ReNEW: Enhancing Lifetime for ReRAM Crossbar based Neural Network Accelerators

Wen Wen[†], Youtao Zhang[§], Jun Yang[†]
[†]Department of Electrical and Computer Engineering, [§]Department of Computer Science
University of Pittsburgh
wew55@pitt.edu, zhangyt@cs.pitt.edu, juy9@pitt.edu

***Abstract*—With analog current accumulation feature, resistive memory (ReRAM) crossbars are widely studied to accelerate neural network applications. The ReRAM crossbar based accelerators have many advantages over conventional CMOS-based accelerators, such as high performance and energy efficiency. However, due to the limited cell endurance, these accelerators suffer from short programming cycles when weights that stored in ReRAM cells are frequently updated during the neural network training phase.**

In this paper, by exploiting the wearing out mechanism of ReRAM cell, we propose a novel comprehensive framework, ReNEW, to enhance the lifetime of the ReRAM crossbar based accelerators, particularly for neural network training. Evaluation results show that, our proposed schemes reduce the total effective writes to ReRAM crossbar based accelerators by up to $500.3\times$, $50.0\times$, $2.83\times$ and $1.60\times$ over two MLC ReRAM crossbar baselines, one SLC ReRAM crossbar baseline and an SLC ReRAM crossbar design with optimal timing, respectively.

I. Introduction

In recent years, the neural networks have gained increasing attentions and been successfully applied to a wide range of applications [1]–[4]. The increasingly growth in the size of datasets and the number of layers in neural networks help to achieve a better prediction accuracy, but also result in dramatically increased computations and expensive data movement from off-chip memory. Conventional CMOS-based general purpose processor such as multi-core CPU [5] and GPGPU [1], or specialized hardware accelerators, such as FPGA [6] and ASIC designs [7], [8], are intensively studied and proposed with software and hardware optimizations for neural network applications, however, they still suffer from the large energy consumption and limited memory bandwidth [9].

To address these issues, resistive memory (ReRAM), with adopting crossbar array structure, is proposed to implement dot-product calculations by leveraging its analog current accumulation feature [9]–[14]. ReRAM crossbars are able to accelerate neural networks computation with low energy consumption and minimized data movement [11], since they have almost zero leakage power and intrinsically support the processing-in-memory (PIM) computation paradigm.

Though ReRAM crossbar based neural network accelerators own these advantages over conventional CMOS-based accelerators, due to the limited cell endurance [13]–[16], they suffer from short programming cycles as weight data stored in ReRAM cells are frequently updated during the neural network training. The write endurance of an ReRAM chips can range from 10^6 to 10^{12} [13], [17], [18] with adopting various resistive materials and different programming schemes. On the other hand, training the state-of-the-art deep neural networks usually demands at least 5 orders of magnitudes of weight updates, which essentially leads to frequent ReRAM cell programming. Therefore, enhancing the lifetime of ReRAM crossbars is the key to facilitate its widespread adoption as hardware accelerators for neural network training.

Conventional wear-leveling techniques for NVM (non-volatile memory) based main memory have been well-studied, mostly with a focus on evenly distributing write requests across pages [19]–[22]. With distinct programming patterns, ReRAM crossbar based neural network accelerators may potentially demand for an innovative approach. Prior efforts on extending ReRAM crossbar based neural network accelerators either manage to squeeze the endurance of the degraded MLC ReRAM cells [14] or exploit the gradient sparsification and regularly perform row-swapping [14]. However, in order to further improve the endurance of ReRAM crossbars for neural network training, it is necessary to investigate optimal programming strategies by exploiting the mechanism of endurance degradation in ReRAM crossbars, while taking characteristics of the target application, i.e., neural network training, as well as crossbar array features into account.

Our goal in this paper is to enhance lifetime for ReRAM crossbar based neural network accelerators. To achieve this, we propose a comprehensive framework, ReREW, which consists of techniques that can effectively prolong ReRAM crossbar lifetime during neural network training. A summary of our main contributions is listed as follows.

- Unlike many of prior studies, we propose to program ReRAM cells in crossbars in SLC (Single Level Cell) mode for neural network training and in MLC (Multi-Level Cell) mode during the inference, in order to fully take the advantage of longer endurance of SLC ReRAM cells during the training and larger capacity of MLC ReRAM cells for the inference.
- Prior studies show that different in-memory data patterns lead to discrepancies in programming latency and voltage stress, which further causes the disparity of actual wearing out degrees of ReRAM cells. Based on this observation,

This work was supported in part by the U.S. National Science Foundation under Grant CCF-1617071, Grant CCF-1718080, Grant CCF-1725657 and Grant CCF-1910413.

978-1-5386-6648-7/19/$31.00 ©2019 IEEE
DOI 10.1109/ICCD46524.2019.00074

we adopt the optimal programming latency and propose to update weights in an optimized order that can maximize the lifetime of ReRAM crossbars.

- We analyze the trade-off between endurance and programming conditions, and then present an endurance analytical model for ReRAM cell in SLC mode with different programming strengths. In addition, an analytical study of the trade-off between programming latency and switching probability is presented. Based on these analyses along with the intrinsic error-tolerance of neural network training, we propose to intentionally shorten the programming time to enhance lifetime of ReRAM crossbars at a cost of possibly unsuccessful ReRAM cell switching.
- Inspired by a conventional wear-leveling technique for NVM based main memory, we also propose to shift and update a group of columns between training iterations, which can effectively spread out writes across the whole crossbar.
- Our experiment evaluations prove that our proposed techniques reduce the total effective writes to ReRAM crossbar based accelerators by up to $500.3\times$, $50.0\times$, $2.83\times$ and $1.60\times$ over two MLC baselines, SLC baseline and SLC design with optimal timing respectively.

In the rest of this paper, we first introduce the basics of ReRAM cell basics, ReRAM crossbar based accelerators and neural network training in Section II. Then we present the motivation of proposed schemes in Section III. In Section IV, we elaborate design details for enhancing lifetime of ReRAM crossbar based accelerators. Experimental results are presented in Section V. We briefly discuss related work in Section VI and conclude the whole paper in Section VII.

II. Background

In this section, we discuss about the fundamentals of ReRAM and its applications as neural network accelerators, and also briefly introduce the neural network training.

A. ReRAM Fundamentals

ReRAM is a passive resistive based non-volatile memory technology, which uses different resistance states to represent data values. Fig. 1 illustrates the structure of an ReRAM cell, which consists of a metal oxide layer sandwiched by two metal electrodes on the top and bottom. According to a prior study [18], different classes of ReRAM with various metal oxide and electrode materials, such as $CuTe_x/HfO_2$ and $CuTe_x/Al_2O_3$, exhibit diverse characteristics such as endurance, retention and scalability.

