Level Scaling and Pulse Regulating to Mitigate the Impact of the Cycle-to-Cycle Variation in Memristor-Based Edge AI System

Jingyan Fu, *Student Member, IEEE*, Zhiheng Liao, and Jinhui Wang, *Senior Member, IEEE*

***Abstract*— As a novel nonvolatile device, the memristor has already delivered many of its promises including low computation complexity and high energy efficiency for the edge artificial intelligence (AI) system in Internet of Things (IoT) applications. However, the intrinsic variability of switching behavior of memristors has been a major obstacle to their implementation. In this study, first, with the Al/TiO_2/TiO_{2-x}/Al stack structure, memristive crossbar chips are fabricated and tested, and then, we present a model that experimentally demonstrates and quantifies the natural stochasticity of cycle-to-cycle variations. Finally, we propose level scaling and pulse regulating methods to mitigate the adverse impact of cycle-to-cycle variations. The relationship of the level of conductance and cycle-to-cycle variation is studied, and the experiment results show an optimal number of the levels to mitigate the impact of cycle-to-cycle variations in the system. Additionally, the system compresses the number of pulses when the conductance is updated by the pulse stimulus to reduce cycle-to-cycle variations, resulting in the great energy and latency reduction. This work paves the way for the adoption of memristors for more efficient applications for the era of the edge computing in IoT.**

***Index Terms*— Artificial intelligence (AI), cycle-to-cycle variation, edge computing, Internet of Things (IoT), level scaling, memristor, neural network, pulse regulating (PR).**

I. Introduction

THE Internet of Things (IoT) system is a network of devices, sensors, and other items of various functionalities that interact and exchange data electronically [1]. Recent years have witnessed significant progress in edge devices and wireless sensor networks, creating unprecedented opportunities to deploy deep learning and artificial intelligence (AI) technologies in IoT, while significantly adding calculation burdens in edges [2]. However, edges consisting of mobile devices and embedded systems usually have limited resources and power, especially when they are used for real-time applications, and such resource and power deficiency will result in recognition and inference accuracy loss in a learning system and even malfunctions in IoT [1]–[7].

The conventional CMOS technology plateaus the process scaling [7], [8], which cannot provide satisfied solutions for the emerging edge computing with designated learning systems [6]. Memristors are theoretically postulated by Chua [9] and later are physically fabricated by the Hewlett-Packard in 2008 [10]. The memristor-based crossbar arrays with storage and computing capability show great potential in the neural-network and machine learning applications [10]–[15]. They are characterized with low computational complexity [16], low power consumption [17], fast switching speed [18], high endurance [19], excellent scalability [20], and CMOS compatibility [21], which are especially appropriate for edge computing in IoT. However, because of the inherent material property of memristors, the intrinsic variation in switching conductance is a major challenge for some applications [22]. For example, cycle-to-cycle variation is the deviation between target conductance and updated conductance when the same updating signal in different updating cycles is applied in a memristor, even when the initial conductance is the same [23]. The memristor-based crossbar arrays suffer from the serious cycle-to-cycle variation especially when they are used in the real-time edge AI system, where the conductance of memristor needs to be updated innumerable times during the training and inference process [24]–[28].

Therefore, in this study, we proposed two methods to mitigate the adverse impact of cycle-to-cycle variations in edge AI systems. Specifically, the following contributions are made in this article.

1) Memristive crossbar chips with the Al/TiO_2/TiO_{2-x}/Al stack structure are fabricated and thoroughly tested based on our in-house technology.
2) We build a mathematical model based on the testing data, which experimentally demonstrates and quantifies the natural stochasticity of cycle-to-cycle variations.

Manuscript received January 17, 2022; accepted January 18, 2022. Date of publication February 10, 2022; date of current version March 28, 2022. This work was supported in part by the National Science Foundation under Grant 1855646 and Grant 1953544. The review of this article was arranged by Editor B. K. Kaushik. *(Jingyan Fu and Zhiheng Liao contributed equally to this work.) (Corresponding author: Jinhui Wang.)*

Jingyan Fu and Zhiheng Liao are with the Department of Electrical and Computer Engineering, North Dakota State University, Fargo, ND 58102 USA (e-mail: jingyan.fu@ndsu.edu; zhiheng.liao@ndsu.edu).

Jinhui Wang is with the Department of Electrical and Computer Engineering, University of South Alabama, Mobile, AL 36688 USA (e-mail: jwang@southalabama.edu).

Color versions of one or more figure in this article are available at https://doi.org/10.1109/TED.2022.3146801.

Digital Object Identifie 10.1109/TED.2022.3146801

3) A level scaling method is proposed to optimize the level number of conductance in memristors to mitigate cycle-to-cycle variations.
4) A pulse regulating (PR) method is proposed in this article. It compresses the number of update pulses to one when the memristors get a conductance update. The target of the proposed technique is to alleviate and compensate inference accuracy degradation resulting from the cycle-to-cycle variation.
5) Based on NeuroSim, a platform is built up to verify the effectiveness of our proposed level scaling and PR methods regarding the accuracy improvement and reductions of energy consumption, system latency, and chip areas.

II. Related Work

In previous works, researchers proposed some solutions including three aspects to mitigate the impact of the cycle-to-cycle variation from memristors.

In software-based and algorithm perspective, a conversion algorithm is invented to map arbitrary matrix values appropriately to memristor conductance to reduce computational errors in [31]; algorithms of the mutual decision between conductance of memristor and Boolean functions are used to tolerate a maximum variation in [32]; a novel off-device neural-network training method is proposed to improve the performance of the neural network in [33]. However, because variations usually come from memristor devices, hardware of the edge AI system, the software-based methods are usually resources-consuming.

In circuit perspective, the smart programming scheme (read the conductance before writing it) and dummy column technologies to eliminate the OFF-state current are utilized to improve immunity to cycle-to-cycle variations in [29], [30], and [34]. The experimental result shows that the accuracy is improved to 95% from 70%. In addition, a variation-aware training scheme is used to enhance training robustness in [35]. However, sophisticated circuits are needed in the above technologies to ensure the quality of conductance switching and either drastically increase the area of circuit and power consumption or bring additional circuit latency.

In device perspective, instead of using a single memristor, the multiple cells technology using several memristors connected in parallel is applied to improve the variation tolerance [36], [37]. But the multiple cells produce area overhead in the system. In addition, the different materials, such as TiO_x as a buffer layer [38] and CeO_2/Ti/CeO_2 tri-layered as an active layer [39], are proposed and investigated to improve the resistance of ratio between high-resistance state and low-resistance state, enhance the endurance of switching, and reduce the variation of the threshold voltage.

Therefore, developing a cycle-to-cycle variation model on memristor is urgent for mitigating the impact, so that it is accessible to apply the great potential and advantages of the memristor in edge AI and IoT applications. Different from the previous approaches, we propose two methods that can mitigate the impact of cycle-to-cycle variations of memristors based on the proposed model.

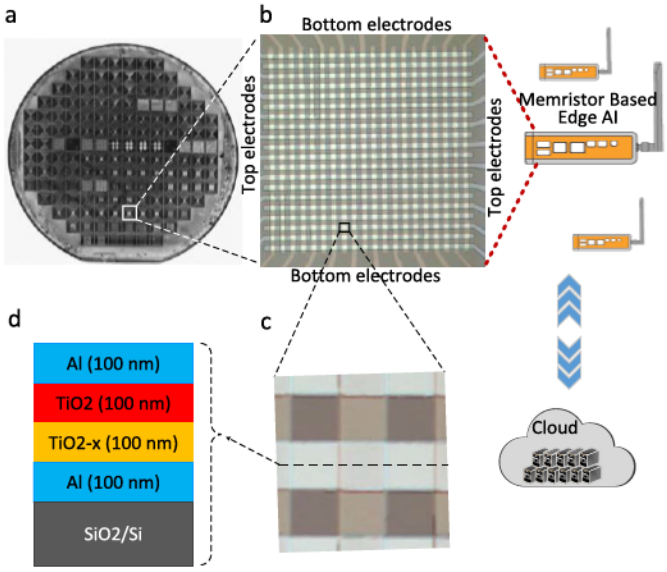

Fig. 1. Memristive crossbar arrays and edge computing schematic. (a) Optical image of a wafer with memristive crossbar arrays. (b) Close-up of chip image showing crossbar arrays. (c) Microscope image showing one memristor device. (d) Cross-sectional schematic of the TiO_2/TiO_{2-x} memristor structure.

