Current Injection: A Hardware Method of Adapting Non-Ideal Effects of ReRAM Based Deep Learning Accelerator

Gengxin Liu¹, Manqing Hu¹, Yitian Zhang², E Du¹, and Dekang Liu³

Abstract Non-ideal effects of ReRAM are the major limitations of deploying crossbar based neural network accelerator in the real world. Noise injection could effectively mitigate the non-ideal effects because it is equivalent to an adaptive regularization to the neural network. However, software-based noise injection involves computation-hungry retraining and data extraction. In this paper, we propose a ReRAM crossbar based neural network accelerator with current injection to adapt non-ideal effects in the ReRAM crossbar. We inject current into the crossbar through randomly set ReRAM cells to add the adaptive regularization, and no retraining and data extraction is needed in our proposal. We evaluate our method on three neural networks: LeNet-5, ResNet-20 and ResNet-50. Results show that current injection can reduce the accuracy degradation due to Stuck at Fault (SAF) and IR drop to 1%, and 5%, respectively.

key words: ReRAM, Neural Network, Accelerator, Current Injection, Crossbar

Classification: Integrated circuits

1. Introduction

In the big data era, the “memory wall” is becoming the toughest challenge as we encounter extreme data and computation-intensive workloads. In those workloads, data movement is much more expensive than the computation itself: DRAM access would consume over 100 times more energy than a floating-point operation [1, 2]. Thus implement computing into a memory array (in-memory computing) appears as a promising approach to address such challenges. As one of the most popular emerging memory technologies, Random-Access-Memory (ReRAM) shows great potential to be the next-generation high-speed memory cell because of its high density and high memory-accessing bandwidth. Moreover, the current-mode weighted summation operation in the ReRAM crossbar intrinsically matches the fused multiply-add (FMA) computation, which is dominant in the artificial neural networks [3, 4]. Hence, ReRAM is also a top candidate for the upcoming in-memory accelerator for neural networks [5, 6, 7]. On the other hand, due to analog computing’s nature, the accuracy of the ReRAM crossbar’s output is hampered by many non-ideal effects, such as wire resistance (IR drop), process variation, and electrical noise. [8]. Many recent works have investigated such issues, which are summarized as follows.

In [9, 10, 11], the authors use different methods with retraining the weights of target Deep Neural Network (DNN) mapped to the crossbar array to mitigate various non-ideal effects. However, retraining the DNN is very compute-intensive. Moreover, typical anti-nonideal effect retraining considers and addresses the deterministic non-ideal effect, leaving non-ideal effects with stochastic behavior remain unsolved.

In [8], He et al. have proven that by injecting noise into the network during the training phase, the crossbar’s non-ideal effects can be adapted. However, their method is based on obtaining the noise profile of each crossbar during training, which is a huge computation burden.

Thus, instead of retraining the DNN whenever deploying a new accelerator, we propose different techniques for non-ideal effects, in which no retraining is needed. Our main contributions in this work can be enumerated as:

- We develop a model of the random set of the ReRAM cells when the writing voltage pulse is lower than the device’s set threshold.

- We add small overhead auxiliary ReRAM cells and develop a ReRAM program paradigm to control the random set of the ReRAM and make sure that all the ReRAM follows the same distribution.

- We propose a current injection methodology by injecting the auxiliary current into the ReRAM crossbar as a variance source to the DNN. Such a method could eventually regularize the DNN model to make it intrinsically adaptive to varying non-ideal effects. Note that such current injection is added to the initial DNN training for crossbar mapping, and no re-training is needed for each particular accelerator.
We validate our result using PytorX [8] on MNIST, CIFAR-10, and ImageNet dataset, respectively.

The rest of this paper is organized as follows: Section 2 shows the background of this research and preliminary results. Section 3 demonstrates our method to randomly set the ReRAM and injecting current into the neural network. Section 4 draws the results and discussions. Finally, Section 5 concludes this paper.

2. ReRAM Crossbar Based Neural Network Accelerator and Its Non-ideal Effects

2.1 Single array as dot-product engine

The primary computation of convolution and fully connected layer in deep convolution neural network is the dot product with bias offset among tensors, and we write element-wise dot-product in a format as:

\[ y = w^T \cdot x + b = \left[ w^T \right] \cdot \begin{bmatrix} x & b \end{bmatrix} \]

where \( w \) and \( x \) are vectorized weights and inputs, respectively, and \( b \) is the bias. We assume that bias can be easily added to the outputs using simple adders. Hence in this paper, we only focus on the computation between the input and the weight. In our hardware mapping, both weights (\( w \in \mathbb{Z} \)) and inputs (\( x \in \mathbb{Z} \)) are quantized into fixed-point representation. We choose 8-bit for both the inputs and the weights, where each one has 8-bit digits in 2’s complement format.