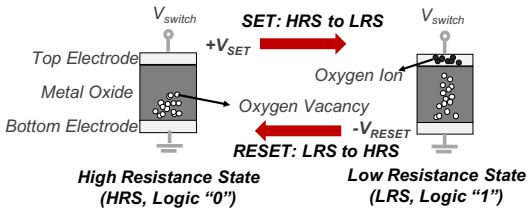


Fig. 1: ReRAM cell structure and basic (SET/RESET) programming operations.

Fig. 1 also depicts two basic programming procedures for ReRAM — RESET and SET, which are reversible switching operations and used to store data in an ReRAM cell. For an SLC ReRAM cell, with a positive voltage larger than a certain threshold applied to the top electrode, the current flowing through cell enables a formation of the conductive filaments (CF) in the metal oxide layer, switching the ReRAM cell to low resistance state (LRS). On the contrary, during the RESET process, which is initialized with a negative voltage on the top electrode, the CFs are ruptured and consequently the cell is switched to high resistance state (HRS). To program an MLC ReRAM cell is much more complicated with consuming significantly more power and time [17], [23] and thereby wears out cells much faster, since an iterative programming, i.e., Program & Verify (P&V), is used to accurately achieve the intermediate resistance levels.

B. ReRAM crossbar and Its Application for Neural Network Computing

Fig. 2 illustrates an ReRAM crossbar architecture, in which each ReRAM cell is connected to a worldline and bitline at their crosspoint. With a voltage stress, ReRAM cell behaves as resistive devices obeying Ohm's law. Hence, the current flowing through each cell depends on its resistance and voltage stress. With a vector of n input voltages $\mathbf{V} = [V_0, ..., V_{n-3}, V_{n-2}, V_{n-1}]$ from wordlines to one particular column of ReRAM cells, as highlighted in red in Fig. 2, aggregated analog current $I = \sum_{n=0}^{n-1} V_i \cdot G_i$ outputs from the bitline, where G_i is the conductance (the reciprocal of resistance, $G_i = 1/R_i$) of the ReRAM cell. If we treat the voltage $\mathbf{V}$ and conductance $\mathbf{G}$ as input vectors, the output $I = \mathbf{V} \cdot \mathbf{G}$ is naturally a result from a mathematical dot-product calculation by $\mathbf{V}$ and $\mathbf{G}$. Since such dot-product operations are predominantly performed in neural network computing, with weight matrices represented by different resistance levels in ReRAM cells, they can be efficiently processed inside ReRAM crossbars.

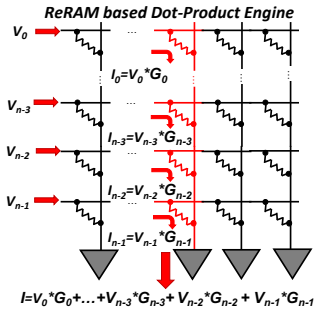


Fig. 2: An ReRAM crossbar based dot-product engine.

C. Neural Network Training

Fig. 3 shows an example of neural network training, which is composed of a forward and a backward propagation. In forward propagation, an input vector $[x_0, x_1, ..., x_n]$ is fed into the network while calculating the intermediate neurons with weight matrices $W_1, W_2, ..., W_4$ in each layer. Afterwards,

an output vector $[y_0, y_1, ..., y_m]$ is computed and taken by a *loss function* to estimate the difference with labeled data. As soon as the loss is obtained, a backward propagation starts by sending the loss back to all layers of the neural network. During this stage, weight matrices are frequently updated with the loss by using: $\Delta \mathbf{W_i} = -\mathcal{LR} \cdot \frac{\partial Loss}{\partial \mathbf{W_i}}$, where $\Delta \mathbf{W_i}$ is the update to each of weight matrix, $\mathcal{LR}$ is the learning rate, and $i = 1, 2, ..., 4$ denotes the number of the layer.

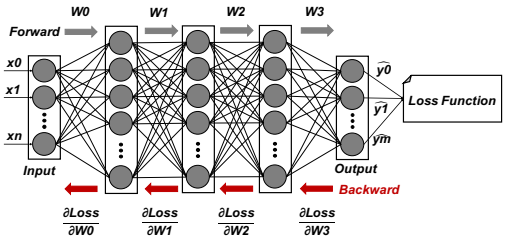


Fig. 3: Neural network training with weight updates.

III. Motivation

In this section, we analyze the ReRAM wearing out mechanism and stochastic switching behaviors, which motivate us to propose innovative solutions for mitigating endurance degradation of ReRAM crossbars during the training.

A. ReRAM Cell Endurance Model

The wearing out mechanism of ReRAM cell has been long studied [18], [24]–[26], which all generally believe excessive programming conditions, such as long programming pulse width and strong pulse amplitude than necessary, i.e., over-SET/-RESET, degrade ReRAM cell endurance. In order to analytically model ReRAM endurance degradation, it is necessary to identify the key factor that can limit the write cycles of an ReRAM cell.

1) Tunneling Gap Distance and R_{off}/R_{on}: Fig. 4 shows the how resistance level is determined during ReRAM cell switching. As we discussed, a formation/rupture of CFs in an ReRAM cell happens during SET/RESET processes. For an instance of RESET process shown in Fig. 4, with a negative voltage stress on top electrode, the CFs are dissolved. A stronger RESET condition can lead to less amount of residual CFs, which thereby exhibits a larger resistance. Based on the ReRAM cell model presented in previous work [27]–[29], the concept of tunneling gap distance g, which denotes an average distance from the top of residual CFs to the top electrode layer, is used to indicate the resistance level of an ReRAM cell during switching. An $I-V$ characteristic equation in an ReRAM cell can be represented as $I = I_0 exp(-g/g_0) sinh(V/V_0)$, where I_0, g_0 and V_0 are fitting constants [28]. In the figure, a tunneling gap $g2$ is larger than $g1$, which implies that a stronger programming condition is needed for switching the cell to $g2$ than $g1$. Consequently, an ReRAM cell with a tunneling gap $g2$ has a larger resistance than the one with $g1$.

Recent studies [30], [31] report that the SET process is abrupt and RESET process is more gradual, and prior studies [27], [32]–[34] also present that RESET takes much longer time and consumes much more energy than SET operation. Therefore, in this paper, we assume SET operation is fast and accurate without consuming much energy, and the endurance degradation principally comes from RESET operation. We also assume that each SET operation accurately switches the cell to R_{on}, and thus the R_{off}/R_{on} ratio is determined by RESET operation condition. However, it is worth noting that our proposed schemes also apply to different ReRAM switching assumptions, such as symmetric SET/RESET operations. With a fixed or variable R_{off}/R_{on} ratio, the relationship between endurance degradation and programming strategies are different, which consequently results in different endurance enhancement solutions.

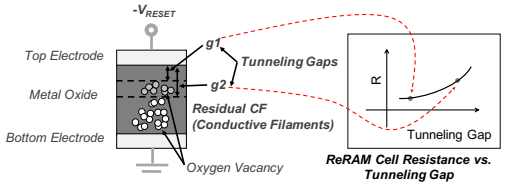


Fig. 4: ReRAM cell switching and its resistance.

