last end M_n.

Second, at the beginning of the training process, all weights are initialized and these weights segment information are stored in memory, which are used to choose input pulses' type for the first-time weight update. Here, the range of conductance is divided by the split points. Third, we follow SGD algorithm with data input, forward propagation, and backward propagation, then get the number of pulses to be applied for each memristor for weight updating. Next, by reading the stored segment information, the pulses duration is selected among the type-fixed pulses. Then, for each memristor after applying selected pulses, a comparison circuit is used to recognize the current segment information, thereby updating $log_2(n)$-bit memory. The comparison circuit works right after the weight update for each memristor to make sure current segment information can be gotten no matter how large the conductance needs to be updated or how much the variation is during the update. Fourth, the third step is repeated until the training operation is completed.

Given that the PL method offers multiple application strategies, which is needed to further investigate, in the next section the PL method is applied in 18 configurations for 10-digit recognition task based on a hardware simulator.

Figs. 10, 11, and 12 show the curves for NL from 0 to 6 of LTP and with NL from 0 to -6 of LTD for 2-segment, 3-segment, and 4-segment models, respectively. For memristors with the same NL, the difference between the curve and the polyline decreases as the segment number increasing. The middle strategy is the split points are selected with the values that can divide the conductance range into n equal segments. Points M in Figs. 10 (a) (b), 11 (a) (b), and 12 (a) (b)

represent split points with middle strategy for 2-segment, 3-segment, and 4-segment models and they are obtained by the Equation 10.

$$G_{Mi} = (G_{max} - G_{min}) \times (i/n) + G_{min} \quad (10)$$

where G_{Mi}, G_{max}, G_{min}, and n represent the conductance of split point *Mi* (*i=1, 2, …, n*), the maximum conductance, the minimum conductance, and the number of the segment. The slope strategy is the split points are selected to minimize the average difference between the actual curve and the polyline. Points M in Figs. 10 (c) (d), 11 (c) (d), and 12 (c) (d) represent split points with slope strategy for 2-segment, 3-segment, and 4-segment models and they are obtained by the algorithm below.

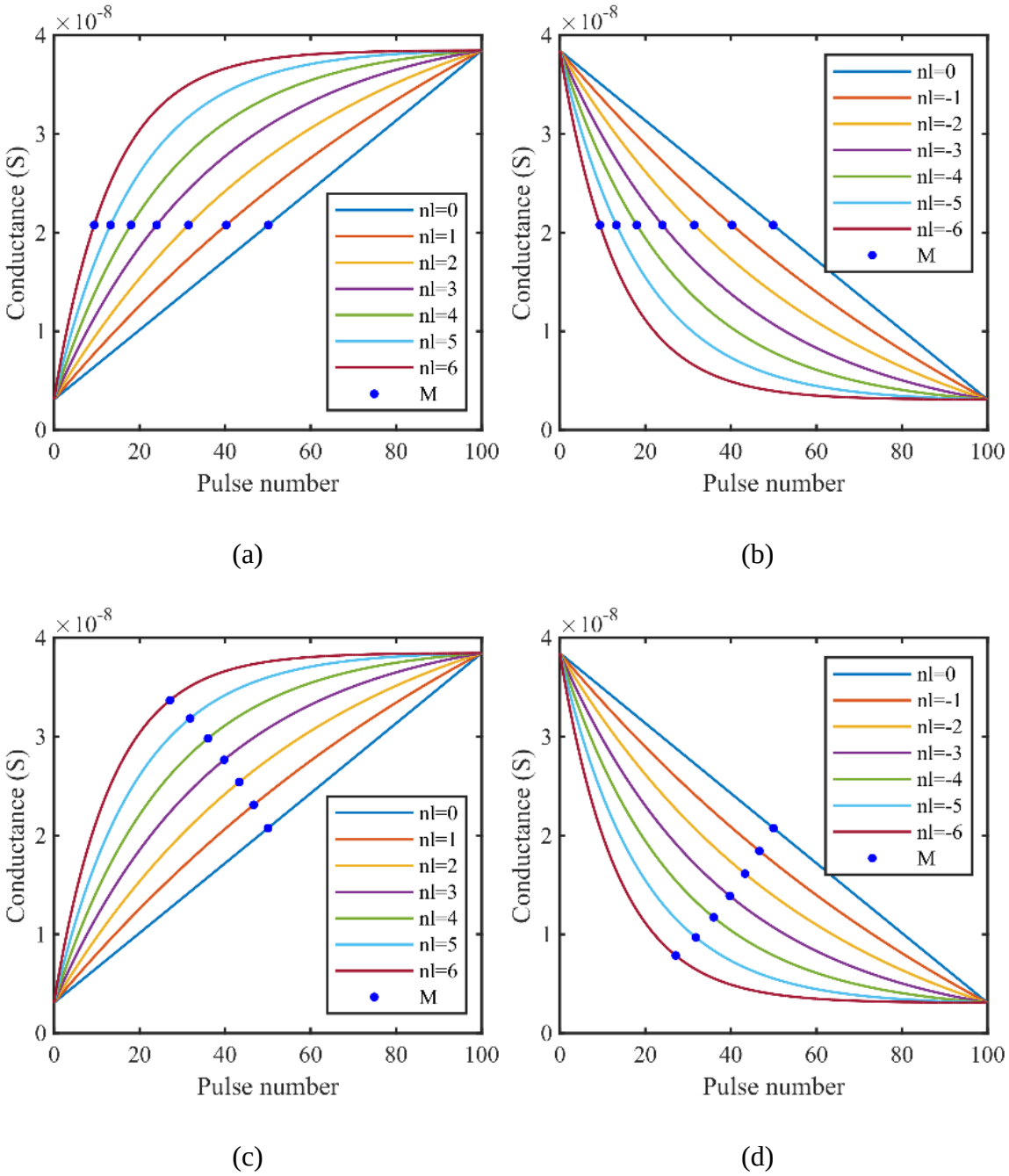


Figure 11. 2-segment model. (a) LTP and middle split-point, (b) LTD and middle split-point, (c) LTP and slope split-point, (d) LTD and slope split-point.

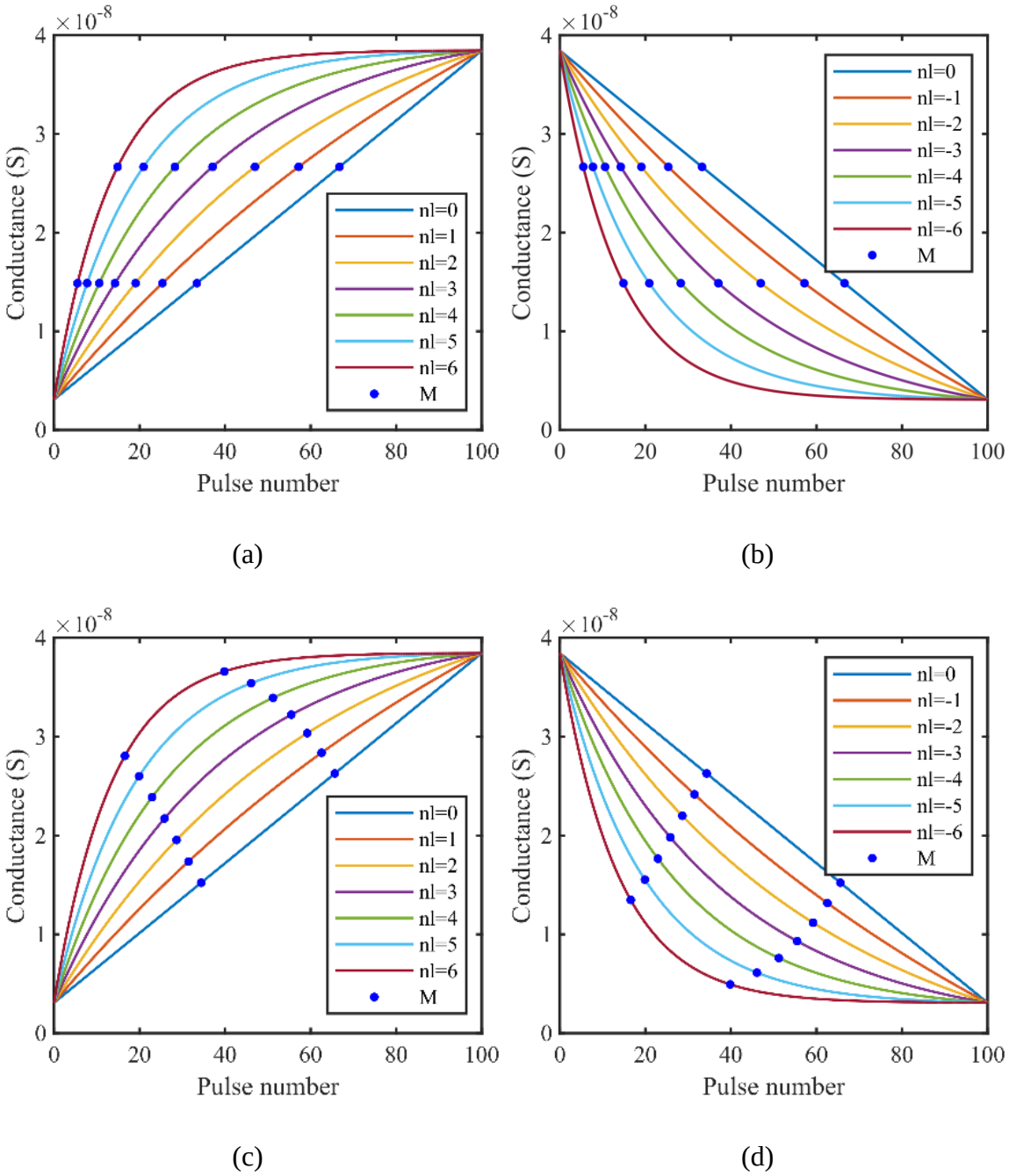


Figure 12. 3-segment model. (a) LTP and middle split-point, (b) LTD and middle split-point, (c) LTP and slope split-point, (d) LTD and slope split-point.

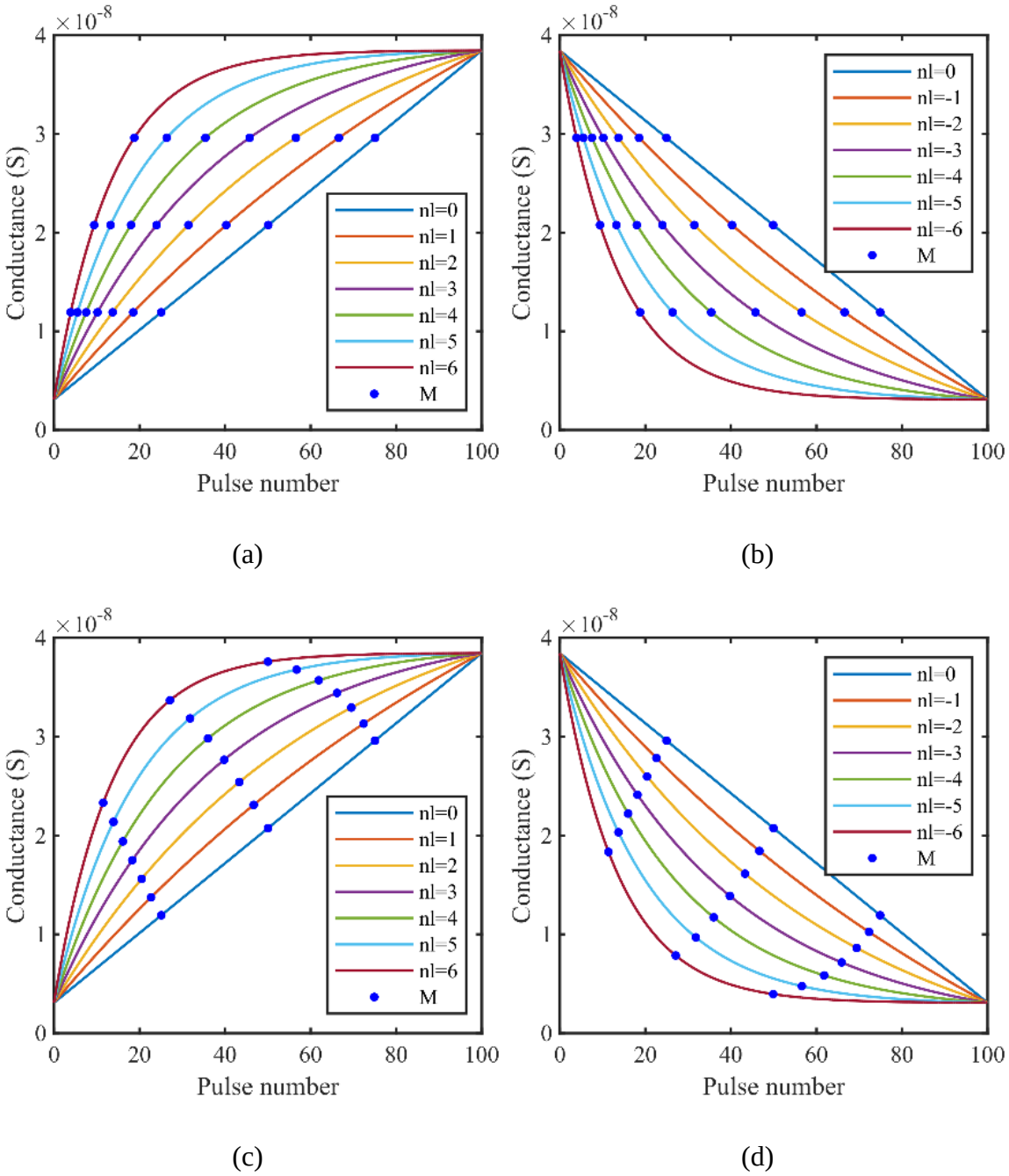


Figure 13. 4-segment model. (a) LTP and middle split-point, (b) LTD and middle split-point, (c) LTP and slope split-point, (d) LTD and slope split-point.

Algorithm: Split Points M with Slope Strategy

Input: memristor conductance change curve function $f_1(x)$, ideal linear conductance change function $f_2(x)$, points number n, cycle index k.

1: **for** i=1, 2, …, k-1 **do**
2: $F(x)=f_1(x)-f_2(x)$
3: $G_{diff\_max}=\max(F(x))$
4: $X_{Mi}=F^{-1}(G_{diff\_max})$
5: M_i (X_{Mi}, $f_1(X_{Mi})$)
6: Update $f_2(x)$ to a function of polyline that ends with M_{min}, M_1, …, M_i, and M_{max}.
7: **end for**
8: select n-1 points from M_1, …, M_{k-1}
9: return M_1, …, M_{n-1}

Figure 14. Algorithm of Split Points M with Slope Strategy.

4.4. Results and Discussion

A comprehensive suite of digit recognition simulations has been conducted to explore the proposed PL method that aims to mitigate the nonlinearity of memristors in a hardware implementation for a neural network. We perform 49 groups simulations to explore 49 different memristor devices whose NL in LTP is from 0 to 6 and NL in LTD is from 0 to -6. In each group, 10 simulations with the same setting are conducted to get a representative average result. As the original case without the PL method, Figure 15 shows the recognition accuracy based on 49 memristors with different NL. The accuracy is represented by a colored square. Figure 16 shows 18 cases that are obtained from different combinations of different segment models and split selection strategies. To compare the overall effectiveness of the PL method among the original case and other 18 cases, the overall accuracy results are calculated by averaging the accuracy using different memristors under 49 NL combinations. Also, the minimum result and the maximum result (expect case without nonlinearity: LTP=0, LTD=0) are shown under each figure in Figures. 15 and 16. It can be concluded that with the PL method, the recognition accuracy of the learning

algorithm can be improved to 87.87%-95.05% for 49 NL cases as shown in Figure 16, as compared to 10.77%-73.18% for that without the PL method as shown in Figure 15.

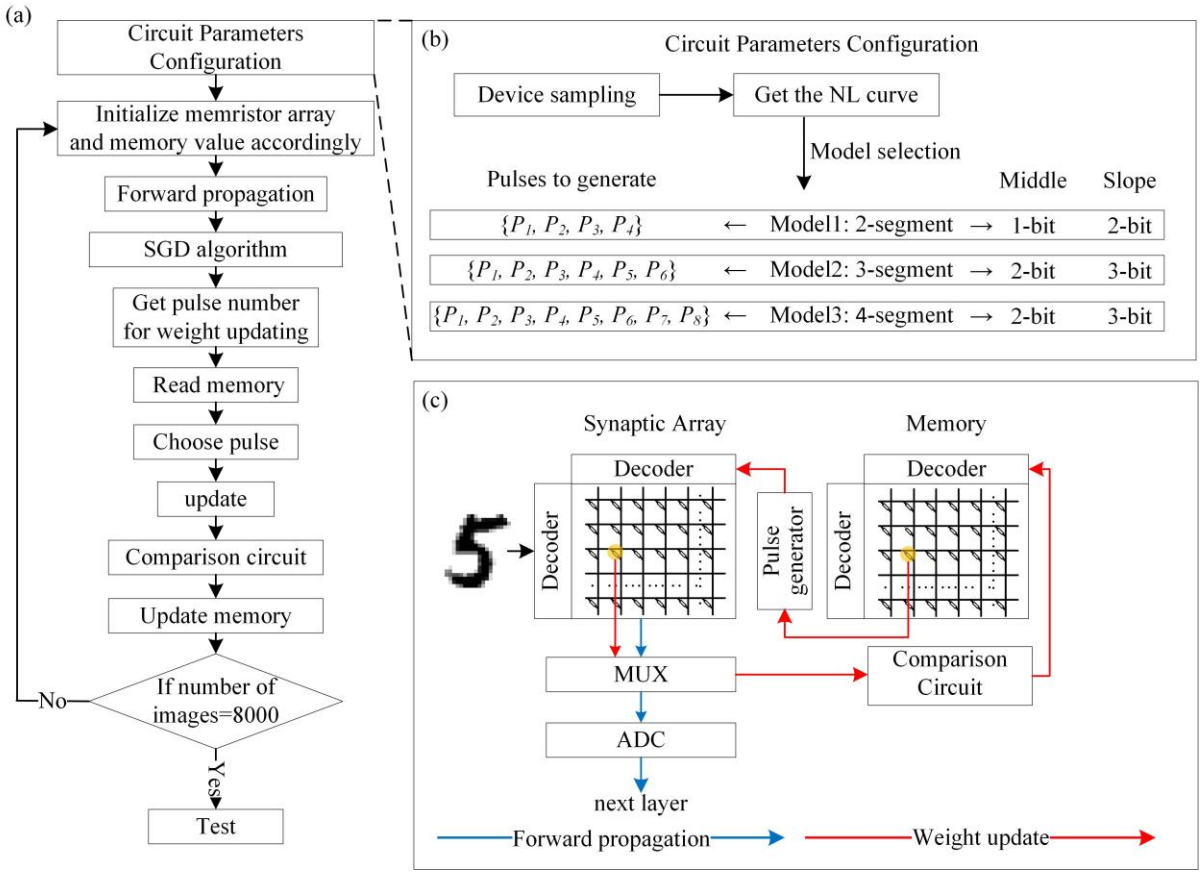


Figure 15. (a) Flow chart of hardware-based neural network, (b) Circuit parameters configuration, (c) Block diagram of circuit flow.

