Channel Coding for Binary Neural Networks Implemented with Noisy Memristor Crossbars

Elsa Dupraz[1] and François Leduc-Primeau[2]
[1] IMT Atlantique, CNRS UMR 6285, Lab-STICC, Brest, France
[2] Department of Electrical Engineering, Polytechnique Montreal, Montreal, QC, Canada

***Abstract*—In the context of in-memory computing, this paper investigates binary neural networks (BNNs) implemented with memristor crossbars. A key issue in these architectures is the difficulty in setting crossbar conductance values with an arbitrary precision, which introduces computational noise into the BNN layers. The paper first introduces a theoretical analysis of the impact of noise on the final computation, deriving an analytical expression for the error probability at the output of a BNN layer. This expression is validated through Monte-Carlo simulations, demonstrating its accuracy in predicting error probabilities. Furthermore, the paper introduces a novel linear block code designed to mitigate the effects of noise in conductance values. This coding scheme is accompanied by a dedicated low-complexity belief propagation decoder that operates over the set of integers. Simulation results indicate a significant improvement in bit error rate for the proposed coding method compared to hard thresholding.**

I. Introduction

In recent years, in-memory computing has emerged as a promising paradigm in hardware implementation, offering the potential to significantly mitigate data transfer bottlenecks between circuit memories and processors. This advancement is particularly crucial for artificial intelligence systems, where such transfers often represent a major impediment to efficiency. Especially, memristor crossbars [1] have attracted attention for their ability to perform dot-product computations in memory, achieved by mapping matrix components to internal memristor conductance values [2], [3]. This innovative approach has been demonstrated in proof-of-concept studies for both deep neural network (DNN) training [4] and inference tasks [5], [6].

In this work, we specifically investigate binary neural networks (BNNs) [7] implemented from memristor crossbars. BNNs are characterized by their binary weight matrices, where the output of each layer is determined through XNOR and PopCount operations. This unique architecture allows for very efficient hardware implementation while maintaining sufficient learning performance. Specifically, our focus lies on the memristor crossbar-based BNN architecture proposed in [8]. In this design, each memristor can take only two values, denoted as g_{ON} and g_{OFF} and each layer is implemented from two crossbars that realize the XNOR and PopCount operations.

However, achieving precise internal conductance values in memristor crossbars is challenging, as highlighted in [2], [3]. This raises a critical question regarding the robustness of in-memory computing with respect to the noise introduced in these conductance values. This issue has been addressed recently in [9]–[11] for full DNNs, considering either quantized [10] or analog [9], [11] conductance values. Theoretical analyses conducted in these works evaluate the mean-squared error (MSE) in the presence of noise. However, the MSE criterion is not directly relevant in the context of BNNs, which exchange binary values between layers. This is why, in this paper, we consider instead an error probability criterion.

In addition, the level of noise affecting the crossbar may sometimes be too high to obtain adequate inference performance. We thus propose in this paper a new channel code construction that can improve the robustness to noise of the BNN. We then rely on a channel decoder to retrieve the correct output of the computation. The main difficulty is to develop a method that preserves the code structure when performing dot-product computation over integer or real values.

In [12], it was proposed to use non-binary LDPC codes for robust dot-product computation from memristor crossbars. The method of [12] considers a bit-slicing structure in which the computation on each bit position is realized by a dedicated crossbar. In addition, it considers one non-binary LDPC code per crossbar, where the order of the non-binary LDPC code is large enough so that the computation cannot be saturated. Therefore, the computation structure of [12] is much more complex than the BNN computation structure of [8], which only relies on two crossbars. In addition, [13] proposed an analog code construction which can correct one single outlying error in the crossbar output vector. In practice, a larger amount of errors, and not only outliers, should be corrected.

In this paper, we first introduce a theoretical analysis to evaluate the effect of noise onto the BNN computation, by providing an analytical expression of the error probability after a BNN layer. We then propose a linear block code construction that aims to correct errors introduced by the noise in the conductance values. The code parity-check matrix has non-zero component values in the alphabet $\{-1, +1\}$ and it is sparse. Therefore, we develop a dedicated belief propagation (BP) decoder that works over the integer set $\mathbb{Z}$. Simulation results show the efficiency of the proposed coding solution at correcting computation errors.

