An Efficient 3D ReRAM Convolution Processor Design for Binarized Weight Networks

Bokyung Kim, *Student Member, IEEE*, Edward Hanson, *Student Member, IEEE*, and Hai Li, *Fellow, IEEE*

***Abstract*—*Convolutional neural networks* (CNNs) have been evolving with tremendous success in visual recognition, obtaining human-level accuracy. The conventional hardware architecture, however, is facing difficulty in realizing real-time and energy-efficient operations on CNN. To efficiently operate CNN algorithms on the hardware, researchers are actively studying *processing-in-memory* (PIM) with *resistive random-access memory* (ReRAM). Digital PIM is particularly attractive because analog designs struggle with undesirable device properties and require additional circuits like analog-to-digital converter and digital-to-analog converter. However, the massive area originated from digital PIM is a hindrance to its applications. In this work, we present a *three-dimensional* (3D) ReRAM convolution logic processor design to tackle the limitation of digital PIM. At the hardware level, we leverage 3D ReRAM to take advantage of its area efficiency. The design simplicity without accuracy loss is accomplished by exploiting *binarized weight networks* (BWNs) at the algorithm level. Specifically, our 3D ReRAM processor computes the convolution of BWN based on a presumed full adder and a split-half addition scheme, which are proposed in this brief to maximize resource consumption efficiency. As a result, the proposed design achieves 3.7× to 5.7× and 5× to 42.5× area- and time-saving according to the bit precision in comparison to the original digital PIM.**

***Index Terms*—Convolutional neural network, CNN, resistive random-access memory, ReRAM, 3D ReRAM.**

I. Introduction

CONVOLUTIONAL neural networks (CNNs) have been quickly evolving, especially in image processing, due to the high effectiveness of convolution at extracting spatial features. ResNet, for example, reported 3.6% error rate in the *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) contest, which even surpasses the human-level performance (5%) [1]. The numbers of parameters and operations are also rapidly increasing when improving performance or extending CNNs to other domains. However, increasing size and complexity limits the execution of CNNs on hardware platforms because implementing large networks in the conventional architecture—*relying on its fundamental structure that separates memory devices from processing units* [2]—requires frequent data movement and considerable resource consumption. The conventional hardware is facing the limitation in executing advanced networks.

To address the limitation, researchers have explored *processing-in-memory* (PIM) hardware architecture that brings computation and memory components closer together. Among previously proposed designs, the architecture based on *resistive random-access memory* (ReRAM) arrays is a particularly compelling candidate [3], [4]. It is efficient in matrix-based operations [5] thanks to its advantages of recording weight values and performing *in situ* computation. PIM designs with an analog approach, however, demand complicated circuits for communicating with other components, e.g., *analog-to-digital converter* (ADC) and *digital-to-analog converter* (DAC) that occupy a substantial area and power consumption. Logical operation-based PIM is a different approach for eliminating the necessity of additional circuits. By using ReRAM as a digital device, it also removes the ambiguity coming from the device variations, a representative non-ideal property of ReRAM [6].

FloatPIM [7] is a state-of-the-art work of the digital PIM, which utilizes the *memristor aided logic* (MAGIC) [8] method for its system. However, an evident limitation in FloatPIM is its area cost. In that work, all the bits of required operands should be placed in the same row and duplicated into multiple rows because the parallel arrangement is essential to execute the MAGIC method. The digital representation further contributes to the area expansion. While the parallelism due to the structural advantage was able to reduce the computation cost, the area expansion remains a primary limitation.

Our work aims to address the limitation by integrating *three-dimensional* (3D) ReRAM at the hardware level and *binarized weight network* (BWN) at the algorithm level. 3D ReRAM has been experimented to load all the parameters and operations of large networks. The structure can dramatically decrease the occupied area. BWN that utilizes binarized weights has emerged to execute the learning algorithms on portable devices [9]–[12]. Binarized weights eliminate the need to multiply between multi-bit numbers and copy-and-paste of all the bits of weights in a kernel. We will leverage the complementary capabilities of the two approaches for improving CNN execution efficiency.

In this brief, we propose a convolution processor design integrating the two approaches. Two operation schemes are

Manuscript received February 4, 2021; accepted March 6, 2021. Date of publication March 22, 2021; date of current version April 30, 2021. This work was supported in part by the Air Force Research Lab (AFRL) under Grant FA8750-18-2-0121, and in part by the National Science Foundation (NSF) under Grant EECS-2023752. This brief was recommended by Associate Editor W. Zhao. *(Corresponding author: Hai Li.)*

The authors are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: bokyung.kim828@duke.edu; edward.t.hanson@duke.edu; hai.li@duke.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2021.3067840.

