Compressed vector-matrix multiplication for Memristor-based ensemble neural networks

Correspondance: phan-anh.vu@cea.fr, thomas.dalgaty@cea.fr

Phan Anh Vu*, François Rummens*, Marielle Malfante*, Bertrand Rivet†, Thomas Dalgaty*

* CEA-List, Université Grenoble-Alpes, Grenoble, France

† Université Grenoble-Alpes, CNRS, Grenoble INP, GIPSA-lab, Grenoble, France

***Abstract*—Using an ensemble of neural networks is an effective means of quantifying the uncertainty of an output prediction. However, the memory cost of storing a large ensemble of neural networks quickly becomes prohibitive and limits their applicability. This paper details a three-stage in-memory computing circuit that performs analog-domain vector-matrix multiplication between an input voltage vector and a rank-1 compressed weight ensemble stored in the conductances of three Memristor arrays. For wide layers (thousands of neurons) and large ensemble sizes (hundreds to thousands of models), this circuit reduces the required number of Memristors by between two and three orders of magnitude relative to a non-compressed ensemble. Similarly, compared to a single neural network, the increase in the number of Memristors may be less than two-fold. We report SPICE simulations of the circuit and observe that 75% of total error does not deviate by more than 25% of the ideal value. A statistical analysis of the circuit explains these observations and offers insights regarding how the circuit may be improved.**

I. Introduction

Safety-critical domains such as medical diagnosis and autonomous navigation require accurate assessment of risk and uncertainty. Deterministic neural networks are unable to properly express uncertainty in their predictions. They often deliver incorrect and overconfident predictions upon encountering inherently confusing or out-of-distribution data [1]. Predictive uncertainty is also useful in many applications, for example, when learning using small and noisy datasets or active learning [2]. Uncertainty quantification in machine learning is often achieved using an ensemble of neural networks [3]. Such ensembles may be obtained by training several models [4], by using Bayesian inference via variational methods [5] or Markov Chain Monte Carlo (MCMC) [6], or even through a combination these methods [7]. The downside of ensemble methods however is twofold: their increased memory and energy requirements relative to a single model. Instead of storing one set of model parameters that are accessed once per inference, it is necessary to store several different model parameters, and access them all for each inference.

In order to reduce the energy cost of ensemble models, implementations of analog-domain Bayesian neural networks using crossbar arrays of Memristors have been proposed [8], [9]. These in-memory implementations promise a significant reduction in energy cost by avoiding the transfer of the model parameters between separate memory and computation units. The vector-matrix multiplication between data and the parameters is achieved in-memory directly between a vector of voltages and the conductance states of the Memristors in the crossbar. The memory cost of implementing ensemble models with Memristors however still remains prohibitive.

One idea that has reduced the memory requirements of digital implementations of neural network ensembles is BatchEnsemble [10], a technique that uses rank-1 approximation to compress the parameter matrix. Remarkably, even with a high compression rate, the performance of rank-1 compressed ensembles obtain an equivalent performance to a non-compressed ensembles [10]. Rank-1 matrix compression requires the combination of vector-vector outer product, matrix-matrix element-wise multiplication and vector-matrix dot product operations. Memristor crossbar arrays however only support the latter. In order to bring the benefits of BatchEnsemble to in-memory computing, this paper presents a new analog Memristive circuit, which we refer to as the Memristor Hadamard Multiplier. By combining this new block with a conventional Memristor crossbar in a three-stage pipeline, we realise the two missing operations.

This paper is structured as follows: Section II explains how the computations of rank-1 ensemble compression can be written in a more convenient form. The terms of the resulting expression are related to the physical currents, voltages and resistances that make the link between the algorithm and the hardware implementation. Section III introduces our novel in-memory computing circuit that performs compressed vector-matrix multiplication for a ensemble neural network layer. Section IV reports Monte Carlo SPICE simulations of the circuit in a CMOS technology to study the impact of process variation on the precision of the circuit. We explain the empirical results with a statistical analysis that offers important insights as to how the circuit may be improved.

II. Implementing Compressed Rank-1 Multiplication In-Memory

This section presents the concept of rank-1 matrix compression, and how it is leveraged in BatchEnsemble [10] to reduce the memory requirements of ensemble neural networks. We then establish a link between the terms of the algorithm and a hardware implementation with physical devices.

A. Concepts

In the following, bold lowercase letters are reserved for vectors. Uppercase bold letters represent matrices. Lowercase

979-8-3315-4127-9/24/$31.00 ©2024 IEEE

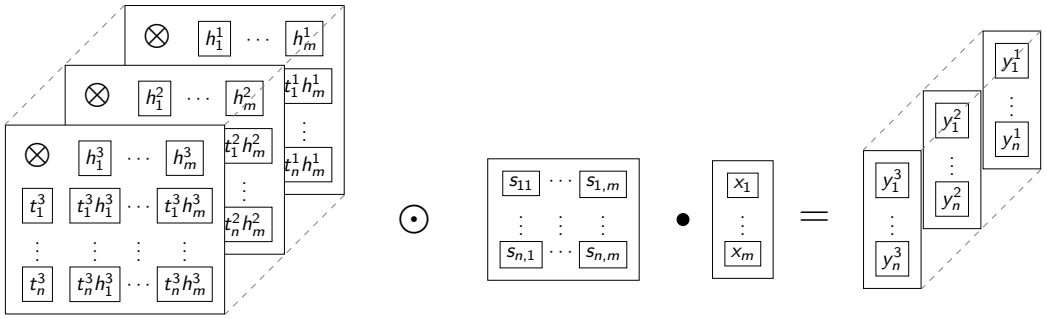


Fig. 1. Computation of BatchEnsemble [10]. s: component of shared matrix, h and t: component of horizontal and tall vectors of rank 1 compression. (th): components of rank-1 outer product matrix. x: component of input vector. y: component of output vector. The ensemble size is denoted by e. The illustration contains only 3 examples to avoid cluttering.

