ABSTRACT Many mainstream applications require multiply-accumulate calculations, such as image processing and neuromorphic computing. Multiply-accumulate calculations using memristor crossbar arrays is a remarkable method for extremely high implementation efficiency, whereas the memristor array fabrication technology is still not mature and it is difficult to fabricate large-scale arrays with high-yield, which will seriously affect the performance of the application running on the RRAM crossbar. This paper proposes an inputs split based calibration method that improves the application accuracy in tolerating variations and stuck-at-fault of memristor devices. To demonstrate the performance of the calibration algorithm, the case of image sharpening and three neural networks architectures are applied for simulation experiments. And experimental results show that the calculation accuracy can be improved by up to 26.83% at 90% yield of crossbar arrays, and the success rate of the algorithm can be as high as 99.3% when there are several arrays cascaded. It is of great significance to the application of arrays in multiply-accumulate computation.

INDEX TERMS memristors, crossbar array, calibration method

I. INTRODUCTION

As demand for neuromorphic computing rapidly develops, the traditional hardware implementation becomes less efficient [1]. In recent years, neuromorphic computing systems based on the conventional FPGA, GPU and CPU are widely used in industry and enterprises [2]–[4], such as mobile or real-time and robotics scenarios. As an important part of a neuromorphic computing system, a neural network often uses a weight matrix to represent a group of synapses [5]. Consequently, the computation process of deep neural networks (DNNs) can be transformed into matrix multiplication, but traditional hardware implementation of neural networks require a large amount of memory and consume excessive system resources [6] [7].

In recent years, the memristor [8] has received significant attention as synapses for neuromorphic systems [9]–[12], and memristor crossbar arrays can easily carry out matrix multiplication [13] which is a computationally expensive operation for neural networks [14]. The basic structure of a memristor crossbar array [15] is shown in Fig. 1. A memristor crossbar is an array with memristive devices at its cross-points, which receives voltage inputs (at its rows) and produces an output current (at its columns) that is the weighted summation of the encoded weights at the column and the input voltage pulses. It is a direct result of the Kirchhoff’s law. The voltage pulses as the input, and the current output along a column from any cross-point will be the product of the conductance at that cross-point. Moreover, the memristor can achieve an integration density of 100Gbits/cm² [16], which is a promising device for large-scale and highly
parallel neuromorphic computing [17] [18]. However, device variations and manufacturing yield are still a big issue in memristor fabrication [19] [20], and the memristor may be damaged by testing cycles and aggressive programming. These situations will remarkably degrade the multiply-accumulate computation accuracy. Unfortunately, there are few researches on improving application accuracy of memristive crossbar arrays. Most of works focused on the memristor fabrication reliability as electronic synapses [21] [22], and some works proposed solutions to neuromorphic computing with few levels [23] [24] or damaged device [25]. In addition, a number of hardware-based solutions [26]–[28] are proposed, however, these methods also bring inevitable power consumption and hardware cost. To maintain the performance affected by stuck-at-faults (SAF) or resistance variations of the application running on the RRAM crossbar, we propose a software-based calibration method for multiply-accumulate computation on RRAM crossbar arrays, which splits the input vector into a standard vector and another input. And to evaluate the performance of the method, the pearson correlation coefficient is introduced to measure the correlation of ideal output and calibration output. To show the performance of calibration method more intuitively, we executed the experiments on image sharpening and some neuromorphic computing architecture based on memristive crossbar arrays [10] [29] [30] to verify the calibration performance of the array in cascade, experimental results demonstrate that the accuracy can be effectively improved by the calibration method when the yield of device is too low.

The rest of this paper is organized as follows, Section II describes the related work of this paper. Section III introduces the calibration methodology for memristor crossbar arrays. Section IV introduces the pearson correlation coefficient to measure the correlation and exhibits the simulation performance of calibration method on a case of image sharpening and some neuromorphic computing architecture. The final Section V concludes the paper.

II. RELATED WORK
As memristor device suffers from various reliability issues, the idea of exploiting fault tolerance to improve the application efficiency of memristor crossbar arrays has attracted some contributions.