The outline of the paper is as follows. Section II presents BNN computation from noisy memristor crossbars. Section III carries the theoretical analysis. Section IV introduces the proposed code construction, and Section V describes an associated decoding algorithm. Section VI discusses the simulation results and Section VII concludes the paper.

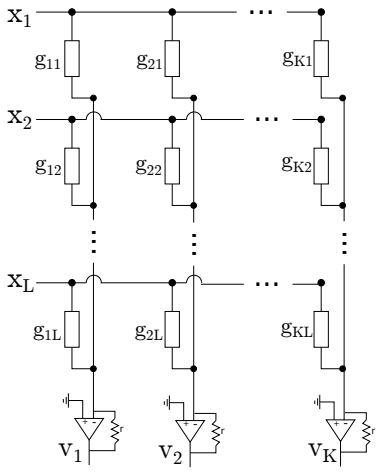


Fig. 1. Circuit diagram of the memristor crossbar.

II. BNNs Implemented with memristor crossbars

In this section, we first introduce generic dot-product computation from memristor crossbars. We then describe how memristor crossbars can be used to implement BNNs.

A. Dot-Product Computation from Memristor Crossbars

We consider the memristor crossbar electronic model of [14], [15], shown in Figure 1. In this model, we use u_i to denote the input voltage on row i. Then, a memristor with conductance value $g_{i,j}$ connects the i-th row to the j-th column, with $i \in [\![1, L]\!]$, $j \in [\![1, K]\!]$. Each column j ends with a transimpedance amplifier (TIA), which transforms the column current into an output voltage v_j. We use r to denote the feedback resistance of the TIA. By resorting to Ohms and Kirchhoff Laws, the outputs v_j can be expressed as

$$\forall j \in [\![1, K]\!], \quad v_j = r \sum_{i=1}^{L} g_{i,j} x_i \,. \tag{1}$$

In addition, as in [8], [16], we consider only two possible levels g_{ON} and g_{OFF} for the conductance values $g_{i,j}$, where $g_{\text{ON}} > g_{\text{OFF}}$.

B. BNNs

We consider a single layer of a feedforward BNN, having a weight matrix W of size $L \times K$, an input vector $\mathbf{x}$ of length L, and an output vectors $\mathbf{z}$ of length K. In BNNs, the weight matrix W and the vectors $\mathbf{x}$, $\mathbf{z}$, are all binary with component values in the alphabet $\{-1, 1\}$. As a result, all computation operations should be adapted to remain in the binary domain [17]. Here, we consider the binary computation of [7], in which for all $j \in [\![1, K]\!]$, each output activation z_j is evaluated as

$$z_j = \text{sign}\left(\sum_{i=1}^{L} w_{i,j} \odot x_i \right) , \tag{2}$$

where $\odot$ is the XNOR operation restated in Table I.

TABLE I
XNOR operation between $w_{i,j}$ and x_i

$w_{i,j}$	x_i	$w_{i,j} \odot x_i$
$+1$	$+1$	$+1$
$+1$	-1	-1
-1	$+1$	-1
-1	-1	$+1$

C. BNNs with Memristor Crossbars

As proposed in [8], we can use two memristor crossbars so as to compute output voltages y_j as

$$y_j = r \sum_{i=1}^{L} \left(g_{i,j}^{(+)} - g_{i,j}^{(-)} \right) x_i \,, \tag{3}$$

where one crossbar contains the values $g_{i,j}^{(+)}$, and the other contains the values $g_{i,j}^{(-)}$. We further set the values $g_{i,j}^{(+)}$ and $g_{i,j}^{(-)}$ from the equation $(g_{i,j}^{(+)} - g_{i,j}^{(-)}) = w_{i,j}(g_{\text{ON}} - g_{\text{OFF}})$. The activation z_j is then obtained by binarizing y_j:

$$z_j = \text{sign}\left(y_j\right) \,. \tag{4}$$

It can be shown that (4) is equivalent to (2).