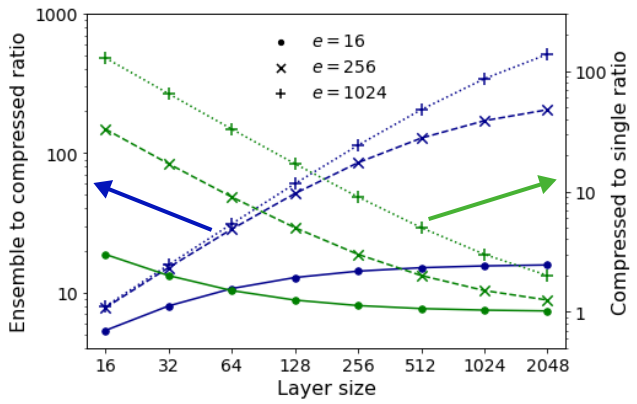


Fig. 2. Ratio of the Memristor count for a rank-1 compressed ensemble compared to full ensemble (blue, left y-axis) and ratio of number of Memristors of a compressed model to a single model (green, right y-axis) over a range of layer sizes. Blue and green arrows indicate to which axis the blue and green curves belong. Three sets of curves are plotted for ensembles sizes, e, of 16 (dot), 256 (cross) and 1024 (plus sign).

letters are used for scalar component of corresponding matrix or vector. The notation $'$ refers to the transpose of a vector or matrix, and vectors are in column format by default. The ensemble size (i.e. the number of neural network parameter matrices) is denoted by e. The core concept of rank-1 matrix compression is depicted in Figure 1. Instead of storing e full size parameter matrices, we only store a single full size matrix and e pairs of vectors. For each model in this ensemble, indexed by the superscript i, a rank-1 compressed parameter matrix $\mathbf{W}^{(i)} \in \mathbb{R}^{n\times m}$ is formed by the outer product between a tall vector $\mathbf{t}^{(i)} = [t_1^{(i)}, \ldots, t_n^{(i)}]' \in \mathbb{R}^n$ and a horizontal vector $\mathbf{h}^{(i)'} = [h_1^{(i)}, \ldots, h_m^{(i)}] \in \mathbb{R}^m$, followed by an element-wise multiplication with a matrix $\mathbf{S} \in \mathbb{R}^{n\times m}$, that is shared between all vector pairs. The corresponding output vector $\mathbf{y}^{(i)} \in \mathbb{R}^n$ for the input data vector $\mathbf{x} \in \mathbb{R}^m$ is then defined as

$$\forall i, \quad \mathbf{y}^{(i)} = \mathbf{W}^{(i)} \cdot \mathbf{x} = \left((\mathbf{t}^{(i)} \otimes \mathbf{h}^{(i)'}) \odot \mathbf{S}\right) \cdot \mathbf{x}, \qquad (1)$$

where $\otimes$ is the vector-vector outer product, $\odot$ is an element-wise multiplication and $\cdot$ is vector-matrix product. The distribution of $\mathbf{y}^{(i)}$ can be used to calculate the uncertainties related to a prediction made regarding input $\mathbf{x}$ [11].

B. Analog hardware implementation

First, rank-1 approximation (Equation 1) can be recast as (see full derivation in Appendix Section A):

$$\mathbf{y}^{(i)} = [(\mathbf{t}^{(i)} \otimes \mathbf{h}^{(i)'}) \odot \mathbf{S}] \cdot \mathbf{x} = [\mathbf{S} \cdot (\mathbf{x} \odot \mathbf{h}^{(i)})] \odot \mathbf{t}^{(i)}, \quad (2)$$

which allows the calculation to be broken down into three distinct steps. For the i^{th} weight matrix in the ensemble, the first step calculates the first intermediate vectors $\mathbf{a}^{(i)}$:

$$\mathbf{a}^{(i)} = \mathbf{x} \odot \mathbf{h}^{(i)}. \qquad (3)$$

The input data $\mathbf{x}$ is represented as a vector of currents that is applied to a vector of Memristors whose resistance values are programmed to match vector $\mathbf{h}^{(i)}$. Due to Ohm's law, the voltage drops across the Memristors will provide the intermediate vector $\mathbf{a}^{(i)}$. It is worth noting that this multiplication is achieved between currents and resistances, and not voltages and conductances as is typical for Memristor crossbar arrays.

The second step leverages the standard Memristor crossbar array architecture in order to calculate the second intermediate vector $\mathbf{b}^{(i)}$:

$$\mathbf{b}^{(i)} = \hat{\mathbf{S}} \cdot \mathbf{a}^{(i)} = \hat{\mathbf{S}} \cdot (\mathbf{x} \odot \mathbf{h}^{(i)}). \qquad (4)$$

By applying the voltages $\mathbf{a}^{(i)}$ from the previous step over the columns of a crossbar, where the Memristor resistances are programmed with the values $\hat{\mathbf{S}}$, the current output at the rows of the crossbar will be equal to $\mathbf{b}^{(i)}$. Note that the values in matrix $\hat{\mathbf{S}}$ are in fact the inverse resistances of each Memristor (i.e., the conductance) where each entry $\hat{s}_{i,j} = 1/s_{i,j}; \forall i, j$.

The final step to compute the output voltage vector $\mathbf{y}^{(i)}$ is in fact identical to the first:

$$\mathbf{y}^{(i)} = \mathbf{b}^{(i)} \odot \mathbf{t}^{(i)} = [\hat{\mathbf{S}} \cdot (\mathbf{x} \odot \mathbf{h}^{(i)})] \odot \mathbf{t}^{(i)}, \qquad (5)$$

The current vector $\mathbf{b}^{(i)}$ from the second step is injected into Memristors whose resistance values are programmed equal to vector $\mathbf{t}^{(i)}$, resulting in the desired output voltage drop $\mathbf{y}^{(i)}$.

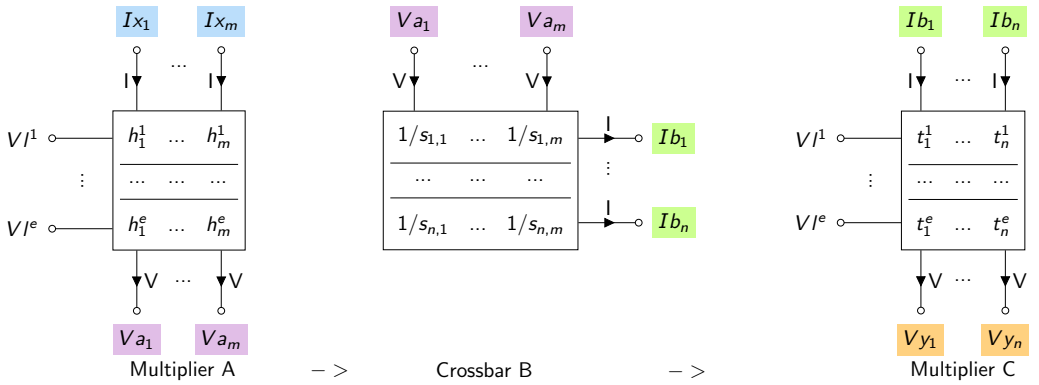


Fig. 3. Summary of analog memristor circuit for BatchEnsemble. Input and output for each block are color-coded to help with visualization. Multiplier A ingests input data as current Ix, and produces voltage Va as output. These voltages are chained to Crossbar B as input, which in turn produces current Ib as output. These currents are injected as input to Multiplier C, which produces the final result as voltages Vy. The voltage selector lines $(Vl^1, ..., Vl^e)$ are the same throughout the whole circuit.