Liu et al. [25] proposed a retraining method for memristor-based neuromorphic computing systems to regain the network weights. This approach classifies synapses weights into insignificant and significant sets: defects on insignificant synapses hardly affects the system accuracy whereas the degradation of the accuracy is giant when the significant synapses fall at bad devices. The algorithm is designed for neural networks on crossbar arrays and requires knowledge of the structure of the neural network.

The error correct code (ECC) [31] with crossbar arrays was utilized to detect errors by using specific codes and recover the corrupted data. In contrast, the memristor-based neuromorphic computing showed a totally different vulnerability on each individual devices [32]. Besides, the computing accuracy reflects the impact and not the logic error of devices.

Li et al. [33] found that the memristor device slowly drifts from its original programmed state (e.g., a voltage of 0.1V causes the device to deviate about 2% from the original state in 1s), so the accuracy of memristor crossbar arrays based system decreases gradually. They presented an inline calibration method to improve system accuracy under drift.

A pre-computation algorithm called "VAT" was proposed by Liu et al. [34], which maps the vital synapses to the memristor device with low variation. In the training process of neural networks, a scalar parameter $\gamma$ is used for evaluating the resistance variation of memristor crossbar arrays. The new weight matrix derived by repeatedly self-adjusting the parameter $\gamma$ is utilized to construct the neural network.

III. PROPOSED CALIBRATION METHOD
Assuming a standard input vector $V_{in}^{STD}$, it has $n$ elements which have a value of $s$ for a $n \times m$ memristor crossbar array.

$$V_{in}^{STD} = \begin{bmatrix} s & s & \cdots & s \end{bmatrix}_{1 \times n}$$  (1)

So a input vector $V_{in}$ (the vector to be calculated) can be described as

$$V_{in} = V_{in}^{STD} + V_{in}' = \begin{bmatrix} V_1 & V_2 & \cdots & V_n \end{bmatrix}_{1 \times n} = \begin{bmatrix} s + V_1 & s + V_2 & \cdots & s + V_n \end{bmatrix}_{1 \times n}$$  (2)

where $V_{in}'$ is the input vector that is actually input into the crossbar array, and $V_{i}'$ is the $i^{th}$ element of the $V_{in}'$ vector. The matrix $G$ is used to describe the conductance of the devices in the crossbar array. The $G$ can be described as

$$G = \begin{bmatrix} G_{1,1} & G_{1,2} & \cdots & G_{1,m} \\ G_{2,1} & G_{2,2} & \cdots & G_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ G_{n,1} & G_{n,2} & \cdots & G_{n,m} \end{bmatrix}$$  (3)

where $G_{n,m}$ represents the conductance of the device in column m of row n. So when the input vectors are $V_{in}'$ and $V_{in}^{STD}$ respectively, the ideal output of crossbar array is described as

$$V_{out}' = V_{in}' \cdot G = \begin{bmatrix} V_1 & V_2 & \cdots & V_n \end{bmatrix}_{1 \times n} \cdot G = \begin{bmatrix} \sum_{i=1}^{n} V_1 \cdot G_{i,1} & \cdots & \sum_{i=1}^{n} V_n \cdot G_{i,m} \end{bmatrix}_{1 \times m}$$  (4)

$$V_{out}^{STD} = V_{in}^{STD} \cdot G = \begin{bmatrix} s & s & \cdots & s \end{bmatrix}_{1 \times n} \cdot G = \begin{bmatrix} \sum_{i=1}^{n} s \cdot G_{i,1} & \cdots & \sum_{i=1}^{n} s \cdot G_{i,m} \end{bmatrix}_{1 \times m}$$  (5)

where $V_{out}'$ represents the ideal output of vector $V_{in}'$ and $V_{out}^{STD}$ represents the ideal output of $V_{in}^{STD}$. Considering the yield of the crossbar array, the change of the conductance of the memristor in the array can be described by matrix $\Delta G$.

