MemSE: Fast MSE Prediction for Noisy Memristor-Based DNN Accelerators

Jonathan Kern, Sébastien Henwood, Gonçalo Mordido, Elsa Dupraz, Abdeldjalil Aissa El Bey, Yvon Savaria, François Leduc-Primeau

MemSE: Fast MSE Prediction for Noisy Memristor-Based DNN Accelerators

Jonathan Kern[1,2], Sébastien Henwood[1], Gonçalo Mordido[1,3], Elsa Dupraz[2], Abdeldjalil Aïssa-El-Bey[2], Yvon Savaria[1], and François Leduc-Primeau[1]
[1]Department of Electrical Engineering, Polytechnique Montreal, Montreal, QC, Canada
[2]IMT Atlantique, CNRS UMR 6285, Lab-STICC, Brest, France
[3]Mila - Quebec AI Institute, Montreal, QC, Canada

***Abstract*—Memristors enable the computation of matrix-vector multiplications (MVM) in memory and, therefore, show great potential in highly increasing the energy efficiency of deep neural network (DNN) inference accelerators. However, computations in memristors suffer from hardware non-idealities and are subject to different sources of noise that may negatively impact system performance. In this work, we theoretically analyze the mean squared error of DNNs that use memristor crossbars to compute MVM. We take into account both the quantization noise, due to the necessity of reducing the DNN model size, and the programming noise, stemming from the variability during the programming of the memristance value. Simulations on pre-trained DNN models showcase the accuracy of the analytical prediction. Furthermore the proposed method is almost two order of magnitude faster than Monte-Carlo simulation, thus making it possible to optimize the implementation parameters to achieve minimal error for a given power constraint.**

I. Introduction

Energy consumption represents one of the most important design objectives for deep neural network (DNN) accelerators, in particular, because it enables low-latency on-device processing. The main source of energy consumption in most accelerators is due to data movement [1], specifically retrieving data from memory and delivering it to processing units. To bypass this problem, in-memory computing techniques show great promise in terms of energy efficiency by directly processing the data in memory. Particularly, memristors are an emerging technology well-suited for neural networks, which allow performing computations, such as dot products, in memory [2]. Despite the energy benefits, programming the values of the conductance in memristor crossbars is an inexact process subject to noise [3]. For instance, existing hardware implementations report precisions from 2 bits [4] to 7.5 bits [5] per memristor. While additional techniques such as bit-slicing [6] may be leveraged to increase precision, this comes at the cost of increased area and energy usage.

DNNs have been shown to be robust to noise affecting the weights, although the amount of noise must be designed carefully to satisfy accuracy constraints [7]–[9]. Over the past few years, implementing neural networks using memristors has attracted a lot of attention [5], [10]. However, recent works focus mostly on the hardware architecture design and experimental results, neglecting theoretical analyses. One exception is [11], which presented a theoretical framework for DNN inference on memristors based on tracking second-order statistical moments. However, they used a crossbar model based on passive summing circuits, rather than active ones as in this paper, and did not consider quantization in the conductance values. Furthermore, the accuracy of the method was only verified on very small DNNs.

This work was supported by an IVADO grant (PRF-2019-4784991664) and by the Samuel-de-Champlain program.

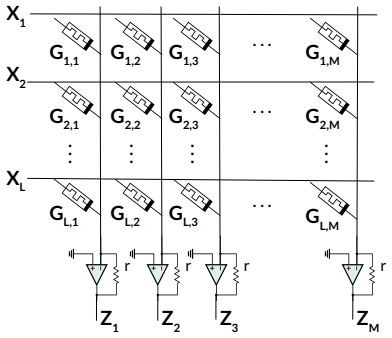

Fig. 1. Memristor crossbar architecture for matrix-vector multiplication.

In this work, we analytically study neural network inference on memristor crossbars. To this end, we provide theoretical computations, which take into account practical implementation non-idealities, such as programming noise and quantization errors. Using these computations, we predict the mean squared error (MSE) at the output of the final layer of a neural network, depending on the network's parameters and scaling factors. Theoretical formulas are also provided to compute the power usage of the memristors crossbars depending on the scaling factors. Finally, we combine these analyses to formulate an optimization problem to minimize the MSE for a desired power usage. Lastly, simulations are performed to verify the accuracy of the theoretical analysis.

