Redundancy and analogue slicing for precise in-memory machine learning - Part II: Applications and benchmark

Giacomo Pedretti, *Member, IEEE,* Piergiulio Mannocci, *Student Member, IEEE,* Can Li, Zhong Sun, John Paul Strachan, and Daniele Ielmini, *Fellow, IEEE*

***Abstract*—In-memory computing (IMC) is attracting interest for accelerating data-intensive computing tasks, such as artificial intelligence (AI), machine learning (ML) and scientific calculus. IMC is typically conducted in the analogue domain in crosspoint arrays of resistive random access memory (RRAM) devices or memristors. However, the precision of analogue operations can be hindered by various sources of noise, such as the non-linearity of the circuit components and the programming variations due to stuck devices and stochastic switching. Here we demonstrate high-precision IMC by a custom program-verify algorithm that uses redundancy to limit the impact of stuck devices and analogue slicing to encode the analogue programming error in a separate memory cell. The Pagerank problem, consisting of the calculation of the principal eigenvector, is shown as a reference problem, adopting a fully-integrated RRAM circuit. We extend these results to also include a convolutional neural network (CNN). We demonstrate a computing accuracy of 6.7 equivalent number of bits (ENOB). Finally, we compare our results to the solution of the same problem by an SRAM-based IMC, showcasing an advantage for the RRAM implementation in terms of energy efficiency and scaling.**

***Index Terms*—In-memory computing, memristor, RRAM, pagerank, memory reliability, neural networks**

I. Introduction

SEVERAL modern computing workloads, such as artificial intelligence (AI) and scientific computing algorithms, require processing a large amount of information in parallel. The von Neumann architecture [1], which forms the fundation for most computing system, is rather inefficient for data-intensive computing since the memory and compute units are separated, thus causing a large amount of time and energy for data movement [2]. Several advancements in computing systems have attempted to mitigate this memory bottleneck, such as implementing large scale parallel architectures (such as in graphic processing units, GPU) or customized systems (such as in the tensor processing unit, TPU) [3]. On the other hand, in-memory computing (IMC) [4] can radically change the computing architecture to perform parallel algebraic operations directly within the memory, usually consisting of emerging non-volatile memories (NVM) such as resistive switching memories (RRAM) [5], or memristors. RRAM can be arranged in compact crosspoint arrays [6] and be programmed in an analogue fashion to represent an $N \times N$ matrix G of arbitrary entries directly within the memory. Then, by applying a voltage vector V as input on the columns and grounding the rows, the current flowing in each row is given by $I_j = \sum_{i=0}^{i=N} G_{ij} V_i$, where V_i is the applied voltage vector, G_{ij} is the conductance matrix, and I_j is the output current. This is equivalent to performing matrix vector multiplication (MVM), which has been demonstrated for neural networks operations [7]–[9], image processing [10], [11], optimization [12]–[14] and the iterative solution of linear equations [15], [16]. Analogue in-memory circuits to solve linear equations in one step have also been recently proposed [17]–[19]. Integrated systems comprising conventional CMOS sensing/routing circuitry and memristors for accelerating computing tasks, have already been developed [20]–[22] demonstrating experimentally the superiority of such systems. Nevertheless, fine tuning of analogue conductance in memristors is still an open challenge [23]. In fact, memristor suffers from different types of variability which is due to the structure of the device itself, where physical properties of the material, such as its defect configuration, are used to obtain different conductance levels, and are inevitably stochastic. Several device and system level techniques have been proposed to reduce memory variability such as redundancy [24], bit-slicing [25] and mixed precision representation [15], [26].

In a companion paper [27] we proposed a technique combining the benefits of redundancy and mixed precision techniques, encoding the error after programming in a separate memory block, a procedure we dubbed analogue slicing (AS). Here, we apply redundancy and AS to the solution of the Pagerank algorithm [28] iteratively executed on an integrated circuit (IC) [22] comprising three 64×64 memristor arrays and sensing/routing circuitry. After demonstrating the beneficial effect of redundancy and analogue slicing, with an error approaching the noise floor of our system, we extend the results to convolutional neural network (CNN) inference [22] and then benchmark our results in comparison to commercial static random access memory (SRAM) [29]. The results suggest the need for such programming strategies to increase the accuracy of IMC, and offers a technology-agnostic solution for mitigating conductance inherited variations.In fact the same procedure can be used with other types of resistive memories,

