Go Unary: A Novel Synapse Coding and Mapping Scheme for Reliable ReRAM-based Neuromorphic Computing

Chang Ma[1], Yanan Sun[1], Weikang Qian[2,3], Ziqi Meng[2], Rui Yang[2] and Li Jiang[1,3]
[1]School of Electronic Information and Electrical Engineering,
[2]University of Michigan-Shanghai Jiao Tong University Joint Institute,
[3]MoE Key Lab of Artificial Intelligence,
Shanghai Jiao Tong University, Shanghai, China

***Abstract*—Neural network (NN) computing contains a large number of multiply-and-accumulate (MAC) operations, which is the speed bottleneck in traditional von Neumann architecture. Resistive random access memory (ReRAM)-based crossbar is well suited for matrix-vector multiplication. Existing ReRAM-based NNs are mainly based on the binary coding for synaptic weights. However, the imperfect fabrication process combined with stochastic filament-based switching leads to resistance variations, which can significantly affect the weights in binary synapses and degrade the accuracy of NNs. Further, as multi-level cells (MLCs) are being developed for reducing hardware overhead, the NN accuracy deteriorates more due to the resistance variations in the binary coding. In this paper, a novel unary coding of synaptic weights is presented to overcome the resistance variations of MLCs and achieve reliable ReRAM-based neuromorphic computing. The priority mapping is also proposed in compliance with the unary coding to guarantee high accuracy by mapping those bits with lower resistance states to ReRAMs with smaller resistance variations. Our experimental results show that the proposed method provides less than 0.45% and 5.48% accuracy loss on LeNet (on MNIST dataset) and VGG16 (on CIFAR-10 dataset), respectively, with acceptable hardware cost.**

***Index Terms*—Neural network, ReRAM, crossbar, MLC, variation, unary coding**

I. Introduction

Neural networks (NNs) contain a huge number of matrix-vector multiplication (MVM) operations. Their performance is limited by the traditional von Neumann architecture. Processing-in-memory (PIM) enables computing inside memory, which reduces the data movement between the processor and memory and has attracted widespread attention. Resistive random access memory (ReRAM) is an attractive novel non-volatile memory (NVM), with desirable characteristics, such as zero-standy power, small area, compatibility with silicon fabrication process, and fast switching capability of multiple resistance levels [1]–[3]. More importantly, PIM can be implemented efficiently with ReRAMs. Recent studies have used ReRAMs to speed up NN computing. Several NN accelerators based on ReRAM crossbar have been developed, such as PRIME [4], ISAAC [5], and PipeLayer [6]. They use conductances of ReRAM, voltages, and currents to represent synaptic weights, inputs, and outputs, respectively. These architectures are faster and consume lower power than CMOS-based accelerators [7].

Due to the immaturity of fabrication process combined with stochastic filament-based switching, ReRAM mainly suffers from the resistance variation problem [8], denoted as the deviation of actual resistance from its target value. The deviation of the resistance affects the precision of the weights, and can significantly degrade the accuracy of NNs [9].

To reduce the device resistance variations, a recent work proposed to repeatedly apply the constant reset pulse [3]. Unfortunately, it is harmful to ReRAM endurance. To overcome the resistance variations, some off-device training methods were presented in [10]–[12]. However, they also have some drawbacks. The methods in [10] and [11] can only handle variation problem in single-layer perception (SLP), which limits the application in deep NNs. Although the method in [12] is applicable to convolution neural networks (CNNs) like VGG and AlexNet, it suffers from significant accuracy loss in the presence of large variations. Moreover, these off-device training methods all utilize the binary coding to represent the synaptic weights, where the most significant bits (MSBs) can amplify the weight deviations. Furthermore, the use of the multi-level cell (MLC) makes the problem even worse.

Accompanying with the recent advance in devices and applications is the development of the emerging computing paradigms. One paradigm receiving much attention today is stochastic computing (SC) [13]. Its basic idea is to use a different coding scheme than traditional binary coding to represent data: It represents a value by a stream of 0s and 1s through the ratio of 1s in the stream. Since each bit in the encoding has the same significance, this is also known as unary coding [14]. The uniform significance of each bit makes the unary coding have a strong tolerance to bit-flip errors.