II. Models

A. Memristor crossbar model

Figure 1 illustrates the architecture of the considered memristor crossbar. In accordance with Ohm's Law and Kirchoff's Law, the conductance at each node is multiplied with the input voltage of the row and these products are then summed

along the column. Finally, a transimpedance amplifier (TIA) converts the current into a voltage at end of each column. In the ideal case, the output of the j-th column is thus given by $z_j = r\sum_{i=1}^{L} g_{i,j}x_i$, where x_i is the voltage at the input of row i, $g_{i,j}$ is the conductance of the memristor at row i and column j, and r is the feedback resistance of the TIA.

However, several practical issues may cause the actual computation to differ from the aforementioned ideal case. Specifically, values may be affected by fabrication variations and noise during programming [12]–[14] as well as quantization errors. With these practical constraints in mind, we consider that the memristors have a conductance ranging from 0 to $G_{\max}$, and divide this range into N possible values. $G_{\max}$ is chosen depending on the desired trade-off between accuracy and power consumption and only needs to be inferior to the maximum physical conductance value. We denote the resulting resolution as $\Delta = \frac{G_{\max}}{N}$. The programmed conductance values can then be represented as random variables $G_{i,j}$, which may be decomposed as

$$G_{i,j} = g_{i,j} + \delta^q_{i,j} + \epsilon^v_{i,j}\,, \tag{1}$$

where $g_{i,j}$ is the desired value, $\delta^q_{i,j}$ is the quantization error, and $\epsilon^v_{i,j}$ is the noise due to variability in conductance programming. We denote by σ_v^2 the variance of ϵ^v. In practice, there can be different σ_v for each possible memristor value, but here, to simplify the notations and computations, we consider that σ_v is constant for all N possible conductance values. The analysis proposed in this paper remains valid in the case where σ_v is allowed to depend on the conductance value.

Since memristors can only store positive values, each weight $w_{i,j}$ is decomposed as $w_{i,j} = w^+_{i,j} - w^-_{i,j}$, where $w^+_{i,j}$ and $w^-_{i,j}$ store the positive and negative value of $w_{i,j}$, respectively. Then w^+ and w^- are converted to the conductance $g^+_{i,j}$ and $g^-_{i,j}$. The matrix-vector multiplication can then be realized as

$$Z_j = \sum_{i=1}^{L} rG^+_{i,j}X_i - \sum_{i=1}^{L} rG^-_{i,j}X_i\,, \tag{2}$$

where Z_j, $G^+_{i,j}$, $G^-_{i,j}$, and X_i are random variables.

B. Computation model

For our theoretical analysis, we consider a neural network composed of convolutional, average pooling, and linear layers as well as differentiable activation functions. For simplicity, we consider that all convolutional layers are converted to linear layers. Moreover, batch normalization is not considered but could be easily incorporated into our analysis.

Because of the range of conductance possible, the matrix weights $w_{i,j}$ is scaled by a factor $c = \frac{G_{\max}}{W_{\max}}$, such that

$$g_{i,j} = cw_{i,j}. \tag{3}$$

We then divide the result of the memristor computations by this same factor c. Denoting by $g^-_{i,j} = cw_{i,j} + \delta^q(cw_{i,j})$ the quantized version of g, where $\delta^q(cw_{i,j})$ is the (deterministic) quantization error, it should be noted that $\delta^q(cw_{i,j}) = c\delta^q(w_{i,j})$. Therefore $\frac{g^-_{i,j}}{c} = w_{i,j} + \delta^q(w_{i,j})$.

For a given linear layer, the conductance values $g_{i,j}$ are computed following (3) and uniformly quantized over the conductance range $[0, G_{\max}]$. Then, the memristors products $\tilde{Z_j}^+ = \sum_{i=1}^{L} rG^+_{i,j}x_i$ and $\tilde{Z_j}^- = \sum_{i=1}^{L} rG^-_{i,j}x_i$ are computed. The difference $\tilde{Z_j} = \tilde{Z_j}^+ - \tilde{Z_j}^-$, as well as its rescaling, $Z_j = \frac{\tilde{Z_j}}{c}$, is performed outside of the memristors crossbars. The non-linear activation function f is applied: $f(Z_j)$. Finally, an average pooling is applied as $A_{i,j} = \mathrm{Avg}(f(Z))_{i,j} = \frac{1}{s^2}\sum_{k=i}^{i+s}\sum_{l=j}^{j+s} f(Z_{k,l})$ where s is the kernel size.

