Loading-Aware Reliability Improvement of Ultra-Low Power Memristive Neural Networks

Shaghayegh Vahdat, Mehdi Kamal, *Senior Member, IEEE*, Ali Afzali-Kusha, *Senior Member, IEEE*, and Massoud Pedram, *Fellow, IEEE*

***Abstract*— In this paper, a method for offline training of inverter-based memristive neural networks (*IM*-NNs), called ERIM, is presented. In this method, the output voltage of the inverter is modeled very accurately by considering the loading effect of the memristive crossbar. To properly choose the size of each inverter, its output load and the required slope of its voltage transfer characteristic (VTC) for an acceptable level of resiliency to the circuit element non-idealities are taken into account. The efficacy of ERIM is investigated by comparing its accuracy to those of two recently proposed offline training methods for *IM*-NNs (RIM and PHAX). The study is performed using IRIS, BCW, MNIST, and Fashion MNIST datasets. Simulation results show that 72% (56%) reduction in average energy consumption of the trained networks is achieved compared to RIM (PHAX) thanks to proper sizing of the inverters. In addition, due to the higher accuracy of the NN mathematical model, ERIM results in significant improvements in the match between the results of high-level modeling and HSPICE simulations while exhibiting lower sensitivity to circuit element variations.**

***Index Terms*— Inverter-based memristive neural networks, offline training, load of memristive crossbar, inverter sizing, sensitivity to variations.**

I. Introduction

ENERGY consumption has become a key challenge in designing NNs especially for data centers as well as portable or battery-powered devices [1]. Examples of approaches presented in the literature to overcome this challenge in digital platforms include computation in memory (CIM) [2], employing approximate arithmetic units [3], and pruning redundant weights [4]. Compared to digital implementations, analog NN implementations result in significant reductions in the computation energy consumption as well as improvements in the speed. For instance, the fabricated memristive-based NN of [5] led to 30× and 110× better speed and energy consumption, respectively, compared to the GPU platform for the MNIST dataset. In addition, memristive structures can be fabricated with a high layout density [6] while enabling parallel execution of the multiplication operations of each layer.

In the hardware implementation of memristive NNs, operational amplifiers (Op-Amp) [7], [8] or inverters [9]–[15] may be employed. Among memristive neuron structures, an inverter based structure can reduce the power consumption of a neuron up to 800× compared to a similar Op-Amp based neuron [9]. This leads to an ultra-low power implementation of NNs which makes *IM*-NNs good candidates for implementation of internet of things (IoT) smart sensors [11].

One of the main limitations of memristive NNs is their accuracy degradation due to circuit element non-idealities. As an example, the accuracy of an Op-Amp based memristive NN under MNIST benchmark may degrade by more than 30% when considering conductance variation of memristors [16]. Several approaches ([16]–[21]) have been proposed in the literature to overcome the memristor non-ideality concern. The situation becomes worse in the case of *IM*-NNs where the complexity of characteristics modeling of inverters and the non-ideality of the constituent circuit elements can significantly degrade the output accuracy of the NN during both training and inference phases. In other words, inaccuracies of inverters VTC models (i.e., the difference between the employed model and actual characteristic of the fabricated device) utilized in the offline training phase may result in large errors in estimating the network outputs. To realize the advantages of *IM*-NNs, first, accurate modeling of the behavior of the network under the ideal conditions (without any variations) is required. Next, a variation mitigation approach must be developed to suppress the effect of said non-idealities. Among the variation mitigation approaches proposed in the literature for *IM*-NNs ([12], [14], [15]), the RIM method [15], in which the VTC slope of the inverters was reduced by adding low resistance grounded resistors to the output node of each inverter, is highly effective. This approach, however, suffers from the power dissipation overhead associated with the added resistors.

In this paper, we introduce an enhanced version of RIM, called ERIM, in which the behavior of the inverters of the NN under the ideal conditions are modeled with high accuracy. In addition, the inverters are properly sized based on their approximate equivalent output resistance and their required VTC slopes. This leads to considerable improvements in the power consumption of the NN given similar NN output accuracy. It is worth noting that in RIM and PHAX approaches,

Manuscript received March 7, 2021; revised May 5, 2021; accepted May 24, 2021. Date of publication June 14, 2021; date of current version July 13, 2021. This article was recommended by Associate Editor J. Zhao. *(Corresponding author: Ali Afzali-Kusha.)*

Shaghayegh Vahdat and Ali Afzali-Kusha are with the School of Electrical and Computer Engineering, University of Tehran, Tehran 14399-57131, Iran (e-mail: vahdat_s@ut.ac.ir; afzali@ut.ac.ir).

Mehdi Kamal is with the School of Electrical and Computer Engineering, University of Tehran, Tehran 14399-57131, Iran, and also with the School of Computer Science, Institute for Research in Fundamental Sciences, Tehran 19538-33511, Iran (e-mail: mehdikamal@ut.ac.ir).

Massoud Pedram is with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90007 USA (e-mail: pedram@usc.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2021.3084867.

Digital Object Identifier 10.1109/TCSI.2021.3084867

sizes of all inverters were chosen to be five times that of a minimum sized inverter, which can result in an unnecessary increase in the power consumption. Furthermore, in these approaches, fixed VTC coefficients were considered for modeling the behavior of all inverters of the network, which can cause modeling errors. Based on the above explanations, the contributions of this work are enlisted as follows:

- ➢ A highly accurate analytical model for the NN inverters is proposed which leads to matching accuracy in the range of 95-100% between the results of the proposed mathematical model of the NN and the HSPICE simulation results.
- ➢ The model suppresses adverse impacts of circuit element non-idealities on the accuracy of the trained network. On average, 0.9% and 1.7% improvements in the accuracy of the NN compared to those of the PHAX and RIM methods in the presence of transistors non-idealities are achieved.
- ➢ An approach for choosing proper sizes for the NN inverters is suggested which reduces the energy consumption, on average, by 56% and 72% compared to those of the PHAX and RIM methods.

The rest of the article is organized as follows. A review of the relevant prior works as well as the general structure of the *IM*-NNs along with the training methods of PHAX and RIM are briefly reviewed in Section II. In addition, the shortcomings of PHAX and RIM methods as well as the importance of selecting proper sizes for the inverters as the motivation of the proposed approach are also dealt with in this section. The proposed method is described in Section III while the results are discussed in Section IV. Finally, the paper is concluded in Section V.

II. Background

In this section, first, prior efforts conducted on reducing the severe effects of circuit elements non-idealities on the network accuracy are reviewed. Next, a mathematical model of an *IM* neuron is presented and two state-of-the-art offline training methods (PHAX [9] and RIM [15]) are studied in more detail. Finally, the motivation of proposing ERIM method is presented.

A. Prior Works

As discussed before, despite the attractive features of memristive crossbars for performing matrix vector multiplication (MVM) operations, they suffer from the accuracy degradation due to individual memristor non-idealities. These non-idealities may be caused by fabrication defects [7], [17], limited conductance states [16], limited write precision [17], or the conductance drift over time [8]. Due to the non-ideality of fabrication process, it is possible that the resistance of some memristors become fixed which results in stuck-at-faults (SAFs). To overcome this problem, one may extract the location and value of the faulty cells and train the network based on the profile of the defective cells [7]. The number of SAFs, however, can be reduced by improving the fabrication yield (i.e., 99.8% fabrication yield of [17]). In addition, online training of memristive NNs can alleviate their fabrication yield issues [8]. However, memristor conductance drift due to its retention limits may significantly degrade the NN accuracy during the inference phase which can be mitigated by employing an error correction approach [18].

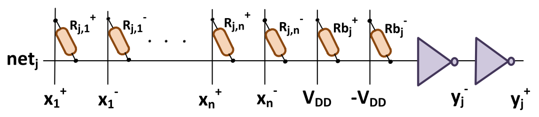


Fig. 1. An inverter-based memristive neuron [14].