$$\Delta G = \begin{bmatrix} \Delta G_{1,1} & \Delta G_{1,2} & \cdots & \Delta G_{1,m} \\ \Delta G_{2,1} & \Delta G_{2,2} & \cdots & \Delta G_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta G_{n,1} & \Delta G_{n,2} & \cdots & \Delta G_{n,m} \end{bmatrix}$$  (6)
After considering the yield, the conductance in the array can be described as
\[
\tilde{G} = \begin{bmatrix}
G_{1,1} + \Delta G_{1,1} & \cdots & G_{1,m} + \Delta G_{1,m} \\
G_{2,1} + \Delta G_{2,1} & \cdots & G_{2,m} + \Delta G_{2,m} \\
\vdots & \ddots & \vdots \\
G_{n,1} + \Delta G_{n,1} & \cdots & G_{n,m} + \Delta G_{n,m}
\end{bmatrix}
\]  
(7)
and the original output of \(V_{in}\) can be described as
\[
\tilde{V}_{out} = \tilde{V}_{in} \cdot \tilde{G} = \begin{bmatrix}
V_{1}' \\
V_{2}' \\
\vdots \\
V_{n}'
\end{bmatrix}_{1 \times n} \cdot \tilde{G} = \begin{bmatrix}
\sum_{i=1}^{n} V_{1}' \cdot \tilde{G}_{1,1} + \cdots + \sum_{i=1}^{n} V_{n}' \cdot \tilde{G}_{n,m}
\end{bmatrix}_{1 \times m}
\]  
(8)
We use the \(\delta'\) to represent the difference between the original output and the ideal output of \(V_{in}'\), which is the error of the output due to the yield issue. So the \(\delta'\) can be described as
\[
\delta' = \tilde{V}_{out} - V_{out}' = \left[ \sum_{i=1}^{n} V_{i}' \cdot \Delta G_{i,1} \right]_{1 \times m}
\]  
(9)
Similarly, the original output of \(V_{in}^{STD}\) is described below.
\[
\tilde{V}_{out}^{STD} = V_{in}^{STD} \cdot \tilde{G} = \begin{bmatrix}
s \\
s \\
\vdots \\
s
\end{bmatrix}_{1 \times n} \cdot \tilde{G} = \begin{bmatrix}
\sum_{i=1}^{n} s \cdot \tilde{G}_{1,1} + \cdots + \sum_{i=1}^{n} s \cdot \tilde{G}_{n,m}
\end{bmatrix}_{1 \times m}
\]  
(10)
Without loss of generality, set \(s = 1\). The original output and error of the \(V_{in}^{STD}\) can be rewritten as
\[
\tilde{V}_{out}^{STD} = V_{in}^{STD} \cdot \tilde{G} = \begin{bmatrix}
1 \\
1 \\
\vdots \\
1
\end{bmatrix}_{1 \times n} \cdot \tilde{G} = \begin{bmatrix}
\sum_{i=1}^{n} \tilde{G}_{1,1} + \cdots + \sum_{i=1}^{n} \tilde{G}_{n,m}
\end{bmatrix}_{1 \times m}
\]  
(11)
\[
\delta^{STD} = \left[ \sum_{i=1}^{n} \Delta G_{i,1} + \cdots + \sum_{i=1}^{n} \Delta G_{i,m} \right]_{1 \times m} = \tilde{V}_{out}^{STD} - V_{out}^{STD}
\]  
(12)
As it can be seen from Eq. (12), we can pre-calculate \(V_{in}^{STD}\) by the matrix \(G\), so the output error of the \(V_{in}^{STD}\) can be calculated from the original output \(V_{out}^{STD}\) (as shown in Eq. (12)). The \(\delta'\) is estimated by \(P\) and \(\delta^{STD}\). And the definition of \(P\) is described as
\[
P = \frac{\sum_{i=1}^{n} V_{i}'}{n}
\]  
(13)
where \(V_{i}'\) is the \(i\)th element of the \(V_{in}'\) vector. The product of \(P\) and \(\delta^{STD}\) is used to approximate the \(\delta_{k}^{V_{in}}\).
\[
\tilde{\delta} = P \cdot \delta^{STD} \approx \delta'
\]  
(14)
where \(\tilde{\delta}\) is an estimate of \(\delta'\). Therefore, the calibrated output \(V_{out}^{STD}\) can be described as
\[
\tilde{V}_{out} = V_{out}^{STD} + \tilde{V}_{out}^{STD} - \tilde{\delta}
\]  
(15)
The calibration method takes \(\tilde{V}_{out}\) as the output of the input vector \(V_{in}\), and Fig. 2 shows the flowchart of the method. According to the principle of the method, the process of the calibration method can be summarized as follows.