G. Pedretti is with Politecnico di Milano, Milano, Italy and Hewlett Packard Labs, Milpitas, CA, USA.

P. Mannocci and D. Ielmini are with Politecnico di Milano, Milano, Italy

C. Li is with the University of Hong Kong (HKU), Hong Kong SAR, China and Hewlett Packard Labs, Milipitas, CA, USA

Z. Sun is with Peking University (PKU), Beijing, China and Politecnico di Milano, Milano, Italy

J. P. Strachan is with Hewlett Packard Labs, Milpitas (CA), USA

Correspondance should be addressed to giacomo.pedretti@polimi.it, john-paul.strachan@hpe.com and daniele.ielmini@polimi.it.

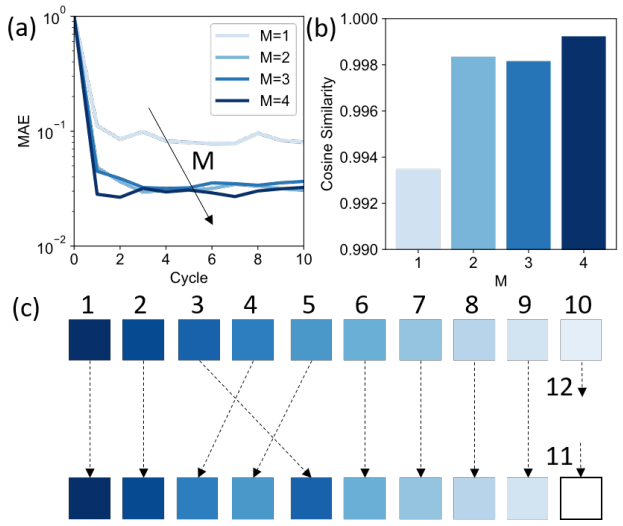

Fig. 1. Effect of redundancy. (a) MAE in computing eigenvector as function of cycles for different M. Note that the MAE saturates after a few cycles due to the inherent speed of convergence of power iteration (b) Cosine similarity as function of M. MAE and cosine similarity shows that even with $M = 2$ the error can be notably reduced

Correlation between the correct ranking (top) and measured ranking (bottom).

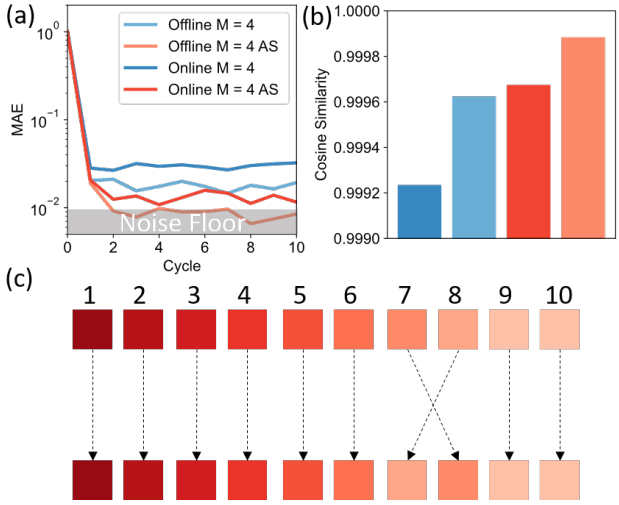

Fig. 2. Effect of AS. (a) MAE in computing eigenvector as function of cycles for online MVM and offline MVM with or without AS technique. (b) Cosine similarity for the 4 computing techniques. (c) Correlation between the correct ranking (top) and measured ranking (bottom).

such as Phase Change Memories (PCM)

II. Programming Strategies Evaluation

To evaluate the effect of redundancy and analogue slicing (AS) on the PageRank solution, we selected the same problem of the companion paper [27], which corresponds to the PageRank of 32 web pages. The goal is to compute the ranking of web pages from the most authoritative to the least authoritative [18]. As previously reported, this can be done by computing the principal eigenvector of a 32×32 stochastic matrix [18] representing the graph problem. The stochastic matrix is created by dividing each column of the boolean link matrix representing connections between each webpage, for the sum over the column itself. In this way the leading eigenvalue is always known and equal to 1. The matrix was programmed to the conductance of the memristors in the IC as previously reported [27].