The two corresponding columns in the positive and the negative array can be described as:

\[ \text{Current} \]

The DAC array can be described as:

\[ \text{DAC} \]

where \( G_{ij}^+ \) and \( G_{ij}^- \) are vectorized weights and inputs, respectively, and \( \Delta \) is the reference voltage, and \( \Delta V_{\text{DAC}} \) is the minimum voltage step of the DAC. Therefore, the current flowing into the differential ADC for \( j \)-th column pair (i.e., two corresponding columns in the positive and the negative array) can be described as:

\[ I_{\text{ADC},j} = \sum_{i=1}^{M} \left( V_{\text{DAC},i} - V_{\text{ref}} \right) \cdot \left( G_{ij,n}^+ - G_{ij,n}^- \right) \]  

and

\[ G^+ = \begin{cases} \Delta G \cdot w + G_{\text{min}}, & \text{if } w \geq 0 \\ G_{\text{min}}, & \text{if } w < 0 \end{cases}, \quad G^- = \begin{cases} \Delta G \cdot w + G_{\text{min}}, & \text{if } w < 0 \\ G_{\text{min}}, & \text{if } w \geq 0 \end{cases} \]

where \( \Delta G \) is the incremental conductance of the ReRAM cell being set. Therefore, Eq. 3 can be reformatted as:

\[ I_{\text{ADC},j} = \left( \Delta V_{\text{DAC}} \cdot \Delta G \right) \sum_{i=1}^{M} w_{d,i} \cdot x_{d,i} \]

In this work, we assume that \( \Delta V_{\text{DAC}} \) and \( \Delta G \) are fixed by hardware, while \( \Delta I_{\text{ADC}} \) is configurable through tuning the reference current source of the ADC. Our method optimizes the layer-wise \( k \), then return it as a fixed value. Thus, the computation workload on digital co-processor is minimized.

2.2 Non-ideal Effects of the ReRAM Crossbar

2.2.1 device defects

There are two kinds of Stuck-at-Fault (SAF) defects in ReRAM devices, which are Stuck-at-Zero (SA0) and Stuck-at-One (SA1), respectively [13, 14, 15]. The SA0 defect is normally caused by shorted devices, which makes the conductance of the defected ReRAM cell stuck at high state (i.e., \( \Delta G + G_{\text{min}} \)). The SA1 defect is resulting from the cell or interconnects wire being permanently open. In that case, the defected ReRAM cells stuck at low conductance state (i.e., \( G_{\text{min}} \)). The ReRAM fabrication results in [16] show 1.75% and 9.04% defect rate for SA0 and SA1, respectively.

2.2.2 wire resistance

In reality, the degradation of the crossbar output current is highly correlated with both input voltage and ReRAM conductance distribution. In [17], Liu et al. employ a gradient search method to compensate for the weight matrix for IR-drop caused by the wire resistance. However, the process needs to be repeated whenever the crossbar has different input signals. In [8], He et al. employ the statistics of current drift introduced by IR drop as a stochastic noise source in the training phase of DNN. However, to obtain the accurate dropout data, the nodal analysis was utilized, imposing
ReRAM is 8.13, with a standard deviation of 0.95. Also, we establish the hardware method in this subsection. The only necessary trimming of our model is to change \( \mu  \) and \( \sigma  \) for each scenario and apply them to Eq. (8) respectively. Fig. 3 shows the model prediction from Eq. (8). Note that the mismatch is now known. Therefore we can calculate the \( \mu  \) and \( \sigma  \) for each device mismatch caused by process variation. Compared to the circuit level model proposed in [22], our model is independent of technology and can be applied to other memristor devices with the Gaussian set process. Therefore our model is compatible with the behavioral level simulation method used for large scale networks [8, 23]. The only necessary trimming of our model is to change \( \mu  \) and \( \sigma  \) based on device characterization.

