Reliability Improvement in RRAM-based DNN for Edge Computing

Md. Oli-Uz-Zaman
Electrical and Computer Engineering
University of South Alabama
Mobile, AL, USA
mo2025@jagmail.southalabama.edu

Saleh Ahmad Khan
Electrical and Computer Engineering
University of South Alabama
Mobile, AL, USA
sk2021@jagmail.southalabama.edu

Geng Yuan, Yanzhi Wang
Electrical and Computer Engineering
Northeastern University
Boston, MA, USA
{yuan.geng, yanz.wang}@northeastern.edu

Zhiheng Liao, Jingyan Fu
Electrical and Computer Engineering
North Dakota State University
Fargo, ND, USA
{zhiheng.liao, jingyan.fu}@ndsu.edu

Caiwen Ding
Computer Science and Engineering
University of Connecticut
Storrs, CT, USA
caiwen.ding@uconn.edu

Jinhui Wang
Electrical and Computer Engineering
University of South Alabama
Mobile, AL, USA
jwang@southalabama.edu

2022 IEEE International Symposium on Circuits and Systems (ISCAS) | 978-1-6654-8485-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISCAS48785.2022.9937260

***Abstract*—Recently, the Resistive Random Access Memory (RRAM) has been paid more attention for edge computing applications in both academia and industry, because it offers power efficiency and low latency to perform the complex analog in-situ matrix-vector multiplication – the most fundamental operation of Deep Neural Networks (DNNs). But the Stuck at Fault (SAF) defect makes the RRAM unreliable for the practical implementation. A differential mapping method (DMM) is proposed in this paper to improve reliability by mitigate SAF defects from RRAM-based DNNs. Firstly, the weight distribution for the VGG8 model with the CIFAR10 dataset is presented and analyzed. Then the DMM is used for recovering the inference accuracies at 0.1% to 50% SAFs. The experiment results show that the DMM can recover DNNs to their original inference accuracies (90%), when the ratio of SAFs is smaller than 7.5%. And even when the SAF is in the extreme condition 50%, it is still highly efficient to recover the inference accuracy to 80%. What is more, the DMM is a highly reliable regulator to avoid power and timing overhead generated by SAFs.**

***Index Terms*—resistive random access memory (RRAM), deep neural network (DNN), edge computing, stuck at fault (SAF), differential mapping method, power, latency**

I. Introduction

The demand of edge devices is steadily increasing with the rapid technological explosion on the Internet of Things (IoT) domain. As the edge devices are getting more complex and require larger computational capability, conventional CMOS technology is becoming difficult to process the ever-growing data with its constrained speed and power consumption [1]. Especially, when DNNs (Deep Neural Networks) have tremendous succeeded in variety of applications including computer vision, natural language processing, medical diagnosis, big data analysis etc. [2], [3] creating the big space for them used as a fundamental element in edge devices, which further challenges CMOS technology based edge computing.

This work was supported in part by the National Science Foundation under Grant 1953544 and Grant 1855646.

Recently, Resistive Random-Access Memory (RRAM) with the attractive features, like non-volatility, faster switching, near-zero leakage power, uncomplicated co-integration with CMOS technology, low programming voltage, and low read/write latency has emerged to replace and/or complement the conventional CMOS technology [4], [5] for running complex DNN algorithms on edge devices.

RRAM was physically manufactured by Hewlett-Packard in 2008 [6] which was based on Professor Chua's postulation from 1971 [7]. RRAM can work for an analog operation to generate multilevel conductive states and memorize the amount of charge that has been flowed through it. It could retain that state until a new pulse is provided to change the previous conductive state [7]. This unique feature can be exploited to implement in-situ matrix-vector multiplication that is the most crucial operation for the weight update in DNNs. Despite having excellent properties, the RRAM-based DNN is yet to be commercially available because of Stuck-At-Fault (SAF) defect resulting from the low yield in the fabrication [8], device initialization (forming), and heavy device utilization [8], [9]. SAFs cause a discrepancy between original weights calculated from the algorithm and the mapped weight to crossbar arrays. Since some RRAMs are forever stuck at the Low Resistive State (LRS) or High Resistive State (HRS) because of the SAF, the accuracy of the DNN model drastically degrades.