A. Effect of redundancy

Redundancy is a programming techniques which can be useful both for decreasing the variability of the programmed conductance, and to recover non-ideal states such as stuck devices. In a PageRank solution a stuck device can change completely the graph connection making the results unreliable. Redundancy typically consists in programming multiple devices with the same target conductance [24], however here we used a previously developed redundancy aware program and verify algorithm [27] where multiple devices are programmed while comparing their average value to the target conductance. We programmed the PageRank problem with different redundancy values M and Fig. 1a shows the measured mean absolute error (MAE), namely:

$$MAE = \frac{\sum_{i=1}^{N} |x_i - x_i'|}{N} \tag{1}$$

where x is the ideal solution and x' is the measured solution in computing the eigenvector of the target matrix as a function of iteration cycles for different M, the number of devices used to represent a matrix entry. It is evident that a redundancy factor of $M = 2$ is enough to recover the accuracy of the solution by reducing the MAE. This can also be seen by the cosine similarity distance plot of Fig. 1b, which is computed as:

$$\text{cosim}(x, x') = \frac{x \cdot x'}{||x||||x'||} \tag{2}$$

These results are possible thanks to the RA-PV algorithm presented in the companion paper [27], which can compensate a stuck device by the conductance of the other $M - 1$ devices. For instance, a stuck on device can be compensated by decreasing the conductance of the other devices, while a stuck off device can be compensated by increasing the conductance of other devices. Fig. 1c shows the comparison between the correct solution (top) and the measured solution, namely the identifier number of the first 10 ranked pages. Some relatively small errors can still be seen, which are due to eigenvector elements being approximately the same, thus resulting in wrong ranking even in the presence of a high precision in the mapped conductance values. We conclude that redundancy is mainly useful to compensate major programming errors, such as stuck devices, which is confirmed by the fact that a redundancy $M = 2$ is generally sufficient to improve the accuracy of the results. However, to further reduce the variability-induced error, more precise encoding techniques should be adopted.

B. Effect of slicing

To further reduce the error, AS can be applied. Here, the first crosspoint array is programmed to map a certain target matrix G_p, resulting in the conductance matrix G_r, differing from the target matrix by an error matrix $G_\epsilon = G_p - G_r$. The error matrix is then multiplied by a suitable gain and programmed in two separate crosspoints, representing the positive and negative entries. The gain is chosen in such a way that G_ϵ can cover the full dynamic range of the conductance window. AS can be combined with redundancy, to allow for the compensation of major errors such as stuck devices.

Fig. 2a shows the MAE as function of iteration cycles for $M = 4$ with and without slicing. Two different approaches in conducting the MVM are compared, namely *online computation*, where the MVM is computed physically within the crosspoint array, and *offline computation*, where the MVM is computed *ex situ* based on the measured G and applied voltages [9]. The difference between online and offline MVM is that errors due to parasitic resistance, also known as IR drops, and peripheral noise are suppressed in the case of offline computation. The results show that AS effectively reduces the MAE until the point where it hits the noise floor of our system, corresponding to $MAE = 0.09$, which is mainly caused by the peripheral readout circuit [27]. This means that we reach the ultimate limit in programming precision, thus programming the conductance to an even higher level of accuracy would not be detectable by our system. The precision may be further enhanced by employing mixed precision [15] and other architectural [25] techniques. Fig. 2b shows the cosine similarity of the computed eigenvectors compared to the ideal values, confirming the increased accuracy by slicing. Fig. 2c shows the comparison between the correct solution (top) and the measured solution, demonstrating that the first 10 webpages are ranked correctly except for a flip between 7^{th} and 8^{th} pages due to nearly identical eigenvector entries. These results support the benefit of redundancy and AS techniques in improving the accuracy of MVM.