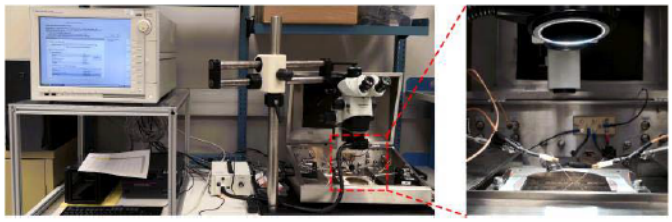

Fig. 2. Testing platform. Kaysight B1500a semiconductor parameter analyzer is used to test memristor crossbar array. The wafer was set on the Micromanipulator probe station and the pads were contacted by gold probe tips.

III. Memristive Crossbar Array and Related Background

A. Memristive Crossbar Array

We fabricate and test the memristive crossbar chips in our laboratory. The optical image and geometry of a TiO_2/TiO_{2-x} based memristive crossbar arrays used in this work is schematically shown in Fig. 1(a). The array is composed of 20 × 20 memristors, as shown in Fig. 1(b). Physically, a memristor is a 40 μm × 40 μm two-terminal device formed by two aluminous electrodes sandwiching a thin active layer, that is, TiO_2/TiO_{2-x} material, to achieve stable tunable multilevel behavior with a nonlinear current–voltage (I–V) relationship, as illustrated in Fig. 1(c). Fig. 1(d) shows that the memristor has an Al/TiO_2/TiO_{2-x}/Al stack in cross section.

I–V characteristics from positive and negative voltage sweeping are carried out using a Kaysight B1500a semiconductor parameter analyzer in the voltage-sweep and voltage-pulse modes. The wafer is set on the Micromanipulator probe station, and the pads are contacted by probe tips, as shown in Fig. 2.

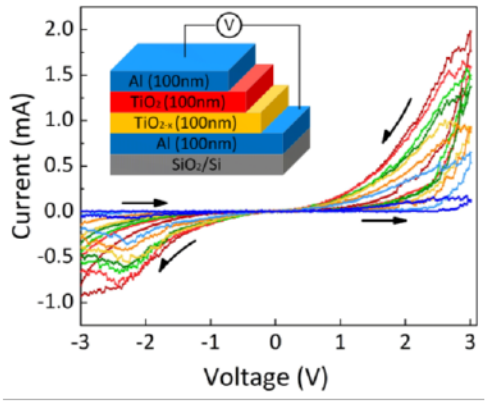

Fig. 3. I–V characteristics from positive and negative voltage sweeping switching (−3 to 3V) in the memristor.

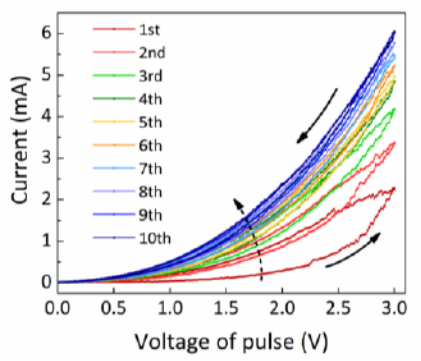

Fig. 4. I–V characteristics from consecutive positive voltage pulses sweeps showing a continuous increase in conductance. The width of pulses is 1 ms and the step of voltage is 1.5 mV.

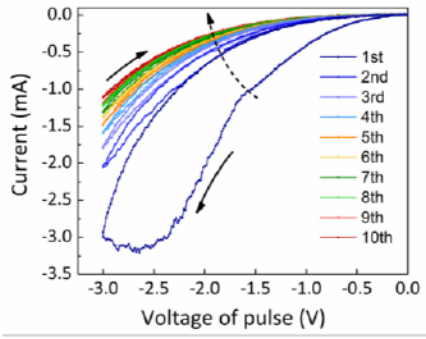

Fig. 5. Consecutive negative voltage pulses sweeps showing a continuous decrease in conductance. The width of pulses is 1 ms and the step of voltage is 1.5 mV.

This TiO_2/TiO_{2-x} memristor displays obvious multilevel behavior, as Fig. 3 shows the current–voltage response of the memristor when the full range voltage sweeps during different cycles. For further investigating this multilevel property, the positive and negative voltage sweeping are separately applied in the same memristor with ten cycles, as shown in Figs. 4 and 5. The conductance of the memristor is changed when the voltage achieves the threshold voltage, which is caused by conducting filament formation across the electrodes [40], [41].

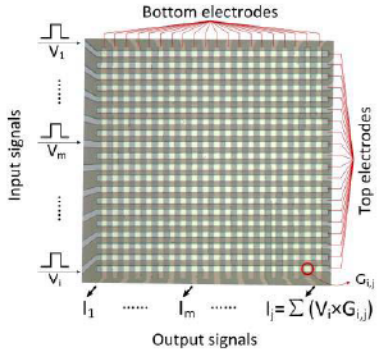

Fig. 6. Hardware implementation of the vector-matrix-multiplication using memristor crossbar. V_i, G_{ij}, and I_j represent the input signal in the ith row, the conductance of the memristor in the jth column and the ith row, and the output current that represents the dot product result of V and G, respectively.

Memristive crossbar arrays carry out the vector-matrix multiplication, as shown in Fig. 6. Every row of the crossbar array gets input voltage pulses that are the vector. Each conductance of the device in every cross point composes the matrix. Every column of the crossbar array transmits an output current that is the sum of multiplication by the input signal and conductance in each cross point. To update the conductance of a memristor that has multilevel conductance from the minimum to the maximum, a positive pulse signal is applied to increase the conductance, which is called long-term potentiation (LTP) [42]. Conversely, long-term depression (LTD) is the process of decreasing the conductance by supplying a negative pulse signal until the conductance gets to the minimum [42]. Multilevel memristors effectively utilize such multivalue conductance to learn the features of data and realize an edge AI system [11], [12], [28].

B. Level of Conductance

In practice, the width of the pulse signal that is used to update the weight cannot be infinitely narrow and limits the accuracy of the conductance updating. Different widths of the pulses change the different amount of the conductance. Therefore, the widths of different pulses that are used for weight update decide the number of the levels, as shown in Fig. 7. The number of these levels can be expressed qualitatively as follows:

$$\lceil (G_{\max} - G_{\min})/W_{\text{pulse}} \rceil = \text{number of levels} \quad (1)$$

where $G_{\max}$ and $G_{\min}$ are the maximum conductance and the minimum conductance and W_{pulse} is the width of the updating pulse. Note that although a higher number of the levels give more precise conductance in the weight update of the memristor, the influence of cycle-to-cycle variation will increase, which is shown in the next section.

C. Cycle-to-Cycle Variation

Since the switching mechanism of the memristor conductance is prompted by the applied voltage, a memristor switches its conductance level from one to another when pulse is larger

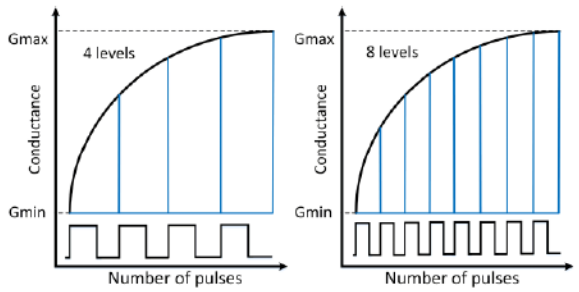

Fig. 7. Level of conductance with different widths of the updating pulses.

