Rescuing Memristor-based Computing with Non-linear Resistance Levels

Jilan Lin[1], Lixue Xia[1], Zhenhua Zhu[1], Hanbo Sun[1], Yi Cai[1], Hui Gao[1], Ming Cheng[1],
Xiaoming Chen[2], Yu Wang[1] and Huazhong Yang[1]
[1]Dept. of E.E., Tsinghua National Laboratory for Information Science and Technology (TNList),
Tsinghua University, Beijing, China
[2]State Key Laboratory of Computer Architecture, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, China

***Abstract*—Emerging memristor devices like metal oxide resistive switching random access memory (RRAM) and memristor crossbar have shown great potential in computing matrix-vector multiplication. However, due to the nonlinear distribution of resistance levels in memristor devices, the state-of-the-art multi-bit cell cannot accomplish the multi-bit computing task accurately. In this paper, we propose fault-tolerant schemes to rescue memristor-based computation with nonlinear resistance levels. We classify the resistance level distributions in memristor devices into three types, and the corresponding models are proposed to analyze the computation characteristics. We propose two theoretical conditions to determine if a memristor device can support multi-bit matrix computation. For the deviated linear model, the least squares method is used to reduce the computing error. When the resistance distribution obeys the proposed power model, a logarithmic operation circuit is used to decode the multiplication results and then accomplish the computing accurately. For the exponential model, since the device cannot complete typical matrix-vector multiplication from hardware level, we propose online and offline quantization methods to make the neural computing algorithms friendly to memristor device. Simulation results show that the root-mean-square error improves around 4% with the linear model and more than 99% with the power model. After quantization, the accuracy of ResNet-18 using memristor with exponential conductance levels can be improved to the same accuracy with ideal linear devices.**

I. Introduction

As the most important operation for many computationally intensive algorithms, matrix-vector multiplication (MVM) has always been the bottleneck for accelerating the computing speed in the area of information processing. Recently, more and more researchers get to focus on completing the MVM on emerging non-volatile memories, such as metal-oxide resistive random access memory (RRAM), phase-changing memory (PCM). With the resistance switching property and crossbar structure, memristor crossbar array can efficiently reduce the computation complexity from O(n^2) to O(1). Furthermore, since multi-level cell (MLC) memristor has shown the ability to store multiple bits in the single cell and significantly improves the crossbar array density, memristor-based multi-bit computing architectures have been proposed to further improve the computing efficiency [1], [2].

Researchers have shown that there are resistance gaps between different memristor states, which means we cannot tune the cell into arbitrary resistance/conductance. Previous memristor-based multi-bit computing architecture designs based on the assumption that the multiple resistance levels in the memristor device are linearly distributed. However, testing results of the actual fabricated multi-level devices have shown that the distribution of resistance levels is not linear. Typically, there are three types of distributions. The resistance levels in [3], [4] present a near-linearly increasing distribution, but the resistance levels are deviated from a linear distribution, which is referred as "linear distribution with deviation" in this paper. Moreover, the resistance levels in devices may obey a power distribution [5] or even an exponential distribution [6], [7].

This work was supported by National Key R&D Program of China 2017YFA0207600, National Natural Science Foundation of China (No.61373026,61622403,61621091), and joint fund of Equipment pre-Research and Ministry of Education (No. 6141A02022608).

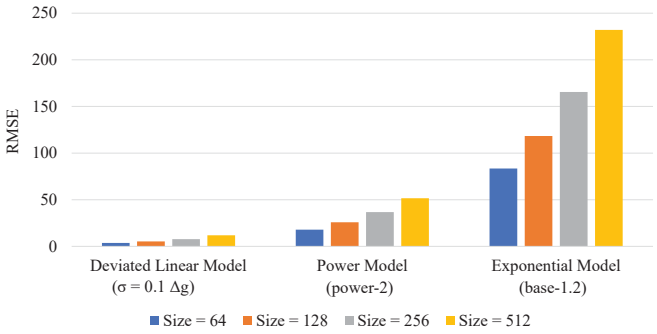

Fig. 1. Computing Root-Mean-Square Error (RMSE). The y-axis shows the RMSE for each case; The x-axis presents different cases.

To accomplish computing, we first need to map the matriX values to the resistances of memristor cells. However, traditional mapping methods using the ideal linear devices are not available for the practical devices with different distributions. For example, for the memristor devices whose resistance levels obey an exponential distribution, the root-mean-square error of the computing output for a 512×512 memristor array can be larger than 200, which is shown in Fig. 1. If we directly map ResNet-18 onto this structure, the accuracy on Cifar-10 dataset is less than 10%. As a result, the existing fabricated memristor-based computing chips are still based on the computation in 1-bit cells [8].

