Mixed-precision architecture based on computational memory for training deep neural networks

S. R. Nandakumar*†, Manuel Le Gallo*, Irem Boybat*‡, Bipin Rajendran†, Abu Sebastian* and Evangelos Eleftheriou*
*IBM Research - Zurich, 8803 Rüschlikon, Switzerland Email: nsi@zurich.ibm.com, ase@zurich.ibm.com
†New Jersey Institute of Technology (NJIT), Newark, NJ 07102, USA
‡Ecole Polytechnique Federale de Lausanne (EPFL), 1015 Lausanne, Switzerland

***Abstract*—Deep neural networks (DNN) have revolutionized the field of machine learning by providing unprecedented human-like performance in solving many real-world problems such as image or speech recognition. Training of large DNNs, however, is a computationally intensive task, and this necessitates the development of novel computing architectures targeting this application. A computational memory unit where resistive memory devices are organized in crossbar arrays can be used to store the synaptic weights in their conductance states. The expensive multiply accumulate operations can be performed in place using Kirchhoff's circuit laws in a non-von Neumann manner. However, a key challenge remains the inability to alter the conductance states of the devices in a reliable manner during the weight update process. We propose a mixed-precision architecture that combines a computational memory unit storing the synaptic weights with a digital processing unit and an additional memory unit that stores the accumulated weight updates in high precision. The new architecture delivers classification accuracies comparable to those of floating-point implementations without being constrained by challenges associated with the non-ideal weight update characteristics of emerging resistive memories. The computational memory unit in a two layer neural network realized using non-linear stochastic models of phase-change memory achieves a test accuracy of 97.40% in the MNIST digit classification problem.**

***Keywords*—Deep learning, In-memory computing, Mixed-precision computing, Phase-change memory**

I. Introduction

Deep neural networks (DNN) including convolutional neural networks, deep belief networks, and Long-Short-Term-Memories are loosely inspired by biological neural networks in which layers of neurons are interconnected by plastic synapses. The neuronal outputs in these networks are real-valued numbers, processed at consecutive iterations. Learning involves the strengthening or weakening of the synapses to optimize a cost function. Through a combination of factors such as the availability of massive labeled datasets and the highly parallel matrix manipulations offered by modern GPUs, these networks have recently achieved considerable success in numerous applications [1].

These software advances have fueled a significant interest in designing non-von Neumann co-processors for training DNNs. A system comprising dense crossbar arrays of resistive memory devices has been proposed to perform the various steps involved in the training of DNNs [2]–[5]. The devices store information in their conductance states [6], [7], which can be used to represent the synaptic weights. The matrix-vector multiplications needed during the forward and backward propagations of different data signals is realized via Kirchhoff's circuit laws in the crossbar. Weight updates can be achieved by modifying the conductance of the resistive memory devices by applying appropriate programming pulses. However, this approach can attain satisfactory training accuracy only with ideal, not-yet-available resistive memory devices [4]. The experimental demonstrations based on existing resistive memory devices have shown reduced classification accuracies because of the difficulty in achieving precise conductance changes in these devices [2], [8]. In parallel, there are some key developments taking place at the algorithmic front with respect to training DNNs using digital arithmetic with reduced precision [9]–[12]. Recent work shows that it is possible to have binary precision for the weights used in the multiply-accumulate operations (during the forward and backward propagations) as long as the precision of the stored weights in which gradients are accumulated is retained [11].

Building on this insight and on our recent work on mixed-precision memcomputing [13], we present a mixed-precision architecture based on computational memory to train DNNs. This is followed by a detailed investigation of various undesirable attributes of the constituent devices in such a computational memory unit and a thorough evaluation of how the proposed architecture copes with such behavior. Our studies suggest that the proposed architecture can deliver classification accuracies comparable to those of floating-point implementations even when all the propagations are done inexactly in the computational memory and when inaccurate conductance updates are done using single-shot programming of memory devices.