In this paper, we exploit the error-tolerance feature of unary coding and propose a novel solution to the reliability problem of ReRAM-based NN accelerators. In summary, our contributions are as follows:

- We employ unary coding, implemented with MLC, for weight representation in ReRAM-based NN design for the first time. We apply it to tolerate the device resis-

This work was partially supported by National Key Research and Development Program of China (No. 2018YFB1403400), National Natural Science Foundation of China (NSFC) under Grant No. 61704104 and Shanghai Sailing Program under Grant No. 17Z111260004. Corresponding authors: Yanan Sun, Weikang Qian, and Li jiang. Emails: {sunyanan, qianwk, ljiang_cs}@sjtu.edu.cn.

tance variations, leading to the accuracy recovery of the ReRAM-based NN accelerators.

- We propose a variation-aware priority mapping to further improve the accuracy. This technique exploits a unique feature of unary coding: the existence of many different ways to represent the same value.
- We demonstrate that our methods can tolerate the resistance variations for typical NNs. The experimental results show large accuracy improvement on multi-layer perception (MLP), LeNet, AlexNet, and VGG16. Compared with the previous state-of-the-art work, our accuracy is higher, even when variation is larger.

The rest of the paper is organized as follows. Section II introduces the background on ReRAM-based NN accelerator and discusses the limitation of the related works. Section III describes the proposed methods, including unary coding based on MLC devices and priority mapping. Section IV presents the experimental results. Section V concludes the paper.

II. Background and Related Works

A. *ReRAM and ReRAM-based NN Computing*

ReRAM is a two-terminal variable resistor with an oxide layer sandwiched between two metal electrodes. The device resistance ranges from low resistance state (LRS) to high resistance state (HRS) and can be tuned by a specific voltage. An MLC ReRAM has middle resistance states (MRSs) between LRS and HRS, which can be achieved by precisely controlling the programming voltage [15]. The MRSs of an MLC device can be used to represent different logic values.

The ReRAM-based crossbar can be employed as a matrix-vector multiplier [16] to accelerate NNs using the architecture shown in Fig. 1. Two crossbars are used to represent the weight matrix, with one storing the positive weights and the other storing the negative weights [17] [18]. The absolute value of each weight is first converted into a binary number. Then, the bits of the binary number are used to configure multiple ReRAM devices located in either positive or negative crossbar depending on the polarity of the weight, while zero is written to the other crossbar. The output is the subtraction result of the two crossbars. In each crossbar, the input data are converted to a vector of voltage signals by a digital-to-analog converter (DAC) on each row. The bitline current is fed to a sample-and-hold (S&H) unit followed by a shared analog-to-digital converter (ADC). Each binary weight is represented by multiple ReRAMs, where each one has different significance in binary coding. The output of ADC is therefore required to be shifted before performing the addition operation to complete the MAC operations.

Despite of the promising applications of ReRAM on NN accelerators, the ReRAM devices suffer from resistance variations where the actual resistance deviates from the target value. There are two primary types of resistance variations in ReRAMs: cycle-to-cycle variation (CCV) and device-to-device variation (DDV) [19]. CCV indicates that different amount of resistance variations can be induced in the same

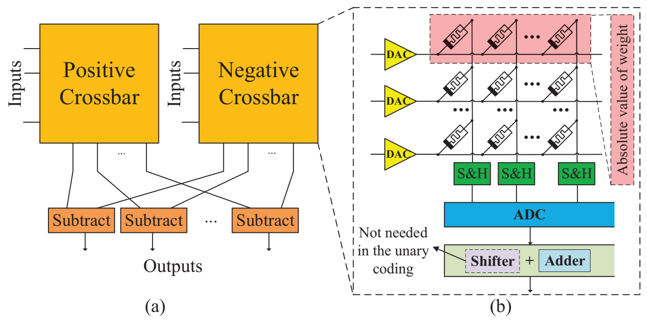


Fig. 1. ReRAM-based NN architecture. (a) Positive and negative weights are represented by two crossbars, and the output is the subtraction of two crossbars. (b) Internal structure of a crossbar, designed to implement MVM.

cell with stochastic filament-based switching for different programming cycles. DDV means that different cells have independent resistance variations due to immature fabrication processes. As the weight matrix is written into the crossbar once and followed by multiple read operations for inference process, ReRAM devices mainly fall into the DDV category for resistance variations. In general, the actual resistance of an ReRAM cell (R') follows a log-normal distribution according to recent studies [3] [20]:

$$R' = R_0 \cdot e^{\theta} \qquad \theta \sim N(0, \sigma^2), \tag{1}$$

where R_0 is a target resistance value of ReRAM. θ follows a normal distribution with zero mean and a standard deviation of σ. A large σ indicates a large resistance variation.

