Received 21 August 2022; revised 11 October 2022 and 25 October 2022; accepted 1 November 2022.
Date of publication 4 November 2022; date of current version 22 December 2022.

Digital Object Identifier 10.1109/JXCDC.2022.3220032

CRUS: A Hardware-Efficient Algorithm Mitigating Highly Nonlinear Weight Update in CIM Crossbar Arrays for Artificial Neural Networks

JUNMO LEE[1], JOON HWANG[2] (Graduate Student Member, IEEE), YOUNGWOON CHO[3], MIN-KYU PARK[2] (Graduate Student Member, IEEE), WOO YOUNG CHOI[2] (Senior Member, IEEE), SANGBUM KIM[3] (Member, IEEE), and JONG-HO LEE[2,4] (Fellow, IEEE)

[1]Department of ECE, Georgia Institute of Technology, Atlanta, GA 30332 USA
[2]Department of ECE and ISRC, Seoul National University, Seoul 08826, South Korea
[3]Department of MSE and ISRC, Seoul National University, Seoul 08826, South Korea
[4]Ministry of Science and ICT, Sejong 30121, South Korea

CORRESPONDING AUTHORS: S. KIM (sangbum.kim@snu.ac.kr) and J.-H. LEE (jhl@snu.ac.kr)

This work was supported in part by the Technology Innovation Program under Grant 20009972 funded by the Ministry of Trade, Industry and Energy (MOTIE, South Korea), in part by the Brain Korea 21 Plus Project in 2022, in part by the National Research Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT under Grant NRF-2020M3F3A2A01081240 and Grant NRF-2021M3F3A2A02037889, and in part by the Samsung Electronics under Grant 0414-20210116.

This article has supplementary downloadable material available at https://doi.org/10.1109/JXCDC.2022.3220032, provided by the authors.

ABSTRACT Mitigating the nonlinear weight update of synaptic devices is one of the main challenges in designing compute-in-memory (CIM) crossbar arrays for artificial neural networks (ANNs). While various nonlinearity mitigation schemes have been proposed so far, only a few of them have dealt with high-weight update nonlinearity. This article presents a hardware-efficient on-chip weight update scheme named the conditional reverse update scheme (CRUS), which algorithmically mitigates highly nonlinear weight change in synaptic devices. For hardware efficiency, CRUS is implemented on-chip using low precision (1-bit) and infrequent circuit operations. To utilize algorithmic insights, the impact of the nonlinear weight update on training is investigated. We first introduce a metric called update noise (UN), which quantifies the deviation of the actual weight update in synaptic devices from the expected weight update calculated from the stochastic gradient descent (SGD) algorithm. Based on UN analysis, we aim to reduce AUN, the UN average over the entire training process. The key principle to reducing average UN (AUN) is to conditionally skip long-term depression (LTD) pulses during training. The trends of AUN and accuracy under various LTD skip conditions are investigated to find maximum accuracy conditions. By properly tuning LTD skip conditions, CRUS achieves >90% accuracy on the Modified National Institute of Standards and Technology (MNIST) dataset even under high-weight update nonlinearity. Furthermore, it shows better accuracy than previous nonlinearity mitigation techniques under similar hardware conditions. It also exhibits robustness to cycle-to-cycle variations (CCVs) in conductance updates. The results suggest that CRUS can be an effective solution to relieve the algorithm-hardware tradeoff in CIM crossbar array design.

INDEX TERMS Artificial neural networks (ANNs), compute-in-memory (CIM), hardware neural network, memristors, neuromorphic devices, neuromorphic systems, nonlinearity, on-chip training, synaptic device.

I. INTRODUCTION

RECENTLY, compute-in-memory (CIM) crossbar arrays having synaptic devices as weight storage elements have emerged as a promising solution to overcome the limitations of von Neumann architecture. In particular, the capability of synaptic devices to tune and store the conductance makes them well-suited for emulating biological behaviors such as long-term potentiation (LTP), long-term depression (LTD), and spike-timing-dependent plasticity [1], [2]. Such capabilities of synaptic devices can be utilized to build up a new paradigm of computation that can solve complex computational tasks with low power consumption.

Numerous types of synaptic devices that rely on various physical mechanisms have been explored in recent years [1], [2], [3]. It is well known that CIM crossbar arrays based on synaptic devices have advantages in size, speed, and power consumption compared to those in conventional digital CMOS circuits [4], [5], [6]. Despite the bright prospects of CIM crossbar arrays, several synaptic device issues must be solved for their practical implementation.

One of the main obstacles hampering the practical application of synaptic devices is nonlinearity in conductance updates. For instance, flash-type synaptic devices employing charge trapping/detrapping dynamics experience nonlinear conductance change during the program and erase operations [7]. In addition, memristor-type synaptic devices utilizing drift and diffusion mechanisms of the ions or vacancies have nonlinear gradual SET/RESET characteristics [8], [9], [10]. Due to this inherent nonlinearity in the conductance modulation process, CIM crossbar arrays suffer from a mismatch between the expected weight change and the actual weight change.

Especially, the nonlinearity issue poses a significant challenge to designing CIM crossbar arrays for on-chip artificial neural network (ANN) training and inference. While ANN can be highly resilient to random effects, excessive nonlinearity in weight updates can be harmful to training [11]. This problem has motivated many researchers to mitigate the effect of nonlinearity through various approaches. However, tuning the nonlinearity through device engineering degrades other synaptic device properties such as ON-OFF ratios and endurance cycles, affecting the performance and cost of the hardware [12], [13]. Moreover, nonlinearity mitigation using pulse schemes requires precise control of the pulse duration/amplitude considering the conductance of individual synaptic devices [14]. Unfortunately, an on-chip implementation of such pulse schemes may raise latency issues and complicate the peripheral circuit design. Therefore, nonlinearity optimization through device engineering or complicated circuit operations may not eventually lead to performance enhancement at the system level. From this point of view, incorporating algorithmic insights into nonlinearity mitigation may be an attractive option to relieve the hardware-algorithm tradeoff when designing CIM crossbar arrays for ANNs.

Accordingly, we present a hardware-efficient weight update scheme named the conditional reverse update scheme (CRUS) to train CIM crossbar arrays composed of synaptic devices with highly nonlinear LTP/LTD characteristics. In this work, we focus on CIM crossbar arrays for ANN training and inference. In contrast to the aforementioned approaches that necessitate device engineering and costly circuit operations, we suggest a hardware-friendly weight update scheme that utilizes algorithmic intuition for accuracy enhancement. For hardware efficiency, we restrict the precision of any additional read and write operations during training to 1 bit. We also introduce metrics named update noise (UN) and average UN (AUN), which quantify the deviation between the actual and ideal weight update. Important hyperparameters in CRUS are optimized by careful algorithmic analysis using UN and AUN. The proposed weight update scheme and algorithmic analysis will be useful in optimizing the system performance of CIM crossbar arrays under highly nonlinear or abrupt weight update conditions.

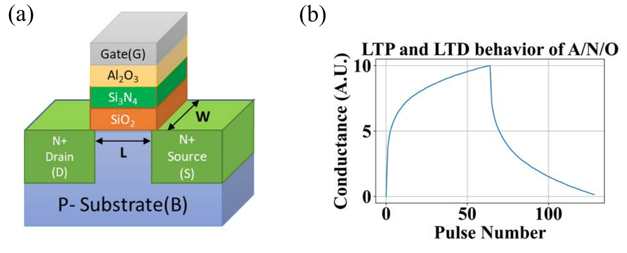

FIGURE 1. (a) Schematic 3-D view of the A/N/O device. (b) LTP and LTD behavior of the A/N/O device.

