A QUANTIZATION-AWARE REGULARIZED LEARNING METHOD IN MULTI-LEVEL MEMRISTOR-BASED NEUROMORPHIC COMPUTING SYSTEM

by

Chang Song

B.S., Peking University, Beijing, China, 2015

Submitted to the Graduate Faculty of

Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science

University of Pittsburgh

2017

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This thesis was presented

by

Chang Song

It was defended on

April 3, 2017

and approved by

Hai Li, Ph.D., Adjunct Associate Professor,
Department of Electrical and Computer Engineering

Yiran Chen, Ph.D., Adjunct Associate Professor,
Department of Electrical and Computer Engineering

Samuel J. Dickerson, Ph.D., Assistant Professor,
Department of Electrical and Computer Engineering

Zhi-Hong Mao, Ph.D., Associate Professor,
Department of Electrical and Computer Engineering and Bioengineering

Thesis Advisor: Hai Li, Ph.D., Adjunct Associate Professor,
Department of Electrical and Computer Engineering

A QUANTIZATION-AWARE REGULARIZED LEARNING METHOD IN MULTI-LEVEL MEMRISTOR-BASED NEUROMORPHIC COMPUTING SYSTEM

Chang Song, M.S.

University of Pittsburgh, 2017

Neuromorphic computing, a VLSI realization of neuro-biological architecture, is inspired by the working mechanism of human-brain. As an example of a promising design methodology, synapse design can be greatly simplified by leveraging the similarity between the biological synaptic weight of a synapse and the programmable resistance (memristance) of a memristor. However, programming the memristors to the target values can be very challenging due to the impact of device variations and the limitation of the peripheral CMOS circuitry. A quantization process is used to map analog weights to discrete resistance states of the memristors, which introduces a quantization loss. In this thesis, we propose a regularized learning method that is able to take into account the deviation of the memristor-mapped synaptic weights from the target values determined during the training process. Experimental results obtained when utilizing the MNIST data set show that compared to the conventional learning method which considers the learning and mapping processes separately, our learning method can substantially improve the computation accuracy of the mapped two-layer multilayer perceptron (and LeNet-5) on multi-level memristor crossbars by 4.30% (11.05%) for binary representation, and by 0.40% (8.06%) for three-level representation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

Over the past two years of studying at University of Pittsburgh, I have learned both knowledge and research skills. Here, I want to thank Dr. Hai Li and Dr. Yiran Chen, for all their supports and guidance during my Master Program. I am also very thankful to Dr. Samuel J. Dickerson and Dr. Zhi-Hong Mao for their kindly consenting to be my committee members. I appreciate all of the extremely helpful suggestion discussion. I thank all the lab members in Dr. Chen's research group as well as all my friends.

1.0 INTRODUCTION

The modern computing industry is constructed atop two supporting pillars: semiconductor manufacturing and computer architecture. Scaling of conventional CMOS devices, however, is approaching its physical limit [1]. Moreover, the increasing gap between the computing power of microprocessors and available memory bandwidth (a.k.a., “memory wall” challenge) is becoming more prominent than ever, greatly hindering continuing performance improvement for the conventional von Neumann architecture [2].

It has been a long-held belief that a biological computing model inspired from the human brain may solve the challenges that the von Neumann architecture faces [3]. The VLSI realization of such a neuro-biological architecture, namely, *neuromorphic computing*, has recently been revitalized by the application of emerging devices [4]. As an example of a promising design methodology, synapse design can be greatly simplified by leveraging the similarity between the biological synaptic weight of a synapse and the programmable resistance (memristance) of a memristor [5].

Operation of a memristor-based neuromorphic computing systems (NCS) requires first training the system to a state that offers the targeted function [6]. A natural way to train NCS is denoted as *online training*, which works by iteratively adjusting the weights of the neural network (or, the resistance of the memristors) to the target by monitoring the discrepancy between the received and desired system output during training. Another way is called *offline training*, which

directly programs the weights of the neural network to pre-calculated values [7]. In a memristor-based NCS, however, programming the resistance of the memristors suffers from intrinsic device parametric and switching variabilities. The number of resistance levels realized on the memristors is often limited, usually only four with reasonable implementation and energy costs of programming circuitry [8]. Mapping the neural network with floating point weights onto the memristors with multi-level resistance states inevitably causes computation accuracy loss, namely, *quantization loss*.

In this thesis, we first perform a systematic analysis to understand the generation of quantization loss in memristor-based NCS. Based on our analysis, we propose a regularized offline learning method that can minimize the impact of quantization loss during neural network mapping, which is automatically minimized during the epochs of the learning process as a part of the optimization target. The efficacy of two tunable regularization terms – cosine and sawtooth functions, are investigated. For application to the MNIST dataset, our results show that the regularized learning method can improve the computation accuracy of two-layer multilayer perceptron (and LeNet-5) on the memristor-based NCS by 4.30% (11.05%) for binary representation, and 0.40% (8.06%) for three-level representation. The results also show that the three-level representation of neural network weights is actually sufficient for the majority of the regularization methods for the given benchmark.

The remainder of the thesis is organized as follows: Chapter 2 presents the preliminaries of memristors and memristor-based NCS; Chapter 3 analyzes the generation of quantization loss in the memristor-based NCS and its impacts on computation accuracy of the NCS; Chapter 4 gives the details of the proposed learning method; Chapter 5 shows the experimental results and the relevant discussions; and Chapter 6 concludes our works.

2.0 PRELIMINARY

2.1 MEMRISTOR BASICS

The first explicit theoretical depiction of the *memristor* appeared in an article written by Prof. Leon Chua [9]. As the 4th fundamental circuit element, a memristor uniquely defines the relationship between magnetic flux and electrical charge. The resistance state (often referred to as *memristance*) of a memristor can be tuned by applying an electrical excitation. In 2008, HP Labs reported that memristive effect was realized by moving the doping front along a TiO_2 thin-film device [10].