4.4.1. Working Flow of the PL Method

As also shown in Figure 16, among 18 cases, the case with configuration of 4-segment, middle strategy, and both applied process (LTD & LTP) can achieve overall accuracy 91.54% and minimum accuracy 87.87% among 49 memristors with various NL. It indicates that utilizing the proposed PL method, even memristors with high nonlinearity (e.g. NL: (6, -6)) can be used as a synapse for an accepted recognition accuracy in neuromorphic hardware. As compared with the original case in Figure 15 whose overall accuracy is 25.16%, the PL method enables at least 20% higher overall accuracy as shown in Figure 16. In particular, for 2-segment, 3-segment, and 4-

segment models with the PL method applied in both processes (LTD & LTP), the overall accuracies are increased by 59.5%, 64.5%, and 66.4%, respectively. Additionally, it can be observed in Figure 16, under the same circumstances, the case with higher segments shows higher overall accuracy. For example, in LTP & LTD with slope strategy, the overall accuracy of the 4-segment model is 91.01%, which is higher than 84.63% of the 2-segment model and 89.67% of the 3-segment model. It indicates that a polyline with more segments fits the actual curve better than that with fewer segments, resulting in a higher overall accuracy.

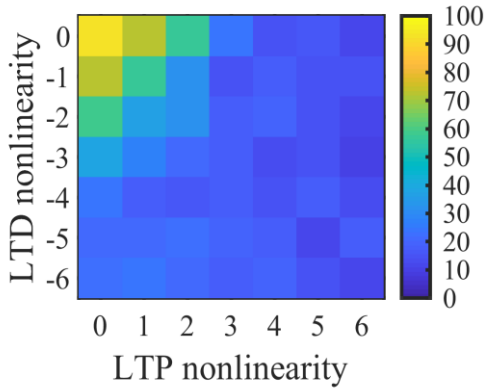


Figure 16. Original recognition accuracy for memristors with nonlinearity property. Minimum accuracy is 10.77%. Overall (Average) accuracy is 25.16%. Maximum accuracy (with NL) is 73.18%. Without nonlinearity accuracy (NL (0,0)) is 95.55%.

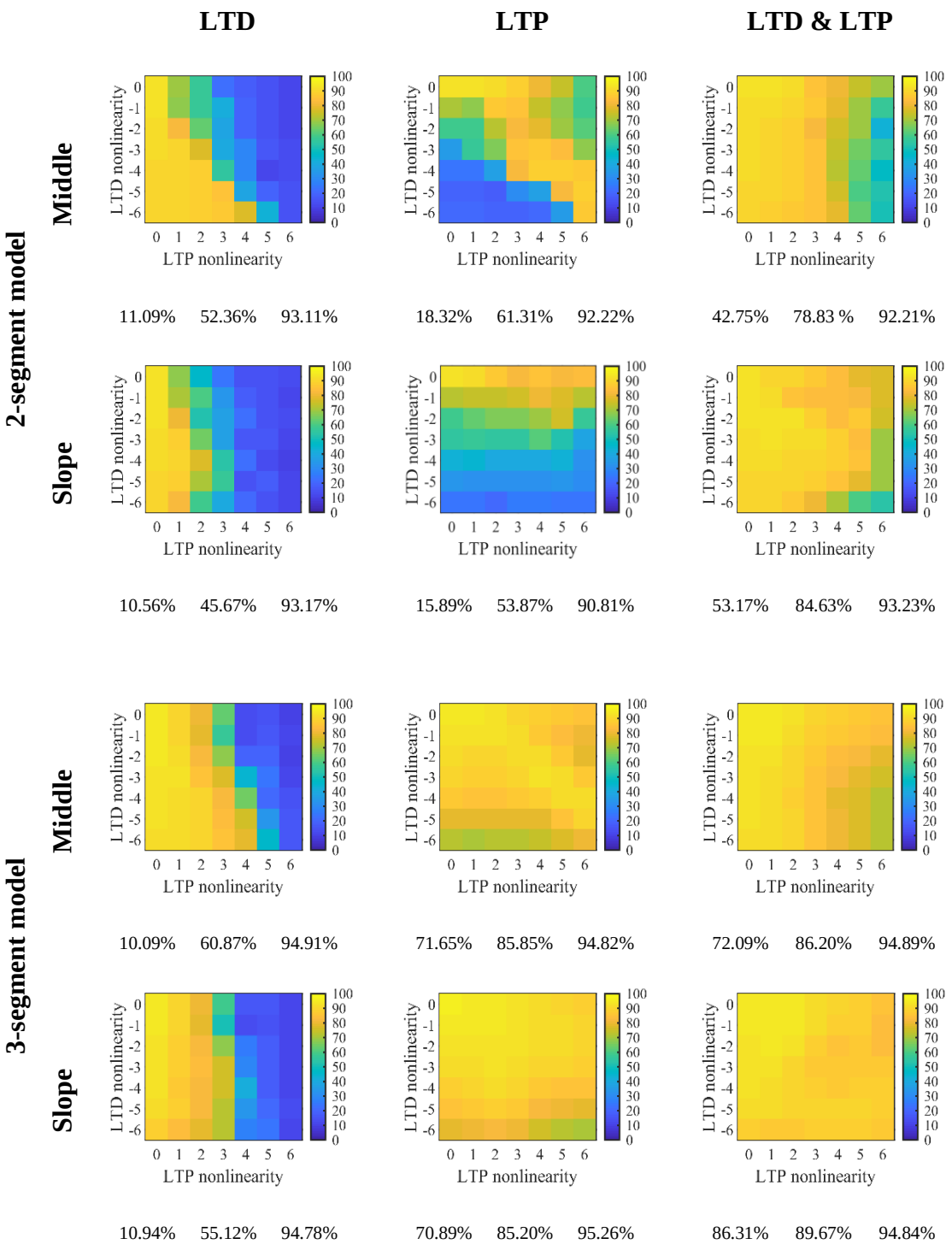


Figure 17. The prediction accuracy of digit recognition with PL method in different configurations. The numbers under each figure are minimum accuracy, overall (average) accuracy, and maximum accuracy.

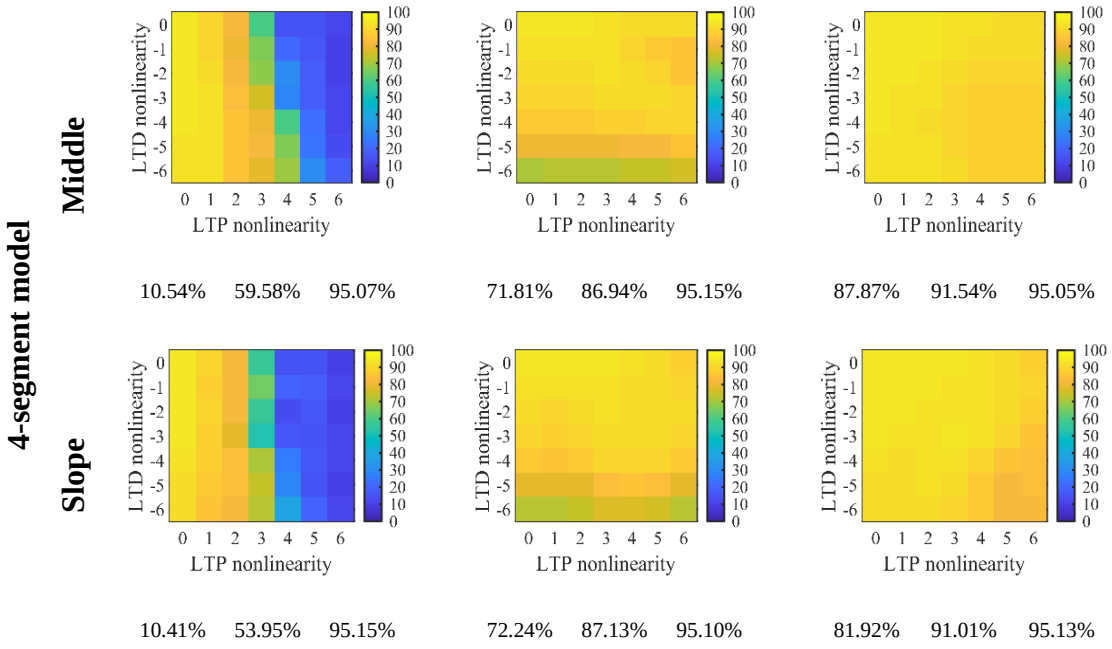


Figure 17. The prediction accuracy of digit recognition with PL method in different configurations (continued). The numbers under each figure are minimum accuracy, overall (average) accuracy, and maximum accuracy.

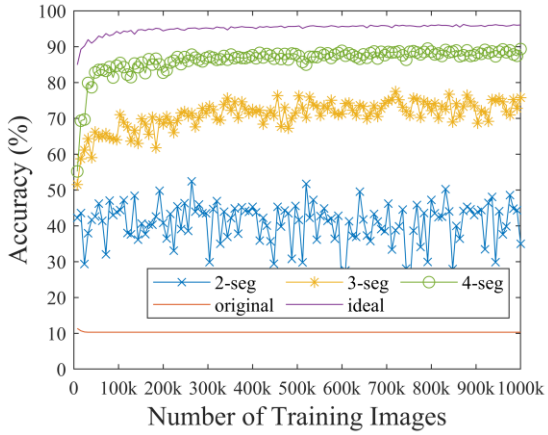


Figure 18. The recognition accuracy of the MNIST handwriting digits when the nonlinearity is NL: (6/-6) that is considered as the worst nonlinearity case.

4.4.2. Stability of Recognition Accuracy

We further analyze the stability of the recognition accuracy enabled by PL. As shown in Figure 17, memristors with NL: (6/-6) that is considered as the worst nonlinearity case is taken as an example to show the recognition accuracy change during the training process. The nonlinearity

property causes the accuracy of the original case to be stuck at 10%. The top line shows the ideal case that obtained by a pure learning algorithm without nonlinearity involved. The blue cross line, yellow star line, and the green circle line show the 2-segment, 3-segment, and 4-segment case with middle strategy and LTD & LTP, respectively. In Figure 17, each line has value fluctuations that are determined by the SGD algorithm. The 4-segment model has less fluctuation as compared to the 3-segment and the 2-segment model. As a result, the model with more segments can get more stable recognition accuracy. In addition, for the training process of neural network, at the beginning, because the weights are initially randomized, they will change a lot to fast converge with increasing training samples. Then the weight will be stable after several epochs. As shown in Figure 17, when the number of training samples reaches 5% of the total, the recognition accuracy begins to stabilize. Therefore, even if there is a large weight change and weight deviation in the early stage of the training, after several initial epochs, the weight updating will shrink down, then a single pulse as input is adequate to induce conductance change to cover such weight updating. Therefore, our method is still suitable for this learning/training mode. Furthermore, in most neural network algorithms, the rate of weight updating can be adjusted by a given parameters, which also could help make the proper function when using the PL method.

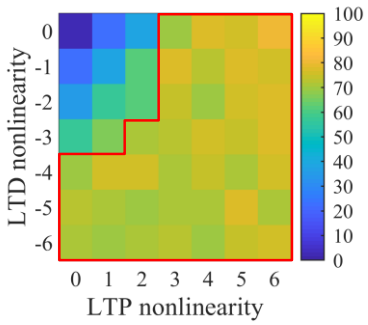
4.4.3. Impact of LTD and LTP

Furthermore, as shown in Figure 16, when the PL method applies only in LTD process and remain LTP process, the nonlinear effect of LTD is mitigated and the NL of LTP becomes the main factor of the accuracy lose. This is because when one process (LTD or LTP) updating performance is improved, the nonlinearity in this process has less effect on the result and the other process performance dominates the results' tendency. Therefore, if an asymmetric device has a relatively small linearity of LTD, but an aggressive nonlinearity of LTP, it is recommended to

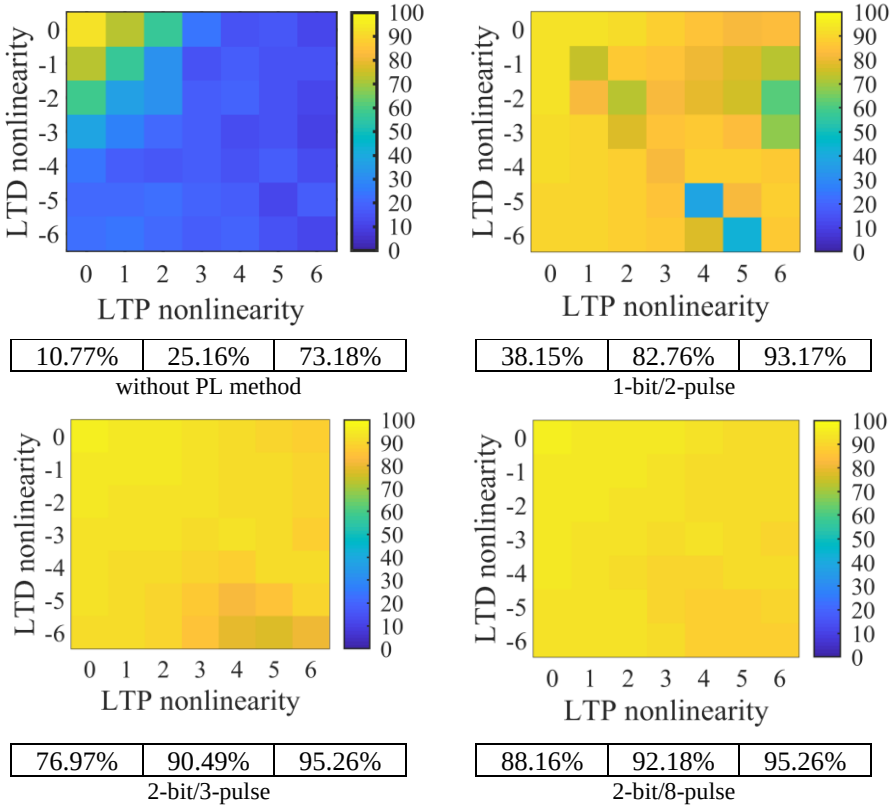
apply the PL method in LTP only, which reduce the cost. Accordingly, the PL method applied in LTP benefits more for memristors with high NL of LTP and low NL of LTD. In our simulations, on the one hand, we use asymmetric devices, on the other hand, the LTP decides the weight increasing process that is used to enhance the connection between two neurons, which is more critical and sensitive. So, in Figure 16, the PL method applied for LTP achieves 8.2%-33.2% higher accuracy than that for LTD. Finally, when the PL method applies in LTD & LTP, the nonlinear of both processes is mitigated, and the overall accuracy is constantly higher than one process condition.

4.4.4. Impact of Split Selection Strategies

In addition, as shown in Figure 16, for 2-segment, 3-segment, and 4-segment models, when the PL method applies in three conditions (LTD, LTP, and LTD & LTP), the middle strategy shows difference as (6.7%, 7.4%, -5.8%), (5.8%, 0.7%, -3.5%), and (5.6%, -0.2%, 0.5%) compared with the slope strategy. It indicates that as the segment increase, the accuracy difference between middle and slope strategy decrease. This is because, with more segment's divisions, the polyline introduced by the PL method fits the conductance curve better so that the difference between these two strategies gets smaller and smaller. For the middle strategy, when the PL applies in LTD (LTP), only lower left corner (upper right corner) of the results are improved. Such unbalanced optimization for various NL causes higher overall (average) accuracy than the slope strategy. However, when the PL method applies in LTD & LTP, the slope strategy shows more uniform accuracy improvement and gets higher overall (average) accuracy improvement than the middle strategy. Accordingly, from the view of overall (average) accuracy, the middle strategy is better for one process implement (LTD or LTP) but the slope strategy is better for both processes (LTD & LTP).



(a)



(b)

Figure 19. (a)The improvement of recognition accuracy. The red outline shows the cases that have accuracy improvement over 70%. (b) The highest accuracy achievement under the same cost. The number under each figure means minimum accuracy, overall accuracy, and maximum accuracy (except ideal case NL: (0,0)).

4.4.5. Impact of NL

The effectiveness of the PL method is also influenced by NL. Figure 18 (a) shows the amount of recognition accuracy improvement using 4-segment PL model compared with the original one without PL applied. Among 49 NL cases (expect case without nonlinearity: LTP=0, LTD=0), 38 cases achieve over 70% accuracy improvement with a large NL, which are ranged with a red curve, 5 cases achieve 55%-70% improvement with a moderate NL, and 5 cases achieve 20%-40% improvement with a small NL.