2) Fixed R_{off}/R_{on} During Programming: With a fixed R_{off}/R_{on} ratio, recent studies [16], [24] reveal a tradeoff between endurance and programming latency that a longer programming pulse without over-RESET the cell can prolong the cell endurance. The hypothesis of this argument is to switch an ReRAM cell to a fixed resistance level, that is to say, the g is unchanged under different switches [24]. The tradeoff of endurance and write latency can be approximated as: $Endurance \approx (t_W/t_0)^C$, where t_W is the write latency and t_0 and C are fitting parameters. We use the same $C = 2$ as [16], [19] in this paper. Based on this observation, a concept of *effective write* is proposed in [19] to estimate the endurance degradation in ReRAM crossbars with taking sneak current issue and RESET latency discrepancy into consideration. As reported in [19], [34], it is necessary to adopt optimal RESET latency at runtime to avoid excessive write strength. In this paper, the effective write is adopted as the metric to estimate the degrees of wearing out an ReRAM cell:

$$EW = \lceil (\frac{t_L}{t})^2 \rceil \tag{1}$$

where t is the variable latency and t_L is the longest latency in a crossbar with given R_{off}/R_{on}.

3) Variable R_{off}/R_{on} During Programming: In contrast to a fixed R_{off}/R_{on} scenario where prolonging RESET duration to mitigate endurance degradation can be used, with a flexible R_{off}/R_{on}, the endurance is improved in a different approach. A recent research [18] demonstrates that the ReRAM endurance is significantly correlated to R_{off}/R_{on} ratio. The larger R_{off}/R_{on} is, the shorter the lifetime of an ReRAM cell can have. Additionally, the programming pulse width, i.e., RESET latency in this paper, is proportional to R_{off}/R_{on} ratio. Based on above two observations, we present an analytical model by using data from [18] with $CuTe_x/HfO_2$ material (which has the best endurance and hence is suitable for neural

network training) to estimate the lifetime of an ReRAM cell in different RESET latencies:

$$Endurance \approx a \cdot e^{b \cdot WM} \tag{2}$$

and

$$WM \approx p_0 \cdot t + p_1 \tag{3}$$

where WM denotes the R_{off}/R_{on}, t is RESET latency and a, b, p_0, p_1 are fitting constants. While shortening the RESET latency linearly decreases the R_{off}/R_{on} ratio, the endurance is exponentially improved by a reduced R_{off}/R_{on}. Equation 2 and 3 together imply that appropriately optimizing RESET latency may help to achieve better endurance.

B. ReRAM Stochastic Switching

Though with an appropriate RESET condition, ReRAM cell can switch to a targeted resistance level by forming a certain tunneling gap g. However, we should also realize that the programming on ReRAM cell is a stochastic switching [35]. Previous studies [35]–[37] report that the switching behaviors of an ReRAM cell is stochastic, and its probability is predictable with modeling the correlation between programming conditions and successful switching rate. It is worth noting that, a successful switching rate here is defined as — how many successful read-out values (SLC reading mode with values '0'/'1') are as expected out of total read attempts under the same read voltage condition [36], which indicates that a targeted R_{off}/R_{on} should be achieved in order to provide enough read margin. Otherwise, a reduced R_{off}/R_{on} can lead to a uncertain switching.

Two major programming conditions — RESET pulse width (time) and height (amplitude), have significant impact on switching probability. They both in fact affect on R_{off}/R_{on} as discussed before. In this work, we use following Equation 4, as reported in [37] with RESET conditions from [34], to model the correlation between switching probability and RESET time under different pulse heights:

$$P = \frac{1}{2} erfc(-\frac{\ln t_w - \ln \tau}{\sqrt{2\sigma}}) \tag{4}$$

where the P is the ReRAM cell switching probability, $erfc(x)$ is a complementary error function, t_w represents RESET pulse width (write latency), τ and σ are fitting parameters.

Fig. 5 plots a group of curves with ReRAM cell switching probabilities at different RESET voltage widths and heights. In this work, we assume these optimized RESET latencies from [34] guarantee a 100% cell switching. When applying a shorter RESET timing than those under the same data pattern, the switching probability is smaller than 100% and can be predictably computed with Equation 4.

IV. Proposed Designs

In this section, we elaborate our proposed framework, ReNEW, which can effectively improve the endurance of ReRAM crossbar based neural network accelerators.

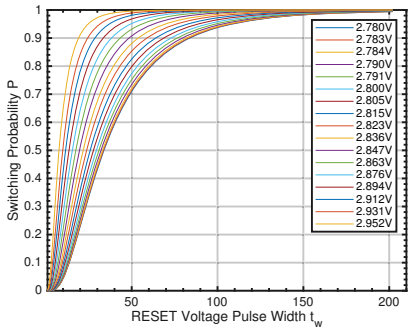


Fig. 5: The correlation between switching probability and RESET voltage width with different RESET pulse heights.

A. Training CNN with SLC ReRAM

As shown in Fig. 6, the ReRAM crossbar based neural network accelerator adopted in this paper has a similar architecture to PRIME [9]. This architecture is composed of several banks, each of which further consists of many ReRAM crossbars. The ReRAM crossbars can be partitioned into *memory array* and *compute array* based on their usage. Memory arrays are to store temporary data, while compute arrays primarily perform in-memory dot-product calculations.

Different from most of prior work [12]–[14], [38] where Multi-Level Cell (MLC) ReRAM crossbars are adopted for neural network training, we propose to program ReRAM crossbars in Single-Level Cell (SLC) mode for training, but well-trained weight matrices for inference task are still programmed in MLC mode. This is because, compared to MLC ReRAM crossbars, using SLC ReRAM crossbars for neural network training owns following advantages: (1) Since an iterative scheme and a large R_{HRS}/R_{LRS} ratio are usually necessary for MLC ReRAM programming, write endurance of an SLC ReRAM cell can be as much as 4-6 magnitudes better than MLC ReRAM according to prior study [17]. (2) With similar reasons to (1), programming an SLC ReRAM cell also performs better than an MLC ReRAM [23] cell in performance and energy-efficiency. As shown in Fig. 6, we use 8-bit fixed-point numbers to represent weight data. It is worth noting that, though we assume bits within the same weight data are stored consecutively in one row, our proposed schemes also apply to different weight mapping approaches without significant changes. To store each of this 8-bit weight data, 8 cells in SLC ReRAM crossbars are required, however, only 2 cells (if 4-bit per cell is assumed) are enough in an MLC crossbar array. Even though MLC ReRAM is several times denser than SLC ReRAM (depends on the resistance levels), the capacity loss is possibly compensated by reloading weights onto SLC ReRAM crossbars for a few times. A comparison of MLC and SLC ReRAM crossbars for neural network training is presented in the experiment section. To enable using multiple cells for representing one weight data, a bit slicing technique [39] is adopted to support dot-product calculations on SLC ReRAM crossbars. In addition, necessary compute units for shift-add operations are also needed to generate final results.