II. The mixed-precision architecture based on computational memory

In Fig. 1, we introduce the mixed-precision computational memory approach to train DNNs. The most expensive operation during the forward and backward propagation is obtaining the weighted sums, which are results of matrix-vector multiplications. A computational memory unit which has resistive memory devices organized in a crossbar array is ideally suited for performing these matrix-vector operations in constant time complexity [14], [15]. The neuron activations x_i are applied as voltages to the word lines using digital-to-analog converters (DACs). Currents proportional to the conductance will flow through the devices and the resulting total current at any bit line calculated following Kirchhoff's law is $I_j = \Sigma_i W_{ji} x_i$. Here W_{ji} represents the device conductance connecting neuron i to the next-layer neuron j. These currents read and digitized using analog-to-digital converters (ADCs) represent the desirable weighted sum operation results. The same crossbar array can be used to perform the matrix multiplication during the back-propagation in the same layer. Here, the errors to be back-propagated, δ_k, are applied as voltages to the bit lines and

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

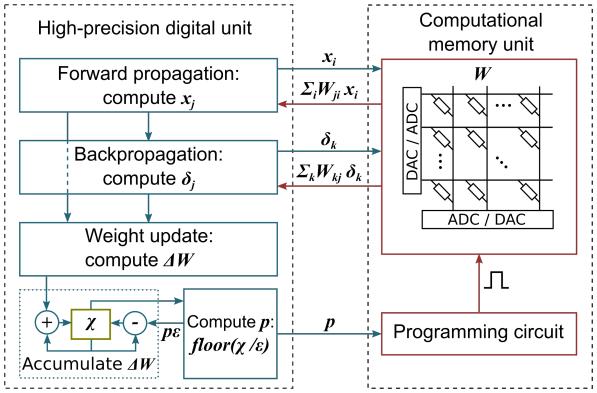

Fig. 1. Mixed-precision architecture based on computational memory. The synaptic weights are stored in a computational memory unit as conductance states of resistive memory devices organized in crossbar arrays. The matrix-vector multiplications associated with the forward and the backward propagation are performed in place in the memory arrays. The weight updates are accumulated in a volatile memory, χ, in high precision until it becomes comparable to the update granularity (ε) of the memory devices. The device updates are integer multiples of ε which are subtracted from χ.

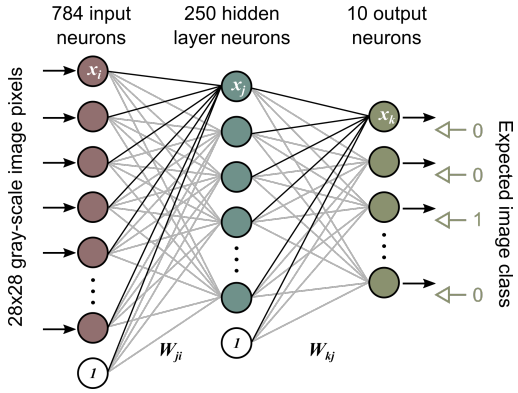

Fig. 2. The neural network used to evaluate the mixed-precision architecture. The objective is handwritten digit classification based on the MNIST data set. There are 784 input neurons, 250 hidden sigmoid neurons, and 10 output sigmoid neurons.

the total current read out from any word line represents a transposed matrix multiplication result ($\Sigma_k W_{kj}\delta_k$).

The desired weight updates are determined as the product of the back-propagated error and the neuron activation, $\Delta W_{ji} = \eta \delta_j x_i$, where η is the learning rate. Even though the computational memory unit can accelerate the forward and the backward propagation significantly, updating the synaptic weights with the desired precision is very challenging. Often, the device conductance representing the synaptic weights has a conductance change granularity dictated by the physical characteristics of the memory device. Let ε be the absolute value of the smallest conductance change that can be reliably achieved in a device. In the proposed approach, the weight updates are accumulated in high precision in a variable χ. The device conductance will be updated only if the magnitude of the accumulated weight update becomes greater than or equal to an integer multiple of ε. The number of programming pulses p to be applied to the resistive memory devices is then determined by flooring χ/ε toward zero, and the same number of εs is subtracted from the χ. Depending on the sign of p, the conductance value of the corresponding device will be increased (potentiated) or decreased (depressed). Note that the actual conductance state of the devices is never read back, and hence it is not possible to confirm whether the requested weight update is accurately attained as an equivalent conductance change in the devices. In spite of this, we will show in the subsequent sections that this scheme works remarkably well and that the performance is often comparable to those of floating-point implementations. This single-shot programming method, which avoids a verification and/or iterative programming step, enables the acceleration of the learning task.