1) Define a \(1 \times n\) standard vector \(V_{in}^{STD}\) that satisfies each element in the vector to be equal, and the value of \(s\) is usually taken as the median of the input data. (via Eq. (1))
2) Calculate the \(V_{out}^{STD}\) by matrix \(G\), measure the original output \(V_{out}^{STD}\), and calculate the \(\delta^{STD}\). (via Eq. (5), Eq. (11) and Eq. (12))
3) Split the input vector \(V_{in}\) into \(V_{in}^{STD}\) and \(V_{in}'\), measure the original output \(\tilde{V}_{in}'\), and calculate the \(P\) by \(V_{in}'\). (via Eq. (2), Eq. (8) and Eq. (13))
4) Calculate the approximate error \(\tilde{\delta}\) by \(P\) and \(\delta^{STD}\). (via Eq. (14))
5) Calculate the final output \(\tilde{V}_{out}\) by \(V_{out}^{STD}\), \(\tilde{V}_{out}\), and \(\tilde{\delta}\). (via Eq. (15))
The \(\Delta G_{i,k}(i = 1, 2, \ldots, n; j = 1, 2, \ldots, m)\) and \(V_{i}'(i = 1, 2, \ldots, n)\) can affect the accuracy, especially when \(\Delta G_{i,k}\) is so large. In other words, when \(\Delta G_{i,k}\) is very large, this also means that the yield of RRAM is very low.

IV. EXPERIMENTS
A. EXPERIMENTS SETUP
All experiments are conducted using an Intel Xeon E7 (2.6 GHz), 16GB DDR4 and a NVIDIA Titan XP graphics card. This work is implemented with the Theano python open-source library to train the neural network models, and HSPICE is also used for some circuit simulations. Experiments are conducted with Monte-Carlo simulation method.

1) Memristor Model
The simulation is conducted based on the Ti/AlOx/TaOx/Pt memristor device, which was proposed in our prior work [22]. The bottom electrode (Pt, 25nm) and the resistive switching layer (AlOx/TaOx, 5nm/3nm) were deposited by electron beam evaporation in the fabrication process. The top electrode (Ti, 30nm) was grown by magnetron sputtering. By the lump, three lithography processes were performed to form the patterns of the three different layers.
Multilevel devices have a resistance range of 1KΩ to 12KΩ, which can be tuned to target resistance by repeated programming, and 200 states achieved by applying triangular pulses are utilized for simulations.

2) Device Defects Generation Method

Device variations and RRAM yield are considered as device defects in our simulations. In detail, the damaged devices denote the single-bit failure (SBF) [25], that means a memristor that fixes in a low (900-1100Ω) or a high (10800-13200Ω) resistance state. When the simulated device occurs a SBF issue, the device is at a high resistance with a probability of 50%, and the probability of 50% is at a low resistance.

Fluctuating devices are also included in the device defects generation. The variation of memristor devices mainly includes two types: switching variation [35] and parametric variation [36]. The driving circuit causes switching variation during the writing or reading cycle, programmed voltage with micro variation leads to a large variation in the memristor resistance. The imperfect fabrication such as oxide thickness, line-edge roughness and random dopants causes the parametric variation. When the resistance of the memristor fluctuates, the change of resistance on RRAM crossbar array, from \( R_{ij} \) to \( \tilde{R}_{ij} \), is described as follows:

\[
\tilde{R}_{ij} \leftarrow R_{ij} \cdot \exp(\theta_{ij}); \theta_{ij} \sim N(0, \sigma^2)
\]  

where the \( \theta_{ij} \) denotes the resistance variation, which follows the lognormal distribution [26]. The parameter \( \sigma \) is utilized to denote the extent of the variation in the experimental simulation. Unless otherwise specified in the following simulation, the default value of \( \sigma \) is 0.8.