Experimental results extracted from fabricated memristive crossbar of [17] revealed that the conductance of the programmed memristors may differ from their desired values where the errors follow Gaussian distributions. The errors can be mitigated using a larger number of closed-loop programming iterations in the cost of programming time increment [17]. Furthermore, the endurance of a memristor may be improved by reducing the number of writes on the memristor [19]. To overcome process variations of memristors, one may employ memristors connected in parallel with the overhead of higher cost and area occupation [20]. As another approach, one may extract the actual resistance variation of the programmed memristors and retrain the network to compensate the variation effects of the weights [21].

In the case of *IM*-NNs, the loading effect of the memristive crossbar and the effect of the process variation on the inverters may also induce large errors in both the training and inference phases. To reduce the effects of the characteristic variations on the accuracy of *IM*-NNs, several approaches have been proposed in the literature [12], [14], [15]. Due to the high VTC slope of inverters, a small variation on the input voltage of the inverter or the horizontal shift of the inverter VTC, may cause large errors on the NN output voltage. To mitigate the variation effects, in the RIM method [15], the VTC slope of the inverter was reduced by adding a grounded resistor to its output node which also reduced the loading effect of the memristive crossbar on the output voltage of each inverter with the overhead of increased power consumption.

As another technique, the inverter VTCs of a fabricated chip can be extracted [12]. The extracted VTCs, however, may not be applicable for modeling the behavior of the inverters connected to a large number of neurons as the VTC coefficients are proportional to the loading effect of the memristive crossbar which was not considered in [12]. As another approach, one may convert a regression question to a classification one to benefit from its higher resilience to variations [14]. In [14], the regression output was discretized and a classifier was trained to choose one of the discrete values as the approximate output value. This approach, however, is practical only for regression or function approximation applications. Therefore, among the existing variation mitigating approaches for *IM*-NNs, RIM is the most practical one.

B. Mathematical Model of an IM Neuron

Consider the j^{th} neuron of the l^{th} layer of a NN which has m^l inputs. Assume that the inputs, their corresponding weights, the bias, the activation function, and the output of this neuron are represented by x_i^l, $w_{j,i}^l$, b_j^l, f, and y_j^l, respectively. Then, the output of this neuron can be calculated as

$$y_j^l = f\left(net_j^l\right) = f\left(\sum\nolimits_{i=1}^{m^l} w_{j,i}^l \times x_i^l + b_j^l\right), \tag{1}$$

where $x_j^l = y_j^{l-1}$. The mathematical operations of (1) can be implemented using an *IM* neuron whose circuit is shown in Fig. 1. In this structure, memristors generate the required weights and biases while the activation functions are implemented by inverters. Writing the KCL equation for this neuron, assuming a zero current flowing into the input terminal of the inverter, leads to

$$net_j = \sum\nolimits_{i=1}^{n} (x_i{}^+ \times \frac{\sigma_{j,i}{}^+}{\gamma_j} + x_i{}^- \times \frac{\sigma_{j,i}{}^-}{\gamma_j}) + V_{DD} \times \left(\frac{\sigma b_j{}^+ - \sigma b_j{}^-}{\gamma_j}\right), \tag{2}$$

where, $\sigma_{j,i}$ represents the conductance of the memristor and $\gamma_j = \sum_{i=1}^{n} (\sigma_{j,i}{}^+ + \sigma_{j,i}{}^-) + \sigma b_j{}^+ + \sigma b_j{}^-$. In addition, $V_{DD}(-V_{DD})$ is equal to the positive (negative) source voltage of the inverters. To make the mathematical model of this neuron similar to (1), (2) may also be rewritten as

$$net_j = \sum_{i=1}^{n} (w_{j,i}{}^+ \times x_i{}^+ + w_{j,i}{}^- \times x_i{}^-) + V_{DD} \times \left(b_j{}^+ - b_j{}^-\right), \tag{3}$$

where the positive (negative) weights are equal to $w_{j,i}{}^+ = \sigma_{j,i}{}^+/\gamma_j$ ($w_{j,i}{}^- = \sigma_{j,i}{}^-/\gamma_j$) and the positive (negative) biases are equal to $b_j{}^+ = \sigma b_j{}^+/\gamma_j$ ($b_j{}^- = \sigma b_j{}^-/\gamma_j$). In the rest of the paper, the first (second) inverter of Fig. 1 is denoted by negative (positive) inverter.

C. PHAX Method

To train an *IM*-NN, it is required to mathematically model the limited conductance range of the employed memristors. Therefore, in the PHAX method [9], a continuous differentiable weight mapping function was employed as

$$w_{j,i}{}^+ = g\left(\theta_{j,i}{}^+\right) = g_2\left(g_1\left(\theta_{j,i}{}^+\right)\right), \tag{4}$$

where θ parameters are trainable variables which define the conductance of the memristors based on the following equations

$$g_2\left(\sigma_{j,i}{}^+\right) = \frac{\sigma_{j,i}{}^+}{\sum_{p=1}^{n}\left(\sigma_{j,p}^+ + \sigma_{j,p}^-\right) + \sigma b_j^+ + \sigma b_j^-} \tag{5}$$

$$\sigma_{j,i}{}^+ = g_1\left(\theta_{j,i}{}^+\right) = \sigma_{min} + \frac{\sigma_{max} - \sigma_{min}}{1 + e^{-\theta_{j,i}^+}}. \tag{6}$$

Here, σ_{max} and σ_{min} represent the maximum and the minimum conductance values of the memristors.

Furthermore, in the PHAX method, VTCs of a nominal inverter and buffer were extracted using HSPICE simulations and fitted to tangent hyperbolic functions ($a + b \times tanh(d$ (x-c))) using MATLAB simulations where (a, b, c, d) are the fitting parameters. Finally, back propagation approach was employed for training the network and calculating the conductance of the memristors. Further details of this training method can be found in [9]. It is worth noting that the fitting parameters of the VTCs depend on the resistive load of the inverters and this dependency is stronger for smaller sized inverters. Therefore, in the PHAX method, all inverters were sized 5× of a minimum sized inverter to mitigate this dependency. However, the dependency still exists and makes PHAX training method not practical for larger NNs where the number of neurons connected to the output node of each inverter is large.

D. RIM Method

High VTC slopes of inverters and buffers make the accuracy of the network very sensitive to the variations [15]. Note that in this work, the average VTC slope of the inverter is considered as its VTC slope (S) which is defined by

$$S = \frac{V_{out}\left(V_{in} = V_{IH}\right) - V_{out}\left(V_{in} = V_{IL}\right)}{V_{IH} - V_{IL}}. \tag{7}$$

According to [15], the inverter VTC slope can be approximated as

$$S \approx \left(g_{mn} + g_{mp}\right) \times \left(r_{o,n}||r_{o,p}||R_M\right) \tag{8}$$

where g_{mn} (g_{mp}) and $r_{o,n}$ ($r_{o,p}$) are the small signal parameters of the NMOS (PMOS) transistor and R_M is equal to the equivalent resistance of the memristive crossbar [15]. In RIM method [15], the inverter VTC slope was reduced by adding a grounded resistor with a low value to the output node of the inverter. This also improved the efficacy of the training method by reducing the sensitivity of the VTC fitting parameters (a, b, c, d) to R_M value. Even though decreasing the VTC slope results in a better accuracy when considering variations, it also results in lower training accuracy in the ideal case which is undesirable. This necessitates a trade-off to find appropriate resistance values for the grounded resistors. Simulation results of [15] revealed that, to ensure an acceptable trade-off between the accuracy of the network in the nominal and variation conditions, the output resistance of the negative (positive) inverter should be chosen such that the coefficient $d_n(b_p d_p)$ becomes equal to 15 (-1). Sizing of the inverters in RIM method is similar to that of PHAX. Further details of RIM method are provided in [15].

E. Motivation

As mentioned in [9], choosing larger sizes for the inverters reduces the sensitivity of the VTC coefficients to the loading effect of the memristive crossbars. In other words, by choosing larger size inverters, the impact of loading effect of the memristive crossbars on the VTC of the inverters reduces. Consequently, assuming constant coefficients for modeling the behavior of the inverters in the offline training phase (*i.e.*, as was done in RIM and PHAX methods) results in smaller errors when larger size inverters are utilized. On the other hand, by increasing the size of an inverter (*i.e.*, by k times), it is expected that its area occupation and power consumption increase by almost k times which are not desirable.