B. *Related Works*

A recent work [3] was presented to reduce the device resistance variations. It repeatedly applies constant reset pulses to the ReRAM devices until the resistance level reaches the target range. However, the frequent write operations for tuning the ReRAM resistance would inevitably shorten the device lifetime and endurance [21].

To overcome the resistance variations, some off-device training methods were proposed for achieving robust NNs. A variation-aware training method called *Vortex* was presented in [10]. It considers the variations during the training phase of the SLP and iteratively tunes a global parameter to obtain a relatively optimal weight matrix. The previous work [11] leveraged the self-healing capability of the NNs to decrease those weights that are mapped to the abnormal ReRAMs with large variations in the training phase for SLP. Unfortunately, these two approaches are not applicable to MLP and CNN where the errors can be accumulated and amplified through multiple layers. Alternatively, the previous work [12] proposed a device variation-aware (DVA) training method which trains the DNN with random noise to get a robust model. DVA training cannot cope with large resistance variation as it does not take into account the variation of each device. Furthermore, the above methods all utilize the traditional binary coding to represent the synaptic weights. With such a coding, the

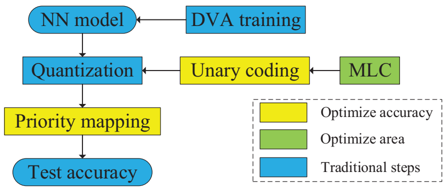


Fig. 2. The overall flow of the proposed method.

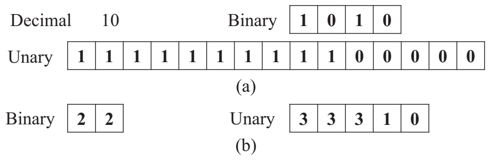


Fig. 3. An example for representing decimal value "10" in binary and unary coding. (a) For SLC. (b) For 2-bit MLC.

bits near the MSB are more significant than the others. The weight deviations can be amplified, which induces significant loss of NN accuracy in the presence of large resistance variations. Besides, the MLC devices are commonly employed for reducing the hardware cost in NNs. Since the significance of an MLC bit is larger than that of a single-level cell (SLC) bit, the accuracy degradation of binary coding-based NNs is even worse under resistance variations.

III. Proposed Method

We propose a novel unary coding with variation-aware priority mapping method to tolerate the resistance variations for MLC ReRAM-based NNs. The overall flow of the proposed method is shown in Fig. 2. An NN (such as MLP and CNN) is trained to get an optimal model. The DVA training method [12] can be employed for obtaining a more robust NN model. Each synaptic weight is then quantized and mapped to the ReRAM-based crossbar based on the proposed unary coding method (see Section III-A) with MLC devices (see Section III-B) for decreasing the weight deviation in the presence of resistance variations. Then, the priority mapping method (see Section III-C) is proposed to map those bits with lower resistance states to the ReRAMs with smaller resistance variations to further reduce the accuracy loss.

A. Proposed Unary Coding Method

In the binary coding, different bits at different positions have different significance. In contrast, the unary coding has all the bits of the same significance. Thus, it essentially uses the number of 1s in the coding to represent a data. Traditionally, the 1s are mapped at the beginning of the unary coding. For example, if a data is quantized to 4 bits in the binary coding, it can represent integers in the range of $[0, 15]$. For the unary coding, 15 bits are needed to represent the same precision. As shown in Fig. 3(a), the decimal value 10, expressed in the binary coding as "1010", is represented by ten 1s and five 0s in the unary coding.

The weight is typically quantized to multiple-bit ReRAMs for reducing data storage and computation complexity in binary coding [18]. However, the variations on the high-order bits magnify the weight deviation from the expectation severely in traditional binary coding. Alternatively, when the weight is unary coded, its value depends on the number of 1s, rather than the position of 1s. This characteristic makes the significance of each bit equivalent, which helps reduce the weight deviation.

TABLE I
The four levels of resistance and conductance values in a 2-bit MLC.

