Phase-Change Memory in Neural Network Layers with Measurements-based Device Models

Phase-Change Memory in Neural Network Layers with Measurements-based Device Models

Carmine Paolino*, Alessio Antolini†,‡, Fabio Pareschi*,‡, Mauro Mangia†,‡
Riccardo Rovatti†,‡, Eleonora Franchi Scarselli†,‡, Gianluca Setti*,‡
Roberto Canegallo§, Marcella Carissimi§, Marco Pasotti§
* DET – Politecnico of Torino, corso Duca degli Abruzzi 24, 10129 Torino, Italy.
† DEI – University of Bologna, viale Risorgimento 2, 40136 Bologna, Italy.
‡ ARCES – University of Bologna, via Toffano 2/2, 40125 Bologna, Italy.
§ STMicroelectronics, Agrate Brianza, Italy.
email: carmine.paolino@polito.it, alessio.antolini2@unibo.it

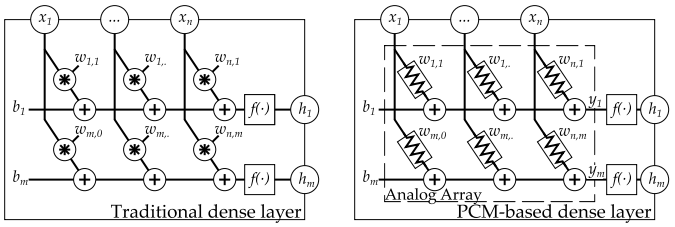
***Abstract*—The search for energy efficient circuital implementations of neural networks has led to the exploration of phase-change memory (PCM) devices as their synaptic element, with the advantage of compact size and compatibility with CMOS fabrication technologies. In this work, we describe a methodology that, starting from measurements performed on a set of real PCM devices, enables the training of a neural network. The core of the procedure is the creation of a computational model, sufficiently general to include the effect of unwanted nonidealities, such as the voltage dependence of the conductances and the presence of surrounding circuitry. Results show that, depending on the task at hand, a different level of accuracy is required in the PCM model applied at train-time to match the performance of a traditional, reference network. Moreover, the trained networks are robust to the perturbation of the weight values, up to 10% standard deviation, with performance losses within 3.5% for the accuracy in the classification task being considered and an increase of the regression RMS error by 0.014 in a second task. The considered perturbation is compatible with the performance of state-of-the-art PCM programming techniques.**

I. Introduction

Phase-change memory (PCM) has arisen as a promising technology for the future of non-volatile memories (NVMs), due to its compatibility with CMOS fabrication processes, its high throughput and read/write endurance [1]. Specifically, the embedded PCM (ePCM) technology reduces process complexity, including at the same time high-power and high-voltage integrated elements [2], [3].

A PCM device exhibits orders of magnitude of difference in conductivity levels when in crystalline or amorphous states, respectively called as SET and RESET. The chalcogenide material the cell is built with can reversibly transition between such states through a programming procedure which melts the material and controls its rate of solidification.

A popular application of PCM technology is Analog In-memory Computing [4], [5], where it enables the compact storage of coefficients while providing means to compute matrix operations within the memory block. A PCM device whose conductance has been programmed to a value $w_{j,k}$, and with a voltage x_i applied across it, will output a current $i_{j,k} = w_{j,k} v_k$. This assumes that the conductance is constant, independent of the applied voltage, and that the summation node is kept at a 0 V level. The latter condition could be obtained by exploiting, for example, the virtual ground node of a non-inverting operation amplifier configuration, however, the former condition is actually not realistic, as physical PCM devices exhibits strong voltage dependence of their conductance. Neglecting for a moment this dependence, collecting the output currents of all the cells in the j-th row, a sum of products is obtained. The operation just described, performed on the whole structure, gives rise to the product between a conductance matrix W of elementary conductances $w_{j,k}$ and the vector of input voltages x_k [6]. The "Analog array" subcircuit in Fig. 1(a) performs this function.



(a) (b)

Fig. 1. (a) Traditional dense and (b) PCM-based dense neural layer structures. The analog array is highlighted in the latter, where we assume that the summation nodes are held at a constant voltage for it to function properly.