4.4.6. Storage and Pulses Cost of PL Method

To implement the PL method, $\log_2(n)$-bit and n pulses with different durations are needed, as shown in Table 1. Although more segments with LTD & LTP cases obtain higher accuracy, more bits of memory and types of pulses are required, increasing the cost overhead. To represent the relationship between average accuracy of 49 various memristors and cost of the PL method, the accuracy index is calculated based on Equation 11. The high accuracy index indicates the high comprehensive efficiency.

$$\text{Accuracy index} = (\text{Accuracy} - C1) / (B + \alpha \times P + C2) \quad (11)$$

where C1 represents the overall accuracy of the original case, which is 25.16%; C2 represents a basic cost of the peripheral circuit, which is determined by a practical situation, and it is set to 7 based on the layout of the hardware simulator [9] that we adopt in this paper; B and P represent the cost of the bits of storage and the cost of pulse generators; α represent the cost ratio between B and P, respectively. The accuracy index range is shown in Table 1 when α ranges from 0.1 to 1, which refers to practical circuit design [9, 65]. Thus, the accuracy index that is shown as a reasonable range to reflect the trade-off between the cost and the accuracy requirement. For example, if α equals 1, the 3-segment model with middle strategy and LTP shows the highest

accuracy index (5.1), which results in less than 4% accuracy loss as compared that with slope strategy and LTD & LTP, but only need half of pulse generators and 66.7% memory cost, it has the highest comprehensive efficiency.

Finally, taking the cost as a primary consideration, we compare the performance of the PL method. Figure 18 (b) shows the highest accuracy achievement under the same cost for 49 memristors among results in Figure 16, which includes three representative cost profiles. Using the PL method with a cost of 1-bit/2-pulse, the overall accuracy can be 82.76%, which is increased by 57.6% compared with the original case. With the cost of 2-bit/3-pulse, the overall accuracy reaches 90.49%, which has 7.73% higher than the cost of 1-bit/2-pulse. The 2-bit/8-pulse case needs nearly three times more pulses than the cost of 2-bit/3-pulse, but the accuracy improvement is approximately the same as 2-bit/3-pulse. Accordingly, although the cases with a cost of 2-bit/3-pulse can enable a better tradeoff, if the cost saving has a higher priority, the 1-bit/2-pulse is a better choice.

Table 1. Cost and Overall Accuracy with the PL Method in Different Cases. [a]

	model	strategy	#Bit	#Pulse	Acc LTD (%)	Acc Index	Acc LTP (%)	Acc Index
LTP or LTD	2	Middle	1	2	52.36	2.7~3.3	61.31	3.6~4.4
		Slope	1	2	45.67	2.1~2.5	45.19	2.0~2.4
	3	Middle	2	3	60.87	3.0~3.8	85.85	5.1~6.5
		Slope	2	3	55.12	2.5~3.2	85.52	5.0~6.5
	4	Middle	2	4	59.58	2.6~3.7	86.94	4.8~6.6
		Slope	2	4	53.95	2.2~3.1	87.17	4.8~6.6
					Accuracy (%)		Accuracy Index	
LTD & LTP	2	Middle	1	4	78.83		4.5~6.4	
		Slope	2	4	84.63		4.6~6.3	
	3	Middle	2	6	86.20		4.1~6.4	
		Slope	3	6	89.67		4.0~6.1	
	4	Middle	2	8	91.54		3.9~6.8	
		Slope	3	8	91.01		3.7~6.1	
original (without PL)			0	0	25.16		0	

[a] *#Bits, #Pulse, and Acc represent cost of the bits of storage for per synapses, cost of the types of pulses to generate for the neural network, and recognition accuracy of simulator, respectively.*

4.4.7. Variations of Memristors

Because of physical limitations of memristors, ON/OFF ratio variation (G_{max}/G_{min}), minimum conductance variation (G_{min}), maximum conductance variation (G_{max}), cycle-to-cycle variation (CtoC), and device-to-device variation (DtoD) [9] exist in the application of memristor based hardware implementation. To explore the effectiveness of the PL method, we investigate these variations following standard/Gaussian distribution N (μ, σ) into consideration using 4-segment model and middle strategy. In our simulations, ON/OFF ratios are configured from 13 to 14. Minimum conductance subjects to N (G_{min}, $\sigma \times G_{min}$), and maximum conductance subjects to N (G_{max}, $\sigma \times G_{max}$). Device-to-device variation represents nonlinearity variation of memristors in crossbar array, which subjects to N (NL(LTP), σ) and N (NL(LTD), σ) distribution and cycle-to-cycle variation represents conductance deviations in each weight update, which is illustrate as Equation 12 [9].

$$G = G + (G_{max} - G_{min}) \times N(0, \sigma) \times Np^{\alpha} \quad (12)$$

where *Np* represents the needed pulse number in each weight update, α represents the impact of *Np* and it is set to be 0.5 in our simulations. Above, LTP, LTD, G_{max}, and G_{min} are fixed parameters for each simulation. Table 2 shows two circumstances of variation and Table 3 shows the recognition accuracy for memristors under different circumstances. In Table 3, for the original circumstances without the PL method, the recognition accuracies are lower than 57%. This is because the SGD algorithm has the property of sublinear convergence under certain conditions. However, after adding nonlinear parameters and variation parameters, such conditions are destroyed, resulting in convergence failure [66-68]. The PL method provides solution on mitigating impact of memristors' nonlinearity property, thereby avoiding convergence breakage. As shown in Table 3, for small variation (Var. 1), the PL method can keep the accuracy large than

70%, which is 55% improvement on average compared with the case without the PL method for NL from (0, 0) to (6, -6). For large variation (Var. 2), recognition accuracies decrease, but they still have 39% improvement on average for NL from (0, 0) to (6, -6). It concludes that even with various variations, the PL method is still efficient to improve the recognition accuracy, although its performance degrades when large variations involve.

Table 2. Variations Cases.

	Variations				
	DtoD σ	CtoC σ	Gmax σ	Gmin σ	Gmax/Gmin
Var. 1[a]	1	1%	18%	18%	14
Var. 2	2	3%	24%	30%	13

[a]Var. 1 and Var. 2 represent two circumstances that contain five variations.

Table 3. Recognition Accuracy with Variations.

		Nonlinearity (LTP, LTD)					
		(1, -1)	(2, -2)	(3, -3)	(4, -4)	(5, -5)	(6, -6)
Without the PL method	Original accuracy	56.7%	31.7%	18.0%	14.8%	11.2%	11.8%
With the PL method	No variations	94.6%	92.8%	90.4%	88.7%	88.1%	88.3%
	With Var. 1	89.1%	85.0%	79.1%	76.0%	74.4%	71.1%
	With Var. 2	78.9%	70.5%	62.5%	59.8%	58.3%	52.5%

4.4.8. Different Neural Network

To verify the effectiveness of the PL method when the neuron network changes or becomes more complex, we conduct simulations of neural network with different number of neurons using 4-segment model and middle strategy. For memristors with NL from (0, 0) to (6, -6), Table 4 shows the recognition accuracy of neural networks with the neuron number in hidden layers are 40, 100, and 160, respectively. The results as over 83% recognition accuracy verify that the PL method is still effective when the structure of the neural network changes (100 neurons in hidden layers resulting in Figs. 5 and 6). This is because the proposed method focuses on optimizing the hardware performance of the neural network by improving device-level. When the neural network

changes or the learning task becomes more complex, the PL method mitigates the nonlinearity impact of the memristor, thereby making the recognition of the memristor-based hardware are close to the results from pure algorithm. Actually, the PL method is independent of the structure of the neural network and complexity of the task.

Table 4. Recognition Accuracy of Networks with Different Neurons.

#N [a]	Nonlinearity of memristor (LTP, LTD)						
	(0,0)	(1,-1)	(2,-2)	(3,-3)	(4,-4)	(5,-5)	(6,-6)
40	92.2%	92.1%	91.0%	88.4%	85.9%	84.7%	83.5%
100	95.5%	94.7%	92.7%	90.7%	88.9%	88.0%	88.5%
160	95.8%	95.2%	92.1%	90.4%	88.7%	87.4%	88.6%

[a] #N *represents the number of neurons in hidden layer of neural network.*

4.4.9. The PL Method and Other Works

Table 5. The Comparison of the State-of-art.

	[2]	[9]	[34]	[35, 36]	This
Without precise read before writing	×	√	√	√	√
Without always change pulse amplitude	√	√	√	×	√
Without always change pulse duration	×	×	√	√	√
Nonlinearity almost disappeares	√	√	×	√	√

The PL method is a simple, feasible, and universal method. As shown in Table 5, as compared with the state-of-art, the PL method does not need to read the conductance of memristor in every write step and does not need to change the amplitude or the duration of the update-pulses each time so that it avoids complex peripheral circuits. What's more, with more segments, the nonlinearity of memristor almost disappears. Accordingly, the PL method is an effective technique to address the weight deviation issue caused by the nonlinearity property of memristors; also, it provides multiple configurations to meet different requirements of various applications.

4.5. Conclusion

In this chapter, a PL method has been presented to mitigate the weight update error caused by the nonlinearity property of memristors. The 2-segment, 3-segment, and 4-segment models in two split selection strategies have been developed, which results 18 configuration cases. As compared to that without the PL method, each case provides over 20% higher overall accuracy. Also, this method realizes higher improvements of the recognition accuracy for higher NL and it achieves at least 87.87% accuracy for all 49 NL cases, which indicates that even devices with high-nonlinearity (e.g. NL:(6, -6)) can be used as a synapse in neuromorphic hardware implementation. Also, when various variations of memristors exist, the PL method is investigated to verify its effectiveness. In addition, the following conclusions can be drawn from our study: 1) The proposed PL method with more segments shows higher overall accuracy and stability; 2) From the view of overall (average) accuracy, the middle strategy is better for one process implement (LTD or LTP) but the slope strategy is better for both processes (LTD & LTP); 3) The PL method realizes higher improvements of the recognition accuracy for higher NL; 4) The PL method provides flexible configurations so that it can achieve 57.6% recognition accuracy improvement in a low-cost situation, but if a higher accuracy is desirable, additional cost can be taken to realize 67.02% accuracy improvement. Finally, as compared with the state-of-art, the PL method avoids complex peripheral circuits, and with more segments, the nonlinearity impact of memristor to the neuromorphic hardware is almost negligible.

5. MEMRISTOR-BASED NEUROMORPHIC HARDWARE IMPROVEMENT FOR PRIVACY-PRESERVING NEURAL NETWORK

Because of collecting a large amount of personal data, when artificial neural network (ANN) is used in human related topic, it has raised great concern on privacy preservation. A robust solution is to introduce noise injection mechanism as differential privacy that promises strong theoretical privacy guarantees. However, privacy-preserving ANN with noisy input data has a substantial risk of reducing the recognition accuracy. Therefore, it is urgently needed to have technologies that can make users' data applied to neural networks while strictly protecting sensitive information. In this paper, a method is proposed to address this accuracy degradation by optimizing the performance of memristor in weight updating processes. Instead of complying with the traditional hardware and algorithm, the proposed method calculates update parameters along a piecewise line by using different input pulses. The proposed method can mitigate nonlinear problem of memristor without pre-reading the precise current conductance each time, thereby avoiding complex peripheral circuits. The effectiveness of the proposed method with respectively 2-segment, 3-segment, and 4-segment models is investigated. The results show that under different nonlinearity and different perturbation noise required by differential privacy theory, the proposed method can increase the recognition accuracy of MNIST handwriting digits by 39.67% on average, which provides more space and margin for privacy-preserving technology.

5.1. Introduction

Privacy preservation is a critical challenge for ANN. The privacy preserving in ANN is to release statistical information from collected datasets without compromising the privacy protection of the individual respondents. An effective method is to introduce a randomized noise mechanism for differential privacy technology to quantify the protection ability. Usually, software-based

machine learning algorithms easily generate such randomized noise [44]. However, noisy and distorted data would lead to a degradation of the recognition accuracy in ANN. Accordingly, solutions to balance privacy preserving and recognition accuracy are indeed needed. One popular solution is adopting a specific algorithm, but with considerable computation overhead, which is neither acceptable for a general-purpose computing system such as data center because of the increasing workload, nor sufficient to satisfy the portable and edge computing system due to the resource-constrained, such as mobile devices, wearable devices, and IoT devices. Therefore, in this paper, we propose a memristor-based neuromorphic hardware improvement to enable privacy-preserving ANN without accuracy degradation. That is to use linear optimization (PL) method to alleviate nonlinearity of memristors for weight updating in ANN to counteract recognition degradation due to noise injection, as shown in Figure 19. Therefore, privacy-preserving ANN provides enough space for randomizing noisy data to ensure that the publicly visible information do not change much if one individual in the dataset changes, which is enabled by differential privacy technology - a strictly provable, quantized, and security-controlled method.

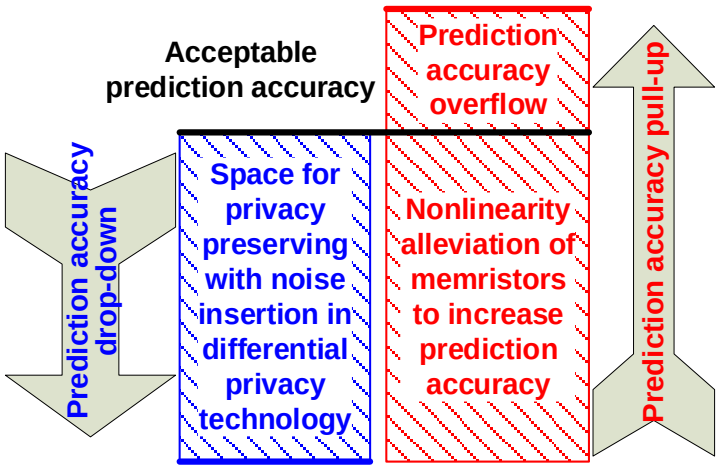


Figure 20. Concept of memristor-based neuromorphic hardware improvement for privacy-preserving ANN.

5.2. Methodology

5.2.1. Memristors

To implement the piecewise linear (PL) method that is introduced in the section 4, firstly, because of variations exist, we need to find the average normalized conductance change curve of over 1000 representative memristor samples. In this case, although the proposed method solves nonlinear problems in varying degrees for each device, it can still greatly alleviate the overall weight-deviations problem of a memristor-based array, which will be discussed in Section V. F. Then, we need to choose split points that can divide the conductance curve into several segments. Thus, the piecewise line can be gotten. Next, the duration of input pulses can be calculated by the slope of the original ideal line, k0, and the slopes of the piecewise line, $\{k_1, k_2, \ldots, k_n\}$. Specifically, the PL method scales the duration of pulses to k_0/k_i times of the original duration to balance the conductance change caused by the nonlinear effect. To implement the PL method, a set of memory, *log2(n)*-bit memory, is also needed to store the segment information, which is used to select a slope before each weight update. Here, n represents the segment number, such as two, three, and four. In this way, the larger/smaller the line's slope is, the shorter/longer the duration can be selected. Finally, after each weight update, the comparison operation should be completed to make sure the memory is updated based on the current conductance range of each memristor. This comparison does not need to read precise conductance of memristors but needs to compare with a reference value to recognize the current segment information of memristors.

5.2.2. Differentially Private Transformation

According to [64], differential privacy is immune to post-processing: A data analyst, without additional knowledge about the private database, cannot compute a function of the output of a private algorithm and make it less differentially private. This property of differential privacy

supports the differentially private transformation algorithm [10, 39]. This paper adopts the Laplacian mechanism and follows the private transformation algorithm [39] where the privacy budget ε can be proved to be calculated with given input data and the neural network by the following equation.

$$\varepsilon = 2\sigma \tag{13}$$

where σ is the noise scale in private transformation algorithm in [39].

5.3. Experimental Evaluation

In this section, digit recognition tasks are used as experimental examples to evaluate the effectiveness of the PL method that aims to mitigate the effect of a memristor's nonlinearity in a hardware implementation. A comprehensive suite of simulations has been conducted to explore the space of privacy-preservation using the proposed PL method in memristor-based ANN. We adopt the neural network hardware platform NeuroSim+ [48] with nonlinearity property injection, as well as private transformation algorithm [39] to perform hardware-based privacy-preserving recognition through the Modified National Institute of Standards and Technology (MNIST) database [19]. The neural network of this simulator includes 400 neurons as input, 100 neurons as a hidden layer, and 10 neurons as an output layer, which is used for recognizing 10 number digits. Each simulation trains up to 125 epochs. Each epoch selects 8,000 images randomly from 60,000 training images and takes 10,000 images as a testing dataset.

5.3.1. Models

The proposed PL method needs to select the split points in order to determine the types of pulses. According to the discussion of the weight update error in Section IV, the more segments, the less weight deviation that is caused by the nonlinearity but with higher circuit cost. Thus, in order to investigate the tradeoff between the recognition accuracy of privacy-preserving ANN and

the cost of the PL method, we conduct three models including 2-segment, 3-segment, and 4-segment models.

