Outstanding Bit Error Tolerance of Resistive RAM-Based Binarized Neural Networks

T Hirtzlin, Marc Bocquet, J.-O Klein, E. Nowak, E. Vianello, Jean-Michel Portal, D. Querlioz

Outstanding Bit Error Tolerance of Resistive RAM-Based Binarized Neural Networks

T. Hirtzlin*, M. Bocquet†, J.-O. Klein*, E. Nowak‡, E. Vianello‡, J.-M. Portal† and D. Querlioz*
*C2N, CNRS, Univ Paris-Sud, Université Paris-Saclay, 91405 Orsay cedex, France
Email: damien.querlioz@u-psud.fr
†Institut Matériaux Microélectronique Nanosciences de Provence, Univ. Aix-Marseille et Toulon, CNRS, France
‡CEA, LETI, Grenoble, France.

***Abstract*—Resistive random access memories (RRAM) are novel nonvolatile memory technologies, which can be embedded at the core of CMOS, and which could be ideal for the in-memory implementation of deep neural networks. A particularly exciting vision is using them for implementing Binarized Neural Networks (BNNs), a class of deep neural networks with a highly reduced memory footprint. The challenge of resistive memory, however, is that they are prone to device variation, which can lead to bit errors. In this work we show that BNNs can tolerate these bit errors to an outstanding level, through simulations of networks on the MNIST and CIFAR10 tasks. If a standard BNN is used, up to 10^{-4} bit error rate can be tolerated with little impact on recognition performance on both MNIST and CIFAR10. We then show that by adapting the training procedure to the fact that the BNN will be operated on error-prone hardware, this tolerance can be extended to a bit error rate of 4×10^{-2}. The requirements for RRAM are therefore a lot less stringent for BNNs than more traditional applications. We show, based on experimental measurements on a RRAM HfO_2 technology, that this result can allow reduce RRAM programming energy by a factor 30.**

I. Introduction

Deep neural networks have made fantastic achievements in recent years [1]. Unfortunately, their high energy consumption limits their use in embedded applications [2], [3]. The in- or near-memory hardware implementation of deep neural networks is widely seen as a solution [2], [4]–[6], as such implementations could avoid entirely the energy cost of the von Neumann bottleneck. This is especially true with the emergence of novel memory technologies such as resistive random access memory (RRAM), phase change memory or spin torque magnetoresistive memory, which have made tremendous progress in recent years [5]. These technologies provide low-area, fast and non-volatile memory cells, which can be embedded at the core of CMOS.

A challenge of the in-memory implementation of neural networks is the high amount of required memory. Binarized Neural Networks (BNNs) have recently emerged as a possibly ideal solution [7], [8]. These neural networks, where the synaptic weights as well as neuron activation are binary values, can achieve near state-of-the-art performance on image recognition tasks, while using only a fraction of the memory used by conventional neural networks [7], [8]. BNNs are therefore an excellent candidate for hardware implementation where the synaptic weights are stored in RRAMs [5], [9].

Nevertheless, despite their outstanding qualities, emerging memories are prone to device variation [4], [10], which can cause bit errors. In conventional applications, this is solved either by relying on error correcting codes [11], or by programming memory cells with high energy pulses that lead to more reliable programming [10]. In this work, based on the experimental measurements of RRAM cells and system level simulations, we investigate the impact of bit errors on in-memory BNNs. We find that BNNs can exhibit outstanding error tolerance, allowing us to avoid these traditional techniques for dealing with RRAM variability.

After presenting the background of the work (section II):

- We show on several tasks that BNNs have a high tolerance to RRAM bit error rate (section III).
- We show that this tolerance can be extended to outstanding levels, if the training process of the BNNs takes into account the fact the neural networks is going to be operated on error-prone hardware (section IV).

II. Background

Binarized Neural Networks are simplifications of conventional neural networks, where synaptic weights, as well as neuron activation values, assume binary values ($+1$ or -1) instead of real values. In these conditions, the equation for neuron activation A:

$$A = f(\sum_i W_i X_i), \tag{1}$$

where X_i are the neuron inputs, W_i the corresponding synaptic weights and f the non-linear activation function, is considerably simplified

$$A = \underset{i}{\mathrm{POPCOUNT}}(XNOR(W_i, X_i)) > T, \tag{2}$$

where POPCOUNT is the function that counts the number of $1s$, and T is a learned threshold.

