LATIM: Loading-Aware Offline Training Method for Inverter-Based Memristive Neural Networks

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

***Abstract*—In this brief, we present a high accuracy training method for inverter-based memristive neural networks (*IM*-NNs). The method, which relies on accurate modeling of the circuit element characteristics, is called LATIM (Loading-Aware offline Training method for Inverter-based Memristive NNs). In LATIM, an approximation method is proposed to estimate the effective load of the memristive crossbar (as the synapses) while two NNs are utilized to predict the voltage transfer characteristic (VTC) of the inverters (as the activation functions). Efficacy of the proposed method is compared with the recent offline training methods for *IM*-NNs, called PHAX and RIM. Simulation results reveal that LATIM can predict the output voltage of the *IM*-NNs, on average, by 14× (6×) and 29× (4×) smaller error for the MNIST and Fashion MNIST datasets, respectively, compared to those of PHAX (RIM) method. In addition, *IM*-NNs trained by LATIM consume, on average, 62% and 53% lower energy compared to PHAX and RIM methods due to proper sizing of the inverters.**

***Index Terms*—Inverter-based memristive neural networks, offline training, effective load of memristive crossbar, VTC prediction, proper sizing.**

I. Introduction

NEURAL networks (NNs) have gained considerable attention due to their large application domain such as image classification [1]. The high energy consumption of NNs, however, has become a challenge for data centers as well as portable or battery-powered devices [2]. To overcome this challenge in digital platforms, several approaches, such as approximate computing [3] and pruning redundant weights or neurons [4], have been utilized. Employing non-digital implementations such as the use of memristors for realizing the NN weights is a promising new approach, which offers considerably lower energy consumption as well as higher computational speed compared to its digital counterparts [5]. In this approach, the matrix-vector multiplications (MVMs) of NNs, which consume a large portion of the computation energy, are implemented by using memristive crossbars. As an example, the fabricated memristor-based NN of [6] led to 30× higher speed and 110× lower energy consumption, compared to a state-of-the-art GPU platform for the MNIST benchmark when achieving a classification accuracy of about 96%.

The activation function of memristive NNs may be implemented by employing operational amplifiers (Op-Amp) [7] or inverters [8]–[14]. In the case of *IM*-NNs, the VTC curves of the inverters generate the required activation functions which lead to up to 800× lower power consumption compared to the case of using Op-Amp-based neurons [8]. Obviously, the speed and power consumption improvements are significantly higher compared to the digital implementation of NNs (i.e., up to 9× and 960× improvements in the speed and power efficiency under Breast Cancer Wisconsin (BCW) benchmark [8]).

Despite these advantages of the inverter-based structures, there is no accurate mathematical circuit model due to the high complexity of the modeling required for the training of these NNs. The work in [8], called PHAX (which stands for physical characteristics aware ex-situ training approach), considered similar activation functions for all neurons of the network and presented a method to model the limited set of memristors resistance values for the offline training of *IM*-NNs. The VTC of the inverter, however, depends strongly on its size (output drive strength) as well as its load resistance, which were not considered in the PHAX training method. This makes the approach inappropriate for NNs with a large number of hidden neurons since the training accuracy strongly depends on how well the activation functions are modeled. To reduce the sensitivity of the inverters VTC coefficients with respect to the loading effect of the memristive crossbars, grounded resistors with high conductance values were connected to the output node of the inverters in RIM method (which stands for offline training of resistor-inverter based memristive neural networks) [14]. Although this approach results in higher modeling accuracy compared to PHAX, it has power overhead due to the added grounded resistors. In addition, circuit elements non-idealities may degrade the accuracy of the *IM*-NNs. Several approaches (i.e., [15], [16], [17]) have been presented in the literature to mitigate the non-ideality issues of memristors. Furthermore, [11], [13], and [14] are the recent works conducted on reducing the non-ideality issues of transistors in *IM*-NNs.