There have been multiple applications of PCM arrays within neural network layers [7]. The suitability of an analog resistive array is immediately clear since the core operation of a neural layer is the dot product of a coefficient matrix and the applied inputs, as in Fig. 1(a). The issues with using an analog array stem from the nonidealities of the elementary conductances therein. Voltage dependence of the conductance values, electrical and programming noise, device ageing, all contribute to a worsening of system performance.

The goal of this work is to propose a general method of training neural networks having a PCM-based layer, starting from measured current/voltage characteristic of physical de-

This document has been created in the context of the StorAIge project. This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 101007321. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Belgium, Czech Republic, Germany, Italy, Sweden, Switzerland, Turkey

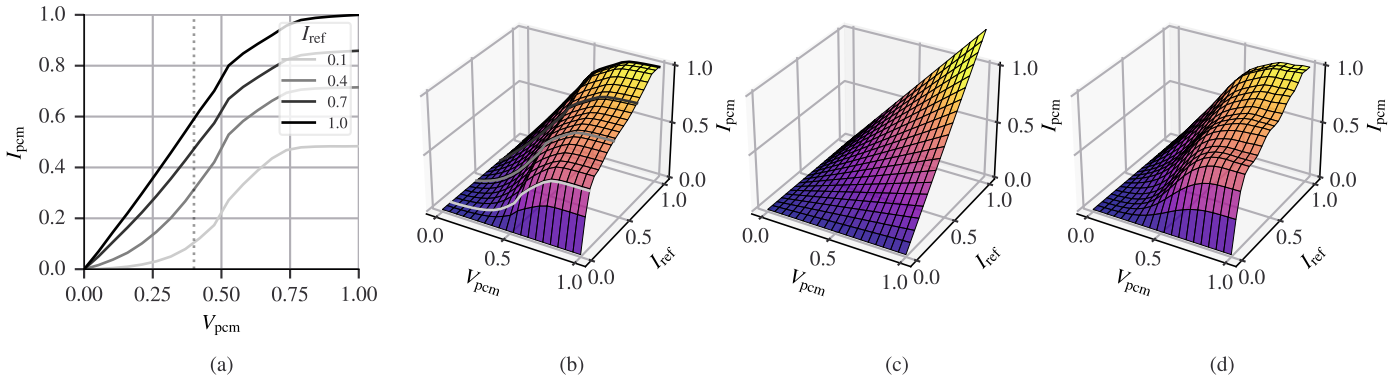


Fig. 2. (a) Average, normalized I/V characteristics of PCM devices in four different states. (b) Spline-based interpolation of the average, normalized PCM behaviours, highlighting the four states depicted in (a). (c) Low-order polynomial fitting of the same data. (c) High-order polynomial fit.

vices. At the core of the methodology is the use of a spline interpolation as the reference numerical model describing the behavior of a PCM cell and its surrounding circuitry, while a polynomial description, more easily manageable by neural network libraries, is applied during the network training. This method is potentially applicable to any other device, provided that a polynomial description is suitable for approximating its behavior. Eventually, physical considerations could be introduced to obtain a more-specific model, all the way up to a complete emulation of of the device, e.g. with a SPICE-like model.

The adopted technology, its characterization and details on the numerical modeling are described in Section II. The necessity for a polynomial description is highlighted in Section III, while the two test setups, a classification task and a regression problem, are analyzed in Section IV. Finally, the conclusion is drawn.

II. PCM Characterization and Numerical Modeling

We have performed measurements on an ePCM test chip designed and manufactured by STMicroelectronics [1] in a 90-nm BJT-CMOS-DMOS (BCD) technology featuring a specifically optimized Ge-rich Ge-Sb-Te (GST) alloy. Its target applications are in the automotive field, where it is used as a digital storage element, with 8 independent 256-kB macrocells and the required analog circuitry. Each ePCM elementary cell is addressed by an NMOS selector [8] and its area occupation is $0.19\,\mu\text{m}^2$. The chip is accessible through a custom evaluation board designed specifically for testing purposes.