D. Stochastic Computation Model

One issue with the computation model of (3) and (4) is that it is difficult to set-up conductance values $g_{i,j}^{(+)}$ and $g_{i,j}^{(-)}$ with an arbitrary precision [18]. As a result, in what follows, we consider random variables $G_{i,j}^{(+)}$ and $G_{i,j}^{(-)}$ to represent noisy conductance values, where each $G_{i,j}^{(+)}$ or $G_{i,j}^{(-)}$ is assumed Gaussian with mean $g_{i,j} \in \{g_{\text{ON}}, g_{\text{OFF}}\}$ and variance σ^2 [3], [19]. With this notation, the stochastic version of (3) and (4) can be written respectively as

$$Y_j = r \sum_{i=1}^{L} \left(G_{i,j}^{(+)} - G_{i,j}^{(-)} \right) X_i \tag{5}$$

and $Z_j = \text{sign}(Y_j)$. In the next section, we develop a theoretical analysis so as to evaluate the effect of noisy conductance values $G_{i,j}^{(+)}$ and $G_{i,j}^{(-)}$ on the computation of Z_j. To perform this analysis, we assume that the targeted conductance values $g_{i,j}$ are fixed, since in the inference phase of a BNN, the weight matrix is always the same at a given layer. On the contrary, given that input vectors to the layer will change from one computation to another, we consider that input voltages X_i are binary random variables taking values in the alphabet $\{-V, +V\}$ with $\mathbb{P}(X = +V) = q$.

III. Theoretical Analysis

In this section, we theoretically evaluate the effect of noise onto the crossbar computation. We first use the MSE criterion which was considered for DNNs in [10], and then evaluate the error probability, which is more relevant for BNNs.

A. Mean-Squared Error of Crossbar Computation

Following the same derivation as in [10], we can express the expectation $\mu_j = \mathbb{E}[Y_j]$ and variance $\gamma_j^2 = \mathbb{V}[Y_j]$ of Y_j as

$$\mu_j = rV(2q-1)\sum_{i=1}^{L}(g_{i,j}^{(+)} - g_{i,j}^{(-)})\,, \tag{6}$$

$$\gamma_j^2 = 2r^2L\sigma^2V^2 + 4r^2V^2q(1-q)L(g_{\text{ON}} - g_{\text{OFF}})^2\,. \tag{7}$$

In addition, given that the terms in the sum in (5) are independent, and considering that L is large enough, we can apply the Central Limit Theorem to show that Y_j is asymptotically Gaussian, with mean μ_j and variance γ_j^2.

We can then use μ_j and γ_j^2 to express the MSE at the output of the crossbar as

$$\text{MSE}(Y_j) = \gamma_j^2 + \mathbb{E}\left[(\mu_j - y_j)^2\right], \tag{8}$$

with

$$\mathbb{E}\left[(\mu_j - y_j)^2\right] = 4r^2\left(\sum_{i=1}^{L}(g_{i,j}^{(+)} - g_{i,j}^{(-)})\right)^2 V^2q(1-q). \tag{9}$$

The MSE can be used as a criterion to evaluate the effect of noise onto the crossbar computation. In addition, the moment expressions in (6), (7), will be used to initialize the BP decoder introduced in Section V.

B. Error Probability

As a more relevant criterion for BNNs, we can consider the error probability p_e of the BNN computation, which is defined as

$$p_e \triangleq \mathbb{P}(Z_j \neq z_j) = \mathbb{P}\left(\text{sign}(Y_j) \neq \text{sign}(y_j)\right). \tag{10}$$