To alleviate the impact of SAF on the accuracy in a RRAM-based DNN, several works have been done before. The inherent self-healing capability of the neural network was proposed where weights with high influences will not be mapped to the defected RRAM cell [10]. SAF rescuing method was proposed where the algorithm identifies the significant weights first and remaps the weights in the area with less defected RRAMs [11]. On-Line Fault Detection [12], stuck-at-fault tolerance in RRAM computing systems [13] and recovering stuck-at-fault defects using matrix transformation [14] were also proposed

to restore the accuracy drop because of SAFs.

Although these techniques can partly recover the performance loss of a neural network, sometimes, they have to face challenges. This is because: 1) considering the sporadic nature of the SAF defect and the huge number of devices in IoT applications, an individual optimization and custom design for each device is too time consuming and is not realistic; 2) considering the complex periphery control circuit, the large hardware overhead in the system has made them an undesirable solution. So, a more universal solution is required to cover all devices irrespective of the distinctiveness of defects in RRAM arrays. In this paper, a new technology - differential mapping is proposed for the RRAM-based DNN to create a strong immunity to the SAF. It does not target to obtain the optimizing pattern for the individual device but cover any RRAM product and can be largely used to improve the overall inference accuracy.

This paper makes the following contributions:1) In order to investigate the accuracy drop in RRAM based DNNs due to SAFs, the weight distribution for the VGG8 model is presented and analyzed. 2) As the SAF defects are immensely random and vary from device to device, the accuracy drop of a DNN is listed for different SAF ratios from 0.1% to 50%. 3) A differential mapping method (DMM) is proposed to achieve an outstanding accuracy recovery even in extremely high SAF cases. This work validates the accuracy improvement in the VGG8 model with the CIFAR10 dataset. 4) The detailed power and latency information is provided to verify that the proposed DMM can effectively suppress power and timing overhead.

II. Methodology

A. SA1 and SA0

A SAF denotes a RRAM with the resistant state fixated to Stuck-At-1 (SA1) or Stuck-At-0 (SA0) [11], [14]. SAFs would significantly degrade the inference accuracy of the RRAM-based DNN, especially when RRAM cells with SAFs have the high ratio in arrays. However, SAFs are usually has high ratio in RRAM arrays [8], [9]. For example, researchers reported that only 63% of the RRAM cells are fault-free for a fabricated 4-Mb RRAM test chip [12]. There are multiple mechanisms to generate SAFs. In the manufacture, since a raw RRAM cell contains extremely high resistance, the newly fabricated RRAM has to experience a forming process before performing the normal read/write operation [9]. A forming process is that a high voltage pulse (as compared to the switching pulse) applies on a RRAM to decrease the resistance level to a normal LRS. Such a sophisticated forming process usually lasts a long time, for example 100 us, which is highly likely to make RRAM over formed. In this case, it becomes difficult to change the existing resistive state and cannot switch to HRS or in-between. Regrading the circuitry, one of mechanisms for inducing SAF is as following, as shown in Fig. 1. Bit Line (BL), Word Line (WL), and Select Line (SL) are the three terminals for each RRAM cell in a crossbar architecture to select a specific cell for performing the read/write operation. The broken WL in RRAM arrays can result in an open circuit to make RRAM cells inaccessible to write. Therefore, currents cannot go through those RRAM cells, if these RRAM cells originally are HRS, they would stay at HRS forever. In general, as reported in [8], the ratio of SA1 to SA0 is 9.04 to 1.54, 85.44% of SAFs in RRAM arrays is SA1. Such an unfortunate situation changes the training and inference accuracy of a RRAM-based DNN to a large extent.