Fig. 3(a) shows a 28×28 pattern which selects from the MNIST dataset [37] that simulated programming in a 32×32 memristor crossbar array. In the process of simulated programming, 1%-20% of errors are generated by the write circuit [17]. Fig. 3(b) demonstrates the resistance distribution of a 15% defects 32×32 crossbar array. From the figure it can be seen that defects (variations and SBF) distribute stochastically across the crossbar array and blur the programmed pattern.

3) Image Sharpening with Calibration Method

In order to show the performance of calibration method more intuitively, a case of image sharpening is introduced. The memristor crossbar array is utilized for edge detection. A laplace mask is used to implement the convolution operation on the input image for achieving the edge of the image, a 3×3 laplace mask can be described as

\[
\text{Laplace} = \begin{bmatrix}
1 & 1 & 1 \\
1 & -8 & 1 \\
1 & 1 & 1
\end{bmatrix} \tag{17}
\]

When the edge detection is completed, the sharpened image can be obtained by subtracting the edge from the original image.

Fig. 4 demonstrates the case of image sharpening. As shown in Fig. 4(a), the classic image 'Lena' with 512×512 pixels is resized to 270×270 pixels after median filtered, and the zero-padding operation was executed around the image for the convenience of processing, so the size of the final image is 272×272. The input image is segmented into 9×9 small fields of 32×32 pixels, and a 1024×1800 crossbar array is used for completing the edge detection. The laplace mask is mapped into the array, which is used for performing the convolution operation. Each 32×32 input field produced 900 outputs by convolution. As shown in Eq. (17), positive and negative weights are existed in mask, so 1800 outputs were produced by differential output.

In detail, The image is divided into RGB channels for processing, each channel sends 32×32 field into the array, and 900 outputs generated after differential processing. The output of the array is the original output, and the output after applying the calibration method is the calibrated output.

Fig. 4(b) shows the ideal output for an array without defects, and Fig. 4(c) illustrates the original output of the array when the array yield is 95%.

Fig. 5(a)-(c) demonstrate histograms of the input voltages of the R/G/B three channels, and the standard voltage \( s \) for each channel is 0.0769V, 0.0380V and 0.0392V respectively.

In order to better evaluate the correlation between original output and calibrated output and ideal output, the Pearson's correlation coefficient was calculated. It ranges from -1 (total negative correlation) to 1 (total positive correlation), with 0 indicating no correlation. Some results are shown in Table 1.

### Table 1. The correlation strength corresponding to different ranges of coefficient.

<table>
<thead>
<tr>
<th>Range of Coefficient</th>
<th>Correlation Intensity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8-1.0</td>
<td>extremely strong correlation</td>
</tr>
<tr>
<td>0.6-0.8</td>
<td>strong correlation</td>
</tr>
<tr>
<td>0.4-0.6</td>
<td>moderate correlation</td>
</tr>
<tr>
<td>0.2-0.4</td>
<td>weak correlation</td>
</tr>
<tr>
<td>0.0-0.2</td>
<td>very weak correlation or no correlation</td>
</tr>
</tbody>
</table>

### Table 2. Average Pearson correlation coefficient of different defect percent.

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Defects</th>
<th>2%</th>
<th>4%</th>
<th>6%</th>
<th>8%</th>
<th>10%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>original</td>
<td>0.386</td>
<td>0.256</td>
<td>0.179</td>
<td>0.138</td>
<td>0.088</td>
<td></td>
</tr>
<tr>
<td>calibrated</td>
<td>0.827</td>
<td>0.766</td>
<td>0.726</td>
<td>0.682</td>
<td>0.650</td>
<td></td>
</tr>
</tbody>
</table>