Device characterization begins with a programming step, where the PCM cells are brought into highly conductive SET states by means of a single current pulse. A higher pulse intensity determines a more conductive state. The RESET state, conversely is associated in this work to a null SET intensity. The current through each of the 5120 available cells has been measured while sweeping the voltage across each cell, for different values of the applied programming pulses.

The "one-shot" programming phase does not include any iterative feedback mechanism to ensure that the programmed cell state is indeed the expected one. As our goal requires the definition of nominal cell behaviours in different conductive states, then the intensity of the applied current pulse does not provide a good measure of the actual state. We have therefore classified the cell behaviour according to the features of the obtained $I(V)$ curves themselves, disregarding the intensity of the programming pulse. Indeed, similar $I(V)$ curves could be obtained by a cell programmed with a low-intensity pulse which in reality acts stronger than intended, or a high-intensity pulse whose result is particularly weak.

Therefore, typical behaviours of ideal PCM cells are obtained by observing all the curves at a fixed voltage V_{ref}, quantizing the current axis I around a set of reference currents $I_{\text{ref}} = \{I_{\text{ref}}^{(0)}, \ldots, I_{\text{ref}}^{(L-1)}\}$ and averaging all the cells belonging to the same quantization bin to obtain the typical behavior associated to that bin. A selection of curves obtained at different $I_{\text{ref}}^{(l)}$ values is shown in Fig. 2(a). More in detail, we define the nominal behaviour of a PCM cell as the average of all collected I/V characteristics whose current is contained in the interval $(I_{\text{ref}}^{(l)} - \Delta I^{(l)}, I_{\text{ref}}^{(l)} + \Delta I^{(l)})$ at voltage V_{ref}, with $\Delta I^{(l)}/I_{\text{ref}}^{(l)} = 5\%$. Having V_{ref} fixed, we then identify the programming state with the value of measure current. Values have been normalized so that applied voltages V, programming states $I_{\text{ref}}^{(l)}$ and output currents I all lie in the $[0, 1]$ range, as shown in Fig. 2(a). In the following, we will only use this normalized data.

To obtain a numerical model of the type $I(V, I_{\text{ref}}^{(l)})$ we have interpolated the typical behaviours, extracted according to the above procedure, using a spline of order 3. The result is depicted in Fig. 2(b). This allows a reduced local complexity of the model, while still describing accurately the features of the underlying surface. The spline model will be here considered as our reference PCM model. At the same time, polynomial models of arbitrary order (in the range 3 to 27) have been fitted to the spline. Fig. 2(c) and Fig. 2(d) highlight the difference in approximation accuracy obtained with different polynomial degrees. The necessity for polynomial models will be clarified in the following section. Suffice it to say that the use of such models within neural networks programming libraries allows

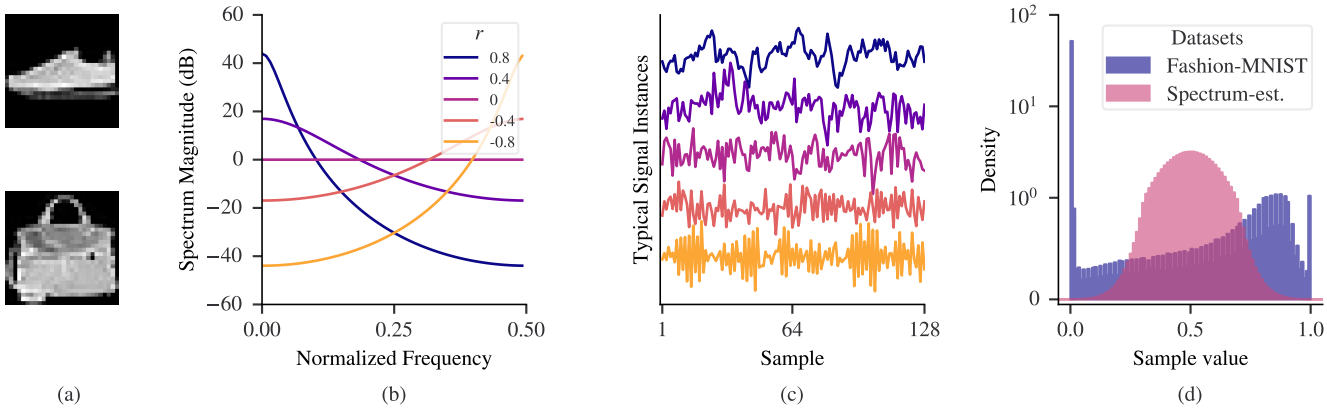


Fig. 3. (a) Examples of Fashion-MNIST instances (b) Spectrums for a selection of of r with corresponding representative instances of length 128 in (c). (d) Normalized histograms of input values for the two datasets being employed. A linear scale is used in the range $[0, 1]$ of the vertical axis.

the automatic differentiator procedures to operate without concerns.