Logic state	0	1	2	3
Conductance	$\frac{1}{200 \cdot LRS}$	$\frac{101}{300 \cdot LRS}$	$\frac{401}{600 \cdot LRS}$	$\frac{1}{LRS}$
Resistance	$200 \cdot LRS$	$3 \cdot LRS$	$1.5 \cdot LRS$	LRS
Symbol	HRS	MRS1	MRS2	LRS

B. Multi-level Cell

An SLC has two resistance levels and can represent a 1-bit data. In contrast, an MLC has more than two resistance levels and can represent a data of more than one bit. For an MLC with 2^k resistance levels, it can represent a data of k bits. We call it a k-bit MLC. For example, to represent a data ranging from 0 to $2^n - 1$, n and $2^n - 1$ SLCs are needed in the binary and unary coding, respectively. When k-bit MLCs are used, only $\left\lceil \frac{n}{k} \right\rceil$ and $\left\lceil \frac{2^n-1}{2^k-1} \right\rceil$ of them are needed in the binary and unary coding, respectively.

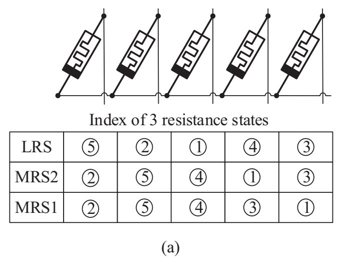
Nowadays, the 2-bit MLCs (representing four different levels from 0 to 3) are widely used in NN architectures with current synthesis technology [5]. Figs. 3(a) and (b) show examples of two different coding methods with SLCs and 2-bit MLCs, respectively. Five 2-bit MLCs have the same representation range as fifteen SLCs for unary coding as shown in Fig. 3. In this case, with the help of MLC, only one third of ReRAMs are needed in comparison with SLC. The four different conductance levels of a 2-bit MLC are typically uniformly distributed. The HRS-to-LRS ratio is assumed to be 200 in this study. The four levels of resistance and conductance values in a 2-bit MLC are listed in Table I. The four different levels of resistances LRS, MRS2, MRS1, and HRS represent four logic states 3, 2, 1, and 0, respectively.

Assume that all the resistance levels follow the log-normal distribution in Eq. (1) with the same σ as discussed in Section II. The conductance G' follows the distribution:

$$G' = G_0 \cdot e^{-\theta} \qquad \theta \sim N(0, \sigma^2), \tag{2}$$

where G_0 is a target conductance value of ReRAM. θ is the same as in Eq. (1). We use variance D to measure the deviation of the weights. For example, as shown in Fig. 3(b), "22" and "33310" are coded in the binary and unary coding, respectively, with 2-bit MLCs. Their variances D are shown in Eqs. (3) and (4), respectively.

$$\begin{aligned} D(22)_b &= D(4^1 \cdot 2 \cdot e^{-\theta_1} + 4^0 \cdot 2 \cdot e^{-\theta_2}) \\ &= [(4^1 \cdot 2)^2 + (4^0 \cdot 2)^2] \cdot D(e^{-\theta}) = 68 \cdot D(e^{-\theta}). \end{aligned} \tag{3}$$



LRS	⑤	②	①	④	③
MRS2	②	⑤	④	①	③
MRS1	②	⑤	④	③	①

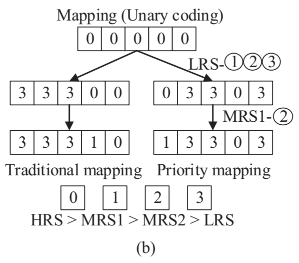


Fig. 4. An example of priority mapping with 2-bit MLCs in the unary coding. (a) Sort the resistance variation of 5 MLCs from small to large for LRS, MRS2, and MRS1 individually. (b) Mapping decimal value "10" to these 5 MLCs in the order of index.

$$\begin{aligned} D(33310)_u &= D(3 \cdot e^{-\theta_1} + 3 \cdot e^{-\theta_2} + \cdots + 0 \cdot e^{-\theta_5}) \\ &= [3^2 + 3^2 + 3^2 + 1^2 + 0^2] \cdot D(e^{-\theta}) = 28 \cdot D(e^{-\theta}). \end{aligned} \quad (4)$$

It shows that the variance of weight in the binary coding is much larger than that in the unary coding, as the variations on the MSBs magnify the weight deviation with MLC devices in the binary coding. Subsequently, unary coded weights have less variations with MLCs compared with binary coded weights, in the presence of resistance variation.