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.
correlation coefficient (PCC) is introduced to measure. The formula of PCC can be described as

$$\rho_{X,Y} = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{(\sum X^2 - \frac{(\sum X)^2}{N})(\sum Y^2 - \frac{(\sum Y)^2}{N})}}$$  \hspace{1cm} (18)

where $\rho_{X,Y}$ is the coefficient, $X$ and $Y$ represent two vectors that need to be measured, and $N$ indicates the dimension of the input vectors. In practical computation, $X$ is replaced by ideal output and $Y$ is replaced by original output or calibrated output. Table 1 demonstrates the correlation strength corresponding to different ranges of correlation coefficients.

A thousand groups of ideal output, original output and calibrated output are used for PCC calculation. Fig. 5(d) shows the correlation coefficients of 1,000 groups of output at 95% yield. It can be seen that original outputs coefficients are below 0.40 and calibrated outputs coefficients are above 0.50, the minimum value of the calibrated output differs from the maximum value of the original output by 0.19. According to the statistics, the average of original outputs coefficient is 0.197, while the calibrated is 0.742. From Table 1, it can be seen that the correlation between the original output and the ideal output is very weak correlation, while the correlation between the calibrated output and the ideal output can be strong correlation.

Table 2 shows the average PCC of different defects. In Table 2, both variation and stuck-at faults are taken into account. In other words, the “variation” and “stuck-at faults” share the same proportion. At 98% array yield, the PCC of original output is 0.386, which corresponds to weak...
correlation, while the PCC of calibrated output can reach 0.827, which corresponds to extremely strong correlation. Obviously, the PCC decreases as the proportion of defects percentage increases. When the yield is only 90%, the PCC of original output is only 0.088, and the average PCC of the calibrated output can still reach 0.650, which remains strong correlation.

B. CALIBRATION IN NEUROMORPHIC COMPUTING
In the experiments, the MNIST dataset [37] is chosen for accuracy verification. The trained synapses weights are converted to conductivity values of memristors that fall within the bounded range of devices state.

Experiments are executed based on a 784 × 10 single-layer perceptron (SLP) [29], a 784 × 100 × 10 multi-layer perceptron (MLP) [30] and a "3-2-1" RRAM crossbar based cascaded architecture [12] which includes six convolutional neural networks. And each 28 × 28 handwritten image is converted to a 784 × 1 vector by voltage generator as the crossbar array input.

1) Single-Layer Perceptron on Crossbar Array
Fig. 6 demonstrates the implementation on crossbar array. It can be seen that a 784 × 10 SLP is mapped into a 784 × 20 memristive array. The matrix $W$ represents a 784 × 10 synaptic networks, which includes positive and negative values.

Therefore, each original value is extended in two parts, $W^+$ and $W^-$. If one element is a negative value, then the value is defined in $W^-$, and the value of $W^+$ is zero. Similarly, the positive value is defined in $W^+$ and the $W^-$ is zero [10]. In the actual mapping of the array, we use the minimum conductance ($S_{min}$) of device to indicate the zero value.

2) Multi-Layer Perceptron on Crossbar Array
In order to verify the performance of the method when two arrays are cascaded, a 784 × 100 × 10 MLP is applied. Fig. 7 shows the implementation on two crossbar arrays. The weight matrix $W_1$ is mapped to a 784 × 200 array. Similarly, matrix $W_2$ is mapped to a 100 × 20 crossbar array, and the output of the first array is treated as the input of the second array after passing through the absolute activation function (Abs) [10]. It can be seen that the outputs of both arrays are all calibrated.

3) Cascaded Architecture on Crossbar Array
In order to further verify the performance of the calibration method which utilizes on multiply crossbar arrays cascaded, the cascaded architecture [12] is implemented with large-scale memristor crossbar arrays. A "3-2-1" cascaded framework is utilized for simulation, which includes six basic computation units (BCU), and each BCU is a convolutional neural network. In each BCU, five 9 × 9 convolution kernels
4) Simulation Results

Device variations and yield are considered in our simulations. The 1%-20% defects in memristor crossbar array were randomly generated. In the experiments, the $s$ we used is determined according to the median of the training dataset (excluded zero value), and here $s = 0.0831$.