To verify our model, we simulate the set process of the ReRAM cell’s random set, through which we can control the amount of current injected into the crossbar. In [18], Z. Jiang et al. have modeled the evolution of the gap distance \( u_r(t) \) by applying Arrhenius law and calculating the probability for oxygen ions to overcome the migration barriers, \( E_A \). In addition to the deterministic dynamics, the generation and migration of oxygen vacancies, which are determined by the kinetic energy of the ions, are inherently random [18]. To verify the I-V characteristics of the ReRAM, we employ the Verilog-A model proposed in [18] with the key parameters shown in Table I. These parameters can be obtained from [19, 20].

To obtain the statistics for both set stochasticity and device-to-device variation, we simulated the pulse numbers required to trigger the set transition (with a fixed 10 ns pulse width) during 300 trials. We also repeated such simulation for 200 different sets of parameters with 6\( \sigma  \) variation of 10%, which is the worst-case scenario found for stable memristor devices noted in the literature [21]. Fig. 2 shows the obtained statistical distribution of the ReRAM set triggered by the input pulse train. In Fig. 2a, for one particular ReRAM, the number of input pulses needs to set the device roughly follows a Gaussian distribution with a mean value \( \mu  \) of 8.27 and standard derivation \( \sigma  \) of 3.17. We also show the cumulative distribution function (CDF) of the distribution, which we used to model the ReRAM device. In Fig. 2b, the distribution of \( \mu  \) and \( \sigma  \) across different parameter settings are shown. The average number of voltage pulses to set the ReRAM is 8.13, with a standard deviation of 0.95. Also, most devices have an average \( \sigma  \) of 3.18, with a standard deviation of about 0.68.

Based on the simulation results, we propose a set model of the ReRAM cells, that can be easily embedded into any machine learning codes. Moreover, we include the process variation in the model to account for device mismatch. Using the CDF shown in Fig. 2b, set condition \( y  \) is:

\[
y = \begin{cases} 
1 & \text{with probability } P = \Phi(n) \\
0 & \text{with probability } 1 - P 
\end{cases}
\]

where \( n  \) is the number of set voltage pulses, and \( \Phi(n) \) is the CDF of Gaussian distribution with process variation:

\[
\Phi(n) = \frac{1}{2} \left[ 1 + \text{erf} \left( \frac{n - \mu_\mu + N(0, \sigma_\mu)}{\left(\mu_\sigma + N(0, \sigma_\sigma)\right) \sqrt{2}} \right) \right]
\]

where \( N(0, x) \) is a Gaussian distribution with zero mean and variance \( x  \). We only sample \( N(0, x) \) once at the initialization of the program because it is the device mismatch caused by process variation. Compared to the circuit level model proposed in [22], our model is independent of technology and can be applied to other memristor devices with the Gaussian set process. Therefore our model is compatible with the behavioral level simulation method used for large scale networks [8, 23]. The only necessary trimming of our model is to change \( \mu  \), \( \sigma  \), \( \mu_\mu  \), and \( \sigma_\sigma \) based on device characterization.

3.2 Hardware Method for Current Injection

Using the statistic model we obtained in the previous section, we establish the hardware method in this subsection. The state-of-the-art fabrication result has shown the 6-bit [24]...
In the present work, we will employ 8-bit multi-level ReRAM cells to represent the weight. To precisely control the magnitude of the injected current, we add an auxiliary ReRAM cell as the current source, which is depicted in Fig. 4.

In Fig. 4, we add auxiliary ReRAM cells, $G_a$, into the crossbar. We apply a small voltage $V_{Ixb} = \Delta V_{DAC}$ to the word-line of the $G_a$. Thus we inject an auxiliary current $I = \Delta V_{DAC} G_a$ into the crossbar, where $V_{DAC}$ is the minimum step voltage stage of the DAC. We employ two program paradigms to our crossbar. One is the method proposed in [26], which is applied to all regular ReRAM cells in Fig. 4. Hence the regular ReRAM cells can be programmed to the exact 8-bit value. Another program paradigm is to program the auxiliary ReRAM cell, $G_{min}$, with a single voltage pulse, so that we can apply the model developed in Sec. 3.1 to predict the conductance after program. The conductance of the auxiliary ReRAM cells is

$$G_a = (G_{max} - G_{min}) N(\mu, \sigma) + G_{min} \quad (9)$$

where $N()$ is the probability density function of the Gaussian distribution, $\mu$ and $\sigma$ can be found in Fig. 2a. Therefore, an auxiliary current $I = V_{Ixb} G_a$ is injected into our crossbar. Consequently, we can rewrite the output current of the crossbar as

$$y = \frac{I_{ADC,i}}{\Delta I_{ADC}} = \frac{1}{k} \left( \sum_{i=1}^{M} w_{d,i} \cdot x_{d,i} + \frac{k G_a}{\Delta G} \right) \quad (10)$$

Henceforth the effective variance of the conductance of the ReRAM cell in the crossbar is $k G_a / \Delta G$.