C. Proposed Variation-aware Priority Mapping Method

From Sections III-A and III-B, each bit in the proposed unary coding with MLCs has equivalent significance for representing a synaptic weight. Therefore, all the bit positions are interchangeable without influencing the weight value. Based on this observation, the priority mapping method is proposed in this section to further improve the NN accuracy.

In the priority mapping implemented with MLC, we first detect the variation of each ReRAM using the detection method in [10]. Assume that M ReRAMs are used to represent a synaptic weight. For each resistance state, each ReRAM is programmed to that target state and the actual resistance value is then sensed to get the deviation [10]. Then, we initialize both the positive and the negative crossbars to HRS. For each resistance state, the amounts of deviation of actual resistance value from the resistance state of all ReRAMs for a synaptic weight are then sorted in an ascending order. An order index from 1 to M is obtained for each ReRAM.

Note that, by Eq. (2), in the presence of variation, a larger conductance G_0 leads to a greater deviation. As a consequence, a larger conductance, corresponding to a lower resistance state, will have a greater impact on the MAC output. The unary bits with LRS are therefore first mapped to the ReRAMs with the smaller indexes for LRS. Then, the unary bit with MRS2 (or MRS1) is mapped to the ReRAMs with smaller index for MRS2 (or MRS1) and not occupied by the LRS's yet. By following this priority, we can reduce the impact of device variations on NN accuracy.

Fig. 4 gives an example of priority mapping. Assume that five 2-bit MLCs shown in Fig. 3(b) are used to represent a weight. First, we test and order the variations of each ReRAM for three different resistance states LRS, MRS2, and MRS1. The index of each resistance state is labeled from ① to ⑤ for these five MLCs as shown in Fig. 4(a). Then, all the ReRAMs are initialized to HRS. Fig. 4(b) shows the mapping process. In traditional mapping, the three 3s are mapped at the beginning of the unary coding, and then the 1. In priority mapping, the three 3s are mapped to the ReRAM with the smaller indexes, LRS-①②③. Then, the 1 is mapped. Since MRS1-① has already been occupied by an LRS, therefore, we turn to MRS1-②, which is not occupied. The rest ReRAMs are in initial HRS's, which do not need to be mapped again.

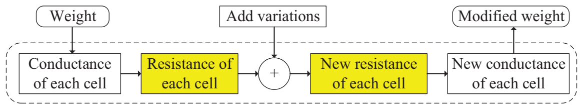


Fig. 5. Evaluation flow to simulate ReRAM variation in NN.

IV. Experimental Results

A. Experimental Setup

We evaluated our proposed method with 4 benchmarks, including MLP, LeNet, AlexNet, and VGG16. The first two are on MNIST dataset, while the last two are on CIFAR-10 dataset. We implemented the NN models with Tensorflow machine learning framework and ran them on a Nvidia TITAN-X GPU.

The experiments were tested by Monte-Carlo simulation method. We used σ shown in Eq. (1) to measure the degree of variation. We chose σ ranging from 0 to 1.0, as the σ in most fabrication processes does not exceed 1.0 [22]. The evaluation flow to simulate ReRAM variation in NN is shown in Fig. 5. We emulated the process of mapping the quantized weights to the corresponding ReRAM by setting the resistance of the ReRAM to a certain level, and then adding a log-normal distributed variation to the resistance. Afterwards, we converted the resistance, containing the variation, into conductance and updated the corresponding weight. Finally, we conducted inference with the modified model on the test dataset to get the classification accuracy. The whole evaluation flow is based on cell's resistance, making the results more accurate.

B. Results and Analysis

To implement the unary coding, the ReRAM-based NN architecture shown in Fig. 1 can be employed with little modification. Since the significance of each bit is equivalent in the unary coding, the shifter is no longer needed, and the "shifter + adder" can be replaced by a single adder. The output of each bitline, after passing the ADC, can be added directly. Apart from these, other modules like DAC, ADC, and S&H are consistent with the traditional architecture shown in Fig. 1. The input data to the DACs and the output data of the adder are binary coded. Only the weights are unary coded.