III. Evaluation of the Mixed-Precision Architecture

A. The simulation framework

The performance of the mixed-precision architecture is analyzed based on its classification accuracy of the MNIST handwritten digit dataset using a neural network as shown schematically in Fig. 2. The number of neurons in the input, the hidden and the output layer is 784, 250, and 10, respectively. The hidden and the output neurons are sigmoid.

The network is trained using the entire training set of 60,000 images for ten epochs, and a test accuracy is reported based on the classification of 10,000 test images. Each 28×28 gray-scale images from the data set are normalized before they are supplied as input to the network. No other preprocessing is performed on the images. We used the quadratic objective function for the back-propagation-based training and used a fixed learning rate. The network achieves 98 % floating point (64-bit) test accuracy when trained using stochastic gradient descent. This classification result is used as *reference* to evaluate the performance of our mixed-precision approach. The final weight distribution from the high-precision training was approximately in the range $[-1, 1]$.

B. Inaccuracies arising from weight updates

In this section, we will evaluate how the proposed architecture copes with the issues associated with the non-ideal weight updates of emerging memory devices. We assume a hypothetical linear device with a fixed n-bit update granularity which covers its conductance range in $2^n - 2$ steps such that there are $2^n - 1$ levels in the absence of conductance change stochasticity. Assuming a similar final weight distribution range as that from the floating-point simulation, we set the update size, $\varepsilon = 2/(2^n - 2)$. We program a device only if the accumulated weight updates exceed ε in magnitude. Even though it is desirable to induce a conductance change corresponding to an integer multiple of ε, the actual conductance change in a device is often stochastic. Therefore, the actual weight update from the device due to a single programming pulse, denoted by $\Delta\widehat{W}$, is modeled as a Gaussian random variable whose mean is ε and whose standard deviation (σ) is a fractional multiple of ε. The neural network synapses are realized with such devices, initialized to {-1, 0, 1} states with a discrete distribution whose variance is normalized by the number of neurons in the pre- and post-synaptic layers. Device read noise and analog-digital converters are ignored at this stage. The simulated classification accuracy with different amounts of stochasticity in the updates is shown in Fig. 3a. When the weight updates are non-stochastic, the test accuracy

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

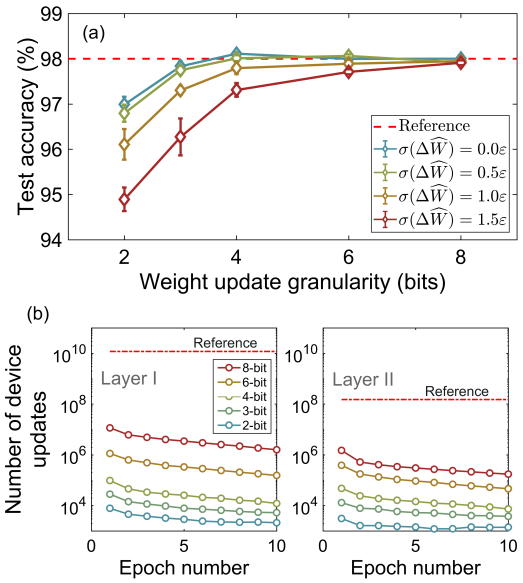

Fig. 3. (a) Effect of granularity and stochasticity associated with weight updates. Linear devices with symmetric potentiation and depression granularity are assumed. The standard deviation of the weight update randomness, $\sigma(\Delta\widehat{W})$ is taken as a multiple of the weight update granularity, ε. The error bars indicate the standard deviation corresponding to five repetitions of the simulation. (b) Sparsity of device programming. The device update count per epoch in each layer is plotted for different values of ε. The number of synapses in each layer times the total training image count is indicated as reference.

drop is only 1% for 2-bit granularity compared to the 64-bit floating-point reference. With 3-bit granularity, the accuracy is very close to that obtained in software simulations. As the amount of stochasticity is increased, the performance degrades with reducing number of bits. However, it is remarkable that even when the standard deviation of the weight update is equal to or greater than the ideal weight update granularity itself, the drop in test accuracy is still within approximately 4%.