To employ *IM* neurons for the design of NNs, one requires a high accuracy model of the network. In this brief, we introduce a high accuracy model for training *IM*-NNs where the loading effect of the memristive crossbar is utilized to determine the proper sizes for the inverters and predict their VTC coefficients. Thus, we train two small NNs and employ them to predict the VTC coefficients of the inverters based on their size and their effective load. Based on the above explanations, the contributions of this work may be summarized as follows:

Manuscript received February 20, 2021; revised March 27, 2021; accepted April 6, 2021. Date of publication April 9, 2021; date of current version September 24, 2021. This brief was recommended by Associate Editor L. A. Camunas-Mesa. *(Corresponding author: Mehdi Kamal.)*

Shaghayegh Vahdat and Mehdi Kamal are with the School of Electrical and Computer Engineering, University of Tehran, Tehran 14395-515, Iran (e-mail: vahdat_s@ut.ac.ir; mehdikamal@ut.ac.ir).

Ali Afzali-Kusha is with the School of Electrical and Computer Engineering, University of Tehran, Tehran 14395-515, Iran, and also with the School of Computer Science, Institute for Research in Fundamental Sciences, Tehran 4563-11155, Tehran (e-mail: afzali@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/TCSII.2021.3072289.

Digital Object Identifier 10.1109/TCSII.2021.3072289

- ➢ Predicting VTC coefficients of the inverters in the circuit by employing a NN-based model.
- ➢ Choosing appropriate sizes for the inverters utilized in *IM* neurons.
- ➢ Presenting a high accuracy method for the offline training of *IM*-NNs.

The remainder of this brief is organized as follows. The general structure of the *IM*-NNs as well as the PHAX and RIM training methods are briefly explained in Section II. The proposed method is presented in Section III. The simulation results are discussed in Section IV and this brief is concluded in Section V.

II. Inverter-Based Memristive NNs

In the rest of this section, first, the network model employed in PHAX [8] and RIM [14] training methods are presented. Then, their shortcomings as the motivation of the LATIM approach are studied.

A. IM-NN Model

An *IM* neuron, which circuit realization is shown in Fig. 1, can generate the weighted sum of the inputs, add it by the bias, and pass the result through the activation function. Writing the KCL equation for this neuron results in

$$net_j = \sum_{i=1}^{n} \left(x_i{}^{+} \times \frac{\sigma_{j,i}{}^{+}}{\gamma_j} + x_i{}^{-} \times \frac{\sigma_{j,i}{}^{-}}{\gamma_j}\right) + V_{DD} \times \left(\frac{\sigma b_j{}^{+} - \sigma b_j{}^{-}}{\gamma_j}\right). \quad (1)$$

where $x_i{}^{+}$ ($x_i{}^{-}$) is the i^{th} positive (negative) input and $y_j{}^{+}$ ($y_j{}^{-}$) is the positive (negative) output of j^{th} neuron. Here, $\sigma_{j,i}$ is the conductance of the memristor and $\gamma_j = \sum_{i=1}^{n} (\sigma_{j,i}{}^{+} + \sigma_{j,i}{}^{-}) + \sigma b_j{}^{+} + \sigma b_j{}^{-}$. Therefore, the positive and negative weights (biases) can be calculated as $w_{j,i}{}^{+/-} = \sigma_{j,i}^{+/-}/\gamma_j$ ($b_j{}^{+/-} = \sigma b_j^{+/-}/\gamma_j$). Note that sum of the weights and biases of each neuron is "1". Therefore, by increasing one weight, other weights and biases decrease.

In PHAX [8], the VTCs of a nominal inverter and buffer (two consecutive inverters) were extracted by HSPICE simulations and fitted to tangent hyperbolic functions (i.e., $a + b \times \tanh(x \times d - c)$ where a, b, c, d are called the VTC or activation function coefficients) to mathematically model their behavior in the training phase. The simulation results of [14], however, revealed that the VTC coefficients of the inverters change due to the loading effect of the memristive crossbars. To reduce this sensitivity, in RIM method [14], a low resistance grounded resistor was connected to the output node of each inverter. This approach leads to a better estimation of the VTC coefficients while imposing some power consumption overhead. In both of the PHAX and RIM methods, the sizes of all inverters as well as the activation functions of all neurons were assumed to be equal. Further details regarding the procedure of the PHAX training method can be found in [8].