II. BIDIRECTIONAL FLASH-TYPE SYNAPTIC DEVICE

In this Section, we present a flash-type memory device that has weight update nonlinearity and abruptness level that are within the target range of this article. A schematic 3-D view of this device is illustrated in Fig. 1(a). The structure of this device is identical to that of an n-channel MOSFET with a gate insulator stack of $Al_2O_3/Si_3N_4/SiO_2$ (A/N/O). The A/N/O device is capable of bidirectional weight (conductance) change through the program (for LTD) or erase (for LTP) operations. The detailed fabrication steps, operating principles, and other applications of the A/N/O device are provided in S. 1 and [15], [16], [17].

Fig. 1(b) shows the measured LTP and LTD behaviors of the A/N/O device. The noticeable characteristic of the LTP curve is that the conductance changes are extremely abrupt in the early stage of pulse application. In this article, we will suggest that minimizing the abruptness of the conductance update may not be a necessary requirement in synaptic device engineering.

III. RELATED WORKS

Recently, various on-chip nonlinearity mitigation methods beyond device-level optimization have been proposed. However, several critical issues remain to be addressed for further development. In this section, we provide three major issues that become the core motivation of CRUS.

First, the nonlinearity level (NL) tested in previous works may be inappropriate to simulate highly nonlinear or abrupt conductance update characteristics. In this work, we use the following equations to model the conductance (G) change

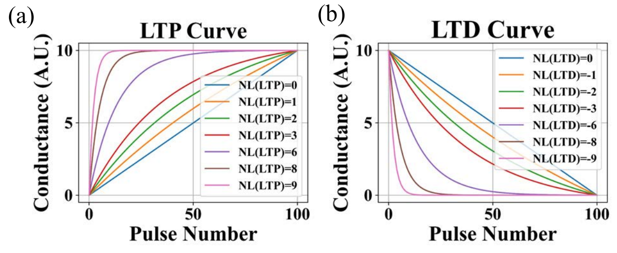

FIGURE 2. **(a) LTP and (b) LTD curves corresponding to different NL(LTP) and NL(LTD) values.**

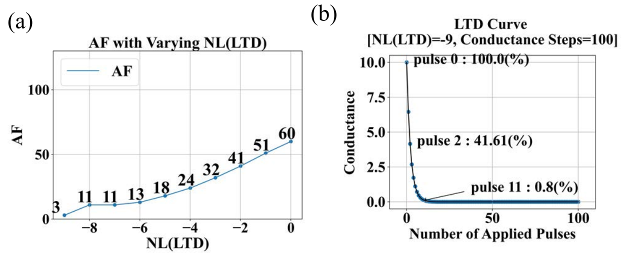

FIGURE 3. **(a) AF of LTD with varying NL(LTD). (b) Abruptness of LTD curve at NL(LTD) = −9. (b) LTD curve with large NL is extremely abrupt in the early stage of pulse application.**

during LTP and LTD process for a given NL [18]:

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

$$G_{\mathrm{LTD}}(P) = B\left(1 - e^{\frac{-(P_{\max}-P)}{A}}\right) + G_{\min} \tag{2}$$

$$B = \frac{G_{\max} - G_{\min}}{1 - e^{-\frac{P_{\max}}{A}}} \tag{3}$$

where $G_{\max}$ and $G_{\min}$ are the maximum and minimum conductance of a synaptic device extracted from experimental data, respectively. $P_{\max}$ is the maximum number of available conductance steps of a synaptic device. A is a fitting parameter dependent on NL (the definition of A is explained in S. 2), and B is a function of A as defined by (3). The NL(LTP) (NL of LTP) and NL(LTD) (NL of LTD) range from −9 to 9. The sign of NL(LTP) (NL(LTD)) determines whether LTP (LTD) curve is convex (positive sign) or concave (negative sign). Some examples of NL(LTP) and NL(LTD), and their corresponding LTP and LTD curves are illustrated in Fig. 2(a) and (b). We normalize the conductance value of synaptic devices to a range between 0 ($G_{\min}$) and 10 ($G_{\max}$) in the rest of this article.

In addition, to measure the magnitude of the abruptness of LTP or LTD characteristics, we define the abruptness factor (AF) as follows:

$$\mathrm{AF} = \begin{cases} 100\dfrac{C_{\mathrm{LTP}}}{\mathrm{P}_{\max}}, & \text{for LTP} \\ 100\dfrac{C_{\mathrm{LTD}}}{\mathrm{P}_{\max}}, & \text{for LTD} \end{cases} \tag{4}$$

where C_{LTP} represents the minimum number of pulses required to increase $G = G_{\min}$ above $G = G_{\min} + 0.6(G_{\max} - G_{\min})$, and C_{LTD} represents the minimum number of pulses required to decrease $G = G_{\max}$ below $G = G_{\max} - 0.6(G_{\max} - G_{\min})$. AF can range from 0 to 60 for the NL range of −9 to 9. Fig. 3(a) presents AF as a function of NL(LTD). Note that (1)–(3) may be unsuitable for modeling the LTP or LTD curves of some synaptic devices [19]. However, (4) can be generally applied to any synaptic device.

Nonlinearity mitigation techniques presented in [20] and [21] were validated on NL ranging from approximately −6 to 6. However, NL values within the range of −6 to 6 might be too small to capture the nonlinear or abrupt update characteristics of some synaptic devices. For example, the NL(LTD) of a reported $Pr_{0.7}Ca_{0.3}MnO_3$ (PCMO) device using the nonlinearity model defined by (2)–(3) is −6.76 [18]. The A/N/O device mentioned above has an AF of 9.375 for LTP and 17.1875 for LTD. Considering that the AF corresponding to NL(LTP) = 6 is 13, setting NL(LTP) = 6 is inappropriate to simulate abrupt LTP behavior in the A/N/O device. Referring to Fig. 3(a), the absolute value of NL has to be above 8 to simulate the abruptness of the A/N/O device.

Second, nonlinearity mitigation methods requiring frequent or accurate conductance accessing may incur high hardware costs, defeating the advantage of using CIM crossbar arrays for ANNs. Some of the successful methods yielding acceptable accuracy under high nonlinearity settings require conductance reading before each weight update event. They also necessitate a specific pulse generator design or precise conductance sensing to guarantee an acceptable level of accuracy [20], [22].

For instance, the refresh method is known as a successful algorithm effectively dealing with devices having aggressive nonlinearity such as phase change memory devices [23], [24]. By periodically sensing and rewriting the conductance values of individual devices, the saturation of conductance is prevented and thus accuracy can be increased. The key advantage of the refresh method is that if LTP is linear enough, high accuracies can be achieved regardless of the NL(LTD) value.

Despite its several benefits including high accuracy and immunity to resistance drift effects [25], the periodical sensing and rewriting process requires careful consideration of the hardware-algorithm tradeoff. It is known that analog-to-digital converter (ADC) is a major bottleneck in CIM systems in terms of area, speed, and power [26], [27], [28], [29]. Thus, using high-resolution ADC circuits for precise sensing and rewriting of individual synaptic weights may cause excessive hardware overhead. Moreover, while increasing the refresh period (number of training iterations between successive refresh processes) decreases power consumption and latency, it also decreases accuracy [11], [30]. Thus, to create practical CIM crossbar arrays, it may be more advantageous to design a weight update method that can achieve high accuracy using infrequent and inaccurate additional circuit operations.