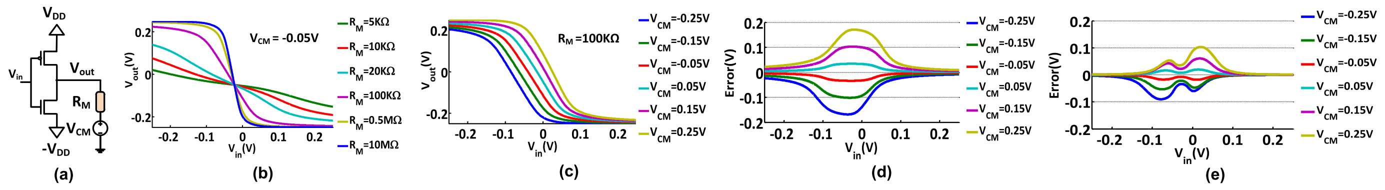


Fig. 2. a) The circuit of an inverter with a resistive load and its VTC for different b) R_M values with $V_{CM} = -0.05$V and c) V_{CM} values with $R_M = 100$KΩ. The VTC approximation error of the d) straight forward and e) proposed formulation for an inverter with size = 2 and $R_M = 100$KΩ.

The delay of the NN significantly depends on the resistance of the memristors that transfer the signals [20]. Furthermore, by increasing the size of inverters by k times, the equivalent output resistance of the inverters (their input and output capacitances) decreases (increase) by almost k times while the resistances of the memristors generating the weights and biases are unchanged. The output resistance of the inverter is connected in series with the memristors. Therefore, their equivalent resistance, which is equal to their sum, decreases by lower than k times. As the input capacitance of the inverter of the next layer is increased by k times, the equivalent RC of the considered path should increase resulting in larger delays. Therefore, increasing the inverter size results in larger area, power, and delay which are not desirable. This means that to have an acceptable accuracy without unnecessary increase in the area, power, and delay, it is required to have a high accuracy network model in which the sizes of the inverters are properly chosen. Therefore, we propose the ERIM method in which inverters are properly sized, thus avoiding excessive energy consumption and area occupation when having higher accuracies in network modeling (nominal condition) as well in the presence of variations.

III. The Proposed Method

In this section, by studying the impact of the memristive crossbar on the inverter VTC, an approach for choosing proper sizes for the inverters is proposed. Then, the proposed algorithm for training *IM*-NNs based on the proposed approach is presented.

A. Inverter VTC Modeling

As mentioned in [15], the equivalent resistance of the memristive crossbar may significantly change the VTC slope of the inverters. Therefore, to model the output voltage of inverters with high accuracy, it is required to consider the equivalent resistance of the memristors as well as the impact of output voltage of other inverters in the same layer. In this work, we employ the Thevenin equivalent circuit of the memristive crossbar to model the behavior of the inverters with a high accuracy. To do so, first, the Thevenin equivalent circuit of each memristive branch connected to the inverter output node is found and then, it is employed to model the output voltage of the inverter.

Output voltage of an inverter depends on the inverter size, its input voltage, and its load. To explain it better, consider the inverter of Fig. 2(a) where R_M (V_{CM}) represents the Thevenin equivalent resistance (voltage) of its memristive

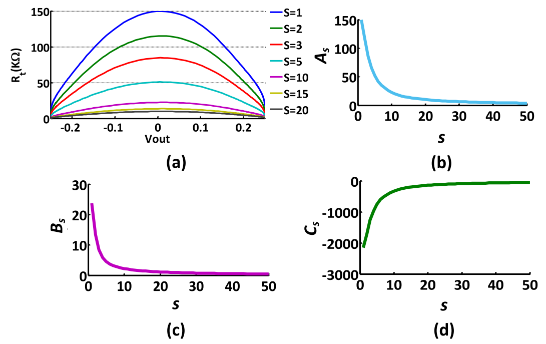


Fig. 3. a) Output resistance of inverters with different sizes based on the inverter output voltage. Diagram of b) A_s, c) B_s, and d) C_s versus the size of inverter (s).

crossbar. Output voltage of the inverter is shown in Fig. 2(b) and Fig. 2 (c) for different R_M and V_{CM} values. The slope of the inverter decreases when its resistive load is reduced. In addition, the VTC diagram moves vertically based on the V_{CM} value. The effect of V_{CM} on the inverter output voltage could be calculated exactly using proper circuit equations. For the sake of modeling simplicity, we calculate the inverter output voltage approximately as

$$V_{out} \approx f_n\left(V_{in}\right)|_{V_{CM}=0} + V_{CM}\frac{R_t}{R_t + R_M} \tag{9}$$

where $R_t = r_p||r_n$ and f_n is the VTC of the inverter assuming a resistive load of R_M with $V_{CM} = 0$. Here, r_n and r_p represent the resistances of the NMOS and PMOS transistors. To demonstrate the significance of the proposed formulation, its approximation error (the difference between the exact and approximate values of V_{out}) is compared to that of the straight forward approach in which V_{out} is approximated by $f_n\left(V_{in}\right)|_{V_{CM}=0}$. Results are depicted in Fig. 2 (d) and (e). The comparison reveals that the proposed method can reduce the error of VTC modeling considerably.

To find V_{out} based on (9), it is required to find R_t value. The amount of R_t based on the output voltage of the inverter is shown in Fig. 3 (a) where the size of the inverter is considered as the running parameter. As shown in this figure, the relation between the output resistance and output voltage of an inverter may follow the behavior of a parabolic function. This inspired us to use parabolic curves (i.e., with the function of $A_s + B_s V_{out} + C_s V_{out}^2$) to model the characteristics shown in curves of Fig. 3 (a). The coefficients A_s, B_s, and C_s are functions of the inverter size (see Fig. 3 (b)-(d)) which can be modeled by the function of $\alpha_\rho / \left(\beta_\rho + s\right)$ where ρ represents

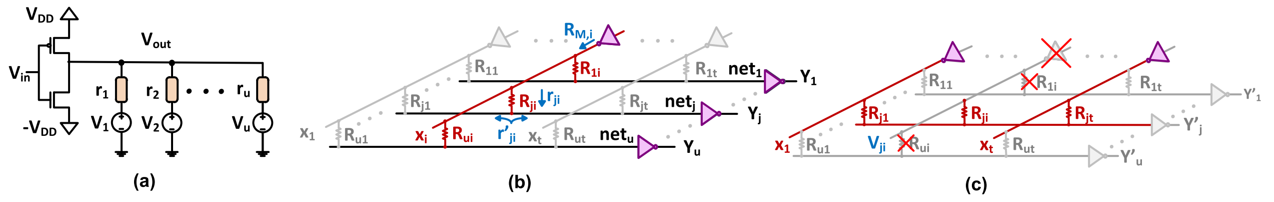


Fig. 4. a) Thevenin equivalent circuit of each branch of memristive crossbar connected to the output node of an inverter. b) Simplified circuit of a memristive crossbar to find the approximate value of R_M. c) Simplified circuit of a memristive crossbar to find the approximate value of V_{CM}.

the considered coefficient (*i.e.*, α_A represents coefficient α of the fitted curve to A_s).

To find R_M and V_{CM} values, consider a memristive crossbar with t inputs and u outputs. In this crossbar, output node of each inverter is connected to u memristive branches where the Thevenin equivalent circuit of each branch is shown in Fig. 4 (a). Note that r_j (V_j) represents the Thevenin equivalent resistance (voltage) of each memristive branch. As the circuit of Fig. 2 (a) is the Thevenin equivalent circuit of Fig. 4 (a), R_M and V_{CM} are obtained by

$$R_M = \frac{1}{\sum_{j=1}^{u} \frac{1}{r_j}}, \tag{10}$$

and

$$V_{CM} = R_M \sum_{j=1}^{u} \frac{V_j}{r_j}. \tag{11}$$

In the next step, r_j and V_j values should be found for each inverter. Consider Fig. 4 (b) which shows the structure of the memristive crossbar (for simplicity, we have not distinguished the biases from the other weights). r_j and V_j of the i^{th} input inverter are represented by r_{ji} and V_{ji} ($i \le t$ and $j \le u$). Note that R_{ji} represents the resistance of the memristor connecting i^{th} input to the j^{th} output. As shown in this figure, r_{ji} can be calculated as

$$r_{ji} = R_{ji} + r'_{ji}. \tag{12}$$

where r'_{ji} is the equivalent resistance of the branches connected to the other inputs of the considered crossbar. The number of these branches is equal to $(t-1)$ where their equivalent conductance can be calculated by adding the equivalent conductance of each branch (see Fig. 4 (b)). Note that connecting resistors in parallel results in an equivalent resistance which is smaller than the resistance of each resistor. Therefore, for a large t value, we can assume that $r'_{ji} \ll R_{ji}$ and hence, neglect its impact on r_{ji}. Therefore, we approximate r_{ji} as R_{ji}.