B. Motivation

The PHAX training method is suitable for training small NNs where the number of memristors connected to the output node of each inverter or the number of hidden neurons in each layer is small. However, as shown in Fig. 2, loading effect of the memristive crossbar (assuming that the equivalent resistance of the memristive crossbar is equal to R_M) may change the VTC coefficients of the inverters which is not considered in PHAX. In addition, as shown in this figure, the VTC coefficients depend on the size of the inverter, too. Note that similar to [8], for an inverter of size s, $(W/L)_p = 2s$ and $(W/L)_n = 1.2s$. As shown in Fig. 2, the coefficients a and b are not strongly dependent on the inverter output load. Therefore, one may consider constant values for these coefficients by choosing proper sizes for the inverters based on their resistive loads. The coefficients c and d, however, exhibit a strong dependence on the inverter size and its output load value. For an accurate offline training of the network, this dependency should be considered. It is worth noting that while these plots have been extracted for inverters in 90nm technology, similar behavior has been observed in our experiments in other technology nodes.

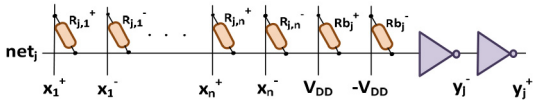

Fig. 1. An inverter-based memristive neuron [13].

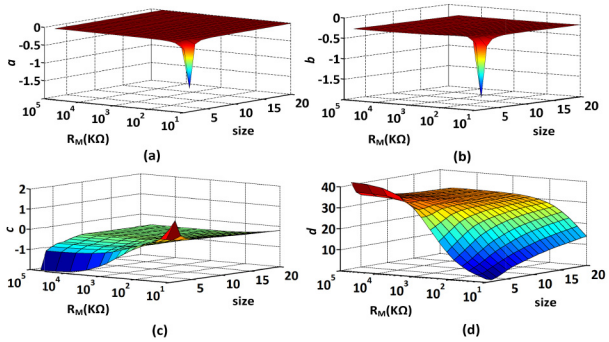

Fig. 2. The fitted coefficients a) a, b) b, c) c, and d) d of the inverter VTC versus the output loading resistance (R_M) as well as the size of the inverter.

VTC changes due to the loading effect of the memristive crossbars can cause large errors in modeling the behavior of the network. In RIM method, the dependency between the coefficients of the activation functions and the loading effect of the memristive crossbar is reduced with the overhead of higher power consumption. This motivated us to develop an approach for choosing proper sizes for the inverters (the first and second inverters of Fig. 1 denoted by negative and positive inverters in the rest of this brief) as well as accurately predicting their VTCs such that effective and reliable training can be performed for the *IM*-NNs without consuming extra power.

III. The Proposed Method

As mentioned before, the loading effect of the memristive crossbar may have large effect on the inverters VTC curves, a phenomenon which was not considered in the PHAX training process. To overcome this shortcoming, we have developed the LATIM method, which its flowchart is depicted in Fig. 3 (a). In LATIM, first, sizes of all inverters are set to some default value, e.g., 5. To model the activation functions, fitting parameters of the VTCs are determined based on the values extracted from HSPICE simulations for unloaded inverters (i.e., with no resistive loads). This initialization is the same as that of the PHAX method [8]. Next, the first epoch of the training is performed and conductance values of the memristors are determined (similar to PHAX). Subsequently, approximate loading

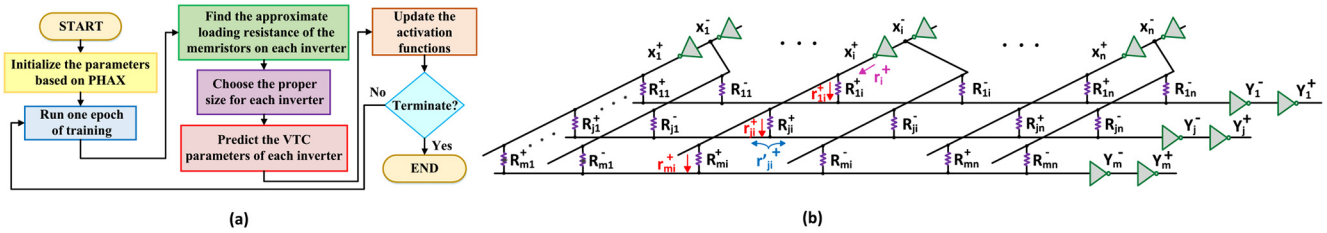

Fig. 3. (a) Flowchart of LATIM. (b) An inverter-based memristive crossbar with n inputs and m outputs.