We calculated the power and area of an in-situ multiply-accumulate (IMA) unit [5] based on the data in ISAAC [5]. The accuracy loss is the difference between the accuracy at $\sigma = 0.8$ and the ideal one of 0.9923. Different from

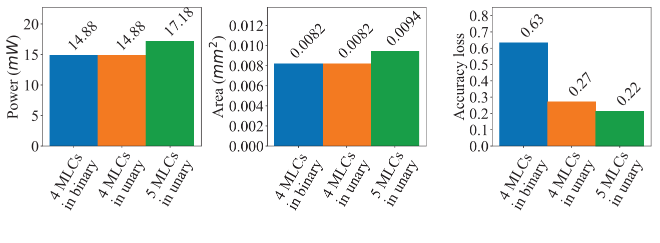


Fig. 6. Power, area, and accuracy loss ($\sigma = 0.8$) of LeNet on MNIST of different architectures using binary coding and unary coding. The weights are all based on 2-bit MLC. The horizontal axis indicates how many ReRAMs are used to represent a weight.

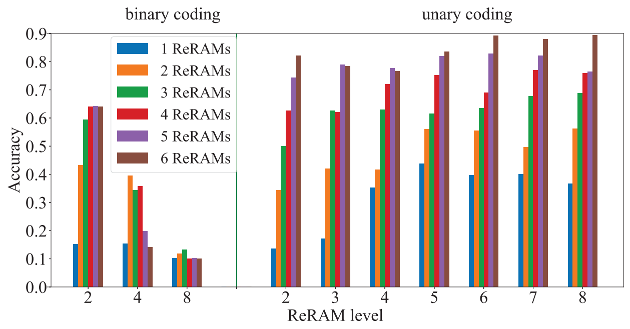


Fig. 7. Accuracy of LeNet on MNIST influenced by ReRAM number and ReRAM level in the binary coding (left) and the unary coding (right), respectively, when $\sigma = 0.8$.

ISAAC, we use four 2-bit MLCs in the binary coding as our baseline, since 8-bit quantized weights can almost reach the accuracy of full precision quantization in NNs. Therefore, the crossbar size, the number of S&H modules, and the frequency of the ADC change accordingly, whose power and area are approximately proportional to the number of ReRAM used to represent a weight. As shown in Fig. 6, the unary coding and the binary coding using four 2-bit MLCs have the same power and area, while the unary coding has lower accuracy loss. If we use five 2-bit MLCs to represent a weight, an even lower accuracy loss can be obtained. Nevertheless, the reduced accuracy loss comes at the cost of higher power and area. As shown in Fig. 6, an 15.47% increase of power and 15% increase of area can further reduce the accuracy loss by 21%. Therefore, if we pursue lower accuracy loss, we can use five 2-bit MLCs to represent a synaptic weight at the cost of higher power and area.

Fig. 7 illustrates the influence of ReRAM number and ReRAM level on the accuracy of LeNet. The accuracy is tested with a σ of 0.8. ReRAM number is the number of ReRAMs used to represent a weight. ReRAM level is the number of levels of each ReRAM cell. An ReRAM of 2 levels corresponds to an SLC, while ReRAMs of 4 and 8 levels correspond to 2-bit and 3-bit MLCs, respectively. In the unary coding, we do not need to set the ReRAM level as a power of 2. Therefore, ReRAMs of 3, 5, 6, and 7 levels are also applicable, making the use of the device more flexible. However, with the binary coding, only ReRAMs of 2, 4, and 8 levels can be used.

In general, with the same ReRAM number and ReRAM level (shown as the same color in Fig. 7), for most measurement points, the unary-based implementation gets higher accuracy than the binary-based one. It shows significant improvement when the unary coding is employed, compared with the binary coding. As shown in Fig. 7, the accuracy of both the unary coding and the binary coding drop significantly, as ReRAM number decreases. This is because, as the number of devices representing a weight is reduced, the resolution of a weight will decrease. Moreover, using a higher ReRAM level makes the accuracy in the unary coding almost unchanged, but resulting in a decrease in the binary coding. This has been explained in Section III-B that the unary coding is less disturbed by the MLC than the binary coding.

Fig. 8 shows the accuracy of AlexNet on CIFAR-10 by varying σ with 2-bit MLCs. Five ReRAMs are used to represent a weight. "Unary+Priority+DVA" means the priority mapping is applied together with the DVA training proposed in [12]. "Binary+DFP+DVA [12]" is the method proposed in [12]. We first trained AlexNet with log-normal distributed noise to get a robust model. Qualitatively speaking, the greater the noise added during the training process, the stronger the model's tolerance to variation. Therefore, we need to add noise during the training phase as much as possible. However, this will make the accuracy of the model drop even if there is no variation. In order to guarantee that the accuracy loss is less than 1% compared with the ideal accuracy of 89.76%, we added the noise with $\sigma = 0.4$. In this case, an accuracy of 89.15% was obtained by the DVA training.