To find V_{ji}, we have to determine the open circuit voltage of the j^{th} branch which circuit is shown in Fig. 4 (c). In other words, to find V_{ji}, all inputs are applied to the crossbar except for i^{th} input (X_i) which is assumed to be a floated node. Therefore, the voltage may be calculated as

$$V_{ji} = \sum_{k \neq i} X_k \times \frac{\sigma_{jk}}{\gamma_{ji}}, \tag{13}$$

where $\gamma_{ji} = \sum_{k \neq i} \sigma_{jk}$. Let us consider the crossbar when it operates in its normal mode (all the inputs are exerted to the crossbar similar to Fig. 4 (b)). In this case, net_j can be calculated as

$$net_j = \sum_{k=1}^{t} X_k \times \frac{\sigma_{jk}}{\gamma_j}, \tag{14}$$

where $\gamma_j = \sum_{k=1}^{t} \sigma_{jk}$. Comparing (13) and (14), it can be inferred that the amount of V_{ji} and net_j would be close to each other, especially, for large t values, and hence, V_{ji} can be approximated as

$$V_{ji} \approx net_j. \tag{15}$$

Now, based on (10), $R_{M,i}$ (the Thevenin equivalent resistance of the i^{th} inverter) can be approximated as

$$R_{M,i} \approx \frac{1}{\sum_{j=1}^{u} \frac{1}{R_{ji}}}. \tag{16}$$

Also, based on (11) and (15), $V_{CM,i}$ (the Thevenin equivalent voltage of the i^{th} inverter) may be approximated as

$$V_{CM,i} \approx R_{M,i} \sum_{j=1}^{u} \frac{net_j}{R_{ji}}. \tag{17}$$

B. Inverter Sizing

To generate a characteristic for the inverter with low sensitivity to the loading effect of the memristive crossbar without increasing the inverter power consumption, choosing proper sizes for the inverters should be targeted. It should be mentioned that similar to [9] and [15], for an inverter of size s, $(W/L)_p = 2s$ and $(W/L)_n = 1.2s$ are assumed. To determine proper sizes, we employ the idea of the RIM method in which a grounded resistor was added to the output node of each inverter and its resistance was chosen by considering a trade-off between the accuracies of the network in the nominal and variation conditions. For given inverter sizes of s_n and s_p for the negative and positive inverters, we find R_n and R_p as the resistances at which the corresponding tangent hyperbolic coefficients d_n and $b_p d_p$ of the inverters become 15 and -1, respectively. The values of R_n and R_p for different inverter sizes are plotted in Fig. 5 which shows that a reciprocal function (*i.e.*, $R_n = A_n/(B_n + s_n)$) can model the relation between these quantities and the sizes of the inverters. The fitting coefficients are extracted using MATLAB simulations.

Next, proper sizes for the inverters should be selected based on their output resistive load as well as their required VTC slope. The equivalent resistance of the memristive crossbar (*i.e.*, R_{Mn} and R_{Mp} for the negative and positive inverters,

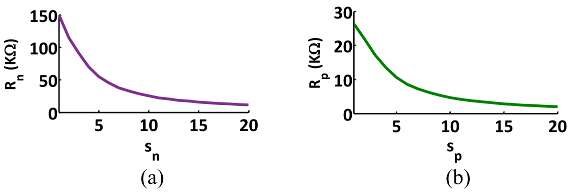


Fig. 5. The amounts of a) R_n and b) R_p versus the size of the inverter.

respectively) can be approximated based on (16). Assume that, similar to RIM method, a grounded resistor is connected to the output node of each inverter (R_{Gn} and R_{Gp} for the negative and positive inverters, respectively). As the VTC coefficients are proportional to the equivalent output resistance of the inverter (*i.e.*, $R_{Mn}||R_{Gn}$), to have inverters with coefficients d_n and b_pd_p almost equal to 15 and -1, respectively, the following equations should be satisfied.

$$R_n = \frac{1}{\frac{1}{R_{Mn}} + \frac{1}{R_{Gn}}} \tag{18}$$

$$R_p = \frac{1}{\frac{1}{R_{Mp}} + \frac{1}{R_{Gp}}} \tag{19}$$

Based on (18), it is obvious that R_{Mn} should be larger than R_n which leads to

$$R_n = \frac{A_n}{B_n + s_n} \leq R_{Mn}. \tag{20}$$

This sets a constraint on the size of the inverter as

$$\frac{A_n}{R_{Mn}} - B_n \leq s_n. \tag{21}$$

Due to the higher power consumption and area cost of larger inverters, we choose the size of the negative inverter to be

$$s_n = \left\lceil \frac{A_n}{R_{Mn}} - B_n \right\rceil. \tag{22}$$

Next, R_n can be extracted based on (20) which helps in finding the resistance of the grounded resistor (R_{Gn}) by employing (18). Similarly, the size of the positive inverter (s_p) and the resistance of its grounded resistor (R_{Gp}) can be chosen.

The same hardware may be used to run different NN applications. To have the capability of choosing the inverter sizes which reduce the power consumption, we propose to employ adjustable size inverters whose internal structure is depicted in Fig. 6 (a). In this structure, inverters with different sizes can be parallelized using transmission gates (TGs). An example of adjustable inverters composed of two and three inverters with different sizes is given in Fig. 6 (b). In the cases where larger size inverters are not connected to the crossbar, their corresponding input ports are floated by the TGs avoiding any dynamic power consumption. The structure of a flexible memristive crossbar composed of similar adjustable negative and positive inverters is drawn in Fig. 6 (c). Note that the required slope for the positive inverters is lower than that of the negative ones. This leads to smaller sizes for the positive inverters which are implemented using adjustable inverters composed of two inverters (see Fig. 6 (b) and Fig. 6 (c)). Since the delay of the NN depends on the capacitances of different nodes (see Subsection II.E), and the use of TGs increase the total capacitance of the network nodes, an efficient design employs a combination of fixed and adjustable size inverters in each layer of the NN as shown in Fig. 6 (d). This way, the area overhead decreases too.

C. ERIM Algorithm

In this part, the information provided in the previous subsections is utilized to propose a training method for *IM*-NNs, called ERIM, which improves the power consumption of the NN while simultaneously reduces the severe effects of circuit parameter variations. The flowchart of ERIM approach is presented in Fig. 7. First, the size of the inverters, their grounded resistors, and the activation functions are determined based on RIM method. Next, the first epoch of training is performed and the conductance values of the memristors are determined. The loading effect of the memristive crossbar is then approximated based on (16) and (17). In addition, sizes of the inverters as well as resistance values of the grounded resistors are chosen based on (18) and (22). As the next step, the activation functions are updated based on (9). Finally, the termination condition is checked. If the termination condition is not satisfied, another epoch of training is performed and the above steps are repeated. In this paper, the number of training epochs is considered as the termination condition.