resistances of the memristors as well as the sizes of the inverters are determined. These parameters are employed to predict the VTC of each inverter. As was shown in Fig. 2, the dependency between the VTC coefficients and the size and resistive load of the inverter is complex. Therefore, we have opted to use pre-trained NNs to estimate the VTC coefficients of an inverter based on its output resistive load and the size of the inverter. Next, the activation functions of the model are updated and the termination condition is checked. As long as the termination condition is not satisfied, next epoch of the training is performed using the updated activation functions. In this brief, the number of training epochs (i.e., 100 epochs) was considered as the termination condition. In the rest of this section, first, the procedure of approximating the loading resistance of the memristive crossbar is explained. Then, the proper inverter sizes are chosen based on the approximated resistive loads. Finally, an approach for predicting the VTC coefficients is presented.

A. Equivalent Resistance of the Memristive Crossbar

Consider a memristive crossbar with n inputs and m outputs as shown in Fig. 3 (b). The equivalent output load of the i^{th} positive inverter, which is denoted by r_i^+, may be calculated from

$$\frac{1}{r_i^+} = \sum_{j=1}^{m} \frac{1}{r_{ji}^+} = \sum_{j=1}^{m} \frac{1}{R_{ji}^+ + r'^+_{ji}} \tag{2}$$

where

$$\frac{1}{r'^+_{ji}} \approx \frac{1}{R_{ji}^-} + \sum_{p \neq i} \left(\frac{1}{R_{jp}^+} + \frac{1}{R_{jp}^-} \right). \tag{3}$$

Because r'^+_{ji} is almost equal to the equivalent resistance of $(2n-1)$ parallel resistors, its value is assumed to be small compared to R_{ji}^+. Therefore, $r_{ji}^+ \approx R_{ji}^+$ which leads to

$$\frac{1}{r_i^+} \approx \sum_{j=1}^{m} \frac{1}{R_{ji}^+}. \tag{4}$$

Similarly, the equivalent load resistance of the negative inverters (i.e., r_i^-) can be found.

B. Inverter Size Selection

As mentioned before, by increasing one weight of an *IM* neuron, the other weights and biases decrease. Therefore, it would be hard for an *IM* neuron to generate large *net* values as it requires several large inputs while their corresponding weights are also large (see (1)). The situation becomes worse when the allowed input swing of the neuron reduces. Noting the fact that the outputs of the negative and positive inverters are utilized as the inputs of the neurons of the next layer, it is not desired to have inverters with small output swings. Since the output swing of the inverter is equal to $2b$, the size of the inverters should be chosen such that their coefficient b does not change considerably due to the output load of the inverters.

By looking at Fig. 2 (a) and (b), for an inverter of size s, a and b may be considered almost constant for the R_M values larger than a minimum resistance which is denoted by RL_s in this work. To have a high modeling accuracy when setting a and b to constant values, the output resistance should be larger than RL_s. Now, we are to find a mathematical equation for finding the proper size of the inverter based on its resistive load. It is worth noting that the VTC of an inverter of size s and resistive load of R_M/s is nearly the same as that of s minimum sized inverters (inverters of size 1) and resistive load of R_M connected in parallel. Therefore, we may approximate RL_s as RL_1/s where RL_1 is defined as the resistance at which the approximation errors of coefficients a and b are small values (*i.e.*, 1% and 2% of $2V_{DD}$ in this work where V_{DD} $(-V_{DD})$ is the positive (negative) supply voltage of the inverter.)