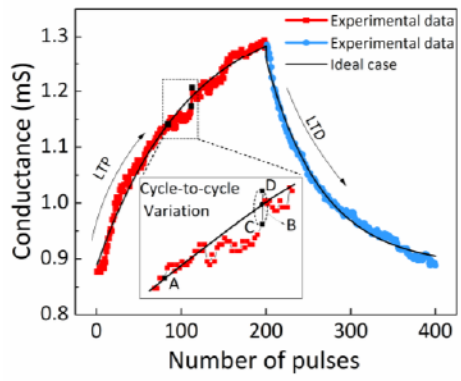

Fig. 8. Cycle-to-cycle variation of memristive device. Experimental data of conductance values measured by pulses during 200 potentiation and 200 depression pulses. Cycle-to-cycle variation is the deviation between target conductance and realistic updated conductance by pulses.

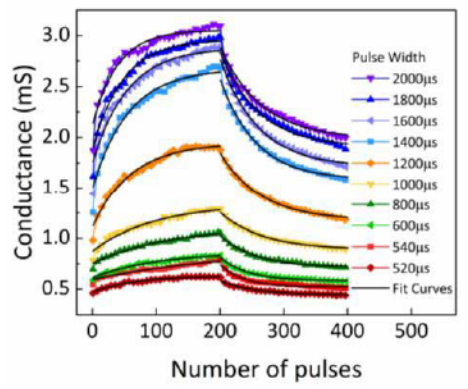

Fig. 9. LTP and LTD process with different pulse widths from 520 to 2000 μs. The black curves are fitted by exponential formula.

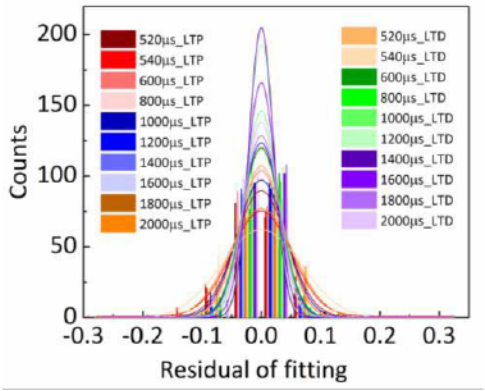

Fig. 10. Residual analysis of fitted Gaussian distribution data after normalization of the deviation from Fig. 9 in different pulse widths.

than a threshold voltage for at least the minimum required time [43]. Simultaneously, cycle-to-cycle variation results in different updated conductance when the same updating signal in different updating cycles is applied in a memristor, even when the initial conductance is the same, as shown in Fig. 8. For instance, for some given updating pulses, a memristor starting at conductance A and target conductance is B may end up between C and D, as shown in the inset of Fig. 8. Memristors exhibit cycle-to-cycle variations because of the shape of the conductive filament, the oxygen vacancy distribution at and around the filament, and the changing location of the active filament between one cycle to the next. These three mechanisms originate from the coexistence of multiple subfilaments and that the active, current-carrying filament may change from cycle to cycle [23]. Thus, cycle-to-cycle variation is a type of inherent randomness associated with the randomness in internal atomic configurations [26], [44], [45]. One of the major obstacles for the implementation of redox-based multilevel memristive memory or logic technology is the large cycle-to-cycle variation [23].

IV. Module of Cycle-to-Cycle Variation and Methods

A. Module of Cycle-to-Cycle Variation

Fig. 9 shows the LTP and LTD process with different pulse widths that are obtained with testing platform, as shown in Fig. 2. We can use the same method in [29] to fit these LTP and LTD experimental data with exponential formulas, as shown in the fitting curves of Fig. 9. Because the fitted curve and stochastic behavior of the cycle-to-cycle variation can be approximated with a Gaussian distribution [45], [46], residual analysis is performed by Gaussian distribution fitting after normalization, as illustrated in Fig. 10.

Because stochastic behavior of the cycle-to-cycle variation can be approximated as Gaussian distribution [44], [45], the value of cycle-to-cycle variation corresponding to one pulse can be defined as

$$\psi = N(0, \delta) \tag{2}$$

where ψ is a cycle-to-cycle variation that is generated for one pulse update process, $N(0, \delta)$ is the Gaussian noise, and δ is the standard deviation. The value of cycle-to-cycle variation corresponding to two pulses can be calculated as

$$\psi_{1+2} = N_1(0, \delta_1) + N_2(0, \delta_2). \tag{3}$$

Because different cycle-to-cycle variations obey Gaussian distributions and that are independent and identically distributed corresponding to different one pulse, $\delta_1 = \delta_2 = \delta$, (3) can be converted as [47]

$$\psi_{1+2} = N(0, \delta_1 + \delta_2) = N(0, 2\delta) = N(0, \delta)^{*}\text{sqrt2}. \tag{4}$$

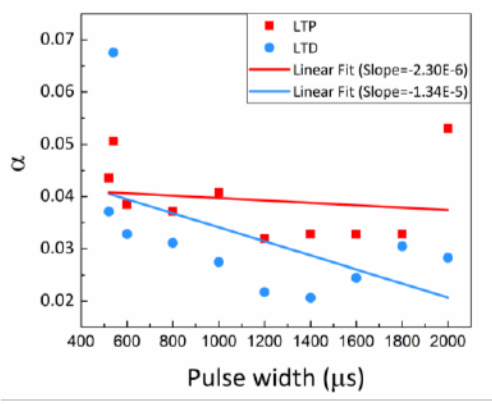

Fig. 11. Extraction of the coefficien between 0 to 1 from the standard deviation of the Gaussian distribution fi ting from Fig. 10 for different pulse widths.

Therefore, we model the total cycle-to-cycle variation that is generated for one memristor at one update process with n pulses and it can be calculated as

$$\Psi_{\text{total}} = N(0, \delta)^* \text{sqrt } n \tag{5}$$

$$\delta = \alpha^*(G_{\max} - G_{\min}) \tag{6}$$

where α is the coefficient that is the percentage of difference of the maximum conductance and the minimum conductance, $G_{\max}$ is the maximum conductance, and $G_{\min}$ is the minimum conductance.

After Gaussian distribution fitting, we get a distribution of α values with different pulse widths, such that the average is 0.03577, as shown in Fig. 11. The lines are linear fitting for α values of LTP and LTD. Both slopes are negative. Therefore, it can be concluded that increasing the pulsewidth does not increase the cycle-to-cycle variation when using the same number of pulses to tune the conductance.

B. Level Scaling Method

A number of the levels are set in the circuit parameter configuration step according to the workflow of Fig. 12(a), which means that there is a certain number of the levels that conductance of memristor can be obtained between the maximum and minimum conductance. The number of the levels will map to the width of the pulse that is generated from the pulse generator in hardware implementation. The higher number of the levels corresponding to the narrower pulses. Theoretically, the system can achieve higher precision of weight for the desired value of conductance. Simultaneously, however, a higher number of the levels introduce larger cycle-to-cycle variation when the system updates the conductance of memristors. This is because the pulse generator produces more pulses to tune the conductance when the algorithm calculates the same Δweight than that system has a lower number of the levels. Therefore, the level scaling method is applying to appropriately reduce the number of the levels. In this work, the number of the levels is a parameter and is set from 10 to 200 and the step is 10.

C. PR Method

For the conventional method to update the conductance of a memristor, according to the value of weight change that is calculated by the algorithm, the control circuit will generate corresponding signals to control the pulse generator for producing positive/negative pulses and tuning the conductance of the memristor. For the PR method, one multiplexer is used to compress the number of pulses for the Δweight and generate only one updating pulse whenever a conductance of a memristor needs to be tuned according to the hardware implementation diagram in path 2 (golden block) of Fig. 12(b). The PR method only applies one pulse and keeps the original width of writing pulses in each weight update. The decoder gets a signal from an arithmetic logic unit (ALU) for selecting one row to update. At the same time, the registers get the values of Δweight that are calculated by an ALU. Then, these values are transmitted to multiplexers as control signals. Multiplexers select one writing pulse that comes from a pulse generator as an output when control signals are enabled. The enabled signal means that the corresponding memristor needs to be updated and that the corresponding Δweight value is greater than or equals to weight change by one pulse. In this way, the PR method directly affects every weight update and minimizes the number of pulses and then avoids the impact of the cycle-to-cycle variation as much as possible.

