Reliable Memristive Neural Network Accelerators Based on Early Denoising and Sparsity Induction

Anlan Yu, Ning Lyu, Wujie Wen, Zhiyuan Yan
Department of Electrical and Computer Engineering
Lehigh University
{any218, nil418, wuw219, zhy6}@lehigh.edu

Abstract— Implementing deep neural networks (DNNs) in hardware is challenging due to the requirements of huge memory and computation associated with DNNs' primary operation—matrix-vector multiplications (MVMs). Memristive crossbar shows great potential to accelerate MVMs by leveraging its capability of in-memory computation. However, one critical obstacle to such a technique is potentially significant inference accuracy degradation caused by two primary sources of errors—the variations during computation and stuck-at-faults (SAFs). To overcome this obstacle, we propose a set of dedicated schemes to significantly enhance its tolerance against these errors. First, a minimum mean square error (MMSE) based denoising scheme is proposed to diminish the impact of variations during computation in the intermediate layers. To the best of our knowledge, this is the first work considering denoising in the intermediate layers without extra crossbar resources. Furthermore, MMSE early denoising not only stabilizes the crossbar computation results but also mitigates errors caused by low resolution analog-to-digital converters. Second, we propose a weights-to-crossbar mapping scheme by inverting bits to mitigate the impact of SAFs. The effectiveness of the proposed bit inversion scheme is analyzed theoretically and demonstrated experimentally. Finally, we propose to use L1 regularization to increase the network sparsity, as a greater sparsity not only further enhances the effectiveness of the proposed bit inversion scheme, but also facilitates other early denoising mechanisms. Experimental results show that our schemes can achieve 40%–78% accuracy improvement, for the MNIST and CIFAR10 classification tasks under different networks.

I. Introduction

Deep neural networks (DNNs) have shown great success in a wide range of tasks due to their strong representing capability and outstanding performance. As the volume of data increases and tasks become more difficult, the scales of DNNs often grow very large, resulting in increasing demand on computation resources. Despite recent efforts on accelerating DNNs using CPUs and GPUs, overwhelming computations needed by DNNs remain one of the primary obstacles to their widespread applications. Thus, fast and efficient DNN accelerating is still one significant research challenge.

Among different types of proposed DNN accelerators, memristive crossbar is one of the most promising techniques due to its fast reactions and low energy costs [1]. Matrix-vector multiplications (MVMs), which account for a significant fraction of DNN computation, are highly accelerated by mapping multiplications and additions to the relationships among voltages, resisters and currents in memristive crossbars. Hence, a memristive crossbar based implementation of MVMs simultaneously provides a high density storage as well as computation in memory [2] and diminishes data movements [3].

Despite the aforementioned advantages, memristive crossbars suffer from some non-ideal effects such as fabrication imperfection, programming imperfection, and conductance variations, all of which result in computational errors and performance degradation of DNNs when performing inferences [4][5]. To combat these errors, one option is to use error correction codes (ECCs) in memristive crossbars. For instance, error correction output codes (ECOCs), which replace one hot representations with specially designed codewords at the decision layer in classification tasks, have been proposed [6][7]. However, when neural networks become deep, ECOCs have limited effectiveness due to error propagation and accumulation from layer to layer. While using ECCs on all intermediate layers in addition to the decision layer can reduce error propagation [8][9], more crossbar resources are needed for redundancy bits in ECCs. For example, implementing a rate-2/3 code will add a redundant resistance for every 2 resistances [8]. To avoid the additional costs from redundancy introduced by using ECCs, in this paper we take a very different approach to combating these errors, that is, redundancy-free denoising at early layers. The main contributions of this paper are:

- We propose a minimum mean square error (MMSE) based intermediate layer denoising scheme, which

*This work is partially supported by National Science Foundation grants CCF-2006748, CCF-2011236 and ECCS-1835278.