IV. Results and Discussion

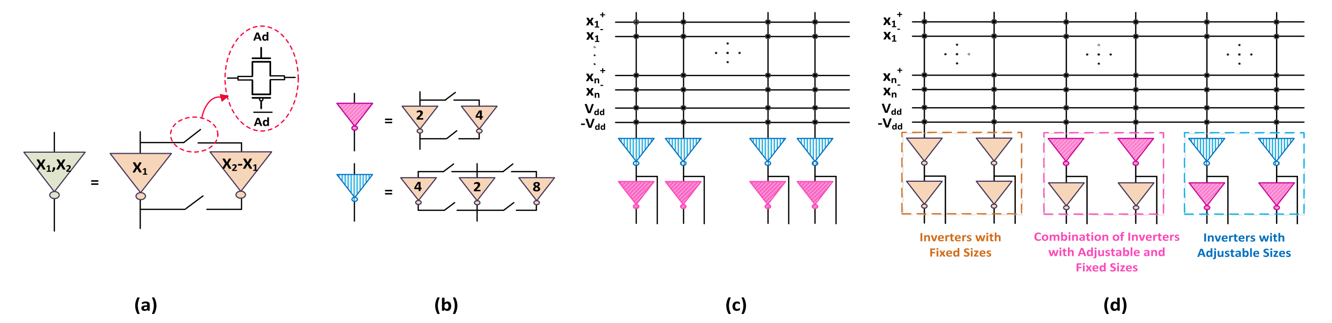
In this section, first, the training accuracy of the proposed method is compared with those of PHAX and RIM methods. Next, the design parameters (delay, power, and energy) of the *IM*-NNs trained by these methods are compared. Finally, the accuracy of the networks in the presence of memristors and transistors non-idealities is studied.

A. Simulation Setup

All networks were trained using TensorFlow where Adam optimizer was employed as the training optimizer. To extract the accuracies of the networks as well as their delays and power consumptions, HSPICE simulations were performed where a 90nm technology and the memristor model of [22] were utilized. The employed memristor had maximum and minimum conductance values of $8\mu\mho$ and $0.12\mu\mho$, respectively [13] and the voltage sources of the network were considered to be equal to ±0.25V. In the initialization step of ERIM, the sizes of all inverters were chosen to be five while R_{Gn} and R_{Gp} for all the negative and positive inverters were selected to be $50K\Omega$ and $10K\Omega$, similar to [15].

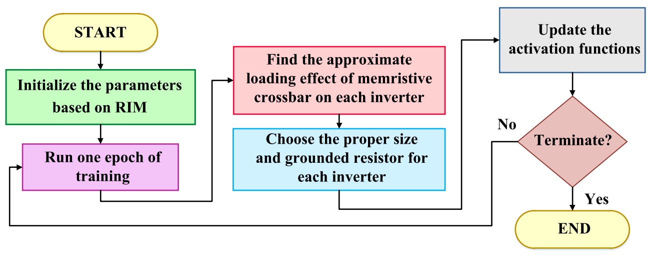
B. Training Accuracy

Output accuracies of the networks trained by PHAX, RIM, and ERIM methods are reported in Table I. They were measured using Python simulations which include simple mathematical model of the network employed in the training phase of the considered methods. For the IRIS and BCW datasets which include 120 (30) and 546 (136) train (test)



Ad
Ad
X1,X2
X1
X2-X1
2
4
4
2
8
Vdd
-Vdd
Inverters with Fixed Sizes
Combination of Inverters with Adjustable and Fixed Sizes
Inverters with Adjustable Sizes
(a)
(b)
(c)
(d)

Fig. 6. a) Proposed implementation of an inverter with adjustable size. b) An example of adjustable inverters composed of inverters with different sizes. c) The structure of a memristive crossbar composed of similar adjustable size inverters. d) The structure of a memristive crossbar composed of fixed and different adjustable size inverters.



START
Initialize the parameters based on RIM
Run one epoch of training
Find the approximate loading effect of memristive crossbar on each inverter
Choose the proper size and grounded resistor for each inverter
Update the activation functions
Terminate?
No
Yes
END

Fig. 7. Flowchart of ERIM.

TABLE I

COMPARISON OF THE ACCURACIES OF THE NETWORKS TRAINED BY PHAX, RIM, AND ERIM APPROACHES EXTRACTED BY PYTHON SIMULATIONS

Dataset	*Network Configuration*	*Training method*	*Train Accuracy (%)*	*Test Accuracy (%)*
IRIS	$4 \rightarrow 5 \rightarrow 3$	PHAX	99.2	100
		RIM	95.8	96.7
		ERIM	95.8	96.7
BCW	$10 \rightarrow 5 \rightarrow 2$	PHAX	99.3	95.6
		RIM	97.6	96.3
		ERIM	97.6	96.3
MNIST	$784 \rightarrow 30 \rightarrow 40 \rightarrow 10$	PHAX	97.9	95.7
		RIM	90.7	91.1
		ERIM	91.1	91.5
Fashion MNIST	$784 \rightarrow 30 \rightarrow 40 \rightarrow 10$	PHAX	92.3	87.0
		RIM	85.0	83.4
		ERIM	84.8	83.2

samples, 1,000 epochs for training were considered while for MNIST and Fashion MNIST datasets with 60,000 (10,000) train (test) samples, 100 epochs of training were performed. The number of inputs, outputs, and hidden neurons of each network are reported as the network configuration in Table I. As was expected, reducing the VTC slope of the inverters in RIM and ERIM methods results in lower accuracies compared to that of PHAX method. As will be shown later, however, RIM and ERIM approaches lead to higher accuracies in the presence of variations as well as better matching between the results generated by HSPICE and high level (*i.e.*, Python) simulations.

The chosen sizes and grounded resistor values of some inverters (i.e., neurons) of the first layer of IM-NNs for the MNIST dataset versus the epochs of ERIM training approach are depicted in Fig. 8 (a) and Fig. 8 (b). Since the number of inverters in the first layer of the considered NN was 60, for a better illustration, we selected some of the neurons and plotted the updating procedure of their sizes and grounded resistor values. The index of the considered inverter is shown as the legend of the figures. In addition, the training accuracy of the NNs trained by PHAX, RIM, and ERIM methods at different training epochs are provided in Fig. 8 (c). The distributions of the conductance values of the trained NNs for BCW dataset are also depicted in Fig. 8 (d)-(f) which are utilized for justifying the electrical parameters of the trained NNs for BCW dataset.

To evaluate the efficacy of the training methods, it is required to measure the accuracy of the networks using HSPICE simulations which include high accuracy model of the network. For this purpose, we selected 1,000 samples from the training datasets and measured the following parameters for each network:

- Acc_{PySim} : The percentage of the samples correctly classified by Python simulations.
- Acc_{SpSim} : The percentage of the samples correctly classified by SPICE simulations.
- $PySp\_Match$: The percentage of samples whose selected classes by Python and SPICE simulations are the same.
- O_i_err : The mean absolute difference between the voltage of the i^{th} layer outputs extracted by Python and SPICE simulations.

The parameters are reported in Table II which demonstrate that for smaller-sized NNs (*i.e.*, for IRIS and BCW benchmarks), the networks trained by PHAX exhibit higher accuracy (Acc_{SpSim}) compared to RIM and ERIM. For larger-sized NNs (*i.e.*, for MNIST and Fashion MNIST benchmarks), the training accuracy of PHAX degrades, while RIM and ERIM methods can model the NN outputs with high accuracy ($PySp\_Match$ values more than 94%). In addition, for larger-sized NNs trained by RIM or ERIM methods,

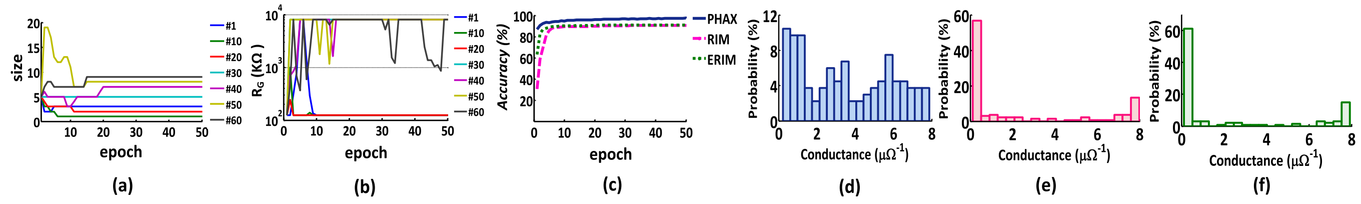


Fig. 8. The procedure of updating a) the size and b) resistance of the grounded resistor of inverters of the first layer of an IM-NN under MNIST benchmark during the training phase of ERIM. c) Training accuracy of the IM-NNs trained by PHAX, RIM, and ERIM methods under MNIST benchmark versus the training epochs. Distribution of memristors conductance values of the NNs trained by d) PHAX, e) RIM, and f) ERIM methods for BCW datasets.