III. Theoretical Analysis

A. MSE prediction

We now derive a theoretical analysis of the performance of a memristor-based implementation of neural network inference. As a proxy of performance, our goal is to predict the MSE between the noisy neural network outputs (computed using memristors) and the full precision (noiseless) outputs. We use the following notations throughout our analysis: $\mathrm{Var}[G_{i,j}] = \sigma^2$, $\mathrm{E}[X_i] = x_i$, $\mathrm{Var}[X_i] = \gamma_i^2$, $\mathrm{Cov}[X_i, X_j] = \gamma_{i,j}$.

The computation at the linear layer followed by the rescaling can be written as

$$Z_j = \frac{r}{c}\sum_{i=1}^{L} G_{i,j}X_i\,, \tag{4}$$

and we can formulate the first and second moments of Z_j as

$$\mu_j = \mathrm{E}[Z_j] = r\sum_{i=1}^{L}(w_{i,j} + \delta^q_{i,j})x_i\,, \tag{5}$$

$$\rho_j^2 = \mathrm{Var}[Z_j] = r^2\Bigg(\sum_{i=1}^{L}\frac{\sigma^2 x_i^2}{c^2} + \gamma_i^2(w_{i,j} + \delta^q_{i,j})^2 + \frac{\gamma_i^2\sigma^2}{c^2} + \sum_{i=1}^{L}\sum_{i'=1,i'\neq i}^{L}(w_{i,j} + \delta^q_{i,j})(w_{i'j} + \delta^q_{i',j})\gamma_{i,i'}\Bigg), \tag{6}$$

$$\rho_{j,j'} = \mathrm{Cov}[Z_j, Z_{j'}] = r^2\sum_{i=1}^{L}\sum_{i'=1}^{L}(w_{i,j}+\delta^q_{i,j})(w_{i'j'}+\delta^q_{i',j'})\gamma_{i,i'}\,. \tag{7}$$

Then, an approximation of the moments after the non-linear activation function f is possible via Taylor expansions [11]:

$$\mathrm{E}[f(Z_j)] \approx f(\mu_j) + \frac{1}{2}f''(\mu_j)\rho_j^2\,, \tag{8}$$

$$\mathrm{Var}[f(Z_j)] \approx \frac{1}{2}g''(\mu_j)\rho_j^2 - f(\mu_j)f''(\mu_j)\rho_j^2\,, \tag{9}$$

$$\mathrm{Cov}[f(Z_j), f(Z_{j'})] \approx f'(\mu_j)f'(\mu_{j'})\rho_{j,j'}\,, \tag{10}$$

where $g = f^2$. From these moments, the MSE of $f(Z_j)$ is

$$\mathrm{MSE}[f(Z_j)] = \mathrm{Var}[f(Z_j)] + (\mathrm{E}[f(Z_j)] - f(z_j))^2\,. \tag{11}$$

The MSE can also be expressed as a function of c as

$$\mathrm{MSE}[f(Z_j)] = \frac{F_{1,j}}{c^4} + \frac{F_{2,j}}{c^2} + F_{3,j} \tag{12}$$

The expressions of $F_{1,j}$, $F_{2,j}$, and $F_{3,j}$ can be computed by substituting (5) and (6) in (8) and (9). Note that if $c \to \infty$, then $\mathrm{MSE}[f(Z_j)] \to F_{3,j}$. Hence, $F_{3,j}$ gives us a lower bound on the MSE. Since this bound does not depend on σ, it is possible to find values of c for any σ that minimize the MSE to $F_{3,j}$.

For average pooling (Section II-B), the moments are

$$\mathrm{E}[A_{i,j}] = \frac{1}{s^2} \sum_{k=i}^{i+s} \sum_{l=j}^{j+s} \mu_{k,l} \,, \tag{13}$$

$$\mathrm{Var}[A_{i,j}] = \frac{1}{s^4} \sum_{k=i}^{i+s} \sum_{l=j}^{j+s} \sum_{m=i}^{i+s} \sum_{n=j}^{j+s} \gamma_{k,l,m,n} \,, \tag{14}$$

$$\mathrm{Cov}[A_{i,j}, A_{i',j'}] = \frac{1}{s^4} \sum_{k=i}^{i+s} \sum_{l=j}^{j+s} \sum_{m=i'}^{i'+s} \sum_{n=j'}^{j'+s} \gamma_{k,l,m,n} \,. \tag{15}$$