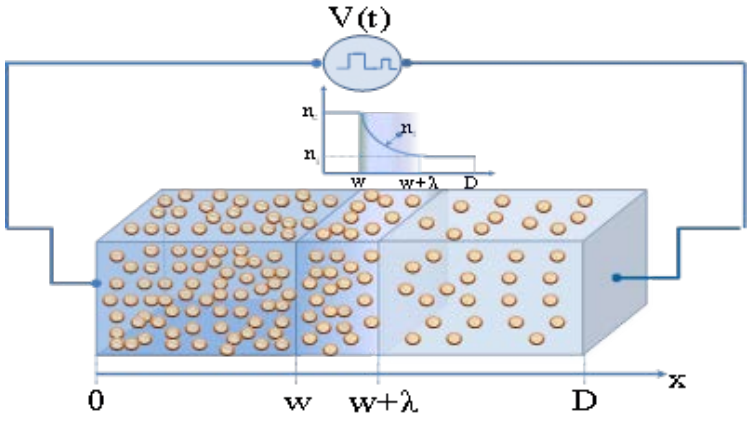

Figure 1. Ion migration filament model of metal-oxide memristors [11].

Figure 1 shows an ion migration filament model of metal-oxide memristors [11]. A metal-oxide layer is sandwiched between two metal electrodes. During the *reset* process, the memristor switches from a low resistance state (LRS) to a high resistance state (HRS). The oxygen ions migrate from the electrode/oxide interface and then recombine with the oxygen vacancies. A partially ruptured conductive filament region with a high resistance per unit length (R_{off}) is formed on the left of the conductive filament region with a low resistance per unit length (R_{on}). Conversely, during the *set* process, the memristor switches from a HRS to LRS and the ruptured conductive filament region shrinks. The resistance of a memristor can be programmed to any arbitrary value between the LRS and HRS by tuning the magnitude and pulse width of the programming current/voltage.

2.2 MEMRISTOR-BASED NCS

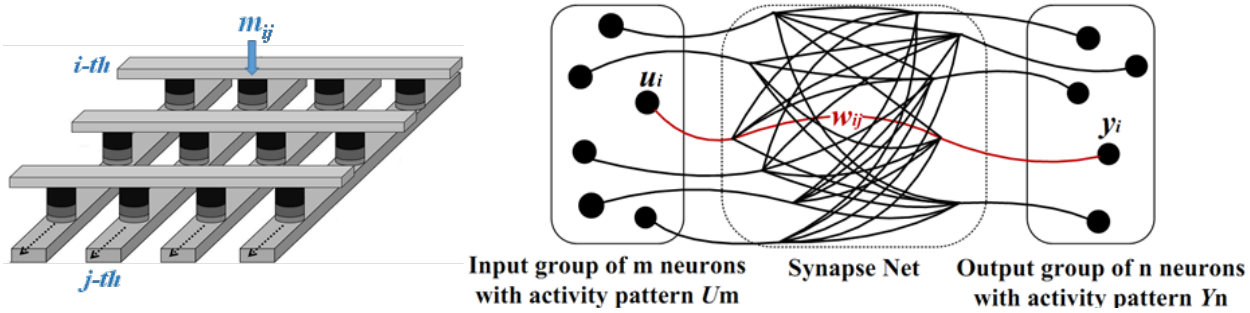

Figure 2. Conceptual views of (a) memristor crossbar and (b) neural network model.

Figure 2(a) depicts a conceptual overview of a memristor crossbar that is used to implement the neural network shown in Figure 2(b). In the neural network, two groups of neurons are connected by synapses, which are realized using the memristors. Input neurons send voltage signals to the memristor crossbar and the output neurons collect the transferred signals (currents) from the input neurons through the memristors and process them with an activation function. Here the amplitude of the signals received at the output neurons is manipulated by different resistances of the memristors, which mimic the synaptic strengths in the neural network. In general, the relationship between the activity patterns of the input neurons *u* and the output neurons *y* can be described by [6]:

$$y_n = \boldsymbol{W}_{n\times m} \cdot u_m. \quad (1)$$

Here the weight matrix $\boldsymbol{W}_{n\times m}$ denotes the synaptic strengths between the two neuron groups.

Feedforward testing: The computation process defined in Eq. (1) is normally called "feedforward testing", which is an important component in the recall process of NCS. As shown in Figure 2(a), a voltage vector is applied to the word-lines (WLs) of the memristor crossbar to represent $\boldsymbol{u}$ while all the bit-lines (BLs) are grounded. Since each memristor has been programmed to a resistance state corresponding to the synaptic weight, the amplitude of the current along each BL reflects the product of the input signal and the synaptic weight. The output current vector $\boldsymbol{y}$ from the crossbar is collected and processed by output "neurons", which may be implemented with CMOS analog circuitry or emerging devices. In practice, the matrix $\boldsymbol{W}$ is often realized by two sets of memristors, which represents the positive and negative elements of $\boldsymbol{W}$, respectively.

Training*:* Another important operation of memristor-based NCS is "training". There are two types of training schemes: *offline training* and *online training*. *Online training* denotes hardware designs that can update $\boldsymbol{W}$ iteratively with the feedback from the output of the NCS [6]. Online

training is generally associated with high circuit design complexity and implementation cost. Moreover, any changes in the training method require redesign of the training circuit. In *offline* training, the matrix $\boldsymbol{W}$ is calculated by a computer based on training data. After obtaining $\boldsymbol{W}$, the memristors in the NCS are directly programmed to a resistance state $\boldsymbol{R}$ that represents $\boldsymbol{W}$, say, $\boldsymbol{R} = 1./\boldsymbol{W}$ (which means take reciprocal by element). A programming pulse with a specific amplitude and duration is then applied to each memristor based on the current resistance state of the device and the target state. As one example, the voltages of the WL and BL connecting to the memristor are set to $+V_{bias}$ and GND, respectively, while all other WLs and BLs are connected to $+V_{bias}/2$. Here $+V_{bias}$ is the bias/programming voltage applied to the memristor. Only the resistance of the memristor that is applied with the full $+V_{bias}$ above the threshold is effectively programmed while the resistances of other memristors in the crossbar remain unchanged. This method is referred to as "half-select" programming scheme [12]. In this work, we select offline training with half-select programming scheme as the baseline of NCS training.