V. Evaluation and Results

A. Platform

In order to evaluate the level scaling method and PR method, the multilayer perceptron platform (MLP platform) is used to emulate the learning classification scenario with Modified National Institute of Standards and Technology (MNIST) handwritten dataset [29]. We adopt the artificial neural network (ANN) hardware platform, NeuroSim+ [29], [48], to perform handwriting inference, as shown in Fig. 12(b) and (c), which is for the fully connected networks structure.

The crossbar array architecture with memristors had been proposed for on-chip implementation of weighted sum and weight update in the training process of learning algorithms [49]. This platform contains three layers with 400 neurons for an input layer, 100 neurons for a hidden layer, and ten neurons for an output layer, as shown in Fig. 12(c). The perceptron neural network is simulated and based on memristive crossbar arrays, which includes a special subset of the memristor to tune the conductance by voltage pulse stimulus. The desired weight update for each layer is calculated in software [50] and then applied to the crossbar by the system, as illustrated in Fig. 12(b) as the hardware implementation block diagram. For the level scaling method, each evaluation trains 125 epochs, and every epoch randomly selects 8000 images from 60 000 training images. For the PR method, each evaluation trains 200 epochs, and every epoch randomly selects 500 images from 60 000 training images. A different set of 10 000 images are included in the testing dataset. Note that the networks will continually learn the feature of an input data after the last epoch since this platform is an online learning

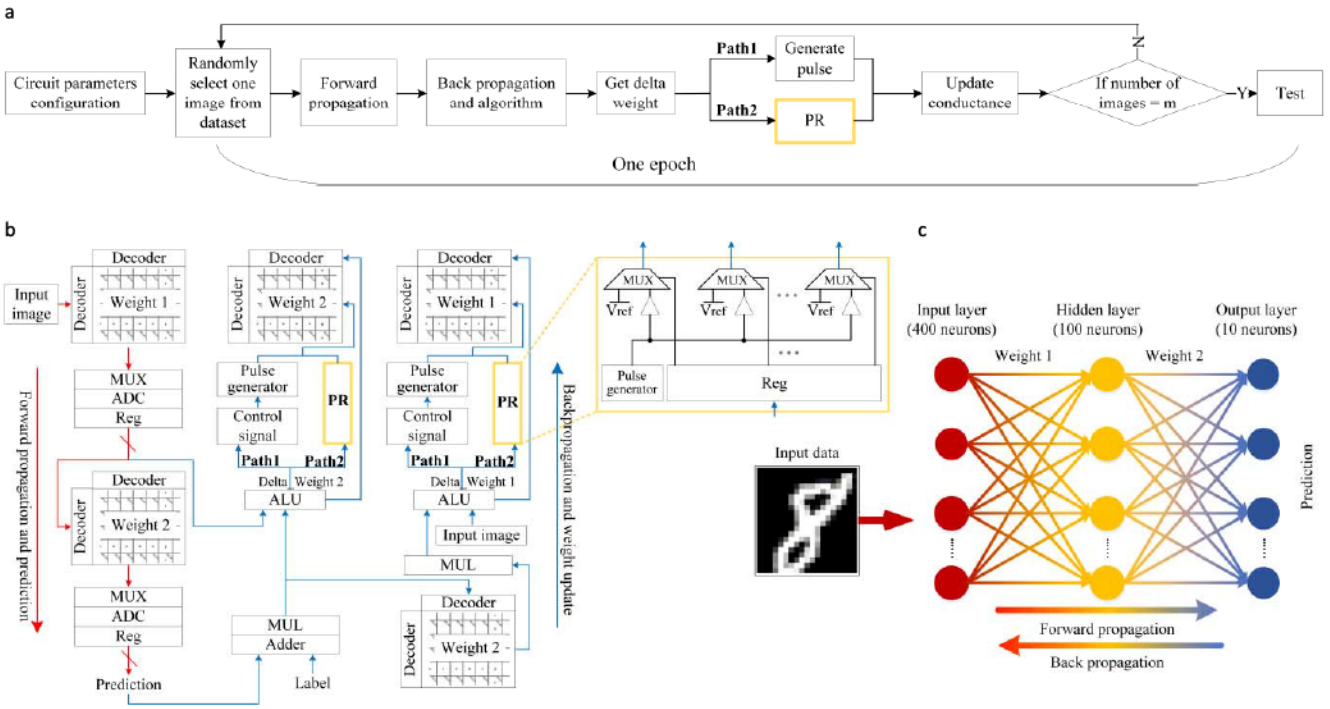

Fig. 12. (a) Workfl w of the MLP platform. Optimized number of the level value and measured cycle-to-cycle variation are set at the circuit parameters configu ation step. m equals 8000 for the path 1 and equals 500 for the path 2. (b) Hardware implementation of the MLP platform. (c) Three-layer fully connected structure of neural network. Input layer, hidden layer, and output layer are composed by different neurons. PR represents the pulse regulating method.

network [48]. In this platform, the parameters of memristors come from the measurement results of our fabricated devices. In summary, this MLP platform is a stand-alone functional platform that is able to evaluate the inference accuracy and device-level performance during the learning process.

B. Results of Level Scaling Method

To study the relationship between the number of the levels and cycle-to-cycle variation, the different numbers of the level for the LTP and LTD are set. The ideal circumstances ($\alpha = 0$) with the number of the levels from 10 to 200 and step 10 are set with five algorithms, as shown in Fig. 13(a)–(e). When the cycle-to-cycle variation is not involved, with the increasing number of the level, the accuracy goes up to the high area (bright area) from the low area (dark area), where the highest accuracy appears at the number of the levels = 200 at LTP and LTD (upper right corner). It can be concluded that increasing the number of the levels does increase the inference accuracy. In bright areas of the figures, the inference accuracies are around 90% in the lower left corner and higher than 93% in the upper right corner.

As a comparison, realistic cases that are with the cycle-to-cycle variation ($\alpha = 0.03577$) are set to obtain optimization with five algorithms, as shown in Fig. 13(f)–(j). When the cycle-to-cycle variation is involved, the values of accuracies are lower than that without cycle-to-cycle variation. Moreover, the bright areas where accuracies are higher than 88% are smaller than the ideal case. Note that, in the top right corner, the accuracies do not go up to the highest with the increasing number of the levels. Even though the boundaries for separating bright and dark areas with different algorithms are different, the boundaries demonstrate regions and locations of the highest inference accuracies. Those number of the levels (LTP/LTD) are 50/40 for the stochastic gradient descent algorithm (SGD), 60/50 for the Momentum, 60/50 for the AdaGrad, 50/50 for the RMSProp, and 50/40 for the Adam. Therefore, the best performance of the given memristors-based edge AI system under the related cycle-to-cycle variation does not occur at the number of the levels = 200 for LTP and LTD, as shown in Fig. 13(f)–(j). Level scaling method optimizes the number of the levels, so that the system achieves higher inference accuracy by mitigating the cycle-to-cycle variation.