In this paper, we propose fault-tolerate schemes to make the memristor devices with non-linear distributions capable of multi-bit computation. The main contributions of this paper include:

1) **Analysis on the distribution of resistance states for state-of-the-art multi-level memristor devices**: Three models are proposed to fit the actual distribution of resistance levels in different type of memristor devices: the linear model, the power model, and the exponential model.
2) **Analysis on the restrictions for the resistance level distributions when mapping a multi-bit value to a memristor cell**: Two conditions are proposed to determine whether a memristor crossbar is capable of accurate computing. We find that only linear distribution model can directly complete the computation accurately.
3) **Rescuing the Computing**: We propose three methods to

improve the computing performance. Weighted least square method is used to reduce the computing error for the linear device with deviation. Non-uniform quantification circuits are used to sense out the non-linear computing results. We prove that exponential model cannot complete computation with more than 2-bit, but the online and offline quantization methods from algorithm layer are proposed to tolerate the non-linearity by the neural algorithms. Fig. 2 shows a brief summary of our proposed schemes.

4) Simulation results show that the voltage modification method can reduce computation error with 4%, and the proposed quantification circuits can reduce the computation error for two orders of magnitudes when using the device with power model. For exponential model, the weight quantization methods can improve the recognition accuracy back to the same as ideal linear situation.

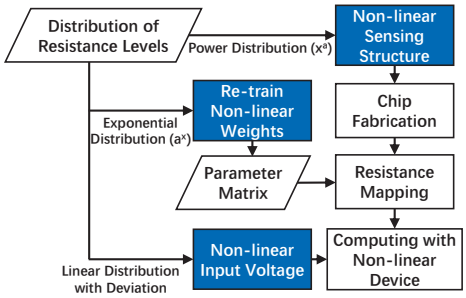

Fig. 2. The flow chart summarizes our work in this paper. We divide the non-linear distribution into three cases: deviated linear model, power model, and exponential model. And we propose voltage design method, re-trained weights in neural network and circuits design to tolerate the fault, respectively.

II. Background

RRAM and PCM are two kinds of emerging non-volatile memories, which are also known as memristor. Multiple memristor cells can construct the crossbar structure, which can be used to perform analog MVM efficiently, with reducing the computation complexity from $O(n^2)$ to $O(1)$ [9]. Previous works have proposed several memristor-based Computing Systems, for instance, memristor-based dot-product operator [10], memristor-based fully-connected neural networks [11], and memristor-based convolutional neural networks [12]. Since the conductances of memristor cells can only be positive, two memristor crossbars are needed to represent an application matrix with both positive and negative parameters [13].

Fig. 3 (a) shows the structure of memristor crossbar array and sensing method for getting results. When using it to complete computating, the relationship between input voltage vector (V^i) and the output voltage vector (V^o) can be expressed as follows:

$$\begin{bmatrix} V_1^o \\ \vdots \\ V_M^o \end{bmatrix} = \begin{bmatrix} c_{1,1} & \cdots & c_{1,N} \\ \vdots & \ddots & \vdots \\ c_{M,1} & \cdots & c_{M,N} \end{bmatrix} \begin{bmatrix} V_1^i \\ \vdots \\ V_N^i \end{bmatrix} \tag{1}$$

And the common used sensing mode is trans-impedance amplifiers (TIAs), as shown in Fig. 3 (a). The mapping relations between matrix parameters and memristor conductances can be represented as follows:

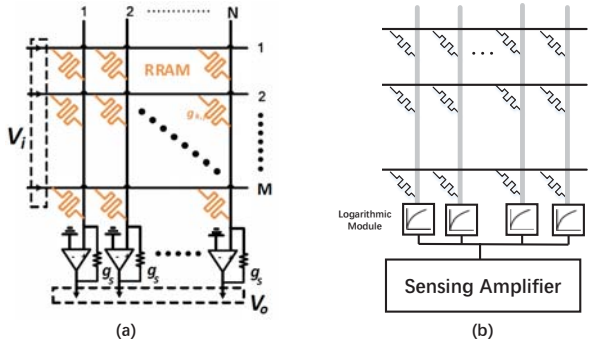

Fig. 3. (a) The memristor crossbar array and sensing mode with amplifier. (b) Peripheral circuits design for power model, where we add the logarithmic operation circuits before sensing.

$$c_{i,j} = -g_{i,j}/g_s \tag{2}$$

where $c_{i,j}$ is the parameter to be mapped to memristor cell, and $g_{i,j}$ is the conductance of memristor cell in the corresponding position of $c_{i,j}$. This sensing mode provides the advantage that conductance of g_s is fixed and $c_{i,j}$ only depends on one conductance $g_{i,j}$, so we can implement the matrix to crossbar array by one-to-one mapping. Therefore, the output of the crossbar array can be expressed as:

$$V_j^o = R_s \sum_{k=1}^{M} V_k^i g_{k,j} \tag{3}$$

III. Multi-level Cell and Fitting Functions

Multi-level cell can help gain much more density and reduce the cost of memristor as we can set more than one bit of information in a single cell. Since the parameter $c_{i,j}$ in matrix needs to be mapped into the conductance $g_{i,j}$ through Equ. (2), studying the distribution of conductances in multi-level cell is important for memristor based computing system. However, researchers still cannot give out a specific model to fit the actual distribution of multi-level cell, since there may be huge differences between memristor cells under a variety of material choices and fabrication methods.