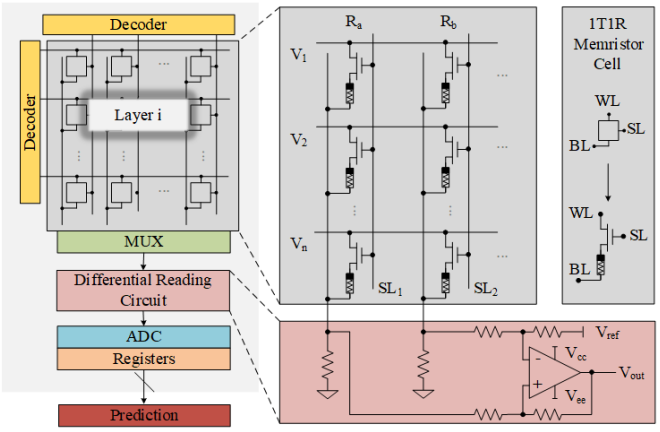

Fig. 1: Hardware Implementation for Differential Mapping Method

B. Weight Distribution

The VGG8 model is trained with the CIFAR10 dataset, weights have an approximately normal distribution, as shown in Fig. 2. The number of weights is located within each specific range. For instance, Fig. 2 shows that the number of weights greater than 0 and less than or equal 1e-07 is more than 2.35 million.

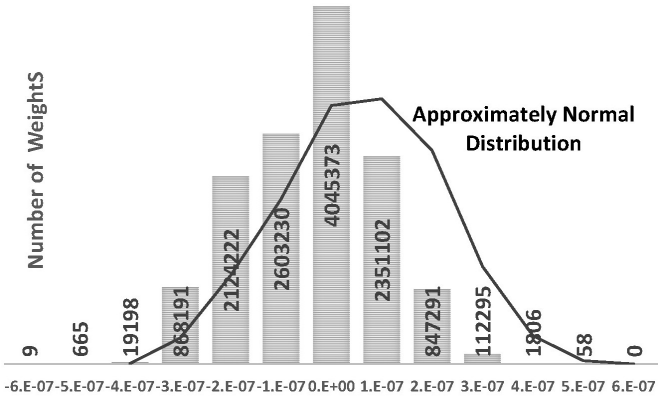

Fig. 2: Overall Weight Distribution of VGG8 Model Trained by CIFAR10 Dataset.

Although the VGG8 model contains more than 12.97 million weights, among which, 5.61 million are negative, 3.31 million are positive, and 4.04 million are neutral, most of them tend towards zero. As listed in Table I, in the VGG8 model, mean values of weights in each layer and all layers approach

0, and generally 99.83% of total weights are between -3e-07 and 3e-07. Therefore, when mapping such a VGG8 model to a RRAM-based DNN, usually most of RRAM cells are set to LRS. However, if the SAF occurs in RRAM, especially the SA1 dominates the SAF [8], it will generate a large deviation between the desired weight value and practical weight value, while the inference accuracy of the RRAM-based DNN will be significantly degraded.

TABLE I: Mean and Standard Deviation (STD) Values of Weights in Each Layer and All Layers of VGG8 Model

Layers	Mean	STD	Layers	Mean	STD
Layer 0	-3.05e-08	1.02e-07	Layer 4	-3.48e-08	1.05e-07
Layer 1	-2.81e-08	1.06e-07	Layer 5	-3.46e-08	1.07e-07
Layer 2	-3.47e-08	1.06e-07	Layer 6	-3.46e-08	1.07e-07
Layer 3	-3.44e-08	1.06e-07	Layer 7	-3.58e-08	1.07e-07
Overall	-3:45e-08	1.07e-07			

C. Differential Mapping Method (DMM)

DMM is proposed in this paper. Originally, in the RRAM-based DNN, one RRAM cell is used to store one weight. In our proposed DMM, one weight value is represented by the difference of values from two RRAM cells in order to create HRS as much as possible.

Since the SA1 dominates SAFs and causes most of the accuracy drop [8], after the DMM, the enormous number of SA1 overlap with HRS in RRAM arrays, which makes the system greatly immune to the SAF. The following sections will provide detailed information for the DMM.

In the traditional mapping method, when a certain weight from the algorithm level is mapped onto a RRAM cell, the weight value is mapped as a conductance or a resistance. So, the weights ranging from [-1, 1] are mapped according to its resistive state [LRS, HRS].