After the completion of neural network training, the trained

weights shall be re-programmed into ReRAM crossbar arrays in MLC mode. This is because, during the training, partitioning and reloading weights onto crossbars are possibly needed due to limited capacity of SLC ReRAM cells. However, an inference task usually demands near real-time response. Hence, reloading weights with using crossbars in SLC mode may be not suitable for inference. Besides, writing well-trained weights into crossbars is not a frequent operation during the inference, therefore, the energy and performance cost are more acceptable than during the training.

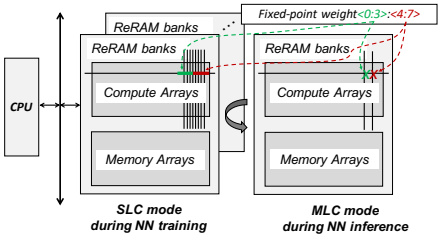


Fig. 6: An overview of ReRAM crossbar based accelerator for neural network computing.

B. Optimized Programming Order

A prior study [19] on write endurance variation shows that having more LRS cells along bitlines benefits the lifetime of ReRAM cells under the premise of no over-SET/-RESET cells. This observation motivates us to optimize the weight updates order by programming cells from '0' to '1' (SET operations) before performing RESET operations. Fig. 7 illustrates a comparison of a sequential weight update, which is row-major order, and the proposed optimized programming scheme that first performs SET operations to increase the number LRS cells in ReRAM crossbars, and then RESET the rest of cells.

By employing the optimized programming order, the effective writes brought by each RESET are expected to decrease, since more LRS cells have been generated beforehand. As a side note, this optimization guarantees successful switching since the targeted R_{off}/R_{on} shall be achieved, and it only prolongs endurance as Equation 1. Therefore, it can be safely applied to either LSB (least-significant bits) columns or MSB (most-significant bits) [1] columns as it basically does not introduce switching errors. Moreover, this optimized programming order does not introduce latency penalty since separate SET and RESET programming phases are also needed in the baseline row-major order [32].

C. Shortened RESET operation

As discussed in Section III, by shrinking R_{off}/R_{on}, the lifetime of ReRAM cells can be exponentially improved. Equation 3 proves that a shortened RESET latency linearly reduces R_{off}/R_{on}. With observations above, we propose to shorten RESET time on selected columns in ReRAM crossbars, such as those contain LSB, to extend their lifetime.

[1]In this paper, we denote MSB/LSB as the most/least significant half in each weight data, e.g., 4 most-significant bits and 4 least-significant bits for an 8-bit weight number.

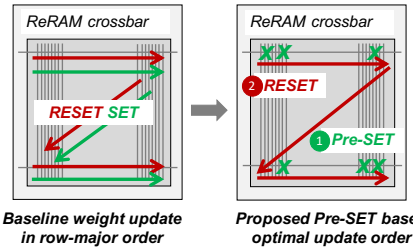


Fig. 7: A comparison of the baseline weight update in row-major order and the proposed optimized programming order.

Though a shortened RESET duration significantly mitigates endurance degradation, in the meantime it potentially brings switching errors, which possibly further results in an accuracy loss. To address this, we propose to exploit the intrinsic error-tolerance characteristic of neural network training, which has been widely reported in prior studies [40], [41], to mitigate the accuracy loss. Intuitively, we are able to achieve a sweet spot in the tradeoff between accuracy and ReRAM crossbar lifetime by tuning the programming time. Table I shows our study on the accuracy degradation with different switching probabilities [2]. Compared to the 8-bit fixed point weight scheme with 100% switching probability, a 95% of switching probability in LSB for both of MLP and CNN model only slightly degrades accuracy by 0.33% and 0.14% respectively. However, if a 95% of switching probability applies to both of MSB and LSB in weight matrices, training of neural network models cannot converge. Therefore, we propose to only shorten the RESET timing on LSB columns in order to avoid a significant accuracy loss.

TABLE I: Model accuracy degradation with different switching probabilities.

Weight Precision	MLP	CNN
4b-MSB (100%)-4b-LSB (100%)	97.86%	90.31%
4b-MSB (100%)-4b-LSB (95%)	97.53%	90.17%
4b-MSB (95%)-4b-LSB (95%)	27.41%	19.56%

A basic workflow of the proposed scheme is shown in Fig. 8. A neural network dependent switching error tolerance, such as 5% of switching error tolerance used in this paper by default, is first decided. This tolerance value is then sent to a look-up table, which maps switching probabilities to their corresponding RESET latencies, to determine the shortened RESET timing. After this, programming commands with optimized RESET timing are issued to LSB columns, while precise programming is applied to MSB columns.

D. Column Group Shift and Update

With shortened RESET operations on LSB columns applied, intuitively there will be a discrepancy in effective writes across LSB and MSB columns in a crossbar array. Besides, a recent study [13] also observed a severe unbalance writes distribution in ReRAM crossbars during the neural network training. To address this issue, we propose to shift and update each half length of weight data within column groups for every training

[2]The experimental methodologies are presented in Section V in detail.

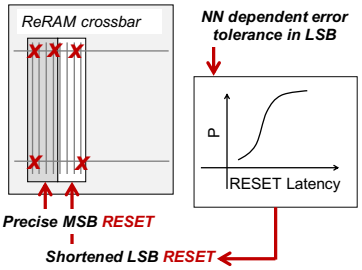


Fig. 8: The precise RESET on MSB columns and shortened RESET on LSB columns.

iteration, as illustrated in Fig. 9. Each column group has all MSB or LSB columns from different weight data that are stored along same bitlines. With assuming 4-bit MSB/LSB and 512 rows in an array, a column group is a 512×4 data chunk that contains 512 4-bit LSB or MSB columns. This technique is inspired by a conventional wear-leveling technique for NVM based main memory [42], which proposed to periodically shift rows and swap pages to improve endurance. However, our design proposes to shift and update weight data on all column groups in each iteration. In addition, shifting and swapping techniques for NVM based main memory incur a huge performance overhead by data exchanging, whereas our proposed shift and update scheme between only requires the address remapping and can perform in each iteration, since weight matrices are updated in each training iteration and not necessarily preserved.

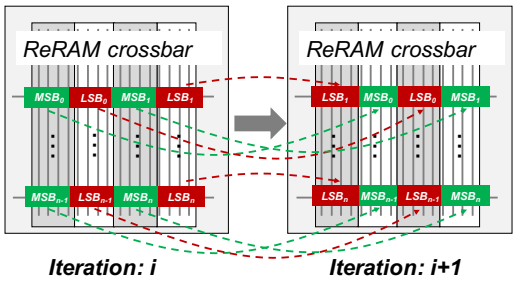


Fig. 9: The proposed column group shift and update scheme.

