Gradient-based Bit Encoding Optimization for Noise-Robust Binary Memristive Crossbar

Youngeun Kim^{*,1}, Hyunsoo Kim^{*,2}, Seijoon Kim^{2}, Sang Joon Kim^{2}, Priyadarshini Panda^{1}
^{1}Department of Electrical Engineering, Yale University, USA
^{2}Samsung Advanced Institute of Technology, South Korea
{youngeun.kim, priya.panda}@yale.edu, {hs0128.kim, seijoon.kim, sangjoon0919.kim}@samsung.com

***Abstract*—Binary memristive crossbars have gained huge attention as an energy-efficient deep learning hardware accelerator. Nonetheless, they suffer from various noises due to the analog nature of the crossbars. To overcome such limitations, most previous works train weight parameters with noise data obtained from a crossbar. These methods are, however, ineffective because it is difficult to collect noise data in large-volume manufacturing environment where each crossbar has a large device/circuit level variation. Moreover, we argue that there is still room for improvement even though these methods somewhat improve accuracy. This paper explores a new perspective on mitigating crossbar noise in a more generalized way by manipulating input binary bit encoding rather than training the weight of networks with respect to noise data. We first mathematically show that the noise decreases as the number of binary bit encoding pulses increases when representing the same amount of information. In addition, we propose Gradient-based Bit Encoding Optimization (GBO) which optimizes a different number of pulses at each layer, based on our in-depth analysis that each layer has a different level of noise sensitivity. The proposed heterogeneous layer-wise bit encoding scheme achieves high noise robustness with low computational cost. Our experimental results on public benchmark datasets show that GBO improves the classification accuracy by $\sim 5-40\%$ in severe noise scenarios.**

***Index Terms*—Memristive crossbar, Binary input encoding, Deep neural network, Binary neural network**

I. Introduction

Memristive crossbars have emerged as a key component for implementing energy-efficient neural network accelerators by performing Matrix-Vector-Multiplication (MVM) in analog domain [1] using emerging Non-Volatile-Memory (NVM) devices. The efficiency of the memristive crossbar can be significantly amplified by implementing Binary-Weight Neural Networks (BWNNs) [6], which consist of binary weights and multi-bit activations. Encoding binary values in NVM devices is much simpler than multi-level representation, therefore, BWNNs can be easily mapped on a binary memristive crossbar [11]. The multi-bit activations of BWNNs can be implemented in an efficient way by converting multi-level activation into temporal binary pulses as demonstrated in [12]. This approach induces less overhead from analog-to-digital conversion and I/O than using multi-level input activation [11]. Overall, BWNNs can be implemented on a binary memristive crossbar with the temporal binary bit encoding scheme.

However, due to the analog nature, the binary memristive crossbar suffers from non-idealities (*i.e.*, noise), such as, device variations, analog-to-digital converter (ADC) and digital-to-analog converter (DAC) error, sneak path, and parasitic resistances [2], [10]. The various types of noise are amplified through multiple layers of neural network implementation on crossbars, resulting in severe performance degradation. To address this problem, the recent algorithm-level approaches train the weight parameters of neural networks in correspondence with the noise data. Chakraborty *et al.* [4] collect SPICE simulation data by considering both linear and non-linear noise in crossbars. However, it is difficult to collect noise data for each crossbar. Therefore, previous noise-aware training methods can be ineffective in large-volume manufacturing where each crossbar has a large device-to-device and circuit-to-circuit variation. Moreover, we assert that there is more room to improve the robustness of the memristive crossbar.