B. Power consumption

We now derive an estimation of the power consumption of the memristor computations. The power consumption of each memristor can be written as $P_{i,j}^{(\mathrm{mem})} = |G_{i,j}| \, X_i^2$, with

$$\mathrm{E}[P_{i,j}^{(\mathrm{mem})}] = \mathrm{E}[G_{i,j} X_i^2] = c \left| w_{i,j} + \delta_{i,j}^q \right| (\gamma_i^2 + x_i^2) \,. \tag{16}$$

Moreover, the power consumption of each transimpedance amplifier (TIA) is

$$P_j^{(\mathrm{TIA})^+} = \frac{(\sum_{i=1}^{L} G_{i,j}^+ X_i)^2}{r} = \frac{\tilde{Z}_j^{+^2}}{r^2} \tag{17}$$

and

$$P_j^{(\mathrm{TIA})^-} = \frac{(\sum_{i=1}^{L} G_{i,j}^- X_i)^2}{r} = \frac{\tilde{Z}_j^{-^2}}{r^2} \,, \tag{18}$$

with

$$\mathrm{E}[P_j^{(\mathrm{TIA})^+}] = c^2 \frac{\rho_i^{+^2} + \mu_i^{+^2}}{r^2} \,, \quad \mathrm{E}[P_j^{(\mathrm{TIA})^-}] = c^2 \frac{\rho_i^{-^2} + \mu_i^{-^2}}{r^2} \,. \tag{19}$$

Hence, the power consumption of each layer is

$$\mathrm{E}[P_{\mathrm{tot}}] = \sum_{j=1}^{L} \left(\sum_{i=1}^{L} \mathrm{E}[P_{i,j}^{(\mathrm{mem})}] + \mathrm{E}[P_j^{(\mathrm{TIA})^+}] + \mathrm{E}[P_j^{(\mathrm{TIA})^-}] \right) . \tag{20}$$

As a function of c, the power of each layer's column is

$$\mathrm{E}[P_{\mathrm{tot}_j}] = c^2 H_{1,j} + c H_{2,j} + H_{3,j} \,, \tag{21}$$

where the expressions of $H_{1,j}$, $H_{2,j}$, and $H_{3,j}$ can be computed by developing the terms of equation (20) from their definitions and equations (5) and (6).

IV. Optimization

The parameter $G_{\max}$ may be chosen with different granularity to balance design complexity and energy efficiency. For instance, one may apply the same $G_{\max}$ to all memristor crossbars, associate a specific $G_{\max}$ to each layer of the neural network, or even use a different $G_{\max}$ per crossbar column. We denote $\boldsymbol{G_{\max}}$ as the set of $G_{\max}$ variables that can be modified to optimize our computations. Depending if we have only one $G_{\max}$ for the whole network or one $G_{\max}$ for each layer, the size of $\boldsymbol{G_{\max}}$ is 1 or P, respectively.

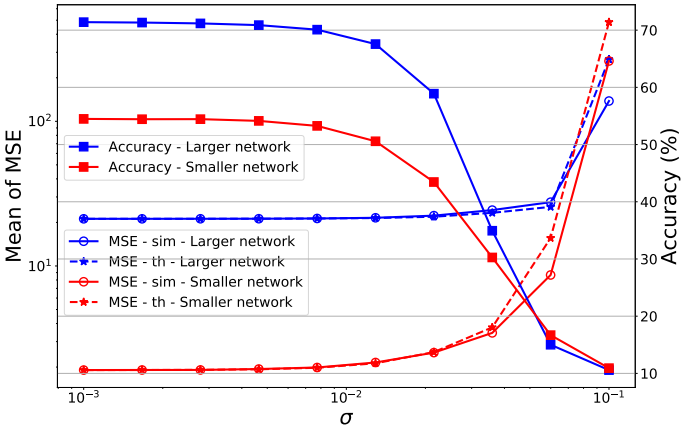

Fig. 2. MSE at the output of the final layers of the smaller and larger network averaged over input examples, in terms of the standard deviation σ of the conductance values.