Here we conclude the distribution from some of the works focus on the characteristics of multi-level cell: the conductances are linearly distributed but with deviation in [3], [4], while the conductances in [6], [7] exponentially increase with the different states. And the distribution in [5] is between linear and exponential, which can be fitted by power function. Therefore, we propose three models to fit some typical conductance distributions of multi-level memristor cell: deviated linear model, power model and exponential model. Moreover, we can formulate the models as follows:

$$\begin{cases} g_k = Ck + \delta_k \\ g_k = Ck^a \\ g_k = Ca^k \end{cases} \tag{4}$$

Here a, C are the model parameters, δ_k describes the deviation from linearity and g_k is the k-th level of conductance. We use the three models to fit the conductance distribution in [3], [5] and [6] and examine the model performance. Fig. 4 shows the fitting results. The realistic conductances are taken by median value in distributions of the papers mentioned and the conductance variation is beyond our consideration since the state-of-art technology can tune device conductance to 1% relative accuracy [14]. We can see from Fig. 4

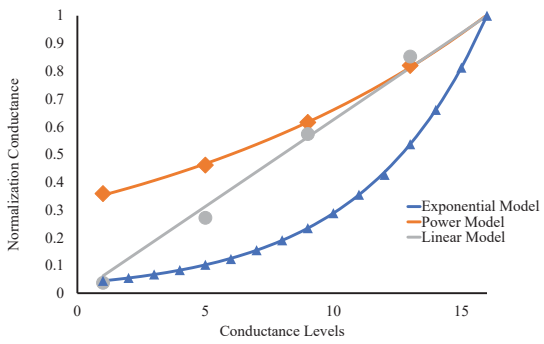

Fig. 4. Fitting Results for three models. The conductances are normalized. We have 16 states for exponential model and 4 states for deviated linear model and power model, according to the works of chip test result.

that our three models can fit the realistic conductances well. We will take those three models for further discussion in following sections.

IV. Computing Restrictions

As we already introduced, when mapping the matrix parameters to memristor cells, both $c_{i,j}$ and $g_{i,j}$ will not be continuous but fixed point values when considering real-life applications:the memristor cell has only fixed number of conductance levels. Then an extremely critical problem rises: even if $c_{i,j}$ and $g_{i,j}$ have the same number of bits, say, n bits, it is easy to find that we cannot just map n-bit $c_{i,j}$ to n-bit $g_{i,j}$ by Equ. (2). Take integers for example, n-bit $c_{i,j}$ could be $1, 2, ..., 2^n$ (here we only take positive integers since one memristor crossbar can only represent positive parameters), then from Equ. (2) we know that $g_{i,j}$ need to be tuned into $g_s, 2g_s, ..., 2^n g_s$, which means the conductance distribution of each level need to be $g_k = k \cdot g_s$. This tell us that memristor crossbar with linear conductance distribution is needed to complete computing task. For the case that $c_{i,j}$ is a fraction, the similar result can be obtained because fixed-point fraction is also linear.

The conclusion above shows that we cannot directly use the memristor crossbar with non-linear distribution as computing unit. Fortunately, since we are using memristor crossbar array, an analog device, to compute discrete value, the quantization circuit, i.e., ADC converter is needed to get computing results. So actually, we do not require g_k strictly equal to $k \cdot g_s$ because the quantization process will eliminate small impact of deviation. For further discussion, we introduce two conditions which need to be satisfied for MVM.