In this paper, we introduce a new perspective on mitigating the inherent noise of a binary memristive crossbar. We solely focus on manipulating the number of pulses of input bit encoding for a memristive crossbar. The proposed method can improve noise robustness significantly without weight parameter tuning, and remains compatible with previous works. Our work is based on two insights: *(1) The inherent noise of a memristive crossbar can be mitigated by increasing the number of input pulses (see Section II-B). (2) Each layer has a different level of noise sensitivity (see Section II-C).* Considering these observations, we present Gradient-based Bit encoding Optimization (GBO) which optimizes layer-wise pulse length for input bit encoding. Before our main training phase, the networks are pre-trained with cross-entropy loss. Then, we fix the weights of networks and only train learnable parameters that represent the importance of each bit encoding strategy for each layer. The objective function consists of two terms with distinct purposes. One is for improving classification accuracy and the other term is for reducing the computational cost. The learnable parameters are optimized with two loss terms and find a saddle point. During inference, we select the bit encoding scheme which has the maximum learnable parameter value in each layer.

One important design parameter of GBO is the search space interval of the pulse length set. The large interval of pulse strategies can induce a huge latency cost for the system. Therefore, to enable a more fine-grained search space in the

* These authors have contributed equally to this work.
† This work was carried out while Youngeun Kim worked as an intern at Samsung Advanced Institute of Technology, South Korea.

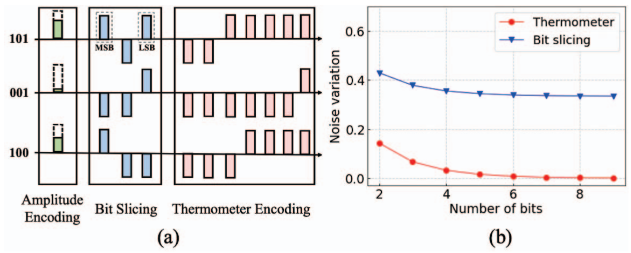

Fig. 1. (a) Illustration of three input bit encoding schemes. (b) Noise variation with respect to the number of bits. Here, we set baseline noise variation as 1.

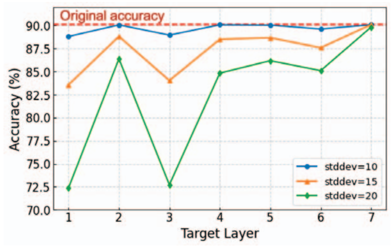

Fig. 2. Layer-wise noise sensitivity analysis. We use VGG9 architecture trained on CIFAR10. Target layer (x-axis) denotes the only layer where Gaussian noise $N(0, \sigma^2)$ is added.

pulse length set, we propose Pulse Length Approximation (PLA) which allows variable fine-grained pulse length while preserving similar information. We found that the activation in deep layers converges to -1 or 1 because of consecutive batch normalization [9] and bounded non-linear function (*e.g.*, Tanh). Based on this, we manipulate the pulse length by adding or subtracting pulses towards -1 or 1 values.

In summary, our contributions are as follows: (1) To the best of our knowledge, this is the first work that analyzes the effect of bit encoding on crossbar noise. (2) The proposed GBO can automatically find the optimal bit encoding scheme. Thus, compared to a heuristic approach (*e.g.*, manually selecting bit encoding for each layer), our work provides a more general solution to various network configurations. (3) Our experimental results show that GBO can effectively mitigate crossbar noise on CIFAR10. The proposed GBO improves the classification accuracy by $\sim 5-40\%$ in severe noise scenarios.

II. Preliminary

A. Crossbar Array Implementation

In this paper, we use binary weights due to their memory efficiency and the maturity of implementation on NVM devices. For activations, we encode N-bit information into sequential binary bit pulses (see Section II-B for details). Thus, our networks have multi-bit activation on a binary memristive crossbar. We use a hyperbolic tangent function as an activation function to confine the input activation range into [-1, 1]. We assume that a simplified Gaussian noise is added after the MVM operation. For a crossbar that stores weight matrix W, we can formulate an output current feature vector as:

$$o = Wx + N(0, \sigma^2), \tag{1}$$

where, x is the input voltage vector, and $N(0, \sigma^2)$ is Gaussian noise from a memristive crossbar.