But in the DMM, one weight is split into a positive and a negative portion and mapped onto two different RRAM cells. Then the recombination of two memristor cells gives the value of the original weight. As a result, the DMM creates a weight ranging from [0, 1] and is mapped on the RRAM cell according to the same resistive state [LRS, HRS].

As the SA1 is 5.2 times as compared with SA0 [8] in RRAM arrays, the emphasis of achieving a hardware that creates an immunity against SA1 is more important. The original value of the algorithmic weight W within the range [-1, 1] will be represented by the DMM as follows:

$$\mathrm{W}_a = \begin{cases} 1 & W \geq 0, \\ 1 - |W| & W < 0, \end{cases} \quad (1) \qquad \mathrm{W}_b = \begin{cases} 1 - W & W > 0, \\ 1 & W \leq 0, \end{cases} \quad (2)$$

where W_a and W_b are the two parts of a single weight that will be stored in two different RRAM cells. And during the computation, the original weight will be extracted from two RRAM cells through a subtraction $(W_a - W_b)$.

Fig. 1 represents the hardware implementation of the proposed DMM. Here the weights W_a, W_b will be mapped on two RRAM cells according to the cells resistive level R_a, R_b respectively. Then an Op-Amp-based subtractor is used to subtract the weights during the reading operation [15], [16], as shown in Fig. 1.

Following this method, at least one of the two RRAM cells will store "1" (HRS) during mapping. Furthermore, a weight with 0 value will make two RRAM cells to separately store "1". So, the creation of more "1s" on the RRAM array will generate more overlaps between such HRS and SA1.

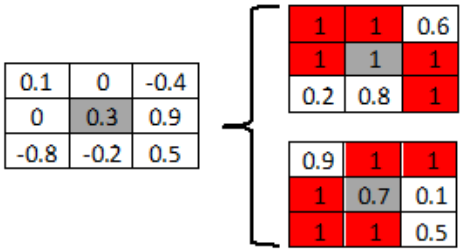

Fig. 3: Differential Mapping Method

For example, as shown in Fig. 3, a weight of 0.3 in the original array will be mapped as 1 and 0.7 in two new RRAM cells and the difference of the two RRAM values will be used as one weight while computing in a DNN. Actually, there are no "1" in the original RRAM array, but in two new arrays, eleven "1s" are generated and "1" is 11/18 of new RRAM arrays. Accordingly, the deviation between the desired weight value and practical weight value will greatly decrease when SAFs occur, preventing the inference accuracy drop.

III. Result and Discussion

DNN+NeuroSIM platform is used to verify our proposed method [17]. DNN+NeuroSIM is an integrated framework, which is developed to emulate DNN on RRAM based hardware accelerator. DNN+NeuroSim framework can support hierarchical organization from the device level (RRAM) to the circuit level (periphery circuits), to chip level (tiles of processing-elements built up by multiple sub-arrays, and global interconnect and buffer) and then to the algorithm level (DNN), enabling the accurate evaluation on the inference or training stage and the circuit-level performance metrics such as the power and latency. Along with DNN+NeuroSIM, the VGG8 model with CIFAR10 dataset as well as RRAM cells and periphery circuits with 40 nm technology are used to evaluate our proposed DMM regarding the inference accuracy, power and latency.

A. SAFs in RRAM Arrays

At the beginning, the original weights are mapped to RRAM arrays and SAFs (SA1 and SA0) are introduced to explore the effect on the DNN. The original inference accuracy achieved by an ideal RRAM-based DNN without SAFs is 90%. As the SAF occurs in RRAM arrays, the accuracy degrades rapidly. As listed in Table II, the accuracy degradation happens as small as 0.2% SAFs. From 2.5% SAFs, the DNN model is completely damaged and achieves only 10% inference accuracy which is merely equivalent to a random guessing.