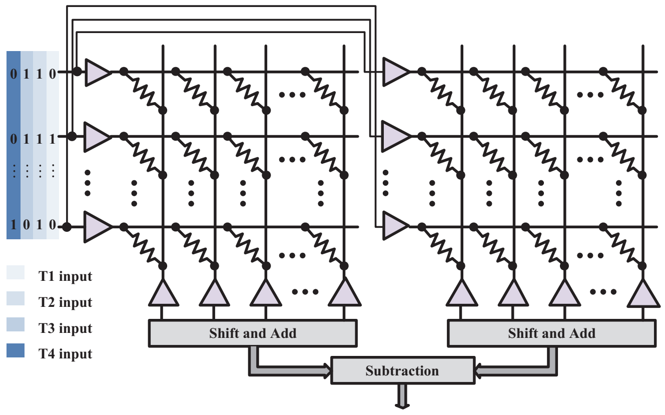


Fig. 1. Structure of crossbar-based memristive DNN accelerator. T1-T4 represent different timing.

uses MMSE estimates of the accurate outputs from intermediate layers to reduce the impact of variations and quantization errors introduced by low resolution analog-to-digital converters (ADCs). To the best knowledge of the authors, this is the first redundancy-free denoising scheme for intermediate layers.

- While previous works compensate for stuck-at-faults (SAFs) by assuming the knowledge of the defect map [10][11], which requires extra effort, a bit inversion scheme is proposed to mitigate the impact of SAFs without the knowledge of the defect map by taking advantage of the imbalance between stuck-at-1 (SA1) and stuck-at-0 (SA0) errors.
- We also propose a sparsity induction scheme so that training will result in more sparse weights. The increased sparsity of the network will enhance the effectiveness of the bit inversion scheme discussed above. Furthermore, we propose another early denoising mechanism by removing noisy multiply-accumulate (MAC) results of hidden nodes connected with all zero weights during inference.
- The effectiveness of both the sparsity induction and the bit inversion schemes is analyzed theoretically and demonstrated experimentally.

We present the experimental evaluations of our proposed schemes on the MNIST and CIFAR10 datasets using multilayer perceptron (MLP), Lenet 5 and Alexnet. Results show that the proposed schemes offer 40%–78% accuracy improvements. Our proposed scheme also shows 20%–70% accuracy improvements compared with the ECOC scheme in [7].

II. Background

A. Model of memristive DNN accelerator

We define the MVM operation in DNNs as

$$\mathbf{y} = \mathbf{W}^T\mathbf{x}, \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{N\times M}$ is the weight matrix, $\mathbf{x} \in \mathbb{R}^{N\times 1}$ is the input and $\mathbf{y} \in \mathbb{R}^{M\times 1}$ is the output.

In this work, two-level memristor cell crossbars are considered, in which each cell has only two states, namely high conductance state and low conductance state. Using a two-level memristor cell to represent one single bit of weights leads to negligible quantization error and higher robustness against variations, compared with using a multi-level memristor cell to represent one weight element. Furthermore, our proposed schemes in this paper can be extended to multi-level memristor crossbars. In each crossbar, binary representations of entries in $\mathbf{W}$ are mapped to conductances: '0's are mapped to low conductance states, and '1's are mapped to high conductance states. To implement the MVM in (1), two such crossbars are needed to deal with positive and negative values separately. Denote conductance matrices on positive crossbar and negative crossbar as $\mathbf{G}_+, \mathbf{G}_- \in \{0,1\}^{N\times MN_w}$, respectively, where a conductance unit is chosen such that high and low conductances are 1 and 0, respectively, without loss of generality. Then, traditional weights-to-crossbar mapping is given by

$$\begin{aligned}\mathbf{G}_{+,(i,jN_w:(j+1)N_w-1)} &= \begin{cases} Q(|w_{i,j}|), & \text{if } w_{i,j} > 0\\ \mathbf{0}, & \text{otherwise}\end{cases}\\ \mathbf{G}_{-,(i,jN_w:(j+1)N_w-1)} &= \begin{cases} Q(|w_{i,j}|), & \text{if } w_{i,j} \le 0\\ \mathbf{0}, & \text{otherwise}\end{cases}\end{aligned} \tag{2}$$

where $Q(|w_{i,j}|)$ is the N_w-bit binary representation of the (i,j)-th element in $\mathbf{W}$.