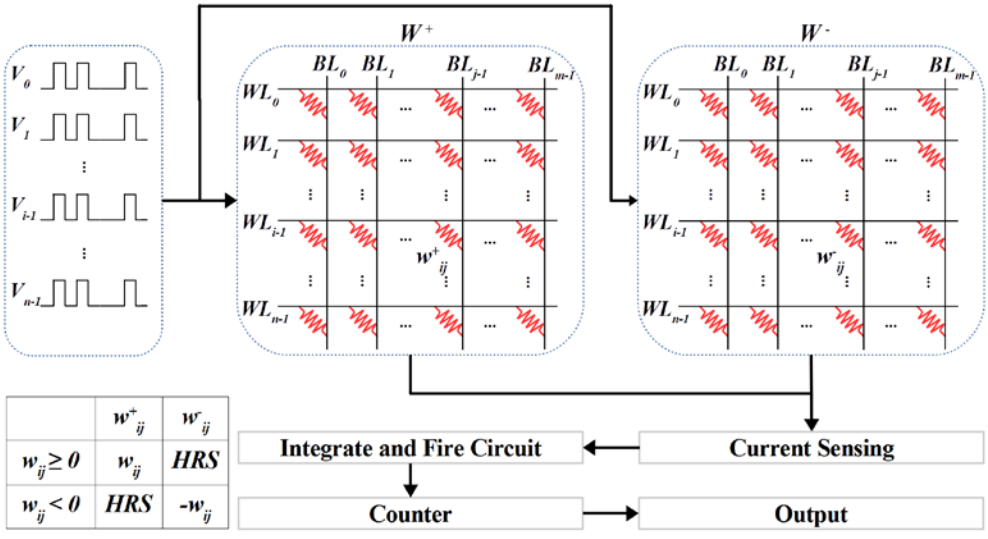

Figure 3. The conceptual diagram of two MBC crossbars with Integrate and Fire Circuit [13].

Figure 3 shows a conceptual diagram of the hardware realization using memristor-based crossbars in the NCS. Two blocks of crossbars, $\boldsymbol{W}^{+}$ and $\boldsymbol{W}^{-}$, represent positive weights and negative weights, respectively. In the offline training stage, $\boldsymbol{W}$ will be written into two crossbars and satisfies: $\boldsymbol{W}^{+} - \boldsymbol{W}^{-} = \boldsymbol{W}$, in which $w_{ij} = 0$ corresponds to the HRS in memristor crossbars. In the testing stage, the current sensing circuit reads the weight-related current and the integrate-and-fire circuit (IFC) is used to convert analog computing data into digital outputs [13].

3.0 QUANTIZATION IN NCS DESIGNS

3.1 WHAT IS QUANTIZATION

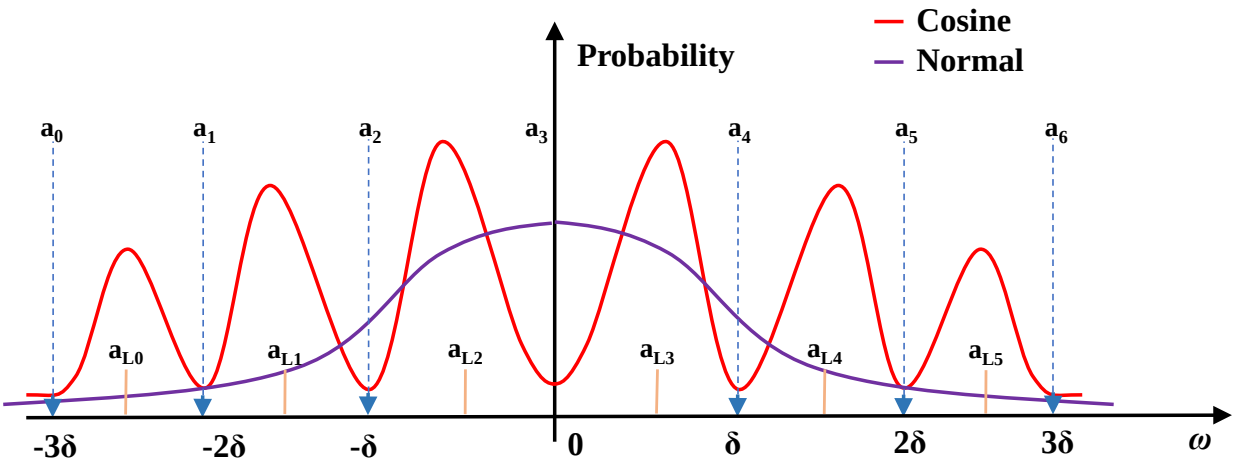

Figure 4. Quantization process in NCS design.

Theoretically, a memristor can be programmed to any arbitrary resistance state. In reality, however, the programming process is limited by the resolution that CMOS circuitry can offer. In memristor-based NCS designs, limited programming resolution requires a *quantization* process that maps each analog weight to one of the values that are represented by the discrete resistance states of the memristors. As illustrated in Figure 4, an analog weight within the range between a_i

and a_{i+1} $(i = 0, \cdots, m-1)$ in the neural network are represented by only one value $a_{Li} \in [a_i, a_{i+1}]$ after quantization. Here m is the number of distinctive levels that the resistance of memristors can be programmed to. The process of quantization is straightforward once each quantization range $[a_i, a_{i+1}]$ and quantization value a_{Li} are determined.

3.2 IMPACT OF DEVICE VARIATIONS

Besides the resolution that the programing circuitry can offer, i.e., programming signal amplitude and duration, another factor that greatly affects the maximum number of the resistance levels of a memristor is device variations, including both parametric and switching variabilities. Many previous studies [14] have proved that the programmed resistance state of the memristor generally follows a lognormal distribution, say, $r \to e^{\theta} \cdot R_q$. Here, $\theta \sim \mathcal{N}(0, \sigma^2)$ and R_q is the nominal resistance level that the memristor should be programmed to, as depicted in Figure 4.