V. Evaluations

In this section, we first introduce our experimental methodologies, and then present the evaluation results.

A. Experimental Setup

As summarized in Table II, we build two neural network models — MLP and CNN, with Theano [43], and then generate weight matrices by training two models on MNIST [44] and SVHN [45] datasets respectively. In this paper, we test model accuracy with using different precisions of fixed-point weight matrices, and assume input data and intermediate feature maps are accurate. Batch normalization is also adopted in neural network models. For the purpose of evaluating the effectiveness of proposed techniques, we develop a simulator to estimate lifetime of ReRAM crossbars, each of which is a 512×512 array and can contain up to 64 8-bit fixed-point weights per row. This simulator takes weight matrices generated from neural network training as inputs. With a fixed-size of ReRAM crossbar, weight matrices are partitioned and reloaded onto crossbars when the size of matrix is too large for a single crossbar. In opposition, multiple small weight matrices can reside in the same physical ReRAM crossbar to maximize the utilization. The *effective write* [19], which is computed with Equation 1, is used to accumulate wearing out effect on each cell. We adopt RESET latencies from [19] with different bitline data patterns as the optimal programming strategy, and use a conservative timing for the naive solution [34]. We also compare our proposed techniques with 512×512 MLC ReRAM crossbar baselines. We assume one MLC ReRAM cell has 16 resistance levels, i.e., 4-bit data per cell, and it is worn out $1000\times$ (`MLC1000`) and $100\times$ (`MLC100`) faster than SLC [17]. We summarize all evaluated schemes as below:

- `MLC100` and `MLC1000` are MLC ReRAM crossbar baselines, assuming to have $100\times$ and $1000\times$ effective writes than baseline SLC for each write respectively.
- `SLC` is an SLC ReRAM crossbar baseline with a conventional programming strategy.
- `SLC-OP` is an SLC ReRAM crossbar baseline with an optimized programming strategy [19].
- `ReNEW` has all proposed techniques to enhance lifetime of ReRAM crossbars. We evaluate `ReNEW` with different switching probabilities, `ReNEW-0.95` (95%, default), `ReNEW-0.90` (90%) and `ReNEW-0.85` (85%).

TABLE II: Neural networks and datasets.

Model	Dataset	Network Topology
MLP	MNIST	784-240-240-10
CNN	SVHN	conv5x32-pool3-conv4x64-pool3-1000-400-10

B. Accuracy and Lifetime of ReRAM Crossbars Tradeoff

Table III summarizes a tradeoff among the model accuracy, bit-width selection and switching probability of MLP and CNN. Our baseline is a 16-bit fixed-point weight scheme, which consists of 8-bit MSB and 8-bit LSB, without switching errors (100% switching probability). In either MLP or CNN model, this baseline weight precision can achieve the best accuracy.

As reported in recent studies [46]–[48], using low bit-width weights for neural network training can reduce computation complexity, energy consumption while maintaining acceptable accuracy. By storing a less number of bits in weight matrices, the lifetime of ReRAM crossbars can be improved. To prove that our proposed schemes can further mitigate endurance degradation with low bit-width weight matrices, we compare 16-bit, 8-bit and 4-bit weights with no errors (100% switching probability) schemes for MLP and CNN models, and conclude that an 8-bit weight scheme for both models achieves the best tradeoff between accuracy and weight length. The 4-bit weight matrices used in either of two models result in a failure of convergence to an acceptable accuracy.

With the 8-bit weight scheme selected, we further evaluate the impact of switching probability on accuracy. As discussed in Section IV, a 95% switching probability in both MSB and LSB of a weight data results in unsuccessful convergence. We then compare the accuracy with different switching

probabilities, i.e., 95%, 90% and 85%, for LSB in weight data from both of MLP and CNN models. With decreased switching probabilities from 95% to 85%, the accuracy loss slightly increases in MLP model by 0.35%, 0.51% and 0.54% compared to the 16-bit weight scheme respectively, however, the accuracy loss of CNN model dramatically increases by 0.51%, 3.23% and 34.3% respectively. Therefore, we select the 95% switching probability for MLP model by default but a sensitivity study of different switching probabilities for LSB in weight matrices is presented in later section. For CNN model, we also select the switching probability of 95% in LSB due to an unacceptable accuracy degradation incurred by other schemes.

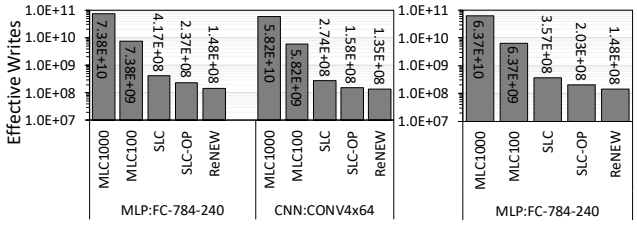
C. Lifetime Improvement

To show the effectiveness of our proposed techniques, we evaluate total effective writes, the maximum number of writes in the worst-case cell for a fully connected layer (`FC-784x240`) in MLP model and a convolutional layer (`CONV4x64`) in CNN model, and also conduct a sensitivity study for `ReNEW` with different switching probabilities.

1) Total Effective Writes: Fig. 10 shows a comparison in total effective writes for MLP layer `FC-784x240` and CNN layer `CONV4x64`. Specifically, in Fig. 10a, we compare the total effective writes among all schemes on ReRAM crossbars until their training converges to the best accuracy. For the `FC-784x240` in MLP, `ReNEW` saves 500.3×, 50.0×, 2.83× and 1.60× total effective writes compared to `MLC1000`, `MLC100`, `SLC` and `SLC-OP` respectively. For `CONV4x64` layer in CNN, `ReNEW` reduces 432.6×, 43.3×, 2.04× and 1.17× total effective writes compared to `MLC1000`, `MLC100`, `SLC` and `SLC-OP` respectively. Since training with `ReNEW` only has 84 epochs, which is 14 less than the ones with 100% switching probability, we also compare the total number of effective writes for MLP layer `FC-784x240` among all schemes with the same number of 84 training epochs (`ReNEW` and baselines for CNN layer `CONV4x64` all experience 98 epochs of training.). Fig. 10b shows that, even with the same number of training epochs, `ReNEW` can still reduce total effective writes by 431.68×, 43.17×, 2.42× and 1.37× than `MLC1000`, `MLC100`, `SLC` and `SLC-OP` respectively.

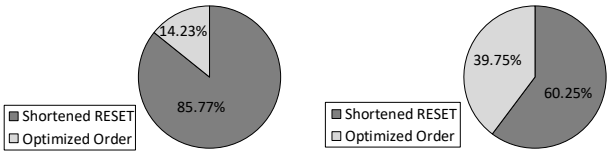
Additionally, we investigate and show the contribution ratio of two techniques — shortened RESET timing and optimized programming order, used to reduce total effective writes, with MLP layer `FC-784x240` in Fig. 11a and CNN layer `CONV4x64` in Fig. 11b. In both of two layers, the major contribution to the reduction in effective writes is from the shortened RESET timing (85.77% and 60.25% respectively). Therefore, with a greater shortened RESET timing, a larger reduction in effective writes is expected to be achieved, especially for those highly error tolerant neural networks.