Assuming one Memristor is used to encode each value of the e vector pairs $\mathbf{t}^{(i)}$ and $\mathbf{h}^{(i)}$ and the matrix $\hat{\mathbf{S}}$, Figure 2 examines the reduction in the number of Memristors used in the rank-1 compressed ensemble relative to an uncompressed ensemble. For ensembles composed of hundreds or thousands of members, the reduction in the number of Memristors is between two and three orders of magnitude relative to an uncompressed ensemble. Besides reducing the surface required to store a model, this method will also reduce the energy consumption by performing less read operations. On a second y-axis on the right, the increase in the number of Memristors relative to a single neural network is also shown. In particular for wide layers, similar in size to those typically used in Transformers, the increase in the total number of Memristors when increasing the ensemble size is minimal. For example, a layer of 2048 neurons and an ensemble size of $e = 1024$ only doubles the number of Memristors compared to a single neural network.

III. In-memory rank-1 multiplication circuit

This section presents the novel Memristor Hadamard Multiplier circuit, and shows how it may be interfaced with the input and output of a Memristor crossbar in order to realise the three steps described in the previous section. The three steps are summarised as three functional blocks (A, B and C) depicted in Figure 3, in which we refer to the new circuit block as *Multiplier* for brevity.

In the following circuit diagrams, a prefix is added to the algorithmic values to denote an electrical quantity. For example, Ix_1 denotes the first component of input vector $\mathbf{x}$ encoded as a current. Va_m is the voltage output by the m^{th} component of Multiplier A. The superscript * is used to denote an approximate analog copy of the original signal. For instance, Ix_1^* is the mirrored current of Ix_1, and Va_m^* is the voltage output by an OPAMP voltage follower with input Va_m. The input and output to each block are colored with the same code to help with visualization in Figure 3, Figure 4, Figure 5 and in Figure 6.

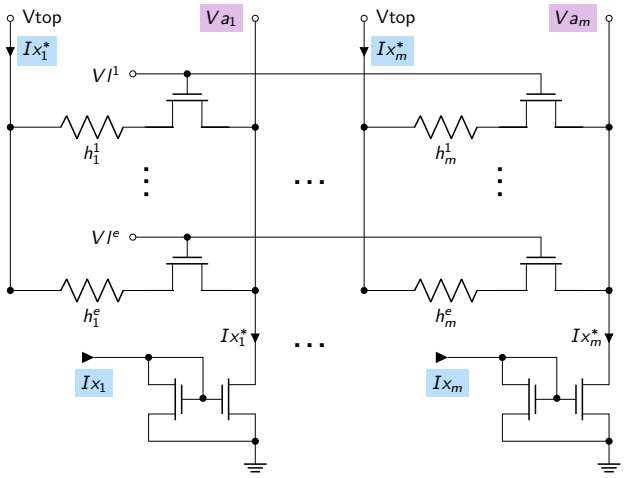


Fig. 4. Multiplier A with input current Ix colored in blue. Output voltage Va is colored in violet. Va run through all transistor sources in a column. Activating a selector voltage line Vl among will direct the input current to the corresponding row. Vl runs through all transistor gates in a row. There are e rows and m columns in this array.

Figure 4 depicts Multiplier A. Each row of resistor values within this circuit represent a vector $\mathbf{h}^{(i)}$. There are e rows, corresponding to the ensemble size. The size of each vector in the row is the layer input dimension, m. Selector transistors in the i^{th} row are opened by setting high the voltage on the selector line $Vl^{(i)}$.

When a row is selected for reading, m input currents are injected by m NMOS current mirrors located at the bottom of the m columns of the Memristor array. The mirrored currents, Ix_j^*, flow through each of the programmed resistances $h_j^{(i)}$ incurring a voltage drop over each of the Memristors. If

we consider the contribution of the selector transistors to be negligible, the voltages measured at the source of the transistors, $Va_j = Vtop - Ix_j^* h_j^{(i)}$.

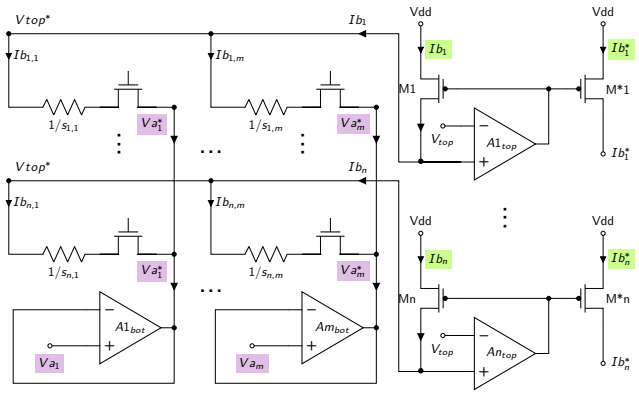


Fig. 5. Crossbar B with voltage followers A_{bot} and buffers A_{top}. The buffers maintain the voltage at Vtop, while mirroring the summing current flowing through each row of the crossbar. The voltage followers regulate the voltage at Va, and act as current sinks. Output voltages from previous stage Multiplier A, colored in violet, serve as input to the voltage followers. The resulting mirrored currents Ib are colored in green. The transistors have not been labelled to simplify the schematic, they serve as selector devices when programming the array.

These voltages are then buffered at the base of each of the m columns in the Memristor crossbar B, depicted in Figure 5 via m OPAMPs, denoted by A_{bot}. The same reference voltage as in the first stage, $Vtop$, is imposed at the end of the n rows of the crossbar using n OPAMPs, denoted A_{top}. The resulting voltage differences over each device in the crossbar will cause a current Ib to flow in each of the crossbar rows equal to $\hat{\mathbf{S}} \cdot (Vtop^* - Va^*)$. These currents are then copied using n current mirrors, resulting in the output Ib^*.