During training, synaptic weights also assume real weights. The binary weights, equal to the sign of the real weight, are used in both the forward and backward passes, while the real weights are updated by the learning rule [7]. Once training is done, the real weights are no longer needed. BNNs are therefore extremely attractive for hardware implementation of inference operation [12] due to their particularly low memory requirement (one bit per synapse), and due to to the fact that

resource-hungry real multiplications in eq. (1) are replaced by simple binary XNOR gates in eq. (2). An optimal implementation would be with RRAM [5], [13]–[15], as RRAM cells are much more compact than SRAM, and yet non-volatile.

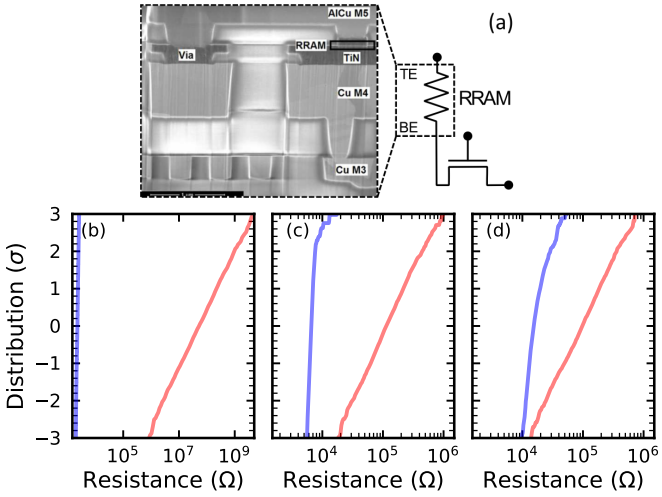

Fig. 1. (a) Transmission electron microscopy image of a RRAM cell embedded in a CMOS process. (b-c) Distribution of the resistance states of RRAM cells in a kilobit array programmed in high resistance state (HRS, red line) and low resistance state (LRS, blue line). The RRAM array is programmed with (b) very strong, (c) strong and (d) weak programming conditions.

TABLE I
RRAM PROGRAMMING CONDITIONS OF FIG. 1(B-D).

Programming condition	**Very strong**	**Strong**	**Weak**
SET compliance current	$600\mu A$	$55\mu A$	$20\mu A$
RESET voltage	$2.5V$	$2.5V$	$1.5V$
Programming time	$100ns$	$100ns$	$100ns$
Bit error rate	$< 10^{-6}$	9.7×10^{-5}	3.3×10^{-2}
Programming energy (SET/RESET)	$120/150pJ$	$11/14pJ$	$4/5pJ$
Cyclability	100	$> 10,000$	$> 10^6$

Nevertheless, the challenge of using RRAMs for in-memory computing is their device variation. Fig. 1(a) presents the transmission electron microscopy image of one of our HfO_2-based RRAM cell, integrated in the backend-of-line of a full CMOS process, on top of the fourth layer of metal, using the same process as [16]. These memory cells can be programmed either in Low Resistance State (LRS, meaning 0) or High Resistance State (HRS, meaning 1). Figs. 1(b-d) show programming statistics of kilobit arrays of such memory cells. Table I summarizes the programming conditions used in Figs. 1(b-d), as well as the corresponding RRAM properties: bit error rate, and cyclability (number of times RRAM devices can be programmed before definitive failure).

In Fig. 1(b), the devices are programmed with "very strong" programming conditions (SET compliance current of $600\mu A$, RESET voltage of $2.5V$, programming time $100ns$). These conditions consume high programming energy, and also lead to device aging, causing low endurance: the RRAM devices cannot be programmed more than 100 times. It is seen in Fig. 1(b) that, despite device variation, the distribution of resistance of the LRS and HRS do not overlap at 3σ, leading to a bit error rate lower than 10^{-6}.

In Fig. 1(c), the devices are programmed with "strong" programming conditions (SET compliance current of $55\mu A$, RESET voltage of $2.5V$, programming time $100ns$). These conditions consume 11 times less programming energy than the previous ones, and also lead to less device aging. The devices can be programmed more than $10,000$ times. The distribution of resistance of the LRS and HRS do not overlap at 3σ, leading to a bit error rate of 9.7×10^{-5}.

In contrast, in Fig. 1(d), the devices are programmed with "weak" programming conditions (SET compliance current of $20\mu A$, RESET voltage of $1.5V$, programming time $100ns$). These conditions consume thirty times less programming energy than the very strong ones, and have very reduced device aging: they can be programmed millions of times The distribution of resistance of the LRS and HRS do overlap significantly, leading to a high bit error rate of 3.3×10^{-2}.