C. Extension to CNN

To assess the generality of the proposed programming techniques, we studied the effect on the accuracy of inference in a CNN. Fig. 3a shows the schematic of a simple CNN for the classification of the MNIST dataset [22]. The CNN consists of a single convolutional layer with 7 kernels of size 20×20, a max pooling layer of size 5×5 and a 112×10 classification layer. The network was trained offline and achieves an inference accuracy of 98.15% with software floating-point precision. We programmed the trained weights as conductance in two separate crosspoint arrays for the convolutional and classification weights, respectively, without any redundancy ($R = 1$) or AS. Fig. 3b shows the probability distribution of the programmed conductance error, namely $G_{error} = G_{target} - G_{prog}$, where G_{target} is the target weight and G_{prog} is the measured weight. The figure also shows the normal fitting of the distribution, corresponding to a mean value $\mu = 4\,\mu S$ and a standard deviation $\sigma = 8\,\mu S$. The inference accuracy was experimentally evaluated online

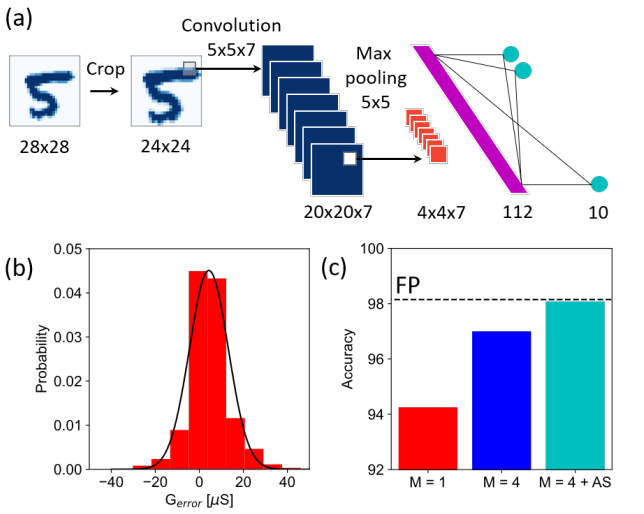

Fig. 3. (a) Schematic illustration of the CNN used. After cropping the MNIST dataset to 24×24 images, a convolutional layer is applied. Its output is fed to a classification layer after a max pooling operation. (b) Probability distribution of the programming error for the CNN weights fitted with a normal distribution. (c) Inference accuracy for different programming techniques.

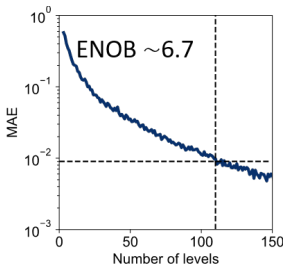

Fig. 4. Simulated MAE as a function of discretization levels and corresponding measured ENOB in hardware.

by applying 10000 test samples, resulting in a classification accuracy of 94.25%, as also shown in Fig. 3c. The relatively low accuracy can be explained by the insufficient precision of the mapped weights which are affected by the programming errors in Fig. 3b. To improve the precision, we simulated MVM in a larger set of crosspoint arrays with a redundancy $M = 4$, assuming the same distribution of Fig. 3b. Fig. 3c shows the simulation results, indicating that the inference accuracy can be increased to 97% with a redundancy $M = 4$. We also combined redundancy and analogue slicing, which further improves the testing accuracy to 98.08%. Similarly to the eigenvector calculation, these results indicate that both redundancy and AS are generally needed to enable the MVM computation precision to approach floating-point. The results also demonstrate the universality of redundancy and AS programming techniques across various IMC applications.

III. Benchmark with SRAM technology

While redundancy and AS are promising solutions to improve the precision of IMC, they come at the cost of a larger circuit area and energy consumption. For instance, assuming a redundancy factor $M = 4$ combined with AS, the number of

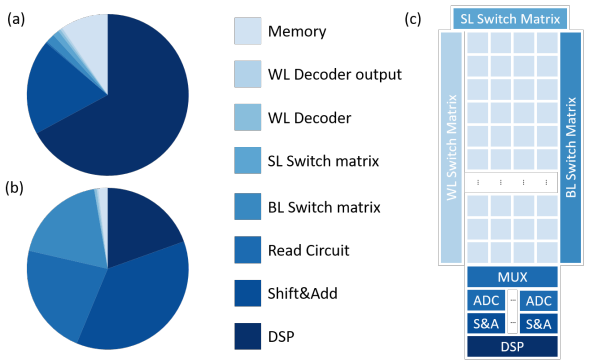

Fig. 5. Area (a) and energy (b) breakdown among the circuit peripherals for the memristor implementation with AS, R=4. Read circuit, Shift&Add and DSP are shared between tiles belonging to a R/AS set, and account for the majority of both area and energy consumption. (c) Tall tiled architecture for implementing redundancy and AS with small circuitry overhead.