Lastly, many studies on CIM lack quantitative analysis on the relationship between the LTP/LTD nonlinearity and accuracy [11], [22], [31]. Quantitative analysis employing proper algorithmic metrics might help elucidate the origin of accuracy degradation under nonlinear weight update conditions.

Accordingly, our work proposes: 1) CRUS, a hardware-efficient weight update rule for mitigating high nonlinearity and 2) evaluation metrics that quantify the impact of nonlinearity on CIM crossbar array performance. In addition to the standard stochastic gradient descent (SGD) algorithm, infrequent conductance reading and writing techniques requiring 1-bit precision are introduced. By optimizing CRUS using the proposed evaluation metrics, we achieve high accuracy even in high NL cases ($|\text{NL(LTD)}| \geq 8$ or $|\text{NL(LTP)}| \geq 8$ except for $|\text{NL(LTP)}| = |\text{NL(LTD)}| = 9$).

IV. UPDATE NOISE AND CONDITIONAL REVERSE UPDATE SCHEME

A. CONCEPT OF UN AND ITS METRICS

We propose metrics called UN and AUN to quantify the deviation between the actual weight update and the ideal weight update. For a single training iteration, UN is defined as follows:

$$\text{UN}\,(W_n) = \|P_n k - \left(W_{n,\text{updated}} - W_n\right)\| \text{ (L2 norm)} \tag{5}$$

where k, P_n, W_n, and $W_{n,\text{updated}}$ are the scaling factor that scales the number of applied pulses into the weight scale, a vector of the number of pulses applied to synaptic devices, a weight vector before weight update, and the weight vector after weight update at the nth training iteration, respectively. The L2 norm calculates the Euclidean distance of the vector coordinate from the origin of the vector space. Note that the term $P_n k$ represents the ideal weight update amount and the term $(W_{n,\text{updated}} - W_n)$ represents the actual weight update amount.

Software-based studies have shown that gradient noises with certain structures have regularization effects [32]. However, to the best of our knowledge, the structure and the effect of the gradient in CIM are not well understood. Moreover, for on-chip training, it is difficult to freely manipulate the gradient in the desired way. Thus, we try to self-adaptively reduce the AUN after training. We use the following equation to evaluate AUN after the end of the entire training process:

$$\text{AUN}\,(W) = \sum_{i=0}^{n} \frac{1}{m}\text{UN}\,(W_i) \tag{6}$$

where W_i, n, and m represent a weight vector at ith training iteration, the total number of training iterations of an algorithm, and the total number of training iterations except for the ones where the synaptic devices do not receive any pulses, respectively. In Section V, we show that AUN provides important insights into the effect of nonlinear weight updates on CIM crossbar training.

B. PROPOSED WEIGHT UPDATE SCHEME

We now introduce a hardware-efficient weight update method named CRUS that can effectively control the AUN of an algorithm. Specifically, we design a weight update method that: 1) reduces AUN evaluated at the end of the training by employing an infrequent and low precision circuit operation; 2) does not require any pulse shape or duration tuning; and 3) is compatible with on-chip training.

In this scheme, differential synapse pairs are used to represent each weight element. For notational simplicity, we refer to synapse pairs as synapses. Each of the synapses consists of two bidirectional synaptic devices with conductance values G_p and G_n. The weight value W that each synapse represents can be expressed as $W = (G_p - G_n/G_{\max} - G_{\min})$.

For hardware simplicity, we divide the weight update process into two phases: the normal update phase and the reverse update phase. When a specific training iteration corresponds to the normal update phase, only LTP updates are used for conductance updates. In contrast, only LTD updates are used during the reverse update phase. In addition, the reverse update phase occurs only once every *rup* (reverse update period). *rup* is a hyperparameter ranging from 2 to 10. By allowing both LTP and LTD updates, we can prevent the saturation of conductance which significantly degrades the performance of the network [11]. Note that the division of the training into two phases also allows individual tuning of the learning rate for LTP updates (α_n) and that for LTD updates (α_r). This strategy can compensate for the inherent asymmetrical update characteristics of synaptic devices [33].

The key idea to effectively reduce AUN under CRUS is to conditionally skip LTD updates (pulses) only during the reverse update phase. Specifically, if a bit ($flag_p$ or $flag_n$ in Algorithm 1) assigned to each G_p or G_n synaptic device is 1, no write pulses are applied to G_p or G_n even if the gradient descent algorithm requires updating it.

The bits assigned to each synaptic device are stored in auxiliary arrays and are updated every *refp* training iteration (*refp* refers to the reference period). $flag_p$ or $flag_n$ is updated to 1 if the conductance of the complementary synaptic device (G_n for $flag_p$, G_p for $flag_n$) is smaller than a hyperparameter G_{th}, and to 0, otherwise. This means that the LTD update is skipped if the complementary synaptic conductance was recently low enough. We will show that by adjusting G_{th}, AUN can be effectively reduced.

For better understanding, we present the pseudocode of the proposed update method and its hardware implementation scheme in Algorithm 1 and S. 3, respectively.

For convenience, we only consider the cases for $|\text{NL(LTP)}| \leq |\text{NL(LTD)}|$. $|\text{NL(LTP)}| \geq |\text{NL(LTD)}|$ cases can be handled identically by swapping the definition of the reverse update phase and the normal update phase and modifying the skip condition accordingly. We will show that the *refp* in CRUS can be larger than the refresh period (*rp*) in the refresh scheme to achieve the same level of accuracy.

V. ALGORITHMIC ANALYSIS OF CRUS

We analyze our proposed weight update rule with NeuroSim [18] using the Modified National Institute of Standards and Technology (MNIST) handwritten digit database [34].

Algorithm 1 Conditional Reverse Update Algorithm

```
$W = \frac{G_p - G_n}{G_{max} - G_{min}}$;
Let $n$ be the total number of training iterations;
Let $rup$ be the reverse update period;
Let $refp$ be the reference period;
Let $G_{th}$ be the threshold value for skipping LTD update;
Let $P_i$ be the number of pulses applied at training iteration $i$;
Let $a_n$ be the learning rate for the normal update phase;
Let $a_r$ be the learning rate for the reverse update phase;
for $i = 1$ to $n$ do
    calculate $\frac{dL}{dW}$;
    /* 1-bit information sensing & storage */
    if $i$ is 1 or an integer multiple of $refp$ then
        store(bool) $flag_p = (G_n < G_{th})$;
        store(bool) $flag_n = (G_p < G_{th})$;
    endif
    /* weight update (reverse update phase) */
    if $i$ is an integer multiple of $rup$ then
        $P_i = |a_r \frac{dL}{dW}| P_{max}$ rounded to the nearest integer;
        if $(\frac{dL}{dW} > 0)$ then
            Apply $P_i$ LTD pulses to $G_n$ if not $flag_n$;
        else
            Apply $P_i$ LTD pulses to $G_p$ if not $flag_p$;
        endif
    /* weight update (normal update phase) */
    else
        $P_i = |a_n \frac{dL}{dW}| P_{max}$ rounded to the nearest integer;
        if $(\frac{dL}{dW} > 0)$ then
            Apply $P_i$ LTP pulses to $G_p$;
        else
            Apply $P_i$ LTP pulses to $G_n$;
        endif
    endif
endfor
```

We consider a two-layer neural network (400–100–10). From this point, we define the synapses located between the input and hidden (hidden and the output) layer as S_{IH} (S_{HO}) and the corresponding weight matrix as W_{IH} (W_{HO}). The details of the network and simulation conditions are explained in S. 4.