From Fig. 8, when $\sigma = 0.8$, the traditional mapping in the unary coding only achieves an accuracy of 20.21%. With "DVA", the traditional mapping gets a 22.7% improvement in accuracy, but is still 22.53% lower than the accuracy obtained by using the proposed priority mapping. This means that the priority mapping can provide more improvement than "DVA". When the priority mapping is combined with "DVA", we get no accuracy loss at $\sigma = 0.6$. Compared with [12], although we have a different ideal accuracy, shown by the dotted line, we get only 3.1% accuracy loss at $\sigma = 0.8$, while [12] has an accuracy decrease of 13% at $\sigma = 0.5$.

It is worth mentioning that the variation detection required by the priority mapping requires extra testing cost. Detecting the variation of all cells takes a long time. As can be seen from the "Unary+DVA" curve in Fig. 8, when σ is less than 0.4, only using unary coding and DVA can achieve a accuracy loss of only 1.7%. This method does not require variation detection, which can eliminate the extra testing cost. Therefore, we can use "Unary+DVA" first if the variation is small.

The overall improvement among four NNs, MLP, LeNet, AlexNet, and VGG16, is listed in Table II. The accuracy is tested when σ is 0.8, which is relatively large for ReRAMs. The "Ideal" row gives the ideal accuracy when no variation is injected. The unary coding methods use five 2-bit MLCs to represent a weight, while the binary coding method uses 4

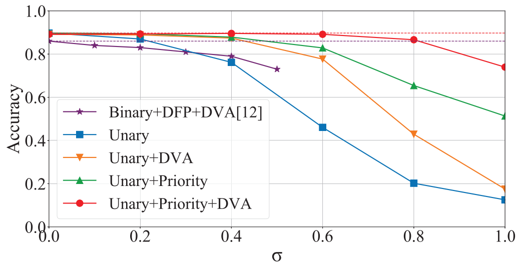


Fig. 8. Accuracy of AlexNet on CIFAR-10 dataset by varying σ.

TABLE II
ACCURACY OF IDEAL, BINARY CODING, UNARY CODING, UNARY CODING+DVA, UNARY CODING+PRIORITY MAPPING, AND UNARY CODING+PRIORITY MAPPING+DVA AT $\sigma = 0.8$ AMONG FOUR NNS.

Neural Network	MLP	LeNet	AlexNet	VGG16
Ideal	98.13%	99.23%	89.75%	93.42%
Binary	64.11%	66.51%	23.01%	10.66%
Unary	91.73%	78.17%	20.12%	12.47%
Unary+DVA	97.24%	97.66%	42.91%	18.83%
Unary+Priority	96.21%	95.76%	65.44%	81.40%
Unary+Priority+DVA	98.07%	98.78%	86.65%	87.94%

SLCs. Generally, a more complex NN results in a larger loss in accuracy. However, we only get 0.45% and 5.48% accuracy loss on LeNet and VGG16 for the "Unary+Priority+DVA" method even with a large device variation of $\sigma = 0.8$.

From "Binary" to "Unary+Priority+DVA", the accuracy improvements are 33.96%, 32.27%, 63.64%, and 77.28% for the four NNs, respectively. All the four NNs have great accuracy improvements with the proposed method. Among them, VGG16 has the biggest improvement, as it has the lowest accuracy in the binary coding. MLP and LeNet have lower improvement because they have higher accuracy in the binary coding, as their dataset and networks are much simpler.

V. CONCLUSION AND FUTURE WORKS

ReRAM crossbars can be used to accelerate NN inference. However, the immature fabrication process of the ReRAM devices leads to severe resistance variations, which degrades the accuracy of the ReRAM-based NN accelerators. We propose a new unary coding method to represent synaptic weights for achieving reliable ReRAM crossbar-based NNs. A variation-aware priority mapping method is also proposed to work together with unary coding to further reduce the NN accuracy loss. Our experimental results show that the proposed method makes the ReRAM-based NN accelerators more tolerant to large device variations.