Digital Object Identifier 10.1109/TCSII.2021.3067840

TABLE I
ERROR RATE (%) ACCORDING TO BINARIZATION SCHEMES AND DATASETS: BINARYCONNECT (BC) [9], BINARIZED NEURAL NETWORKS (BNN) [10], [11], XNOR-NET (XNOR) [12], AND BINARY-WEIGHT-NETWORKS (BWN)

	No binarization	**Weight binarization**	Activation and weight binarization
MNIST	1.3±0.2	**1.29±0.08 (BC)**	1.4 (BNN)
SVHN	2.44	**2.30 (BC)**	2.53 (BNN)
CIFAR-10	10.94	**9.90 (BC)**	10.15 (BNN) 10.17 (XNOR)
ImageNet	43.3	**46.2 (BWN)** **64.6 (BC)**	55.8 (XNOR) 72.1 (BNN)

built to efficiently compute convolution layers of BWN. First, we propose a presumed full adder scheme to simplify the addition in FloatPIM. The scheme can be easily implemented in the crossbar architecture without additional cost for implementation. Second, a split-half addition method is proposed by exploiting the structural advantage of 3D ReRAM. Also, we suggest a bit representation converter to generalize our method. Our experiment shows that the proposed design can obtain 3.7× to 5.7× reductions of the execution area, in comparison with the original convolution of digital PIM in BWN. According to the bit representation methods, 5× to 42.5× time-saving effect can be obtained.

II. BACKGROUND

A. Binarized Weight Network

Neural networks could be highly redundant. Algorithm-level techniques like sparsity regularization and quantization are emerging to reduce the redundancies of networks [13], [14]. Specifically, binarization, which is an extreme form of quantization, is attractive for alleviating large resource demands. Since Courbariaux *et al.* proposed binarizing only the weights, related works looked towards binarizing both activations and weights [9]–[11]. By limiting bit numbers of weights and/or activations to a single bit, the storage, data movement cost, and computation complexity greatly improve. Moreover, binarized networks exhibit competitive accuracy. As shown in Table I, binarizing only weights induces only a small accuracy drop from the baseline, which can be concealed by the improved efficiency. *Binarized weight network* (BWN), hence, is adopted in this brief.

B. ReRAM-Based Logic

Kvatinsky *et al.* proposed a design composed of only ReRAM devices to implement the logic gates in a crossbar, called MAGIC [8]. Fig. 1(a) describes the method of NOR. Input data are stored in the devices, and the output device initially is set as a *low resistance state* (LRS, logic '1'). The bottom electrodes of all the devices are connected without any voltage sources. Then an operating source is applied to the top electrodes of input devices, whereas the top of the output device is grounded. According to the voltage distribution law, the input states determine the voltage given to the output device. If at least one input device is at LRS, the output device state will be changed to *high resistance state* (HRS, '0'); otherwise, remain at LRS. This behavior corresponds to the NOR gate operation.

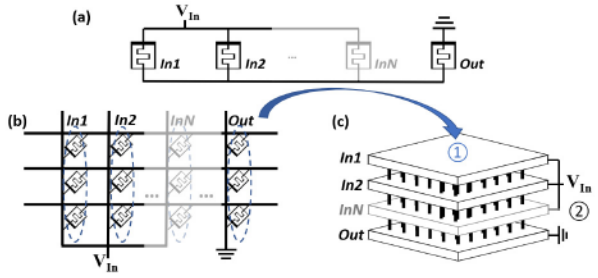

Fig. 1. NOR operation on various structures: (a) the basic structure; (b) the 2D crossbar array architecture; (c) two options in a 3D array: ① plane- and ② pillar-wise NOR.

TABLE II
SUMMARY OF TWO CONDITIONS AND CORRESPONDING RESULTS WHEN COMPUTING *Sum*

	Case 1	***Case 2***
If ..	$C_{out} == 1$ *or* $(A+B+C)' == 1$	$C_{out} == 0$ *and* $(A+B+C)' == 0$
$((A+B+C)'+C_{out})'$ is ..	0	1
then, Sum is ..	$(A'+B'+C')'$	1

The basic NOR gate structure can be extended to 2D and 3D arrays, as depicted in Figs. 1(b) and (c), respectively. In the 2D array, the horizontal word lines are connecting the top of input and output devices, and the bit lines are biased, as elaborated in the basic structure. Two options are available in the 3D architecture: the *plane-wise* and *pillar-wise* NOR operations. The plane-wise NOR is indeed applying the 2D array NOR method to each plane. With the vertical pillars connecting the engaged devices, the pillar-wise NOR can be achieved by applying voltages to the plane. Since the fabrication technology limits the number of layers, the plane-wise method is more practical and intuitive.