These currents are then injected into Multiplier C, the third and final stage shown in Figure 6. This block is very similar to the first circuit in Figure 4, except that the currents are sourced from the PMOS transistors (those at the output of the second stage), and the output voltages are read out from the top terminal of the Memristor instead of the source of the transistor. The selector signals Vl are the same as those applied to the first block. The resulting vector of n voltages, Vy, will be equal to $Ib^* t^{(i)}$, thereby computing the output for the i^{th} parameter matrix of the ensemble. In order to compute the full distribution of e outputs, it is simply required to hold the input currents at Ix as the $Vl^{(i)}$ signals are sequentially activated from $i = 1$ to $i = e$.

Note that a crossbar implementing a single model also requires voltage regulation on the columns and rows. Therefore, the total number of OPAMPs required in our circuit is the same as for a single model. In contrast, relative to non-compressed Memristor-based ensemble circuits, the number of OPAMPs will be reduced by a factor equal to the ensemble size, e.

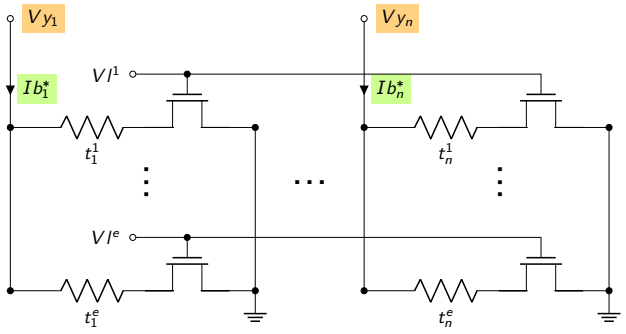


Fig. 6. Multiplier C with input currents Ib colored in green and final output voltages Vy colored in orange. The selector lines Vl are the same signals as in Figure 4. Activating one selector voltage will direct the current to the corresponding line. We can retrieve the final result vector by reading voltages Vy.

IV. Circuit Simulation and Analysis

To assess the feasibility of the presented circuit, we performed Monte Carlo simulations using a SPICE model. Over multiple simulation runs, the parameters of the circuit transistors are sampled from the process variation distributions of a 130nm CMOS technology. We use two transistor types from this technology, one with a nominal voltage of 1.2V and a second transistor with a thicker gate oxide that has a nominal voltage of 1.8V. As is often the case in Memristor-based in-memory computing circuits, we use the 1.8V rated transistor as the selector transistor for the Memristors.

A. Experimental Setup

We perform thirty-two Monte Carlo runs for an instance of the circuit where $m = n = e = 16$, i.e. all three blocks are composed of 16×16 matrices. The input currents and device resistances are randomly sampled and then kept constant over all Monte Carlo runs. The input currents were sampled from a Uniform distribution between 50nA and 150 nA. The resistances in each of the three stages were sampled from a Uniform distribution between 500kΩ and 1.5MΩ. These current and resistance ranges were designed to keep the voltage drop over each Memristor around a few hundred mV, beyond which point many Memristors may undergo unintended programming. This resistance range could, for example, be achieved by emerging spintronic Memristors [12], although the circuit is agnostic to a specific Memristor technology. In order to keep the resistance ranges homogeneous across the three layers, it was required to scale the currents output by Crossbar B down by a factor of ten. This is achieved simply by setting the transistor width ratio to ten for the current mirrors at the output of Figure 5.

We use an ideal OPAMP circuit model with a gain of 100 and cascoded current mirrors in order to minimise the current copying error. The pulse width of the Vl selector lines was set to 3μs with a rise and fall time of 1μs, resulting in a total period of 5μs to compute the output of one ensemble member.

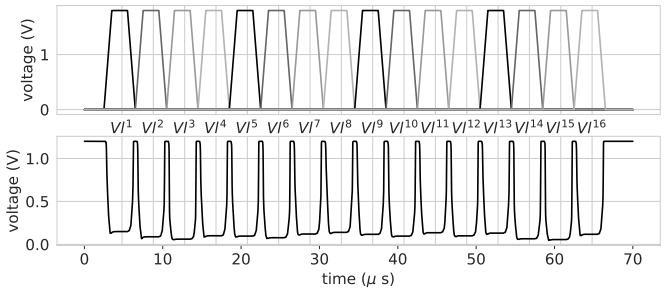


Fig. 7. After activating a voltage selector line, we can read out the output voltage signal corresponding to the selected model sample. Note the voltage of the selector line is 1.8V because because this is the nominal voltage of the selector transistors. Top: 16 selector voltage lines $Vl^{(i)}$ as consecutive single pulse signal. Bottom: first component of output vector $Vy_1^{(i)}$.

B. Circuit Validation

First, in order to demonstrate the basic functionality of the circuit, we plot the voltages output at the node $Vy_1^{(i)}$ during a single Monte Carlo run in Figure 7. We iterate over the sixteen rank-1 compressed parameter matrices of the ensemble, from $i = 1$ to $i = 16$. This is achieved by setting the different Vl selector lines high in consecutive pulses (the top panel of Figure 7) while maintaining the same input current at Multiplier A. Due to the different resistances values of each ensemble member, it can be observed that the voltage output by the circuit varies on each period. For a neural network ensemble, these different output voltages might represent a predictive distribution from which an uncertainty estimate could be obtained.

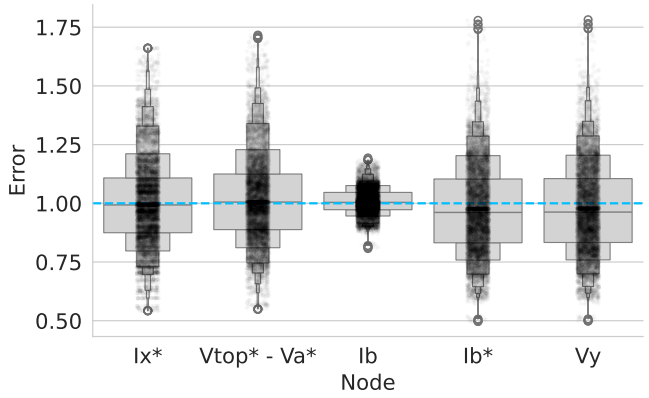


Fig. 8. Error as ratio of (observed / target) value for essential nodes. The middle line of each box plot is the median. Each box in either direction halves the tail area. For example, 1st box above median is 0.5 + 0.25 = 0.75 quantile, 1st box below median is 0.25 quantile. 2nd box above median is 0.75 + 0.125 = 0.875 quantile, 2nd box below median is 0.125 quantile, etc. (see [13])

Figure 8 plots the error ratios (i.e. the observed voltage or current values divided by the values expected in an ideal circuit) for five important circuit nodes over all Monte Carlo runs using box-plots. Each box plot is a double marginalization: over all thirty-two Monte Carlo simulation runs, and over the sixteen columns of the two multipliers or the sixteen rows of the crossbar.