Robust computation of NCS requires maintaining minimum distinction between resistance levels in order to keep the referencing error rate below a threshold. Hence, the number of the resistance levels that a memristor can be programmed to is further limited by memristor device variations in addition to the resolution of the programming circuitry. In offline training, obtaining a high precision of memristor resistance levels, say, more than four levels (2-bit), requires precise output signal monitoring and programming signal control [8]. The overheads quickly become unaffordable when the scale of the NCS increases. Therefore, an approach that can achieve high computation accuracy of the memristor-based NCS with limited precision of the quantized neural network is important in neuromorphic computing research.

As aforementioned, the weights of neural network are indeed shown as the conductance of the memristors. Since the resistance states of the memristors follow a lognormal distribution, it is widely accepted that the values of $1/a_{Li}$ shall be evenly distributed between $1/a_{L(m-1)}$ and $1/a_{L0}$ to achieve the maximum distinction between the adjacent resistance levels when device variations are taken into account. Here a_{L0} and $a_{L(m-1)}$ are the lowest and the highest nominal resistance levels of the memristors, which correspond to the LRS and HRS, respectively.

3.3 QUANTIZATION LOSS

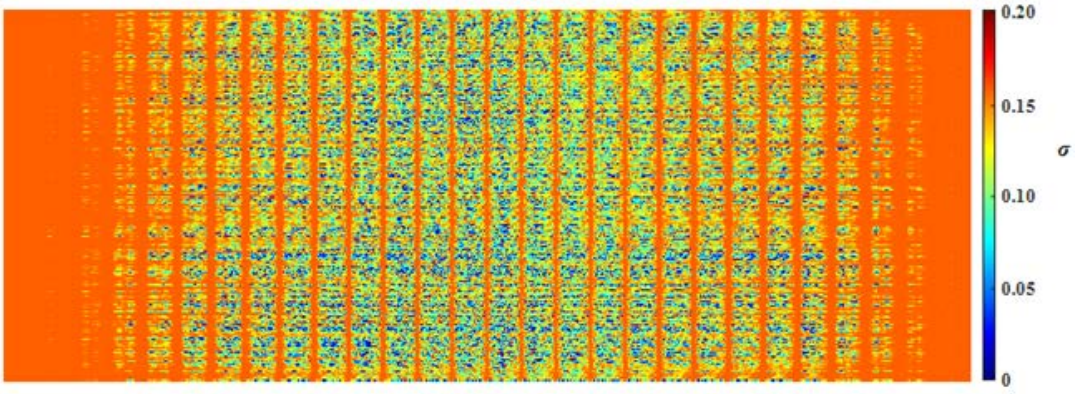

Figure 5. An example of the deviation between the trained analog weight matrix of the neural network and the quantized weight matrix presented on a 300×784 memristor crossbar.

The deviation between the trained analog weight matrix of the neural network $\boldsymbol{W}$ and the quantized weight matrix $\boldsymbol{W_q}$ presented on the memristor crossbar results in quantization loss of the NCS. Figure 5 virtualizes such a deviation on a 300×784 memristor crossbar of a NCS that is used for

MNIST applications [15]. After quantization, the testing accuracy of the NCS reduces from 95.66% down to 89.27%, even without including device variations.

Based on Eq. (1), the output current *y* at each column of the memristor crossbar in the NCS is a linear combination of the products between the input signals and the programmed weights on the column. Hence, the "accumulated squared error (ASE)" on the weight matrix $\boldsymbol{W}$ is often used to measure the impact of quantization and device variations as:

$$ASE = \left\|\boldsymbol{W} - \boldsymbol{W_q} \cdot e^{\boldsymbol{\theta}}\right\| = \sum_{i=1}^{m}\sum_{j=1}^{n}(w_{ij} - w_{q,ij} \cdot\ e^{\theta_{ij}})^2. \quad (2)$$

As shown in Eq. (2), ASE calculates the total squared deviation between the trained weight matrix $\boldsymbol{W}$ and the actually mapped and programmed weight $\boldsymbol{W_q} \cdot e^{\boldsymbol{\theta}}$ on the memristors. Here $\boldsymbol{\theta}$ is a $m \times n$ matrix where $\theta_{ij} \sim \mathcal{N}(0, \sigma^2)$. A larger ASE generally implies a higher quantization loss.

Once the weight values of a neural network are obtained, we can estimate the quantization error from a statistic point of view. For instance, according to central limit theorem (CLT), a common approximation of the weight value distribution is Gaussian distribution, i.e., $w \sim \mathcal{N}(\mu, \sigma^2)$. Here, μ and σ are the mean and the variance of all the weights of the neural network, respectively. Thus, the probability density function (PDF) of w can be expressed as:

$$p\ = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(w-\mu)^2}{2\sigma^2}}. \quad (3)$$

For illustration purpose, here we assume $\mu = 0$ and the lowest and the highest resistance levels of the memristors bound 3-sigma deviations of the weight distribution as $w \in [-3\sigma, +3\sigma]$. In quantization process, we divide the bounded range into several segments $X_1, X_2, \cdots, X_m$, each of which is assigned with a quantized weight value w_{qi} $(i = 1, \cdots, m)$. Since the positive and the negative elements of the weight matrix $\boldsymbol{W}$ are represented by different memristor sets/crossbars, respectively, we can assign equal number of the quantization segments *M* to the positive and

negative weights, such as $m = 2M$ and $X_i = \left[\frac{(i-M-1)\cdot 3\sigma}{M}, \frac{(i-M)\cdot 3\sigma}{M}\right)$, $i = 1, \cdots, 2M$. The mean square error (MSE) between the weight w and its quantized value w_q can be then expressed by:

$$MSE_{org} = \frac{1}{2M}\sum_{i=1}^{2M} MSE(X_i) = \frac{1}{2M}\sum_{i=1}^{2M}\int_{(i-M-1)\cdot 3\sigma / M}^{(i-M)\cdot 3\sigma / M}[w - \frac{(2i-2M-1)\cdot 3\sigma}{2M}]^2 . \quad (4)$$

When $M = 3$ (three-level memristor), $MSE_{org} \approx 8.327\sigma^2$.