Entries of the input $\mathbf{x}$ are represented in a binary format, and then mapped to a matrix of voltages through 1-bit digital-to-analog converters (DACs) [3]. We denote the voltage matrix as $\mathbf{V} \in \{0,1\}^{N_i\times N}$, where a voltage unit is chosen such that high voltage equals 1, and low voltage equals 0, without loss of generality. Then the output currents $\mathbf{J}_+ \in \mathbb{N}^{N_i\times MN_w}$ and $\mathbf{J}_- \in \mathbb{N}^{N_i\times MN_w}$ are $\mathbf{J}_+ = \mathbf{V}\mathbf{G}_+$ and $\mathbf{J}_- = \mathbf{V}\mathbf{G}_-$, respectively. After that, the currents $\mathbf{J}_+$ and $\mathbf{J}_-$ are quantized by ADCs (with appropriate unit), and the quantized outputs are assembled by shifters and adders to give the digital output at each crossbar. The final output $\mathbf{y}$ is obtained by subtracting the results of negative crossbar from the results of the positive crossbar:

$$\mathbf{y} = f(\mathbf{J}_+) - f(\mathbf{J}_-), \tag{3}$$

where $f(*)$ represents the shift-and-add process. The structure of the memristive crossbar is shown in Fig. 1.

B. Types of errors for memristive crossbars

B.1 Variations

In memristive crossbar structure, dynamic errors usually come from conductance variation, sensing current variation, or resistance drifting. In this paper, we model these errors as additive white *Gaussian* noise (AWGN) on the current output at each column on the crossbar. Errors added on to the MVM results can be considered as a summation of *Gaussian* noise, which is also *Gaussian*. Hence the distorted currents $\mathbf{J}_{+,n}$ and $\mathbf{J}_{-,n}$ are given by $\mathbf{J}_{+,n} = \mathbf{J}_+ + \mathbf{n}_+$ and $\mathbf{J}_{-,n} = \mathbf{J}_- + \mathbf{n}_-$, where $\mathbf{n}_+, \mathbf{n}_- \sim \mathcal{N}(\mathbf{0}, \sigma_n^2\mathbf{I})$ are independent AWGN vectors.

B.2 Stuck-at-faults (SAFs)

SAFs happen when memristor cells are stuck at certain conductance levels and cannot be programmed, and they usually result from fabrication imperfection and aging. SAFs at the memristor cells can cause errors in crossbar-based MVM implementation during the DNN inference. SAFs are usually unbalanced, i.e. ratio of SA1 is much higher than that of SA0. We note that SAFs are not necessarily errors. For example, if a cell is stuck at 0, the cell is in error only if the correct state of the cell is 1. Since the cells considered in this work have binary states and are subject to SA0 and SA1 simultaneously, we model a cell as a random variable with three possible outcomes—SA0, SA1, and not stuck—with probabilities e_0, e_1, and $1 - e_0 - e_1$, respectively.

III. Reliable Crossbar Computing Schemes

A. Crossbar output denoising scheme based on MMSE

In this subsection, a crossbar output denoising scheme is proposed to relieve the distortion caused by variations in the intermediate layers based on linear minimum mean squared error estimation. According to MMSE, the output current for positive crossbar $\mathbf{J}_+$ is estimated by

$$\hat{\mathbf{J}}_+ = \frac{\sigma_j^2}{\sigma_j^2 + \sigma_n^2}\mathbf{J}_{+,n} + \frac{\sigma_n^2}{\sigma_j^2 + \sigma_n^2}\mathsf{E}[\mathbf{J}_+], \tag{4}$$

where σ_n^2 and σ_j^2 are the variances of $\mathbf{n}_+$ and $\mathbf{J}_+$, respectively, and $\mathsf{E}[\mathbf{J}_+]$ is the mean of $\mathbf{J}_+$. After that, each element in $\hat{\mathbf{J}}_+$ is assigned to its closest natural number to get the final hard decision output $\hat{\mathbf{J}}_{+,h}$. $\hat{\mathbf{J}}_{-,h}$ for negative crossbar is obtained in the same fashion.

Three parameters in Eq. (4) need to be determined: σ_n^2, σ_j^2 and $\mathsf{E}[\mathbf{J}_+]$. Since σ_n^2 is independent from both input and crossbar data, it can be measured via a one-time noise estimation process offline. σ_j^2 and $\mathsf{E}[\mathbf{J}_+]$ shall be characterized online as they are determined by the percentage of '1's and '0's on the crossbar and input data. In practice, we estimate these two values by directly using a batch of training data offline, as our results in Section IV show.