We first consider a synaptic device with NL(LTP) $= 1$, NL(LTD) $= -9$, and $P_{max} = 100$. By setting NL(LTD) $= -9$, the AF of LTD becomes 3 and the conductance falls to 0.8% of its maximum conductance within 11 pulses, as seen from Fig. 3(a) and (b). Starting from this case, we show that our proposed method can be extended to other highly nonlinear cases. We only consider the cases where NL(LTP) and NL(LTD) have different signs.

Now we examine how CRUS effectively reduces the UN of individual synapses. The UN of a synapse representing one-element weight vector W can be expressed as UN(W). By setting a G_{th} that ensures low UN(W) of individual synapses, UN(W_{IH}) and UN(W_{HO}) can be reduced at each training iteration, and therefore AUN(W_{IH}) and AUN(W_{HO}) can be minimized. To estimate the expected amount of UN reduction for a given conductance configuration (G_p, G_n), we calculate the expected UN of an individual synapse when the reverse period is set to 2. Here, not considering the actual training situation, we roughly assume that the pulses to be applied

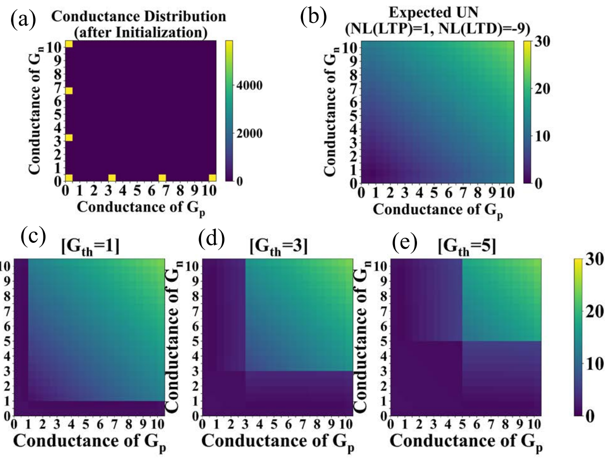

FIGURE 4. (a) Synaptic conductance distribution after initialization. (b) Expected UN without LTD pulse skip condition (G_{th} = 0). Expected UN for (c) G_{th} = 1, (d) G_{th} = 3, and (e) G_{th} = 5.

during each weight update are extracted from a uniform [−3, 3] distribution and the reverse update phase occurs with a probability of 50%. To further analyze how the expected UN varies with the conductance configuration (G_p, G_n), we also group the synapses based on their conductance configurations as follows:

(0,0) synapse: $G_p < G_{th}$, $G_n < G_{th}$.
(0,1) synapse: $G_p < G_{th}$, $G_n \geq G_{th}$.
(1,0) synapse: $G_p \geq G_{th}$, $G_n < G_{th}$.
(1,1) synapse: $G_p \geq G_{th}$, $G_n \geq G_{th}$.

Fig. 4(a)–(e) shows the initialized conductance configuration distribution and the expected UN of different synapse types under varying G_{th}. It can be seen that the expected UNs of the (0,0), (0,1), and (1,0) synapses are relatively small compared to those of the (1,1) synapses. This is because the LTD pulse skip condition applied to (0,0), (0,1), and (1,0) synapses effectively suppress the expected UN. Thus, it is implied that reducing the number of (1,1) synapses during training is important to minimize AUN.

We, therefore, discuss how to decrease the number of (1,1) synapses during training. The conversion from (0,1) or (1,0) synapses to (1,1) synapses during training should be reduced to minimize AUN. According to previous work, in a network composed of bidirectional synaptic devices, the conductance values tend to converge around the "symmetry point" [35]. This is the conductance value G_{sym} that satisfies $|G'_{LTP}(G^{-1}_{LTP}(G_{sym}))| = |G'_{LTD}(G^{-1}_{LTD}(G_{sym}))|$ [here, $G_{LTP}(P)$ and $G_{LTD}(P)$ are the functions used in (1) and (2), respectively]. Note that this is also the point to which the conductance value eventually converges after one LTP pulse and one LTD pulse are continually applied to a synaptic device in an alternating fashion. A similar phenomenon is observed under CRUS as well (S. 5). G_p (G_n) in (0,1) ((1,0)) synapses, which are capable of bidirectional conductance update, converge around the symmetry point as training proceeds (S. 6).

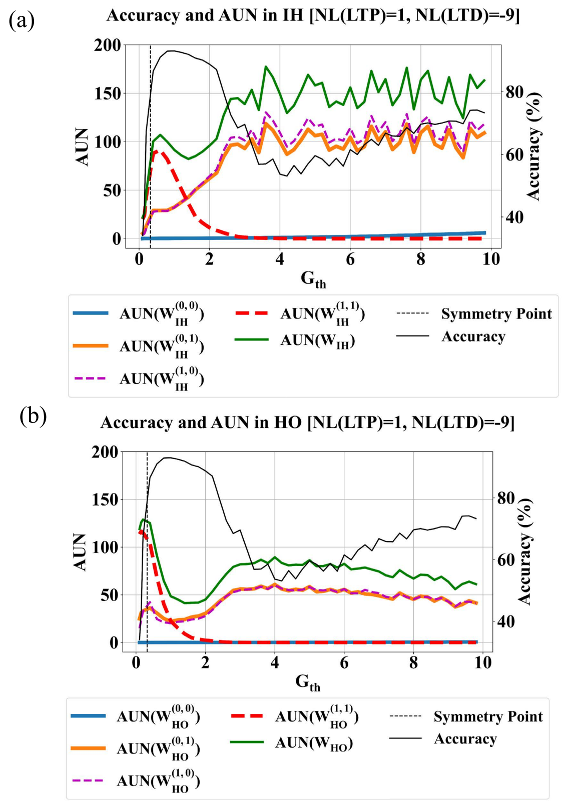

FIGURE 5. Plot of AUNs corresponding to the entire synapses and each synapse type calculated for (a) W_{IH} and (b) W_{HO}. Here, the AUN values are scaled by a factor of 10 000. Test accuracies are also shown.

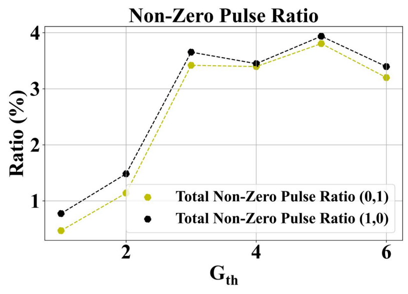

FIGURE 6. Average ratio of (0,1) and (1,0) synapses that receive non-zero pulses during the reverse update phase with respect to G_{th}.

Therefore, increasing the average distance between G_{th} and the conductance distribution of $G_p\,(G_n)$ in (0,1) ((1,0)) synapses reduces the probability of the G_ps (G_ns) crossing over G_{th} and switching the synapse type to (1,1) (S. 7). S. 5–7 provide further analyses of the aforementioned observations. Therefore, the key strategy to reduce the number of (1,1) synapses is to increase G_{th}.

Now we analyze how increasing G_{th} affects AUN values of certain layers and synapse types. To calculate the AUN of a specific synapse type, we extend the definition of AUN hereafter. AUN($W^{\text{synapse type}}$) is calculated by masking the weight and pulse elements that do not correspond to the superscripted synapse type for P_n and W_n in the argument of the summation in (6).