To evaluate p_e, we first remark that Y_j in (5) can be rewritten as

$$\begin{aligned} Y_j &= r\left(\sum_{i=1}^{L}\left(g_{i,j}^{(+)} - g_{i,j}^{(-)}\right)X_i + \sum_{i=1}^{L}\left(N_{i,j}^{(+)} - N_{i,j}^{(-)}\right)X_i\right) \\ &\triangleq r(U_j + N_j)\,, \end{aligned} \tag{11}$$

where $N_{i,j}^{(+)}$ and $N_{i,j}^{(-)}$ follow Gaussian distributions $\mathcal{N}(0,\sigma^2)$. The random variable U_j is discrete and takes values $u = (-L+2k)V(g_{\text{ON}} - g_{\text{OFF}})$, where $k \in [\![0, L]\!]$ and

$$\mathbb{P}\left(U_j = u\right) = \binom{L}{k}q^k(1-q)^{L-k}\,. \tag{12}$$

In addition, since $N_{i,j}^{(+)}$ and $N_{i,j}^{(-)}$ are Gaussian random variables with zero mean, by resorting to the characteristic function, we can show that N_j follows a Gaussian distribution $\mathcal{N}(0, 2V^2L\sigma^2)$.

The error probability p_e can then be expressed as

$$p_e = \mathbb{P}(-N_j < U_j < 0) + \mathbb{P}(0 < U_j < -N_j) + \frac{1}{2}\mathbb{P}(U_j = 0)\,.$$

In this expression, $\mathbb{P}(U_j = 0)$ comes from (12), and

$$\mathbb{P}(-N_j < U_j < 0) = \sum_{u<0}\mathbb{P}(U_j = u)\frac{1}{2}\left(1 - \Phi\left(\frac{-u}{\sqrt{2LV^2\sigma^2}}\right)\right),$$

$$\mathbb{P}(0 < U_j < -N_j) = \sum_{u>0}\mathbb{P}(U_j = u)\frac{1}{2}\left(1 - \Phi\left(\frac{u}{\sqrt{2LV^2\sigma^2}}\right)\right),$$

where Φ is the Gaussian error function. This completes the calculation of the error probability, which can then be evaluated numerically directly from the previous formulas.

IV. Channel Coding

We now propose a channel code construction to lower the effect of noise at the output of a BNN layer. Note that the code is for Y in (5). Indeed, we cannot decode the outputs Z directly, due to the non-linearity introduced by the sign operator.

A. Code Construction

We consider a linear block code with generator matrix C of size $K \times N$ and parity-check matrix H of size $M \times N$ with $M = N - K$. Here, we consider that the components of both matrices C and H take values in the alphabet $\{-1, 0, +1\}$. This constraint will allow us to derive an integer BP decoder, presented in Section V. We then set

$$C = [\text{I}_K, -D] \tag{13}$$

$$H = [D^T, \text{I}_M]\,, \tag{14}$$

where I_K and I_M are identity matrices of size $K \times K$ and $M \times M$, respectively, and D is a matrix of size $K \times (N - K)$ with components in the alphabet $\{-1, 0, +1\}$. From (13) and (14), we see that

$$CH^T = 0\,. \tag{15}$$

Contrary to standard linear block codes, this expression is evaluated from the standard matrix multiplication in $\mathbb{R}$, rather than using finite-field algebra, which will allow preserving the code structure through the sum operation in (5).

In what follows, we further assume that the parity-check matrix H is sparse. The sparsity condition together with the fact that the components of H take values in $\{-1, 0, +1\}$ will allow us to develop a low-complexity BP decoder. In our simulations, the matrix H will be constructed as an LDGM code [20], [21], and C will be obtained after identifying the matrix D in H.

B. Row Encoding

In our construction, as in [13], the rows of the crossbar are encoded separately in the sense that for row i, $(N - K)$ additional conductance values $\tilde{g}_{i,j}$ will be calculated, with $j \in [\![K+1, N]\!]$. Denote by $\mathbf{g}_i = [g_{i,1}, \cdots, g_{i,K}]$ the i-th row of the crossbar. Row encoding consists of computing the L vectors $\tilde{\mathbf{g}}_i = \mathbf{g}_i C$ of length N. Note that following the model in Section II, the vector elements $g_{i,j}$ take values in the set $\{g_{\text{ON}}, g_{\text{OFF}}\}$. According to (15), we show that

$$\tilde{\mathbf{g}}_i H^T = 0\,, \tag{16}$$

which of course implies that $x_i \tilde{\mathbf{g}}_i H^T = 0$.