2) The Maximum Number of Effective Writes in Worst-case Cell: In addition to the evaluation for total effective writes, we also compare the maximum number of effective writes in the worst-case ReRAM cell, which experiences the most accumulated effective writes during the training, among all



(a) (b)

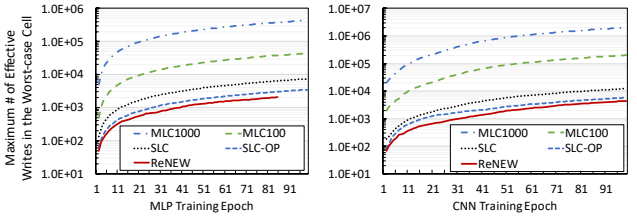
Fig. 10: Total effective writes comparison for MLP and CNN models. (a) Training with different epochs until a convergence to the best accuracy. (b) Effective writes comparison for the MLP layer `FC-784x240` among all schemes with the same number of 84 training epochs.



(a) (b)

Fig. 11: The contribution ratio of shortened RESET timing and optimized programming order techniques for the reduction in effective writes with (a) MLP layer `FC-784x240` and (b) CNN layer `CONV4x64`.

schemes with MLP layer `FC-784x240` in Fig. 12a, and CNN layer `CONV4x64` in Fig. 12b. For the `FC-784x240` in MLP, with 14 less epochs of training, `ReNEW` reduces the maximum number of effective writes by 212.79×, 21.28×, 3.60× and 1.70× compared to `MLC1000`, `MLC100`, `SLC` and `SLC-OP` respectively. However, even with the same number of training epoch of 84, `ReNEW` can still reduce the maximum effective writes by 182.67×, 18.27×, 3.13× and 1.48× than `MLC1000`, `MLC100`, `SLC` and `SLC-OP` respectively. For `CONV4x64` layer in CNN, `ReNEW` can help to improve the maximum effective writes by 460.03×, 46.00×, 2.82× and 1.33× in 98 training epochs when compared to `MLC1000`, `MLC100`, `SLC` and `SLC-OP` respectively. These experimental results prove that our proposed techniques can help to evenly distribute writes across all ReRAM cells during the neural network training.



(a) (b)

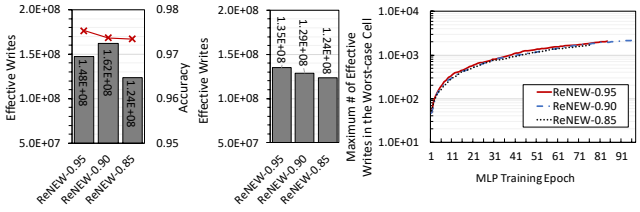
Fig. 12: A comparison of the maximum number of effective writes in the worst-case ReRAM cell for (a) MLP layer `FC-784x240` and (b) CNN layer `CONV4x64`.

3) Sensitivity to Switching Probability: As discussed previously, by decreasing the switching probability for MLP

TABLE III: Tradeoff between model accuracy loss and precisions of weight data.

MLP			CNN		
Weight Precision (Switching Prob.)	**Accuracy (Accuracy Loss)**	**Epoch**	**Weight Precision (Switching Prob.)**	**Accuracy (Accuracy Loss)**	**Epoch**
8b-MSB (100%)-8b-LSB (100%)	97.88% -	85	8b-MSB (100%)-8b-LSB (100%)	90.68% -	97
4b-MSB (100%)-4b-LSB (100%)	97.86% (-0.02%)	98	4b-MSB (100%)-4b-LSB 100%)	90.31% (-0.37%)	98
4b-MSB (100%)-4b-LSB (95%)	97.53% (-0.35%)	84	4b-MSB (100%)-4b-LSB (95%)	90.17% (-0.51%)	98
4b-MSB (100%)-4b-LSB (90%)	97.37% (-0.51%)	97	4b-MSB (100%)-4b-LSB (90%)	87.45% (-3.23%)	57
4b-MSB (100%)-4b-LSB (85%)	97.34% (-0.54%)	77	4b-MSB (100%)-4b-LSB (85%)	56.34% (-34.34%)	100
4b-MSB (95%)-4b-LSB (95%)	27.41% (-70.47%)	92	4b-MSB (95%)-4b-LSB (95%)	19.56% (-71.12%)	17
2b-MSB (100%)-2b-LSB (100%)	16.45% (-81.43%)	20	2b-MSB (100%)-2b-LSB (100%)	19.56% (-71.12%)	0

model from 95% to 85%, the accuracy loss does not significantly increases. To better understand the tradeoff between endurance improvement and switching probability, we test the lifetime enhancement for `ReNEW` with different switching probabilities, i.e., 95% (`ReNEW-0.95`), 90% (`ReNEW-0.90`) and 85% (`ReNEW-0.85`), as shown in Fig. 13. Specifically, Fig. 13a compares the total effective writes and accuracy loss in MLP layer `FC-784x240` for `ReNEW-0.95`, `ReNEW-0.90` and `ReNEW-0.85`. Overall, `ReNEW-0.85` outperforms `ReNEW-0.95` and `ReNEW-0.90` by 1.19× and 1.31× in total effective writes reduction, while at a cost of 0.19% and 0.03% degraded accuracy respectively. `ReNEW-0.90` has a larger number of total effective writes than `ReNEW-0.95` since it is trained with more epochs. As shown in Fig. 13b, with a same number of 77 training epochs, `ReNEW-0.90` reduces a greater number of effective writes than `ReNEW-0.95`, and `ReNEW-0.85` still has the smallest number of effective writes (1.09× and 1.04× smaller than `ReNEW-0.95` and `ReNEW-0.90` respectively). This proves that, the smaller switching probability is, a greater endurance improvement is achieved with given the same amount of updates. Fig. 13c presents a comparison of maximum number of effective writes for MLP layer `FC-784x240`, wherein `ReNEW-0.85` achieves a smaller number maximum effective writes than `ReNEW-0.95` and `ReNEW-0.90` by 1.28× and 1.20× when all three schemes converge to the best accuracy. With the same number of 77 training epochs, `ReNEW-0.85` outperforms `ReNEW-0.95` and `ReNEW-0.90` by 1.10× and 1.07× respectively.



(a) (b) (c)

Fig. 13: A sensitivity study for `ReNEW` with different switching probabilities in MLP layer `FC-784x240`: (a) total effective writes and accuracy with different training epochs, (b) total effective writes with a same number of training epochs, and (c) the maximum number of effective writes.