Next, we estimate the output load of the inverter, which can be approximated using (4). This resistance should not be lower than RL_s providing a constraint on the size of the inverter. As an example, for the i^{th} negative inverter with the size of s_i^- and approximate output resistance equal to r_i^-, the following equation must be satisfied

$$r_i^- \geq RL_{s_i^-} = \frac{RL_1}{s_i^-}, \tag{5}$$

which results in $s_i^- \geq RL_1/r_i^-$. To keep the power and area as low as possible, the size of the inverter may be chosen as

$$s_i^- = \left\lceil \frac{RL_1}{r_i^-} \right\rceil. \tag{6}$$

C. Modeling VTC of the Inverter

To model the VTC of each inverter, we have to find the value of the coefficients a, b, c, and d. Because sizes of the inverters are chosen based on (6), the output resistance is always higher than RL_s. Therefore, constant values can be considered for coefficients a and b. Modeling coefficients c and d is more complicated. These coefficients strongly depend on the output load and the size of the inverter (see Fig. 2 (c) and (d)). To model this dependency, we train two NNs, called NN_c and NN_d, which can predict the value of the coefficients c and d, respectively, based on the output load and the inverter size. It is worth noting that training of NN_c and NN_d is performed once for a given CMOS technology. Therefore, the training procedure of NN_c and NN_d does not induce overhead for training of *IM*-NNs.

IV. Results and Discussion

In this section, the training accuracy of the proposed method is compared with those of PHAX and RIM. The comparative study includes both of system and circuit level simulations.

A. Framework of the Comparative Study

For the considered approaches, training phases were performed by Python (TensorFlow) where the Adam optimizer was employed as the training optimizer. To compare the modeling accuracies as well as the electrical parameters of the networks (i.e., delay, power, and energy), the trained networks for IRIS, BCW, MNIST, and Fashion MNIST datasets were implemented in HSPICE using a 90nm technology and the memristor model presented in [18]. The numbers of inputs, outputs, and hidden neurons of each network are reported as the network configuration in Table I. The employed memristor has maximum and minimum conductance values of $8\mu\mho$ and $0.12\mu\mho$, respectively [12] and the voltage sources of the network were set to ± 0.25V. In addition, NN_c and NN_d were trained using 2,500 training samples in Python (Keras) where mean squared error was considered as the cost function. These networks have three hidden layers with 10, 8, and 8 neurons, respectively.

B. Training Accuracy

The accuracy of the networks trained by PHAX, RIM, and LATIM methods are reported in Table I. Results were generated by using Python simulations after 1,000 epochs of training for IRIS and BCW datasets with 120 (30) and 546 (136) train (test) samples which are implemented using small NNs. In addition, 100 epochs of training were performed for the MNIST and Fashion MNIST benchmarks with 60,000 (10,000) train (test) samples which require larger NNs. The average size of the inverters in each layer is also reported in Table I. As the results show, LATIM and PHAX methods have almost similar accuracy values for the test set which are larger than that of the RIM method in all of the considered datasets except for BCW.

C. Modeling Accuracy and Electrical Parameters

In this subsection, the modeling accuracies as well as electrical parameters of NNs trained by the considered training methods are compared. The modeling accuracy was defined based on the closeness of the results generated by Python to those obtained from the HSPICE simulations. The Python model is based on the simple network model used in PHAX, RIM, or LATIM while the HSPICE model includes complex equations, more accurately modeling the real behavior of the network. We selected 2,000 samples from the training and test datasets and measured the classification accuracy (denoted by *Acc*) and average error of NN output voltage prediction (denoted by *OE*) for each network which are reported in Table II. *OE* is measured by finding the average absolute difference between the results of Python and HSPICE simulations which represents how well the simple mathematical equations of PHAX, RIM, and LATIM can model the real complex behavior of the network. As the results reveal, when the network size is small (*i.e.*, for the IRIS and BCW datasets), *Acc* of the networks trained by PHAX or LATIM is more than 96.3% whereas this parameter is more than 95.8% for the RIM method. Thus, all the training methods work well for small NNs. When the network size increases (*i.e.*, for the MNIST and Fashion MNIST datasets), the performance of PHAX deteriorates whereas RIM and LATIM still have acceptable accuracies. In addition, in the proposed LATIM method, *OE* is considerably lower than those of PHAX and RIM methods when the network size is large. As an example, *OE* of LATIM is, on average, 14× (6×) and 29× (4×) smaller than those of PHAX (RIM) for MNIST and Fashion MNIST benchmarks. In addition, due to proper sizing of the inverters, the power consumption of the networks trained by LATIM is reduced, on average, by 38% and 75% compared to those of RIM and PHAX methods, respectively.