We then implement the BNN weight matrix computation from two crossbars of size $L \times N$ that contain the conductance values defined in the L vectors $\tilde{\mathbf{g}}_i$. We now consider that the crossbar outputs a noisy vector $\tilde{\mathbf{Y}}$ of length N from the operation defined in (5).

C. Linear Combinations of Codewords

We use $\tilde{\mathbf{y}}$ to denote the vector of length N with components $\tilde{y}_j = \sum_{i=1}^{L} \tilde{g}_{i,j} x_i$. We now show that with the above code construction, the vector $\tilde{\mathbf{y}}$ is a codeword. Consider the product $\tilde{\mathbf{y}} H^T = \mathbf{u}$, where $\mathbf{u}$ is a vector of length M. For $m \in [\![1, M]\!]$,

$$u_m = \sum_{j=1}^{N} \tilde{y}_j h_{m,j} = \sum_{j=1}^{N} \left(\sum_{i=1}^{L} \tilde{g}_{i,j} x_i \right) h_{m,j} \,. \tag{17}$$

Since the two sums are standard sums over $\mathbb{R}$, they commute. Therefore, (16) allows us to write

$$u_m = \sum_{i=1}^{L} x_i \sum_{j=1}^{N} \tilde{g}_{i,j} h_{m,j} = 0 \,. \tag{18}$$

This shows that $\tilde{\mathbf{y}}$ is a codeword. Therefore, we now introduce a BP decoder so as to retrieve $\tilde{\mathbf{y}}$ from the noisy output $\tilde{\mathbf{Y}}$.

V. Integer BP decoder

Given that the matrix H is sparse, we can derive a BP decoder so as to correct errors in the noisy vector $\tilde{\mathbf{Y}}$. The decoder we propose is close in essence to a non-binary LDPC decoder, except that it manipulates integers over $\mathbb{Z}$. Since $\mathbb{Z}$ is not a ring, our BP decoder only works given that the components of H take values in the alphabet $\{-1, 0, +1\}$.

A. Tanner Graph

The parity-check matrix H can be represented by a Tanner graph that connects N variable nodes (VN) to M check nodes (CN). In the Tanner graph, there is an edge between VN n and CN m if the component $H_{m,n}$ is non-zero, and this edge has label $H_{m,n}$. The set of CNs connected to VN n is denoted $\mathcal{C}_n$, and the set of VNs connected to CN m is denoted $\mathcal{V}_m$.

B. Decoder Initialization

To initialize the decoder, we construct N message vectors $\mathbf{m}_j^{(0)}$ of length $2\Delta+1$, where Δ is a parameter of the decoder. The value of Δ should be chosen so that $2\Delta + 1$ covers the full range of $\tilde{y}_j$. For all $j \in [\![1, N]\!]$, the message vector $\mathbf{m}_j^{(0)}$ is calculated as

$$\forall \delta \in [\![-\Delta, \Delta]\!], m_j^{(0)}(\delta) = \log \frac{\mathbb{P}(\tilde{Y}_j | \tilde{y}_j = 0)}{\mathbb{P}(\tilde{Y}_j | \tilde{y}_j = \delta)} = \frac{\delta^2 - 2\tilde{Y}_j}{2\gamma_j^2} \,.$$

The second equality is obtained by considering that $\tilde{Y}_j$ is Gaussian, as discussed in Section III-A. The expression for γ_j^2 is provided in (7).

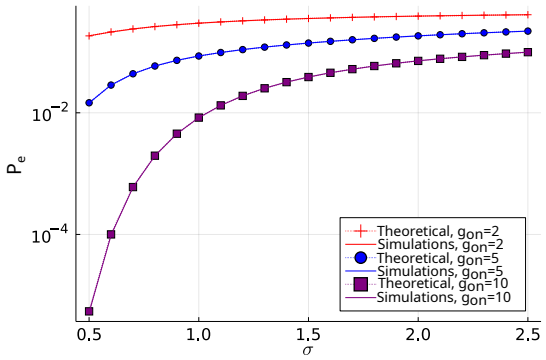


Fig. 2. Comparison of the error probability p_e obtained from the theoretical analysis and from Monte-Carlo simulations.