For the first node Ix^* (i.e. the mirrored input current), three quarters of the current mirror error is within the $\pm 25\%$ error range. In extreme cases this can extend up to 70%, indicating that, even with the cascoding, the input current mirror introduces a large error early in the calculation. The second node reported in the plot is the voltage difference $(Vtop^* - Va^*)$ applied in the crossbar array. Reassuringly, the variance of the error distribution barely increases from the first node Ix^*. This observation stems from the fact that the amplifiers that impose $Vtop$ and Va both have a consistent negative absolute error with low variance (between four and five millivolts) (see Figure 13 and Figure 14 in Appendix Section C). Since Va^* is subtracted from $Vtop$ the combined error is generally around one millivolt and, as the voltage drop is typically on the order of a hundred millivolts, the additional error is negligible. Intriguingly, the error variance reduces dramatically between the node $(Vtop^* - Va^*)$ and the node Ib, of Crossbar B. We will investigate this surprising result in detail in Section IV-C.

Consistent with the input current mirror in the first stage, the current copy of Ib to Ib^* (Figure 5) once again increases the spread of errors in the fourth box-plot. Note that a small negative bias is introduced at this stage due to the imperfect scaling of the current mirror at the output of Crossbar B. The final box-plot in Figure 8 shows the distribution of error at the output of the circuit, Vy. Since this stage involves only a multiplication by the resistances encoding the vector $\mathbf{t}$, it remains largely the identical to the error of Ib^*.

C. Analysis of error variance reduction

This section applies statistics to analyse the intriguing reduction in error variance observed in Figure 8 between the input $(Vtop^* - Va^*)$ and output nodes of the second stage, Ib. The starting point of our analysis is to rewrite Equation 4 in scalar form, where each term is considered to be a random variable. As a reminder, m is the dimension of the horizontal vector $\mathbf{h}^{(i)}$ of the parameter matrix at index i. The superscript i is hereafter omitted to simplify notation. The k^{th} column voltage that is output by Multiplier A after the analog copy is considered to be a random variable with a known expectation $E[a_k]$ and variance $V[a_k]$. The errors in each column are assumed to be independent of each other (verified empirically from Figure 9 in Appendix Section C). Although the resistance values were fixed over all Monte Carlo simulations, in reality Memristor programming is subject to significant variability in the resistance value [9]. Thus we find it informative to also consider $s_{j,k}$ to be a random variable that is independent with respect to a_k.

For row j of the Crossbar b_j, the expectation $E[b_j]$ of the current sum may therefore be expressed as:

$$E[b_j] = E[\sum_{k=1}^{m} s_{j,k}^{-1} a_k] = \sum_{k=1}^{m} E[s_{j,k}^{-1} a_k] = \sum_{k=1}^{m} E[s_{j,k}^{-1}] E[a_k] \tag{6}$$

The first equality in Equation 6 is simply a substitution of the definition for the term b_j. The second equality stems from linearity of expectation, where the expectation of a sum is cast as a sum of expectations. For the last equality, since the two random variables are independent, the expectation of a product is equal to the product of expectations.

Because we have observed that $E[a_k]$ is centered on the ideal value (verified empirically in Figure 9 in Appendix Section C) and $E[s_{j,k}^{-1}]$ is also centered on its true value [8], we deduce that b_j should in turn be centered on its true value. This corresponds to what is observed in Figure 8 in the third box-plot, where output current does not appear to be biased.

We continue by analysing the quantity in which we are most interested: the variance $V[b_j]$. This variance can be expressed as:

$$\begin{aligned} V[b_j] &= V[\sum_{k=1}^{m} s_{j,k}^{-1} a_k] = \sum_{k=1}^{m} V[s_{j,k}^{-1} a_k] \\ &= \sum_{k=1}^{m} V[s_{j,k}^{-1}] V[a_k] + E[s_{j,k}^{-1}]^2 V[a_k] + E[a_k]^2 V[s_{j,k}^{-1}] \end{aligned} \tag{7}$$

The first equality is simply a substitution for the definition of b_j. The second equality comes from independence of two random variables, which allows the variance of a sum to be written as the sum of variances. The last equality is an application of Appendix Section B, which describes how variances of a product of random variables may be expanded.

We observe that, for each row of Crossbar B, the variance of b_j is a sum of three products composed of four distinct terms, each with their own physical implication. The variance $V[b_j]$ can be reduced by reducing each constituent. First, we may reduce $V[s_{j,k}^{-1}]$ by having a more precise memristor programming, for example with a higher precision of digital to analog conversion. Similarly, reducing $V[a_k]$ requires designing voltage and current copy circuits with less error. Third, using a higher Memristor resistance magnitude will shrink the term $E[s_{j,k}^{-1}]^2$. Finally, a smaller voltage drop over the devices can also reduce the total variance by decreasing $E[a_k]^2$.

In our Monte Carlo simulations we implicitly consider the Memristor resistances that are sampled at the beginning of the simulation to be exact, such that $s_{j,k}^{-1}$ is deterministic. In this case, $V[s_{j,k}^{-1}] = 0$ and $E[s_{j,k}^{-2}] = s_{j,k}^{-2}$, such that only one term in Equation 7 remains:

$$V[b_j] = \sum_{k=1}^{m} s_{j,k}^{-2} V[a_k]. \tag{8}$$

It is clear that the variance at the output of the second stage is simply the variance incurred in the first stage divided by the square of the Memristor resistance values. In our simulations, $s_{j,k}$ are on the order of a Mega Ohm (0.5 - 1.5 MΩ). This highlights the important of using technologies that can support large resistance ranges in the second stage. Hence, a division by $s_{j,k}^2$ significantly shrinks the variance $V[a]$, as indeed observed in Figure 8. Note that, as the array size m grows, the error variance increases because of the sum of m non-negative terms in Equation 7.