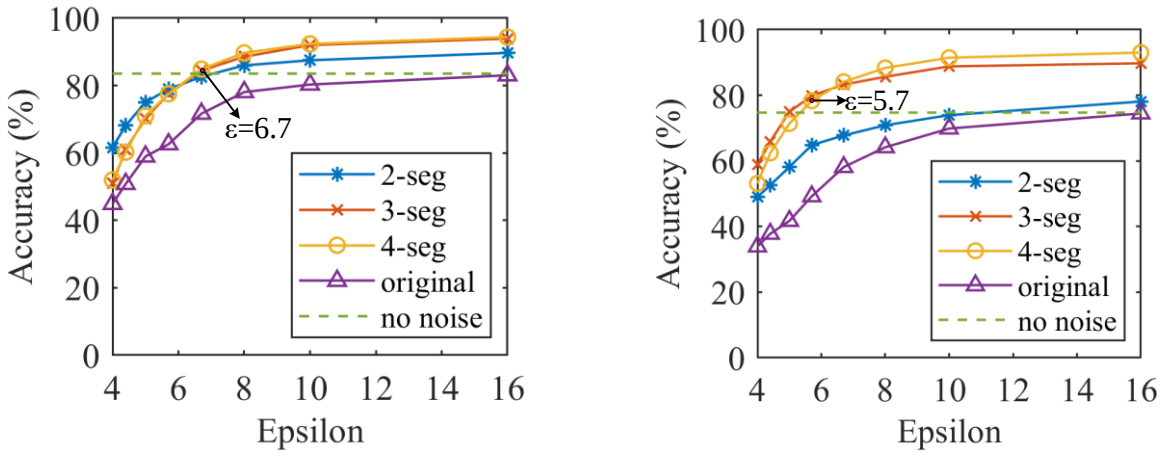
As shown in Figures 10, 11, and 12, for 2-segment, 3-segment, and 4-segment models, split points are selected where they can divide the conductance range into two, three, and four equal parts, respectively.

5.3.2. Impact of Private Perturbation

To explore the impact of PL method on the memristor-based privacy-preserving ANN, simulations with ε from 4 to 16 that reflects the strength of private preservation are conducted. The smaller the ε is, the larger the noise injection is needed, vice versa. To compare ANN with different nonlinearity of memristor, firstly, we conduct six groups of simulations with six different memristors that have the same absolute nonlinear value of LTP and LTD process. As shown in Figure 20, in each figure, the accuracy increases as the ε increases and all cases with the PL method have better performance as compared with the original case without the PL method applied. Among the three models of the PL method, the model with a higher segment shows higher accuracy. What's more, as the memristors' nonlinearity of ANN increase from (1/-1) to (6/-6), the differences between different models become larger and larger.

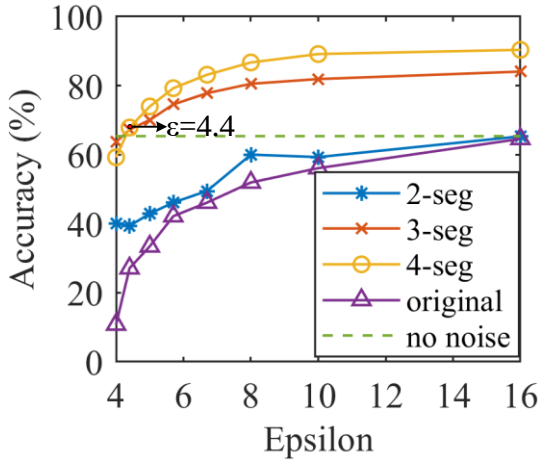
It concludes that the accuracy of 4-segment model not only keeps an accuracy over 70% when ε is larger than 5, but also has less than 10% accuracy difference from NL (1/-1) to NL (6/-6). However, the original model without the PL method gets much more increasing accuracy loss (at least 10%) as nonlinearity increases. Therefore, the 4-segment PL method shows more benefits as the nonlinearity of memristor increases in ANN. Furthermore, in some cases, the accuracy with the privacy preservation gets even higher accuracy when the PL method applied than the case without privacy preservation, for example, for 4-segment model, in NL (1/-1), (2/-2), (3/-3), (4/-

4), (5/-5), and (6/-6), when $\varepsilon \geqslant$ 6.7, 5.7, 4.4, 4, 4, and 4, respectively. This indicates the PL method makes the memristor-based ANN hardware get more space for privacy preservation, which would lead to stronger privacy preservation. In addition, the PL method provides various PL models that can be chosen according to the nonlinearity of memristor. When the nonlinearity of the memristor device is relatively small, such as (1/-1) and (2/-2), the results of the 3-segment model are similar to that of the 4-segment model. Therefore, in this case, considering trade-off the performance and cost, the 3-segment is a better choice.

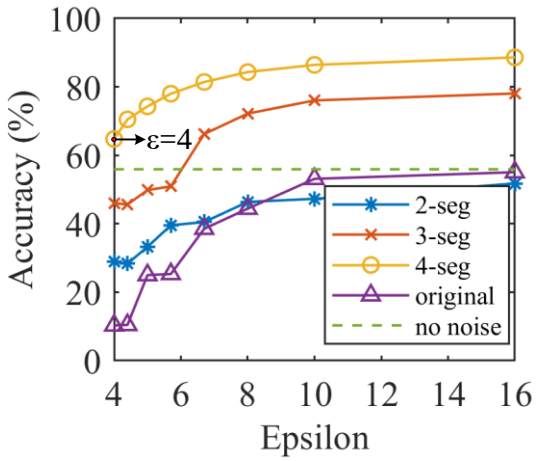


(a) NL (1/-1) (b) NL (2/-2)

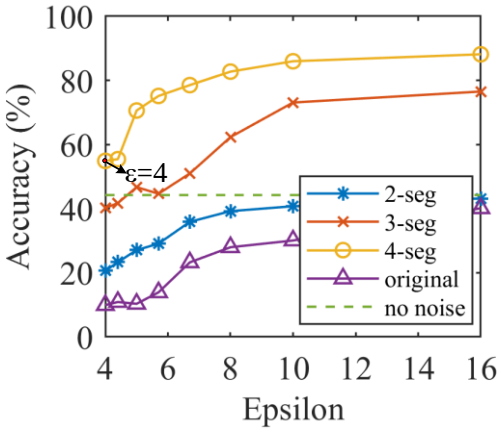
Figure 21. Recognition accuracy of the MNIST handwriting digits applying four method models with different private perturbation. The NL (x/-y) means the LTP nonlinearity of memristor is x and the LTD nonlinearity of memristor is y.



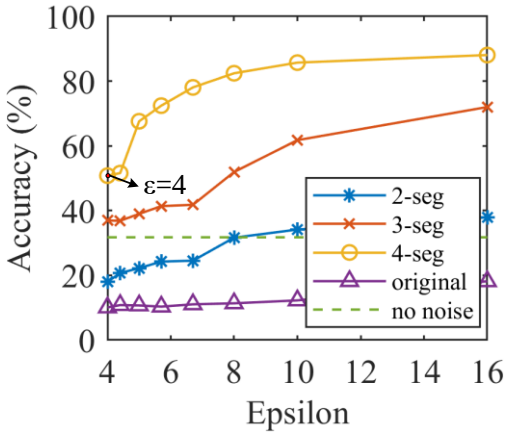
(c) NL (3/-3)



(d) NL (4/-4)

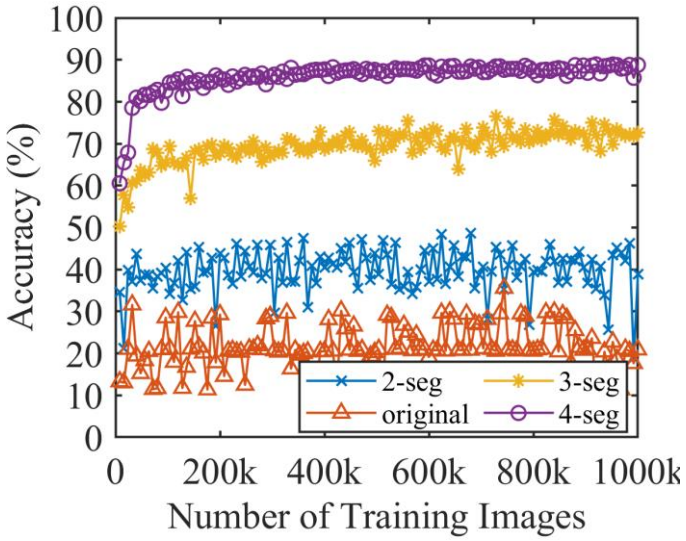


(e) NL (5/-5)

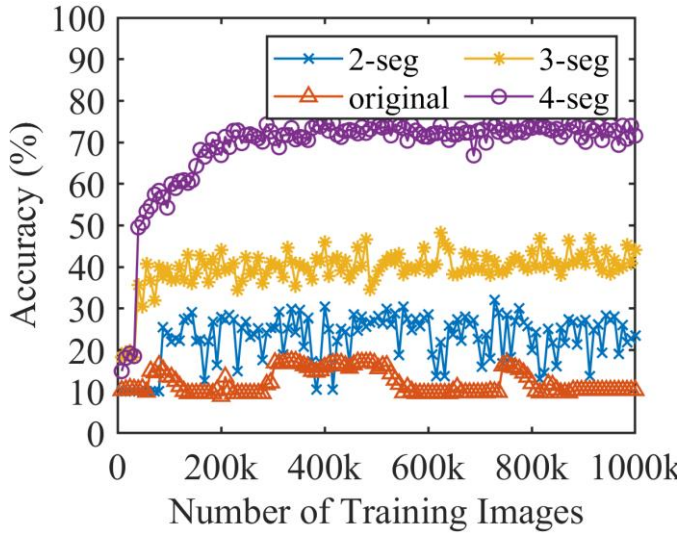


(f) NL (6/-6)

Figure 21. Recognition accuracy of the MNIST handwriting digits applying four method models with different private perturbation (continued). The NL (x/-y) means the LTP nonlinearity of memristor is x and the LTD nonlinearity of memristor is y.

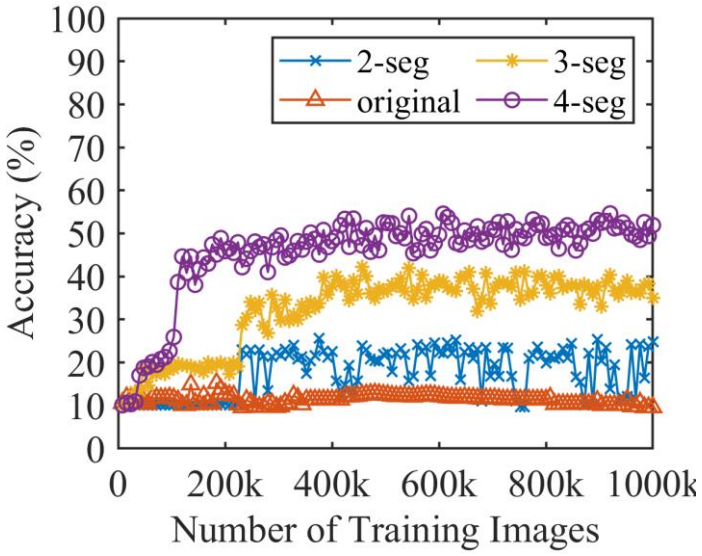


(a) small noise (ε=16)



(b) middle noise (ε=5.7)

Figure 22. Recognition accuracy of the MNIST handwriting digits with the training images when the nonlinearity is NL (6/-6) that is considered as the worst nonlinearity case.



(c) large noise (ε=4)

Figure 22. Recognition accuracy of the MNIST handwriting digits with the training images when the nonlinearity is NL (6/-6) that is considered as the worst nonlinearity case (continued).

5.3.3. Models Stability

Then, we take the worst nonlinearity case of memristor (NL (6/-6)) as an example to explore the impact of different PL method models with ε = 16, 5.7, and 4, on recognition accuracy during the training process.

As shown in Figure 21, the accuracy of each PL model has a short time of fluctuation before convergence, which is decided by the Stochastic Gradient Descent (SGD) algorithm of the hardware simulator. However, these fluctuations are different for different models. The 4-segment model always gets the highest accuracy as well as the most stable accuracy plateau and the 3-segment model is also more stable than the 2-segment model. The fluctuation of accuracy increases as the noise level increases. In addition, because a larger segment model makes weight updating more fit to the real memristor, causing a smaller weight updating deviation, the case with more segments has higher and more stable recognition accuracy during the training process.

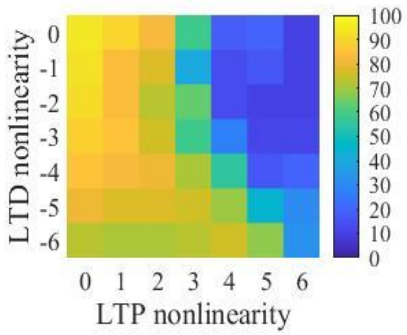


Figure 23. Recognition accuracy of the MNIST handwriting digits without the privacy-preservation and PL method. The average accuracy of the 49 memristors' nonlinearity cases is 55.97%.

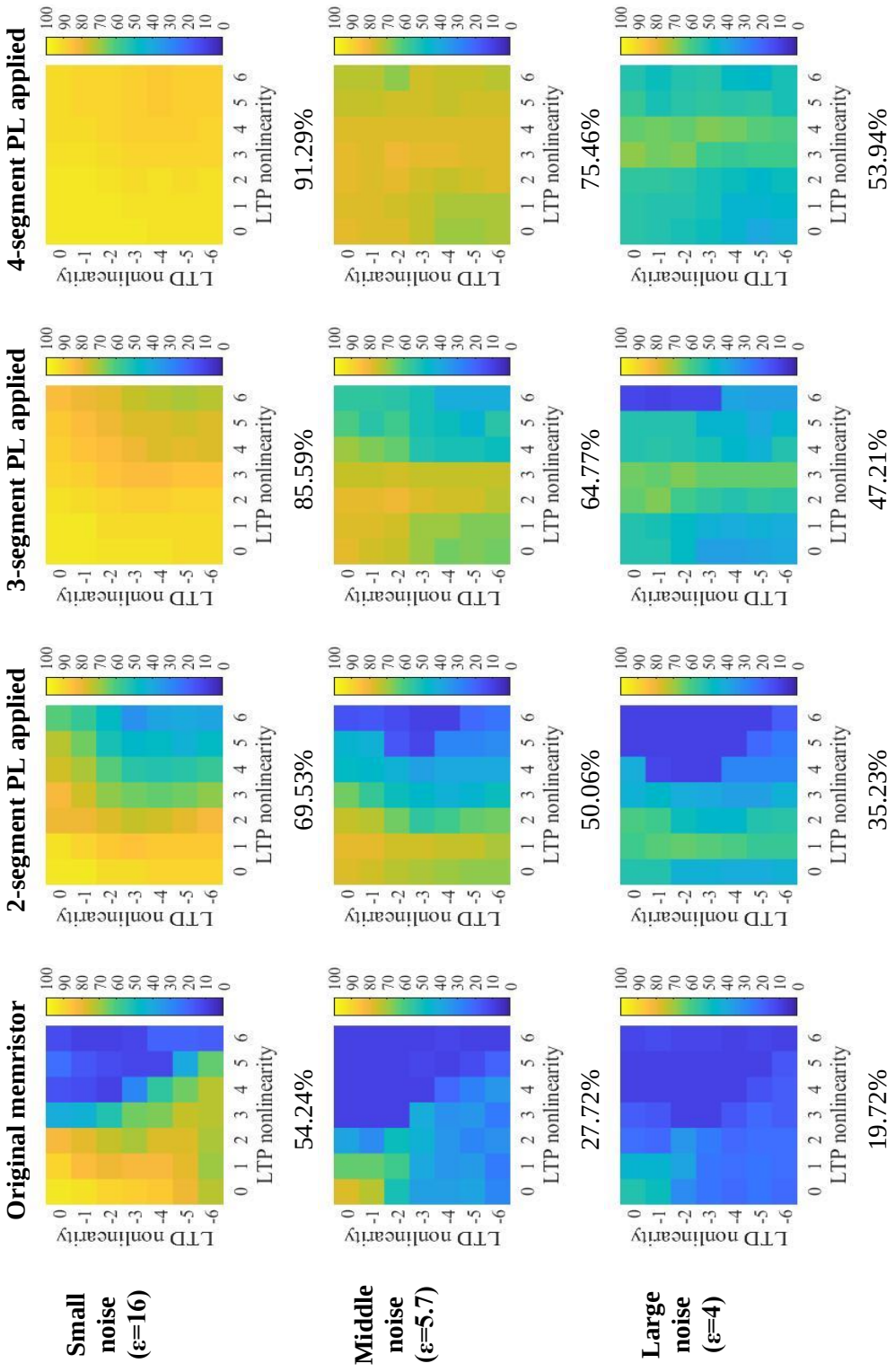
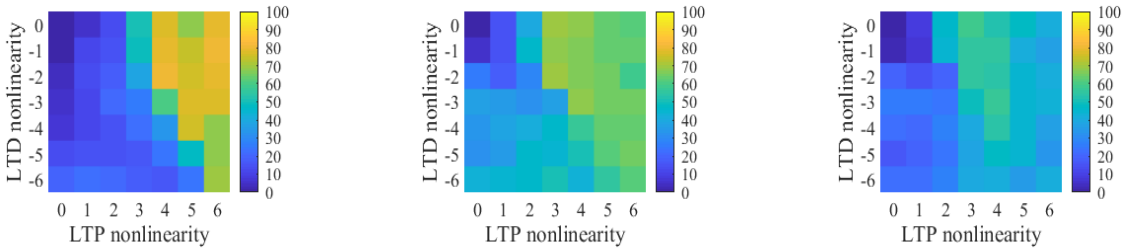


Figure 24. The recognition accuracy of memristor-based ANN with different models for 49 nonlinearity cases of memristor.