C. Results of PR Method

During the training process of ANN, the weight change that is calculated by the algorithm is converted to positive/negative pulses that is n in (5) to update the conductance of the memristor. According to parameter, n, in (5), drastically change in conductance by positive or negative pulses for LTP or LTD process causes more cycle-to-cycle variations in corresponding memristors. Thus, the PR method is to truncate the number of updating pulse to the one ($n = 1$) at each LTP or LTD process, as shown in path 2 of Fig. 12(b) for the hardware implementation and in path 2 of Fig. 12(a) for the processing flowchart. The conventional system originally has writing pulses whose widths are appropriate to tune the conductance of a memristor. Simultaneously, each writing pulse is identical during different update processes. The proposed PR method, instead of updating the weight by the number of the pulses that are directly converted from the weight change in each iteration,

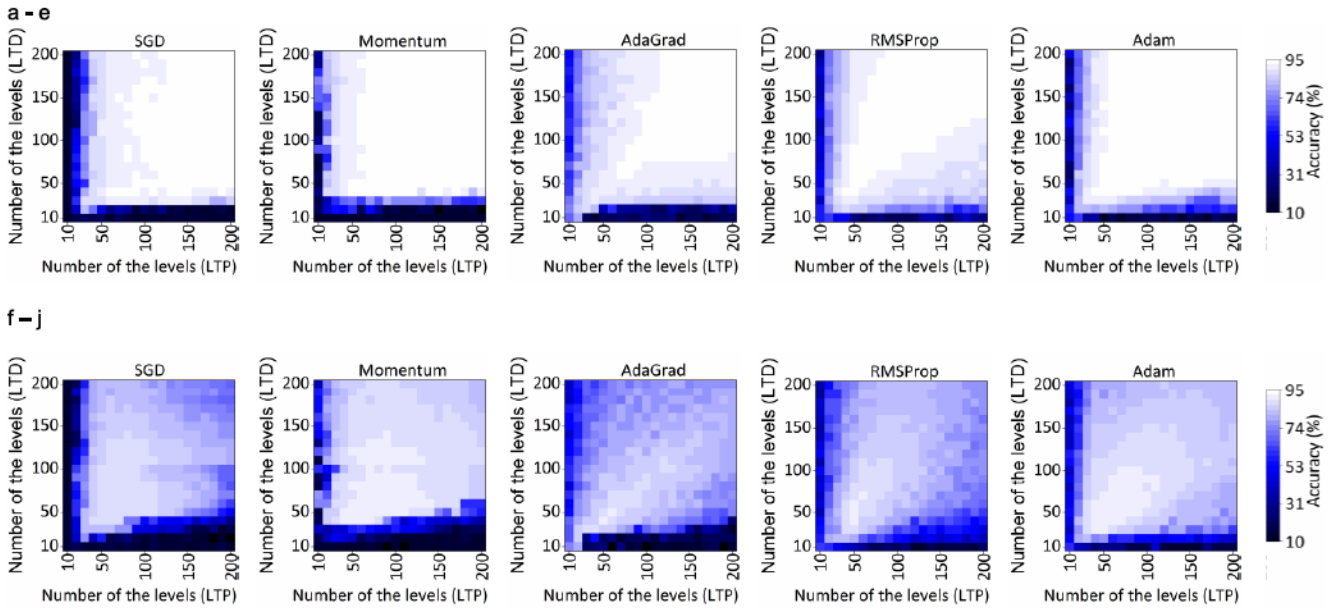

Fig. 13. (a)–(e) Inference accuracy without cycle-to-cycle variation ($\alpha = 0$) with different LTP and LTD numbers of the levels (from 10 to 200, step is 10) in fi e algorithms. (f)–(j) Inference accuracy with different LTP and LTD numbers of the level values (from 10 to 200, step is 10) in fi e algorithms under measured cycle-to-cycle variation ($\alpha = 0.03577$).

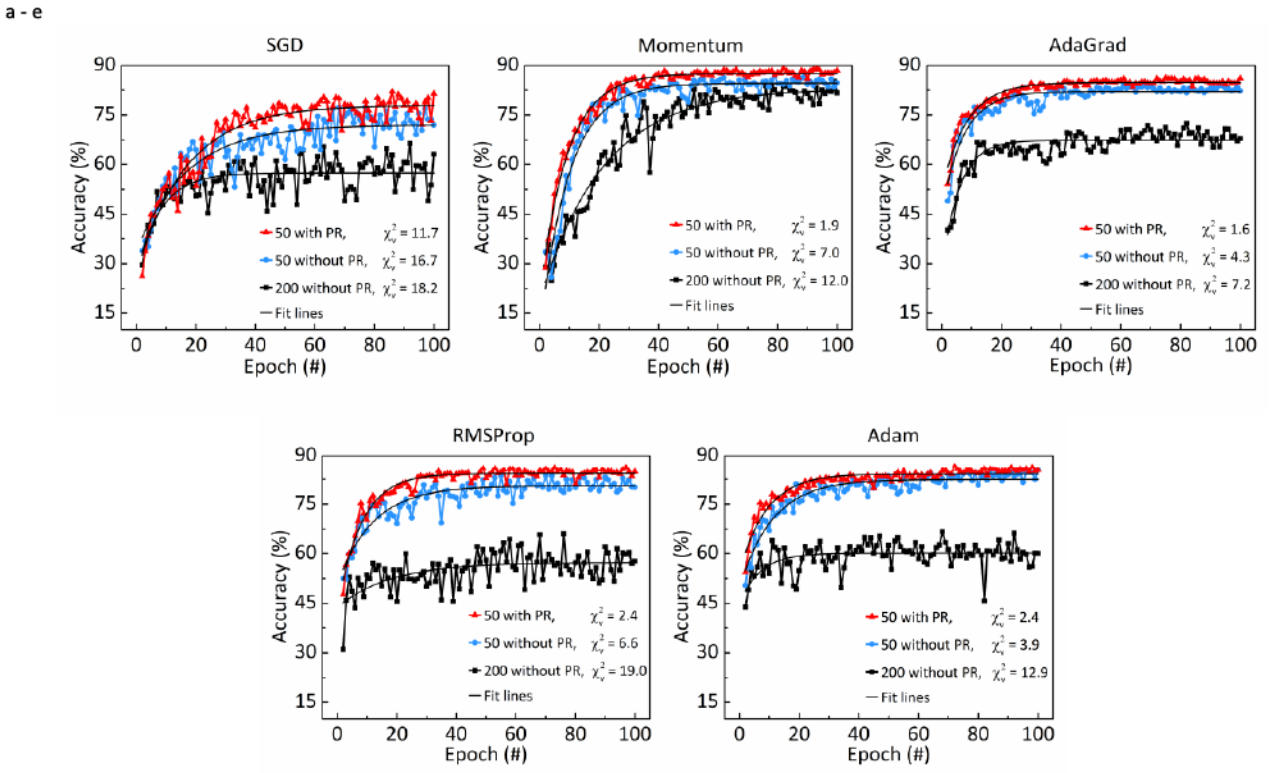

Fig. 14. PR method for mitigating cycle-to-cycle variation. (a)–(e) Inference accuracy with/without the PR method in 5 algorithms. 50 and 200 are the number of the levels in different evaluations. χ_v^2 is thereduced Chi-Sqr values with analyzing data. PR represents the pulse regulating method. Each curve includes 100 epochs and each epoch includes 500 images.

only applies one pulse and keeps the original width of writing pulses, as shown in golden block (path 2) of Fig. 12(b). To verify the effectiveness of the proposed PR method, we adopt the ANN hardware platform, NeuroSim+ [29], [48], to perform the PR method. The PR is suitable for all five algorithms that the accuracies are higher than that without the PR method, as shown in Fig. 14(a)–(e). Note that, for evaluating the effect of the PR method, 200 epochs with 500 images for each

TABLE I
ENERGY AND LATENCY EVALUATION WITH AND WITHOUT THE PULSE REGULATING METHOD

Algorithm	Energy (mJ)			Latency (Min)		
	With PR	Without PR	Saved (%)	With PR	Without PR	Reduced (%)
SGD	0.313	0.361	**13.310**	15.059	17.728	**15.057**
Momentum	0.501	0.575	**12.888**	23.912	32.340	**26.062**
AdaGrad	0.642	0.671	**4.233**	31.333	37.290	**15.974**
RMSProp	1.451	1.729	**16.104**	75.962	105.288	**27.854**
Adam	1.072	1.197	**10.394**	54.422	68.703	**20.787**

PR represents pulse regulating method. Each evaluation includes 100 epochs and each epoch includes 500 images with 50 levels of conductance. 32nm technology complementary metal–oxide–semiconductor (CMOS) part is evaluated in this work. For the process of weight update, the voltage of the LTP and LTD is 3.2 V, 2.8 V, respectively. The pulse width of weight update is 600 μs for both LTP and LTD.

epoch are set. The negative impact of the PR method is reducing the learning speed, which only exists at the several beginning learning epochs and is reflected by the red curves below the blue curves in Fig. 14(a)–(e). Although the learning speed is reduced by the PR method at the several beginning learning epochs, all inference accuracies of five algorithms have significant improvement with the PR method after the whole training process. In addition, the PR method effectively produces a smoother convergence of the training process, which reduces the excessive fluctuation of the inference accuracy. The regressions are carried out by the exponential model to fit the experimental data without and with the PR method. The reduced Chi-Sqr values that are represented as χ_v^2 with the PR method are smaller (closer to 1) than that without the PR method, as shown in Fig. 14(a)–(e), which demonstrates that the fluctuation of the inference accuracy is reduced by the PR method.