To train the network with current injection, we add $k G_a / \Delta G$ as a random "bias" to every layer. Thus, Our approach could be easily integrated into any deep learning framework.

### 4. Results and Discussions

In order to investigate on the proposed neural network accelerator under various non-ideal effects, we utilized the comprehensive simulation framework PytorX proposed by He et al. in [8] as our machine learning framework. PytorX is implemented in Python and based on PyTorch, a mainstream DNN training and evaluation platform. We deeply tweaked PytorX’s circuit solver to include the Random model in Sec. 3.1. To demonstrate the effectiveness of our proposed method, we take the classical object recognition task as an example. Three network structures (LeNet-5 variant, ResNet-20 [27] and ResNet-50 [27]) on different datasets (MNIST [28], CIFAR-10, and imagenet [29]) are studied in this section. Tab. 2 shows the simulation parameter applied to our in-house tweaked Pytorx.

| Symbol | Description | Value |
|--------|-------------|-------|
| $f$    | Operating frequency | 0.01–1 GHz |
| $V_{dd}$ | Supply voltage | 3.3 V |
| $V_{ref}$ | Reference voltage | 1.67 V |
| $G_{min}/G_{max}$ | min/max conductance | 375/0.375 $\mu$S |
| $\Delta G$ | Conductance step size | 2.7 $\mu$S |
| $T$ | Temperature | 300 K |
| $D$ | Crossbar Dimension | 32, 64, 128 |
| $V_{pulse}$ | Set voltage pulse for $G_a$ | 1.3 V |

#### 4.1 Current Injection Results for SAF

In [8, 14, 10], the authors show in their experiments that SAF degrades accuracy drastically with wider and deeper network structure on a larger dataset. Fig. 5 depicts the test accuracy on MNIST, CIFAR-10 and imagenet datasets using LeNet-5 and ResNet-20, respectively. The SAF rate we used is ranging from 0% to 10% for both SA0 and SA1. SA0 is caused by shorted devices, hence the short current introduced by the defective device can easily overload the output ADC and cause a significant error. Thus it is hard to directly perform fix against the SA0 cell. We employ the row flipping transformation method proposed by B. Zhang et al. [30] to change SA0 errors to SA1 errors effectively. Hence we propose a software and hardware co-design method to reduce the effect of SAF.
1) Firstly, we convert all SA0 error to SA1 error using row flipping transformation.
2) Secondly, we employ hardware current injection to reduce the accuracy degradation caused by both SA0 and SA1. We assume that the SAF error is independent of crossbar size. Thus in Fig. 5, we only show the results with crossbar size of $64 \times 64$. The accuracy degradation caused by SAF increases when the network becomes deeper. For MNIST data set, the accuracy of LeNet-5 decreases to 30% when SA0 rate approaches 8%. On the other hand, the MNIST dataset can obtain an accuracy of 75% even when SA1 rate is 10%. For the CIFAR-10 dataset, the accuracy of the ResNet-20 drops to 30% when the SA0 rate is 4% and is near 70% when SA1 is 10%. Finally, for the imagenet, only 3% SA0 defect rate can lead to a complete system malfunction (with accuracy under 20%) of the ResNet-50 network.

Moreover, when the network topology changes, the distribution needs to be recalculated. Using hardware-based current injection, on the other hand, we can provide a one-shot solution to the IR-drop, at a price of hardware overhead. We demonstrate our result in Tab. III.

![Fig. 5: Test accuracy comparison of the $64 \times 64$ ReRAM crossbar has SAF errors, with and without variance injection.](image)

After we applied our countermeasure to the SAF defect, the accuracy of degradation can be greatly suppressed. In Fig. 5, the SAF-free accuracy is 99.81%, 95.65%, and 80.12% for MNIST, CIFAR-10 and imagenet, respectively. With our error correction method, the worst-case accuracy is 97±0.08% for MNIST, 94±0.09% for CIFAR-10, and 78±0.09% for imagenet, where the accuracy gap is within 1%.