For convenience, the data, which is defined over positive values for both the applied voltage and the programming state, has been extended towards negative values along the programming axis, so that $I(V, -I_{\text{ref}}^{(l)}) = -I(V, I_{\text{ref}}^{(l)})$ with $V, I_{\text{ref}}^{(l)} > 0$. Even if negative values for the programming state are not physically meaningful, the actual hardware implementation can operate so that the contribution of a particular cell is negative on the output. A convenient side effect, is that the artificially-introduced symmetry makes the polynomial representation more well-behaved and reduces the number of significant coefficients, hence the computational burden.

Although relevant metrics could be observed to determine the optimal polynomial representation, monitoring the absolute or relative representation errors, the ultimate validation is obtained on the neural network test setups themselves.

III. Neural Training with PCM Layers

In a traditional dense layer, the core operation for the j-th neuron is $h_j = f(b_j + \sum_i w_{j,i} x_i)$. Aiming towards a circuital implementation where inputs are voltages, and they are weighted by conductances programmed in different states, the expression becomes $h_j = f(b_j + \sum_i I(x_i, w_{j,i}))$, where we neglect any additional term introduced by electrical noise, programming noise or even quantization of the inputs or the outputs.

Training a layer defined as such does not pose any difficulty, provided that $I(x_i, w_{j,i})$ is differentiable with respect to the weights [9]. Standard software frameworks are able to automatically differentiate even complex expressions if described in terms of their library of operators [10]. To the best of our knowledge, no spline primitives are available as elementary operators, hence the need to have an alternative description of the synapses, expressed in terms of available operators. Such a description is in our case of polynomial type. Potentially, a more physically-informed model could be used as well, though as our measurement data includes significant effects from the access devices surrounding the PCM cells, we have preferred to have a unique model that could describe the behaviour of the entire circuital block over the full voltage domain.

An additional advantage is that, at training time, the network could be trained on a model of the neuron which is computationally simpler, though potentially less accurate, enabling a faster exploration of different network topologies.

Two case studies will be analyzed in the following: a classification tasks performed on the Fashion-MNIST dataset [11], and a regression problem, in which the network has to estimate a parameter describing the spectral content of randomly generated signal instances.

IV. Results

In the following we will show numerical results on the training of neural networks in which one layer is PCM-based. In all setups, the performance of a neural network employing only conventional dense layers and having the same structure, is used as a reference. To train the PCM-based network, the PCM synapses are always described by their polynomial model, with an arbitrarily selected degree and by identifying $L = 10$ different reference currents. An initial performance metric is thus obtained, related exclusively to the use of the polynomial. The final evaluation is then performed on the same network, preserving the trained weights, but replacing in the PCM-based layer the polynomial model with the spline one, representing our reference model for nominal PCM devices.

Since in a physical implementation the state of a PCM cell cannot be programmed to arbitrary accuracy, we also test the robustness of the network towards this kind of perturbation. We model the variation of the PCM state with a white gaussian noise added to the nominal values of the weights (i.e., that suggested by training) during the final evaluation. The variance of the weight noise is normalized to the nominal value, so that their ratio is fixed. Clipping is then applied to ensure that the noisy weights are still within the validity range of the numerical models.

Two different applications are shown, trying to highlight the different features of the setups presented in this work. The optimizer in all setups is Adam, with parameters: learning rate

equal to 10^{-4} and the exponential decay rate for the 1st and 2nd moments equal to 0.9 and 0.999. Results are condensed in Fig. 4.