In applications, this issue of device variability can be dealt with several strategies. Either, we can use very strong programming conditions, causing high energy consumption, larger cell area (as the transistor associated with RRAM cells need to be able to drive high currents), and low endurance. Alternatively, we can rely on error correcting codes [11], causing a very significant overhead to implement decoding circuits [17]. Other strategies such as write termination [18], [19] and adaptive programming [20] can also reduce the impact of device variability, but with the cost of strong area overhead. Now, we look how BNNs can deal with the issue of device variation in a simpler fashion. This is especially important as some strategies proposed to enhance bit error tolerance in synaptic weights such as the reliance on sign-magnitude representation [21] do not apply to BNNs.

III. BIT ERROR TOLERANCE WITH TRADITIONAL TRAINING METHOD

To investigate the impact of bit errors due to RRAM device variation in BNNs with synaptic weights stored in RRAMs, we performed multiple BNNs simulations. We first consider a fully connected neural network with two hidden layers of 4096 neurons, a softmax cross entropy loss function and dropout, illustrated in Fig. 2. We trained this neural network on a Nvidia Tesla V100 GPU to perform the traditional MNIST handwritten character recognition task [22] for 1000 epochs. The BNNs were simulated in the python language using the deep learning Tensorflow framework, and an identical training procedure than [7]. Without any bit error, this network achieves a 1.6% error rate on the test dataset.

We then test the neural network again, but introducing random bit errors on the synaptic weights, i.e randomly flipping a fraction $+1$ weights to -1, and of -1 weights to $+1$. Fig 3(a) shows the test recognition rate on MNIST as the

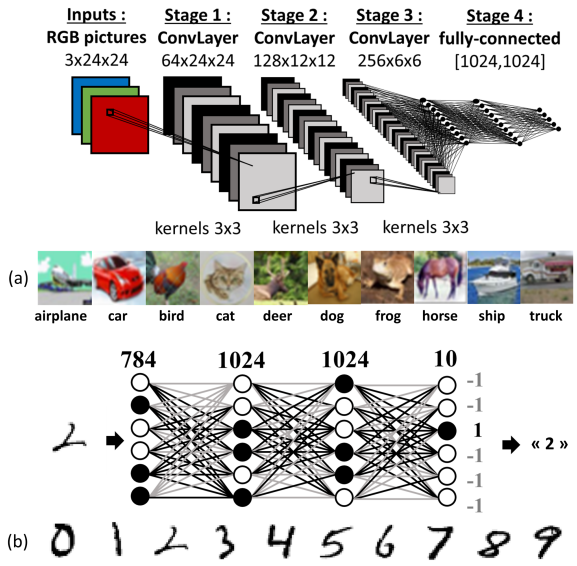

Fig. 2. Binarized Neural Networks considered within this paper. (a) Fully connected network for MNIST handwritten digit recognition. (b) Convolutional neural network for CIFAR10 image recognition.

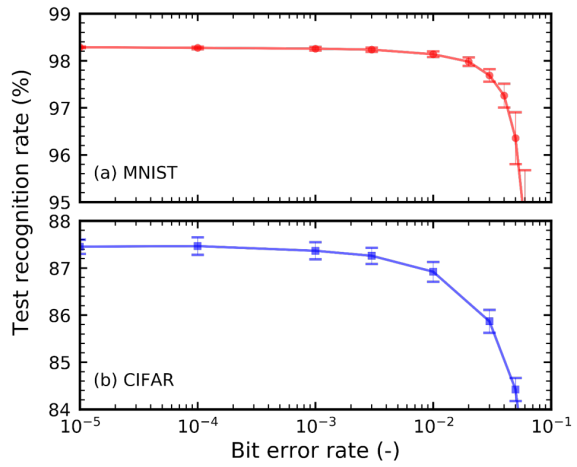

Fig. 3. Recognition rate on the test dataset of (a) the fully connected neural network for MNIST and (b) the convolutional neural network for CIFAR10, as a function of the bit error rate over the weights during inference. The neural networks have been trained in a standard fashion, without weight errors.

function of the weight bit error rate. We see that bit error rate as high as 10^{-3} does not affect the recognition rate. With an error rate of 10^{-2} the recognition rate is only slightly reduced from 1.6% to 1.8%.