We denote the number of rows on each crossbar, percentage of '1's in $\mathbf{V}$ and percentage of '1's in $\mathbf{G}_+$ as r, $p_{1,v}$ and $p_{1,g}$, respectively. j represents a single element in $\mathbf{J}_+$, which is the MAC result for one single column. v is an element in $\mathbf{V}$ and g is an element in $\mathbf{G}_+$. We also assume the bit value on memristor cells and inputs are independent. Then random variable j can be obtained by $j = \sum_{k=1}^{r} v_k g_k$. Accordingly, $\mathsf{E}[j]$ is

$$\mathsf{E}[j] = \mathsf{E}[\sum_{k=1}^{r} v_k g_k] = r p_{1,v} p_{1,g}. \tag{5}$$

σ_j^2 can be calculated by

$$\sigma_j^2 = \mathsf{E}[j^2] - \mathsf{E}^2[j] = r^2 p_{1,v} p_{1,g}(1 - p_{1,v} p_{1,g}). \tag{6}$$

Based on Eqs. (5) and (6), $\mathsf{E}[\mathbf{J}_+]$ and σ_j^2 can be estimated and substituted in Eq. (4). Once the coefficients $\frac{\sigma_j^2}{\sigma_j^2+\sigma_n^2}$ and $\frac{\sigma_n^2}{\sigma_j^2+\sigma_n^2}$ are calculated before the inference, these two coefficients are constants and used for all the images during inference. Thus, the division operations in Eq. (4) are carried out offline and hence their complexity is negligible.

B. Bit inversion (BI) mapping scheme

In the conventional weights-to-crossbar mapping shown in Eq. (2), positions of negative/positive weights are set to 0 on positive/negative crossbars ideally, which results in a lot more '0's than '1's on crossbars, and SA1 will be more likely to cause errors.

For the above reason, and consider the fact that SA0 ratio is much smaller than SA1 ratio, we propose the following mapping rules to diminish the impact of SAFs:

$$\begin{aligned}
\mathbf{G}_{+,(i,jN_w:(j+1)N_w-1)} &= \begin{cases} \mathbf{1}, & \text{if } w_{i,j} > 0 \\ \overline{Q(|w_{i,j}|)}, & \text{otherwise} \end{cases} \\
\mathbf{G}_{-,(i,jN_w:(j+1)N_w-1)} &= \begin{cases} \mathbf{1}, & \text{if } w_{i,j} \le 0 \\ \overline{Q(|w_{i,j}|)}, & \text{otherwise} \end{cases}
\end{aligned} \tag{7}$$

where $\overline{Q(|w_{i,j}|)}$ is the one's complement of $Q(|w_{i,j}|)$. This mapping does not change the results of Eq. (3) and leads to more '1's on crossbars after the mapping.

Next we will evaluate the effectiveness of our proposed bit inversion (BI) mapping scheme theoretically. Let (a, b)

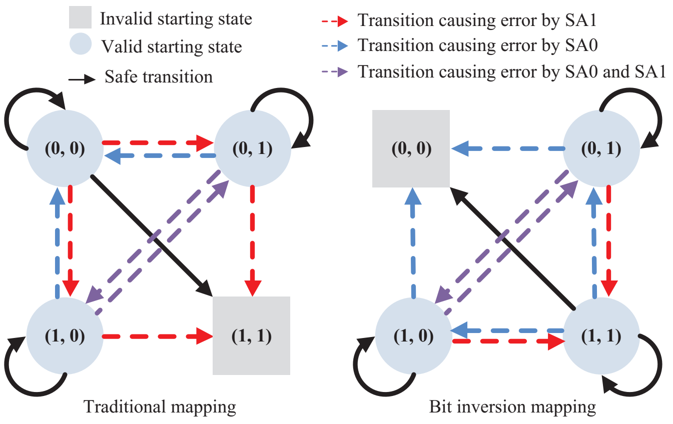


Fig. 2. State transition diagram with SAFs.