In this mixed precision scheme, weight update accumulation can reduce the number of required device programming instances by more than two orders of magnitude, as smaller updates are combined and applied together to the device. The device updates becomes sparser as the weight update granularity, ε becomes larger (Fig. 3b).

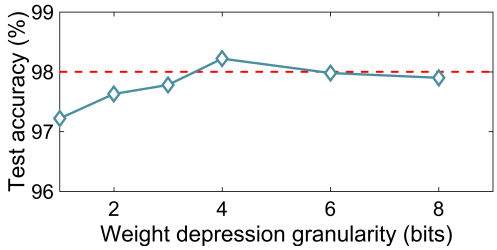

Fig. 4. Effect of asymmetric conductance response. The test accuracy, when trained with devices of fixed 8-bit potentiation granularity and variable depression granularity, is plotted as a function of the depression granularity (expressed in bits). Weight updates are assumed to be deterministic.

Next, we study the influence of asymmetric conductance update response. We assume a device with fixed but unequal potentiation and depression granularity. The mixed-precision method can cope with this behavior by using different thresholds, ε_P and ε_D, for conductance increment and decrement, respectively. For example, in Fig. 4, we assume an 8-bit potentiation granularity and the depression granularity is varied. The 1-bit depression corresponds to a situation where the update granularity, ε_D, equals the entire weight range in contrast to the previous definition of ε. The weight updates are assumed to be deterministic. The resulting test accuracies show less than 1% drop, demonstrating the efficacy of the proposed scheme to tolerate device update asymmetry effectively.

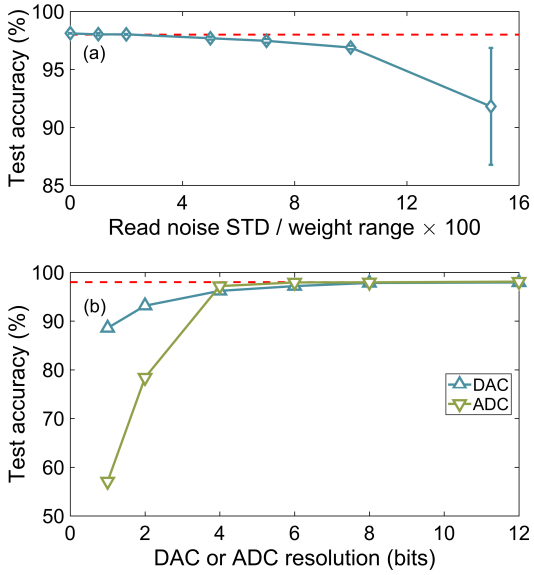

Fig. 5. (a) Effect of read noise. A 4-bit weight update granularity with no stochasticity is assumed. The standard deviation of the Gaussian distributed additive read noise associated with the weights is varied as a fraction of the total weight range. Error bars are plotted from 5 repetitions. (b) Effect of finite resolution data converters. The weight update granularity is assumed to be 4-bit, without stochasticity and read noise. The curve with triangle indicates simulation results where DACs are used at the crossbar input whereas the output current is read back in floating-point precision. The curve with inverted triangle indicates results where the crossbar input has floating point precision whereas ADCs are used for reading back the output current.

C. *Inaccuracies arising from matrix-vector multiplication*

In this section, we study the influence of conductance fluctuations and finite resolution of data converters. Resistive memory devices typically exhibit fluctuations in conductance arising from trapping/detrapping processes [16]. The effect of this read noise is tested by adding a zero mean white Gaussian noise to the linear device model. The noise is added to the weights used in matrix multiplications in the forward and the backward propagations during training. The same approach is followed during the testing phase. The standard deviation of the noise is varied as a fraction of the total weight range, and the resulting test accuracies are shown in Fig. 5a. It can be seen that the methodology is robust to a read noise of up to 5% of the total weight range.