D. Analysis of the output error

We now apply the same analysis to the output of the third stage, Vy, in order to understand the main contributing factors of the final error in the whole circuit. The input to the final block is a current corresponding to b_j^*. It was observed empirically that b_j^* is mirrored with an offset β_j. This offset is also a random variable due to process variation, and is assumed to be independent from other offsets and from the incoming currents b_j. Therefore, the expectation and variance of this copied current can be computed as:

$$b_j^* = b_j + \beta_j, \tag{9}$$

$$E[b_j^*] = E[b_j] + E[\beta_j], \tag{10}$$

$$V[b_j^*] = V[b_j] + V[\beta_j]. \tag{11}$$

This result can then be used to determine the expressions for the expectation and variance of y_j:

$$y_j = b_j^* t_j \tag{12}$$

$$E[y_j] = E[b_j^* t_j] = E[b_j^*] E[t_j] \tag{13}$$

$$V[y_j] = V[b_j^* t_j] \tag{14}$$

$$= V[b_j^*] V[t_j] + E[b_j^*]^2 V[t_j] + V[b_j^*] E[t_j]^2 \tag{15}$$

The negative offset in the 2 final box-plots of Figure 8 comes from the fact the expectation $E[b_j^*]$ is biased by the expectation $E[\beta_j]$.

The result for the variance is very similar to that of the previous section: a sum of three products and four unique constituents. In order to reduce the spread of errors therefore, it is required to minimise $V[t_j]$ with a more precise Memristor programming procedure. Conversely to Crossbar B, it is preferable to decrease the range of resistance values in Multiplier C in order to reduce $E[t_j]$. Limiting the mirrored current flowing out of Crossbar B also increases the precision by reducing $E[b_j^*]$. Finally, the implication of the term $V[b_j^*]$ is twofold. The variance in the estimation of b_j in the first two stages of the pipeline should be reduced. Strategies for this were outlined in the previous section. A second factor is the error introduced by the current mirror. From the empirical observations in Figure 8, the latter is most certainly responsible for the large increase in the error variance between the third stage Ib and fourth stage Ib^* of the pipeline.

V. Conclusion

We have presented a novel three-stage in-memory computing circuit capable of massively reducing the memory requirements of Memristor-based ensemble models (i.e. Memristor-based Bayesian neural networks) by using rank-1 compression. This improves the feasibility of executing ensemble neural networks on Memristive hardware for the purposes of quantifying the uncertainty related to predictions made by the model. It was observed how the number of Memristors for ensembles comprising thousands of models, and with big layer sizes common in models like Transformers, was only doubled relative to a single neural network using our method.

Monte Carlo SPICE simulations of the circuit were presented in order to study the impact of process variation on the analog-domain computation. A statistical analysis was performed in order to explain empirical observations from these simulations, in particular to understand an intriguing reduction in the error variance at the output of the second stage of the circuit. This analysis highlighted ways in which the precision of the circuit could be improved. For example, it was understood that the total error variance could be reduced by combining lower resistance values in the first and third stages, with higher resistance values in the second stage. Furthermore, the current mirrors were understood to be a particularly important source of error variance, as well introducing a slight bias into the calculation. The circuit requires two sets of current mirrors (at the input to the first stage, and at the output of the second stage) and therefore improving their design will be crucial to increase the precision of the circuit. An important next step will be to tape-out a test circuit that will allow realistic quantification of power-consumption, area and latency. One source of error that was not considered in the Monte Carlo simulations was the inherent variability of the programmed Memristor resistances. The statistical analysis highlighted that this variability will reduce the circuit precision. An appealing recent solution for this is to exploit this variability by applying Memristor-based in-memory computing circuits within the framework of Bayesian inference instead [14], [9], [15], [16]. From this perspective, Memristor programming error becomes an integral part of the calculation instead of an error source. Furthermore, an ambitious goal would be to implement a compressed Bayesian on-chip learning algorithm [8], [17] on our circuit. By doing so, not only does processes variation error become an in-built characteristic of the model, but new application domains based on adaptation in uncertain and evolving environments may be opened up.

Website

Additional content may be published on this web page: https://phanav.github.io/posts/hardware/memristor_hadamard_multiplier.html

Acknowledgment

The authors would like to thank the following people for their help: Laurent Alacoque, Sébastien Ricavy, Matthieu Faye, Tifenn Hirtzlin, Tarcisius Januel, Djohan Bonnet, Eiji Kawasaki, Guillaume Régis and Diego Puschini.

APPENDIX A
RANK-1 CALCULATION

This appendix presents the detail of Rank 1 calculation using the following symbols:

$\otimes$ is the outer product and $\odot$ is the element-wise multiplication, $\cdot$ is matrix-vector product. $'$ refers to the transpose of a vector or matrix. Vectors are in column format by default. i: index of the selected model sample in the ensemble. $\mathbf{y}^{(i)}$: prediction vector. $\mathbf{t}^{(i)}$: tall vector. $\mathbf{h}^{(i)'}$: horizontal vector. $\mathbf{S}$: shared matrix. $\mathbf{x}$: input vector.

$$
\begin{aligned}
\mathbf{y}^{(i)} &= \mathbf{W}^{(i)} \cdot \mathbf{x} = [(\mathbf{t}^{(i)} \otimes \mathbf{h}^{(i)'}) \odot \mathbf{S}] \cdot \mathbf{x} \\
&= \left[\left(\begin{vmatrix} t_1^{(i)} \\ \dots \\ t_d^{(i)} \end{vmatrix} \otimes \begin{vmatrix} h_1^{(i)} & \dots & h_d^{(i)} \end{vmatrix} \right) \odot \begin{vmatrix} s_{1,1} & \dots & s_{1,d} \\ \dots & \dots & \dots \\ s_{d,1} & \dots & s_{d,d} \end{vmatrix} \right] \cdot \begin{vmatrix} x_1 \\ \dots \\ x_d \end{vmatrix} \\
&= \begin{vmatrix} t_1^{(i)} h_1^{(i)} s_{1,1} x_1 + \dots + t_1^{(i)} h_d^{(i)} s_{1,d} x_d \\ \dots \\ t_d^{(i)} h_1^{(i)} s_{d,1} x_1 + \dots + t_d^{(i)} h_d^{(i)} s_{d,d} x_d \end{vmatrix} \\
&= \begin{vmatrix} s_{1,1} x_1 h_1^{(i)} + \dots + s_{1,d} x_d h_d^{(i)} \\ \dots \\ s_{d,1} x_1 h_1^{(i)} + \dots + s_{d,d} x_d h_d^{(i)} \end{vmatrix} \odot \begin{vmatrix} t_1^{(i)} \\ \dots \\ t_d^{(i)} \end{vmatrix} \\
&= \left[\begin{vmatrix} s_{1,1} & \dots & s_{1,d} \\ \dots & \dots & \dots \\ s_{d,1} & \dots & s_{d,d} \end{vmatrix} \cdot \left(\begin{vmatrix} x_1 \\ \dots \\ x_d \end{vmatrix} \odot \begin{vmatrix} h_1^{(i)} \\ \dots \\ h_d^{(i)} \end{vmatrix} \right) \right] \odot \begin{vmatrix} t_1^{(i)} \\ \dots \\ t_d^{(i)} \end{vmatrix} \\
&= [\mathbf{S} \cdot (\mathbf{x} \odot \mathbf{h}^{(i)})] \odot \mathbf{t}^{(i)}
\end{aligned}
$$