$$\begin{aligned}
P_{err,t} &= P_{(0,0)\to(1,0),t} + P_{(0,0)\to(0,1),t} + P_{(0,1)\to(1,1),t} + P_{(1,0)\to(1,1),t} + P_{(0,1)\to(0,0),t} + P_{(1,0)\to(0,0),t} + P_{(1,0)\to(0,1),t} + P_{(0,1)\to(1,0),t} \\
&= 2e_1(1-e_1)P_{(0,0),t} + e_1(1-e_0)P_{(0,1),t} + e_1(1-e_0)P_{(1,0),t} + e_0(1-e_1)P_{(0,1),t} + e_0(1-e_1)P_{(1,0),t} + e_1e_0P_{(1,0),t} + e_1e_0P_{(0,1),t} \\
&= (e_1 - e_0 - 2e_1^2 + e_0e_1)P_{(0,0),t} + e_1 + e_0 - e_1e_0
\end{aligned} \tag{8}$$

$$\begin{aligned}
P_{err,b} &= P_{(0,1)\to(1,1),b} + P_{(1,0)\to(1,1),b} + P_{(0,1)\to(0,0),b} + P_{(1,0)\to(0,0),b} + P_{(1,1)\to(1,0),b} + P_{(1,1)\to(0,1),b} + P_{(1,0)\to(0,1),b} + P_{(0,1)\to(1,0),b} \\
&= e_1(1-e_0)P_{(0,1),b} + e_1(1-e_0)P_{(1,0),b} + e_0(1-e_1)P_{(0,1),b} + e_0(1-e_1)P_{(1,0),b} + 2e_0(1-e_0)P_{(1,1),b} + e_1e_0P_{(1,0),b} + e_1e_0P_{(0,1),b} \\
&= (e_0 - e_1 - 2e_0^2 + e_0e_1)P_{(1,1),b} + e_1 + e_0 - e_1e_0
\end{aligned} \tag{9}$$

denote the state of two corresponding conductances on positive and negative crossbars, where a and b are values of the conductances on positive and negative crossbars, respectively. SAFs can cause transitions of states and lead to errors, as illustrated in Fig. 2. For both mapping schemes, states in blue are valid starting states and states in grey are invalid starting states, e.g. state $(1,1)$ is invalid for traditional mapping scheme in Eq. (2) because at least one of the state is 0 according to Eq. (2). Black solid arrows are safe transitions that no error occurs on the final subtraction result, while red, blue and purple dash arrows represent transitions that cause errors. For instance, transitions from $(0,0)$ to $(1,1)$ and from $(1,1)$ to $(0,0)$ do not cause errors on the subtraction result.

Let us denote the prior probability of (a,b) using traditional mapping scheme and BI mapping scheme as $P_{(a,b),t}$ and $P_{(a,b),b}$, respectively. The probability of state transitions is denoted as $P_{(a,b)\to(a',b')}$. A cell suffers from SA1 and SA0 with probability e_1 and e_0, respectively. The effective error rates of the traditional mapping scheme $P_{err,t}$ and the proposed BI mapping scheme $P_{err,b}$ are calculated in Eq. (8) and Eq. (9), respectively.

From Eq. (2) and Eq. (7), we observe that $P_{(0,0),t} = P_{(1,1),b}$ because the positions of $(0,0)$ states using traditional mapping scheme are exactly the same as the positions of $(1,1)$ states using proposed BI mapping scheme. Then the difference of $P_{err,t}$ and $P_{err,b}$ is calculated as

$$\begin{aligned}
P_{err,diff} &= P_{err,t} - P_{err,b} \\
&= 2P_{(0,0),t}(e_1 - e_0)(1 - e_1 - e_0).
\end{aligned} \tag{10}$$

Note that $1 - e_1 - e_0 \geq 0$ is the probability of a cell not being stuck, therefore $P_{err,diff} \geq 0$ and the proposed BI mapping scheme works no worse than the traditional mapping scheme as long as $e_1 > e_0$ (SA1 ratio is higher than SA0 ratio). Additionally, we observe that the higher $P_{(0,0),t}$ is, the higher the improvement is. Fig. 3 shows the error rates of the two mapping schemes and their difference versus $P_{(0,0),t}$. In the figure, it is assumed that $e_1 = 0.0904$ and $e_0 = 0.0175$ [4][10].