III. METHOD

In this section, we will elaborate on the proposed convolution processor based on digital 3D ReRAM architecture. First, the design concept of the *presumed full adder* is described in Section III-A. Section III-B introduces 3D ReRAM with the *split-half addition* for the multi-bit addition. At the end, Section III-C presents the overall processor design comprehensively.

A. Presumed Full Adder

This work uses NOR as the primary operation because the NOR function can be completed in one cycle. An addition operation can be expressed by a set of NOR functions such as:

$$C_{out} = ((A+B)' + (B+C)' + (C+A)')' \quad \text{and} \tag{1}$$

$$Sum = (((A'+B'+C')' + ((A+B+C)' + C_{out})')')'. \tag{2}$$

where A and B are input bits, and C represents the carry-in bit, which is C_{out} from the previous one-bit addition. As shown

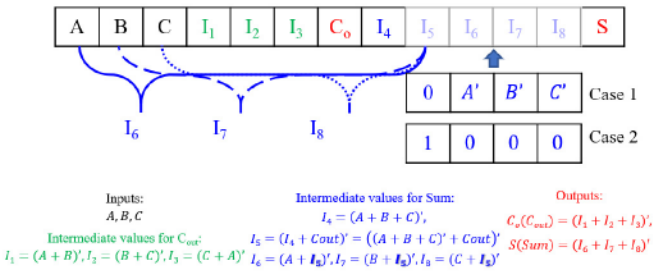

Fig. 2. Simplified one-bit full adder implemented in crossbar architecture.

in Eq. (1), C_{out} is computed in four NOR operations. Eq. (2) produces *Sum*, which can be split into two cases according to the value of $((A+B+C)'+C_{out})'$. When the term results in 0, *Sum* can be simplified as $(A'+B'+C')'$; otherwise, *Sum* will be 1. Table II summarizes the two conditions and the corresponding results.

We can simplify the operation of the one-bit addition based on this observation. Fig. 2 illustrates the one-bit *Sum* process in a row of the crossbar array. Each cell denotes one bit, and the position of each bit is represented conceptually. Black, green, blue, and red characters indicate inputs, intermediate values for C_{out}, intermediate values for *Sum*, and outputs, respectively.

The key of this design is to detect the values of C_{out} and $(A+B+C)'$ as a conventional detection circuit may degrade the latency and area. Rather than adding a circuit detecting C_{out} and $(A+B+C)'$, here, we devise a new strategy by utilizing I_5 in the calculation of I_6, I_7, and I_8. More specifically, we obtain the correct results by NORing the input bits and I_5. If $I_5=0$, NORing with each input is equivalent to the negation of each bit; Otherwise, when $I_5=1$, I_6, I_7, and I_8 will be 0 regardless of the other operand. After I_5 is obtained, we only need to negate every input to compute $(A'+B'+C')'$. In this way, $(I_6+I_7+I_8)'$ will always draw the correct *Sum* output.

The one-bit addition can be achieved in 10 cycles without any extra overhead, which is about a 16.7% reduction when compared to 12 cycles of the original full adder [7]. The effect of cycle reduction can further improve when expanding to 32-bit and 64-bit additions.

B. 3D ReRAM With Split-Half Addition

2D designs demand all bits computed in order, and accordingly, large resource consumption is still inevitable. We exploit 3D ReRAM as our basic structure to offset the resource consumption caused by the sequential bit-wise addition. Specifically, after inputs are split into halves, each half is stored in different layers. The first half bits, including the *most significant bit* (MSB), are duplicated and stored in both Layer 1 and Layer 2, assuming carry-in is 0 and 1, respectively. The rest bits are in Layer 3. All the layers are computed simultaneously, and then final values are chosen according to C_{out} from Layer 3.

In an example of adding 0010 and 0101, Layer 1 and Layer 2 are for 00 and 01 (the first half bits), and Layer 3 is for 10 and 01 (the rest of the inputs). Layer 1 computes as if the carry-out from Layer 3, that is, the carry-in to Layer 1 is 0; whereas, Layer 2 outputs a result based on the carry-in 1. After completing the addition of the halves, either Layer 1 or Layer 2 will be chosen following the real carry-out from Layer 1. Therefore, in this example, Layer 1 will be adopted because the carry-out from Layer 3 is 0. As such, a 32-bit addition shows an effect of 16-bit addition by processing the layers simultaneously.