An additional source of noise in the matrix-vector multiplication is the quantization error from the DACs and ADCs at the crossbar periphery. During forward propagation, the neuron activations evaluated in the digital domain are converted to analog voltages using DACs before they are applied to the word lines of the crossbar array. Then the weighted sum obtained as

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

currents in the bit lines are read back using ADCs. Similarly, the back-propagated error is converted to analog voltage when applied to the crossbar array. The range for the DACs are fixed for sigmoid and tanh neuron activations, whereas for ReLU neurons this could be a challenge, as their range depends on the data and weight distribution. Here, we chose sigmoid neurons for our network, which fixed the DAC range in the forward propagation. Also, we normalized the back-propagated errors to a fixed range which becomes the input for the DACs during backward propagation. The normalization factor is multiplied with the learning rate during the weight update calculation. The range for ADCs in the forward propagation is fixed since the digitized values become input for sigmoid neurons. Further, the range for ADCs for the back-propagated error could be fixed to limit the maximum weight update. To study the effect of DACs and ADCs separately, the bit precision of one of them is varied, whereas the other variables are represented in floating-point precision. Fig. 5b shows that an 8-bit resolution is sufficient to avoid any noticeable degradation in test accuracy.

D. Phase-change memory synapses

Phase-change memory (PCM) is a relatively mature resistive memory technology that has found applications in the space of storage-class memory [17] and novel computing paradigms such as neuromorphic computing [18]–[20] and computational memory [13], [21]–[23]. It is based on the property of chalcogenide alloys, typically compounds of Ge, Sb and Te, whose electrical conductivities differ drastically depending on whether they are in the ordered crystalline phase or in the disordered amorphous phase. It is possible to achieve a continuum of conductance values in these devices by partial crystallization or amorphization [24], [25]. This analog storage capability makes PCM particularly well suited for computational memory applications. However, PCM devices exhibit most of the non-idealities we described earlier, such as granularity, stochasticity, and asymmetric and non-linear conductance response.

To evaluate the suitability of PCM devices for the mixed-precision approach to train DNNs, we developed a model that captures the essential physical attributes of PCM devices. The model is created based on characterization data from approximately 10,000 devices integrated in 90 nm CMOS technology [26]. The devices are subjected to 20 programming pulses of fixed amplitude, and each state is read 50 times to eliminate read noise. The mean and standard deviation of the extracted conductance change (ΔG) versus the average initial conductance for each programming pulse are fitted using piece-wise linear models as shown in Fig. 6a, b. Assuming the ΔG to be a Gaussian random variable, the device cumulative pulse response is simulated, and the statistical plot of the resulting stochastic model behavior is plotted in Fig. 6c.

This device model was used in the simulations to study the influence on training DNNs. Two PCM devices in differential configuration with weight refresh [2], [27] are used for the network weight simulation. The conductances are initialized to a normal distribution around 2 μS whose standard deviation is normalized based on the number of neurons in the pre- and post-synaptic layers. Resulting test accuracy after 10 epochs of training was 97.78% (Fig. 6d). Incorporating a fixed read noise (zero mean Gaussian noise with experimentally measured

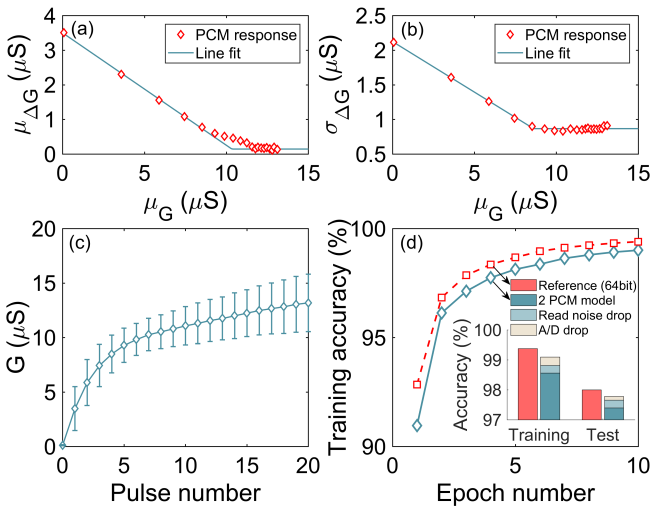

Fig. 6. PCM device model. Piece-wise linear approximations to (a) the mean, μ, and (b) the standard deviation, σ, of the experimentally measured ΔG from $Ge_2Sb_2Te_5$-based PCMs for 50 μA, 50 ns programming pulses as a function of their average current conductance state, μ_G, are used to model the device. (c) The resulting model cumulative conductance evolution in pulse programming simulation. (d) Training using PCM models. Two non-linear PCM device models in differential configuration are used at the cross-points for the neural network weights. Training convergence and test accuracies (inset) are shown. Device-model based network simulation achieves 97.78% test accuracy. Additional drop from the read noise (0.26%) and analog-digital converters (0.12%) are indicated.