C. Message Computation

The decoder works in Λ iterations. At iteration $\ell \geq 1$, the message vector from VN n to CN m is denoted $\boldsymbol{\eta}_{n,m}^{(\ell)}$, and the message vector from CN m to VN n is denoted $\boldsymbol{\nu}_{m,n}^{(\ell)}$. CN messages $\boldsymbol{\nu}_{m,n}^{(\ell)}$ are computed in three steps. We first form messages $\boldsymbol{p}_{n,m}^{(\ell)}$ as

$$\forall \delta \in [\![-\Delta, \Delta]\!], \quad p_{n,m}^{(\ell)}(\delta) = \frac{\exp\left(-\eta_{n,m}^{(\ell-1)}(H_{m,n}\delta)\right)}{\sum_{\delta'=-\Delta}^{\Delta} \exp\left(-\eta_{n,m}^{(\ell-1)}(\delta')\right)} \,. \tag{19}$$

We then compute messages $\boldsymbol{q}_{m,n}^{(\ell)}$ as

$$\boldsymbol{q}_{m,n}^{(\ell)} = \mathrm{FFT}^{-1}\left(\prod_{n' \in \mathcal{V}_m \setminus n} \mathrm{FFT}(\mathbf{p}_{n,m}^{(\ell)}) \right) , \tag{20}$$

where the operator FFT is the standard Fast Fourier Transform. At the third step, we compute messages $\boldsymbol{\nu}_{m,n}^{(\ell)}$ as

$$\forall \delta \in [\![-\Delta, \Delta]\!], \quad \nu_{n,m}^{(\ell)}(\delta) = \log \frac{q_{m,n}^{(\ell)}(0)}{q_{m,n}^{(\ell)}(H_{m,n}\delta)} \,. \tag{21}$$

VN messages $\boldsymbol{\eta}_{n,m}^{(\ell)}$ are then computed as

$$\boldsymbol{\eta}_{n,m}^{(\ell)} = \sum_{m' \in \mathcal{C}_n \setminus m} \boldsymbol{\nu}_{m,n}^{(\ell)} \,. \tag{22}$$

At the end of iteration ℓ, we set the estimated value $\hat{y}_n$ to be the value δ that minimizes the sum of messages $\sum_{m' \in \mathcal{C}_n} \nu_{m,n}^{(\ell)}(\delta)$ incoming to node n. The decoder stops after Λ iterations, or if the stopping criterion $\hat{\mathbf{y}} H^T = 0$ is satisfied. This decoder is similar to a non-binary LDPC BP decoder. The message initialization as well as VN and CN message calculation are identical, except that the standard FFT is considered in (20), instead of the Hadamard transform.

VI. Simulation Results

A. Accuracy of the Theoretical Analysis

We first verify the accuracy of the theoretical analysis proposed in Section III-B, without considering channel coding.

We generate a binary weight matrix with random components and size $K = 10$, $L = 1000$, and we set parameters $q = 0.8$, $r = 1$, $V = 1$, $g_{\text{OFF}} = 1$. In Figure 2, we show the error probability p_e at the output of the layer with respect to σ, by considering different values for g_{ON}. In this figure, p_e was either calculated from the theoretical analysis, or measured from Monte-Carlo simulations. In these simulations, p_e for a given pair (σ, g_{ON}) was obtained by averaging over 1000 simulations. We first observe the accuracy of the proposed theoretical analysis, in the sense that in all the considered cases, the curves for the theoretical p_e and for the simulated p_e are superimposed. We also observe that as expected, p_e increases as σ increases, and decreases as g_{ON} increases.