B. Input Bit Encoding Schemes for Memristive Crossbar

In general, the input voltage of the crossbar (N-bit) can be converted into pulses in different ways as shown in Fig. 1(a). One of the most commonly used encoding techniques is encoding an input voltage to the amplitude of 2^N voltage levels. However, this will require a high-precision DAC to interface with the crossbar that incurs huge area and energy consumption. Moreover, because of the wide voltage range for multi-bit representation, synapse cells should guarantee a linear I–V curve over the range of input voltage [5], [8]. Compared to multi-level input representation, binary input encoding brings less circuit overhead from analog-to-digital conversion [11]. *Bit slicing* [3] is one of the most popular binary encoding techniques, which represents pulses with the same sequence of input bits. Unfortunately, weighted summation (based on bit position) induces more noise to the output current (Eq. 2). To address this, we utilize *Thermometer coding* [14] where the number of positive pulses is proportional to a representation level. We show that *Thermometer coding* can achieve higher noise robustness than the bit slicing approach in the following sub-sections.

Noise analysis of binary bit encoding: We analyze the effect of two binary bit encoding schemes, *i.e.*, *Bit slicing* and *Thermometer Coding*. As we convey b-bits through multiple time-steps (*i.e.*, pulses), Gaussian noise is accumulated. We assume that Gaussian noise is independent at each time-step. According to the bit encoding scheme, we reformulate the equation for MVM operation (Eq. 1) with crossbar noise.

Bit slicing generates p pulses which have the same sequence with bit representation. Therefore, each pulse has a different contribution to the output result according to its bit position:

$$o = \frac{1}{\sum_{i=0}^{p-1} 2^i} \sum_{i=0}^{p-1} 2^i W x_i + N(0, \frac{\sum_{i=0}^{p-1} (2^i)^2}{(\sum_{i=0}^{p-1} (2^i))^2} \sigma^2). \tag{2}$$

The first term represents the MVM results, and the second term shows the accumulated noise.

Thermometer coding represents p-level information with $p-1$ pulses, where p can be any positive integer. The output current is accumulated across p pulses:

$$o = \frac{1}{p} \sum_{i=0}^{p-1} W x_i + N(0, \frac{1}{p} \sigma^2). \tag{3}$$

Through mathematical analysis, we can observe the following: (1) We quantitatively compare the normalized noise variance of the two bit encoding schemes with respect to b-bit information. In Fig. 1(b), we present the trend of noise variation with respect to the number of bits. Here, *Thermometer coding* shows higher robustness compared to *Bit slicing* when they represent same bit information. Therefore, we use *Thermometer coding* as a baseline bit encoding scheme to build more noise robust crossbars. (2) For both encoding schemes, the noise variance is inversely proportional to the number of pulses. This indicates that noise can be mitigated by increasing the number of pulses when representing the same amount of bit information. This key observation is leveraged in our layer-wise variable bit encoding optimization.

C. Layer-wise Noise Sensitivity Analysis

Uniformly increasing the latency across all layers brings longer processing time for crossbars. To efficiently take advantage of the noise suppression effect, we vary the length of bit coding in each layer according to the noise robustness. This is based on the observation that each layer has different noise sensitivity. In Fig. 2, we present noise sensitivity of each layer by adding Gaussian noise $N(0, \sigma^2)$ to the output feature map at that layer only. Noise in different layers has different impact on classification accuracy. Since different architectures have different layer configurations, a heuristic approach for selecting bit encoding for each layer is not a desirable solution. Therefore, a general framework that can support variable bit encoding scheme is required.

III. Methodology

In this section, we first introduce Gradient-based Bit encoding Optimization (GBO), which finds optimal bit encoding for a robust memristive crossbar implementation. Then, we present Pulse Length Approximation (PLA) that enables fine-grained search space for gradient-based optimization.