TABLE II

COMPARISON OF THE ELECTRICAL PARAMETERS AS WELL AS MODELING ACCURACIES OF PHAX, RIM, AND ERIM TRAINING METHODS

Dataset	*Training Method*	Acc_{PySim} (%)	Acc_{SpSim} (%)	*PySp_Match* (%)	O_1*_err* (mV)	O_2*_err* (mV)	O_3*_err* (mV)	*Delay* (ns)	*Power* (μW)	*Energy* (fJ)
IRIS	PHAX	99.2	97.5	98.3	13	7	-	4.0	5.8	23.1
	RIM	95.8	95.8	100	18	8	-	1.5	30.5	45.8
	ERIM	95.8	95.8	100	21	10	-	1.1	9.6	10.5
BCW	PHAX	99.3	98.5	99.3	18	4	-	3.1	6.2	19.3
	RIM	97.6	97.4	99.8	19	3	-	1.2	29.3	35.1
	ERIM	97.6	97.6	99.6	36	7	-	1.0	9.0	9.0
MNIST	PHAX	97.3	28.0	28.5	162	151	106	2.1	164.7	345.9
	RIM	90.8	89.8	96.8	33	55	45	1.6	315.7	505.1
	ERIM	91.4	91.1	97.7	27	21	12	1.3	110.2	143.3
Fashion MNIST	PHAX	92.6	26.6	26.9	161	244	263	2.1	174.6	366.7
	RIM	85.5	83.0	94.6	31	56	39	1.4	304.7	426.6
	ERIM	85.4	83.6	95.0	25	21	12	1.4	106.4	149.0

the modeling error of each layer output voltage (O_i_err) is considerably lower than that of the PHAX method. As an example, O_3_err (the modeling error of the output layer) of ERIM is, on average, 8.8× (21.9×) smaller than that of PHAX for the MNIST (Fashion MINIST) dataset. Furthermore, the accuracies (Acc_{SpSim}) of the networks trained by ERIM are slightly higher than those of RIM. Additionally, in ERIM, the modeling error of the hidden layer output voltages are reduced, on average, by 40% for MNIST and Fashion MNIST datasets compared to those of the RIM method.

C. Hardware Evaluation

To evaluate the impact of different training methods on the hardware implementations of the networks, their delays, power dissipations, and energy consumptions were extracted using HSPICE simulations where the results are reported in Table II. As the results reveal, the delays of the networks trained by RIM or ERIM are considerably lower than those of PHAX which is compatible with the results of [15]. It is worth noting that the delay of the network inversely depends on the conductance of the memristors which transfer the signal. Let us consider the distribution of memristor conductance values for the BCW dataset for PHAX, RIM, and ERIM methods which were plotted in Fig. 8 (d)-(f). As shown in these figures, in the latter two approaches, most of the conductance values are almost equal to $0.12\mu\Omega^{-1}$ or $8\mu\Omega^{-1}$ where the first value represents weights with the value of ≈0 not having a significant impact on the *net* voltages of the inverters. This implies that signals pass through the memristors with conductance of $8\mu\Omega^{-1}$ to generate the *net* voltages. However, in PHAX structure, many conductance values are in the range of $\left[1\mu\Omega^{-1}, 8\mu\Omega^{-1}\right]$. This shows that the *net* voltages of the inverters depend on almost all of the inputs of that neuron. As the delay of the network inversely depends on the conductance of the memristors, the delay of the NNs trained by PHAX is higher. In addition, in ERIM, inverters are properly sized based on their equivalent output resistances. This results in considerable improvement in the power consumption compared to that of RIM, and hence, on average, 66% reduction in the power consumption compared to RIM method as well as energy consumption improvement up to 59% compared to PHAX method are achieved.

D. Network Accuracy in the Presence of Variations

To evaluate the accuracy of the trained NNs in the presence of variations, two scenarios in which Gaussian random variations were applied to the memristor conductance [5], [14], [15], [19], [23] or the threshold voltage, width, and, length of the transistors [9], [11]–[15] were studied. These scenarios are denoted by 10VAR_M (20VAR_M) with the variability (σ/μ) of 10% (20%) exerted to the memristors conductance and 5VAR_T (10VAR_T) with the variability of 5% (10%) applied to the transistors parameters. Furthermore, 1,000 simulations were performed to extract the mean and standard deviation of the network's output accuracy, denoted by μ_{Acc} and σ_{Acc}. It is worth noting that in the memristor model used in this study, the memristor conductance values were assumed to be changed continuously as a function of the applied voltage pulse duration or amplitude [13]. Since the conductance levels of the actual memristors are discrete, we also considered the impact of limited conductance levels (*i.e.*, the ability of storing *k*-bit data) of the memristors in the study. Results of the studies are reported in Table III which reveal that the adverse effects of variations on the

TABLE III

COMPARING THE ACCURACY OF THE NETWORKS TRAINED BY PHAX, RIM, AND ERIM APPROACHES IN THE CONSIDERED BENCHMARKS UNDER DIFFERENT VARIATION CONDITIONS AS WELL AS LIMITED CONDUCTANCE LEVELS OF MEMRISTORS

Dataset	*Training Method*	*Memristor Precision*			*10VAR_M*		*20VAR_M*		*5VAR_T*		*10VAR_T*	
		2-bit	*3-bit*	*4-bit*	μ_{Acc} (%)	σ_{Acc} (%)	μ_{Acc} (%)	σ_{Acc} (%)	μ_{Acc} (%)	σ_{Acc} (%)	μ_{Acc} (%)	σ_{Acc} (%)
IRIS	PHAX	96.7	96.7	96.7	96.4	4.1	93.6	7.7	97.3	3.3	93.3	7.2
	RIM	93.3	96.7	96.7	95.8	1.5	94.4	2.9	95.8	1.7	93.9	3.6
	ERIM	93.3	96.7	96.7	96.4	1.0	94.9	2.8	96.0	1.6	94.2	3.4
BCW	PHAX	95.6	94.8	96.3	95.8	0.5	95.6	0.5	95.7	0.5	95.2	1.7
	RIM	96.3	96.3	96.3	96.1	0.3	95.9	0.4	95.9	0.4	95.7	0.5
	ERIM	95.6	95.6	95.6	95.9	0.4	95.9	0.4	96.0	0.4	96.0	0.4
MNIST	RIM	89.0	88.7	89.0	89.0	0.2	88.8	0.3	86.6	1.3	81.0	4.4
	ERIM	89.1	89.1	89.1	89.0	0.3	88.7	0.4	88.2	0.8	85.0	2.7
Fashion MNIST	RIM	82.2	81.9	82.2	82.3	0.3	82.3	0.4	80.8	1.3	76.7	3.3
	ERIM	82.5	82.5	82.6	82.7	0.3	82.5	0.5	81.8	1.1	78.9	3.1

TABLE IV

COMPARISON OF THE ELECTRICAL PARAMETERS AS WELL AS MODELING ACCURACIES OF DIFFERENT IMPLEMENTATIONS OF NNs TRAINED BY ERIM METHOD

Dataset	*NN Structure*	Acc_{PySim} (%)	Acc_{SpSim} (%)	$PySp\_Match$ (%)	O_1_err (mV)	O_2_err (mV)	O_3_err (mV)	*Delay* (ns)	*Power* (μW)	*Energy* (fJ)
MNIST	Fixed	90.8	89.6	96.7	25	56	32	1.4	535	739
	FAd		89.8	95.4	26	30	19	1.5	148	221
	AAd		88.9	94.9	28	53	33	2.9	223	642
Fashion MNIST	Fixed	84.1	82.5	95.2	18	44	18	1.3	541	679
	FAd		83.0	95.7	20	21	8	1.5	159	245
	AAd		82.1	94.2	21	40	16	2.7	229	618

network accuracy are mitigated when RIM or ERIM methods are employed. As an example, ERIM outperforms PHAX in all the cases except for the IRIS benchmark in 5VAR_T condition. Also, σ_{Acc} of the networks trained by RIM and ERIM methods are considerably lower than that of PHAX method. In addition, ERIM outperforms RIM method in all cases except for BCW and MNIST benchmarks under 10VAR_M and 20VAR_M conditions where slight reductions of 0.2% and 0.1% in μ_{Acc} are observed. Furthermore, the results reveal that in larger NNs (*i.e.*, for MNIST and Fashion MNIST datasets), the sensitivity of the network output accuracy to the transistors variations increases, resulting in larger reductions in the network accuracy. This reduction is smaller for ERIM compared to that of RIM though. In addition, the results reveal that considering small number of discrete levels for the memristors does not have a significant impact on the accuracy of the NNs trained by the RIM or ERIM methods. This originates from the fact that the conductance values of the memristors for the networks trained by these methods are usually stuck at $\sigma_{\max}$ or $\sigma_{\min}$.