TABLE II: Accuracy Degradation and Restoration with Different SAF Conditions

SAF (SA1 & SA0)	Accuracy No SAF	Accuracy Before DM[a]	Accuracy After DM
0.1%	90%	90%	90%
0.2%		89%	90%
0.5%		80%	90%
1.0%		42%	90%
2.5%		10%	90%
5.0%		10%	90%
7.5%		10%	90%
10%		10%	89%
15%		10%	89%
20%		10%	88%
30%		10%	87%
40%		10%	81%
50%		10%	80%

[a]DM: Differential Mapping

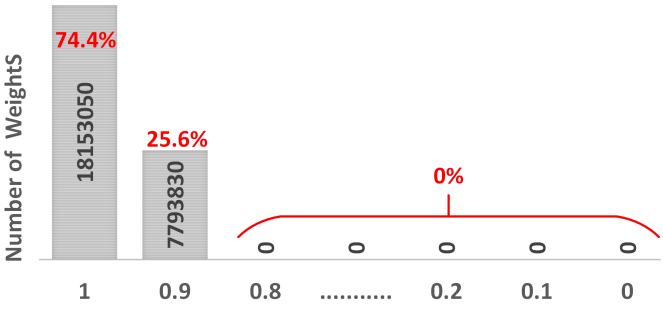

Fig. 4: Weights Distributions with the DMM

B. DMM Used in RRAM Arrays with SAFs

Our proposed DMM is used to restore the inference accuracy in RRAM-based DNNs. As listed in Table II, all inference accuracies in different SAF conditions are restored significantly. When SAFs from 0.2% to 7.5%, the RRAM-based DNN regains the original inference accuracy (90%), and even when SAFs is 50% in the extreme condition, the DMM is still highly efficient to recover the inference accuracy to 80%, as shown in Table II.

In the VGG8 model as shown in Fig. 4, after the DMM is applied, two RRAM arrays (each one includes 12.97 million elements) are employed for its original weights. In one array, the number of "1" is 7.35 million as 56.67% of all elements. The other array contains 9.65 million of "1" which is 74.40% of all elements. As shown in Fig. 4, totally, more than 18 million "1" are created. If 0.1%-7.5% SAFs exists in the RRAM-based DNN, most of SA1 will overlap with "1" in two new RRAM arrays. What is more, even in new RRAM arrays in the DMM, more than 7 million of weights (25.6%) less than 1 but greater than or equal 0.9 (sub-1) are mapped, it means even if some SA1s do not overlap with "1" cell, they will be mapped to such sub-1 cells, the difference between SAF1 and sub-1 cell is really small and cannot damage the DNN model. This is why the DMM provides the great immunity to SAFs. Only when SAFs is over 50%, more SAFs cannot overlap with "1" cell, so the accuracy can only be restored to 80%. Also, as discussed in Section II-A, only 14.56% of SAFs in RRAM arrays is SA0, so the inference of SA0 is neglectable. Therefore, in general, the efficiency of the DMM is remarkable as compared to the conventional mapping method.

C. Power and Latency Analysis

In this study, the DMM is applied in offline learning scenario, accordingly the energy consumed by the RRAM cell is for reading procedure. So, in RRAM-based DNNs, the energy consumption of each RRAM cell is calculated using the following equation [17],

$$\mathrm{E}_{cell} = RI_r^2 NT_{PULSE} \tag{3}$$

where R, I_r, N, and T_{PULSE} are respectively the resistance of a RRAM cell, the reading current, and number of applied reading pulses, and pulse width. Total energy consumption for a synaptic core is calculated using the sum of the energy for all RRAM cells [18].

Before the DMM is applied, SAFs with the low ratio from 0.1% to 2.5% nearly do not generate the overhead of the energy. But, with SAFs (mainly SA1) increasing, more and more defective RRAM cells with a larger R are inserted to replace original RRAM cells, and then the energy becomes much larger than before. As listed in Table 5, with 20% SAFs, the energy enlarges 3× than before. After the DMM is applied, although two RRAM cells are used to replace one cell, the total resistance R is not changed, $R = R_a + R_b$ as shown in Fig. 1. What is more, because the DMM use "1" (HRS) to cover most of SAFs and avoid many insertions of larger R, as listed in Table III, even with 20% SAFs, there is no significant changes of the energy consumption appeared after the DMM is applied.