To minimize the MSE for a specific power constraint, the global optimization problem can be formulated as

$$\min_{\boldsymbol{G_{\max}}} \max \mathrm{MSE}[f(Z^P)] \,, \tag{22}$$

subject to $\mathrm{E}[P_{\mathrm{tot}}] \leq \mathcal{V}$ and ${G_{\max}}^{(p)} > 0$. This corresponds to finding the best set of scaling constants c that minimizes MSE for a desired total power usage. The problem may be solved approximately using a heuristic optimizing search.

V. Simulations

We trained two convolutional neural networks on CIFAR-10 composed of five pairs of convolutional and average pooling layers and a final linear layer. Each subsequent convolutional layer in the smaller model has 2, 4, 8, 16, and 16 filters, as opposed to the 16, 32, 64, 128, and 128 filters of the larger model. We used a kernel size of 3 and a unit stride for all convolutional layers. Both models used the Softplus activation function and were trained for 164 epochs using stochastic gradient descent (SGD) with momentum, weight decay, and an initial learning rate of 0.1 (decayed by 10 at epochs 81 and 122). The number of quantized values N is set to 128 and r is set to 1.

A. Accuracy of the theoretical analysis

Figure 2 shows the mean of the MSE on the final layer outputs of the smaller and larger models computed over 100 different inputs. These values are plotted both based on simulations and on the analytical formula presented in Section III. We observe a close match between the theoretical and simulated MSE, especially in the high accuracy regime. Moreover, we see an inverse correlation between MSE and accuracy, which confirms predicting MSE to be a good proxy for estimating performance degradation. Moreover, as σ decreases, the MSE converges to a value dependent on the quantization error.

With a Tesla P100 GPU, the mean runtime for the MSE computation of the small network on a batch of 64 inputs with $\sigma = 0.01$ using our method is 27 ms. Under the same

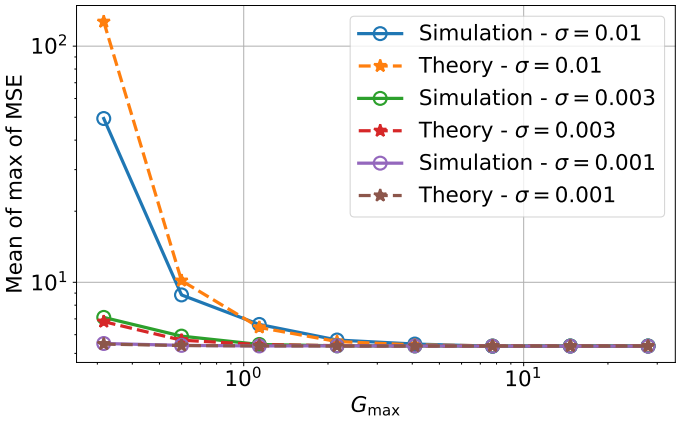

Fig. 3. Mean of the maximum of the MSE of the output of the smaller network's final layer depending on $G_{\max}$ and σ values.

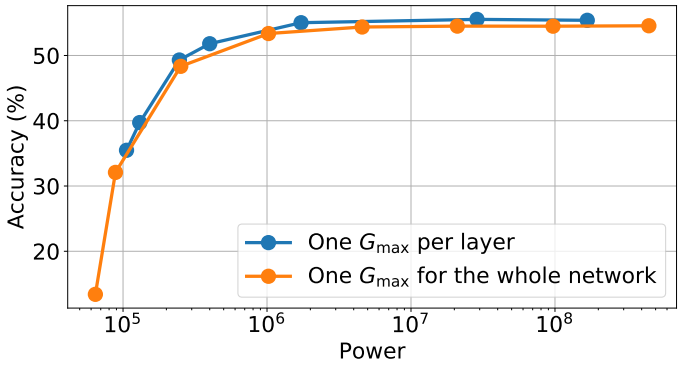

Fig. 4. Maximization of the smaller network accuracy using a genetic algorithm minimizing the maximum of the MSE for different power constraints with $\sigma = 0.01$

conditions, using 200 Monte-Carlo trials takes on average 2.3 seconds to reach a MSE within 2% of the true MSE 98% of the time. This 85× speedup showcases the usefulness of our method in practice.