memory devices used to represent a matrix entry for PageRank acceleration is 12. To assess the impact of the increased area and energy on the performance of IMC, we conducted high-level circuit simulations of our IMC hardware compared to an SRAM implementation for the computation of PageRank. The simulations were conducted by using a publicly available tool [29], adapted to the case of eigenvector calculation.

First, we evaluated the equivalent number of bits ($ENOB$) that is reached in our IMC system based on the error in the PageRank calculation. To this purpose, we simulated the solution of PageRank for an increasing number of discrete levels of the input, output and matrix entries. Fig. 4 shows the resulting MAE for Pagerank solution as a function of the number of levels. The MAE in the figure reaches the experimentally measured value $MAE = 0.09$ in Fig. 2 for 108 discrete levels, corresponding to $ENOB = 6.7$. Note that the $ENOB$ of the integrated analogue-to-digital converter (ADC) in our IMC chip was less than 6, however the effective precision was increased by means of a shift-and-add MVM technique with 8 bit precision [25], thus the overall precision is only limited by the memristor variation and the peripheral noise rather than from design limitations. We therefore compared out IMC system to an equivalent SRAM-based computing with a precision of 7 bit.

Figure 5 shows the area (a) and energy (b) breakdown for the memristive implementation of the accelerator. Notably, the memory overhead is relatively small while vertical peripherals responsibile for sensing the current and post-processing the digital information such as the read circuit, shift and add unit and DSP are responsible for most of the energy and area occupation. For this reason, since 12 devices are needed to represent the information for redundancy and AS, we adopted a 'tall' architecture as shown in Fig. 5(c) where the vertical units (read circuit and DSP) are shared among sets of different tiles. Each tile is then used to implement a redundant unit, so that in a given set with shared-peripherals R tiles map the MSB values, R tiles map the positive error matrix and R tiles map the negative error matrix according to the AS schematic. Note that only the energy consumption for calculating the eigenvector was considered here, since the write operation is infrequent and typically needed only if a new node (or page) is added to the graph.

After optimizing the memristive IMC architecture, we estimate its performance compared with SRAM implementation in $32\,\mathrm{nm}$ technology node [29], which roughly corresponds to the dimension of our fabricated memristor, whose size is 40 nm. The system is clocked at $f_{CLK} = 2\,\mathrm{GHz}$ and a conductance window from $1\,\mathrm{to}\,100\,\mu\mathrm{S}$ is considered for RRAM arrays. Note that the conductance could be optimized, as there exists a trade-off between reducing the errors due to IR drops and limiting the slow down due to charging the BL [23].

Fig. 6 shows a comparison of the area occupation (a), latency (b), energy consumption (c) and energy efficiency evaluated in TOPS/W (d) for SRAM and RRAM implementations. The benchmark indicates a 1.5×, 8× and 190× reduction in area, improvements in latency and energy consumption, respectively, resulting in an efficiency of 5 TOPS/W corresponding to a 10× increase compared to SRAM. While higher efficiency has been already shown with memristive IMC [21], these results suggest that even with an unreliable memristive device, an efficiency larger than an equivalent precision implementation with SRAM can be reached, in fact accelerators based on SRAM still suffer from a relatively large energy consumption due to the cost of retaining the data in a volatile memory. Finally, we estimate the scaling behavior of the accelerators by studying the efficiency as a function of the number N of webpages in a PageRank problem, i.e. the number of rows/columns of the matrix for the eigenvector calculation. Fig. 7 shows a comparison of the efficiency as a function of N for SRAM and RRAM implementation. SRAM efficiency decreases with N almost linearly, due to increased energy consumption and latency in accessing several memory cells in parallel, while RRAM accuracy reaches a maximum for $N \sim 200$, corresponding to the maximum parallelism that can be achieved. In fact, at a fixed number of eigenvector calculations per unit time, the solution of a larger problem results in increased throughput since typically the number of equivalent operations required in digital systems scales polynomially with the problem size. However, the dimension of the problem can be scaled until a certain point where implementing it directly on crosspoint array is not desirable anymore due to increased overhead of peripherals. Thus, for larger problems, an optimization of the architecture should be performed [25].