To check that this surprising result is not due to the simplicity of the considered neural network, we also studied a deep convolutional neural network, trained on the much more difficult image recognition CIFAR10 task [23]. The structure of the neural network is presented in Fig. 2. Without bit errors, the test recognition rate is 87.6%. All simulations were performed with the Tensorflow deep learning framework [24].

We then perform the simulations including errors. Fig 3(b) shows the test recognition rate on CIFAR10 as the function of the weight bit error rate. Overall, the system is only slightly less robust than in the MNIST case. With an error rate of 10^{-3} the recognition rate is reduced from 87.6% to 87.4%. With an error rate of 10^{-2} it drops to 86.9%.

We have seen in section II that in RRAMs, the choice of programming conditions determines the bit error rate. As we only need ensure a bit error rate of 10^{-4} to avoid any recognition rate degradation in both MNIST and CIFAR10 tasks, we therefore do not need to use "very strong" programming conditions of Fig. 1 (b) and Table I, but can use the strong conditions. This saves programming energy, cell area and enhance the RRAM endurance, with regards to more conventional applications that would require the reliability of the very strong programming conditions.

IV. Adapting the Training Method can Extend the Bit Error Tolerance

We now show that the already high robustness seen in Fig. 3 can be further enhanced if an appropriate training method is used. For this purpose, we retrain the BNNs, but this time including bit errors *during the training process*, and not only during the testing phase. This way, the training process takes into account the fact that the BNN will be implemented on error-prone RRAM-based systems. The devices subject to errors are chosen independently in training and testing phase: the training phase assumes that the RRAM-based systems will have errors, but does not know which devices will be affected.

More precisely, to train the neural network, we added errors at each iteration. The forward-pass is computed with errors on weights, e.g. 10% are changed from the original weights W to W_{error}. During the backward-pass, we reuse the same value of weights W_{error} instead of W to backpropagate through the whole depth of the neural network.

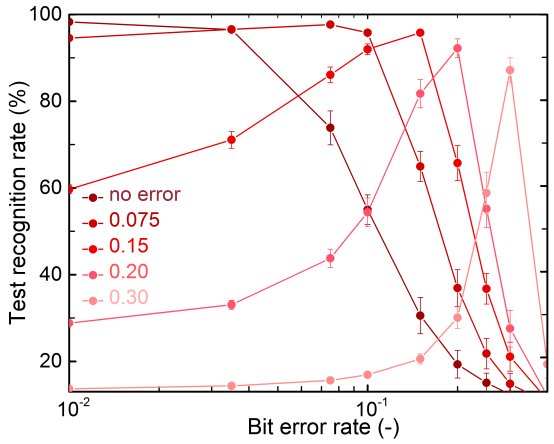

Fig. 4. Recognition rate on the test dataset of the fully connected neural network for MNIST as a function of the bit error rate over the weights during inference. Dark red curve: no weight error considered during training. Other curves: the adapted training was used, each curve corresponding to a different bit error rate on the weights during training.

We now check if this training approach leads to more robust neural networks. We performed the adapted training procedure

with different error rates during training. Fig. 4 shows the test error rate on the MNIST task, as a function of bit error rate during testing, for several error rates during training. We see that these curves exhibit a maximum when error rate during testing matches the error rate during training: the network indeed finds a structure particularly adapted to the number of bit errors. The recognition rate at this maximum can be very high. For a bit error rate of 0.15, the recognition rate is 95.9%. If errors had not been taken into account during training (dark red line), recognition rate would only be 30.4%. This is quite astonishing that we can get the neural network to function so well, with 15% of incorrect weights!

The fact that the recognition rate exhibits a maximum as a function of bit error rate at test time seems counter intuitive. This can however be understood due to the particular properties of BNNs. In these networks, the value of the neurons is binarized, following eq. (2). Having or not having bit errors in the weight W_i can shift the mean value of $\mathrm{POPCOUNT}_i(XNOR(W_i, X_i))$. This effect is very strong in the first layer of the neural network, because in MNIST, the input pixels are mostly binary (black or white). Because of this, there can be a situation where when the bit error rate is shifted, the POPCOUNT value is always below threshold, or always above threshold T. Then, the neuron always has the same value and is useless for classification. For example, for a training bit error rate of 0.3, and no bit error at test time, 48% of the first layer neurons are in this situation. Whereas only 13% of these neurons are in this situation if the 0.3 bit error rate is used both during training and testing.