Furthermore, because the updating pulses are regulated to one in each iteration, the number of updating pulses has been significantly saved for 100 epochs, taking RMSProp as an example, which effectively saves the energy consumption up to 16.104% and reduces the latency up to 27.854%, as shown in Table I. Every iteration has the designated reading latency since the process of a vector-matrix multiplication is executed using a parallel reading strategy. However, the system updates its weight row by row, which indicates that a parallel writing strategy cannot be implemented for all rows at the same time; otherwise, the system will have unacceptable area overhead [51]. Each row's writing latency is determined by the maximum number of writing pulses as a critical path. Thereby, the main latency for crossbar arrays is writing latency that strongly depends on the maximum update pulses of each row. With the PR method, the maximum number of the writing pulses decreases to one, which reduces the total latency of the system. In the system without the PR method, each row needs registers and counter to record and control the updating process since the time of updating process in the different training iterations is probably different [48]–[50]. In the system with the PR method, those two components (registers and counter) are not needed because the selected row only uses one pulse to update the conductance of the memristor. Instead, one multiplexer is added to the system. Therefore, the PR method optimizes the inference accuracy, improves energy efficiency, and reduces system latency and area.

VI. DISCUSSION

For a given memristive crossbar array, the distribution of cycle-to-cycle variation can be modeled by (2). At the same time, according to Fig. 7, a lower number of the levels mean larger conductance change between two consecutive pulses, and the system uses wider and fewer updating pulses for the same weight change that is calculated through any machine learning algorithm. According to experiment results and (5), the wider pulse does not increase the cycle-to-cycle variation and fewer updating pulses correspond to a smaller n, which reduces the cycle-to-cycle variation. Therefore, level scaling is an effective method to mitigate the impact of cycle-to-cycle variation. Note that an extremely low number of levels will influence the accuracy of the conductance, which means some desired values of conductance cannot be achieved, as shown in Fig. 7, so reducing the precision of the system. This influence is also reflected by the low inference accuracy, as shown in the low (dark) number of the levels area of Fig. 13. Thereby, for multilevel memristive crossbar arrays that are used in machine learning systems, the highest inference accuracy of the system occurs when the memristor uses an optimized number of levels rather than the highest number of levels.

In further comparison of Fig. 13(a)–(e) and Fig. 13(f)–(j), some accuracies with cycle-to-cycle variations and with certain LTP/LTD levels are higher than that without cycle-to-cycle variation (ideal circumstance), as shown in Fig. 15(a)–(e). The black squares represent the negative values, which mean that the inference accuracy with cycle-to-cycle variation is higher than that without the cycle-to-cycle variation. This is because only integer number of pulses are generated in circuit. As for the mechanism of the updating process, the amount of conductance that is increased or decreased will be calculated by the algorithm, and then, the accurate number of pulses is gotten accordingly. However, in the circuit level (hardware), only the integer number of pulses are available to update the conductance. Hence, the truncation function for an integer number of pulses is employed for hardware implementation. The updated weight gets the deviation by an integer number of pulses, but when the cycle-to-cycle variation is involved in every weight update, in some cases, they make the updated

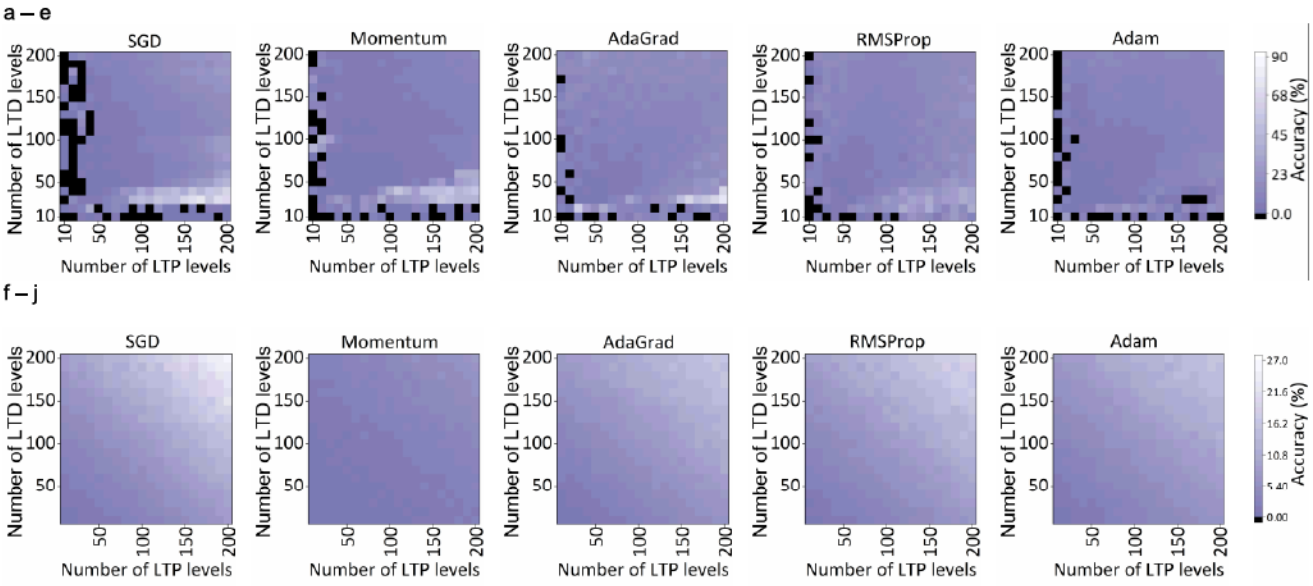

Fig. 15. (a)–(e) Difference of accuracy between ideal case ($\alpha = 0$) and that with cycle-to-cycle variation ($\alpha = 0.03577$) with the truncation function. (Truncation function converts Δweight to the integer number of pulses.) The black squares represent the negative values. (f)–(j) Difference of accuracy between ideal case ($\alpha = 0$) and that with cycle-to-cycle variation ($\alpha = 0.03577$) without the truncation function.