A. Learning to Optimize Input Bit Encoding Strategy

Our objective is to optimize a layer-wise bit encoding scheme by using gradient-based optimization. Let each layer l of a network has a set Ω which consists of m different pulse scaling factors, thus $\Omega^l = \{n_0^l, n_1^l, \ldots, n_{m-1}^l\}$, $n \in \mathbb{Z}$. With the pulse scaling factor n, we can reformulate Eq. 3 as follows:

$$o_n = \frac{1}{np}\sum_{i=0}^{np-1} W\hat{x_i} + N(0, \frac{1}{np}\sigma^2), \tag{4}$$

where, $\hat{x_i}$ is the modified input binary pulse according to the pulse scaling factor. By doing this, we can preserve the original output while reducing noise by $1/n$. A conceptually naïve way to increase the number of pulses is an ensemble which repeats the same coding by n times (where, n is a positive integer).

For each coding scheme (*i.e.*, pulse scaling factor), we assign learnable parameters $\lambda_k^l, k \in \{0, \ldots, m-1\}$. In order to generate the probability representation of each coding scheme, we compute α_k^l based on λ_k^l as $\alpha_k^l = \frac{e^{\lambda_k^l}}{\sum_{z\in\Omega} e^{\lambda_z^l}}$. Therefore, during training, we average the impact of each coding scheme based on α_k^l value. We can represent an output activation as:

$$o^l = Wo^{l-1} + \sum_k \alpha_k^l N(0, \frac{1}{n_k^l p}\sigma^2). \tag{5}$$

For the last layer, we calculate the classification probability using a softmax function. The objective function consists of two terms as follows:

$$L = L_{ce} + \gamma \sum_l \sum_k \alpha_k^l n_k^l p. \tag{6}$$

The first term L_{ce} is the standard cross-entropy loss to maximize the classification accuracy, and the second term is a regularization term to minimize latency budget across all layers. The parameter γ is a balancing parameter between two losses. Overall, by optimizing Eq. 6, the model finds a saddle point between two distinct optimization directions. The learnable parameter λ_k^l is updated following gradient calculation:

$$\Delta\lambda_k^l = \frac{\partial L}{\partial \lambda_k^l} = \frac{L}{\partial o^l}\frac{\partial o^l}{\partial \alpha_k^l}\frac{\partial \alpha_k^l}{\partial \lambda_k^l}. \tag{7}$$

Here, all derivative terms are continuous and differentiable. Therefore, we can update the bit encoding control parameters as $\lambda_k^l = \lambda_k^l - \eta\Delta\lambda_k^l$, where η denotes the learning rate. During inference, we select the bit encoding strategy which has the maximum importance score λ_k^l, *i.e.*, $n_{optimal}^l = \text{argmax}_{n_k\in\Omega}\lambda_k^l$. Importantly, we first train the weight parameters with cross-entropy loss. After that, we only train learnable parameters λ_k^l from a pre-trained model. This ensures a stable convergence.

B. Pulse Length Approximation (PLA)

In our GBO, we use the ensemble strategy for mitigating crossbar noise. However, the ensemble strategy is only available for positive integer n, which can increase latency significantly. For example, given that we use 8-pulse *Thermometer coding* (p = 8), the number of pulses of the ensemble cases can be $\{8, 16, 24, \ldots\}$. Also, this coarse search space can induce a sub-optimal solution for gradient-based training with the same latency budget. Thus, there is a need to enable more fine-grained pulse search space.

To address this, we present Pulse Length Approximation (PLA). The main idea is to allow flexible pulse length (*i.e.*, float n) by approximating the original pulse information. Specifically, we approximate the modified input signal $\hat{x_i}$ towards -1 or 1 according to its sign. This strategy is easy to be implemented by adding or removing pulses from the original pulse code. Our PLA is based on the observation that the activation in deep layers converges to [-1, 1]. This is because a BN layer span the range of activation by normalization, and a bounded non-linear activation function (*e.g.*, Tanh) limits the range of activation into [-1, 1]. Note that, the approximated pulses cannot exactly represent the original representation levels that leads to approximation error. To empirically resolve such concern, in Table I, we report the classification accuracy with respect to n pulses (i.e., denoted as PLA$_n$). The results show that the performance degradation caused by the approximation error is negligible.