Consider n-bit memristor cell and m-bit input voltage, which means the memristor cell has $L_g = 2^n$ conductance levels and the input voltage has $L_v = 2^m$ levels, while each level of conductance/voltage represents a certain n-bit/m-bit value: We suppose that the y-th level of conductance g_y represents value b_y and the x-th level of voltage V_x represents value a_x. Here both b_y and a_x are fixed-point. When looking into the Equ. (1), it could be found that we first do numerical multiplication and then the addition, both of which are two linear operations. Therefore, if we still want to get the correct answer with non-linear conductance distribution, the two linear operation conditions need to take into concern:

$$\begin{aligned} &V_{x_1} g_{y_1} = V_{x_2} g_{y_2} \ or \ V_{x_1} g_{y_1} \approx V_{x_2} g_{y_2}, \\ &when \ a_{x_1} b_{y_1} = a_{x_2} b_{y_2} \end{aligned} \tag{5}$$

$$\begin{aligned} &I_{x_1 y_1} + I_{x_2 y_2} = I_{x_3 y_3} + I_{x_4 y_4} \\ &or I_{x_1 y_1} + I_{x_2 y_2} \approx I_{x_3 y_3} + I_{x_4 y_4}, \\ &when \ a_{x_1} b_{y_1} + a_{x_2} b_{y_2} = a_{x_3} b_{y_3} + a_{x_4} b_{y_4} \end{aligned} \tag{6}$$

Where $I_{x_1 y_1} = V_{x_1} g_{y_1}$. These two conditions are to make sure that the computation currents representing the same value will be quantized to the same output results. And the two conditions can be strictly expressed by inequations:

$$V_{x_1} g_{y_1} < V_{x_2} g_{y_2}, \ when \ x_1 y_1 < x_2 y_2 \tag{7}$$

$$\begin{aligned} &I_{x_1 y_1} + I_{x_2 y_2} < I_{x_3 y_3} + I_{x_4 y_4}, \\ &when \ x_1 y_1 + x_2 y_2 < x_3 y_3 + x_4 y_4 \end{aligned} \tag{8}$$

Where we can get $a_{x_1} b_{y_1} < a_{x_2} b_{y_2}$ from $x_1 y_1 < x_2 y_2$.Since the conductance levels are already fixed for a certain memristor crossbar, all we can do is to figure out how to design V_x or some additional circuits to make condition (7), (8) satisfied. We will discuss those methods in Section IV. Here we will show that the power model can satisfy the restriction (7) but not restriction (8), and both restrictions cannot be satisfied by the exponential model with more than 2 bits.

For power model, $g_y = C_1 y^a$, we only need to take $V_x = C_2 x^a$, then $V_x g_y = C(xy)^a$. It is clear that $C(xy)^a < C(pq)^a$ when $xy < pq$, i.e., $V_x g_y < V_p g_q$. But for exponential model, fix V_1 and consider the restriction for V_2:

$$\begin{aligned} &V_1 g_{2y-1} < V_2 g_y < V_1 g_{2y+1}, && y = 1, 2, ..., L_g/2 \\ &V_1 C a^{2y-1} < V_2 C a^y < V_1 C a^{2y+1}, && y = 1, 2, ..., L_g/2 \\ &V_1 a^{y-1} < V_2 < V_1 a^{y+1}, && y = 1, 2, ..., L_g/2 \end{aligned}$$

When $y \geq 3$, the inequations above have no solutions since we need to satisfy both $V_2 < V_1 a^2$ and $V_2 > V_1 a^2$.

However, when it comes to condition (8), both power model and exponential model cannot complete addition operation: the set of values (conductances) is not closed to addition operation. Take a simple example for explanation, if we just use 1-bit voltage which means we only have one voltage state V or 0. Then for integers or fractions, which are closed to addition operation, we can find $Va_p + Va_q = Va_t$ while $Vg_p + Vg_q = Vg_t$ does not exist.

On the other hand, even we got a crossbar with linear conductance distribution, it can be exactly $g_k = Ck$ due to the fabrication deviation. So for an approximately linear device, $g_k = Ck + \delta$, when apply voltage V onto such cell, we will get $Vg_k = VCk + V\delta$, where VCk is the correct answer but $V\delta$ is the deviation. Then, if either V or δ is big enough, it will cause computing error.

V. Tolerating Schemes

In the section above, we discuss how real-life devices cannot be used directly for computation, since most of the conductance distributions are not linear, and for power distribution model we can get the multiplication operation done by voltage design but the addition operation cannot be done directly, while for exponential distribution model both operations cannot be done. Even approximately linear conductance distribution will also cause computing error due to deviation. In this section, we proposed three methods to tolerate the non-linear distribution.

A. Least Square Method for Approximately Linear Model

For many memristor based computing architectures, the thresholds for sensing are linear when using ADC converter to complete quantization. For n-bit memristor cell and m-bit voltage, we have $g_y = C_1 y$, $V_x = C_2 x$ to represent the fixed-point matrix and vector value b_y, a_x, where $1 \leq y \leq 2^n$, $1 \leq x \leq 2^m$. Then the ideal z-th level of output current will be $I_z = Cz = Cxy$, where $C = C_1 C_2$, I_z represent value of $c_z = a_p b_q$, and $1 \leq z = xy \leq 2^{m+n}$. And we can see that quantized step for I_z is C. Then for models deviated from linearity, we propose least square based method to design V_x

in order to make such computing has the smallest computing error. The main idea is to let single-cell multiplication result be as close to Cxy as possible. Consider the squared error when apply voltage V_x to a single cell: $(g_y V_x - Cxy)^2$, then the whole error is:

$$L = \sum_k \sum_j (g_k V_j - I_k)^2 = \sum_k \sum_j (g_k V_j - Ckj)^2 \quad (9)$$