3.4 IMPACT OF WEIGHT VALUE DISTRIBUTION ON QUANTIZATION LOSS

The previous analysis shows that the weight value distribution of a neural network greatly affects the quantization loss in NCS designs. Here, let's consider a different weight value distribution that can be expressed by:

$$w_{new} = \frac{\sum_{i=1}^{2M} a_i \cdot N(\mu_i, \sigma_i^2)}{\sum_{i=1}^{2M} a_i} . \quad (5)$$

In this new weight value distribution, $2M$ quantization segments are retained as the same as the original distribution in Chapter 3.3. However, the contained weight values in each segment now follow a Gaussian distribution, where $\mu_i = \frac{(2i-2M-1)\cdot 3\sigma}{2M}$ and $\sigma_i \leq \frac{\sigma}{2M}$. As such, the *MSE* between w_q and w_{new} becomes:

$$MSE_{new} = \frac{\sum_{i=1}^{2M} a_i \cdot \sigma_i^2}{\sum_{i=1}^{2M} a_i} \leq \frac{\sigma^2}{4M^2} < MSE_{org} . \quad (6)$$

The above mathematic analysis shows that different weight value distribution may introduce different quantization loss, which indeed inspires this work: *if we can optimize the weight value distribution by considering the quantization process during learning process, we may be able to reduce the quantization loss and therefore improve the computation accuracy of the NCS.*

4.0 REGULARIZED LEARNING METHOD

4.1 TRADITIONAL REGULARIZATION

In the study of machine learning, regularization is referred to as the process of introducing additional information to prevent overfitting. Without loss of generality, the regularization term can be expressed by a complexity penalty added to the cost function of the learning process, such as

$$J' = J + \lambda \cdot R(J). \tag{7}$$

Here coefficient λ controls the importance of the regularization term and $R(J)$ is the added complexity in addition to the original cost function J.

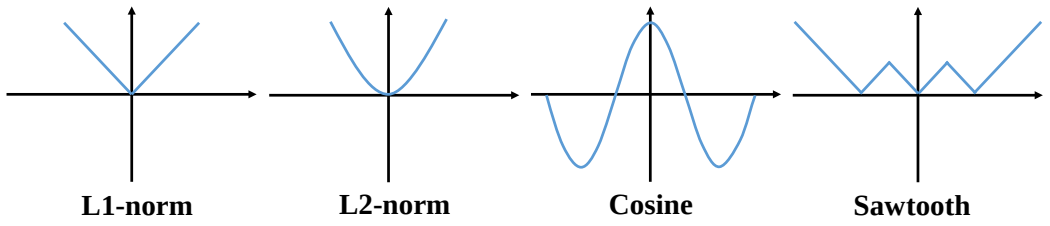

Figure 6. Four regularizations investigated in this thesis.

Two popular regularization methods are L1-norm regularization and L2-norm regularization as:

$$R_{L1}(J) = \|w\|, \text{ and} \tag{8}$$

$$R_{L2}(J) = \|w\|^2, \tag{9}$$

respectively, as depicted in Figure 6. The hypothesis behind these two regularization methods is that a bigger weight in a neural network is likely to produce larger impact on the output of system. Compared to L1-norm, L2-norm gives more bias to the large weights during cost function optimization and pushes the weight value distribution to the small value side. When utilizing the MNIST dataset [15], for example, both cases result in very similar classification rates (i.e., 95.639% for L1-norm and 95.654% for L2-norm) with the same network structures. Nonetheless, none of these two regularization methods consider the quantization loss that were discussed in Chapter 3.3.

4.2 COSINE REGULARIZATION

An obvious derivation from the analysis in Chapter 3.3 is that if the trained weights are "clustered" around the quantized weights, the quantization loss will be effectively reduced. For a NCS with multi-level representation, the trained weights shall be clustered around each level, as is discussed in Chapter 3.4.

In observation of the periodic property of multi-level representation, we propose a novel cosine regularization to intentionally cluster the weight values during the learning process as:

$$R_{cosine}(J) = \cos(\omega \cdot \boldsymbol{W}). \tag{10}$$

Compared to L1-norm and L2-norm regularizations, the cosine regularization demonstrates the following two major advantages: 1) The periodicity of cosine function can be leveraged to reshape the distribution of the weight values so that the weight values are concentrated around the corresponding quantized levels; 2) Compared to L1-norm and L2-norm, the additional parameter

ω in cosine function can help the fine-tuning process to further improve the NCS accuracy, indirectly reducing the quantization loss.

4.3 OPTIMIZATION OF COSINE REGULARIZATION

We have two parameters for tuning cosine regularization: ω and λ. ω is the parameter that defines the periodicity of the cosine regularization and hence the quantized levels. λ controls the contribution (strength) of the cosine regularization term to the cost function. How to optimize these parameters is critical to improving the testing accuracy of the trained neural network. In machine learning practices, the optimum values of these parameters vary with specific applications and data. Hence, a fine-tuning process is often needed.

One practical way to perform fine-tuning is to scan the concerned range of parameters and compare the learning results. In this work, we propose a stochastic training process with dynamic learning rates, i.e., changing the learning rate in some specified epochs. The training samples are randomly selected from the training dataset to minimize the influence of data selection bias. The training processes are repeated by scanning the values of ω and λ until the optimum parameters/accuracy are obtained.

4.4 SAWTOOTH REGULARIZATION

Cosine regularization is not the only periodic regularization function that satisfies the advantage statement in Chapter 4.2. A periodic extension of L1-norm regularization - sawtooth regularization

- possesses the characteristic similar to that of cosine regularization, as also shown in Figure 6. Sawtooth regularization also has two parameters λ and τ, which define the height and the base-side of the triangles, respectively. Same as their counterparts in cosine regularization, λ and τ define the strength and periodicity of the sawtooth regularization, respectively. Here the sawtooth shape is maintained only within the concerned range, as shown in Figure 6.