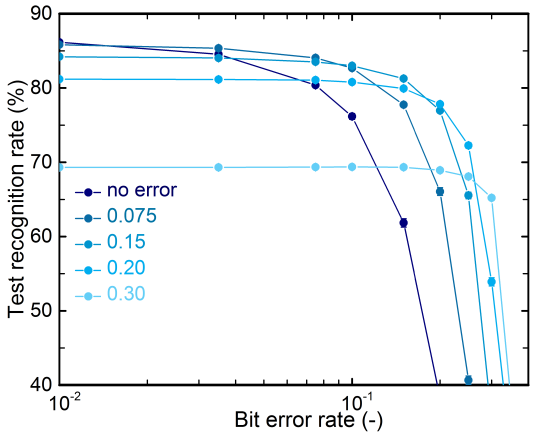

Fig. 5. Recognition rate on the test dataset of the convolutional neural network for CIFAR10 as a function of the bit error rate over the weights during inference. Navy curve: no weight error considered during training. Other curves: the adapted training was used, each curve corresponding to a different bit error rate on the weights during training.

Fig. 5 shows the results of the same study on the CIFAR10 dataset. The results look slightly different than the MNIST case, in that the curves do not exhibit maxima: if the network has been trained with a given bit error rate, having less bit errors during testing is not detrimental.

This can be explained by going back to the principle of BNNs. We observed that when presenting CIFAR10, adding bit errors does not shift significantly the mean values of $\mathrm{POPCOUNT}_i(XNOR(W_i, X_i))$. The difference with MNIST comes from the fact that MNIST images are mostly black and white, whereas the CIFAR10 images feature rich colors. Because of that, the error-induced shift of the mean value of $\mathrm{POPCOUNT}_i(XNOR(W_i, X_i))$ is significant in the case of the first layer of a neural network during MNIST, but much less significant on CIFAR10.

Once again, using this procedure, extremely high amount of bit errors can be tolerated. Bit error rates up to 4×10^{-2} do not affect the recognition rate. For a bit error rate of 0.15, the recognition rate is 81.5%, instead of 62.2% if errors had not been taken into account during training (navy line).

When using RRAM devices, using the adapted training procedure therefore allows us using directly the "weak" programming conditions of Fig. 1 and Table I, despite its high bit error rate. This can allow us to benefit from its low programming energy, cell area and high endurance of more than one million cycles.

V. Conclusion

The in-memory implementation of Binarized Neural Networks (BNNs) with emerging memories such as RRAM is an exciting road for ultralow energy embedded artificial intelligence. A typical challenge of emerging memories is their device variation, which can lead to bit errors. In this work, we have seen that BNNs can tolerate a very high number of bit errors. The BNN is trained in a conventional manner, up to 10^{-4} bit error rate on the synaptic weights can be tolerated on both MNIST and CIFAR task. If the BNN, was trained with a specific procedure taking into account the fact that it will be operated on error-prone hardware, up 4×10^{-2} bit error rate can be tolerated!

These results can have strong impact on emerging memory design and optimization toward neural network implementation. If one accepts to increase device variability and bit error rate, we saw that is is possible to increase the programming energy efficiency by a factor of thirty, decrease the area and increase the endurance of RRAM cells to more than one million cycles. This can significantly enhance their benefits. The results of this paper also highlight the robustness of neural networks, even in the highly digital BNN form, and their adaptability to the constrains of hardware.

Acknowledgment

This work was supported by the European Research Council Starting Grant NANOINFER (715872).

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[2] Editorial, "Big data needs a hardware revolution," *Nature*, vol. 554, no. 7691, p. 145, Feb. 2018.

[3] A. Suleiman, Y.-H. Chen, J. Emer, and V. Sze, "Towards closing the energy gap between hog and cnn features for embedded vision," in *Proc. ISCAS*. IEEE, 2017, pp. 1–4.

[4] D. Ielmini and H.-S. P. Wong, "In-memory computing with resistive switching devices," *Nature Electronics*, vol. 1, no. 6, p. 333, 2018.

[5] S. Yu, "Neuro-inspired computing with emerging nonvolatile memorys," *Proc. IEEE*, vol. 106, no. 2, pp. 260–285, 2018.

[6] D. Querlioz, O. Bichler, A. F. Vincent, and C. Gamrat, "Bioinspired programming of memory devices for implementing an inference engine," *Proc. IEEE*, vol. 103, no. 8, pp. 1398–1416, 2015.

[7] 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," *arXiv preprint arXiv:1602.02830*, 2016.