average standard deviation) and 8-bit analog-digital converters during training and testing resulted in an additional 0.38% drop in accuracy. We also tested the training performance where each synapse is realized using a single PCM device model at the cross-point, exploiting the capability of the scheme to cope with the strongly asymmetric conductance response. The final test accuracy for the MNIST dataset classification was 96.5%, indicating the robustness of our scheme. Note that, with improved PCM devices and synaptic architectures these numbers are likely to increase [28], [29].

IV. Conclusion

In this work, we presented a mixed-precision architecture based on computational memory to train DNNs. The central idea is to use a computational memory unit in conjunction with a high-precision processing unit. The computationally expensive matrix-vector multiplications arising during the data propagation stages of the learning algorithms are realized using the computational memory unit which can compute the multiply-accumulation in constant time complexity. The weight updates are accumulated in high precision and are transferred only sporadically to the computational memory unit, reducing the device programming overhead. This mixed-precision approach overcomes the non-ideal weight update characteristics of resistive memory devices due to the limited granularity, stochasticity, asymmetry, and non-linearity associated with the device conductance changes. It achieves a simulated test accuracy of 97.40% on the MNIST handwritten digit classification problem using a two layer neural network in which the computational memory units are realized using models of state-of-the-art 90 nm phase-change memory devices and conductance updates are based on single-shot programming.

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] G. W. Burr, R. M. Shelby, S. Sidler, C. Di Nolfo, J. Jang, I. Boybat, R. S. Shenoy, P. Narayanan, K. Virwani, E. U. Giacometti *et al.*, "Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element," *IEEE Transactions on Electron Devices*, vol. 62, no. 11, pp. 3498–3507, 2015.

[3] I. Kataeva, F. Merrikh-Bayat, E. Zamanidoost, and D. Strukov, "Efficient training algorithms for neural networks based on memristive crossbar circuits," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2015, pp. 1–8.

[4] T. Gokmen and Y. Vlasov, "Acceleration of deep neural network training with resistive cross-point devices: design considerations," *Frontiers in Neuroscience*, vol. 10, p. 333, 2016.

[5] G. W. Burr, R. M. Shelby, A. Sebastian, S. Kim, S. Kim, S. Sidler, K. Virwani, M. Ishii, P. Narayanan, A. Fumarola *et al.*, "Neuromorphic computing using non-volatile memory," *Advances in Physics: X*, vol. 2, no. 1, pp. 89–124, 2017.

[6] L. Chua, "Resistance switching memories are memristors," *Applied Physics A*, vol. 102, no. 4, pp. 765–783, 2011.

[7] H. S. P. Wong and S. Salahuddin, "Memory leads the way to better computing," *Nature Nanotechnology*, vol. 10, no. 3, pp. 191–194, 2015.

[8] I. Boybat, M. Le Gallo, T. Moraitis, Y. Leblebici, A. Sebastian, and E. Eleftheriou, "Stochastic weight updates in phase-change memory-based synapses and their influence on artificial neural networks," in *13th Conference on Ph. D. Research in Microelectronics and Electronics (PRIME)*. IEEE, 2017, pp. 13–16.

[9] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 1737–1746.

[10] L. K. Muller and G. Indiveri, "Rounding methods for neural networks with low resolution synaptic weights," *arXiv preprint arXiv:1504.05767*, 2015.

[11] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, 2015, pp. 3123–3131.

[12] P. Merolla, R. Appuswamy, J. Arthur, S. K. Esser, and D. Modha, "Deep neural networks are robust to weight binarization and other non-linear distortions," *arXiv preprint arXiv:1606.01981*, 2016.