However, sawtooth regularization still has many differences from cosine regularization that affect the training effect. First, the cosine function and its derivative are both continuous and easy to compute, while sawtooth function's derivative is not. Second, cosine function has infinite number of local minima's in $[-\infty, \infty]$. Some trained weights may fall into the minima's that are far from zero. The local minima's of the sawtooth function, however, are distributed over only the concerned range. This helps to maintain the trained weights within a specified, in our case, near-zero, range. As we will show in our experimental results in Chapter 5, the above differences generate considerable impact on the training process of the NCS.

5.0 EXPERIMENTAL RESULTS

5.1 PLATFORM AND BENCHMARK

To evaluate our proposed regularized learning methods, we implement a two-layer multilayer perceptron (MLP) and train it in MATLAB for targeting the MNIST digits classification dataset [15], which contains 60,000 training samples and 10,000 testing samples. The input data of the MLP are 28×28 pixels images and 64 hidden neurons are used, requiring a 784×64 matrix to store the hidden weights. The output is a 10×1 matrix, which indicates ten classes from ‘0’ to ‘9’. As a result, the output weights are stored on a 64×10 matrix. For each layer, we use two memristor crossbars to represent the positive and the negative weights, respectively.

As aforementioned in Chapter 4.3, in our proposed stochastic training process, 100 training samples are randomly chosen from the training dataset in each epoch to eliminate the influence of accidental errors. 1,000 epochs are performed in each training and the learning rate is halved in specified epochs to train the neural network completely. The optimal parameters of the cosine and the sawtooth regularizations are obtained by scanning the concerned range. In all simulations, the learning rate remains fixed for the same epochs to ensure a fair comparison between different regularizations. Each experiment runs 10 times and the results are averaged to eliminate result fluctuations. This results in a total of 100,000 samples for each case which is proved in pre-test to

be sufficient for preventing accidental error and over-fitting. Table 1 summarizes the parameters of the memristors, MBC and IFC designs used in our NCS simulations.

Table 1. Simulation Parameters for MLP [7]

Device	R_{on}	R_{off}	σ	ω $(2\pi/\tau)$	λ
	10 kΩ	1 MΩ	0 - 2.0	10 - 150	0.01 - 0.1
MBC	V_{bias}	V_{read}	**Hidden Layer**		**Output Layer**
	2.9 V	1.0 V	784×64		64×10
IFC [13]	V_{dd}	V_{th}	**Power**		**Max Throughput**
	1.2 V	0.5 V	0.48 *pJ*/spike		568.2M spikes/sec

Our results show that a classification accuracy of 95.64% can be achieved using L1-norm regularization, which is even slightly better than the result (95.3%) reported in [15] about a two-layer neural network with 300 hidden neurons. The result of L2-norm regularization is also very similar (95.65%). Hence, for simplicity, we use L1-norm regularization as the baseline example in our discussions.

5.2 OPTIMAL PARAMETER SELECTION OF COSINE REGULARIZATION

As previously discussed, during the fine-turning process, we vary the values of parameters of each regularization and look for the peak accuracy that can be reached. For example, our simulations show that the best classification rate of the designed NCS with binary memristor resistance levels is 92.65%. The corresponding optimal values of ω and λ are 15 and 0.02, respectively.

An important observation from the results is that the values of ω and λ resulting in the highest trained accuracy are not necessarily the ones which supply the highest accuracy after quantization. In fact, the selection of ω imposes very minimum impact on the trained accuracy while increasing λ causes monotonic degradation of the trained accuracy. After quantization, however, there exists optimal values of ω and λ.

5.3 EFFICACY OF REGULARIZATION METHOD

Table 2. Accuracy comparison of different regularizations (MLP)

	Regularization	**Trained Accuracy**	**Quantized Accuracy**
Binary	*L1-norm*	95.66%	89.27%
	L2-norm	95.32%	91.52%
	Cosine	95.51%	92.65%
	Sawtooth	95.41%	93.57%
3-level	*L1-norm*	95.64%	93.90%
	L2-norm	95.49%	93.83%
	Cosine	95.59%	94.11%
	Sawtooth	95.63%	94.30%

Table 2 compares the NCS accuracy using different regularizations before and after quantization. Here the weights are represented as a binary memristor resistance level. Slightly higher or similar trained accuracies are achieved in L1-norm (95.66%) and L2-norm regularizations (95.32%) compared to cosine regularization (95.51%). After quantization, the NCS accuracies of L1-norm and L2-norm regularizations significantly drop down to 89.27% and 91.52%, respectively. Cosine regularization, however, can achieve a considerably higher quantization accuracy of 92.65% at the

optimal [ω, λ] = [15, 0.02], indicating the good efficacy of our proposed learning methods. Note that here the impacts of device variations have not been considered yet in the simulations.

We also evaluate the efficacy of the regularized learning methods when the programming resolution of the memristors increases to three levels. For all regularizations, the trained accuracies are almost the same as that in binary-level while the quantized accuracies are all improved. It implies that increasing the resolution of the weights stored on the memristors can effectively enhance the NCS accuracy by reducing the quantization loss. Nonetheless, cosine regularization achieves higher quantized accuracy than L1/L2-norm, which (94.11%) is very close to the originally trained one (95.59%). This result implies that three-level representation may be sufficient for MINIST applications. Continuing to increase the number of levels may result in only marginal benefit from quantization. Here the peak quantized accuracy of the NCS happens when [ω, λ] = [20, 0.01].