A. Fashion-MNIST Classification

The dataset is made of grayscale images of clothing articles, in a 28×28 pixel format. Two examples are shown in Fig. 3(a). The neural network topology being considered has an input-flattening layer followed by a single dense layer with sigmoid activation functions and 10 output nodes. The loss function is the sparse categorical cross-entropy. While the conventional reference network has no constraints on the weights, in the PCM-based one we have introduced a "bathtub" regularization to force them within the [0, 1] range. This implies a physical realization requiring only positively-contributing PCM synapses on each layer output.

To asses the performance we use here the accuracy defined as the correct classification rate. Analyzing the results shown in Fig. 4(a), a monotonic trend is clear, with networks trained on a high-order polynomial model almost matching the performance of the reference network.

The fact that the weights obtained by training a low-order polynomial, as that depicted in Fig. 2(c) is already sufficient to solve the classification task with ~ 0.78 accuracy has been associated to the statistical distribution of pixel intensities. Being their density concentrated around the extremes of the available range, as shown in Fig. 3(d), the inherent nonlinearity of the models is not significantly excited. The model feature that matters is that their output is different for low and high input values. Both the spline and polynomials being employed possess such a feature, resulting in a limited performance drop with respect to the reference case.

The application of noise on the trained weights only becomes significant around 10% relative standard deviation, with a performance loss still within 3.5% of the original one. State-of-the-art iterative programming techniques of the physical devices may indeed be able to achieve such a level of programming accuracy [12], [13].

B. Spectral Estimation Regression

The second task being evaluated is a regression problem artificially constructed so that the nonlinearity of the PCM $I(V)$ characteristic can be excited even more.

The problem is that of estimating the properties of the Fourier-spectrum of random signal instances. Given a value $-1 < r < 1$, let us define an autocorrelation matrix K such that $K_{j,k} = r^{|j-k|}$, $1 \leq j, k \leq n$, with n the length of the signal instances being observed. The power spectrum of the stationary stochastic process thus defined is:

$$\Psi(f) = \frac{1 - r^2}{1 + r^2 - 2r\cos(2\pi f)} \,.$$

Its profile is high-pass for $-1 < r < 0$, flat/white for $r = 0$ and low-pass for $0 < r < 1$. Examples of spectra for different values of r are shown in Fig. 3(b), with corresponding representative signal instances depicted in Fig. 3(c).

Given a value for r, signals can be generated by computing instances of a multivariate gaussian distribution $\mathcal{N}(0, K)$. Inverting the relationship between the power spectrum and r is not possible, and the neural network has to estimate it by looking at each signal instance and providing an answer in the $[-1, 1]$ range.

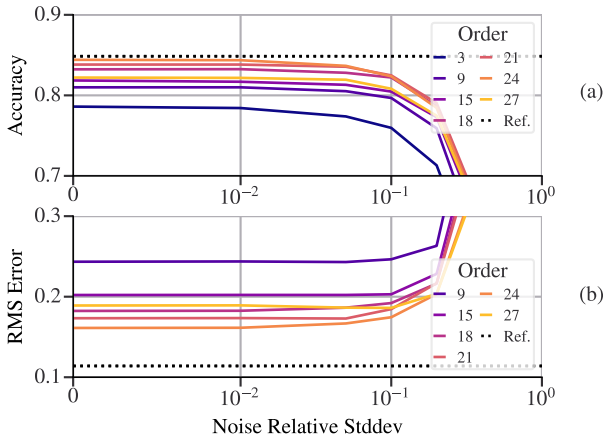


Fig. 4. Results for (a) Fashion-MNIST classification, and (b) spectrum estimation regression. The black, dotted line represents the performance obtained by a neural network employing standard dense layers, without noise. Solid lines refer to the performance of networks using the spline PCM model, with the weights trained on the polynomial description of the device, and additional noise included during the evaluation phase.

The network structure being tested operates on signal instances of 32 samples and it has three dense layers of size 256, 256 and 1. The first two layers have relu activation functions, while the output layer has none. The loss function is the mean squared error, while the performance metric being observed is the root mean squared (RMS) error. A conventional network with such a structure achieves a 0.114 RMS estimation error.