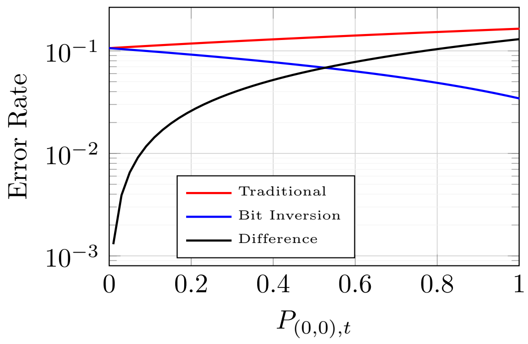


Fig. 3. Error rate comparison for the traditional mapping scheme and the proposed bit inversion mapping scheme.

C. Sparsity induction (SI) scheme

From both Fig. 3 and Eq. (10) we observe that a greater $P_{(0,0),t}$ leads to a greater impact for the BI scheme. Inspired by this, we use L1 regularization to improve the network sparsity during training to help reduce error rate caused by SAFs. Due to L1 regularization, training will result in more sparse weights and hence a greater $P_{(0,0),t}$. We refer to this scheme as sparsity induction (SI).

With greater sparsity in the network, some internal nodes in the fully connected layers and some feature maps in the convolutional layers are all zero because of the weights connected to them are all zero. With the knowledge of accurate weight in advance, we simply remove the MVM results of these nodes or feature maps:

$$y_i = \begin{cases} 0, & \text{if } \sum_{j=0}^{M} |w_{j,i}| = 0 \\ f(\mathbf{J}_{+,n})_i - f(\mathbf{J}_{-,n})_i, & \text{otherwise} \end{cases} \tag{11}$$

where y_i is the i-th element in MVM output $\mathbf{y}$. We note that this constitutes another early denoising mechanism, as this will eliminate the propagation of all types of errors through these outputs.

IV. Evaluation

A. Experimental setting

To evaluate the performance of our proposed computing schemes, we simulated the memristive crossbar model as explained in Section II with 3 NNs: multilayer perceptron (MLP), Lenet 5 and Alexnet to learn MNIST and CIFAR10 datasets. MNIST dataset is a 10-class handwritten digit dataset with 60,000 training images and 10,000 testing images. CIFAR10 is a 10-class color image dataset with 50,000 training images and 10,000 testing images. Table I shows the test accuracy without any error as well

TABLE I
Experimental Settings

Network	Dataset	Accuracy	Configuration
MLP	MNIST	98.81%	$784-256-256-256-10$
Lenet 5	MNIST	99.19%	$28\times 28-6c5-2s-16c5-2s-120-84-10$
Alexnet	CIFAR10	71.77%	$32\times 32-64c11-2s-192c5-2s-384c3-256c3-256c3-2s-10$

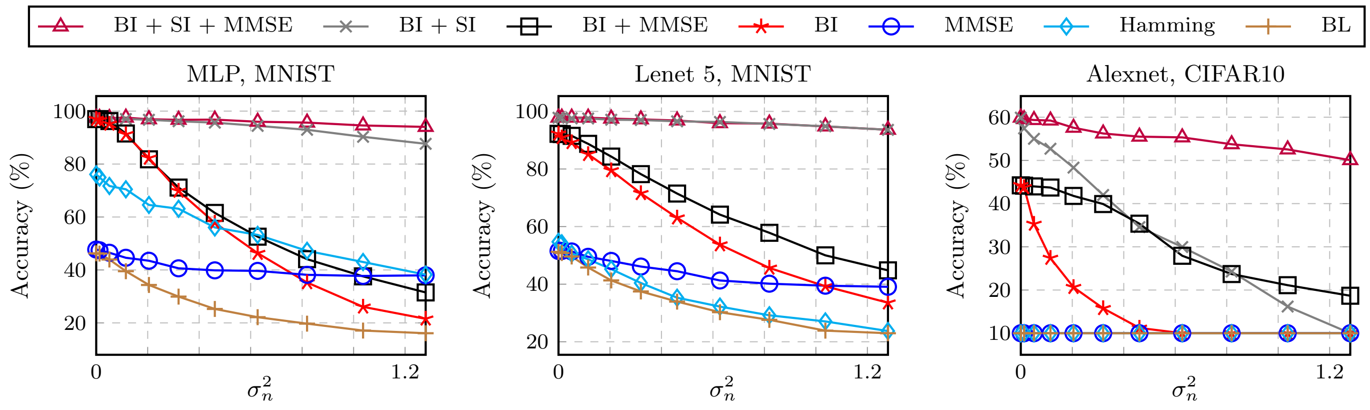


Fig. 4. Accuracy comparison with SAFs and variations.

as the dataset used, network configuration of these 3 networks. The default crossbar size is 128×128 and weights and inputs are represented as $N_w = 16$ bits and $N_i = 8$ bits, respectively. SAFs and variations are considered in the experiments. SA1 ratio and SA0 ratio are chosen as 0.0904 and 0.0175 [10]. Variations are modeled as AWGN $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma_n^2\mathbf{I})$ and σ_n^2 represents the variance of noise added on each column of MAC result considering binary representation on the crossbar.