B. Code Performance

for a fixed $\sigma^2 = 1.0.4$t

We now evaluate the performance of the proposed coding scheme, in terms of BER. We consider the same model as in Section VI-A for the binary weight matrix. We consider three setups, which correspond to codes of different length N. In the first setup, we consider a crossbar with $K = 9$ and $L = 10$, and the LDGM code construction of [21], which contains no short cycles of length 4, with $N = 15$ and code rate $R = 0.6$. In the second setup, we consider a crossbar with $K = 144$ and $L = 10$, and an LDGM code with $N = 180$ and $R = 0.6$. In the third setup, we consider a crossbar with $K = 288$ and $L = 10$, and an LDGM code with $N = 360$ and $R = 0.6$. The second and third codes were obtained by quasi-cyclic (QC) extension of the first code, where the QC coefficients are chosen so as to limit the number of short cycles in H. For the BP decoder, we consider a message size $\Delta = 100$, and $\Lambda = 10$ iterations.

Figure 3 shows the BER with respect to g_{ON}, for the three setups and for fixed values $g_{\text{OFF}} = 1$ and $\sigma^2 = 1$. We observe that the BER after decoding is significantly smaller than the BER achieved without a channel code, by about two orders of magnitude in this example. In addition, as expected, we further observe an additional BER gain when the code length N increases.

VII. Conclusion

In this paper, we considered BNNs implemented from noisy memristor crossbar, and provided accurate numerical expressions of the error probability at the output of a BNN layer. We then developed linear block code constructions together with an integer BP decoder in order to correct error introduced by the noise in the computation. Future works will include considering a full BNN with several layers, and evaluating the overhead in terms of hardware complexity and energy consumption induced by the proposed code construction.

References

[1] D. B. Strukov, G. S. Snider *et al.*, "The missing memristor found," *Nature*, vol. 453, no. 7191, p. 80, 2008.

[2] S. Liu, Y. Wang *et al.*, "A memristor-based optimization framework for artificial intelligence applications," *IEEE Circuits and Systems Magazine*, vol. 18, no. 1, pp. 29–44, 2018.

[3] E. Dupraz, F. Leduc-Primeau *et al.*, "Turning to information theory to bring in-memory computing into practice," *IEEE BITS the Information Theory Magazine*, vol. 3, no. 3, pp. 64–77, 2023.

[4] S. Ambrogio, P. Narayanan *et al.*, "Equivalent-accuracy accelerated neural-network training using analogue memory," *Nature*, vol. 558, no. 7708, pp. 60–67, 2018.

[5] V. Joshi, M. Le Gallo *et al.*, "Accurate deep neural network inference using computational phase-change memory," *Nature communications*, vol. 11, no. 1, pp. 1–13, 2020.

[6] A. Ankit, I. E. Hajj *et al.*, "Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference," in *Proc. of the Twenty-Fourth Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 715–731.

[7] I. Hubara, M. Courbariaux *et al.*, "Binarized neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[8] Y. Kim, W. H. Jeong *et al.*, "Memristor crossbar array for binarized neural networks," *AIP Advances*, vol. 9, no. 4, p. 045131, 2019.

[9] E. Dupraz and L. R. Varshney, "Noisy in-memory recursive computation with memristor crossbars," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 804–809.

[10] J. Kern, S. Henwood *et al.*, "MemSE: Fast MSE prediction for noisy memristor-based DNN accelerators," in *IEEE Int. Conf. on Artificial Intelligence Circuits and Systems*, 2022.

[11] E. Dupraz, L. R. Varshney, and F. Leduc-Primeau, "Power-efficient deep neural networks with noisy memristor implementation," in *2021 IEEE Information Theory Workshop (ITW)*. IEEE, 2021, pp. 1–5.

[12] Q. Lou, T. Gao *et al.*, "Embedding error correction into crossbars for reliable matrix vector multiplication using emerging devices," in *Proc. of the ACM/IEEE Int. Symp. on Low Power Electronics and Design*, 2020, pp. 139–144.

[13] R. M. Roth, "Analog error-correcting codes," *IEEE Transactions on Information Theory*, vol. 66, no. 7, pp. 4075–4088, 2020.

[14] M. Hu, J. P. Strachan *et al.*, "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," in *53nd Design Automation Conference (DAC)*, 2016.

[15] C. Li, M. Hu *et al.*, "Analogue signal and image processing with large memristor crossbars," *Nature Electronics*, 2018.