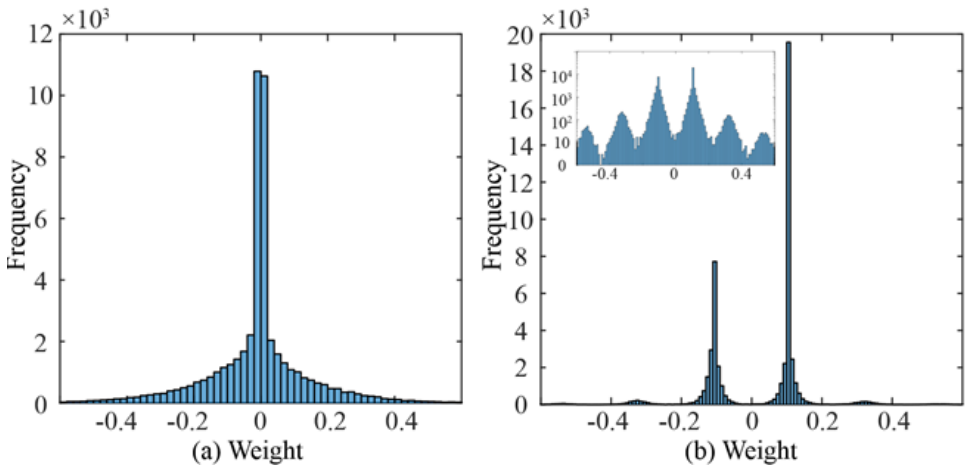

Figure 7. Comparison of weight distributions of (a) L1-norm and (b) cosine regularizations after training for three-level representation.

Figure 7 compares the weight distributions of L1-norm and cosine regularizations for three-level representation. The small figure in Figure 7(b) shows the same weight distribution plotted with *x*-axis in log scale. Compared to L1-norm regularization, training weights cluster around the six quantized levels (including positive and negative ones) in cosine regularization, showing a better tolerance to quantization loss.

5.4 TOLERANCE TO RESISTANCE VARIATIONS

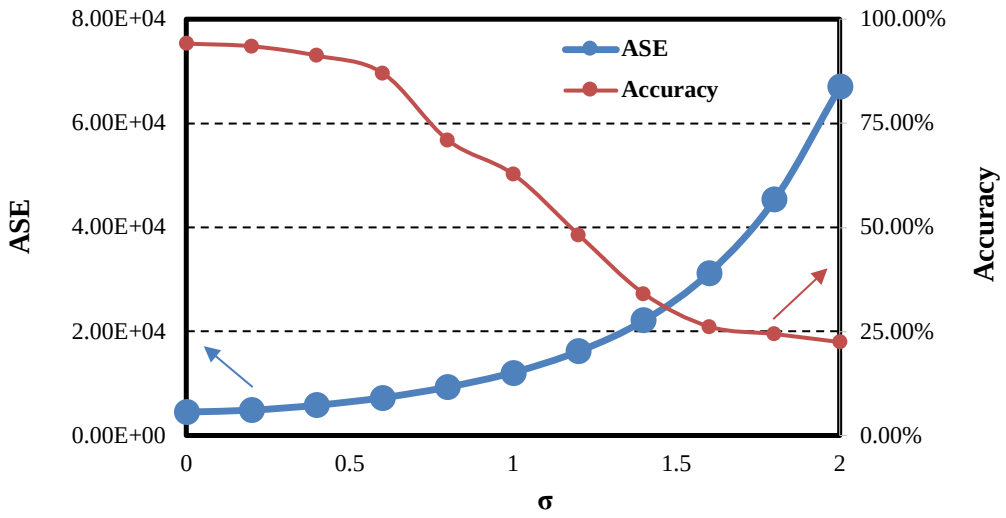

Figure 8. Tradeoffs between the resistance variations, ASE and NCS accuracy for cosine regularization with three-level representation.

Memristor resistance variation introduces additional inaccuracy to NCS computation atop quantization loss. Figure 8 shows the tradeoff between the memristor resistance variations, the

ASE, and the NCS accuracy for cosine regularization with three-level representation. Here σ of the lognormal distribution of the memristor resistance varies from 0 to 2. As σ increases, the ASE keep increasing, leading to continuous drop of the NCS accuracy. Furthermore, the derivatives of both ASE and NCS accuracy increase when σ increases. As σ increases from 0 to 0.6, the NCS accuracy drops only 7.09% (from 94.11% to 87.02%), but with σ increases to 1.2, the increase of ASE is 3.25× of the former chapter and the NCS accuracy drops significantly to 48.16%. These results prove the discussion in Chapter 3.3 and show that the NCS accuracy is still limited by device variations regardless of regularization method used.

5.5 SAWTOOTH REGULARIZATION

Similar simulations are performed on sawtooth regularization to evaluate the corresponding property. As summarized in Table 2, sawtooth regularization achieves similar trained accuracy but the highest quantized accuracy among all the regularizations for both binary and three-level memristor resistance levels. The corresponding optimal values of λ and τ are $[\lambda, \tau]$ = [0.02, 0.419] for binary and $[\lambda, \tau]$ = [0.02, 0.314] for three-level memristor resistance levels. These results imply that sawtooth regularization offers the best capability to reduce quantization loss for two-layer MLP.

5.6 LENET-5 ON THE MEMRISTOR-BASED NCS

We also evaluate our method on LeNet-5 [15], which is a convolutional neural network (CNN) with convolutional layers, subsampling layers, and one fully connected layer. In our simulation, LeNet-5 is trained in MATLAB (1,200 training epochs/case) using MNIST as input dataset and then two convolutional layers and one fully connected layer are mapped to the memristor-based NCS. The trained and quantized accuracies of four different regularizations before and after quantization are as illustrated in Figure 9, the margin between the dotted line and the bar indicates the quantization loss.