Take partial derivative of V_j then we can get optimized V_j:

$$V_j = \frac{jC \sum_k k g_k}{\sum_k g_k^2} \quad (10)$$

Moreover, when complete computing, we may find that the possibility of one specific output value depends on the distribution of inputs and the possibility indicates the importance of corresponding error. So to optimize the least square method we can weight different error:

$$L = \sum_k \sum_j w_{kj} (g_k V_j - I_{kj})^2 = \sum_k \sum_j w_{kj} (g_k V_j - Ckj)^2 \quad (11)$$

Where w_{ij} is the weight for result I_{kj}. Therefore, V_j is expressed as:

$$V_j = \frac{Cj \sum_k k w_{kj} g_k}{\sum_k w_{kj} g_k^2} \quad (12)$$

We can choose the value of w_{ij} through applying the possibility that result I_{kj} appears. And for those vector and matrix elements are uniformly distributed, $w_{kj} = 1$ for all k, j will be implemented.

B. Peripheral Circuit for Power Model

We already know that we can use power-increased voltages to make computation available for those conductances with power distribution, but we still need to solve the addition problem. Since we can read out the correct answer after single-cell multiplication, here we use a peripheral circuit to complete the addition operation after reading.

The computing result of g_y and V_x : $I_{xy} = C(xy)^a$ is powered increasing, and we need non-uniform quantized circuit to convert $C(xy)^a$ to xy, then the addition operation can go on, which means relation of output and input of the circuit is $i_{out} = (i_{in}/C)^{1/a}$. One direct solution is using ADC converter and digital circuits, but the cost is huge: for instance, 4-bit memristor cell and 3-bit voltage will get $7a$-bit answer, and if $a = 2$, we need 14-bit converter, which is far from efficient. Fortunately, there is a simple logarithmic operational circuit with only 6 transistors having the characteristic of $i_{out} = \alpha \ln(\beta i_{in})$ [15], which may be used to approximately approach the compression ability of $i_{out} = (i_{in}/C)^{1/a}$.

First we add a bias to get $i_{out} = \alpha \ln(\beta i_i n + 1)$. To achieve the approaching process, we also define a mean square error function:

$$L = \sum_x \sum_y (xy - \alpha \ln(\beta (xy)^a + 1))^2 \quad (13)$$

To minimize the equation above, we calculate the partial derivative:

$$\begin{aligned} \frac{\partial L}{\partial \alpha} &= -\sum_x \sum_y 2 \ln(\beta (xy)^a + 1)(xy - \alpha \ln(\beta (xy)^a + 1)) \\ \frac{\partial L}{\partial \beta} &= -\sum_x \sum_y \frac{2\alpha (xy)^a}{\beta (xy)^a + 1} (xy - \alpha \ln(\beta (xy)^a + 1)) \end{aligned} \quad (14)$$

The equations above are difficult to get analytic solutions for α and β, but we can calculate the numerical solutions with gradient descent method. Here we provides four couples of α and β, respective for $a = \sqrt{2}, 2, 2.5, 3$, which are listed in Table 2. We plot the approaching performance compare with expected linearity in Fig. 5 – for example, we want n from n^2, and output is $\alpha \ln(\beta n^2 + 1)$. We discuss the detail performance in Section VI.

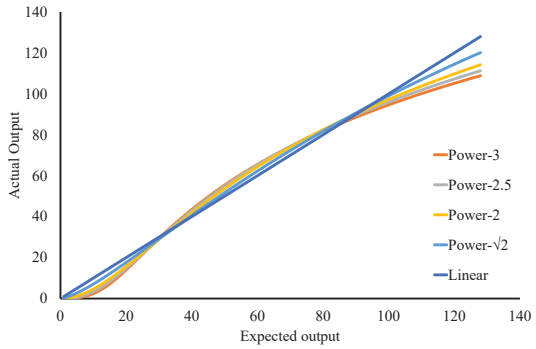

Fig. 5. Performance of Logarithmic Operation Circuit. Four different bases are chosen: $\sqrt{2}, 2, 2.5, 3$. The x-axis present the value we want and the y-axis present actual value. The ideal linear line is plotted to refer to.

As shown in Fig. 3 (b), we have to mention that using the peripheral circuits for compressing power value to linear value will increase the time complexity from $O(1)$ to $O(n)$, since we can only do vector-vector mulitplication. But it is still worth the cost otherwise we even cannot use the crossbar for computing integers or fractions.