C. Overall Process

Fig. 3(a) illustrates the overall data processing flow with the following three steps: (1) Multiply the activation and the binarized weight; (2) Convert the multiplied number to our custom bit representation; and (3) Execute the *split-haft addition* with the *presumed full adder* 3D ReRAM.

In BWN, a weight value could be $+1$ or -1; thus, the multiplication with weights only affects the sign bit. The controller of the processor determines the sign of the input after detecting the weight value. The multiplied values are given to the converter in order to support the proposed method, as illustrated in Fig. 3(b).

Our bit representation adjusted the bit numbers from the original fixed-point representation: our custom representation has 1-, 7-, and 24-bit for the sign, integer, and fractional parts, respectively. While the fixed-point can be used without converting, the IEEE-754 representation is processed by our conversion. At first, the sign bit is stored, and then, the mantissa part is shifted leftward or rightward according to the most significant bit of the exponent bits. The last three insignificant bits of the exponential part determine how many the mantissa bits will be shifted. Then '1', which was omitted in the floating-point representation, is appended in front of the shifted mantissa bits. Except for the sign bit and shifted bits, the rest of the bits are filled by 0s to be 32-bit numbers. For negative signed numbers, we follow the 2's complement method in the end. According to the required range for the networks, the bit allocation for the integer and fractional part can be adjusted. Our shifter supports the conversion from the floating-point representation to such a custom representation method.

Our 3D ReRAM is composed of the three layers as a result of the split-half addition method. While the conventional vertical 3D ReRAM consists of stacked 2D layers with upright standing devices, we utilize a modified *vertical cross-point ReRAM* (VRRAM) [15] for our purpose. VRRAM is denser and more cost-effective due to the smaller number of critical fabrication steps, lithography and etch [16]. In the architecture, two devices, which are colored gray in the lump, are placed along the lines between two electrode lines (blue). Each device is located between a pillar (black) and an electrode row (blue). Every two ReRAM rows and two electrode rows are separated from each other, as shown in the subfigures of Fig. 3(c). Furthermore, we modified the structure by splitting the electrode lines to avoid the undesired correlation between the ReRAM lines that share the same electrode line. The approach increases the area slightly, which can be made up of the doubled density of the 3D structure compared to the 2D ReRAM array.

Afterward, the stored numbers are summed up according to our presumed full adder scheme, and each half is computed

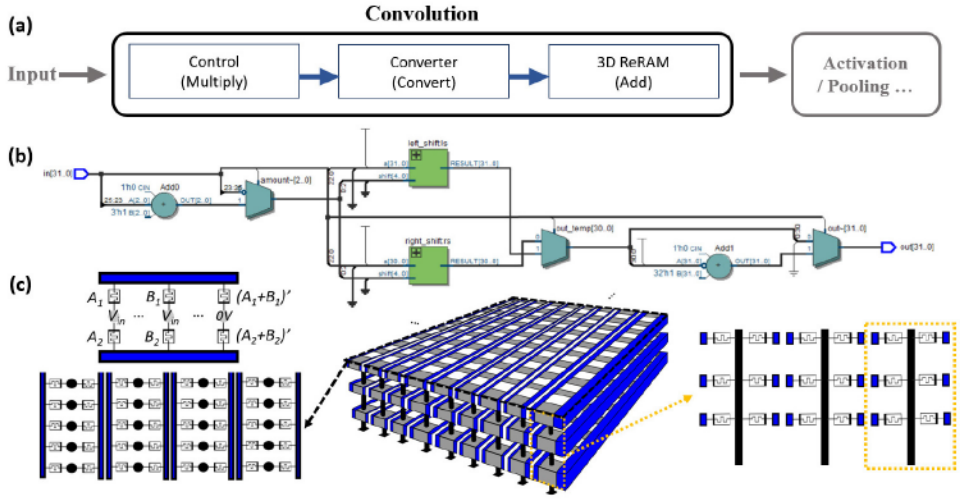

Fig. 3. (a) The overall flow with the converter and the 3D ReRAM architecture and (b) a schematic of the suggested converter to support our method. (c) Top- and side-view details of 3D ReRAM.

as explained previously in 3D ReRAM. The final values of all *multiply-accumulate* (MAC) operations will be read and propagated to the other layer within two cycles. We adopt the designs in [7] for other operations, such as pooling and activation.