5.3.4. Privacy-preserving Space for Various Nonlinearity

Since for a memristor, the NL of LTP is not usually equal to the NL of LTD [17-20], we further investigate 49 different memristor devices whose NLs of LTP and LTD are from 0 to 6 (-6). For those 49 memristors, we perform 49 simulations in four models (original case without the PL method, 2, 3, and 4 segment models) and with eight privacy protection noises, respectively. We also simulated the ANN without privacy protection and PL method, resulting in Figure 22, where the recognition accuracy result is represented by a colored square and the average accuracy of these 49 memristor nonlinearity cases is 55.97%, which is regarded as a parameter that reflects the overall performance of the original ANN. Next, Figure 23 shows recognition accuracies of ANN hardware without PL applied and with three PL models applied, respectively. It shows the results of three noise level applied (ε=16 (small), 5.7 (middle), and 4 (large)). To compare the overall effectiveness of the proposed method, the average accuracy results are calculated by averaging the accuracy of 49 NL cases, which is shown under each figure. The results show an increasing accuracy, when the segment in PL model increases, the noise level decreases, and the nonlinearity decreases, respectively. When the 4-segment PL model is applied and ε equals 5.7, the average can reach 75.46% that is a 47.74% improvement compared to the result 27.72% for that without the PL model. Figure 24 shows the accuracy improvement in three noise level applying the 4-segment model. It shows that the average accuracy improvements are 37.05%, 47.73%, and 34.22%, respectively. Moreover, to study the average accuracy improvement of the PL method, Table 6 lists the average accuracy for four models with the ε between 4 and 16. As listed in Table 6, when the PL method applied, the case with the average accuracy that is larger than the case without noise (55.97%), gives more space for privacy-preservation.

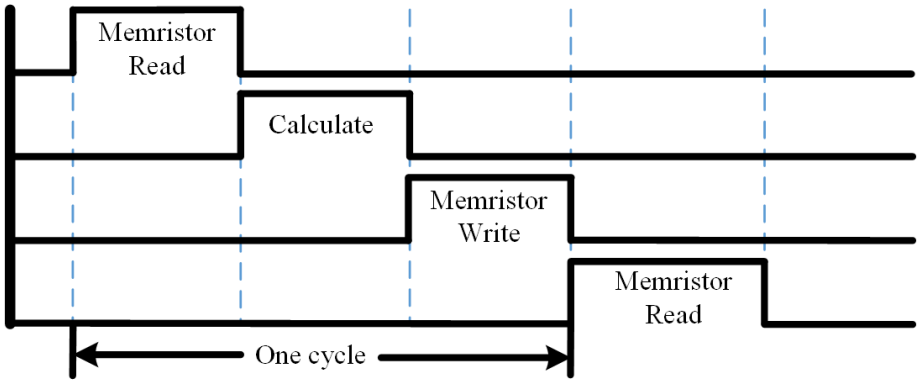


37.05%
(a) small noise (ε=16)

47.73%
(b) middle noise (ε=5.7)

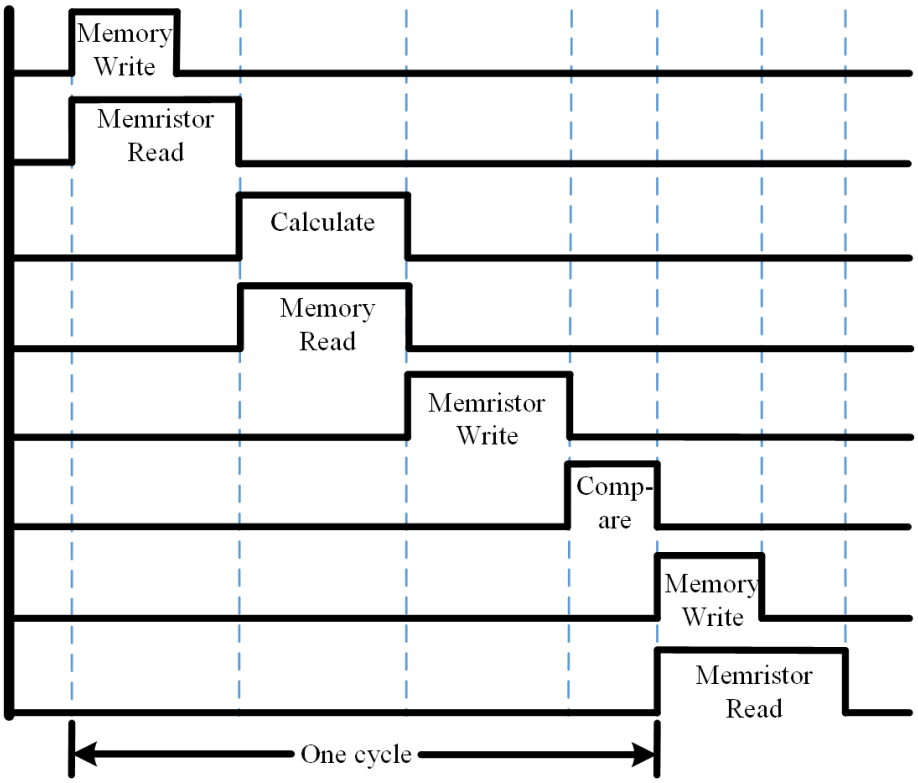
34.22%
(c) large noise (ε=4)

Figure 25. Recognition accuracy improvement of 4-segment LO model. The average accuracy improvement of each figure is shown under each figure.



(a) Without PL method

Figure 26. One cycle timing schematic in weight update process. (a) Without PL method, (b) With PL method. The memory represents the added memory component and the memristor represents the memristor that acts as a synapse in neural network.



(b) With PL method

Figure 26. One cycle timing schematic in weight update process (continued). (a) Without PL method, (b) With PL method. The memory represents the added memory component and the memristor represents the memristor that acts as a synapse in neural network.

Table 6. Cost and Average Accuracy with Different Models.

Model		original	2-seg	3-seg	4-seg
Bit # cost		0	1	2	2
Pulses # cost		2	4	6	8
Average Accuracy (%) for 49 NL cases [a]	ε=4	19.72%	35.23%	47.21%	53.94%
	ε=4.4	20.70%	39.33%	53.66%	**60.94%**
	ε=5	24.37%	45.02%	**58.62%**	**68.87%**
	ε=5.7	27.72%	50.06%	**64.77%**	**75.46%**
	ε=6.7	33.56%	55.20%	**71.95%**	**81.17%**
	ε=8	40.93%	**62.73%**	**78.31%**	**85.94%**
	ε=10	48.07%	**65.59%**	**82.69%**	**89.18%**
	ε=16	54.25%	**69.56%**	**85.59%**	**91.29%**
	No noise	55.97%	-	-	-

[a] *The data in bold show the cases that have higher accuracy than the original case without LO method applied (55.97%) as well as without privacy preservation.*

5.3.5. Cost Analysis

Models with more segments have better performance of ANN, but they need more cost of storage space and circuits to generate more types of pulses as listed in Table 6. However, some nonlinearity cases, their performance with 2-segment or 3-segment is similar to 4-segment model, such as memristors with NL (1/-1) in Figure 23 (a). Therefore, although the 4-segment model achieves the highest average accuracy for 49 NL cases, we should consider different NL cases independently to lower the unnecessary cost of storage space and circuits. Also, the PL method brings additional access in order to read and write the memory used for storing segment information. As shown in Figure 25, these accesses do not induce extra latency, because they are conducted with memristors' reading and calculation at the same time, nearly hidden and covered in memristors' operations. The proposed method only brings latency overhead for the comparison operation, as shown in Figure 25. But such comparison time is only a small ratio in one cycle. Finally, as compared with state-of-art works [2, 9, 34-36, 39], the proposed method needs less circuit complexity including a simpler pulse generator for the same amplitude pluses.

5.3.6. Variations of Memristors

The physical mechanism of the conductance modulation in most prospective synaptic devices is typically an ionic reconfiguration process based on electro/thermo-dynamics. This thermally activated ion migration and process variations are responsible for unavoidable variations including nonlinearity, device-to-device, cycle-to-cycle, and ON/OFF conductance variations [7, 9, 69]. Considering variations exist among real devices, we simulate five variations that subject to standard normal distribution N (μ, σ) to explore the effectiveness of the 4-segment PL method. In our simulation, minimum conductance, maximum conductance, and device-to-device variation

subject to N (G_{max}, $\sigma \times G_{max}$), N (G_{min}, $\sigma \times G_{min}$), and N ($NL(LTP)$, σ) and N ($NL(LTD)$, σ), respectively. Cycle-to-cycle variation is illustrated as Equation 14 [9].

$$G = G + (G_{max} - G_{min}) \times N(0, \sigma) \times Np^{\alpha} \quad (14)$$

where Np represents the needed pulse number in each weight update, α represents the impact of Np and it is set to 0.5 in our simulations. ON/OFF ratios are configured as 16 in variation 1 and 14 in variation 2. For Variations 1 and 2 in Table 7, we set σ of minimum conductance, maximum conductance, device-to-device, and cycle-to-cycle variation as 6%, 6%, 1/1, 1, and 18%, 18%, 3/3, 3, respectively [3].

In the presence of variations, the proposed method can keep recognition accuracies higher than 75% when noise level and variations are both small. Under the same circumstance without the PL method, recognition accuracies are not higher than 12%. Although as the variations and noise level increase, the accuracies of cases with the PL method decrease, they are still much higher than the accuracies without the PL method. It concludes that in real device condition when various variations exist, the proposed PL method is still proved to be an effective method for privacy preserving.

Table 7. Recognition Accuracy with Different Variations.

Epsilon		16		5.7		4	
Method		PL	Original	PL	Original	PL	Original
Variation 1	(2/-2)	88.89%	10.98%	76.75%	11.02%	61.59%	11.51%
	(4/-4)	84.12%	10.99%	69.27%	10.81%	50.56%	11.39%
	(6/-6)	76.57%	11.36%	47.45%	10.90%	36.60%	9.53%
Variation 2	(2/-2)	73.22%	10.44%	39.63%	10.35%	37.63%	13.27%
	(4/-4)	64.15%	11.01%	36.94%	10.74%	29.76%	12.17%
	(6/-6)	58.09%	11.73%	38.80%	11.51%	29.43%	10.11%

5.3.7. Comparison with Other Works

As listed in Table 8, instead of optimizing the algorithm for privacy preservation [39], the proposed PL method is simple and feasible to addresses accuracy degradation due to privacy preservation by optimizing ANN in the hardware. As compared with the state-of-art [2, 9, 34-36, 39], the PL method does not need to read the precise conductance of memristor before every write operation and does not need to change the amplitude of the update-pulses each time so that it avoids complex peripheral circuits. What's more, with a 4-segments model, the PL method almost immune to the nonlinearity of memristor. Because our simulations are all based on the standard SGD algorithm and a regular hardware simulator, the recognition results still have a large space to be improved by using a more efficient ANN algorithm or by high-performance memristor devices. The method we propose is a universal method that works for all memristor-based hardware with nonlinear characteristics. Accordingly, the PL method is an effective technique to address the weight deviation issue caused by the nonlinearity property of memristors in privacy-preserving ANN; also, it provides multiple configurations to meet different requirements of privacy preservation.

Table 8. The Comparison of the State-of-art.

	[39]	[37]	[34]	[2]	[35]	[36]	This work
Without precise read-before-write	√	√	√	×	√	√	√
Without always change pulse amplitude	√	√	√	√	×	×	√
Without always change pulse duration	√	×	√	×	√	√	√
Almost immune to nonlinearity	×	√	×	√	√	√	√
No need for algorithm optimization	×	√	√	√	√	√	√

5.4. Conclusion

In this paper, the linear optimization (PL) method is proposed to improve the performance of memristor-based privacy-preserving ANN and it is verified based on MNIST database, the differentially private algorithm, and the memristor-based neural network simulator. Instead of adopting the traditional algorithm-based technology, the PL method focuses on hardware implementation to enable privacy preserving ANN. It does not need to read the precise conductance of memristor before every write operation in weight-updating process and does not need to change the amplitude of the update-pulses each time in ANN, which avoids complex peripheral circuits. The 2-segment, 3-segment, and 4-segment models for 49 types of memristors with nonlinearity from (0/-0) to (6/-6) have been developed to investigate the effectiveness of the proposed method, the results indicate 34.22% to 47.73% average recognition accuracy improvement when the privacy budget ε ranges from 4 to 16. This concludes: 1) the proposed privacy-preserving ANN has an increasing accuracy, when the segment in PL model increases, the noise level decreases, and the nonlinearity decreases, respectively; 2) a PL model with more segments not only has a stronger immunity to nonlinearity but also gets higher and more stable accuracy; 3) the proposed method is proved to be effective when variations exist. Furthermore, in some cases, since the accuracy with privacy preservation gets even higher accuracy, the PL method is applied to provide more space and margin for privacy preservation. Finally, the PL method aims at mitigating the nonlinearity impact of memristor devices; therefore it can be adapted to many other memristor-based hardware systems. Consequently, the PL method is proved to be an effective technique that can prevent accuracy loss and increase privacy preservation space for privacy-preserving ANN.

6. MEMRISTOR BASED VARIATION ENABLED DIFFERENTIALLY PRIVATE LEARNING SYSTEMS FOR EDGE COMPUTING IN IOT

Edge AI (Artificial Intelligence) achieves real-time local data analysis for IoT systems, enabling low-power and high-speed operation, but comes with privacy-preserving requirements. Memristor based computing system is a promising solution for edge AI, but it needs a low-cost privacy protection mechanism due to limited resources. In this paper, we propose a Noise Distribution Normalization (NDN) method to add Gaussian distributed noise through hardware implementation, thereby achieving differential privacy in edge AI. Instead of using traditional algorithmic noise-insertion methods, we take advantage of inherent cycle-to-cycle variations of memristors during the weight-update process as the noise source, which does not incur extra software or hardware overhead. In one case study, the proposed method realizes ultra-low-cost DP-SGD (Differentially Private Stochastic Gradient Descent) for edge AI in IoT systems, achieving a 3.5% to 15.5% average recognition accuracy improvement under different noise levels, as compared with a baseline mechanism.

6.1. Introduction

Artificial intelligence involves many sensitive data, such as private, corporate, and national privacy information, which urgently needs effective privacy protection technologies. Differential privacy [64] is a popular solution that can provide a quantifiable indicator for privacy protection. From the perspective of differential privacy, machine learning algorithms could be designed to perform privacy preserving learning by introducing random noise [64]. However, the great problem is existing: 1) Machine learning training involves massive calculation, which introduces a large workload for hardware, especially for some real time applications such as online learning. 2) When learning systems involve privacy protection, noise injection is needed in each training

stage [44, 70]. Such a high-cost training application further challenges hardware technology as well as impedes the development of privacy protection in ANN.

To decrease the cost, many algorithm level solutions are proposed, for example, binary neural networks (BNN) [71] and neural network compression [72]. In addition, various hardware accelerators, such as Field Programmable Gate Arrays (FPGA), Application Specific Integrated Circuit (ASIC), Graphics Processing Units (GPUs), and Tensor Processing Unit (TPU), are developed. Nevertheless, the memory wall always exists [8] because of traditional Von Neumann architecture. Thus, emerging notions such as resistive computing, quantum computing, molecular computing, neuromorphic computing, memristor devices, quantum dots, and spin-wave devices, are explored. Among those technologies, memristor-based in-memory processing architecture is a promising candidate because memristors have desirable metrics and CMOS process compatibility [73]. The notion of memristor was predicted dozens of years ago and its physical realization was demonstrated by Hewlett-Packard Lab in 2008 [4, 5]. A memristor has a simple three-layer structure whose conductance value can be changed with the applied pulse, which can achieve multiple conductance states, small scale size (less than 2 nm), high switching speed (less than 1 ns), and low programming power consumption [8]. Because of these metrics, memristor-based crossbar architecture achieves fast dot-product operations [74], which have been applied to fabricate neural network circuits such as compression/filtering [2] and image classification [35, 36].

As for privacy protection in ANN, in [40], low-voltage static random-access memory (SRAM) chips are used to add failure as noise for training data, but the noise only follows a uniform distribution and does not satisfy the differential privacy theory for Gaussian and Laplace distribution. In [45, 46], in order to generate random numbers with high randomness, dedicated

random number generation modules, such as physical unclonable function (PUF) and random number generator, are designed. These modules are accurate but require significant additional circuitry. Also, all the above work is based on CMOS technology, and not compatible with memristor-based learning systems. In [47], an advanced learning system is implemented based on memristor arrays, but noise in training data for theoretical privacy guarantees use software methods to seriously complicate calculation.

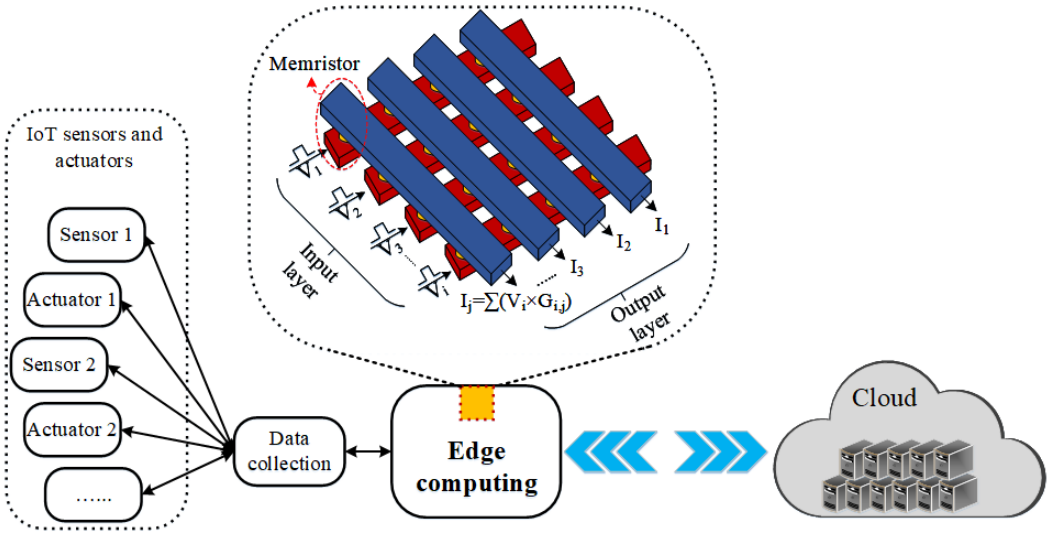


Figure 27. Hardware implementation of neural networks using memristor crossbar. V_i, $G_{i,j}$, and I_j represent the input signal in the i^{th} row, the conductance of the memristor in the j^{th} column and i^{th} row, and the output current that represents the dot product result of V and G, respectively.