C. Using Exponential Model for Neural Network

We are not able to use memristor crossbar with exponential model for MVM with more than 2 bits as it even cannot complete multiplication on single cell, for fixed-point value cannot be mapped onto the cells.

However, researchers find that we do not necessarily need linear fixed-point value to represent the weights in neural network because training networks with weight decay leads to final weights that are distributed non-uniformly around 0 [16], and this indicates that using weights of exponential representations can achieve the same performance as float weights does. The study also shows that using base-2 and base-$\sqrt{2}$ weights and inputs can achieve the same performance with almost no loss in classification performance.

So if we want to make memristor crossbar with exponential conductances distribution suitable for computing, we can train the neural network with fixed weight value, i.e., base-a exponential value corresponding to specific memristor. Considering that the weights in neural network are normalized, for a n-bit and base-a memristor device, the weights need to be quantized to $0, a^{-n+1}, a^{-n+2}$, ..., 1. We use the quantization method as follows:

$$\widetilde{x} = Clip(b^{Round(\log_b x)}, b^{-m+1}, 1) \quad (15)$$

Where

$$Clip(x, \min, \max) = \begin{cases} 0, & x < \min \\ 1, & x > \max \\ x, & else \end{cases} \quad (16)$$

Note that during the training process, we still need to adjust the full-precision weights but we use quantization at inference. This can be done with an extra quantization layer [16] or store unit [17].

VI. Simulation Results

We conduct three experiments to evaluate our tolerating schemes. For approximately linear model and power model, which can still complete the fixed-point MVM, we test the root mean square error (RMSE) of our methods. And for the exponential model, we test the accuracy of several neural networks with the corresponding quantized weights.

TABLE I

THE MSER OF COMPUTATION BY ORIGINAL VOLTAGES AND LEAST SQUARES OPTIMIZED VOLTAGES. WE SET THREE VARIANCES THAT MAKE CONDUCTANCES DEVIATE FROM LINEARITY: $0.05\Delta g$, $0.10\Delta g$, AND THE SIZE OF CROSS BAR IS SET TO 64, 128, 256, 512.

size	$0.05\Delta g$			$0.10\Delta g$			$0.15\Delta g$		
	Original RMSE	Optimized RMSE	Improvement	Original RMSE	Optimized RMSE	Improvement	Original RMSE	Optimized RMSE	Improvement
512	5.93	5.68	4.22%	11.85	11.33	4.37%	17.88	17.08	4.46%
256	3.92	3.76	4.10%	7.77	7.46	4.07%	11.67	11.20	4.05%
128	2.68	2.59	3.51%	5.34	5.14	3.75%	7.93	7.64	3.65%
64	1.89	1.82	3.43%	3.70	3.57	3.35%	5.42	5.24	3.27%

TABLE II

THE MSER OF COMPUTATION BY ORIGINAL MEMRISTOR CROSSBAR AND MEMRISTORR CROSSBAR WITH OUR PERIPHERAL CIRCUITS. WE SET POWER VALUE THAT MAY BE USED TO FIT THE REALISTIC MEMRISTOR CONDUCTANCES DISTRIBUTION: $\sqrt{2}, 2, 2.5, 3$. THE PARAMETERS FOR CIRCUITS DESIGN ARE ALSO PRESENTED BELOW.

size	Power-$\sqrt{2}$ $\alpha = 82.55$ $\beta = 3.443 \cdot 10^{-3}$			Power-2 $\alpha = 36.42$ $\beta = 1.345 \cdot 10^{-3}$		
	Original RMSE	Optimized RMSE	Improvement	Original RMSE	Optimized RMSE	Improvement
512	1514	51.60	96.6%	10710	98.39	99.1%
256	1089	36.75	96.6%	7515	68.88	99.1%
128	770.5	25.84	96.6%	5334	49.11	99.1%
64	583.3	17.95	96.7%	3736	34.54	99.1%
size	Power-2.5 $\alpha = 25.16$ $\beta = 4.443 \cdot 10^{-4}$			Power-3 $\alpha = 19.34$ $\beta = 1.324 \cdot 10^{-4}$		
	Original RMSE	Optimized RMSE	Improvement	Original RMSE	Optimized RMSE	Improvement
512	43971	123.8	99.7%	1.755e05	138.3	99.9%
256	30987	87.57	99.7%	1.250e05	97.96	99.9%
128	22110	61.59	99.7%	8.726e04	70.02	99.9%
64	15543	44.06	99.7%	6.189e04	49.14	99.9%