IV. Experiment and Result

We simulated the 3D ReRAM circuit by HSPICE to validate the functionality of the presumed adder on the 3D stacked layers. The ReRAM device was modeled by the generalized modeling method [17], [18] in Verilog-A. We trained and tested the MNIST dataset in BWN via PyTorch. After then, we implemented the overall process by using MATLAB and performed convolutions with extracted values from PyTorch. The same results were obtained from both platforms.

We tested the NOR operation with several models by varying parameters. It turned out that the ReRAM device models could be used as long as the applying voltage was considered with R_{ON} and R_{OFF} to match the NOR behavior. When measuring one of the models used to examine the presumed addition scheme, R_{ON} and R_{OFF} were 17.5Ω and 1.75kΩ. The set, reset, and applied voltages were 3V, −1 V, and a value between 2V and 3V, respectively. We then checked the state variable, x, which directly reflects the state of the device among the modeling parameters. When x is almost 0, it corresponds to HRS, logically 0. If the state is close to 1, the device is LRS. Based on [8], we evaluated the device state after every cycle, and distinguished between LRS and HRS by whether the state is under the half state or not. The final results corresponded to the original full adder's output.

Before estimating the effect of the overall process, we first evaluated the BWN impact to discriminate it from the overall effect. Then the effect of the presumed full adder and split-half addition was estimated. By avoiding the use of ADC/DAC that consumes significant power and area, the overhead of peripheral circuits was negligible; therefore, we compared only the crossbar array component for our evaluation. The following data are based on 1-MAC operation since the convolution kernel size is varied according to layers and network models. We compared energy and time of before and after BWN, based on the cost functions of multi-bit addition and multiplication provided by [7] and [19], respectively. Table III summarizes the estimated energy and latency per MAC operation, before and after applying BWN.

TABLE III
Energy and Time Consumption per MAC Operation to Estimate the Effect of BWN, Comparing to FloatPIM [7]

	Energy (pJ)			Latency (μs)		
	Before	*After*	×	*Before*	*After*	×
Fixed-16	0.5029	0.0557	9.03	1.9074	0.2112	9.03
Fixed-32	1.9714	0.1114	17.7	7.4778	0.4224	17.7
bfloat-16	87.022	86.9175	1.00	0.7639	0.3668	2.08
Float-32	265.6	264.6	1.00	4.9743	1.2772	3.89

It is noteworthy that, in energy, the fixed representation shows a more considerable reduction, while the floating-point numbers exhibit a small difference. The reason is that whereas high energy consumption is required on multiplication in the case of fixed-point representation, floating-point numbers need high resources for addition. In other words, fixed-point numbers take more advantage of BWN because applying binarization gets rid of the necessity of multiplication of full-precision bits. As a result, the eye-catching effect in energy can be observed in the fixed-point only. In the time comparison, although fixed-point still shows the significant impact of binarization, a large time-saving effect is achieved in both representation methods because the resource consumption in floating numbers depends less on addition. The presumed and split-half methods don't reduce energy much due to the increase of cells, but the time can be further saved with both schemes: 1.2× and 2×, that is, 2.4× in total. Hence, as considering the number above as well as the results in Table III, 5× to 42.5× time-saving can be expected after the overall convolution processing.

While energy and time largely depend on each device's property, the area based on the number of cells is constant. Thus, we estimated the area in every stage and plotted in

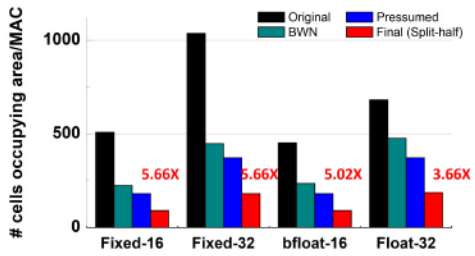

Fig. 4. The number of cells occupying area per MAC in accordance with the schemes and different representation methods.

TABLE IV
THE NUMBER OF TOTAL MACS [20] IN LENET-5 [21], ALEXNET [22], VGG-16 [23], RESNET [1]

	LeNet-5	AlexNet	VGG-16	ResNet-50
# total MACs	2.3M	724M	15.5G	3.9G

Fig. 4. The required cells for fixed precision multiplication and floating-point addition are calculated based on the data from [19] and [7], respectively. BWN leads to about or over $2\times$ saving effect in all cases. The presumed adder with enough bits and split-half addition schemes show $1.17\times$ and $2\times$ decrease effect, respectively. Both schemes give the $2.34\times$ area-saving consequently. Hence, the predicted total effect of converting an original convolution in CNN to our proposed method in BWN further increases: we can save the area from $3.7\times$ to $5.7\times$ according to the bit representation method.