B. Performance evaluation

B.1 Performance with both SAFs and variations

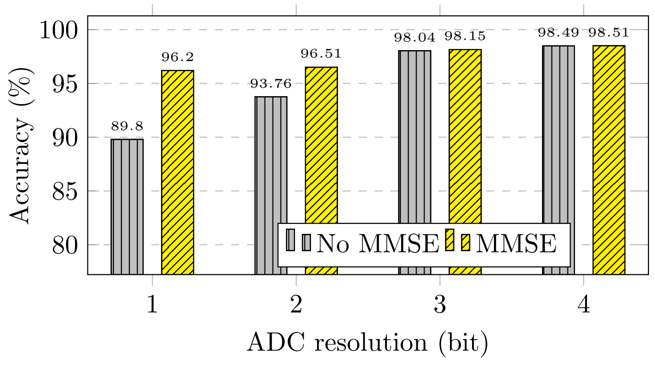
Taking both SAFs and variations into consideration during inference, seven different settings are compared for all three NNs: 1) use proposed BI mapping, SI, and MMSE in the intermediate layers, 2) only use proposed BI mapping and SI, 3) only use proposed BI mapping and MMSE, 4) only use proposed BI mapping, 5) only use proposed MMSE, 6) use Hamming ECOC on the decision layer and 7) baseline (BL), i.e., no mechanisms are used to combat the errors during inference. Fig. 4 shows the accuracy comparison. Among the seven settings, using BI mapping, SI, and MMSE simultaneously achieves the best accuracy and achieves 51%–78%, 46%–70% and 40%–50% accuracy improvements compared with the baseline for MLP, Lenet 5 and Alexnet, respectively. Using Hamming code on the decision layer increases the fault tolerant capability for MLP, but it does not provide enough protection for Lenet 5 and Alexnet. Due to the depth of both Lenet 5 and Alexnet, error propagation and accumulation are more pronounced and hence adding redundancy in the decision layer alone is not enough.

B.2 Evaluation with ADC resolution

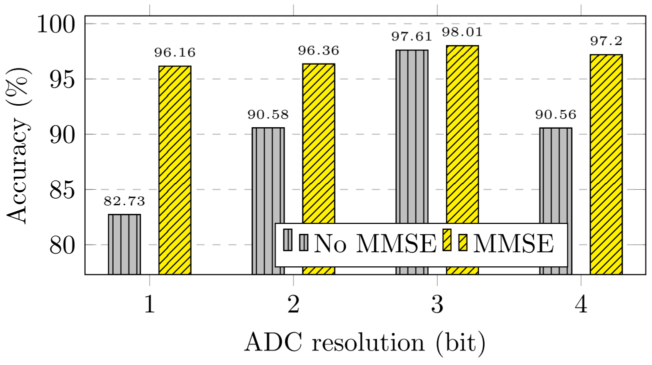
Though MMSE shows limited accuracy improvement for simpler tasks and smaller networks, i.e. MLP and Lenet 5 for MNIST, MMSE also helps reduce ADC resolution while maintaining the inference accuracy. Fig. 5 shows the accuracy of MLP learning MNIST versus ADC resolution with and without variations. Theoretically a 7-bit ADC is enough to represent MAC results with crossbar size 128×128. However, from Fig. 5(a) we observe that 3-bit ADCs are sufficient for MLP due to the sparse nature of weights without considering variations. As ADC resolution decreases to just 1 bit, baseline accuracy (without MMSE) decreases to 89.8%. With the help of MMSE, the inference accuracy recovers to 96.2%, which is even higher than the baseline accuracy with 2-bit ADC resolution. Considering variations, MMSE is able to cope with both variations and ADC resolution errors. When $\sigma_n^2 = 0.4608$, as shown in Fig. 5(b), the baseline accuracy shows a different trend compared with the case without variations, i.e. baseline accuracy first increases and then decreases as ADC resolution becomes better. This is because with the same noise level, coarsely grained ADCs help filter out small noise while finer grained ADCs include those small noise at the output of ADCs. We also observe that with the help of MMSE, 1-bit ADC suffers only less than 2% accuracy degradation compared with baseline accuracy for 3-bit ADCs, leading to smaller areas and lower energy costs.