A. Experimental Setup

We choose 64, 128, 256 and 512 as the sizes of matrix; the number of memristor bits is set to 4 (16 conductance levels) with 3-bit voltage (8 voltage levels). The standard deviation describing how conductances deviate from linearity (σ of δ_k) is set to $0.05\Delta g, 0.10\Delta g, 0.15\Delta g$, where Δg is the conductance width of two neighboring state. For power model, we choose 4 powers a: $\sqrt{2}, 2, 2.5, 3$. Besides, whether the parameters are integers or fractions is not essential, since it only decides that computing results may overflow or underflow. Here we assume that we have full-precision sensing mode and use integers for computing to avoid the overflow problem. Moreover, we use two crossbar arrays to represent positive and negative number, respectively, and we assume all the input voltages are non-negative. Finally, we randomly generate 1000 sets of deviated linear conductances and 100 couples of matrices and vectors for multiplication in each set. We evaluate our voltage design method by averaging RMSE over 1000 sets and 100 couples. For power model, we generate 10000 couples of matrices and vectors and compare RMSE with original result..

On the other hand, we take LeNet-5 and ResNet-18 to evaluate our quantization method for exponential model and the testing sets are MNIST and Cifar-10, respectively. Here we conduct our experiments on 2-bit, 3-bit, 4-bit memristor cells and the exponential bases we choose are 1.2 (fitting the memristor states in [6]), $\sqrt{2}$, 2, 3.

B. Performance for Matrix-Vector Multiplication

We compare the RMSE of our least squares designed voltage with unoptimized voltage. Table 1 shows the overall results for the 1000 sets of different conductance distribution. We can see that the least squares method improve the RMSE by more than 3%. Also, the method has better performance in larger size memristor crossbar array. We can see that the RMSE reduction increases from around 3.3% to 4.4% as the size increases from 64x64 to 512x512.

Table 2 shows the results for the power model. We can see huge improvement benefiting from our peripheral circuits: the RMSE gets reduced by 2 orders of magnitude for power-2 and even 3 orders of magnitude for power-3. We can see that the computation error with original crossbar structure increases rapidly with lager power, while our logarithmic operation shows better performance. Under the condition of 4-bit memrister cell, 3-bit voltage and 256×256 crossbar size, we can conclude that computing result using power model based memristor crossbar is far from correct, since the range of computing answer is from $-16 \times 8 \times 256 = -32768$ to 32768, but the RMSE gets more than 7000 with power more than 2. With our additional circuits, the error reduce to less than 100. Note that the error is even larger than 100000 when using power-3 model this is because that we sense the output current with full precision and the current is much larger than expected since the conductance increases rapidly with power model.

C. Performance for Neural Network

We first implement our base-1.2, $\sqrt{2}, 2, 3$ weights to trained neural network and get the accuracy on test set. We then finetune the models with training set and test the accuracy again. We conduct our experiments with two neural networks: Lenet-5 and ResNet-18, and the data sets we choose are MNIST and Cifar-10, respective for the two neural networks. The results are presented in Table 3, where using our memristor cell based quantization method to quantize the trained models. We can find that both small base with high precision and proper base with lower precision, like base-1.2 with 4-bit precision or base-2 with 2-bit precision, are able to achieve nearly the same performance as baseline. The accuracy decays when we quantize

TABLE III

THE INFERENCE ACCURACY (%) FOR THE TWO NEURAL NETWORKS AFTER QUANTIZING TRAINED MODELS. THE RESULTS UNDER DIFFERENT BIT WIDTH AND BASE OF EXPONENTIAL CONDUCTANCES ARE LISTED..

LeNet-5 on MNIST					ResNet-18 on Cifar-10				
Accuracy with Full Precision: 98.70					Accuracy with Full Precision: 85.40				
	Base-1.2	Base-$\sqrt{2}$	Base-2	Base-3		Base-1.2	Base-$\sqrt{2}$	Base-2	Base-3
2	97.96	96.70	95.40	98.07	2	29.48	40.90	79.48	76.68
3	97.71	98.59	98.55	98.07	3	63.47	82.76	82.09	76.55
4	98.71	98.64	98.41	98.07	4	84.79	84.46	82.16	76.56

TABLE IV

THE INFERENCE ACCURACY (%) FOR THE TWO NEURAL NETWORKS AFTER FINETUNING WITH QUANTIZED WEIGHTS. THE RESULTS UNDER DIFFERENT BIT WIDTH AND BASE OF EXPONENTIAL CONDUCTANCES ARE LISTED.