VI. Related Work

Neural network computing with ReRAM. Recent studies on neural network accelerators exploit the natural analog current accumulation feature of ReRAM crossbar architecture to implement dot-product calculations [9], [11]–[13], [38], [39], [49]–[61], wherein there are many [12]–[14], [38], [61] supporting neural networks training in ReRAM crossbars.

Wear leveling techniques for Crossbar ReRAM. Wear-leveling techniques for NVM based memories have been widely studied, which share a general idea of evenly distributing write accesses across pages [19]–[22]. To address the endurance issue of ReRAM crossbar based neural network accelerators, a software and hardware co-optimization is proposed [14]. Unfortunately, this scheme only works for MLC ReRAM crossbars. Prior work [13] exploits gradient sparsification in neural networks and a row remapping scheme to improve ReRAM endurance, which is in fact complementary to our designs. A recent study on using low-precision weights [46] for CNN training can be also used to mitigate ReRAM crossbar endurance degradation. As shown in our evaluations, this is also orthogonal to our design since `ReNEW` can further improve endurance in low bit-width weight matrices during the training.

Improving endurance by exploiting stochastic switching. An approximate switching scheme is also proposed to improve the endurance in [62] for NVM based FF design, but this work is lack of an analytical study between switching probability and enhanced lifetime.

VII. Conclusion

In this paper, we are motivated by analyses of the endurance degradation mechanism in ReRAM cell, and then propose a novel framework, ReNEW, to enhance the lifetime of the ReRAM crossbar based neural network accelerators, especially for neural network training that requires frequent weight updates. The experimental evaluations show that, our proposed `ReNEW` reduces the total effective writes to ReRAM crossbar based accelerators by up to 500.3×, 50.0×, 2.83× and 1.60× over two MLC baselines, SLC baseline and SLC design with optimal timing respectively.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[3] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[4] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[5] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on cpus," in *in Deep Learning and Unsupervised Feature Learning Workshop, NIPS*. Citeseer, 2011.

[6] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2015, pp. 161–170.

[7] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun *et al.*, "Dadiannao: A machine-learning supercomputer," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2014, pp. 609–622.

[8] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2017, pp. 1–12.

[9] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *Proceedings of the 43rd International Symposium on Computer Architecture*. IEEE Press, 2016, pp. 27–39.

[10] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: programming 1t1m crossbar to accelerate matrix-vector multiplication," in *Design Automation Conference (DAC), 2016 53nd ACM/EDAC/IEEE*. IEEE, 2016, pp. 1–6.

[11] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *Proceedings of the 43rd International Symposium on Computer Architecture*. IEEE Press, 2016, pp. 14–26.

[12] L. Song, X. Qian, H. Li, and Y. Chen, "Pipelayer: A pipelined reram-based accelerator for deep learning." HPCA, 2017.

[13] Y. Cai, Y. Lin, L. Xia, X. Chen, S. Han, Y. Wang, and H. Yang, "Long live time: improving lifetime for training-in-memory engines by structured gradient sparsification," in *Proceedings of the 55th Annual Design Automation Conference*. ACM, 2018, p. 107.

[14] S. Zhang, G. L. Zhang, B. Li, H. H. Li, and U. Schlichtmann, "Aging-aware lifetime enhancement for memristor-based neuromorphic computing," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1751–1756.

[15] H. Wu, X. H. Wang, B. Gao, N. Deng, Z. Lu, B. Haukness, G. Bronner, and H. Qian, "Resistive random access memory for future information processing system," *Proceedings of the IEEE*, 2017.

[16] L. Zhang, B. Neely, D. Franklin, D. Strukov, Y. Xie, and F. T. Chong, "Mellow writes: Extending lifetime in resistive memories through selective slow write backs," in *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*. IEEE, 2016, pp. 519–531.

[17] D. Niu, Q. Zou, C. Xu, and Y. Xie, "Low power multi-level-cell resistive memory design with incomplete data mapping," in *2013 IEEE 31st International Conference on Computer Design (ICCD)*. IEEE, 2013, pp. 131–137.

[18] C. Nail, G. Molas, P. Blaise, G. Piccolboni, B. Sklenard, C. Cagli, M. Bernard, A. Roule, M. Azzaz, E. Vianello *et al.*, "Understanding rram endurance, retention and window margin trade-off using experimental results and simulations," in *Electron Devices Meeting (IEDM), 2016 IEEE International*. IEEE, 2016, pp. 4–5.

[19] W. Wen, Y. Zhang, and J. Yang, "Wear leveling for crossbar resistive memory," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[20] M. Zhao, L. Jiang, Y. Zhang, and C. J. Xue, "Slc-enabled wear leveling for mlc pcm considering process variation," in *Proceedings of the 51st Annual Design Automation Conference*. ACM, 2014, pp. 1–6.

[21] X. Zhang and G. Sun, "Toss-up wear leveling: Protecting phase-change memories from inconsistent write patterns," in *Proceedings of the 54th Annual Design Automation Conference 2017*. ACM, 2017, p. 3.

[22] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali, "Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2009, pp. 14–23.

[23] C. Xu, D. Niu, N. Muralimanohar, N. P. Jouppi, and Y. Xie, "Understanding the trade-offs in multi-level cell reram memory design," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2013, pp. 1–6.

[24] D. B. Strukov, "Endurance-write-speed tradeoffs in nonvolatile memories," *Applied Physics A*, vol. 122, no. 4, p. 302, 2016.

[25] Y. Y. Chen, B. Govoreanu, L. Goux, R. Degraeve, A. Fantini, G. S. Kar, D. J. Wouters, G. Groeseneken, J. A. Kittl, M. Jurczak *et al.*, "Balancing set/reset pulse for $> 10^{10}$ endurance in HfO_2/Hf 1t1r bipolar rram," *IEEE Transactions on Electron devices*, vol. 59, no. 12, pp. 3243–3249, 2012.

[26] P. Huang, B. Chen, Y. Wang, F. Zhang, L. Shen, R. Liu, L. Zeng, G. Du, X. Zhang, B. Gao *et al.*, "Analytic model of endurance degradation and its practical applications for operation scheme optimization in metal oxide based rram," in *Electron Devices Meeting (IEDM), 2013 IEEE International*. IEEE, 2013, pp. 22–5.

[27] S. Yu, X. Guan, and H.-S. P. Wong, "On the switching parameter variation of metal oxide rram—part ii: Model corroboration and device design strategy," *IEEE Transactions on Electron Devices*, vol. 59, no. 4, pp. 1183–1188, 2012.

[28] P.-Y. Chen and S. Yu, "Compact modeling of rram devices and its applications in 1t1r and 1s1r array design," *IEEE Transactions on Electron Devices*, vol. 62, no. 12, pp. 4022–4028, 2015.

[29] Z. Jiang, S. Yu, Y. Wu, J. H. Engel, X. Guan, and H.-S. P. Wong, "Verilog-a compact model for oxide-based resistive random access memory (rram)," in *Simulation of Semiconductor Processes and Devices (SISPAD), 2014 International Conference on*. IEEE, 2014, pp. 41–44.