What is more, as for memristors, some researchers point out the existence of non-ideal properties that includes non-linearity, device-to-device variation, cycle-to-cycle variation, maximum conductance variation, and minimum conductance variation [9, 47, 75, 76]. These non-ideal properties degrade the accuracy of a memristor-based learning system, however, such variations can be also considered as inherent resources for noise generation that is necessary for differentially private learning systems.

In this paper, we take the cycle-to-cycle variation as an advantage to realize hardware-based Gaussian noise injection. As a consequence, this paper explores differentially private

learning systems and proposes a hardware-based solution by utilizing the non-ideal properties of memristors. Also, optimization methods are proposed to improve the utility of neural networks. The proposed methods add Gaussian noise distribution to a system without adding computational complexity and introducing extra hardware, which greatly improves the power and computation efficiency.

6.2. Gaussian Distribution of Cycle-to-Cycle Variation

6.2.1. Mathematical Expression of Cycle-to-Cycle Variation

As discussed in Section III.D, when a memristor is incented by an input pulse, its conductance is changed not only by the designed value, $(G_{max} - G_{min})/N_{Level}$, but also by cycle-to-cycle variation, X. Here X is modeled as standard normal distribution, $N(0, \sigma)$, where the noise scale, σ, is determined by the pulse width [45, 46, 53-57]. When taking n input pulses as an example, the total conductance variation (G_{var}) generated in one memristor can be represented by Equation 15.

$$G_{var} = X_1 + X_2 + \cdots + X_n$$
$$X_1 \sim N(0, \sigma_1^2),\ X_2 \sim N(0, \sigma_2^2), \ldots,\ X_1 \sim N(0, \sigma_n^2) \tag{15}$$

where $X_1, X_2, \ldots\ldots, X_n$ are cycle-to-cycle variation variables that are introduced by input pulses from the 1st pulse to the nth pulse, respectively; and $\sigma_1, \sigma_2, \ldots, \sigma_n$ represent the standard deviation of the noise of each input. Because the widths of weight updating pulses are the same and applied independently, $X_1, X_2, \ldots\ldots, X_n$ are independent and identically distributed random variables, which indicates their means and variance are identical ($\sigma_1 = \sigma_2 = \ldots\ldots = \sigma_n = \sigma_{in}$). Then, based on probability theory [77, 78], the variable G_{var} follows a joint Gaussian distribution that can be represented as shown in Equation 16, with a probability density function (PDF) shown in Equation 17.

$$Gvar \sim N\ (0, n{\sigma_{in}}^2) \tag{16}$$

$$f(G_{var}) = \frac{1}{\sigma_{in}\sqrt{2n\pi}} e^{-\frac{1}{2n}\left(\frac{G_{var}}{\sigma_{in}}\right)^2} \tag{17}$$

6.2.2. Cycle-to-Cycle Variations of Positive and Negative Pulse Pairs

As discussed above, due to cycle-to-cycle variation, which is an inherent characteristic of memristors, even in a routine weight update process, model parameters of a memristor-based hardware system suffer from random noise addition. An input pulse introduces cycle-to-cycle variation by changing memristor conductance by (G_{max} - G_{min})/N_{Level}; and as shown in Figure 8, a positive pulse and negative pulse make conductance increase and decrease, respectively. Thus, when the number of such positive and negative pulses are equal, this is equivalent to adding cycle-to-cycle noise that follows a Gaussian distribution. For example, when we apply one positive pulse, the conductance of the memristor changes from G_0 to G_1, as shown in Equation 17. And after then applying one negative pulse, it changes from G_1 to G_2, as shown in Equation 18, where $X_1 \sim N\ (0, \sigma_1^2)$ and $X_2 \sim N\ (0, \sigma_2^2)$. Now add Equations 18 and 19, and let $X_1 + X_2 = G_{add}$, which results in Equation 20. Finally, let $\sigma_1^2 + \sigma_2^2 = \sigma_{add}^2$, which yields Equation 20.

$$G_1 = G_0 + ((G_{max} - G_{min})/N_{Level}) \times 1 + X_1 \tag{18}$$

$$G_2 = G_1 - ((G_{max} - G_{min})/N_{Level}) \times 1 + X_2 \tag{19}$$

$$G_2 = G_0 + X_1 + X_2 = G_0 + G_{add}$$

$$G_{add} \sim N(0, \sigma_{add}^2) \tag{20}$$

Based on probability theory [77, 78], X_1 and X_2 are independent random variables. Assuming that $\sigma_1^2 + \sigma_2^2 = \sigma_{add}^2$, then after applying a pair of positive and negative pulses, the effective noise introduced to the target memristor follows a joint Gaussian distribution, $N\ (0, \sigma_{add}^2)$. Hence, applying m positive/negative pulse pairs to a memristor injects noise that follows

the joint Gaussian distribution, $N(0, m\sigma_{add}^2)$, without requiring additional circuitry. With such a noise injection method, the scale of injected noise is decided by *m*, the number of positive/negative pulse pairs, and their pulse width.

It should be noted that due to the various types of memristors, when we apply this positive/negative pulse pair method, the intrinsic characteristics of the specific memristor need to be considered. For memristors discussed in [79-82], the conductance only changes when the amplitude of the input pulse is larger than the threshold voltage, and its duration is larger than the switching time, which is the necessary condition for noise injection. Therefore, for this kind of memristor, the minimum noise injection is constrained by its threshold voltage and switching time. Nevertheless, as for memristors discussed in [5, 83, 84], the conductance change and cycle-to-cycle variation exist whenever any input is applied, so, the injected noise can be set at a much smaller scale.

In this paper, we denote noise injection method by using positive/negative pulse pairs as the PN method.

6.3. Noise Injection in Accordance with DP

For a memristor in a learning system, when *n* pulses are applied (assume *n* is a positive integer), the injected noise, $G_{var,}$ regarded as system built-in noise, follows a joint distribution $N(0, n\sigma_{in}^2)$. However, for different memristors, the injected noise is not identical, and the independent Gaussian noise is impractical to be obtained in a trackable form. Hence, this causes difficulty in tracking the consumed privacy budget and evaluating utility loss. Therefore, this paper proposes a hardware-based noise distribution normalization (NDN) method to normalize introduced noise for all weights (conductance of memristors) in a learning model. The proposed method transfers non-beneficial random noise of cycle-to-cycle variations into a valuable measure

for privacy protection by using the proposed PN method, which improves the utility of the privacy-preserving neural network with an excellent privacy guarantee.

6.3.1. Noise Distribution Normalization (NDN) Method

As discussed in Section IV, for each memristor, G_{var} is built-in noise of an AI algorithm. G_{add} by PN method is adjustable noise for implementation of adjustable noise injection. Variables G_{var} and G_{add} are independent but not identical Gaussian random variables. Let, $G_{noise}=G_{var}+G_{add}$, then we get the following equation.

$$G_{noise} \sim N(0, n\sigma_{in}^2 + m\sigma_{add}^2) \tag{21}$$

We propose Noise Distribution Normalization (NDN) method to achieve DP protection by hardware implementation. To ensure that every gradient carries noise with the same distribution injected, first we clip the gradient into n_c, which is a constant value. Second, for one memristor, if the needed pulses $n >= n_c$, let $n = n_c$; otherwise, keep n value and let $m=(n_c - n)*{\sigma_{in}}^2/\sigma_{add}^2$. Given that the proposed method is hardware-based, where σ is related to actual characteristics of memristors, when we implement this method, σ_{add} can be tuned to a value that ensures m is an integer. Hence, when the needed pulses are $n \geq n_c$, $n\sigma_{in}^2 + m\sigma_{add}^2 = n_c\sigma_{in}^2$; and when the needed pulses are $n < n_c, n\sigma_{in}^2 + m\sigma_{add}^2 = n\sigma_{in}^2 + ((n_c - n) * {\sigma_{in}}^2/\sigma_{add}^2)) * \sigma_{add}^2 = n_c\sigma_{in}^2$. In so doing, for every memristor at each weight update, G_{noise} follows a $N(0, n_c\sigma_{in}^2)$ Gaussian distribution, with a PDF as shown in Equation 22.

$$f(G_{noise}) = \frac{1}{\sigma_{in}\sqrt{2\pi n_c}} e^{-\frac{1}{2n_c}\left(\frac{G_{noise}}{\sigma_{in}}\right)^2} \tag{22}$$

6.3.2. Privacy Analysis

To ensure that our design preserves the DP notion, we need to first guarantee that each gradient descent step is (ε, δ)-differentially private. For our crafted Gaussian distribution and PDF

with $G_{noise} \sim N\,(0,\, n_c\sigma_{in}^2)$ that we use to sample noise, by the standard definition in [64], one needs to ensure that $\sigma_{in}\sqrt{n_c} \geq \sqrt{2\log\,(1.25/\delta)}/\varepsilon$ and $\varepsilon < 1$. For the former inequality, it can be easily satisfied by adjusting the number of pulses, n_c, while the later inequality is selected artificially at runtime. In light of this, our Gaussian distribution can guarantee the DP notion for each iteration in Figure 8.

Next, we need to prove the overall process of running the DP-SGD algorithm in memristor crossbar for the DP notion, and theoretically evaluate how much privacy budget costs to preserve DP. First, since at each iteration we independently sample random noise from the identical Gaussian distribution $N\,(0,\, n_c\sigma_{in}^2)$ by applying the same number of pulses to the memristor crossbar, according to the composition theorem of (ε, δ)-differential privacy [64], the overall process still preserves the DP notion. Next, based on the privacy amplification theorem [85], for the process of random sampling with probability *L/N*, as in Figure 8, each iteration is technically $O((\varepsilon L/N), \delta L/N)$-differentially private. Then, the strong composition theorem implies that, for a sufficiently large number of iterations, T (i.e., we expect $T \gg N/L$ and each sample is examined multiple times), the overall budget cost is $\varepsilon_{tot} = \Omega((L/N)\sqrt{T\log(1/\delta)\log\,(T/\delta)}/\sigma_{in}\sqrt{n_c})$. It is worth noting that [44] developed a moment accountant technique to obtain a much tighter bound on the accumulated privacy budget, which allows for using a smaller variance value in the Gaussian distribution. Here, our intention is only to prove and showcase the preservation of DP and its estimated cost, despite loose privacy loss. Interested readers are referred to [44] for the details on the moment accountant analysis.

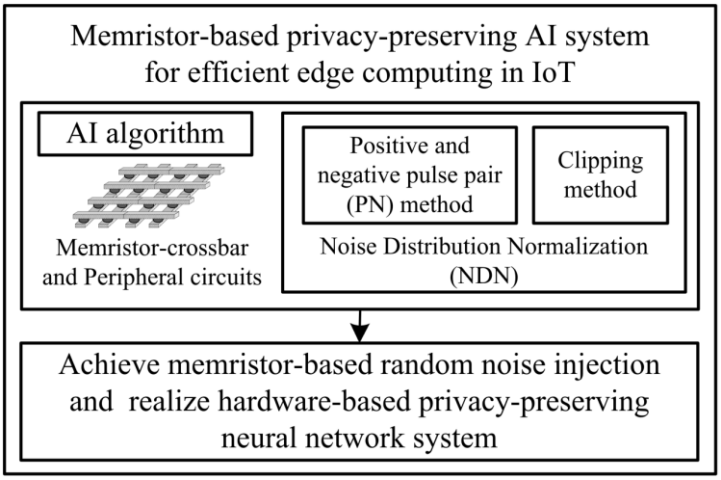


Figure 28. Outline of the proposed methods.

6.4. Case Study and Discussions

As discussed in Section V, the proposed memristor-based NDN method can effectively realize DP by injecting normalized random noise to edge AI in IoT systems. In this Section, based on a hardware machine learning platform that consists of memristor crossbar array and peripheral circuits, we perform a case study that implements the DP-SGD algorithm by noise distribution normalization (NDN).

6.4.1. Implementation of DP-SGD via Hardware in Edge AI

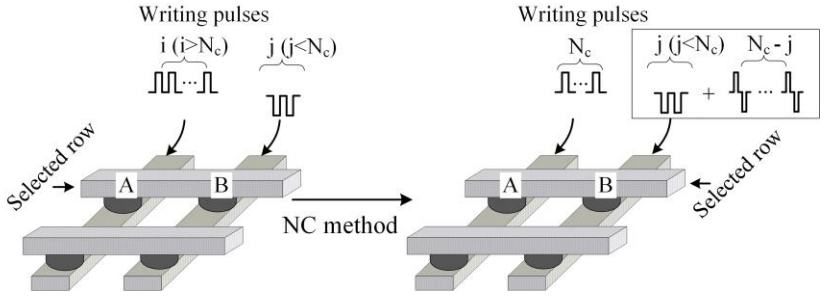
As illustrated in Figure 8, two modifications (Clip Gradient and Add Noise) are needed to ensure that the proposed stochastic gradient descent (SGD) follows a DP algorithm. For the first modification, in order to constrain how much each individual training sample can influence the resulting gradient computation (model parameters), the sensitivity of each gradient needs to be bounded. For the second modification, it is necessary to randomize the behavior of the algorithm to make it statistically impossible to identify whether a particular training sample is included in the training dataset, which can be achieved by adding random noise to the clipped gradients.

For the first modification, Figure 8 shows that the gradient clipping process needs to calculate the L_2 norm of the gradient matrix. These processes inevitably increase the computational

load of the system. We propose a clipping method that is the first step of the NDN method to meet the requirement of clipping without matrix calculation. The clipping operation sets an upper boundary on gradients to bound the influence of each individual example on gradients. A smaller upper boundary has a stronger limitation of the gradient. When the NDN clip boundary is small enough, the L_2 norm of the gradient is always less than 1. In this circumstance, L_2 norm computing is simplified because 1 is always chosen as the maximum value in the first modification according to the DP-SGD shown in Figure 8. As shown in Figure 28, in our hardware implementation, this method uses simple hardware units - comparators to compare gradient value with a reference gradient value. When the gradient is clipped, the maximum number of weight update pulses is fixed. Accordingly, the system saves the cost of matrix calculation and also ensures that the degree of each training sample's impact on model parameters is bounded.

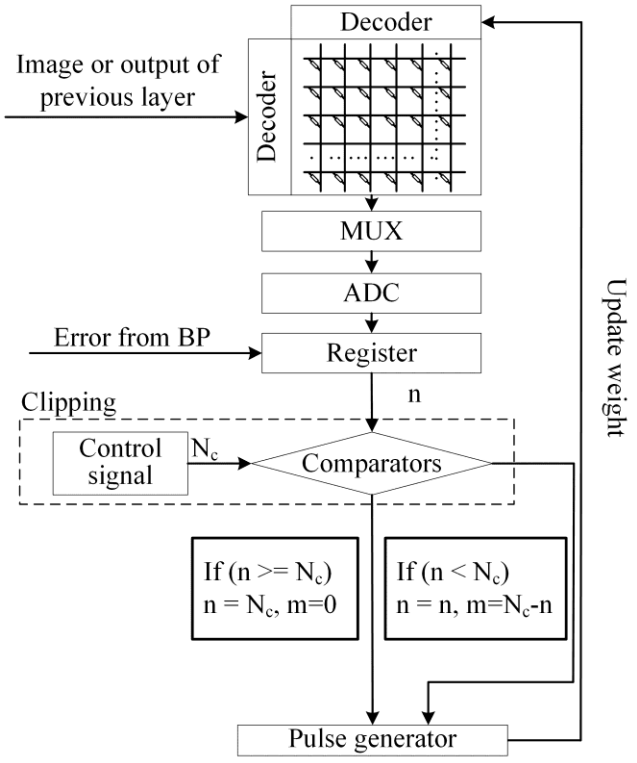
The second modification can be implemented by the PN method, which is the second step of the NDN method. The PN method adds random noise by applying extra input pulse pairs to memristors. As discussed in Section IV, in each step, when extra PN pairs of input pulses are added to each memristor, the amount of the designated change in conductance caused by such positive and negative pulses counteract each other, such that only random noise is added to the memristor conductance. Since the weight increase and decrease need different programming voltage polarities, the weight update process (writing process for the model parameter) requires two steps with positive and negative voltages, respectively. Figure 28b shows the hardware implementation flow of the NDN method. Hence, when using hardware that combines a memristor array-based learning circuit along with the NDN method, the DP-SGD can be implemented using the existing hardware with minimal additions.

To verify our proposed methods for edge AI, we adopt the neural network hardware platform, NeuroSim+ [48]. This is a memristor-based circuit model of neuro-inspired architectures to emulate the circuit behavior of an online learning recognition scenario with Modified National Institute of Standards and Technology (MNIST) [21] dataset. The neural network topology of this simulator includes input layer, hidden layer, and out-put layer, with 400 neurons, 100 neurons, and 10 neurons, respectively. The simulator emulates hardware to train the network with images randomly chosen from the training dataset MNIST, which includes 60,000 images, and classify the testing dataset with 10,000 images. The training process has two parts, feed-forward propagation and backpropagation, which includes weighted sum operation, neuron activation operation, recognition, and deviation calculation. The deviations are used to update the conductance of memristors using identical positive input pulses or identical negative input pulses. We integrate our proposed NDN method into this simulator to train privacy-preserving multi-layer neural networks, such that we can only use hardware that consists of memristor array and peripheral circuits to realize DP-SGD behavior.



(a)

Figure 29. Workflow of the NDN method, where *n* represents the number of pulses that are used to update a weight and *m* represents the number of positive and negative pulse pairs. (a) Example: after applying the NDN method, for memristor A $n=N_c$, $m=0$; for memristor B, $n=j$, $m=N_{c-j}$. (b) Hardware implementation flow of the Noise Distribution Normalization method.