LeNet-5 on MNIST					ResNet-18 on Cifar-10				
Accuracy with Full Precision: 98.70					Accuracy with Full Precision: 85.40				
	Base-1.2	Base-$\sqrt{2}$	Base-2	Base-3		Base-1.2	Base-$\sqrt{2}$	Base-2	Base-3
2	98.00	98.09	98.59	98.59	2	82.02	83.53	85.14	84.99
3	98.27	98.66	98.52	98.64	3	84.05	85.23	85.17	84.89
4	98.66	98.58	98.47	98.51	4	85.64	85.73	82.28	85.24

through small base with low precision, or large base. The reasons may be that small base with low precision cannot represent values around 0 and bigger base cannot represent values that are far from 0. For example, the quantization steps under 2-bit precision and base-1.2 are $\{0, 0.5787, 0.6944, 0.8333, 1\}$ where weights between 0 and 0.5 cannot be quantized perfectly. The quantization steps under 3-bit precision and base-3 are $\{0, 0.0005, 0.0014, 0.0041, 0.0123, 0.0370, 0.1111, 0.3333, 1\}$ where weights between 0.333 and 1 also cannot be quantized to a similar value.

The results after finetuning are presented in Table 4, The results show that the neural network can perfectly tolerate the exponential weights: the accuracy remains almost the same in each case, no matter the base is $1.2, \sqrt{2}, 2, 3$, or testing on LeNet-5, ResNet-18. We can also see that slight accuracy decay may occur for smaller base and less bits. This is easy to understand because when quantization with small base through our method, the values concentrate around 1 as we discussed above, but we already introduced that actual weights in trained model concentrate around 0.

VII. CONCLUSION

In this work, we study the impacts of non-linear distribution of conductances that make memristor crossbar based matrix-vector multiplication unachievable. Two computing restrictions are set to make sure whether the distribution is suitable for computing. We test the restrictions for the three models we proposed to fit the realistic memristor conductance distribution. We use least squares based multi-level voltage design method to get least computing RMSE, and we design a peripheral circuit to make power distributed memristor computable. For the exponential model that cannot complete usual dot product operation, we propose an exponentially quantization method to quantize the weights of neural network so that the exponential model can get into application. Finally, the simulation results show that the least squares method gain around 4% improvement of RMSE and more than 99% improvements on RMSE is observed for power model. We also find that our methods benefit more when the size of crossbar increases. On the other hand, using exponentially quantized weights corresponding to memristor conductance distribution shows big success in LeNet-5 and ResNet-18, as the accuracy of recognition decays less than 1%.

REFERENCES

[1] P. Chi *et al.*, "PRIME: a novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," in *International Symposium on Computer Architecture*, 2016, pp. 27–39.

[2] M. Cheng *et al.*, "TIME: A training-in-memory architecture for memristor-based deep neural networks," in *DAC*, 2017, p. 26.

[3] F. Bedeschi *et al.*, "A bipolar-selected phase change memory featuring multi-level cell storage," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 217–227, 2009.

[4] W. C. Chien *et al.*, "Multi-level operation of fully CMOS compatible WOX resistive random access memory (CMOS)," in *IEEE International Memory Workshop*, 2009, pp. 1–2.

[5] J. Li *et al.*, "A novel reconfigurable sensing scheme for variable level storage in phase change memory," in *IEEE International Memory Workshop*, 2011, pp. 1–4.

[6] N. Papandreou *et al.*, "Programming algorithms for multilevel phase-change memory," in *IEEE International Symposium on Circuits and Systems*, 2011, pp. 329–332.

[7] S. R. Lee *et al.*, "Multi-level switching of triple-layered TaOx RRAM with excellent reliability for storage class memory," in *VLSI Technology*, 2012, pp. 71–72.

[8] P. Yao *et al.*, "Face classification using electronic synapses," *Nature Communications*, vol. 8, p. 15199, 2017.

[9] B. Li *et al.*, "Memristor-based approximated computation," in *ISLPED*, 2013, pp. 242–247.

[10] M. Hu *et al.*, "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication."

[11] X. Liu *et al.*, "Reno: A high-efficient reconfigurable neuromorphic computing accelerator design," in *DAC*, 2015, p. 66.

[12] A. Shafiee *et al.*, "ISAAC: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *International Symposium on Computer Architecture*, 2016, pp. 14–26.

[13] L. Xia *et al.*, "Technological exploration of RRAM crossbar array for matrix-vector multiplication," *Journal of Computer Science and Technology*, pp. 3–19, 2016.

[14] F. Alibart, L. Gao, B. D. Hoskins, and D. B. Strukov, "High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm." *Nanotechnology*, vol. 23, no. 7, p. 075201, 2012.

[15] M. A. Alabsi and K. M. Altamimi, "A current-mode controllable logarithmic function circuit using MOSFET in subthreshold," in *World Congress on Engineering and Computer Science*, 2012.

[16] D. Miyashita, E. H. Lee, and B. Murmann, "Convolutional neural networks using logarithmic data representation," *arXiv:1603.01025*, 2016.

[17] S. Zhou *et al.*, "DoReFa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv:1606.06160*, 2016.