Fig. 9 illustrates the original accuracy and calibrated accuracy with defects percent, and the solid lines in the figure represent the average accuracy of each set of data. In each percent, the positions of fluctuating devices and damaged devices are randomly generated, and 100 experiments are executed in each defect percent. As figures shown, with the increase of defect percentages, the accuracy rate decreases, in other words, the computation errors of arrays increase.

Fig. 9(a) shows the comparison between the calibrated accuracy and the original accuracy of SLP ($784 \times 10$), MLP ($784 \times 10 \times 10$) and (c) cascaded framework ("3-2-1"). (d) Bar chart of the number of failed and successful of calibration method.

Table 3 shows the performance comparison between this work and the "Vortex" algorithm [34]. It can be seen that this work dramatically outperforms the "Vortex", especially when the resistance fluctuates greatly ($\sigma \geq 1.0$). From the table it can be seen that this work significantly surpasses "Vortex" by 29.36% accuracy with $\sigma = 1.0$ and 42.20% accuracy with $\sigma = 1.2$.

Table 4 demonstrates the performance of two methods on SLP ($784 \times 10$) and MLP ($784 \times 256 \times 10$), assuming percentage of defects only in weight matrix $W_1$, only in weight matrix $W_2$, or in both $W_1$ and $W_2$. It can be observed that Ref. [25] achieves 91.53% SLP accuracy and 92.93% MLP ($W_1$ and $W_2$) accuracy respectively, this work surpasses previous work by 0.12% on SLP accuracy and 0.10% MLP accuracy.
V. CONCLUSION

In this paper, a calibration method is proposed for tolerating variations and stuck-at-fault on RRAM crossbar, which effectively pulls up the calculation accuracy. Three neuromorphic computing architectures and a case study of image sharpening were mapped into the array for experiments. From the simulation results above, the accuracy of calculations has been greatly improved (nearly 30% of cascaded framework). Experiments show that the number of successes of the calibration method is far greater than the number of failures, the success rate of the method can be as high as 99.3%. Compared with state-of-the-art algorithm, the proposed method can obtain desired accuracy with defects on crossbar arrays.

REFERENCES

SHENG-YANG SUN received the B.S. degree in information engineering from the National University of Defense Technology, Changsha, China, where he is currently pursuing the M.S. degree in electronic science and information technology. His current research interests include neuromorphic computing, memristor modeling, neuromorphic hardware and deep learning.

QINGJIANG LI received the Ph.D. degree in electronic science from the National University of Defense Technology, Changsha, China, in 2014. He is currently an Associate Professor with the College of Electronic Science, National University of Defense Technology. His current research interests include memristor characterization and applied circuits.

HUI XU received the Ph.D. degree in information engineering from the National University of Defense Technology, Changsha, China, in 1995. He is currently a Professor with the College of Electronic Science, National University of Defense Technology. His current research interests include circuits and signal processing systems.

JIWEI LI received the M.S. degree in electrical engineering from the National University of Defense Technology, Changsha, China, where he is currently pursuing the Ph.D. degree in electronic science and technology. His current research interests include circuit design, spiking neural network.

ZHIWEI LI received the M.S. degree in electronic science and technology from the National University of Defense Technology, Changsha, China, where he is currently pursuing the Ph.D. degree in electronic science and technology. His current research interests include circuit design, 3-D memory technology, memristor modeling, and memristive system.

YI SUN received the M.S. degree in electrical engineering from the National University of Defense Technology, Changsha, China, where he is currently pursuing the Ph.D. degree in electronic science and technology. His current research interests include neuromorphic computing and memristor characterization.

HAIJUN LIU received the Ph.D. degree in information and telecommunication systems from the National University of Defense Technology, Changsha, China, in 2010. He is currently an Associate Professor with the College of Electronic Science, National University of Defense Technology. His current research interests include the study of memristor and memristive system, and applications based on two-terminal resistive switches.