(b)

Figure 29. Workflow of the NDN method, where *n* represents the number of pulses that are used to update a weight and *m* represents the number of positive and negative pulse pairs (continued). (a) Example: after applying the NDN method, for memristor A $n=N_c$, $m=0$; for memristor B, $n=j$, $m=N_c-j$. (b) Hardware implementation flow of the Noise Distribution Normalization method.

6.4.2. Results of Clipping NDN

To explore the effect of the Clipping step of NDN, the Clipping method is applied to a fully connected neural network without the PN method and without cycle-to-cycle variation. As discussed in Section VI.A, the clipping operation limits the scale of gradient, and a smaller clip boundary has a larger limitation of the gradient. Also, as each gradient value is clipped, the number of pulses is also clipped, and the influence of each image is limited. In our simulation, we use a clip boundary to clip the gradient of each weight. Then, the number of pulses for each weight is obtained, where the number of Pulse = rounding (clipping (gradient) * learning_rate * N_{Level}). The rounding operation converts the value to its nearest integer number. In our case, when the clip boundary is less than 0.2, the L2 norm of the gradient is always smaller than 1. Therefore, the

clipping method saves the matrix calculation cost of DP-SGD based hardware systems. Figure 29 shows the recognition accuracy of MNIST handwriting digits as the clip boundary value changes. These results indicate that as the clip boundary decreases, the recognition accuracy stays at a high level, which concludes that the clipping operation does not degrade performance of the three-layer neural network based on the DP-SGD algorithm.

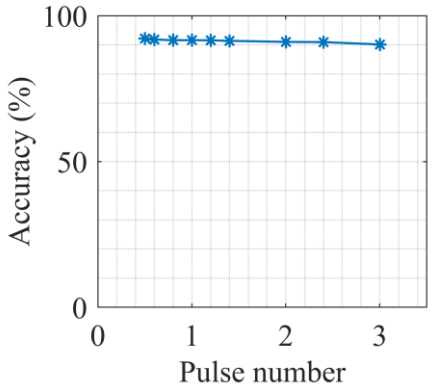


Figure 30. Recognition accuracy of MNIST under various clip boundaries.

6.4.2. Results of NDN Method

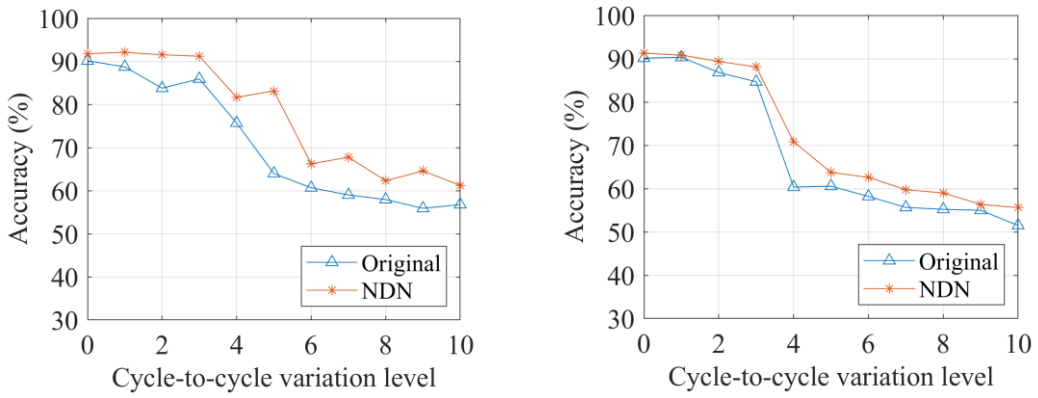
Since many types of memristors exist, the proposed methods are explored with various configurations, including the twelve noise levels shown in Table 9. Figure 30 shows recognition accuracy of MNIST handwriting digits under various noise levels with and without NDN method. The one without NDN method is named Original, and it does not consider cycle-to-cycle variation as noise injection, but instead adds Gaussian noise via software. As shown in Figure 30, under Nc=1 and Nc=2, the average recognition accuracy of these ten variation circumstances with NDN method is improved by 7.4% and 3.5%, respectively, as compared with the Original case. Figure 30 also provides a recognition accuracy comparison, but with different Nc and different variation levels (levels from 2 to 8). It shows that the NDN method has 15.5% higher accuracy on average as compared to the Original method. This is because for memristor-based learning system in the Original case, cycle-to-cycle variation still exists, in addition to the noise added via software,

which leads to prediction accuracy loss. Such loss will be larger than the case with NDN method that only considers the inherent cycle-to-cycle variation without additional noise injection via software. These results support the proposed NDN method as an effective optimization method that can improve the utility of a memristor-based differentially private learning model.

Table 9. Sigma Parameter of Cycle-to-Cycle Variation*

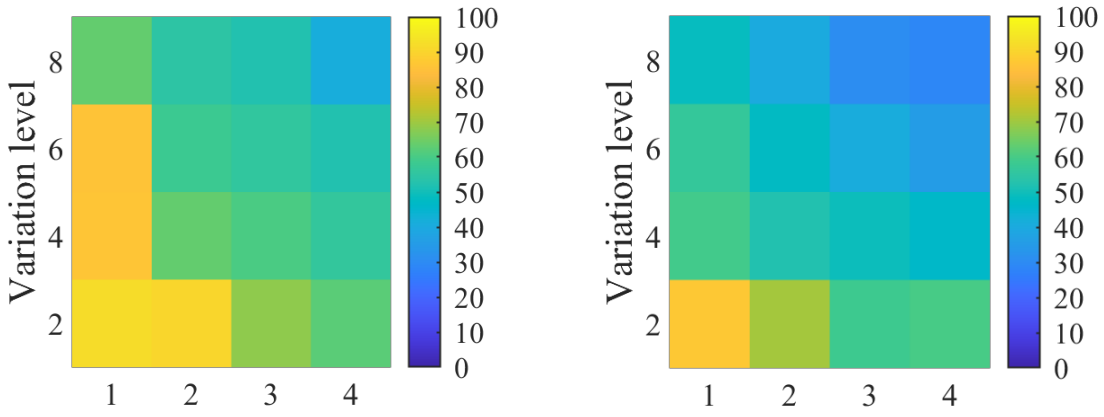
Level	1	2	3	4	5	6
$\sigma/(G_{max}\text{-}G_{min})$	1%	2%	3%	4%	5%	6%
Level	7	8	9	10	11	12
$\sigma/(G_{max}\text{-}G_{min})$	7%	8%	9%	10%	11%	12%

*The σ value of Gaussian Distribution for Cycle-to-Cycle variation is represented by the percentage of memristor's conductance range [24], [29].



(a) N_c=1, (b) N_c=2

Figure 31. Recognition accuracy of MNIST handwriting digits under various noise levels.



(a) NDN (b) Original

Figure 32. Recognition accuracy of MNIST handwriting digits under various noise levels and different number of pulse pairs: (a) using NDN method, where the x-axis represents the number of pulse pairs, N_c; (b) Original case, where the x-axis has the same noise scale as (a).

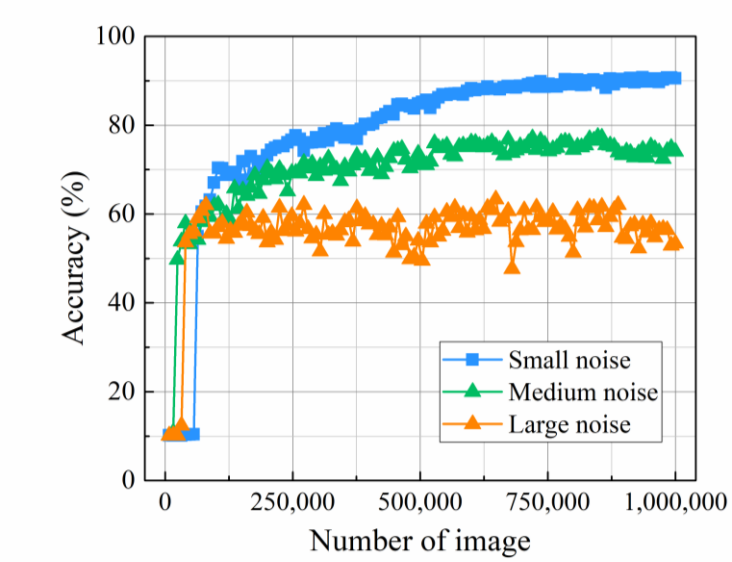


Figure 33. Recognition accuracy of training process using NDN method under three noise levels, where σ for small noise, medium noise, and large noise equals 3%, 6%, and 12%, respectively.

6.4.3. Comparison with Existing Work

As listed in Table 10, as compared with the state-of-art, [40, 46, 47], instead of implementing the DP-SGD algorithm for privacy preservation by a traditional computing system, the proposed NDN method adds Gaussian noise for memristor-based hardware using inherent cycle-to-cycle variation. Thus, the memristor-based machine learning system does not require an

additional random noise generator. Also, the scale of injected noise can be adjusted by changing the number of PN pulse pairs. For the DP-SGD algorithm, the Clipping method of NDN limits the impact of each training data on model parameters and saves the cost of the L_2 norm matrix calculation.

Table 10. The Comparison with State-of-the-Art.

	[47]	[40]	[46]	This work
Realize hardware-based privacy protection	×	√	×	√
Without extra random noise generator	×	√	×	√
Without always adding random noise for each cell	×	×	×	√
Compatible with memristor-based hardware	√	×	√	√

6.4.4. Nonlinearity

In general, the amount of conductance change of memristors sometimes is different as the number of pulses increases, which is attributed to the nonlinearity (NL) of the weight modulation. In other words, every pulse results in a different response in the weight modulation depending on the current weight state [7] when we apply a pair of positive and negative pulses to a memristor with the NL consideration, as shown in Figure 39, where $G_X = G_{X1} - G_{X2}$, Equation 20 can be rewritten as follows:

$$G_2 = G_0 + G_X + G_{add} \tag{23}$$

where $G_0 + G_X$ has the same conductance range as G_0, which is from G_{min} to G_{max}. When we map the memristors' conductance to the gradient of DP-SGD, both G_0 in Equation 20 and $G_0 + G_X$ in Equation 23 are mapped to the same range, which is from 0 to 1. Thus, when we consider the NL effect of memristors, the global sensitivity of gradient remains unchanged, and so will the noise scale of G_{add} (i.e., variance of the Gaussian distribution). Therefore, the NDN method still conforms to the DP theory.

We conduct experiments to explore the NDN method considering the NL of memristors. We adopt the NL definition from [86], where NL ranges from 0 to 1. As shown in Table 11, with NL ranging from 0.05 – 0.25, recognition accuracy with NDN is still 9.3% on average higher than the Original case without NDN. This shows the effectiveness of the proposed method, where the NL is being used as an advantage instead of degrading accuracy, as already discussed in Section VI.B.2. Although the NL property does degrade the performance of memristor based learning system, a memristor-based learning system can achieve an acceptable performance with the NL and the proposed NDN method. Moreover, researchers have proposed memristors with a small NL [18] as well as methods to solve the NL issue [41, 76]. Thus, the proposed method is still effective even with the NL property.

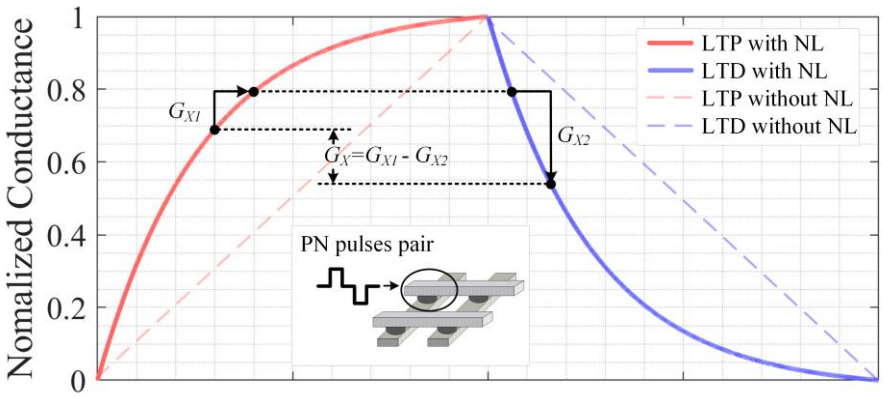


Figure 34. Nonlinearity effect on conductance modulation of memristors.

Table 11. Recognition Accuracy with NDN Method and Nonlinearity.

Nonlinearity (NL)	0.00	0.05	0.15	0.25
NDN	92.4%	90.2%	77.5%	59.3%
Original	91.0%	87.3%	67.2%	44.5%

6.4.5. Scalability and Endurance

Many applications employing a neural architecture use memristors in the edge, where our proposed method can be adopted, including for example, face classification with artificial neural network (ANN) structure [87, 88], handwritten digits classification [89] and image processing [2] with convolutional neural network (CNN) structure, memristor-based edge detection [90], and pattern recognition with recurrent neural network (RNN) [91]. Our proposed method is generic and can be applied to any memristor-based learning system. However, the number of neurons and layers in deep learning neural networks may cause scalability issue for a memristor-based array. [92] solves this issue by using a chip-level hierarchical architecture that divides large arrays into groups of synaptic sub-arrays and connects each sub-array using an H-tree structure. To further explore network scalability, we simulated the learning tasks introduced in Section VI.B using the various network structures listed in Table 12, which shows that power increases with number of neurons, but accuracy also increases; hence, there is a tradeoff between power and accuracy. Note that memristor array power is calculated based on wire resistance, reading, and the weight update writing process.

Another challenge for memristor crossbar arrays is the sneak path issue, which severely degrades read sensing margin [93]. One solution is to increase the minimum conductance value, but this degrades ON/OFF ratio of memristors. Our experiments utilize a one-transistor one-resistor (1T1R) array to avoid sneak path current problems [48].

Table 12. Total Power of Memristor-Based Array in Learning Device.

Number of Hidden Layer Neurons	50	100	150	200
Power (10^{-7} J)	3.2	4.3	5.3	7.1
Recognition Accuracy (%)	90.2	91.1	92.2	92.8

The failure rate, endurance, and aging issues also impact the performance of memristor-based edge systems. Typical memristor failure rate is less than 10% [94]. As shown in Table 13, we have conducted learning tasks with up to 10% failure rate, while recognition accuracy shows no obvious degradation. This indicates that neuromorphic architectures are more robust to variations because in the learning process, weights updates frequently in each epoch to compensate for mismatch resulting from variations. Memristors have high endurance (120 billion cycles) and retention (10 years) even when they undergo high frequency writing and reading in learning systems [86], which qualifies them for most edge computing IoT applications.

Table 13. Recognition Accuracy with Various Failure Rates.

Failure Rate	0%	5%	10%
Recognition Accuracy	92.4%	90.7%	90.6%

Additionally, the conductance range of memristors may deviate from the original state over time. As shown in Table 14, the recognition accuracy decreases by 27.74% when conductance drifting is 30%, but it does not show a significant decrease when it is 10% or 20%.

Table 14. Recognition Accuracy with Conductance Drifting.

Maximum Conductance Drift	0%	10%	20%	30%
Recognition Accuracy	92.4%	92.0%	91.4%	63.7%

6.5. Conclusions

To meet the high-speed, low-power, and low computing-cost requirements of edge computing in IoT systems, we propose a universal memristor-based method that can be used to realize a privacy-preserving learning system. The proposed Noise Distribution Normalization (NDN) method consists of a Positive/Negative Pulse Pair (PN) method and a Clipping method. The PN method can generate adjustable Gaussian noise based on cycle-to-cycle variation of

memristors, without extra hardware or a random noise generator, making it possible to meet the noise-injection requirement of the Differential Privacy (DP) mechanism. And the Clipping method that uses comparator units can normalize the introduced noise from the PN method.

In the case study of a memristor-based neural network hardware platform, we implement the DP-SGD algorithm via hardware-based NDN method, and at the same time avoid the L_2 norm calculation of gradient matrices, thereby reducing computational cost. Experiment results indicate a 3.5% to 15.5% average recognition accuracy improvement using the proposed NDN method and a 9.3% average improvement when the nonlinearity of memristors is considered, as compared to the Original case that adds noise via software. Also, the scalability and endurance for the proposed system are considered. Consequently, the proposed method is an effective technique that provides a low-cost hardware solution for the notion of DP in memristor-based learning systems.

7. CONCLUSIONS

Analog architectures integrate memory and computation and perform computation via certain analog operations, which has potential to achieve significant power and performance improvement compared to conventional von-Neumann architectures. This research focuses on hardware techniques for improving performance of memristor-based mix signal learning system.

First, a linear optimization method is proposed to address the accuracy degradation by optimizing the performance of memristor in the weight updating processes. Instead of complying with the traditional hardware and algorithm, it calculates the update parameters along a piecewise line by using different input pulses. The proposed method can mitigate the nonlinear problem of memristor without prereading the precise current conductance each time, thereby avoiding complex peripheral circuits. The PL method provides flexible configurations so that it can achieve 57.6% recognition accuracy improvement in a low-cost situation, but if a higher accuracy is desirable, additional cost can be taken to realize 67.02% accuracy improvement. As compared with the state-of-art, the PL method avoids complex peripheral circuits, and with more segments, the nonlinearity impact of memristor to the neuromorphic hardware is almost negligible.

Second, the proposed PL method is applied for memristor based privacy preserving neural network. We use PL method to alleviate nonlinearity of memristors for weight updating in ANN to counteract recognition degradation due to noise injection. Therefore, privacy-preserving ANN provides more space for randomizing noisy data, which ensures that the publicly visible information do not change much if one individual in the dataset changes. The 2-segment, 3-segment, and 4-segment models for 49 types of memristors with nonlinearity from (0/-0) to (6/-6) have been developed to investigate the effectiveness of the proposed method, the results indicate

34.22% to 47.73% average recognition accuracy improvement when the privacy budget ε ranges from 4 to 16.