TABLE III: Power and Latency Comparison Before and After Differential Mapping

SAF (SA1 & SA0)	Before DM[a]		After DM	
	Energy (J)	Latency (s)	Energy (J)	Latency (s)
0.1%	2.55e-05	66.490e-05	2.55e-05	66.480e-05
1.0%	2.44e-05	66.499e-05	2.55e-05	66.479e-05
2.5%	2.55e-05	66.479e-05	2.55e-05	66.480e-05
20%	7.13e-05	12.019e-04	2.54e-05	66.480e-05

[a]DM: Differential Mapping

Latency is basically the time required for the output voltage to reach the switching voltage threshold [18]. It can be calculated using the following equation [17],

$$Latency = \tau_f \sqrt{\ln(v_s)^2 + \frac{2\beta(1-v_s)}{rampInput \times \tau_f}} \tag{4}$$

where v_s and $rampInput$ are the normalized switching voltage threshold (typically 0.5) and input voltage ramp rate. $\beta = 1/(g_m R)$ is the reciprocal of the normalized input transconductance g_m times the output resistance R. τ_f is the total RC time constant at the output node. So, Equation (5) can be transferred as

$$Latency = RC \sqrt{\ln(v_s)^2 + \frac{2(1-v_s)}{rampInput \times RC \times g_m R}} \tag{5}$$

So, $Latency \propto \sqrt{mR^2C^2 + nC}$. m and n are constants. The total latency is calculated as the sum of gate level latency of all sub-circuits [19] [20].

Similar with the energy analysis, before the DMM is applied, SAFs from 0.1% to 2.5% nearly do not generate the overhead of the latency. But, with SAFs increasing, more and more defective RRAM cells with a bigger RC are inserted to replace original RRAM cells, and then the latency becomes much larger than before. As listed in Table 5, with 20% SAFs, the latency enlarges about $2\times$ than before. The DMM is applied to avoid many insertions of larger RC, as listed in Table III, even with 20% SAFs, there is no significant changes regarding the latency.

Therefore, the DMM acts as a great regulator to hold the same energy and latency for all RRAM-based DNNs with various SAF conditions.

IV. Comparison

As listed in Table IV, compared with the state-of-art, our proposed method has some advantages. Till date, other works on handling SAF defects require individual optimization because most of them need to firstly identify the defective cells and the most significant weights. Then they map those weights to the fault free regions or cells. Additionally, a complex control circuit is required and generates the significant circuitry overhead. Our proposed DMM creates a universal solution that would work in any RRAM device irrespective of the randomness of the SAF defects. It is significantly functional even in extreme hardware failure cases without adding peripheral circuits. Besides, another unique feature is that it is able to restore accuracy to a large extent against the nonlinear property which is an inevitable problem of RRAM cells. Furthermore, the power and latency are thoroughly analyzed in our work before and after the DMM to exhibit characteristics of RRAM-based DNNs in different SAF conditions which the other works often overlook.

TABLE IV: Comparison of the State-of-Art

Items	[10]	[11]	[12]	[13]	[14]	This Work
No Individual Optimization	×	×	×	×	×	√
No Complex Control Circuit	×	×	×	×	√	√
Extreme SAF Considerations	×	×	×	×	√	√
Power Analysis	×	×	×	×	×	√
Latency Analysis	×	×	×	×	×	√

V. Conclusion

Emerging RRAM are popular for the DNN hardware design and have demonstrated their potential in boosting the speed and energy efficiency of an analog matrix-vector multiplication. However, due to the immature fabrication technology and heavy utilization, commonly occurring SAFs seriously degrade inference accuracies of RRAM-based DNNs. In this paper, a differential mapping method (DMM) is proposed to mitigate SAF defects. Based on the VGG8 model with CIFAR10 dataset, the experiment results show that the DMM can recover the DNN to 90%, when the ratio of SAFs is smaller than 7.5%. And even when SAFs is in the extreme condition 50%, it is still highly efficient to recover the inference accuracy to 80%. Finally, the DMM can remove overhead of power and latency even when SAFs hold the large ratio in RRAM arrays.