As seen from Fig. 5(a) and (b), increasing G_{th} does not always lead to a decrease in AUN(W_{IH}) and AUN(W_{HO}). Moreover, Fig. 5(a) shows that smaller AUN(W_{IH}) does not necessarily imply higher accuracy, suggesting minimizing AUN(W_{IH}) is not an optimal direction in accuracy optimization. In the rest of this section, we examine several important factors that determine the behaviors of accuracy, AUN(W_{IH}), and AUN(W_{HO}) under varying G_{th}. Based on this analysis, we finally show that AUN(W_{HO}) should be crucially considered for G_{th} optimization.

First, the UN reduction effect of CRUS decreases as G_{th} increases. We attribute this trend to the increase in the average ratio of (0,1)/(1,0) synaptic devices that receive non-zero pulses (during the reverse update phase) with a G_{th} increase. Pulses not being applied to (0,1)/(1,0) synapses during the reverse update phase implies two cases: 1) LTD pulses are applied to $G_n\,(G_p)$ in (0,1) ((1,0)) synapses, so they are skipped according to CRUS and 2) original weight update amount is calculated to be zero. In 1), UN is suppressed by the skip update effect as discussed earlier, and in 2), UN is zero. Therefore, if the input pulse pattern changes with G_{th}, AUN can be affected. Fig. 6 shows a $\sim$4$\times$ increase in the average ratio of (0,1)/(1,0) synapses that receive nonzero pulses (during the reverse update phase) with a G_{th} increase. This is correlated with the increase in AUN(W_{HO}) and AUN(W_{IH}) from $G_{th} \sim 1$ as seen in Fig. 5(a) and (b). Thus, G_{th} should not be increased above the point where the increase in AUN($W_{IH}^{(0,1)}$), AUN($W_{IH}^{(1,0)}$), AUN($W_{HO}^{(0,1)}$), and AUN($W_{HO}^{(1,0)}$) starts to offset the decrease in AUN($W_{IH}^{(1,1)}$), and AUN($W_{HO}^{(1,1)}$).

Second, the weight distribution of W_{HO} can significantly affect the behavior of AUN(W_{IH}) and accuracy at low G_{th}. Fig. 5(a) shows that when G_{th} is lower than the symmetry point ($\sim$0.32), both AUN(W_{IH}) and accuracy sharply decrease with decreasing G_{th}. This differs from the case of AUN(W_{HO}) [see Fig. 5(b)], which generally has a contrary relationship with accuracy with respect to G_{th} throughout the entire G_{th} range. If the symmetry point increases above G_{th}, conversion from (1,1) to (0,1) or (1,0) rarely happens. In addition, both G_p and G_n keep converging around the symmetry point until the end of training. Thus, not only an excessive UN is caused due to a large number of (1,1) synapses, but also the G_ps and G_ns of (1,1) synapses in S_{HO} become densely populated around the symmetry point as training proceeds [see Fig. 7(a)]. Consequently, the number of low magnitude weights in W_{HO} substantially increases at low G_{th}($G_{th} < 0.32$).

Similar to a reported observation that excessive sparsity of weights leads to a vanishing gradient problem [36], a large increase in the number of low-magnitude weights in W_{HO} limits gradient flow to W_{IH} when G_{th} is smaller than the symmetry point. The vanishing gradient problem retards

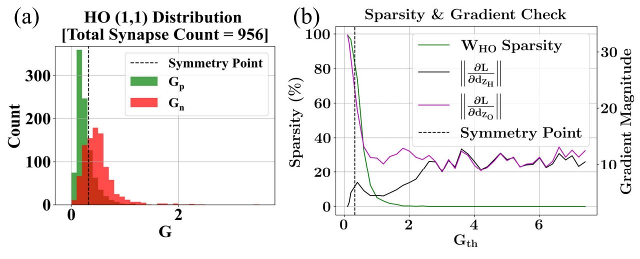

FIGURE 7. (a) G_p and G_n distribution of (1,1) synapses located between the hidden and the output layer at low G_{th}(G_{th} < 0.32). (b) Derivative of loss function (L) with respect to the hidden layer pre-activations (z_H) and the output layer pre-activations (z_O) averaged over the whole training process and sparsity of W_{HO}. Here, sparsity is defined by the ratio of weights of magnitude <0.2.

training in early layers and significantly drops the accuracy of a network [37]. Fig. 7(b) shows that the magnitude of the gradient passed to W_{IH} significantly decreases as G_{th} decreases. Referring to Fig. 4(a), this means that the synapses corresponding to W_{IH} will mostly stay at their initialized states, where the expected UN is relatively low, throughout training. As a result, AUN(W_{IH}) and accuracy both drop sharply when G_{th} decreases below the symmetry point.

From the above analysis, it can be concluded that AUN(W_{HO}) is the most influential factor that determines accuracy. When G_{th} is smaller than the symmetry point, AUN(W_{HO}) reflects the number of (1,1) synapses, which is inversely related to the magnitude of the gradient W_{IH} receives. When G_{th} is larger than the symmetry point, the average effect of weight update deviation throughout training is a determining factor in accuracy. Thus, AUN(W_{HO}) generally has a contrary relationship with accuracy with respect to G_{th} [see Fig. 5(b)]. Using these observations, we show that AUN(W_{HO}) can be used for accuracy optimization in CRUS. Fig. 8 shows the correlation between the G_{th} that minimizes AUN(W_{HO}) and the optimized G_{th} in terms of accuracy under different NL settings. The result implies that accuracy can easily be optimized by fine-tuning G_{th} around the value that yields minimal AUN(W_{HO}).

VI. OPTIMIZATION METHODOLOGY AND SYSTEM PERFORMANCE ANALYSIS FOR GENERAL HIGH NL CONDITIONS

A. RESULTS FOR VARIOUS DEVICE NLs AND HYPERPARAMETER CONDITIONS

In this section, we show that the G_{th} optimization methodology is applicable to other high NL and hyperparameter conditions. We examine several factors that increase/decrease the symmetry point or the location of the conductance distribution of G_p (G_n) in (0,1) ((1,0)) synapses. Following the analysis in Section IV, G_{th} needs to be optimized accordingly in such cases. If the symmetry point or the position of the conductance distribution increases, G_{th} should be set higher to suppress G_{th} crossover events. In the opposite case,

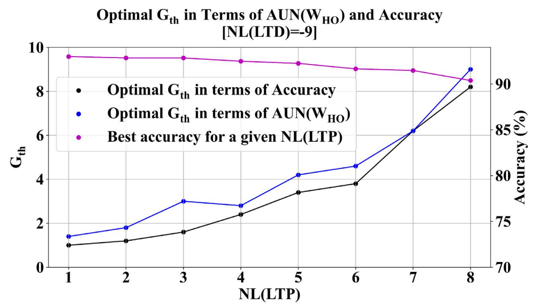

FIGURE 8. Optimal G_{th} in terms of AUN(W_{HO}), that in terms of accuracy, and the best accuracy with varying NL(LTP) where NL(LTD) is fixed to −9.