Despite of the significantly improved NN accuracy with tolerance to the device variations, the unary coding requires more ReRAMs and induces higher hardware overhead for representing the same weight precision as the binary coding. To solve this problem, several techniques can be explored in future in compliance with the unary coding, such as hybrid coding and layer-wise adaptive weight precision.

REFERENCES

[1] Y. Sun *et al.*, "Energy-efficient nonvolatile SRAM design based on resistive switching multi-level cells," IEEE Transactions on Circuits and Systems II (TCAS-II), vol. 66, no. 5, pp. 753-757, May 2019.
[2] S. Kvatinsky *et al.*, "MAGIC-memristor-aided logic," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 61, no. 11, pp. 895-899, November 2014.
[3] S. R. Lee *et al.*, "Multi-level switching of triple-layered TaOx RRAM with excellent reliability for storage class memory," Symposium on VLSI Technology (VLSIT), pp. 71-72, June 2012.
[4] P. Chi *et al.*, "PRIME: a novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), pp. 27-39, June 2016.
[5] A. Shafiee *et al.*, "ISAAC: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars," ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), pp. 14-26, June 2016.
[6] L. Song, X. Qian, H. Li and Y. Chen, "Pipelayer: a pipelined ReRAM-based accelerator for deep learning," IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 541-552, February 2017.
[7] Y. Chen *et al.*, "Dadiannao: a machine-learning supercomputer," 47th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 609-622, December 2014.
[8] H. Li *et al.*, "Variation-aware, reliability-emphasized design and optimization of RRAM using SPICE model," Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1425-1430, March 2015.
[9] Y. Long, T. Na, S. Mukhopadhyay, "ReRAM-based processing-in-memory architecture for recurrent neural network acceleration," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 12, pp. 2781-2794, December 2018.
[10] B. Liu *et al.*, "Vortex: variation-aware training for memristor X-bar," 52nd ACM/EDAC/IEEE Design Automation Conference, pp. 1-6, June 2015.
[11] L. Chen *et al.*, "Accelerator-friendly neural-network training learning variations and defects in RRAM crossbar," Design, Automation & Test in Europe Conference & Exhibition (DATE) , pp. 19-24, March 2017.
[12] Y. Long, X. She and S. Mukhopadhyay, "Design of reliable DNN accelerator with un-reliable ReRAM," Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1769-1774, March 2019.
[13] A. Alaghi, W. Qian and J. P. Hayes, "The promise and challenge of stochastic computing," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 8, pp. 1515-1531, August 2018.
[14] S. Mohajer, Z. Wang and K. Bazargan, "Routing magic: performing computations using routing networks and voting logic on unary encoded data," International Symposium on Field-Programmable Gate Arrays, pp. 77-86, 2018.
[15] L. Gao, P. Chen and S. Yu, "Programming protocol optimization for analog weight tuning in resistive memories," IEEE Electron Device Letters, vol. 36, no. 11, pp. 1157-1159, November 2015.
[16] M. Hu *et al.*, "Dot-product engine for neuromorphic computing: programming 1t1m crossbar to accelerate matrix-vector multiplication," ACM 53rd annual design automation conference, pp. 19, 2016.
[17] B. Li, Y. Wang, Y. Wang, Y. Chen and H. Yang, "Training itself: mixed-signal training acceleration for memristor-based neural network," 19th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 361-366, January 2014.
[18] Z. Zhu *et al.*, "A configurable multi-precision CNN computing framework based on single bit RRAM," 56th Annual Design Automation Conference, June 2019.
[19] Y. Fang *et al.*, "Improvement of HfOx-based RRAM device variation by inserting ALD TiN buffer layer," IEEE Electron Device Letters, vol. 39, no. 6, pp. 819-822, June 2018.
[20] A. Fantini1 *et al.*, "Intrinsic switching variability in HfO2 RRAM," 5th IEEE International Memory Workshop, pp. 30-33, May 2013.
[21] B. Chen *et al.*, "Physical mechanisms of endurance degradation in TMO-RRAM," IEEE International Electron Devices Meeting (IEDM), pp. 12.3.1-12.3.4, December 2011.
[22] A. Grossi *et al.*, "Fundamental variability limits of filament-based RRAM," IEEE International Electron Devices Meeting (IEDM) , pp. 4.7.1-4.7.4, December 2016.