References

[1] M. M. Waldrop, "The chips are down for moore's law," *Nature News*, vol. 530, no. 7589, pp. 144–147, Feb. 2016.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, Oct. 2018.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[4] E. Vianello, O. Thomas, M. Harrand, S. Onkaraiah, T. Cabout, B. Traoré, T. Diokh, H. Oucheikh, L. Perniola, G. Molas *et al.*, "Back-end 3d integration of hfo 2-based rrams for low-voltage advanced ic digital design," in *2013 International Conference on IC Design & Technology (ICICDT)*, May 2013, pp. 235–238.

[5] H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, and M.-J. Tsai, "Metal–oxide rram," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, Jun. 2012.

[6] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008.

[7] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on Circuit Theory*, vol. 18, no. 5, pp. 507–519, Sep. 1971.

[8] C.-Y. Chen, H.-C. Shih, C.-W. Wu, C.-H. Lin, P.-F. Chiu, S.-S. Sheu, and F. T. Chen, "Rram defect modeling and failure analysis based on march test and a novel squeeze-search scheme," *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 180–190, Jan. 2015.

[9] A. A. Gismatulin, G. N. Kamaev, V. N. Kruchinin, V. A. Gritsenko, O. M. Orlov, and A. Chin, "Charge transport mechanism in the forming-free memristor based on silicon nitride," *Scientific Reports*, vol. 11, no. 1, pp. 1–10, Jan. 2021.

[10] L. Chen, J. Li, Y. Chen, Q. Deng, J. Shen, X. Liang, and L. Jiang, "Accelerator-friendly neural-network training: Learning variations and defects in rram crossbar," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, Mar. 2017, pp. 19–24.

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

[12] L. Xia, M. Liu, X. Ning, K. Chakrabarty, and Y. Wang, "Fault-tolerant training with on-line fault detection for rram-based neural computing systems," in *54th Annual Design Automation Conference*, Jun. 2017, pp. 1–6.

[13] L. Xia, W. Huangfu, T. Tang, X. Yin, K. Chakrabarty, Y. Xie, Y. Wang, and H. Yang, "Stuck-at fault tolerance in rram computing systems," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 1, pp. 102–115, Nov. 2017.

[14] B. Zhang, N. Uysal, D. Fan, and R. Ewetz, "Handling stuck-at-fault defects using matrix transformation for robust inference of dnns," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2448–2460, Oct. 2019.

[15] O. Krestinskaya, A. P. James, and L. O. Chua, "Neuromemristive circuits for edge computing: A review," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 1, pp. 4–23, Jan. 2019.

[16] G. Yuan, Z. Liao, X. Ma, Y. Cai, Z. Kong, X. Shen, J. Fu, Z. Li, C. Zhang, H. Peng, N. Liu, A. Ren, J. Wang, and Y. Wang, "Improving dnn fault tolerance using weight pruning and differential crossbar mapping for reram-based edge ai," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, Apr. 2021, pp. 135–141.

[17] X. Peng, S. Huang, H. Jiang, A. Lu, and S. Yu, "Dnn+neurosim v2.0: An end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–14, Dec. 2020.

[18] J. Fu, Z. Liao, J. Liu, S. C. Smith, and J. Wang, "Memristor-based variation-enabled differentially private learning systems for edge computing in iot," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9672–9682, Jun. 2021.

[19] Z. Liao, J. Fu, and J. Wang, "Ameliorate performance of memristor-based anns in edge computing," *IEEE Transactions on Computers*, vol. 70, no. 8, pp. 1299–1310, Aug. 2021.

[20] J. Fu, Z. Liao, and J. Wang, "Cycle-to-cycle variation enabled energy efficient privacy preserving technology in ann," in *2020 IEEE 33rd International System-on-Chip Conference (SOCC)*, Sep. 2020, pp. 66–71.