[8] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *Proc. ECCV*. Springer, 2016, pp. 525–542.

[9] M. Bocquet, T. Hirztlin, J.-O. Klein, E. Nowak, E. Vianello, J.-M. Portal, and D. Querlioz, "In-memory and error-immune differential rram implementation of binarized deep neural networks," in *IEDM Tech. Dig.* IEEE, 2018, p. 20.6.1.

[10] D. R. B. Ly, A. Grossi, C. Fenouillet-Beranger, E. Nowak, D. Querlioz, and E. Vianello, "Role of synaptic variability in resistive memory-based spiking neural networks with unsupervised learning," *J. Phys. D: Applied Physics*, 2018.

[11] F. Zhang, D. Fan, Y. Duan, J. Li, C. Fang, Y. Li, X. Han, L. Dai, C. Chen, J. Bi *et al.*, "A 130nm 1mb hfox embedded rram macro using self-adaptive peripheral circuit system techniques for 1.6 x work temperature range," in *Proc. A-SSCC*. IEEE, 2017, pp. 173–176.

[12] K. Ando, K. Ueyoshi, K. Orimo, H. Yonekawa, S. Sato, H. Nakahara, M. Ikebe, T. Asai, S. Takamaeda-Yamazaki, T. Kuroda *et al.*, "Brein memory: A 13-layer 4.2 k neuron/0.8 m synapse binary/ternary reconfigurable in-memory deep neural network accelerator in 65 nm cmos," in *Proc. VLSI Symp. on Circuits*. IEEE, 2017, pp. C24–C25.

[13] X. Sun, X. Peng, P.-Y. Chen, R. Liu, J.-s. Seo, and S. Yu, "Fully parallel rram synaptic array for implementing binary neural network with (+ 1,-1) weights and (+ 1, 0) neurons," in *Proc. ASP-DAC*. IEEE Press, 2018, pp. 574–579.

[14] X. Sun, S. Yin, X. Peng, R. Liu, J.-s. Seo, and S. Yu, "Xnor-rram: A scalable and parallel resistive synaptic architecture for binary neural networks," *algorithms*, vol. 2, p. 3, 2018.

[15] T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang, "Binary convolutional neural network on rram," in *Proc. ASP-DAC*. IEEE, 2017, pp. 782–787.

[16] A. Grossi, E. Nowak, C. Zambelli, C. Pellissier, S. Bernasconi, G. Cibrario, K. El Hajjam, R. Crochemore, J. Nodin, P. Olivo *et al.*, "Fundamental variability limits of filament-based rram," in *IEDM Tech. Dig.* IEEE, 2016, pp. 4–7.

[17] S. Gregori, A. Cabrini, O. Khouri, and G. Torelli, "On-chip error correcting techniques for new-generation flash memories," *Proc. IEEE*, vol. 91, no. 4, pp. 602–616, 2003.

[18] M.-F. Chang, J.-J. Wu, T.-F. Chien, Y.-C. Liu, T.-C. Yang, W.-C. Shen, Y.-C. King, C.-J. Lin, K.-F. Lin, Y.-D. Chih *et al.*, "19.4 embedded 1mb reram in 28nm cmos with 0.27-to-1v read using swing-sample-and-couple sense amplifier and self-boost-write-termination scheme," in *Proc. ISSCC*. IEEE, 2014, pp. 332–333.

[19] F. Su, W.-H. Chen, L. Xia, C.-P. Lo, T. Tang, Z. Wang, K.-H. Hsu, M. Cheng, J.-Y. Li, Y. Xie *et al.*, "A 462gops/j rram-based nonvolatile intelligent processor for energy harvesting ioe system featuring nonvolatile logics and processing-in-memory," in *Proc. Symp. VLSI Technol.* IEEE, 2017, pp. T260–T261.

[20] G. Sassine, C. Nail, L. Tillie, D. A. Robayo, A. Levisse, C. Cagli, K. E. Hajjam, J.-F. Nodin, E. Vianello, M. Bernard *et al.*, "Sub-pj consumption and short latency time in rram arrays for high endurance applications," in *Proc. IRPS*. IEEE, 2018, pp. P–MY.

[21] P. N. Whatmough, S. K. Lee, D. Brooks, and G.-Y. Wei, "Dnn engine: A 28-nm timing-error tolerant sparse deep neural network processor for iot applications," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 9, pp. 2722–2731, 2018.

[22] 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, 1998.

[23] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[24] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.