[30] W. Wu, H. Wu, B. Gao, N. Deng, S. Yu, and H. Qian, "Improving analog switching in hfo x-based resistive memory with a thermal enhanced layer," *IEEE Electron Device Letters*, vol. 38, no. 8, pp. 1019–1022, 2017.

[31] J. Lin and J.-S. Yuan, "Analysis and simulation of capacitor-less reram-based stochastic neurons for the in-memory spiking neural network," *IEEE transactions on biomedical circuits and systems*, no. 99, pp. 1–14, 2018.

[32] C. Xu, D. Niu, N. Muralimanohar, R. Balasubramonian, T. Zhang, S. Yu, and Y. Xie, "Overcoming the challenges of crossbar resistive memory architectures," in *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*. IEEE, 2015, pp. 476–488.

[33] H. Zhang, N. Xiao, F. Liu, and Z. Chen, "Leader: Accelerating reram-based main memory by leveraging access latency discrepancy in crossbar arrays," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016*. IEEE, 2016, pp. 756–761.

[34] W. Wen, L. Zhao, Y. Zhang, and J. Yang, "Speeding up crossbar resistive memory by exploiting in-memory data patterns," in *Computer-Aided Design (ICCAD), 2016 IEEE/ACM International Conference on*. IEEE, 2017, pp. 261–267.

[35] H. Li, P. Huang, B. Gao, X. Liu, J. Kang, and H.-S. P. Wong, "Device and circuit interaction analysis of stochastic behaviors in cross-point rram arrays," *IEEE Transactions on Electron Devices*, vol. 64, no. 12, pp. 4928–4936, 2017.

[36] S. Gaba, P. Knag, Z. Zhang, and W. Lu, "Memristive devices for stochastic computing," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 2592–2595.

[37] M. Shirasawa, M. E. Dlamini, and Y. Kamakura, "Kinetic monte carlo simulation for switching probability of reram," in *Future of Electron Devices, Kansai (IMFEDK), 2016 IEEE International Meeting for*. IEEE, 2016, pp. 1–1.

[38] X. Qiao, X. Cao, H. Yang, L. Song, and H. Li, "Atomlayer: a universal reram-based cnn accelerator with atomic layer computation," in *Pro-*

ceedings of the 55th Annual Design Automation Conference. ACM, 2018, p. 103.

[39] 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.

[40] L. Zhao, Y. Zhang, and J. Yang, "Aep: An error-bearing neural network accelerator for energy efficiency and model protection," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2017, pp. 1047–1053.

[41] P. Gysel, M. Motamedi, and S. Ghiasi, "Hardware-oriented approximation of convolutional neural networks," *arXiv preprint arXiv:1604.03168*, 2016.

[42] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," in *ACM SIGARCH computer architecture news*, vol. 37, no. 3. ACM, 2009, pp. 14–23.

[43] T. T. D. Team, R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov *et al.*, "Theano: A python framework for fast computation of mathematical expressions," *arXiv preprint arXiv:1605.02688*, 2016.

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

[45] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[46] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[47] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *International Conference on Machine Learning*, 2015, pp. 1737–1746.

[48] Y. Cai, T. Tang, L. Xia, M. Cheng, Z. Zhu, Y. Wang, and H. Yang, "Training low bitwidth convolutional neural network on rram," in *Proceedings of the 23rd Asia and South Pacific design automation conference*. IEEE Press, 2018, pp. 117–122.

[49] M. N. Bojnordi and E. Ipek, "Memristive boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning," in *High Performance Computer Architecture (HPCA), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 1–13.

[50] D. Fujiki, S. Mahlke, and R. Das, "In-memory data parallel processor," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 2018, pp. 1–14.

[51] A. Nag, R. Balasubramonian, V. Srikumar, R. Walker, A. Shafiee, J. P. Strachan, and N. Muralimanohar, "Newton: Gravitating towards the physical limits of crossbar acceleration," *IEEE Micro*, vol. 38, no. 5, pp. 41–49, 2018.

[52] B. Feinberg, U. K. R. Vengalam, N. Whitehair, S. Wang, and E. Ipek, "Enabling scientific computing on memristive accelerators," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 367–382.

[53] M. K. F. Lee, Y. Cui, T. Somu, T. Luo, J. Zhou, W. T. Tang, W.-F. Wong, and R. S. M. Goh, "A system-level simulator for rram-based neuromorphic computing chips," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 15, no. 4, p. 64, 2019.

[54] C. Liu, M. Hu, J. P. Strachan, and H. Li, "Rescuing memristor-based neuromorphic design with high defects," in *Design Automation Conference (DAC), 2017 54th ACM/EDAC/IEEE*. IEEE, 2017, pp. 1–6.

[55] L. Ni, Z. Liu, W. Song, J. J. Yang, H. Yu, K. Wang, and Y. Wang, "An energy-efficient and high-throughput bitwise cnn on sneak-path-free digital reram crossbar," in *Low Power Electronics and Design (ISLPED, 2017 IEEE/ACM International Symposium on*. IEEE, 2017, pp. 1–6.

[56] M. V. Beigi and G. Memik, "Thermal-aware optimizations of reram-based neuromorphic computing systems," in *Proceedings of the 55th Annual Design Automation Conference*. ACM, 2018, p. 39.

[57] H. Ji, L. Jiang, T. Li, N. Jing, J. Ke, and X. Liang, "Hubpa: high utilization bidirectional pipeline architecture for neuromorphic computing," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. ACM, 2019, pp. 249–254.

[58] P. Bruel, S. R. Chalamalasetti, C. Dalton, I. El Hajj, A. Goldman, C. Graves, W.-m. Hwu, P. Laplante, D. Milojicic, G. Ndu *et al.*, "Generalize or die: Operating systems support for memristor-based accelerators," in *2017 IEEE International Conference on Rebooting Computing (ICRC)*. IEEE, 2017, pp. 1–8.

[59] L. Gao, F. Alibart, and D. B. Strukov, "Analog-input analog-weight dot-product operation with ag/a-si/pt memristive devices," in *VLSI and System-on-Chip, 2012 (VLSI-SoC), IEEE/IFIP 20th International Conference on*. IEEE, 2012, pp. 88–93.

[60] L. Han, Z. Shen, Z. Shao, H. H. Huang, and T. Li, "A novel reram-based processing-in-memory architecture for graph computing," in *Non-Volatile Memory Systems and Applications Symposium (NVMSA), 2017 IEEE 6th*. IEEE, 2017, pp. 1–6.

[61] W. Wang and B. Lin, "Trained biased number representation for reram-based neural network accelerators," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 15, no. 2, p. 15, 2019.

[62] M. Biglari, T. Lieske, and D. Fey, "High-endurance bipolar reram-based non-volatile flip-flops with run-time tunable resistive states," in *2018 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*. IEEE, 2018, pp. 1–6.