As mentioned in Section III.B, the same hardware may be used to run different NN applications. Assume a 3-layer NN with 64 neurons in the hidden layers and 16 outputs is utilized to predict the outputs of MNIST and Fashion MNIST datasets. The number of required hidden neurons for the considered benchmarks is assumed to be 30 and 40 while the number of outputs is 10. To evaluate the efficacy of our proposed method in which adjustable size inverters are used to reduce the power consumption, we considered three implementations for the NN. In the first structure (denoted by Fixed in Table IV), all of the hidden neurons were implemented using fixed size inverters with the size of 14. In the second structure (denoted by AAd in Table IV), all of the negative (positive) inverters were realized using adjustable inverters composed of three (two) inverters similar to those in Fig. 6 (c). In the third structure (denoted by FAd in Table IV), the NN consisted of a combination of fixed and adjustable size inverters. In this structure, 30 hidden neurons were implemented using fixed size positive and negative inverters with the size of 1. In addition, 25 hidden neurons were implemented using adjustable (fixed) size negative (positive) inverters where the adjustable inverters were composed of two inverters, similar to those in Fig. 6 (b), and the fixed size inverters had the size of 2. The rest of neurons were implemented using adjustable inverters composed of three (two) inverters for realizing negative (positive) inverters (similar to Fig. 6 (d)).

To evaluate the efficacy of our proposed method, the training was performed for MNIST and Fashion MNIST datasets and the trained neurons were mapped on the considered NN structures. The electrical parameters as well as the accuracy of the NNs are reported in Table IV. The results reveal that using large fixed size inverters results in acceptable accuracies while leads to a significant increase in the power consumption. On the other hand, by employing adjustable size inverters (AAd structure), the power consumption is reduced compared to the Fixed structure with the overhead of

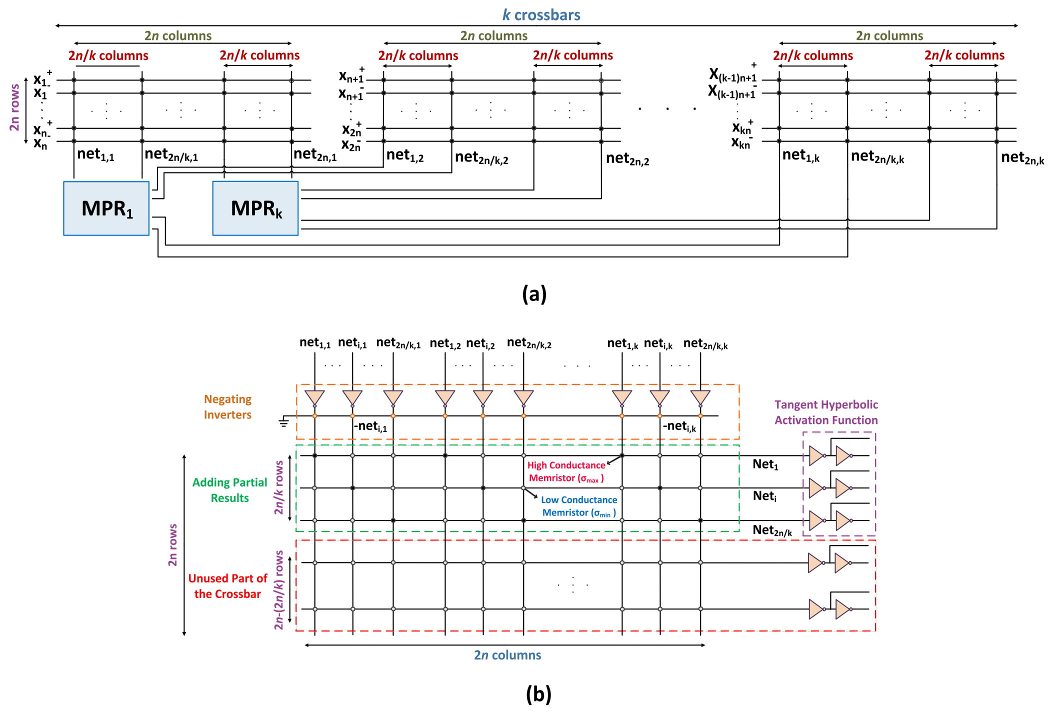


Fig. 9. a) Circuit level implementation of IM-NNs considering a limited size for the memristor crossbars. b) The internal structure of MPR unit.

increased delay due to the added transmission gates. Employing a combination of the fixed and adjustable size inverters (FAd structure) leads to a better implementation which has acceptable accuracy and delay as well as the lowest power consumption.

It is worth noting that in the case where the memristive crossbar size is not large enough to implement the whole weights of a layer, the weights of each layer can be distributed on several crossbars where the partial results of each crossbar can be merged to generate the final results of the considered layer. To do so, negating inverters (for generating the partial results) and additional memristor crossbars (for averaging the partial results) can be added to the hardware implementation. As an example, assume that the total size of the available memristive crossbars is $2n \times 2n$ where the inputs of the network are distributed among k individual crossbars (as shown in Fig. 9 (a)). In this structure, $net_{i,j}$ represents the i^{th} partial *net* voltage generated by the j^{th} crossbar. Each memristive crossbar can be divided into k parts (each part has $2n/k$ columns) and apply them to Merge Partial Results (MPR) unit as shown in Fig. 9 (a). The internal structure of MPR unit which consists of the negating inverters, memristive crossbars (for producing the average of partial results), and activation function generators is shown in Fig. 9 (b). The negating inverters whose VTC slopes are reduced by the grounded memristors generate almost $-net_{i,j}$ values. Writing KCL equations for the node Net_i results in

$$Net_i = \sum_{j=1}^{k}\left(-net_{i,j} \times \frac{\sigma_{max}}{\gamma_i}\right) + \sum_{j=1}^{k}\sum_{p\neq i}\left(-net_{p,j} \times \frac{\sigma_{min}}{\gamma_i}\right), \quad (23)$$

where $\gamma_i = k\sigma_{max} + (2n-k)\sigma_{min}$. Assuming $\sigma_{max} \gg \sigma_{min}$, one may approximate Net_i as

$$Net_i \approx \sum_{j=1}^{k}\left(-net_{i,j} \times \frac{\sigma_{max}}{\gamma_i}\right). \quad (24)$$

In other words, the amount of Net_i can be approximated by the weighted sum (or almost average) of $-net_{i,j}$ values for $j \leq k$. Therefore, the partial results ($net_{i,j}$) can be merged and then applied to other inverters with larger VTC slopes for generating the required tangent hyperbolic activation functions.

V. Conclusion

In this paper, an offline training method, called ERIM, was presented for training inverter-based memristive neural networks (*IM*-NNs). In this method, the loading effect of the memristive crossbar on the output voltage of the inverters was approximated and employed to select proper sizes for the inverters. High accuracy of the proposed NN mathematical model resulted in better matching between the results generated by Python and HSPICE simulations compared to the case of PHAX, a recently proposed offline training method for *IM*-NNs. In addition, similar to RIM, another recently proposed training approach, to mitigate the severe effects of variations on the network accuracy, the VTC slope of the inverters was reduced by adding grounded resistors to the inverter output nodes. Due to proper selection of the inverters sizes, the power consumption of the networks trained based on ERIM method was considerably lower than that of RIM method. In addition, ERIM resulted in higher accuracy in the presence of variations.