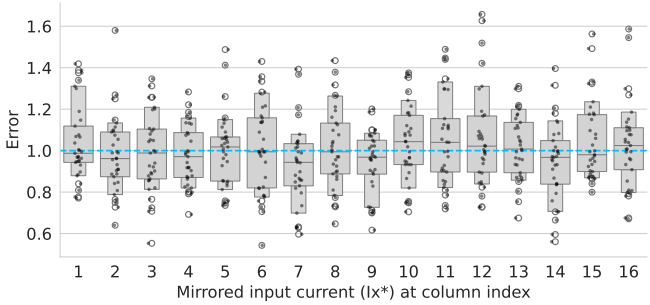
APPENDIX B
VARIANCE OF PRODUCT OF RANDOM VARIABLES

With independent random variables X and Y:

$$
\begin{aligned}
V[XY] &= E[X^2 Y^2] - E[XY]^2 \\
&= E[X^2] E[Y^2] - E[X]^2 E[Y]^2 \\
&= (V[X] + E[X]^2)(V[Y] + E[Y]^2) - E[X]^2 E[Y]^2 \\
&= V[X]V[Y] + E[X]^2 V[Y] + E[Y]^2 V[X]
\end{aligned}
$$

APPENDIX C
ADDITIONAL EXPERIMENT RESULT

Below are the simulation result of some essential nodes in the circuit for the first selected model parameter sample (Only Vl^1 is high, $Vl^2, ..., Vl^{16}$ are all low).



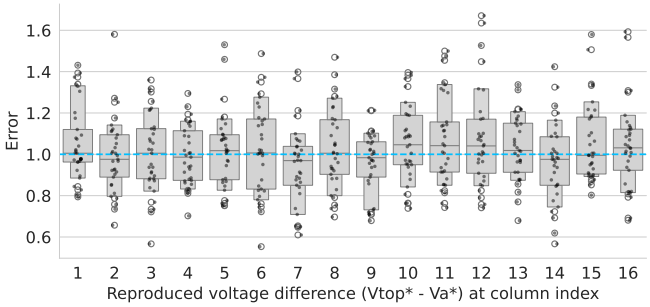


Fig. 9. Error as ratio of (observed / target) value of reproduced voltage difference between $Vtop^*$ and Va^* for the first selected model sample. The 16 columns of multiplier a are on the horizontal axis. For each column, the box plot represents the variance between 32 Monte Carlo runs.

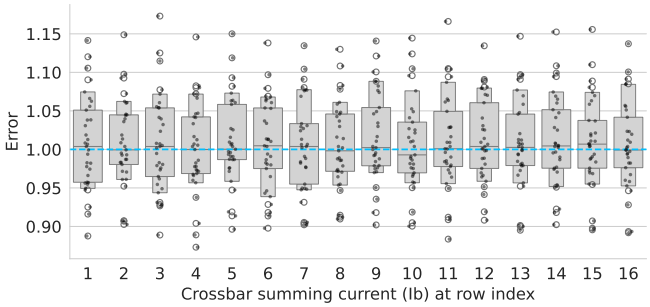


Fig. 10. Error as ratio of (observed / target) value of summing current from each row of crossbar b for the first selected model sample. The 16 rows are on the horizontal axis. For each row, the box plot represents the variance between 32 Monte Carlo runs.

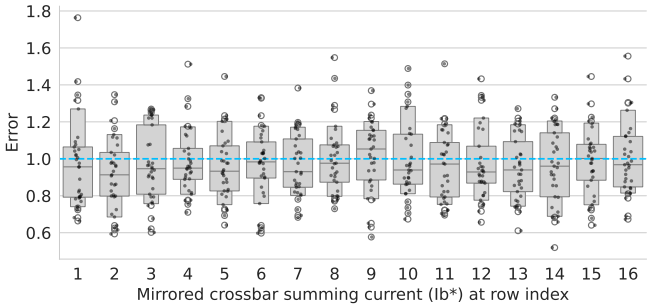


Fig. 11. Error as ratio of (observed / target) value of mirrored summing current from each row of crossbar b for the first selected model sample. The 16 rows are on the horizontal axis. For each row, the box plot represents the variance between 32 Monte Carlo runs.

REFERENCES

[1] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2019, no. 1254, pp. 14 003–14 014.

[2] Y. Gal, R. Islam, and Z. Ghahramani, "Deep Bayesian Active Learning with Image Data," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1183–1192. [Online]. Available: https://proceedings.mlr.press/v70/gal17a.html

[3] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, M. Shahzad, W. Yang,

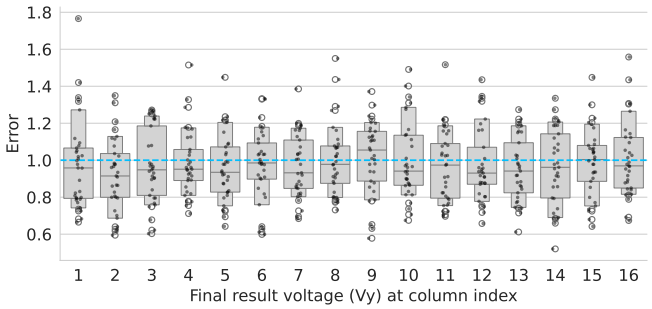


Fig. 12. Error as ratio of (observed / target) value of final result voltage for the first selected model sample. The 16 columns of multiplier c are on the horizontal axis. For each row, the box plot represents the variance between 32 Monte Carlo runs.