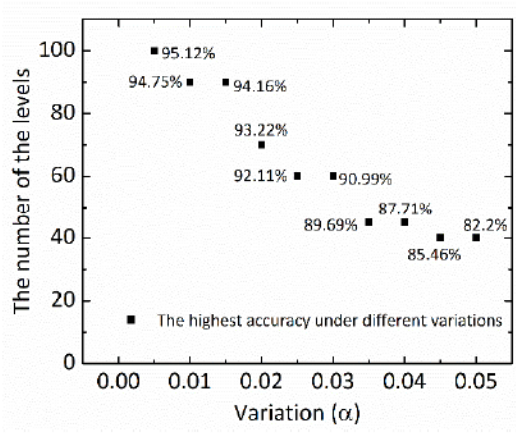

Fig. 16. Number of the levels with the highest accuracy under different cycle-to-cycle variations.

weight to achieve closer to the accurate weight that algorithm requires and then the system gets even higher inference accuracy. In order to highlight the influence of this factor, the truncation function is disabled in the control group simulation, and we get the difference of accuracy between ideal case and that with cycle-to-cycle variation, as shown in Fig. 15(f)–(j). Note that, all differences of inference accuracies are positive, which means the accuracy in ideal case is higher than that with cycle-to-cycle variations. Therefore, the cycle-to-cycle variations, sometimes, enable an integer number of pulses to achieve more accurate weight that algorithm requires, and then, the system results in even higher inference accuracy than the system with the truncation function without cycle-to-cycle variation. That is why some accuracies with cycle-to-cycle variation and with certain LTP/LTD levels in Fig. 13(a)–(e) are higher than that without cycle-to-cycle variation in Fig. 13(f)–(j).

With our device, the value of α is 0.03577. Thus, the $\sim$50 levels are the best choice because the value with each level step is $((1 - (-1)/50) = 0.04$, which is close to 0.03577. As shown in Fig. 16, the number of the levels with the highest accuracy decreases through the increasing of the variations. In fact, the reason for this correspondence is that an excessively large number of levels will cause the value of the variation to exceed the conductance value of a single level. Simultaneously, a too small level number will cause the weight value to lose too much precision and reduce the final inference accuracy.

According to the mechanism of the cycle-to-cycle variation, the PR method efficiently reduces the cycle-to-cycle variation by compressing the number of update pulse to one with n parameter in (5). For every updating, the cycle-to-cycle variation is limited with one pulse's impact, which minimizes the cycle-to-cycle variation for the system. Note that the inference accuracies have significant improvement with the PR method, as shown in Fig. 14(a)–(e). The reasons are two aspects: 1) the PR method minimizes the cycle-to-cycle variation and 2) each update step uses at most one pulse to tune conductance. One pulse to tune conductance means that smaller steps are achieved in the direction of convergence, while a big step will make the learning jump over minimum point of weight [52]. What's more, energy consumption and system latency are correspondingly reduced when the PR method is adopted in the system by compressing the number of update pulse to one.

VII. Conclusion

In order to mitigate the impact of cycle-to-cycle variations in Memristor-based edge AI system, we fabricate TiO_2/TiO_{2-x} memristors and derive a closed model. We propose the level scaling and the PR methods that are simple, feasible, and universal methods to effectively mitigate the impact of

cycle-to-cycle variations. We prove that because of cycle-to-cycle variations, the inference accuracy in the maximum number of the levels is not optimal for the real device. For different materials-based multilevel memristors, the level scaling method can be used to optimize the memristor-based edge AI system by selecting appropriately the number of the levels. Similarly, the PR method mitigates the impact of cycle-to-cycle variation by compressing the number of updating pulses to one as well as improves energy efficiency up to 16.104% and reduces system latency up to 27.854%. Furthermore, both methods can be implemented at the edge computing, which paves the way for the adoption of memristors for more efficient applications for the era of the IoT.

ACKNOWLEDGMENT

Portions of this work were conducted in the Core Research Facility at North Dakota State University, which receives partial support from the NSF through the NNCI program (Award Number ECCS-1542202). The authors would like to thank Yong Wang, Greg Strommen, and Fred Haring, who assisted with fabrication processes and measurements/testing.

REFERENCES

[1] R. H. Weber and R. Weber, *Internet of Thing*, vol. 12. Heidelberg, Germany: Springer, 2010.

[2] A. Fayyazi, M. Ansari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "An ultra low-power memristive neuromorphic circuit for Internet of Things smart sensors," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1011–1022, Jan. 2018, doi: 10.1109/JIOT.2018.2799948.

[3] Y. Deng, "Deep learning on mobile devices: A review," *Proc. SPIE*, vol. 10993, May 2019, Art. no. 109930A, doi: 10.1117/12.2518469.

[4] N. D. Lane *et al.*, "DeepX: A software accelerator for low-power deep learning inference on mobile devices," in *Proc. 15th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, Apr. 2016, pp. 1–12, doi: 10.1109/IPSN.2016.7460664.

[5] J. Wang, J. Zhang, W. Bao, X. Zhu, B. Cao, and P. S. Yu, "Not just privacy: Improving performance of private deep learning in mobile cloud," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2407–2416, doi: 10.1145/3219819.3220106.

[6] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generat. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013, doi: 10.1016/j.future.2013.01.010.

[7] R. S. Williams, "What's next? [the end of Moore's law]," *Comput. Sci. Eng.*, vol. 19, no. 2, pp. 7–13, Mar. 2017, doi: 10.1109/MCSE.2017.31.

[8] M. M. Waldrop, "The chips are down for Moore's law," *Nature*, vol. 530, no. 7589, pp. 144–147, Feb. 2016, doi: 10.1038/530144a.

[9] L. O. Chua, "Memristor-the missing circuit element," *IEEE Trans. Circuit Theory*, vol. CT-18, no. 5, pp. 507–519, Sep. 1971, doi: 10.1109/TCT.1971.1083337.

[10] D. B. Strukov, G. S. Snider, D. R. Stewart, and S. R. Williams, "The missing memristor found," *Nature*, vol. 453, pp. 80–83, May 2008, doi: 10.1038/nature06932.

[11] Z. Wang *et al.*, "Fully memristive neural networks for pattern classification with unsupervised learning," *Nature Electron.*, vol. 1, pp. 137–145, Feb. 2018, doi: 10.1038/s41928-018-0023-2.

[12] M. A. Zidan, J. P. Strachan, and W. D. Lu, "The future of electronics based on memristive systems," *Nature Electron.*, vol. 1, no. 1, pp. 22–29, Jan. 2018, doi: 10.1038/s41928-017-0006-8.

[13] L. Xia *et al.*, "Switched by input: Power efficient structure for RRAM-based convolutional neural network," in *Proc. 53rd ACM/EDAC/IEEE Design Automat. Conf. (DAC)*, Jun. 2016, pp. 1–6, doi: 10.1145/2897937.2898101.

[14] 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, doi: 10.1109/JETCAS.2019.2910749.

[15] J. Fu, Z. Liao, and J. Wang, "Memristor-based neuromorphic hardware improvement for privacy-preserving ANN," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 12, pp. 2745–2754, Dec. 2019, doi: 10.1109/TVLSI.2019.2923722.

[16] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, "TEAM: Threshold adaptive memristor model," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 1, pp. 211–221, Nov. 2012, doi: 10.1109/TCSI.2012.2215714.

[17] M. D. Pickett and R. S. Williams, "Sub-100 fJ and sub-nanosecond thermally driven threshold switching in niobium oxide crosspoint nanodevices," *Nanotechnology*, vol. 23, no. 21, p. 215202, May 2012, doi: 10.1088/0957-4484/23/21/215202.

[18] A. C. Torrezan, J. P. Strachan, G. Medeiros-Ribeiro, and R. S. Williams, "Sub-nanosecond switching of a tantalum oxide memristor," *Nanotechnology*, vol. 22, no. 48, Nov. 2011, Art. no. 485203, doi: 10.1088/0957-4484/22/48/485203.

[19] M.-J. Lee *et al.*, "A fast, high-endurance and scalable non-volatile memory device made from asymmetric Ta_2O_{5-x}/TaO_{2-x} bilayer structures," *Nature Mater.*, vol. 10, no. 8, pp. 625–630, Jul. 2011, doi: 10.1038/nmat3070.

[20] S. Pi, P. Lin, and Q. Xia, "Cross point arrays of 8 nm×8 nm memristive devices fabricated with nanoimprint lithography," *J. Vac. Sci. Technol. B, Nanotechnol. Microelectron., Mater., Process., Meas., Phenomena*, vol. 31, no. 6, Nov. 2013, Art. no. 06FA02, doi: 10.1116/1.4827021.