[13] 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," *arXiv preprint arXiv:1701.04279*, 2017.

[14] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: programming 1T1M crossbar to accelerate matrix-vector multiplication," in *53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2016, pp. 1–6.

[15] 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, p. 784, 2017.

[16] D. Fugazza, D. Ielmini, S. Lavizzari, and A. Lacaita, "Distributed-Poole-Frenkel modeling of anomalous resistance scaling and fluctuations in phase-change memory (PCM) devices," in *IEEE International Electron Devices Meeting (IEDM)*, 2009, pp. 1–4.

[17] G. W. Burr, M. J. Brightsky, A. Sebastian, H.-Y. Cheng, J.-Y. Wu, S. Kim, N. E. Sosa, N. Papandreou, H.-L. Lung, H. Pozidis *et al.*, "Recent progress in phase-change memory technology," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 2, pp. 146–162, 2016.

[18] T. Tuma, A. Pantazi, M. Le Gallo, A. Sebastian, and E. Eleftheriou, "Stochastic phase-change neurons," *Nature Nanotechnology*, vol. 11, no. 8, pp. 693–699, 2016.

[19] T. Tuma, M. Le Gallo, A. Sebastian, and E. Eleftheriou, "Detecting correlations using phase-change neurons and synapses," *IEEE Electron Device Letters*, vol. 37, no. 9, pp. 1238–1241, 2016.

[20] S. Sidler, A. Pantazi, S. Woźniak, Y. Leblebici, and E. Eleftheriou, "Unsupervised learning using phase-change synapses and complementary patterns," in *International Conference on Artificial Neural Networks*. Springer, 2017, pp. 281–288.

[21] P. Hosseini, A. Sebastian, N. Papandreou, C. D. Wright, and H. Bhaskaran, "Accumulation-based computing using phase-change memories with fet access devices," *IEEE Electron Device Letters*, vol. 36, no. 9, pp. 975–977, 2015.

[22] A. Sebastian, T. Tuma, N. Papandreou, M. Le Gallo, L. Kull, T. Parnell, and E. Eleftheriou, "Temporal correlation detection using computational phase-change memory," *Nature Communications*, vol. 8, no. 1, p. 1115, 2017.

[23] M. Le Gallo, A. Sebastian, G. Cherubini, H. Giefers, and E. Eleftheriou, "Compressed sensing recovery using computational memory," in *IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2017.

[24] N. Papandreou, A. Pantazi, A. Sebastian, M. Breitwisch, C. Lam, H. Pozidis, and E. Eleftheriou, "Multilevel phase-change memory," in *17th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*. IEEE, 2010, pp. 1017–1020.

[25] A. Sebastian, M. Le Gallo, and D. Krebs, "Crystal growth within a phase change memory cell," *Nature Communications*, vol. 5, p. 4314, 2014.

[26] S. R. Nandakumar, I. Boybat, M. Le Gallo, A. Sebastian, B. Rajendran, and E. Eleftheriou, "Supervised learning in spiking neural networks with MLC PCM synapses," in *75th Annual Device Research Conference (DRC)*. IEEE, 2017, pp. 1–2.

[27] M. Suri, O. Bichler, D. Querlioz, O. Cueto, L. Perniola, V. Sousa, D. Vuillaume, C. Gamrat, and B. DeSalvo, "Phase change memory as synapse for ultra-dense neuromorphic systems: Application to complex visual pattern extraction," in *IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2011, pp. 4.4.1–4.4.4.

[28] W. W. Koelmans, A. Sebastian, V. P. Jonnalagadda, D. Krebs, L. Dellmann, and E. Eleftheriou, "Projected phase-change memory devices," *Nature communications*, vol. 6, p. 8181, 2015.

[29] I. Boybat, M. L. Gallo, S. R. Nandakumar, T. Moraitis, T. Parnell, T. Tuma, B. Rajendran, Y. Leblebici, A. Sebastian, E. Eleftheriou *et al.*, "Neuromorphic computing with multi-memristive synapses," *arXiv preprint arXiv:1711.06507*, 2017.

978-1-5386-4881-0/18/$31.00 ©2018 IEEE