REFERENCES

[1] N. Zheng and P. Mazumder, "Artificial neural networks in hardware," in *Learning in Energy-Efficient Neuromorphic Computing: Algorithm and Architecture Co-Design*. Hoboken, NJ, USA: Wiley, 2020, pp. 61–118.

[2] X. Si *et al.*, "24.5 A twin-8T SRAM computation-in-memory macro for multiple-bit CNN-based machine learning," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2019, pp. 396–398.

[3] S. Vahdat, M. Kamal, A. Afzali-Kusha, and M. Pedram, "TOSAM: An energy-efficient truncation- and rounding-based scalable approximate multiplier," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 5, pp. 1161–1173, May 2019.

[4] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*. [Online]. Available: https://arxiv.org/abs/1510.00149

[5] P. Yao *et al.*, "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, Jan. 2020.

[6] X. Chen, J. Jiang, J. Zhu, and C.-Y. Tsui, "A high-throughput and energy-efficient RRAM-based convolutional neural network using data encoding and dynamic quantization," in *Proc. 23rd Asia South Pacific Design Automat. Conf. (ASP-DAC)*, Jeju, South Korea, Jan. 2018, pp. 123–128.

[7] I. Yeo, M. Chu, S.-G. Gi, H. Hwang, and B.-G. Lee, "Stuck-at-fault tolerant schemes for memristor crossbar array-based neural networks," *IEEE Trans. Electron Devices*, vol. 66, no. 7, pp. 2937–2945, Jul. 2019.

[8] J. Chen *et al.*, "High-precision symmetric weight update of memristor by gate voltage ramping method for convolutional neural network accelerator," *IEEE Electron Device Lett.*, vol. 41, no. 3, pp. 353–356, Mar. 2020.

[9] M. Ansari *et al.*, "PHAX: Physical characteristics aware *ex-situ* training framework for inverter-based memristive neuromorphic circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 8, pp. 1602–1613, Aug. 2018.

[10] S. Vahdat, M. Kamal, A. Afzali-Kusha, and M. Pedram, "LATIM: Loading-aware offline training method for inverter-based memristive neural networks," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, early access, Apr. 9, 2021, doi: 10.1109/TCSII.2021.3072289.

[11] A. Fayyazi, M. Ansari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "An ultra low-power memristive neuromorphic circuit for Internet of Things smart sensors," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1011–1022, Apr. 2018.

[12] A. BanaGozar, M. A. Maleki, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Robust neuromorphic computing in the presence of process variation," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Lausanne, Switzerland, Mar. 2017, pp. 440–445.

[13] M. Ansari, A. Fayyazi, M. Kamal, A. Afzali-Kusha, and M. Pedram, "OCTAN: An on-chip training algorithm for memristive neuromorphic circuits," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 12, pp. 4687–4698, Dec. 2019.

[14] S. Vahdat, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Interstice: Inverter-based memristive neural networks discretization for function approximation applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 7, pp. 1578–1588, Jul. 2020.

[15] S. Vahdat, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Offline training improvement of inverter-based memristive neural networks using inverter voltage characteristic smoothing," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 67, no. 12, pp. 3442–3446, Dec. 2020.

[16] W.-Q. Pan *et al.*, "Strategies to improve the accuracy of memristor-based convolutional neural networks," *IEEE Trans. Electron Devices*, vol. 67, no. 3, pp. 895–901, Mar. 2020.

[17] C. Li *et al.*, "Analogue signal and image processing with large memristor crossbars," *Nature Electron.*, vol. 1, no. 1, pp. 52–59, Jan. 2018.

[18] B. Li, Y. Wang, Y. Chen, H. H. Li, and H. Yang, "ICE: Inline calibration for memristor crossbar-based computing engine," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2014, pp. 1–4.

[19] Y. Luo and S. Yu, "Accelerating deep neural network *in-situ* training with non-volatile and volatile memory based hybrid precision synapses," *IEEE Trans. Comput.*, vol. 69, no. 8, pp. 1113–1127, Aug. 2020.

[20] J. Rajendran, R. Karri, and G. S. Rose, "Improving tolerance to variations in memristor-based applications using parallel memristors," *IEEE Trans. Comput.*, vol. 64, no. 3, pp. 733–746, Mar. 2015.

[21] S. Jin, S. Pei, and Y. Wang, "A variation tolerant scheme for memristor crossbar based neural network designs via two-phase weight mapping and memristor programming," *Future Gener. Comput. Syst.*, vol. 106, pp. 270–276, May 2020.

[22] C. Yakopcic, T. M. Taha, G. Subramanyam, and R. E. Pino, "Generalized memristive device SPICE model and its application in circuit design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 8, pp. 1201–1214, Aug. 2013.

[23] H. Kim *et al.*, "Efficient precise weight tuning protocol considering variation of the synaptic devices and target accuracy," *Neurocomputing*, vol. 378, pp. 189–196, Feb. 2020.

Shaghayegh Vahdat received the B.Sc. and M.Sc. degrees in electrical engineering from the University of Tehran, Tehran, Iran, in 2014 and 2016, respectively, where she is currently pursuing the Ph.D. degree in electrical engineering with the School of Electrical and Computer Engineering. Her current research interests include approximate computing, neuromorphic computing, hardware implementation of neural systems, reliability of neural networks at the presence of process variation, and low-power high-performance design of digital arithmetic units.

Mehdi Kamal (Senior Member, IEEE) received the B.Sc. degree from the Iran University of Science and Technology, Tehran, Iran, in 2005, the M.Sc. degree from the Sharif University of Technology, Tehran, in 2007, and the Ph.D. degree from the University of Tehran, Tehran, in 2013, all in computer engineering. He is currently an Assistant Professor with the School of Electrical and Computer Engineering, University of Tehran. His current research interests include reliability in nanoscale design, approximate computing, neuromorphic computing, design for manufacturability, embedded systems design, and low-power design.

Ali Afzali-Kusha (Senior Member, IEEE) received the B.Sc. degree from the Sharif University of Technology, Tehran, Iran, in 1988, the M.Sc. degree from the University of Pittsburgh, Pittsburgh, PA, USA, in 1991, and the Ph.D. degree from the University of Michigan, Ann Arbor, MI, USA, in 1994, all in electrical engineering. He was a Post-Doctoral Fellow with the University of Michigan from 1994 to 1995. He has been with the University of Tehran since 1995, where he is currently a Professor with the School of Electrical and Computer Engineering and the Director of the Low-Power High-Performance Nanosystems Laboratory. He was a Research Fellow with the University of Toronto, Toronto, ON, Canada, and the University of Waterloo, Waterloo, ON, Canada, in 1998 and 1999, respectively. His current research interests include low-power high-performance design methodologies from the physical design level to the system level, new memory, and digital design and implementation paradigms.

Massoud Pedram (Fellow, IEEE) received the Ph.D. degree in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, CA, USA, in 1991.

He is currently the Stephen and Etta Varra Professor with the Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA, USA. He holds ten U.S. patents and has authored four books, 12 book chapters, and over 140 archival and 350 conference papers. His current research interests include low-power electronics, energy-efficient processing, and cloud computing to photovoltaic cell power generation, energy storage, and power conversion, and RT level optimization of VLSI circuits to synthesis and physical design of quantum circuits. He and his students have received six conferences and two IEEE Transactions best paper awards for the research. He was a recipient of the 1996 Presidential Early Career Award for Scientists and Engineers and an ACM Distinguished Scientist. He was the Founding Technical Program Co-Chair of the 1996 International Symposium on Low-Power Electronics and Design and the Technical Program Chair of the 2002 International Symposium on Physical Design. He has served as the Editor-in-Chief for the *ACM Transactions on Design Automation of Electronic Systems*. He has also served on the Technical Program Committee of a number of premiere conferences in his field.