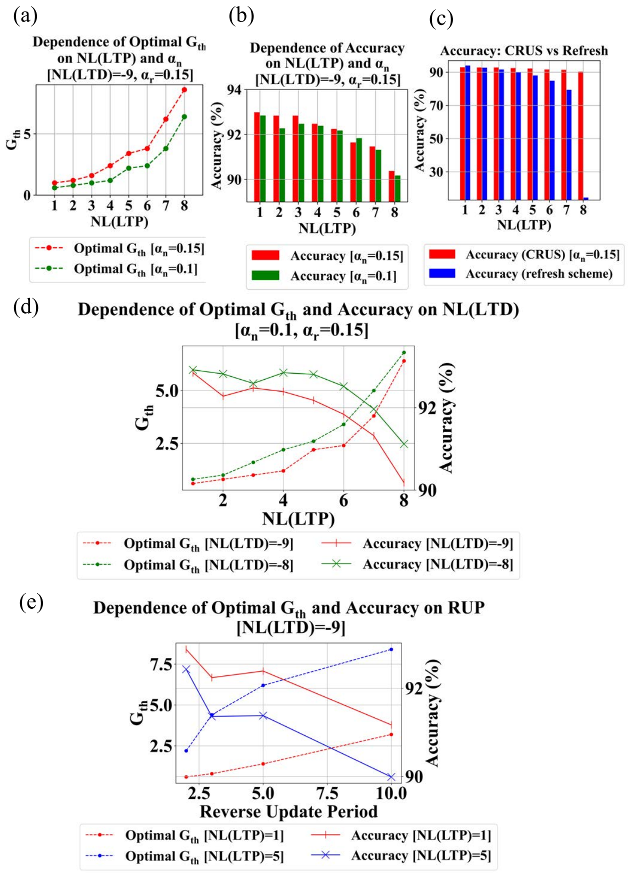

FIGURE 9. NL(LTP) and α_n dependence of (a) optimized G_{th} and (b) accuracy in CRUS. (c) Accuracy comparison between the refresh scheme and CRUS. Here, NL(LTD), α_n, and α_r are fixed to −9, 0.15, and 0.15, respectively. (d) NL(LTD) dependence of accuracy and optimized G_{th}. (e) *rup* dependence of accuracy and optimized G_{th} in CRUS.

G_{th} should be set lower to reduce the effect of increasing AUN($W_{HO}^{(0,1)}$) and AUN($W_{HO}^{(1,0)}$). For the simulation, *rp*, *refp*, and P_{max} are set to 2, 2, and 100, respectively. *rup* is set to 2 for Fig. 9(a)–(d). α_r is fixed to 0.15. The implementation details of the refresh method are explained in S. 8.

Increasing NL(LTP) is expected to increase the optimal G_{th} since the symmetry point is shifted upward

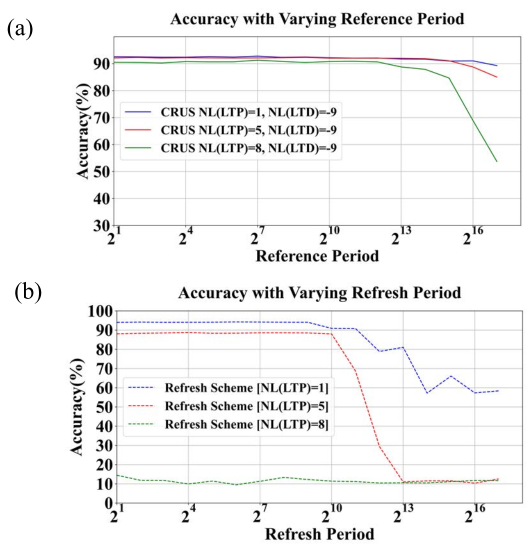

FIGURE 10. **Accuracy with varying (a) *refp* and (b) *rp*.**

[see Fig. 9(a)]. Decreasing NL(LTD) is expected to have the opposite effect. Therefore, the optimal G_{th} decreases with decreasing NL(LTD) [see Fig. 9(d)].

Similarly, the learning rate corresponding to the normal update phase (α_n) and that corresponding to the reverse update phase (α_r) also affect the choice of G_{th}. When a larger α_n is chosen while α_r is fixed, the conductance distribution of G_p (G_n) in (0,1) ((1,0)) synapses shifts upward. The reason is that the LTP pulses are applied more frequently than when $\alpha_n = \alpha_r$. Thus, as α_n increases, G_{th} needs to be increased to enhance accuracy, as shown in Fig. 9(a) and (b).

Fig. 9(b) and (d) clearly shows that CRUS yields >90% accuracy for high NL cases ($|\text{NL(LTD)}| \geq 8$ or $|\text{NL(LTP)}| \geq 8$ except for $|\text{NL(LTP)}| = |\text{NL(LTD)}| = 9$) by setting an optimal G_{th} value. In addition, CRUS yields better accuracy than the refresh method if NL(LTP) exceeds 1.

Setting a high *rup* provides a similar effect as setting a large α_n since LTP updates happen more frequently with higher *rup*. Increasing *rup* may offer latency benefits, as the time required to sense the stored bits during the reverse update phase can be reduced. Fig. 9(e) shows that the optimal G_{th} increases with increasing *rup*. However, the optimized accuracy tends to decrease when *rup* increases. Therefore, the tradeoff between training speed and accuracy should be considered when adjusting *rup*.

B. EFFECT OF THE REFERENCE PERIOD ON THE TRAINING PERFORMANCE AND COMPARISON WITH THE REFRESH METHOD

The key advantage of CRUS is that *refp* can be larger than *rp*, implying a significant reduction in hardware cost. Fig. 10(a) and (b) shows that CRUS could maintain the accuracy level at *refp* = 2 (>90%) up to a larger additional circuit operation period, even for the worst case [NL(LTP) = 8, NL(LTD) = −9] we are considering (>90% for $refp \leq 2^{12}$). As explained in Section III, the frequent refresh process may incur excess power, area, and latency overhead due to ADCs. The capability of achieving high accuracy under infrequent and inaccurate additional circuit operations implies a relaxation in the ADC resolution and a reduction of costly memory/ADC circuit operations. This suggests that CRUS has the potential to provide excellent hardware efficiency in CIM crossbar array operation.

TABLE 1. **Comparison of accuracies and device/hardware requirements among different nonlinearity mitigation techniques.**

Weight Update Method	Weight Precision (Levels)	Input Image Pixel Precision (Levels)	Neuron Precision (Levels)	Nonlinearity Condition	Accuracy (%)
Our implementation of [20] (4 segment model & middle strategy for LTP & LTD)	100	2	256	NL(LTP)=8 NL(LTD)=-8	70.92
				NL(LTP)=1 NL(LTD)=-9	76.72
Our method [Reference period=4096]	100	2	256	NL(LTP)=8 NL(LTD)=-8	91.14
				NL(LTP)=1 NL(LTD)=-9	92.41
[31]	64	256	256	ANL=0.8	87.8
Our Method [Reference Period=4096]	64	4	256	NL(LTP)=8 NL(LTD)=-8 [ANL=0.99]	89.66
Our method [Reference period=4096]	64	4	256	NL(LTP)=4 NL(LTD)=-9 [ANL=0.86]	92.89

VII. SYSTEM PERFORMANCE COMPARISON WITH OTHER NONLINEARITY MITIGATION SCHEMES

Table 1 compares the accuracies achieved by CRUS against those from previous nonlinearity mitigation methods. We use the MNIST dataset and set equal or poorer hardware/device conditions compared to those of previous works for accuracy comparison. For reasonable comparison, we do not consider works that do not specify all hardware/device conditions listed in Table 1. For [20], we perform additional simulations to test its effectiveness under severe NL cases. The asymmetric nonlinearity (ANL) factor denotes a metric proposed to evaluate the asymmetry between LTP and LTD curves [31]. 0 is the case of ideal symmetry, and the asymmetry increases as the ANL value increases. Note that the accuracies of [20] and [31] in Table 2 are not the highest reported values in each reference, which were achieved under lower NLs or higher weight precision.