[21] Q. Xia *et al.*, "Memristor-CMOS hybrid integrated circuits for reconfigurable logic," *Nano Lett.*, vol. 9, no. 10, pp. 3640–3645, Sep. 2009, doi: 10.1021/nl901874j.

[22] X. Guan, S. Yu, and H.-S.-P. Wong, "On the switching parameter variation of metal-oxide RRAM—Part I: Physical modeling and simulation methodology," *IEEE Trans. Electron Devices*, vol. 59, no. 4, pp. 1172–1182, Apr. 2012, doi: 10.1109/TED.2012.2184545.

[23] C. Baeumer *et al.*, "Subfilamentary networks cause cycle-to-cycle variability in memristive devices," *ACS Nano*, vol. 11, no. 7, pp. 6921–6929, Jul. 2017, doi: 10.1021/acsnano.7b02113.

[24] R. Zhu, S. Chang, H. Wang, Q. Huang, J. He, and F. Yi, "A versatile and accurate compact model of memristor with equivalent resistor topology," *IEEE Electron Device Lett.*, vol. 38, no. 10, pp. 1367–1370, Aug. 2017, doi: 10.1109/LED.2017.2736006.

[25] S. Choi, P. Sheridan, and W. D. Lu, "Data clustering using memristor networks," *Sci. Rep.*, vol. 5, no. 1, pp. 10492–10502, May 2015, doi: 10.1038/srep10492.

[26] 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, doi: 10.1109/TED.2019.2906249.

[27] M. A. Lastras-Montaño and K.-T. Cheng, "Resistive random-access memory based on ratioed memristors," *Nature Electron.*, vol. 1, no. 8, pp. 466–472, Aug. 2018, doi: 10.1038/s41928-018-0115-z.

[28] S. Kim, M. Lim, Y. Kim, H.-D. Kim, and S.-J. Choi, "Impact of synaptic device variations on pattern recognition accuracy in a hardware neural network," *Sci. Rep.*, vol. 8, no. 1, pp. 2638–2645, Feb. 2018, doi: 10.1038/s41598-018-21057-x.

[29] 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)*, Jan. 2016, pp. 194–199, doi: 10.1109/ICCAD.2015.7372570.

[30] T. Tang *et al.*, "Spiking neural network with RRAM: Can we use it for real-world application?" in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2015, pp. 860–865.

[31] 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 Automat. Conf. (DAC)* Jun. 2016, pp. 19–25, doi: 10.1145/2897937.2898010.

[32] J. Rajendran, H. Maenm, R. Karri, and G. S. Rose, "An approach to tolerate process related variations in memristor-based applications," in *Proc. 24th Int. Conf. VLSI Design*, Jan. 2011, pp. 18–23, doi: 10.1109/VLSID.2011.49.

[33] L. Chen *et al.*, "Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2017, pp. 19–24, doi: 10.23919/DATE.2017.7926952.

[34] K. M. Kim *et al.*, "Voltage divider effect for the improvement of variability and endurance of TaOx memristor," *Sci. Rep.*, vol. 6, no. 1, pp. 20085–20091, Feb. 2016, doi: 10.1038/srep20085.

[35] B. Liu, H. Li, Y. Chen, X. Li, Q. Wu, and T. Huang, "Vortex: Variation-aware training for memristor X-bar," in *Proc. 52nd Annu. Design Automat. Conf.*, Jun. 2015, pp. 1–6, doi: 10.1145/2744769.2744930.

[36] J. Rajendran, R. Karri, and G. S. Rose, "Improving tolerance to variations in memristor-based applications using parallel memristors," *IEEE Trans. Comput.*, vol. 64, no. 3, pp. 733–746, Mar. 2015, doi: 10.1109/TC.2014.2308189.

[37] J. Rajendran, R. Karri, and G. S. Rose, "Parallel memristors: Improving variation tolerance in memristive digital circuits," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2011, pp. 2241–2244, doi: 10.1109/ISCAS.2011.5938047.

[38] Y. Li *et al.*, "Improvement of resistive switching characteristics in ZrO_2 film by embedding a thin TiO_X layer," *Nanotechnology*, vol. 22, no. 25, pp. 254028–254034, May 2011, doi: 10.1088/0957-4484/22/25/254028.

[39] A. M. Rana *et al.*, "Endurance and cycle-to-cycle uniformity improvement in tri-layered CeO_2/Ti/CeO_2 resistive switching devices by changing top electrode material," *Sci. Rep.*, vol. 7, no. 1, pp. 39539–39554, Jan. 2017, doi: 10.1038/srep39539.

[40] Y. Yang, P. Gao, S. Gaba, T. Chang, X. Pan, and W. Lu, "Observation of conducting filament growth in nanoscale resistive memories," *Nature Commun.*, vol. 3, no. 1, pp. 732–740, Mar. 2012, doi: 10.1038/ncomms1737.

[41] T. Chang, S.-H. Jo, K.-H. Kim, P. Sheridan, S. Gaba, and W. Lu, "Synaptic behaviors and modeling of a metal oxide memristive device," *Appl. Phys. A, Solids Surf.*, vol. 102, no. 4, pp. 857–863, Feb. 2011, doi: 10.1007/s00339-011-6296-1.

[42] A. Williamson *et al.*, "Synaptic behavior and STDP of asymmetric nanoscale memristors in biohybrid systems," *Nanosc.*, vol. 5, no. 16, pp. 7297–7303, Jun. 2013, doi: 10.1039/C3NR01834B.

[43] M. Uddin, M. S. Hasan, and G. S. Rose, "On the theoretical analysis of memristor based true random number generator," in *Proc. Great Lakes Symp. VLSI*, May 2019, pp. 21–26, doi: 10.1145/3299874.3317981.

[44] T. Tuma, A. Pantazi, M. L. Gallo, A. Sebastian, and E. Eleftheriou, "Stochastic phase-change neurons," *Nature Nanotechnol.*, vol. 11, pp. 693–699, May 2016, doi: 10.1038/nnano.2016.70.

[45] H. Jiang *et al.*, "A novel true random number generator based on a stochastic diffusive memristor," *Nature Commun.*, vol. 8, no. 1, pp. 882–891, Oct. 2017, doi: 10.1038/s41467-017-00869-x.

[46] C. Ye *et al.*, "Physical mechanism and performance factors of metal oxide based resistive switching memory: A review," *J. Mater. Sci. Technol.*, vol. 32, no. 1, pp. 1–11, Jan. 2016, doi: 10.1016/j.jmst.2015.10.018.

[47] D. S. Lemons and P. Langevin, *An Introduction to Stochastic Processes in Physics*. Baltimore, MD, USA: JHU Press, Jan. 2003, pp. 189–194, doi: 10.1119/1.1526134.

[48] P.-Y. Chen, X. Peng, and S. Yu, "NeuroSim+: An integrated device-to-algorithm framework for benchmarking synaptic devices and array architectures," in *IEDM Tech. Dig.*, vol. 4, Dec. 2017, pp. 6.1.1–6.1.4, doi: 10.1109/IEDM.2017.8268337.

[49] 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 Design Integr. Circuits Syst.*, vol. 37, no. 12, pp. 3067–3080, Dec. 2018.

[50] M. A. Ponti, L. S. F. Ribeiro, T. S. Nazare, T. Bui, and J. Collomosse, "Everything you wanted to know about deep learning for computer vision but were afraid to ask," in *Proc. 30th SIBGRAPI Conf. Graph., Patterns Images Tuts. (SIBGRAPI-T)*, Jan. 2018, pp. 17–41, doi: 10.1109/SIBGRAPI-T.2017.12.

[51] L. Gao *et al.*, "Fully parallel write/read in resistive synaptic array for accelerating on-chip learning," *Nanotechnol.*, vol. 26, no. 45, pp. 455204–455214, Oct. 2015, doi: 10.1088/0957-4484/26/45/455204.

[52] N. Buduma and N. Locascio, *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms*. Newton, MA, USA: O'Reilly Media, 2017.