IV. Experiments

A. Experimental Setup

We evaluate our method on CIFAR10 with VGG 9 architecture [13]. Our implementation is based on PyTorch. For the pre-training stage, we use standard SGD with momentum 0.9, weight decay 5e-4. The base learning rate is set to 1e-3 and we use step-wise learning rate scheduling with a decay factor 10 at 50%, 70%, and 90% of the total number of epochs. Here, we set the total number of epochs to 60 for both datasets. We also quantize the activation and weight to 9-levels and binary representation, respectively, during pre-training. Thus, activations are converted to the *Thermometer coding* with 8 pulses. For GBO training, we use ADAM optimizer with learning rate 1e-4. We only train λ_k^l parameter during 10

TABLE I
RESULTS ON CIFAR10 WITH VGG9 ARCHITECTURE

Method	Noise σ	# pulses in each layer	Avg.# pulses	Acc. (%)
Baseline	10	[8, 8, 8, 8, 8, 8, 8]	8	83.94
PLA$_{10}$	10	[10, 10, 10, 10, 10, 10, 10]	10	85.38
PLA$_{12}$	10	[12, 12, 12, 12, 12, 12, 12]	12	85.58
PLA$_{14}$	10	[14, 14, 14, 14, 14, 14, 14]	14	86.24
PLA$_{16}$	10	[16, 16, 16, 16, 16, 16, 16]	16	88.27
GBO ($\sim$PLA$_{10}$)	10	[10, 10, 8, 10, 10, 4, 6]	9.71	86.36
GBO ($\sim$PLA$_{14}$)	10	[16, 6, 12, 6, 10, 14, 16]	14.85	88.27
Baseline	15	[8, 8, 8, 8, 8, 8, 8]	8	62.27
PLA$_{10}$	15	[10, 10, 10, 10, 10, 10, 10]	10	71.09
PLA$_{12}$	15	[12, 12, 12, 12, 12, 12, 12]	12	74.61
PLA$_{14}$	15	[14, 14, 14, 14, 14, 14, 14]	14	77.53
PLA$_{16}$	15	[16, 16, 16, 16, 16, 16, 16]	16	82.95
GBO ($\sim$PLA$_{10}$)	15	[8, 12, 8, 10, 14, 14, 8]	10.42	76.35
GBO ($\sim$PLA$_{14}$)	15	[16, 16, 14, 16, 16, 16, 6]	14.28	82.73
Baseline	20	[8, 8, 8, 8, 8, 8, 8]	8	31.46
PLA$_{10}$	20	[10, 10, 10, 10, 10, 10, 10]	10	42.94
PLA$_{12}$	20	[12, 12, 12, 12, 12, 12, 12]	12	51.89
PLA$_{14}$	20	[14, 14, 14, 14, 14, 14, 14]	14	58.80
PLA$_{16}$	20	[16, 16, 16, 16, 16, 16, 16]	16	67.49
GBO ($\sim$PLA$_{10}$)	20	[12, 10, 8, 8, 14, 14, 6]	10.28	46.33
GBO ($\sim$PLA$_{14}$)	20	[16, 16, 16, 14, 14, 14, 12]	14.57	71.53

TABLE II
SYNERGY EFFECT WITH THE NOISE-AWARE TRAINING.

Method (Acc / avg. #pulses)	$\sigma = 10$	$\sigma = 15$	$\sigma = 20$
Baseline	83.94 / 8	62.27 / 8	31.46 / 8
NIA [7]	88.35 / 8	84.84 / 8	78.78 / 8
GBO	86.36 / 9.71	76.35 / 10.21	46.33 / 10.28
NIA [7] + GBO	88.93 / 9.71	86.45 / 10.24	81.33 / 10.28
NIA [7] + PLA	88.91 / 10	85.17 / 10	80.29 / 10

epochs. We set pulse scaling set Ω as [0.5, 0.75, 1, 1.25, 1.5, 1.75, 2] that results in a pulse length set as [4, 6, 8, 10, 12, 14, 16].