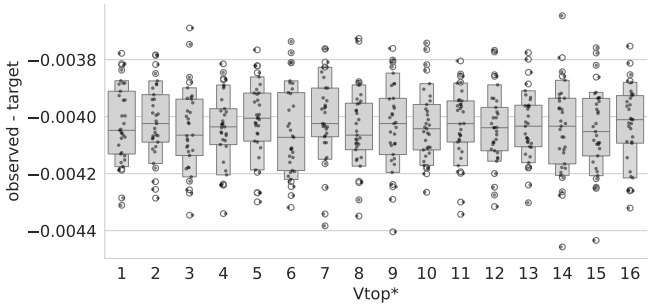


Fig. 13. Absolute error of Vtop voltage follower: (observed - target) = (Vtop* - Vtop), for the first selected model sample. The 16 rows of crossbar b are on the horizontal axis. For each row, the box plot represents the variance between 32 Monte Carlo runs.

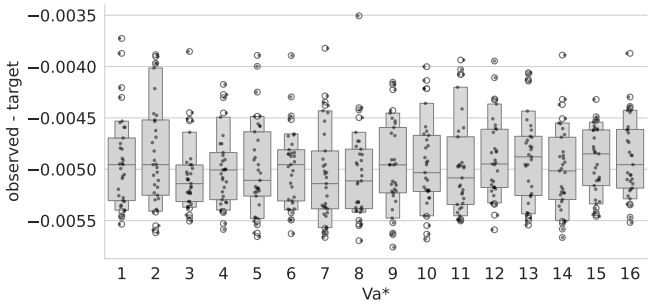


Fig. 14. Absolute error of Va voltage follower buffer: (observed - target) = (Va* - Va), for the first selected model sample. The 16 columns of multiplier a are on the horizontal axis. For each row, the box plot represents the variance between 32 Monte Carlo runs.

R. Bamler, and X. X. Zhu, "A survey of uncertainty in deep neural networks," *Artif. Intell. Rev.*, vol. 56, pp. 1513–1589, 2023. [Online]. Available: https://doi.org/10.1007/s10462-023-10562-9

[4] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/hash/9ef2ed4b7fd2c810847ffa5fa85bce38-Abstract.html

[5] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, pp. 1613–1622.

[6] P. Izmailov, S. Vikram, M. D. Hoffman, and A. G. G. Wilson, "What Are Bayesian Neural Network Posteriors Really Like?" in *International Conference on Machine Learning*. PMLR, 2021, pp. 4629–4640. [Online]. Available: https://proceedings.mlr.press/v139/izmailov21a.html

[7] M. Dusenberry, G. Jerfel, Y. Wen, Y. Ma, J. Snoek, K. Heller, B. Lakshminarayanan, and D. Tran, "Efficient and Scalable Bayesian Neural Nets with Rank-1 Factors," in *International Conference on Machine Learning*. PMLR, 2020, pp. 2782–2792. [Online]. Available: https://proceedings.mlr.press/v119/dusenberry20a.html

[8] T. Dalgaty, N. Castellani, C. Turck, K.-E. Harabi, D. Querlioz, and E. Vianello, "In situ learning using intrinsic memristor variability via Markov chain Monte Carlo sampling," *Nature Electronics*, vol. 4, no. 2, pp. 151–161, 2021. [Online]. Available: https://www.nature.com/articles/s41928-020-00523-3

[9] D. Bonnet, T. Hirtzlin, A. Majumdar, T. Dalgaty, E. Esmanhotto, V. Meli, N. Castellani, S. Martin, J.-F. Nodin, G. Bourgeois, J.-M. Portal, D. Querlioz, and E. Vianello, "Bringing uncertainty quantification to the extreme-edge with memristor-based Bayesian neural networks," *Nature Communications*, vol. 14, no. 1, p. 7530, 2023. [Online]. Available: https://www.nature.com/articles/s41467-023-43317-9

[10] Y. Wen, D. Tran, and J. Ba, "BatchEnsemble: An Alternative Approach to Efficient Ensemble and Lifelong Learning," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=Sklf1yrYDr

[11] A. G. Wilson and P. Izmailov, "Bayesian Deep Learning and a Probabilistic Perspective of Generalization," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 4697–4708. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/322f62469c5e3c7dc3e58f5a4d1ea399-Abstract.html

[12] T. Shibata, T. Shinohara, T. Ashida, M. Ohta, K. Ito, S. Yamada, Y. Terasaki, and T. Sasaki, "Linear and symmetric conductance response of magnetic domain wall type spin-memristor for analog neuromorphic computing," *Applied Physics Express*, vol. 13, no. 4, p. 043004, 2020. [Online]. Available: https://dx.doi.org/10.35848/1882-0786/ab7e07

[13] H. Hofmann, H. Wickham, and K. Kafadar, "Letter-Value Plots: Boxplots for Large Data," *Journal of Computational and Graphical Statistics*, vol. 26, no. 3, pp. 469–477, 2017. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/10618600.2017.1305277

[14] T. Dalgaty, E. Esmanhotto, N. Castellani, D. Querlioz, and E. Vianello, "Ex Situ Transfer of Bayesian Neural Networks to Resistive Memory-Based Inference Hardware," *Advanced Intelligent Systems*, vol. 3, no. 8, p. 2000103, 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/aisy.202000103

[15] Y. Lin, Q. Zhang, B. Gao, J. Tang, P. Yao, C. Li, S. Huang, Z. Liu, Y. Zhou, Y. Liu, W. Zhang, J. Zhu, H. Qian, and H. Wu, "Uncertainty quantification via a memristor Bayesian deep neural network for risk-sensitive reinforcement learning," *Nature Machine Intelligence*, vol. 5, no. 7, pp. 714–723, 2023. [Online]. Available: https://www.nature.com/articles/s42256-023-00680-y

[16] A. Sebastian, R. Pendurthi, A. Kozhakhmetov, N. Trainor, J. A. Robinson, J. M. Redwing, and S. Das, "Two-dimensional materials-based probabilistic synapses and reconfigurable neurons for measuring inference uncertainty using Bayesian neural networks," *Nature Communications*, vol. 13, no. 1, p. 6139, 2022. [Online]. Available: https://www.nature.com/articles/s41467-022-33699-7

[17] T. Dalgaty, S. Yamada, A. Molnos, E. Kawasaki, T. Mesquida, F. Rummens, T. Shibata, Y. Urakawa, Y. Terasaki, T. Sasaki, and M. Duranton, "Scaling-up Memristor Monte Carlo with magnetic domain-wall physics," in *MLNCP2023 - 37th NeurIPS Machine Learning with New Compute Paradigms Workshop*, 2023. [Online]. Available: https://hal.science/hal-04590374