As mentioned in Section III, Fu et al. [20] were successful in dealing with NL values of −6 to 6. The results of [20] in Table 1 imply that CRUS can be an attractive algorithm/hardware solution targeting devices in the high nonlinearity regime. Moreover, compared with [31], CRUS can achieve high accuracy under worse LTP/LTD asymmetry and more limited hardware conditions (lower input pixel precision). Interestingly, CRUS still achieves >89% accuracy under a lower number of conductance states (=64) when the input pixel precision is relaxed.

TABLE 2. Summary of hardware features among state-of-the-art nonlinearity mitigation techniques.

Hardware Features	[20]	Refresh	[21]	[31]	CRUS
Do not read before every weight update	X	O	O	O	O
Do not generate multiple pulses for each weight update	X	O	O	O	O
Additional conductance accessing during weight updates do not exist or can be performed infrequently and inaccurately	X	X	O	O	O
Do not require auxiliary arrays or memory circuits to store previous weight information	X	X	O	O	X
High accuracy under severe nonlinearity (NL(LTP)≥8 and NL(LTD)≥8)	X	X	X	X	O

* "O" means the statement is true, and "X" means it is false

Table 2 presents an overall summary of the hardware features of the state-of-the-art nonlinearity mitigation techniques. One may argue that introducing an auxiliary array and reading/writing circuitries for stored bits can offset the advantage of on-chip CIM crossbar array training. However, since the precision required for reading/writing stored bits is only 1-bit, the area/speed/power costs due to ADCs and memory overhead are expected to be small [26], [27], [28], [29]. Furthermore, the increased area overhead due to the auxiliary array can be alleviated by stacking the synaptic arrays vertically using 3-D integration schemes [38], [39], [40]. As the operations performed in the auxiliary array are identical to those in the main array, the peripheral circuit for both arrays can also be shared [38]. Given its capability to achieve high accuracy under aggressive nonlinearity conditions with manageable hardware costs, CRUS is expected to bring considerable system-level benefits.

VIII. ROBUSTNESS TO CYCLE-TO-CYCLE VARIATIONS IN CONDUCTANCE UPDATES

Cycle-to-cycle variation (CCV) is another crucial factor to be considered in designing algorithms for CIM crossbar arrays [41], [42]. Therefore, we analyze the performance of CRUS under CCV. The equations used to model the variation at every conductance update event are as follows [18]:

$$G_{i,\text{updated}} = G_i + \Delta G_i + Z\sqrt{P}_i,\ Z\sim \mathrm{N}\left(0,\sigma^2\right) \qquad (7)$$

$$\sigma = \beta\left(G_{\max} - G_{\min}\right) \qquad (8)$$

where G_i, ΔG_i, $G_{i,\text{updated}}$, $\mathrm{N}(0,\sigma^2)$, P_i, and β are the conductance value at ith training iteration before the weight update, the amount of conductance update at ith training iteration assuming no variation, the conductance value at ith training iteration after weight update, the normal distribution with mean 0 and standard deviation σ, the number of applied pulses at ith training iteration, and the variation coefficient respectively. Fig. 11 shows that CRUS is highly resilient to CCV compared to the refresh method. Moreover, for high β,

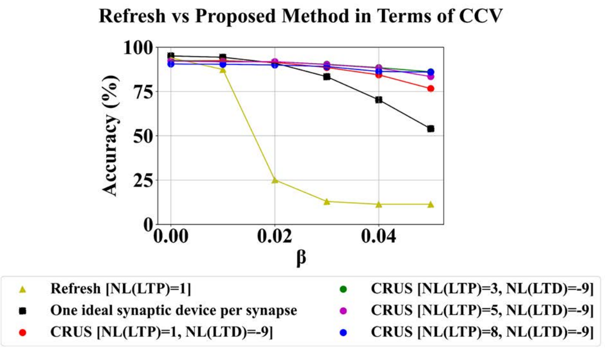

FIGURE 11. Comparison of robustness to CCV between different methods. Here, NL(LTD) = −9, and *refp* and *rup* are set to 4096 and 2, respectively. *rp* for the refresh scheme is set to 2048.

CRUS with nonlinear synapses outperforms the "one ideal synaptic device per synapse" method (an ideal CIM crossbar architecture where a device with perfectly linear LTP and LTD is used to represent each weight). This could be due to: 1) frequent LTD update skips and 2) robustness to weight update perturbation provided by the nonlinear weight update [43]. More detailed discussions on 1) and 2) are provided in S. 9.

IX. CONCLUSION

In this article, CRUS has been proposed to mitigate highly nonlinear and the abrupt weight update of synaptic devices in CIM crossbar arrays. The key advantage of CRUS is its capability to achieve high accuracy under highly nonlinear LTP and LTD characteristics utilizing infrequent and 1-bit precision circuit operations. In particular, we have demonstrated using NeuroSim that CRUS achieves >90% accuracy on the MNIST classification task under $P_{\max} = 100$, $|\mathrm{NL(LTP)}| \geq 8$ or $|\mathrm{NL(LTD)}| \geq 8$ cases (except for the $|\mathrm{NL(LTP)}| = |\mathrm{NL(LTD)}| = 9$ case). Based on the AUN and the symmetry point analysis under various situations, we have introduced methodologies to optimize G_{th} depending on NL(LTP), NL(LTD), learning rate, and *rup*. System performance comparisons with different nonlinearity mitigation techniques have implied that CRUS exhibits a superior hardware-accuracy tradeoff. Finally, we have demonstrated that CRUS exhibits robustness to CCV in conductance updates.

CRUS and the AUN analysis would be more appealing if their validity could be tested on different datasets and other complex tasks. Especially, the generalization of AUN analysis to more complicated networks could provide a better understanding of nonlinear weight updates. Moreover, devising a synaptic architecture where the synaptic device and stored bit coexist within a single cell could be a practical solution to further lessen hardware requirements. We believe the proposed weight update scheme and the AUN analysis can be attractive options when designing CIM crossbar arrays consisting of devices with high nonlinearity.

REFERENCES

[1] G. W. Burr et al., "Neuromorphic computing using non-volatile memory," *Adv. Phys., X*, vol. 2, no. 1, pp. 89–124, 2017.

[2] S. Yu, "Neuro-inspired computing with emerging nonvolatile memorys," *Proc. IEEE*, vol. 106, no. 2, pp. 260–285, Feb. 2018.

[3] Q. Wan, M. T. Sharbati, J. R. Erickson, Y. Du, and F. Xiong, "Emerging artificial synaptic devices for neuromorphic computing," *Adv. Mater. Technol.*, vol. 4, no. 4, Apr. 2019, Art. no. 1900037.

[4] S. B. Eryilmaz, S. Joshi, E. Neftci, W. Wan, G. Cauwenberghs, and H.-S.-P. Wong, "Neuromorphic architectures with electronic synapses," in *Proc. 17th Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2016, pp. 118–123.

[5] C. Li et al., "Efficient and self-adaptive in-situ learning in multilayer memristor neural networks," *Nature Commun.*, vol. 9, no. 1, pp. 1–8, 2018.

[6] T. Gokmen and Y. Vlasov, "Acceleration of deep neural network training with resistive cross-point devices: Design considerations," *Frontiers Neurosci.*, vol. 10, p. 333, Jul. 2016.