4.2 Current Injection Results for IR Drop

The IR-drop caused by wire resistance is another dominant factor which may cause the system to malfunction. For a specific wire technology, the IR-drop is not only determined by the crossbar dimension but also strongly correlated to both input voltage magnitudes and conductance distribution of ReRAM cells. Different from SAF, the IR-drop is very sensitive to the crossbar size [8]. In [8], He et al. have provided a software-based noise injection method to mitigate the IR drop effect. However, to extract the current distribution of every crossbar in the network is computation-hungry.

From Tab. III, for the MNIST dataset, a direct mapping of neural network to $32 \times 32$, $64 \times 64$, and $128 \times 128$ crossbars will result in accuracies of 95%, 35%, and 15%, respectively. For deeper networks, such as ResNet-20 on CIFAR-10 and ResNet-50 on imagenet, crossbar malfunctions at crossbar sizes of $64 \times 64$ and $128 \times 128$. Even with a smaller crossbar size of $32 \times 32$, the accuracy degradation is significant. Our results show that hardware current injection can work very well at small ($32 \times 32$) and median ($64 \times 64$) crossbar sizes on MNIST and CIFAR-10 data set. Due to the very deep architecture of ResNet50, the accuracy degradation is still significant for the imagenet dataset, particularly on $128 \times 128$ crossbars. We also compare our results with the results in [8]. Our result is comparable to the software noise injection method. One should note that due to the huge computation burden, the authors did not report the result for the deep network (on CIFAR and imagenet data sets) and large crossbar size, e.g., $128 \times 128$.

5. Conclusion

We propose current injection—a hardware approach to adapt the ReRAM crossbar based deep learning accelerator to non-ideal effects. Our method covers two non-ideal effects—SAF errors and IR drops. We evaluate our method with three neural networks: LeNet-5, ResNet-20, and ResNet-50, on three data sets: MNIST, CIFAR-10, and imagenet. Using a two-step software and hardware co-design method, we reduce the accuracy degradation owing to SAF to within 1%. The current injection also attenuates the accuracy degradation caused by IR drop, especially on small ($32 \times 32$) and median ($64 \times 64$) crossbars, generating a reduced accuracy degradation within 5%.

References

[1] Stephen W Keckler et al.: “Scaling the “Memory Wall”: Designer track,” IEEE/ACM International Con-
ference on Computer-Aided Design (2012) 271 (DOI: 10.1145/2429384.2429437).

[2] S. Liu et al.: “Gpus and the future of parallel computing,” IEEE Micro 31 (2011) 7 (DOI: 10.1109/MM.2011.89).

[3] Linghao Song et al.: “Pipepilot: A pipelined reram-based accelerator for deep learning,” IEEE International Symposium on High Performance Computer Architecture (HPCA), (2017) 541 (DOI: 10.1109/HPCA.2017.55).

[4] Ping Chi et al.: “Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory,” ACM SIGARCH Computer Architecture News 44 (2016) 27 (DOI: 10.1109/ISCA.2016.13).

[5] Shuangchen Li et al.: “Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories,” Proceedings of the 53rd Annual Design Automation Conference (2016) 1 (DOI: 10.1145/2897937.2898064).

[6] Jie Lin and Jiann-Shiun Yuan: “A scalable and reconﬁgurable in-memory architecture for ternary deep spiking neural network with reram based neurons,” Neurocomputing, 375 (2020) 102 (DOI: 10.1016/j.neucom.2019.09.082).

[7] Jiann-Shiun Yuan et al.: “Ultra-low-power design and hardware security using emerging technologies for internet of things,” Electronics 6 (2017) 67 (DOI: 10.3390/electronics6030067).

[8] Zhezhi He et al.: “Noise injection adaption: End-to-end reram crossbar non-ideal effect adaption for neural network mapping,” Proceedings of the 56th Annual Design Automation Conference 2019 (2019) 1 (DOI: 10.1145/3316781.3317870).

[9] Indranil Chakraborty, et al.: “Technology aware training in memristive neuromorphic systems for nonideal synaptic crossbars,” IEEE Transactions on Emerging Topics in Computational Intelligence 5 (2018) 335 (DOI: 10.1109/TETCI.2018.2829919).

[10] Lerong Chen et al.: “Accelerator-friendly neural-network training: learning variations and defects in ram crossbar,” Proceedings of the Conference on Design, Automation & Test in Europe (2017) 19 (DOI: 10.23919/DATE.2017.7926952).