[16] S. Kvatinsky, G. Satat *et al.*, "Memristor-based material implication (imply) logic: Design principles and methodologies," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, pp. 2054–2066, 2013.

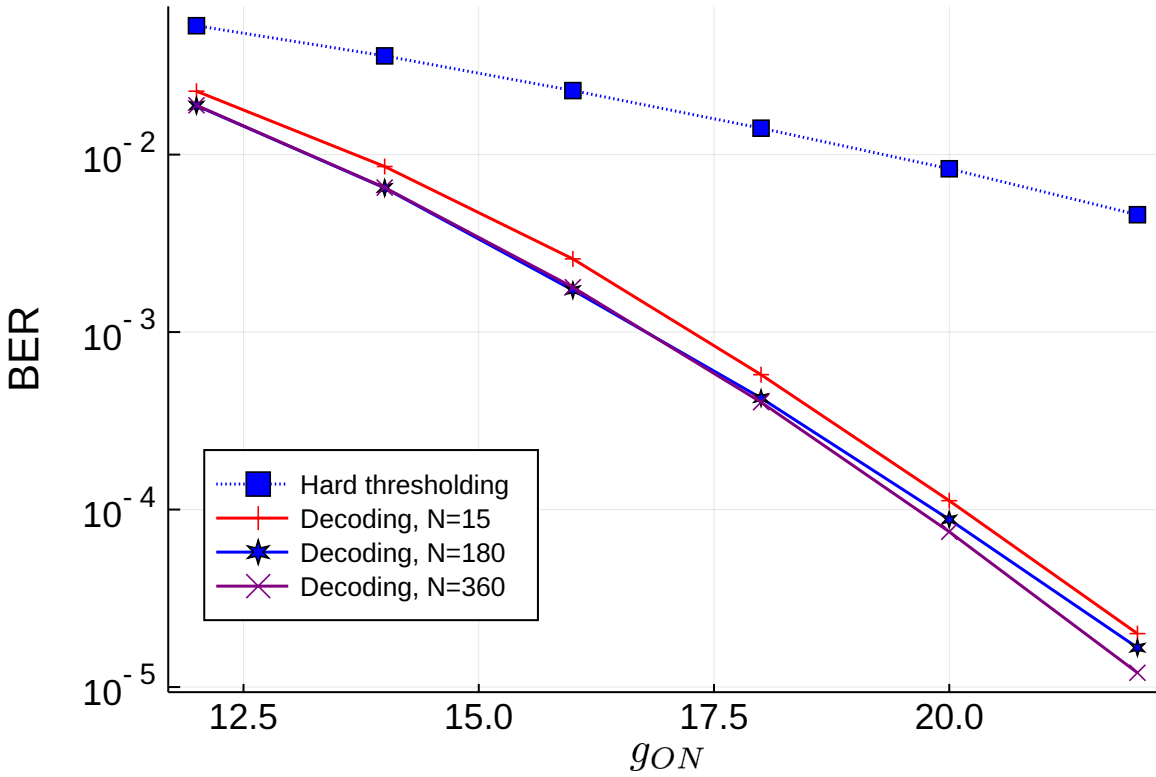
[17] T. Simons and D.-J. Lee, "A review of binarized neural networks," *Electronics*, vol. 8, no. 6, p. 661, 2019.

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

[19] S. Jain, A. Ranjan *et al.*, "Computing in memory with spin-transfer torque magnetic RAM," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 3, pp. 470–483, 2017.

[20] J. Garcia-Frias and W. Zhong, "Approaching shannon performance by iterative decoding of linear codes with low-density generator matrix," *IEEE Communications Letters*, vol. 7, no. 6, pp. 266–268, 2003.

[21] P. Suthisopapan, M. Kupimai *et al.*, "Design of high-rate LDGM codes," in *4th Int. Conf. on Communications and Networking in China*, 2009.


BER
10^-2
10^-3
10^-4
10^-5
12.5
15.0
17.5
20.0
g_{ON}
Hard thresholding
Decoding, N=15
Decoding, N=180
Decoding, N=360