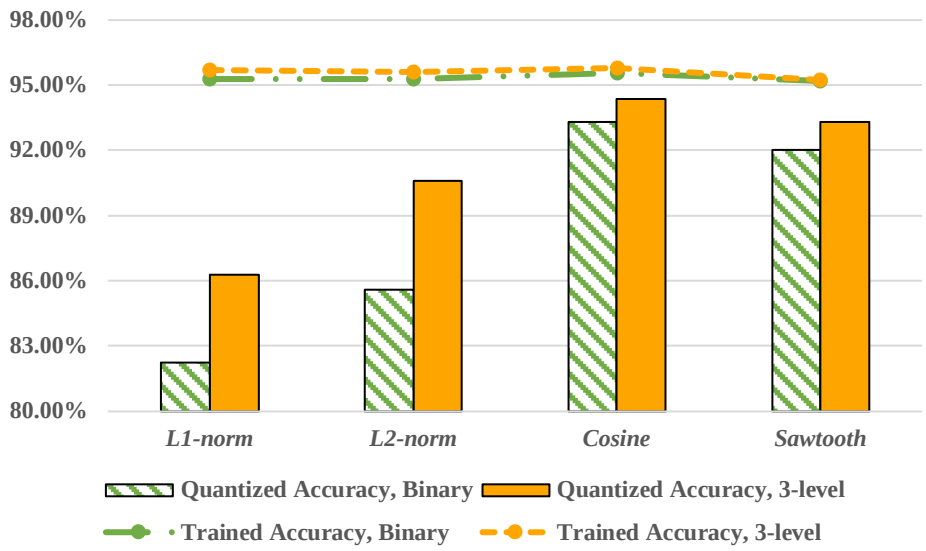

Figure 9. Accuracy comparison of different regularizations on LeNet-5.

From the results, we can come to the conclusion that our proposed regularizations outperform the traditional methods in both binary representation and three-level representation. Using binary

representation, cosine and sawtooth regularizations show better performance than L1-norm and L2-norm. Cosine regularization achieves the highest quantized accuracy (93.29%), 11.05% higher than the quantized accuracy of L1-norm regularization (82.24%), which is the lowest among four regularizations. With three-level representation, cosine and sawtooth regularizations still have a better quantization loss tolerance than L1-norm and L2-norm, although the accuracy difference between the quantized accuracies of cosine (94.35%) and L1-norm (86.29%) shrinks to 8.06%. This is because three-level representation is more precise than binary representation, as we discussed in Chapter 5.3.

Compared with two-layer MLP, the differences between using proposed regularizations and traditional regularizations in LeNet-5 is much more significant because as the complexity of the neural network or the total number of layers increases, quantization loss has a greater negative impact on the NCS designs.

6.0 CONCLUSION

In this thesis, we propose a regularized learning method that can take into account the quantization loss in the memristor-based NCS design with limited number of resistance levels. Two regularizations, cosine and sawtooth, are introduced to the cost function of learning process in order to concentrate the trained weights around the quantized levels for quantization loss reduction. For MNIST applications, experimental results show that compared to conventional learning method with L1/L2-norm regularizations, our learning method can substantially improve computation accuracy of the mapped MLP and LeNet-5 on the memristor-based NCS. The regularization selection and the optimal parameters are related to both application type and network topology, which will be investigated in the future. Although we use the memristor-based NCS as the example to demonstrate the efficacy of the regularized learning methods, our methods can be easily extended to other hardware platforms that suffer from low weight precision.

BIBLIOGRAPHY

[1] A. Mahesri and V. Vardhan, "Power consumption breakdown on a modern laptop," in *International Workshop on Power-Aware Computer Systems*, vol. 3471, pp. 165-180, 2004.

[2] A. Sally, "Reflections on the memory wall," in *Proceedings of the 1st Conference on Computing Frontiers*, pp. 162, 2004.

[3] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, et al., "TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537-1557, Oct. 2015.

[4] M. Sharad, D. Fan, and K. Roy, "Ultra low power associative computing with spin neurons and resistive crossbar memory," in *Proceedings of the 50th Design Automation Conference*, pp. 1-6, Jun. 2013.

[5] B. Liu, X. Li, Q. Wu, T. Huang, H. Li, and Y. Chen, "Vortex: Variation-aware training for memristor X-bar," in *Proceedings of the 52nd Design Automation Conference*, pp. 1-6, Jun. 2015.

[6] M. Hu, H. Li, Y. Chen Q. Wu, and G. Rose, "BSB training scheme implementation on memristor-based circuit," *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 80-87, Apr. 2013.

[7] B. Liu, H. Li, Z.-H. Mao, Y. Chen, T. Huang, and W. Zhang. "Digital-assisted noise-eliminating training for memristor crossbar-based analog neuromorphic computing engine," in *Proceedings of the 50th Design Automation Conference*, pp. 1-6, Jun. 2013.

[8] M. Wu, Y. Lin, W. Jang, C. Lin, and T. Tseng, "Low-power and highly reliable multilevel operation in 1T1R RRAM," *Electron Device Letters*, vol. 32, no. 8, pp. 1026-1028, Aug. 2011.

[9] L. Chua, "Memristor-the missing circuit element", *IEEE Transactions on Circuit Theory*, vo. 18, no. 5, pp. 507-519, Sep. 1971.

[10] D. Strukov, G. Snider, D. Stewart, and S. Williams, "The missing memristor found," *Nature*, pp. 80-83, 2008.

[11] L. Zhang, Z. Chen, J. Yang, B. Wysocki, N. McDonald, and Y. Chen, "A compact modeling of TiO_2-TiO_{2-x} memristor," *Applied Physics Letters*, pp. 153503, 2013.

[12] J. Liang and H. S. P. Wong, "Cross-point memory array without cell selectors—device characteristics and data storage pattern dependencies," *IEEE Transactions on Electron Devices*, vol. 57, no. 10, pp. 2531-2538, Oct. 2010.

[13] C. Liu, B. Yan, C. Yang, L. Song, Z. Li, B. Liu, et al., "A spiking neuromorphic design with resistive crossbar," in *Proceedings of the 52nd Design Automation Conference*, p. 14, Jun. 2015.

[14] S. R. Lee, Y.-B. Kim, M. Chang, K. M. Kim, C. B. Lee, J. H. Hur, et al., "Multi-level switching of triple-layered TaOx RRAM with excellent reliability for storage class memory," *Symposium on VLSI Technology*, pp. 71-72, Jan. 2012.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.