IV. Conclusion

This work addresses the computational impact of the programming strategies, namely redundancy and AS, in a fully integrated circuit comprising CMOS routing/sensing peripherals with memristor arrays. Redundancy and AS were experimentally shown to enable a near-optimal ranking in PageRank problem and recover the inference accuracy in a CNN close to that of floating point. Despite the overhead in area and energy consumption, the memristive IMC outperforms

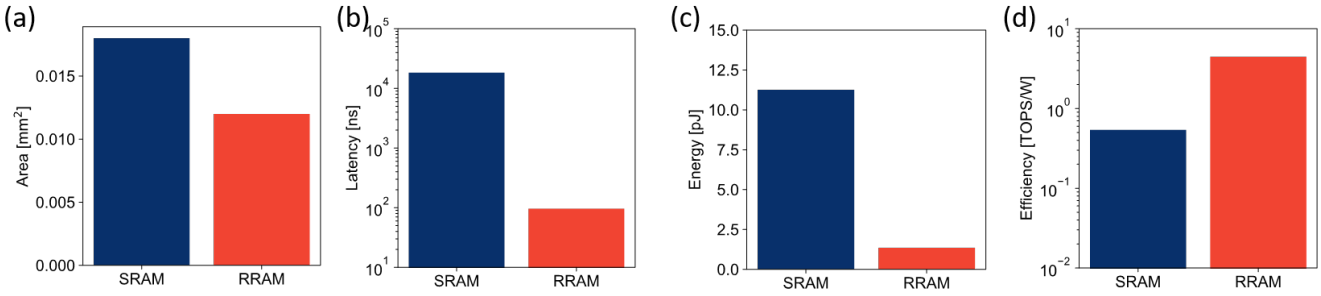

Fig. 6. Area (a), latency (b), energy (c) and energy efficiency (d) for this work implementation with redundant and analogue sliced RRAM compared with a same accuracy (i.e. 7bit) SRAM.

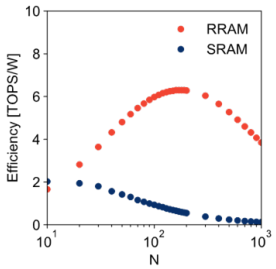

Fig. 7. Energy efficiency as function of the problem size N for computing eigenvectors.

a commercial SRAM implementation. The results support our novel redundancy and precision schemes developed in the memristor matrix representations to enable algebraic and artificial intelligence accelerators of high accuracy.

ACKNOWLEDGMENT

This work was supported in part by the European Research Council (grant ERC-2014-CoG-648635-RESCUE and grant ERC-2018-PoC- 842472-CIRCUS).

REFERENCES

[1] J. von Neumann, "First Draft of a Report on the EDVAC," 1945.

[2] M. A. Zidan, J. P. Strachan, and W. D. Lu, "The future of electronics based on memristive systems," *Nature Electronics*, vol. 1, no. 1, pp. 22–29, Jan. 2018. [Online]. Available: http://www.nature.com/articles/s41928-017-0006-8

[3] N. P. Jouppi, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, C. Young, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, N. Patil, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Patterson, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, G. Agrawal, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, R. Bajwa, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, S. Bates, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, D. H. Yoon, S. Bhatia, and N. Boden, "In-Datacenter Performance Analysis of a Tensor Processing Unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture - ISCA '17*. Toronto, ON, Canada: ACM Press, 2017, pp. 1–12. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3079856.3080246

[4] D. Ielmini and H.-S. P. Wong, "In-memory computing with resistive switching devices," *Nature Electronics*, vol. 1, no. 6, pp. 333–343, Jun. 2018. [Online]. Available: http://www.nature.com/articles/s41928-018-0092-2