C. Overhead discussions

In this subsection, we discuss the overhead introduced by the three proposed schemes. Since BI only changes



(a) MLP, MNIST, $\sigma_n^2 = 0$



(b) MLP, MNIST, $\sigma_n^2 = 0.4608$

Fig. 5. Accuracy comparison with different ADC resolutions.

the weights-to-crossbar mapping, no additional hardware overhead is required. As for the SI scheme, efforts are focused on sparse training and removing internal MAC results, extra computation is not needed, either. According to Eq. (4), MMSE computation only needs two multipliers and an adder, since coefficients $\sigma_j^2/(\sigma_j^2+\sigma_n^2)$ and $\sigma_n^2/(\sigma_j^2+\sigma_n^2)$ are pre-computed constants as discussed in Section III. In addition to combating the variations of memristive crossbar, MMSE can also filter out noise incurred by ADC quantization. This leads to a lower ADC resolution requirement without degrading the inference accuracy. With a lower ADC resolution, the cost of MMSE can be further reduced. Given the low bit-width multipliers and adders used in MMSE, the hardware implementation can be simplified as a few AND/XOR gates. Moreover, the overhead can be even smaller by selectively applying MMSE to important layers/nodes and using power 2 to approximate the parameters.

V. Conclusions

In this paper, we propose a set of schemes that can diminish the impact of both variations and SAFs for two-level memristive crossbar. To prevent variation propagation in the flow of inference and combat error caused by low ADC resolution, an MMSE based crossbar output denoising scheme is proposed. The BI mapping scheme is proposed to combat the asymmetric SAFs, and the effective error rate caused by SAFs is analyzed. By constraining the weight to increase sparsity in the network, internal output suppression can be applied to further mitigate error propagation. Experimental results show our schemes deliver 40%–78% accuracy improvements. The proposed schemes are adaptable to multi-level memristive crossbars. Future works will focus on extending the proposed schemes to more general scenarios.

References

[1] H.-S. P. Wong *et al.*, “Metal–oxide RRAM,” *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, 2012.

[2] L. Ni, Z. Liu, H. Yu, and R. V. Joshi, “An energy-efficient digital ReRAM-crossbar-based CNN with bitwise parallelism,” *IEEE Journal on Exploratory solid-state computational devices and circuits*, vol. 3, pp. 37–46, 2017.

[3] A. Shafiee *et al.*, “ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars,” *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 14–26, 2016.

[4] C.-Y. Chen *et al.*, “RRAM defect modeling and failure analysis based on march test and a novel squeeze-search scheme,” *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 180–190, 2014.

[5] C. Wang *et al.*, “Cross-point resistive memory: Nonideal properties and solutions,” *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 24, no. 4, pp. 1–37, 2019.

[6] T. Liu and W. Wen, “Making the fault-tolerance of emerging neural network accelerators scalable,” in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, IEEE, 2019, pp. 1–5.

[7] T. Liu *et al.*, “A fault-tolerant neural network architecture,” in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, IEEE, 2019, pp. 1–6.

[8] Q. Lou *et al.*, “Embedding error correction into crossbars for reliable matrix vector multiplication using emerging devices,” in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, 2020, pp. 139–144.

[9] B. Feinberg, S. Wang, and E. Ipek, “Making memristive neural network accelerators reliable,” in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, IEEE, 2018, pp. 52–65.

[10] L. Chen *et al.*, “Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar,” in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2017, pp. 19–24.

[11] Z. He, J. Lin, R. Ewetz, J.-S. Yuan, and D. Fan, “Noise injection adaption: End-to-end ReRAM crossbar nonideal effect adaption for neural network mapping,” in *Proceedings of the 56th Annual Design Automation Conference*, 2019, pp. 1–6.