Despite various overwriting options, the overall saving will be proportional to the total MACs. As Table IV presents, advanced networks include a large number of MACs across all the layers. Accordingly, the networks with more MAC operations will benefit more from our methods.

V. CONCLUSION

This brief presents an efficient 3D ReRAM convolution processor design for BWN. In ReRAM-based analog PIM systems, the resource consumed by the additional components has emerged as a limitation. Undesirable properties of devices also hinder the practical use of ReRAM. In this work, we design the convolution processor with two main approaches: 3D architecture with digital PIM and BWN for design simplicity. To be specific, our processor computes the convolution layers with two efficient schemes: *presumed adder* is a simplified full adder design based on two operation cases; and *split-half addition* exploits the structural advantage of 3D by placing split operand bits in different layers and computing them simultaneously. As a result, the original digital PIM convolution exhibits from $3.7\times$ to $5.7\times$ area-saving effect with our proposed processing. $5\times$ to $42.5\times$ time-saving effect is also estimated. This work is applicable to PIM designs entailing the NOR-based full adder.

ACKNOWLEDGMENT

The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied by the AFRL or NSF.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[2] R. Islam *et al.*, "Device and materials requirements for neuromorphic computing," *J. Phys. D, Appl. Phys.*, vol. 52, no. 11, 2019, Art. no. 113001.
[3] L. Chua, "Memristor-the missing circuit element," *IEEE Trans. Circuit Theory*, vol. 18, no. 5, pp. 507–519, Sep. 1971.
[4] D. B. Strukov, G. S. Snider, D. R. Steward, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.
[5] M. Hu, H. Li, Q. Wu, and G. S. Rose, "Hardware realization of BSB recall function using memristor crossbar arrays," in *Proc. IEEE DAC Design Autom.*, 2012, pp. 498–503.
[6] S. Yu, "Neuro-inspired computing with emerging nonvolatile memorys," *Proc. IEEE*, vol. 106, no. 2, pp. 260–285, Feb. 2018.
[7] M. Imani, S. Gupta, Y. Kim, and T. Rosing, "FloatPIM: In-memory acceleration of deep neural network training with high precision," in *proc. ACM/IEEE 46th Annu. Int. Symp. Comput. Archit. (ISCA)*, 2019, pp. 802–815.
[8] S. Kvatinsky *et al.*, "MAGIC—Memristor-aided logic," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 61, no. 11, pp. 895–899, Nov. 2014.
[9] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.
[10] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4107–4115.
[11] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to $+1$ or -1," 2016. [Online]. Available: arXiv:1602.02830.
[12] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 525–542.
[13] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2074–2082.
[14] T. Gale, E. Elsen, and S. Hooker, "The state of sparsity in deep neural networks," 2019. [Online]. Available: arXiv:1902.09574.
[15] H. S. Yoon *et al.*, "Vertical cross-point resistance change memory for ultra-high density non-volatile memory applications," in *Proc. Symp. VLSI Technol.*, 2009, pp. 26–27.
[16] L. Zhang, S. Cosemans, D. J. Wouters, B. Govoreanu, G. Groeseneken, and M. Jurczak, "Analysis of vertical cross-point resistive memory (VRRAM) for 3D RRAM design," in *Proc. 5th IEEE Int. Memory Workshop*, 2013, pp. 155–158.
[17] C. Yakopcic, T. M. Taha, G. Subramanyam, R. E. Pino, and S. Rogers, "A memristor device model," *IEEE Electron Device Lett.*, vol. 32, no. 10, pp. 1436–1438, Oct. 2011.
[18] C. Yakopcic, T. M. Taha, G. Subramanyam, and R. E. Pino, "Memristor SPICE model and crossbar simulation based on devices with nanosecond switching time," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2013, pp. 1–7.
[19] A. Haj-Ali, R. Ben-Hur, N. Wald, and S. Kvatinsky, "Efficient algorithms for in-memory fixed point multiplication using magic," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2018, pp. 1–5.
[20] M. Z. Alom *et al.*, "A state-of-the-art survey on deep learning theory and architectures," *Electronics*, vol. 8, no. 3, p. 292, 2019.
[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: arXiv:1409.1556.