Figure 3 shows the mean of the maximum MSE on the output of the smaller model depending on the value of $G_{\max}$, for different values of σ. Once again, we observe that the theoretical computations accurately predict the simulation results. Moreover, we see the predicted convergence to the theoretical bound. Such bound is reached faster as σ decreases. In Figures 2 and 3, we notice that for a high ratio of noise to $G_{\max}$ there is a gap between theoretical and simulation results. This is likely due to the Taylor expansions used for approximating the moments after the activation function.

B. Numerical optimization

Figure 4 shows the results of optimizing $\boldsymbol{G_{\max}}$ following (22) for the smaller network. For each power constraint, a genetic optimizer was run for 100 generations with a population size of 50 and a sample of 100 inputs for computing the mean of the theoretical MSE and power consumption of the network. The proposed approach allows to efficiently find the value(s) of $G_{\max}$ that minimize MSE (maximize accuracy) for a given power constraint. As expected, adding degrees of freedom by allowing a different $G_{\max}$ for each layer leads to improved performance, although the benefit is marginal in this case.

VI. Conclusion

In this work, we studied the implementation of DNN models using memristors crossbars. Using second-degree Taylor expansions, we proposed approximate theoretical formulas of the MSE at the output of the network, as well as theoretical computations of the power usage of the memristors. We then considered an optimization problem for maximizing task performance under a power usage constraint. The theoretical analysis makes it feasible to solve this optimization problem numerically since its computing time is faster than using simulations by almost two orders of magnitude.

References

[1] A. Pedram, S. Richardson, M. Horowitz, S. Galal, and S. Kvatinsky, "Dark memory and accelerator-rich system optimization in the dark silicon era," *IEEE Design Test*, 2017.

[2] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature Nanotechnology*, 2020.

[3] A. Chen and M.-R. Lin, "Variability of resistive switching memories and its impact on crossbar array performance," in *Int. Reliability Physics Symp.*, 2011.

[4] E. Pérez, C. Zambelli, M. K. Mahadevaiah, P. Olivo, and C. Wenger, "Toward reliable multi-level operation in RRAM arrays: Improving post-algorithm stability and assessing endurance/data retention," *IEEE Journal of the Electron Devices Society*, 2019.

[5] 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*, 2018.

[6] S. Diware, A. Gebregiorgis, R. V. Joshi, S. Hamdioui, and R. Bishnoi, "Unbalanced bit-slicing scheme for accurate memristor-based neural network architecture," in *IEEE Int. Conf. on Artificial Intelligence Circuits and Systems*, 2021.

[7] S. Henwood, F. Leduc-Primeau, and Y. Savaria, "Layerwise noise maximisation to train low-energy deep neural networks," in *IEEE Int. Conf. on Artificial Intelligence Circuits and Systems*, 2020.

[8] G. B. Hacene, F. Leduc-Primeau, A. B. Soussia, V. Gripon, and F. Gagnon, "Training modern deep neural networks for memory-fault robustness," in *IEEE Int. Symp. on Circuits and Systems*, 2019.

[9] T. Hirtzlin, M. Bocquet, J.-O. Klein, E. Nowak, E. Vianello, J.-M. Portal, and D. Querlioz, "Outstanding bit error tolerance of Resistive RAM-based binarized neural networks," in *IEEE Int. Conf. on Artificial Intelligence Circuits and Systems*, 2019.

[10] 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*, 2018.

[11] E. Dupraz, L. R. Varshney, and F. Leduc-Primeau, "Power-efficient deep neural networks with noisy memristor implementation," in *IEEE Information Theory Workshop*, 2021.

[12] S. Liu, Y. Wang, M. Fardad, and P. K. Varshney, "A memristor-based optimization framework for artificial intelligence applications," *IEEE Circuits and Systems Magazine*, 2018.

[13] A. J. Pérez-Ávila, G. González-Cordero, E. Pérez, E. P.-B. Quesada, M. Kalishettyhalli Mahadevaiah, C. Wenger, J. B. Roldán, and F. Jiménez-Molinos, "Behavioral modeling of multilevel HfO2-based memristors for neuromorphic circuit simulation," in *Conf. on Design of Circuits and Integrated Systems*, 2020.

[14] V. Milo, C. Zambelli, P. Olivo, E. Pérez, M. K. Mahadevaiah, O. G. Ossorio, C. Wenger, and D. Ielmini, "Multilevel HfO2-based RRAM devices for low-power neuromorphic networks," *APL Materials*, 2019.