[11] Yang Zhang et al.: “Cacf: A novel circuit architecture co-optimization framework for improving performance, reliability and energy of reram-based main memory system,” ACM Transactions on Architecture and Code Optimization (TACO) 15 (2018) 1 (DOI: 10.1145/3195799).

[12] Norman P Jouppi et al.: “In-datacenter performance analysis of a tensor processing unit,” Proceedings of the 44th Annual International Symposium on Computer Architecture (2017) 1 (DOI: 10.1145/3079856.3080246).

[13] Cong Xu et al.: “Overcoming the challenges of crossbar resistive memory architectures,” 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA) (2015) 476 (DOI: 10.1109/HPCA.2015.7056056).

[14] Lixue Xia et al.: “Stuck-at fault tolerance in rram computing systems,” IEEE Journal on Emerging and Selected Topics in Circuits and Systems 8 102 (DOI: 10.1109/JETCAS.2017.2776980).

[15] Sparsh Mittal: “A survey of reram-based architectures for processing-in-memory and neural networks,” Machine learning and knowledge extraction 1 (2019) 75 (DOI: 10.3390/make1001005).

[16] Ching-Yi Chen et al.: “Rram defect modeling and failure analysis based on march test and a novel squeeze-search scheme,” IEEE Transactions on Computers 64 (2015) 180 (DOI: 10.1109/TC.2014.12).

[17] Beye Liu et al.: “Reduction and ir-drop compensations techniques for reliable neuromorphic computing systems,” Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design (2014) 63 (DOI: 10.1109/ICCAD.2014.7001330).

[18] Z. Jiang et al.: “Verilog-a compact model for oxide-based resistive random access memory (rram),” 2014 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD) (2014) 41–44 (DOI: 10.1109/SISPAD.2014.6931558).

[19] Jie Lin and Jiann-Shiun Yuan: “Analysis and simulation of capacitor-less reram-based stochastic neurons for the in-memory spiking neural network,” IEEE transactions on biomedical circuits and systems 12 (2018) 1004 (DOI: 10.1109/TBCAS.2018.2843286).

[20] Jie Lin and Jiann-Shiun Yuan: “Capacitor-less rram-based stochastic neuron for event-based unsupervised learning,” 2017 IEEE Biomedical Circuits and Systems Conference (BioCAS) (2017) 1 (DOI: 10.1109/BIOCAS.2017.8325169).

[21] Shimeng Yu et al.: “Stochastic learning in oxide binary synaptic device for neuromorphic computing,” Frontiers in Neuroscience, 7 (2013) 186 (DOI: 10.3389/fnins.2013.00186).

[22] Rawan Naous et al.: “Stochasticity modeling in memristors,” IEEE Transactions on Nanotechnology 15 (2015) 15 (DOI: 10.1109/TNANO.2015.2493960).

[23] Bodo Rueckauer et al.: “Conversion of continuous-valued deep networks to efficient event-driven networks for image classiﬁcation,” Frontiers in neuroscience 11 (2017) 682 (DOI: 10.3389/fnins.2017.00682).

[24] S. Yu et al.: “Scaling-up resistive synaptic arrays for neuro-inspired architecture: Challenges and prospect,” 2015 IEEE International Electron Devices Meeting (IEDM) (2015) 17.3.1 (DOI: 10.1109/IEDM.2015.7409718).

[25] Fabien Alibart et al.: “High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm,” Nanotechnology 23 (2012) 075201 (DOI: 10.1088/0957-4484/23/7/075201).

[26] Cong Xu, et al.: “Understanding the trade-offs in multi-level cell reram memory design,” Design Automation Conference (DAC) (2013) 1 (DOI: 10.1145/2463209.2488867).

[27] Kaiming He et al.: “Deep residual learning for image recognition,” Proceedings of the IEEE conference on computer vision and pattern recognition (2016) 770 (DOI: 10.1109/cvpr.2016.90).

[28] Yann LeCun et al.: “Gradient-based learning applied to document recognition,” Proceedings of the IEEE 86 (1998) 2278 (DOI: 10.1109/5.726791).

[29] Jia Deng et al.: “Imagenet: A large-scale hierarchical image database,” 2009 IEEE conference on computer vision and pattern recognition (2009) 248 (DOI: 10.1109/CVPR.2009.5206848).

[30] Baogang Zhang et al.: “Handling stuck-at-faults in memristor crossbar arrays using matrix transformations,” Proceedings of the 24th Asia and South Paciﬁc Design Automation Conference (2019) 438 (DOI: 10.1145/3287624.3287707).