[5] D. Ielmini, "Resistive switching memories based on metal oxides: mechanisms, reliability and scaling," *Semiconductor Science and Technology*, vol. 31, no. 6, p. 063002, Jun. 2016. [Online]. Available: http://stacks.iop.org/0268-1242/31/i=6/a=063002?key=crossref.ba6cab0bca4179e152c380f4045bc2b1

[6] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nature Nanotechnology*, vol. 8, no. 1, pp. 13–24, Jan. 2013. [Online]. Available: http://www.nature.com/articles/nnano.2012.240

[7] M. Hu, C. E. Graves, C. Li, Y. Li, N. Ge, E. Montgomery, N. Davila, H. Jiang, R. S. Williams, J. J. Yang, Q. Xia, and J. P. Strachan, "Memristor-Based Analog Computation and Neural Network Classification with a Dot Product Engine," *Advanced Materials*, vol. 30, no. 9, p. 1705914, Mar. 2018. [Online]. Available: http://doi.wiley.com/10.1002/adma.201705914

[8] C. Li, D. Belkin, Y. Li, P. Yan, M. Hu, N. Ge, H. Jiang, E. Montgomery, P. Lin, Z. Wang, W. Song, J. P. Strachan, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, and Q. Xia, "Efficient and self-adaptive in-situ learning in multilayer memristor neural networks," *Nature Communications*, vol. 9, no. 1, p. 2385, Dec. 2018. [Online]. Available: http://www.nature.com/articles/s41467-018-04484-2

[9] S. Ambrogio, P. Narayanan, H. Tsai, R. M. Shelby, I. Boybat, C. di Nolfo, S. Sidler, M. Giordano, M. Bodini, N. C. P. Farinha, B. Killeen, C. Cheng, Y. Jaoudi, and G. W. Burr, "Equivalent-accuracy accelerated neural-network training using analogue memory," *Nature*, vol. 558, no. 7708, pp. 60–67, Jun. 2018. [Online]. Available: http://www.nature.com/articles/s41586-018-0180-5

[10] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves, Z. Li, J. P. Strachan, P. Lin, Z. Wang, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, and Q. Xia, "Analogue signal and image processing with large memristor crossbars," *Nature Electronics*, vol. 1, no. 1, pp. 52–59, Jan. 2018. [Online]. Available: http://www.nature.com/articles/s41928-017-0002-z

[11] P. M. Sheridan, F. Cai, C. Du, W. Ma, Z. Zhang, and W. D. Lu, "Sparse coding with memristor networks," *Nature Nanotechnology*, vol. 12, no. 8, pp. 784–789, Aug. 2017. [Online]. Available: http://www.nature.com/articles/nnano.2017.83

[12] M. R. Mahmoodi, H. Kim, Z. Fahimi, H. Nili, L. Sedov, V. Polishchuk, and D. B. Strukov, "An Analog Neuro-Optimizer with Adaptable Annealing Based on 64x64 0T1R Crossbar Circuit," in *2019 IEEE International Electron Devices Meeting (IEDM)*. San Francisco, CA: IEEE, 2019, pp. 14.7.1–14.7.4.

[13] F. Cai, S. Kumar, T. Van Vaerenbergh, X. Sheng, R. Liu, C. Li, Z. Liu, M. Foltin, S. Yu, Q. Xia, J. J. Yang, R. Beausoleil, W. D. Lu, and J. P. Strachan, "Power-efficient combinatorial optimization using intrinsic noise in memristor Hopfield neural networks," *Nature Electronics*, Jul. 2020. [Online]. Available: http://www.nature.com/articles/s41928-020-0436-6

[14] G. Pedretti, P. Mannocci, S. Hashemkhani, V. Milo, O. Melnic, E. Chicca, and D. Ielmini, "A Spiking Recurrent Neural Network With Phase-Change Memory Neurons and Synapses for the Accelerated Solution of Constraint Satisfaction Problems," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 6, no. 1, pp. 89–97, Jun. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9086758/

[15] M. Le Gallo, A. Sebastian, R. Mathis, M. Manica, H. Giefers, T. Tuma, C. Bekas, A. Curioni, and E. Eleftheriou, "Mixed-precision in-memory computing," *Nature Electronics*, vol. 1, no. 4, pp. 246–253, Apr. 2018. [Online]. Available: http://www.nature.com/articles/s41928-018-0054-8