Third, system A hardware solution of memristor based in-memory computing is proposed to enable energy efficient privacy preserving technology in ANN. The proposed solution breaks the limitations of traditional software-based noise-adding mechanisms of DP. We utilize inherent cycle-to-cycle variations of memristors and apply the proposed variation-based pulse pair method during the weight update process. The proposed method can generate adjustable Gaussian noise based on cycle-to-cycle variation of memristors, without extra hardware or a random noise generator, making it possible to meet the noise-injection requirement of the Differential Privacy (DP) mechanism. In the case study of a memristor-based neural network hardware platform, we implement the DP-SGD algorithm via hardware-based method. Experiment results indicate a 3.5% to 15.5% average recognition accuracy improvement using the proposed method and a 9.3% average improvement when the nonlinearity of memristors is considered. Consequently, the proposed method is an effective technique that provides a low-cost hardware solution for the notion of DP in memristor-based learning systems.

REFERENCES

[1] L. B. Kish, “End of Moore's law: thermal (noise) death of integration in micro and nano electronics,” *Physics Letters A*, vol. 305, no. 3-4, pp. 144-149, 2002.

[2] 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.

[3] 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.

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

[5] 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.

[6] B. Rajendran, Y. Liu, J.-s. Seo, K. Gopalakrishnan, L. Chang, D. J. Friedman, and M. B. Ritter, “Specifications of nanoscale devices and circuits for neuromorphic computational systems,” *IEEE Transactions on Electron Devices*, vol. 60, no. 1, pp. 246-253, 2012.

[7] 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. 1-7, 2018.

[8] 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.

[9] 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.

[10] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis." pp. 265-284.

[11] R. Shokri, and V. Shmatikov, "Privacy-preserving deep learning." pp. 1310-1321.

[12] B. Govoreanu, G. Kar, Y. Chen, V. Paraschiv, S. Kubicek, A. Fantini, I. Radu, L. Goux, S. Clima, and R. Degraeve, "10× 10nm 2 Hf/HfO x crossbar resistive RAM with excellent performance, reliability and low-energy operation." pp. 31.6. 1-31.6. 4.

[13] 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.

[14] B. J. Choi, A. C. Torrezan, J. P. Strachan, P. Kotula, A. Lohn, M. J. Marinella, Z. Li, R. S. Williams, and J. J. Yang, “High‐speed and low‐energy nitride memristors,” *Advanced Functional Materials*, vol. 26, no. 29, pp. 5290-5296, 2016.

[15] K.-H. Kim, S. Hyun Jo, S. Gaba, and W. Lu, “Nanoscale resistive memory with intrinsic diode characteristics and long endurance,” *Applied Physics Letters*, vol. 96, no. 5, pp. 053106, 2010.

[16] J. Zhou, F. Cai, Q. Wang, B. Chen, S. Gaba, and W. D. Lu, “Very low-programming-current RRAM with self-rectifying characteristics,” *IEEE Electron Device Letters*, vol. 37, no. 4, pp. 404-407, 2016.

[17] 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.

[18] L. Gao, I.-T. Wang, P.-Y. Chen, S. Vrudhula, J.-s. Seo, Y. Cao, T.-H. Hou, and S. Yu, “Fully parallel write/read in resistive synaptic array for accelerating on-chip learning,” *Nanotechnology,* vol. 26, no. 45, pp. 455204, 2015.

[19] S. Park, A. Sheri, J. Kim, J. Noh, J. Jang, M. Jeon, B. Lee, B. Lee, B. Lee, and H.-J. Hwang, "Neuromorphic speech systems using advanced ReRAM-based synapse." pp. 25.6. 1-25.6. 4.

[20] J. Woo, K. Moon, J. Song, S. Lee, M. Kwak, J. Park, and H. Hwang, “Improved synaptic behavior under identical pulses using AlO x/HfO 2 bilayer RRAM array for neuromorphic systems,” *IEEE Electron Device Letters,* vol. 37, no. 8, pp. 994-997, 2016.

[21] 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.

[22] G. W. Burr, R. M. Shelby, S. Sidler, C. Di Nolfo, J. Jang, I. Boybat, R. S. Shenoy, P. Narayanan, K. Virwani, and E. U. Giacometti, “Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element,” *IEEE Transactions on Electron Devices,* vol. 62, no. 11, pp. 3498-3507, 2015.

[23] G. Burr, P. Narayanan, R. Shelby, S. Sidler, I. Boybat, C. di Nolfo, and Y. Leblebici, "Large-scale neural networks implemented with non-volatile memory as the synaptic weight element: Comparative performance analysis (accuracy, speed, and power)." pp. 4.4. 1-4.4. 4.

[24] S. Sidler, I. Boybat, R. M. Shelby, P. Narayanan, J. Jang, A. Fumarola, K. Moon, Y. Leblebici, H. Hwang, and G. W. Burr, "Large-scale neural networks implemented with non-volatile memory as the synaptic weight element: Impact of conductance response." pp. 440-443.

[25] M. Suri, O. Bichler, D. Querlioz, O. Cueto, L. Perniola, V. Sousa, D. Vuillaume, C. Gamrat, and B. DeSalvo, "Phase change memory as synapse for ultra-dense neuromorphic systems: Application to complex visual pattern extraction." pp. 4.4. 1-4.4. 4.

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

[27] Z. Wang, M. Yin, T. Zhang, Y. Cai, Y. Wang, Y. Yang, and R. Huang, “Engineering incremental resistive switching in TaO x based memristors for brain-inspired computing,” *Nanoscale,* vol. 8, no. 29, pp. 14015-14022, 2016.

[28] J.-H. Bae, S. Lim, B.-G. Park, and J.-H. Lee, “High-density and near-linear synaptic device based on a reconfigurable gated Schottky diode,” *IEEE Electron Device Letters,* vol. 38, no. 8, pp. 1153-1156, 2017.

[29] B. Chen, J. Kang, P. Huang, Y. Deng, B. Gao, R. Liu, F. Zhang, L. Liu, X. Liu, and X. Tran, "Multi-level resistive switching characteristics correlated with microscopic filament geometry in TMO-RRAM." pp. 1-2.

[30] X. Xu, H. Lv, Y. Li, H. Liu, M. Wang, Q. Liu, S. Long, and M. Liu, “Degradation of gate voltage controlled multilevel storage in one transistor one resistor electrochemical metallization cell,” *IEEE Electron Device Letters,* vol. 36, no. 6, pp. 555-557, 2015.

[31] H. Kim, M. P. Sah, C. Yang, and L. O. Chua, "Memristor-based multilevel memory." pp. 1-6.
[32] A. Bagheri-Soulla, and M. Ghaznavi-Ghoushchi, “Memristor based circuit design using charge and attached capacitor,” *Microelectronics journal*, vol. 55, pp. 53-63, 2016.
[33] A. Bagheri-Soulla, and M. Ghaznavi-Ghoushchi, “A high-precision time-domain RRAM state control approach,” *Microelectronics journal*, vol. 74, pp. 94-105, 2018.
[34] I.-T. Wang, C.-C. Chang, L.-W. Chiu, T. Chou, and T.-H. Hou, “3D Ta/TaO x/TiO2/Ti synaptic array and linearity tuning of weight update for hardware neural network applications,” *Nanotechnology*, vol. 27, no. 36, pp. 365204, 2016.
[35] C. Li, D. Belkin, Y. Li, P. Yan, M. Hu, N. Ge, H. Jiang, E. Montgomery, P. Lin, and Z. Wang, “Efficient and self-adaptive in-situ learning in multilayer memristor neural networks,” *Nature communications*, vol. 9, no. 1, pp. 1-8, 2018.
[36] M. Hu, C. E. Graves, C. Li, Y. Li, N. Ge, E. Montgomery, N. Davila, H. Jiang, R. S. Williams, and J. J. Yang, “Memristor‐based analog computation and neural network classification with a dot product engine,” *Advanced Materials*, vol. 30, no. 9, pp. 1705914, 2018.
[37] M. Al-Rubaie, and J. M. Chang, “Privacy-preserving machine learning: Threats and solutions,” *IEEE Security & Privacy*, vol. 17, no. 2, pp. 49-58, 2019.
[38] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson, “Scalable private learning with pate,” *arXiv preprint arXiv:1802.08908*, 2018.
[39] 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.
[40] L. Yang, and B. Murmann, "Approximate SRAM for energy-efficient, privacy-preserving convolutional neural networks." pp. 689-694.
[41] C. Xu, J. Ren, L. She, Y. Zhang, Z. Qin, and K. Ren, “EdgeSanitizer: Locally differentially private deep inference at the edge for mobile data analytics,” *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5140-5151, 2019.
[42] P. C. M. Arachchige, P. Bertok, I. Khalil, D. Liu, S. Camtepe, and M. Atiquzzaman, “Local differential privacy for deep learning,” *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5827-5842, 2019.
[43] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, “Semi-supervised knowledge transfer for deep learning from private training data,” *arXiv preprint arXiv:1610.05755*, 2016.
[44] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy." pp. 308-318.
[45] 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.
[46] M. Uddin, M. S. Hasan, and G. S. Rose, "On the theoretical analysis of memristor based true random number generator." pp. 21-26.
[47] 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*, vol. 27, no. 12, pp. 2745-2754, 2019.
[48] 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.

[49] F. Alibart, L. Gao, B. D. Hoskins, and D. B. Strukov, “High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm,” *Nanotechnology,* vol. 23, no. 7, pp. 075201, 2012.
[50] S. Choi, P. Sheridan, and W. D. Lu, “Data clustering using memristor networks,” *Scientific reports,* vol. 5, no. 1, pp. 1-10, 2015.
[51] L. Goux, A. Fantini, R. Degraeve, N. Raghavan, R. Nigon, S. Strangio, G. Kar, D. Wouters, Y. Y. Chen, and M. Komura, "Understanding of the intrinsic characteristics and memory trade-offs of sub-μA filamentary RRAM operation." pp. T162-T163.
[52] C. Baeumer, R. Valenta, C. Schmitz, A. Locatelli, T. O. Mentes, 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.
[53] 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.
[54] T. Tuma, A. Pantazi, M. Le Gallo, A. Sebastian, and E. Eleftheriou, “Stochastic phase-change neurons,” *Nature nanotechnology,* vol. 11, no. 8, pp. 693-699, 2016.
[55] Y. Pang, B. Gao, D. Wu, S. Yi, Q. Liu, W.-H. Chen, T.-W. Chang, W.-E. Lin, X. Sun, and S. Yu, "25.2 A reconfigurable RRAM physically unclonable function utilizing post-process randomness source with< 6× 10− 6 native bit error rate." pp. 402-404.
[56] Y. Gao, D. C. Ranasinghe, S. F. Al-Sarawi, O. Kavehei, and D. Abbott, “Memristive crypto primitive for building highly secure physical unclonable functions,” *Scientific reports,* vol. 5, no. 1, pp. 1-14, 2015.
[57] 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.
[58] Y. Gao, D. C. Ranasinghe, S. F. Al-Sarawi, O. Kavehei, and D. Abbott, “Emerging physical unclonable functions with nanotechnology,” *IEEE access,* vol. 4, pp. 61-80, 2016.
[59] Y. Gao, C. Jin, J. Kim, H. Nili, X. Xu, W. Burleson, O. Kavehei, M. van Dijk, D. C. Ranasinghe, and U. Rührmair, “Efficient erasable PUFs from programmable logic and memristors,” *Cryptology ePrint Archive,* 2018.
[60] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: information leakage from collaborative deep learning." pp. 603-618.
[61] K. Liu, C. Giannella, and H. Kargupta, "A survey of attack techniques on privacy-preserving data perturbation methods," *Privacy-Preserving Data Mining,* pp. 359-381: Springer, 2008.
[62] E. Hesamifard, H. Takabi, and M. Ghasemi, “Cryptodl: Deep neural networks over encrypted data,” *arXiv preprint arXiv:1711.05189,* 2017.
[63] C. Briggs, Z. Fan, and P. Andras, “A Review of Privacy Preserving Federated Learning for Private IoT Analytics,” *arXiv preprint arXiv:2004.11794,* 2020.
[64] C. Dwork, and A. Roth, “The algorithmic foundations of differential privacy,” *Found. Trends Theor. Comput. Sci.,* vol. 9, no. 3-4, pp. 211-407, 2014.
[65] Y. Kim, Y. Zhang, and P. Li, “A reconfigurable digital neuromorphic processor with memristive synaptic crossbar for cognitive computing,” *ACM Journal on Emerging Technologies in Computing Systems (JETC),* vol. 11, no. 4, pp. 1-25, 2015.
[66] A. Rakhlin, O. Shamir, and K. Sridharan, “Making gradient descent optimal for strongly convex stochastic optimization,” *arXiv preprint arXiv:1109.5647,* 2011.

[67] A. Bennar, and J.-M. Monnez, “Almost sure convergence of a stochastic approximation process in a convex set,” *International Journal of Apllied Mathematics,* vol. 20, no. 5, pp. 713-722, 2007.
[68] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, “Robust stochastic approximation approach to stochastic programming,” *SIAM Journal on optimization,* vol. 19, no. 4, pp. 1574-1609, 2009.
[69] D. Ielmini, “Modeling the universal set/reset characteristics of bipolar RRAM by field-and temperature-driven filament growth,” *IEEE Transactions on Electron Devices,* vol. 58, no. 12, pp. 4309-4317, 2011.
[70] C. Zhang, P. Patras, and H. Haddadi, “Deep learning in mobile and wireless networking: A survey,” *IEEE Communications surveys & tutorials,* vol. 21, no. 3, pp. 2224-2287, 2019.
[71] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks." pp. 525-542.
[72] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” *arXiv preprint arXiv:1510.00149,* 2015.
[73] F. Cai, J. M. Correll, S. H. Lee, Y. Lim, V. Bothra, Z. Zhang, M. P. Flynn, and W. D. Lu, “A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations,” *Nature Electronics,* vol. 2, no. 7, pp. 290-299, 2019.
[74] Z. Wang, C. Li, P. Lin, M. Rao, Y. Nie, W. Song, Q. Qiu, Y. Li, P. Yan, and J. P. Strachan, “In situ training of feed-forward and recurrent convolutional memristor networks,” *Nature Machine Intelligence,* vol. 1, no. 9, pp. 434-442, 2019.
[75] J. Fu, Z. Liao, N. Gong, and J. Wang, "Linear optimization for memristive device in neuromorphic hardware." pp. 453-458.
[76] 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,* vol. 9, no. 2, pp. 377-387, 2019.
[77] A. Cedilnik, K. Kosmelj, and A. Blejec, “The distribution of the ratio of jointly normal variables,” *Metodoloski zvezki,* vol. 1, no. 1, pp. 99, 2004.
[78] D. S. Lemons, "An introduction to stochastic processes in physics," American Association of Physics Teachers, 2003.
[79] S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, “MAGIC—Memristor-aided logic,” *IEEE Transactions on Circuits and Systems II: Express Briefs,* vol. 61, no. 11, pp. 895-899, 2014.
[80] 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.
[81] S. Kvatinsky, A. Kolodny, U. C. Weiser, and E. G. Friedman, "Memristor-based IMPLY logic design procedure." pp. 142-147.
[82] J. Rajendran, H. Manem, R. Karri, and G. S. Rose, "Memristor based programmable threshold logic array." pp. 5-10.
[83] Y. N. Joglekar, and S. J. Wolf, “The elusive memristor: properties of basic electrical circuits,” *European Journal of physics,* vol. 30, no. 4, pp. 661, 2009.
[84] Z. Biolek, D. Biolek, and V. Biolkova, “SPICE Model of Memristor with Nonlinear Dopant Drift,” *Radioengineering,* vol. 18, no. 2, 2009.

[85] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, “What can we learn privately?,” *SIAM Journal on Computing,* vol. 40, no. 3, pp. 793-826, 2011.
[86] S. Pi, C. Li, H. Jiang, W. Xia, H. Xin, J. J. Yang, and Q. Xia, “Memristor crossbar arrays with 6-nm half-pitch and 2-nm critical dimension,” *Nature nanotechnology,* vol. 14, no. 1, pp. 35-39, 2019.
[87] 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.
[88] P. Yao, H. Wu, B. Gao, S. B. Eryilmaz, X. Huang, W. Zhang, Q. Zhang, N. Deng, L. Shi, and H.-S. P. Wong, “Face classification using electronic synapses,” *Nature communications,* vol. 8, no. 1, pp. 1-8, 2017.
[89] C. Yakopcic, M. Z. Alom, and T. M. Taha, "Memristor crossbar deep network implementation based on a convolutional neural network." pp. 963-970.
[90] D. J. Mannion, A. Mehonic, W. H. Ng, and A. J. Kenyon, “Memristor-based edge detection for spike encoded pixels,” *Frontiers in neuroscience,* vol. 13, pp. 1386, 2020.
[91] G. M. T. Xavier, F. G. Castañeda, L. M. F. Nava, and J. A. M. Cadenas, “Memristive recurrent neural network,” *Neurocomputing,* vol. 273, pp. 281-295, 2018.
[92] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, "DNN+ NeuroSim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies." pp. 32.5. 1-32.5. 4.
[93] C.-M. Jung, J.-M. Choi, and K.-S. Min, “Two-step write scheme for reducing sneak-path leakage in complementary memristor array,” *IEEE transactions on nanotechnology,* vol. 11, no. 3, pp. 611-618, 2012.
[94] L. Sun, N. Zheng, T. Zhang, and P. Mazumder, “Fault modeling and parallel testing for 1T1M memory array,” *IEEE Transactions on Nanotechnology,* vol. 17, no. 3, pp. 437-451, 2018.