The weighting coefficients of the PCM-based layer in this case have been constrained in the range [-1, 1] by a "bathtub" regularization. From an implementation point of view, this requires a way for a PCM cell, to have a negative contribution on the sum of synapses currents. The practical approach could involve enabling a current mirror on the specific branch or having two arrays of conductances, with opposite contributions on the output [14].

Results in Fig. 4(b) highlight a monotonic trend up to order 24, with a sudden worsening of performance observed at 27.

The detrimental effect of the additive noise on the weights is still under control for 10% relative standard deviation, with variations on the order of 0.014 RMS error. It is striking to observe a minimal performance increase when weight noise is applied to the network trained on the order-27 polynomial.

V. Conclusion

We have proposed a way of including arbitrary synapse models within a neural layer, targeting specifically phase-change memory devices. Two test setups have validated the procedure, a classification task on the Fashion-MNIST dataset and an artificially constructed regression task. The injection of noise on the trained weights has highlighted the robustness of the networks to a point that makes the devices promising candidates in actual circuital implementations.

REFERENCES

[1] G. W. Burr, B. N. Kurdi, J. C. Scott, C. H. Lam, K. Gopalakrishnan, and R. S. Shenoy, "Overview of candidate device technologies for storage-class memory," *IBM Journal of Research and Development*, vol. 52, no. 4/5, pp. 449–464, Jul. 2008. doi: 10.1147/rd.524.0449

[2] G. D. Sandre *et al.*, "A 90nm 4Mb embedded phase-change memory with 1.2V 12ns read access time and 1MB/s write throughput," in *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*, Feb. 2010, pp. 268–269. doi: 10.1109/ISSCC.2010.5433911

[3] M. Pasotti *et al.*, "A 32-KB ePCM for Real-Time Data Processing in Automotive and Smart Power Applications," *IEEE J. Solid-State Circuits*, vol. 53, no. 7, pp. 2114–2125, Jul. 2018. doi: 10.1109/JSSC.2018.2828805

[4] C. Paolino *et al.*, "Compressed Sensing by Phase Change Memories: Coping with Encoder non-Linearities," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. Daegu, Korea: IEEE, May 2021, pp. 1–5. doi: 10.1109/ISCAS51556.2021.9401176

[5] W. Haensch, T. Gokmen, and R. Puri, "The Next Generation of Deep Learning Hardware: Analog Computing," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 108–122, Jan. 2019. doi: 10.1109/JPROC.2018.2871057

[6] 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. doi: 10.1073/pnas.1815682116

[7] V. Joshi *et al.*, "Accurate deep neural network inference using computational phase-change memory," *Nature Communications*, vol. 11, no. 1, p. 2473, May 2020. doi: 10.1038/s41467-020-16108-9

[8] M. Carissimi *et al.*, "2-Mb Embedded Phase Change Memory With 16-ns Read Access Time and 5-Mb/s Write Throughput in 90-nm BCD Technology for Automotive Applications," *IEEE Solid-State Circuits Letters*, vol. 2, no. 9, pp. 135–138, Sep. 2019. doi: 10.1109/LSSC.2019.2935874

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986. doi: 10.1038/323533a0

[10] M. Abadi *et al.*, "TensorFlow: large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org.

[11] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," *arXiv:1708.07747 [cs, stat]*, Sep. 2017.

[12] G. F. Close *et al.*, "A 256-Mcell Phase-Change Memory Chip Operating at 2+ Bit/Cell," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 6, pp. 1521–1533, Jun. 2013. doi: 10.1109/TCSI.2012.2220459

[13] A. Antolini, E. Franchi Scarselli, A. Gnudi, M. Carissimi, M. Pasotti, P. Romele, and R. Canegallo, "Characterization and Programming Algorithm of Phase Change Memory Cells for Analog In-Memory Computing," *Materials*, vol. 14, no. 7, p. 1624, Mar. 2021. doi: 10.3390/ma14071624

[14] M. Le Gallo, A. Sebastian, G. Cherubini, H. Giefers, and E. Eleftheriou, "Compressed Sensing With Approximate Message Passing Using In-Memory Computing," *IEEE Transactions on Electron Devices*, vol. 65, no. 10, pp. 4304–4312, Oct. 2018. doi: 10.1109/TED.2018.2865352