[7] J.-M. Choi, E.-J. Park, J.-J. Woo, and K.-W. Kwon, "A highly linear neuromorphic synaptic device based on regulated charge trap/detrap," *IEEE Electron Device Lett.*, vol. 40, no. 11, pp. 1848–1851, Nov. 2019.

[8] S. Park et al., "Neuromorphic speech systems using advanced ReRAM-based synapse," in *IEDM Tech. Dig.*, pp. 25.6.1–25.6.4.

[9] H. Wu et al., "Device and circuit optimization of RRAM for neuromorphic computing," in *IEDM Tech. Dig.*, 2017, pp. 11.5.1–11.5. 4.

[10] H. Liu, M. Wei, and Y. Chen, "Optimization of non-linear conductance modulation based on metal oxide memristors," *Nanotechnol. Rev.*, vol. 7, no. 5, pp. 443–468, Oct. 2018.

[11] G. W. Burr et al., "Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element," *IEEE Trans. Electron Devices*, vol. 62, no. 11, pp. 3498–3507, Nov. 2015.

[12] J. Woo and S. Yu, "Impact of selector devices in analog RRAM-based crossbar arrays for inference and training of neuromorphic system," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 9, pp. 2205–2212, Sep. 2019.

[13] S. A. Khan and S. Kim, "Comparison of diverse resistive switching characteristics and demonstration of transitions among them in al-incorporated HfO^2-based resistive switching memory for neuromorphic applications," *RSC Adv.*, vol. 10, no. 52, pp. 31342–31347, 2020.

[14] P.-Y. Chen et al., "Mitigating effects of non-ideal synaptic device characteristics for on-chip learning," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2015, pp. 194–199.

[15] M.-K. Park et al., "Field effect transistor-type devices using high-K gate insulator stacks for neuromorphic applications," *ACS Appl. Electron. Mater.*, vol. 2, no. 2, pp. 323–328, 2019.

[16] Y.-T. Seo, M.-K. Park, J.-H. Bae, B.-G. Park, and J.-H. Lee, "Implementation of synaptic device using various high-K gate dielectric stacks," *J. Nanosci. Nanotechnol.*, vol. 20, no. 7, pp. 4292–4297, Jul. 2020.

[17] M.-K. Park et al., "CMOS-compatible low-power gated diode synaptic device for hardware-based neural network," *IEEE Trans. Electron Devices*, vol. 69, no. 2, pp. 832–837, Feb. 2022.

[18] P.-Y. Chen, X. Peng, and S. Yu, "NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 37, no. 12, pp. 3067–3080, Dec. 2018.

[19] D. Querlioz, P. Dollfus, O. Bichler, and C. Gamrat, "Learning with memristive devices: How should we model their behavior?" in *Proc. IEEE/ACM Int. Symp. Nanosc. Architectures*, Jun. 2011, pp. 150–156.

[20] J. Fu, Z. Liao, N. Gong, and J. Wang, "Mitigating nonlinear effect of memristive synaptic device for neuromorphic computing," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 377–387, Jun. 2019.

[21] W. Zhang et al., "Hardware-friendly stochastic and adaptive learning in memristor convolutional neural networks," *Adv. Intell. Syst.*, vol. 3, no. 9, 2021, Art. no. 2100041.

[22] W.-Q. Pan et al., "Strategies to improve the accuracy of memristor-based convolutional neural networks," *IEEE Trans. Electron Devices*, vol. 67, no. 3, pp. 895–901, Mar. 2020.

[23] M. Ito et al., "Lightweight refresh method for PCM-based neuromorphic circuits," in *Proc. IEEE 18th Int. Conf. Nanotechnol.*, Jul. 2018, pp. 1–4.

[24] O. Bichler, M. Suri, D. Querlioz, D. Vuillaume, B. DeSalvo, and C. Gamrat, "Visual pattern extraction using energy-efficient '2-PCM synapse' neuromorphic architecture," *IEEE Trans. Electron Devices*, vol. 59, no. 8, pp. 2206–2214, Aug. 2012.

[25] M. Suri et al., "Interface engineering of PCM for improved synaptic performance in neuromorphic systems," in *Proc. 4th IEEE Int. Memory Workshop*, May 2012, pp. 1–4.

[26] S. Yu, X. Sun, X. Peng, and S. Huang, "Compute-in-memory with emerging nonvolatile-memories: Challenges and prospects," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Mar. 2020, pp. 1–4.

[27] A. Singh, M. A. Lebdeh, A. Gebregiorgis, R. Bishnoi, R. V. Joshi, and S. Hamdioui, "SRIF: Scalable and reliable integrate and fire circuit ADC for memristor-based CIM architectures," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 5, pp. 1917–1930, May 2021.

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

[29] M. Hu et al., "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," in *Proc. 53rd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2016, pp. 1–6.

[30] G. Burr et al., "Large-scale neural networks implemented with non-volatile memory as the synaptic weight element: Comparative performance analysis (accuracy, speed, and power)," in *IEDM Tech. Dig.*, 2015, pp. 4.4.1–4.4.4.

[31] C.-C. Chang et al., "Mitigating asymmetric nonlinear weight update effects in hardware neural network based on analog resistive synapse," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 1, pp. 116–124, Mar. 2018.

[32] J. Wu, W. Hu, H. Xiong, J. Huan, V. Braverman, and Z. Zhu, "On the noisy gradient descent that generalizes as sgd," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 10367–10376.

[33] A. Fumarola et al., "Accelerating machine learning with non-volatile memory: Exploring device and circuit tradeoffs," in *Proc. IEEE Int. Conf. Rebooting Comput. (ICRC)*, Oct. 2016, pp. 1–8.

[34] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.

[35] H. Kim et al., "Zero-shifting technique for deep neural network training on resistive cross-point arrays," 2019, *arXiv:1907.10228*.

[36] P. Wimmer, J. Mehnert, and A. Condurache, "FreezeNet: Full performance by reduced storage costs," in *Proc. Asian Conf. Comput. Vis.*, 2020, pp. 1–17.

[37] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1310–1318.

[38] T. Gokmen and W. Haensch, "Algorithm for training neural networks on resistive device arrays," *Frontiers Neurosci.*, vol. 14, p. 103, Feb. 2020.

[39] S. Hyun Jo, T. Kumar, S. Narayanan, W. D. Lu, and H. Nazarian, "3D-stackable crossbar resistive memory based on field assisted superlinear threshold (FAST) selector," in *IEDM Tech. Dig.*, Dec. 2014, pp. 6.7.1–6.7.4.

[40] C.-H. Wang et al., "3D monolithic stacked 1T1R cells using monolayer MoS_2 FET and hBN RRAM fabricated at low (150°C) temperature," in *IEDM Tech. Dig.*, 2018, pp. 22.5.1–22.5.4.

[41] J.-H. Lee, D.-H. Lim, H. Jeong, H. Ma, and L. Shi, "Exploring cycle-to-cycle and device-to-device variation tolerance in MLC storage-based neural network training," *IEEE Trans. Electron Devices*, vol. 66, no. 5, pp. 2172–2178, May 2019.

[42] X. Sun and S. Yu, "Impact of non-ideal characteristics of resistive synaptic devices on implementing convolutional neural networks," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 3, pp. 570–579, Sep. 2019.

[43] S. Dong et al., "A backpropagation with gradient accumulation algorithm capable of tolerating memristor non-idealities for training memristive neural networks," *Neurocomputing*, vol. 494, pp. 89–103, Jul. 2022.