B. Experimental Results

In Table I, we present the performance of baseline, PLA, and GBO. PLA_n denotes PLA with n pulses. We set σ in noise modeling as [10, 15, 20], which can cover the wide range of noise levels in a crossbar. We obtain 90.80% accuracy without considering crossbar noise. Note that we can obtain different bit encoding solution based on trade-off parameter γ in Eq. 6. In the table, we report two GBO cases that have similar latency with PLA_{10} and PLA_{14}. From Table I, we can observe the following: (1) PLA improves the performance for all noise scenarios. This shows that crossbar noise can be mitigated by increasing the number of pulses. (2) GBO enables heterogeneous bit encoding strategies. According to the optimization results, we present the number of pulses in each layer and averaged pulse number. The results show that GBO achieves better accuracy compared to PLA with similar number of pulses.

C. Synergy with the Previous Noise-aware Training

In this experiment, we show the synergy effect between GBO and Noise-Injection Adaptation (NIA) [7]. As shown in Table II, compared to *Baseline*, using both GBO and NIA together outperform the baseline in terms of accuracy. Note that, GBO aims to only manipulate the input encoding scheme, therefore it is natural that the overall performance gain is lower than fine-tuning weight parameters which can represent sophisticated noise distribution. Combining two techniques brings further performance gain for all possible noise scenarios. The results show that GBO reduces the impact of noise intensity (*i.e.*, noise deviation), enhancing the effect of noise-aware training.

V. CONCLUSION

For the first time, we explore the impact of binary encoding techniques on crossbar noise. Interestingly, we found that the noise can be mitigated by increasing the length of bit encoding. Based on this observation, we propose Gradient-based Bit encoding Optimization (GBO) which enables heterogeneous bit encoding schemes across all layers. Through experiments, we observe that GBO effectively addresses the inherent noise problem. Moreover, combining GBO with the previous noise-aware training (*i.e.*, NIA) shows a synergy effect in terms of accuracy improvement.

ACKNOWLEDGEMENT

This work was carried out while Youngeun Kim worked as an intern at Samsung Advanced Institute of Technology, South Korea. The research was funded in part by C-BRIC, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, and the National Science Foundation (Grant#1947826).

REFERENCES

[1] Stefano Ambrogio et al. Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature*, 558(7708):60–67, 2018.
[2] Abhiroop Bhattacharjee et al. Neat: Non-linearity aware training for accurate, energy-efficient and robust implementation of neural networks on 1t-1r crossbars. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.
[3] Mahdi Nazm Bojnordi and Engin Ipek. Memristive boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 1–13. IEEE, 2016.
[4] Indranil Chakraborty et al. Geniex: A generalized approach to emulating non-ideality in memristive xbars using neural networks. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.
[5] Ping Chi et al. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. *ACM SIGARCH Computer Architecture News*, 44(3):27–39, 2016.
[6] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.
[7] Zhezhi He et al. Noise injection adaption: End-to-end reram crossbar non-ideal effect adaption for neural network mapping. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.
[8] Miao Hu et al. Dot-product engine for neuromorphic computing: Programming 1t1m crossbar to accelerate matrix-vector multiplication. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2016.
[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.
[10] Shubham Jain and Anand Raghunathan. Cxdnn: Hardware-software compensation methods for deep neural networks on resistive crossbar systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(6):1–23, 2019.
[11] Leibin Ni et al. Distributed in-memory computing on binary rram crossbar. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(3):1–18, 2017.
[12] Ali Shafiee et al. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News*, 44(3):14–26, 2016.
[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
[14] T Soliman et al. Ultra-low power flexible precision fefet based analog in-memory computing. In *2020 IEEE International Electron Devices Meeting (IEDM)*, pages 29–2. IEEE, 2020.