[16] M. A. Zidan, Y. Jeong, J. Lee, B. Chen, S. Huang, M. J. Kushner, and W. D. Lu, "A general memristor-based partial differential equation solver," *Nature Electronics*, vol. 1, no. 7, pp. 411–420, Jul. 2018. [Online]. Available: http://www.nature.com/articles/s41928-018-0100-6

[17] Z. Sun, G. Pedretti, E. Ambrosi, A. Bricalli, W. Wang, and D. Ielmini, "Solving matrix equations in one step with cross-point resistive arrays," *Proceedings of the National Academy of Sciences*, vol. 116, no. 10, pp. 4123–4128, Mar. 2019. [Online]. Available: http://www.pnas.org/lookup/doi/10.1073/pnas.1815682116

[18] Z. Sun, E. Ambrosi, G. Pedretti, A. Bricalli, and D. Ielmini, "In-Memory PageRank Accelerator With a Cross-Point Array of Resistive Memories," *IEEE Transactions on Electron Devices*, vol. 67, no. 4, pp. 1466–1470, Apr. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8982173/

[19] Z. Sun, G. Pedretti, A. Bricalli, and D. Ielmini, "One-step regression and classification with cross-point resistive memory arrays," *Science Advances*, vol. 6, no. 5, p. eaay2378, Jan. 2020. [Online]. Available: https://advances.sciencemag.org/lookup/doi/10.1126/sciadv.aay2378

[20] F. Cai, J. M. Correll, S. H. Lee, Y. Lim, V. Bothra, Z. Zhang, M. P. Flynn, and W. D. Lu, "A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations," *Nature Electronics*, vol. 2, no. 7, pp. 290–299, Jul. 2019. [Online]. Available: http://www.nature.com/articles/s41928-019-0270-x

[21] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian, "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, Jan. 2020. [Online]. Available: http://www.nature.com/articles/s41586-020-1942-4

[22] C. Li, J. Ignowski, X. Sheng, R. Wessel, B. Jaffe, J. Ingemi, C. Graves, and J. P. Strachan, "CMOS-integrated nanoscale memristive crossbars for CNN and optimization acceleration," in *2020 IEEE International Memory Workshop (IMW)*. Dresden, Germany: IEEE, May 2020, pp. 1–4. [Online]. Available: https://ieeexplore.ieee.org/document/9108112/

[23] D. Ielmini and G. Pedretti, "Device and Circuit Architectures for In-Memory Computing," *Advanced Intelligent Systems*, vol. 2, no. 7, p. 2000040, Jul. 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/aisy.202000040

[24] I. Boybat, M. Le Gallo, S. R. Nandakumar, T. Moraitis, T. Parnell, T. Tuma, B. Rajendran, Y. Leblebici, A. Sebastian, and E. Eleftheriou, "Neuromorphic computing with multi-memristive synapses," *Nature Communications*, vol. 9, no. 1, p. 2514, Dec. 2018. [Online]. Available: http://www.nature.com/articles/s41467-018-04933-y

[25] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. Seoul, South Korea: IEEE, Jun. 2016, pp. 14–26. [Online]. Available: http://ieeexplore.ieee.org/document/7551379/

[26] C. Mackin, H. Tsai, S. Ambrogio, P. Narayanan, A. Chen, and G. W. Burr, "Weight Programming in DNN Analog Hardware Accelerators in the Presence of NVM Variability," *Advanced Electronic Materials*, vol. 5, no. 9, p. 1900026, Sep. 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/aelm.201900026

[27] G. Pedretti, P. Mannocci, C. Li, Z. Sun, J.-P. Strachan, and D. Ielmini, "Redundancy and analog slicing for precise in-memory machine learning - part i: Programming techniques," Nov. 2021.

[28] Z. Sun, G. Pedretti, E. Ambrosi, A. Bricalli, and D. Ielmini, "In-Memory Eigenvector Computation in Time O (1)," *Advanced Intelligent Systems*, p. 2000042, May 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/aisy.202000042

[29] P.-Y. Chen, X. Peng, and S. Yu, "NeuroSim: A Circuit-Level Macro Model for Benchmarking Neuro-Inspired Architectures in Online Learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 3067–3080, Dec. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8246561/