TABLE I
Comparison of the Accuracies of the Networks Trained by PHAX, RIM, and LATIM Approaches Extracted by Python Simulations As Well As the Average Size of the Inverters of Each Layer

Dataset	*Network Configuration*	*Training method*	*INV Average Size*			*Accuracy (%)*	
			L1	*L2*	*L3*	*Train*	*Test*
IRIS	4 → 5 → 3	PHAX	5	5	-	99.2	100
		RIM	5	5	-	95.8	96.7
		LATIM	1.4	1	-	99.2	100
BCW	10 → 5 → 2	PHAX	5	5	-	99.3	95.6
		RIM	5	5	-	97.6	96.3
		LATIM	1.5	1	-	98.3	95.6
MNIST	784 → 30 → 40 → 10	PHAX	5	5	5	97.9	95.7
		RIM	5	5	5	90.7	91.1
		LATIM	5	1.5	1	96.0	95.2
Fashion MNIST	784 → 30 → 40 → 10	PHAX	5	5	5	92.3	87.0
		RIM	5	5	5	85.0	83.4
		LATIM	8.2	1.3	1	88.9	86.2

TABLE II
Comparison of the Accuracy and Electrical Parameters of *IM*-NNs Trained by PHAX, RIM, and LATIM Methods Extracted by HSPICE Simulations

Dataset	*Training Method*	*Train Samples*		*Test Samples*				
		Acc (%)	*OE* (mV)	*Acc* (%)	*OE* (mV)	*Delay* (ns)	*Power* (μW)	*Energy* (fJ)
IRIS	PHAX	97.5	7	96.7	7	4.0	5.8	23.1
	RIM	95.8	8	96.7	8	1.5	30.5	45.8
	LATIM	98.3	8	100	7	3.0	3.7	11.2
BCW	PHAX	98.5	4	96.3	4	3.1	6.2	19.3
	RIM	97.4	3	96.3	3	1.2	29.3	35.1
	LATIM	98.2	7	96.3	8	1.8	2.8	5.1
MNIST	PHAX	28.0	106	31.7	107	2.1	164.7	345.9
	RIM	89.8	45	88.9	46	1.6	315.7	505.1
	LATIM	93.6	7	92.3	8	2.3	77.8	396.5
Fashion MNIST	PHAX	26.6	263	24.5	258	2.1	174.6	366.7
	RIM	83.0	39	82.1	39	1.4	304.7	426.6
	LATIM	86.1	9	83.7	9	1.9	158.4	306.0

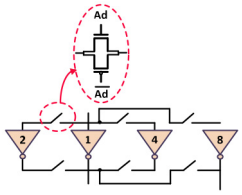

Fig. 4. Proposed implementation of an inverter with adjustable size.

The LATIM approach can also be employed for *IM*-NNs where the size of the inverters is fixed (*e.g.*, using an *IM-NN* hardware accelerator for different NNs). In this case, the inverter size selection step is skipped and all of the VTC coefficients can be determined using pre-trained NNs while the equivalent output resistance of the inverters are found based on (4). However, to have the capability of choosing proper sizes for the inverters to reduce the power consumption, we propose to employ inverters with adjustable sizes as the activation function generators of the *IM-NN* accelerator. The internal structure of the inverters is depicted in Fig. 4. In this structure, inverters with different sizes can be parallelized using transmission gates.

When implementing the network on silicon, (process) variations would affect the performance of the network, and hence, one should resort to variation mitigation approaches for tackling the variation in the hardware implementation of the NN and/or those devised for the training phase to reduce the impacts of the variation on the performance of the realized NN. Since LATIM can predict the voltages of different nodes of the NN with high accuracy, different variation mitigation approaches can be added to the proposed network model to reduce the severe effects of variations on the NN outputs.

To investigate the impact of variations on the network accuracy, Gaussian random variations were applied to the memristor conductance [6], [19] as well as the threshold voltage, width, and length of the transistors [8], [11], [13], [14]. Let us denote the variation scenarios of 20VAR_M with the variability (σ/μ) of 20% for the memristors conductance values as well as 5VAR_T for representing the variability of 5% for the transistor parameters and perform 1,000 simulations. The simulations provided the mean network accuracies of 93.6% and 98.9% under 20VAR_M condition and 97% and 98.8% under 5VAR_T condition for IRIS dataset, for the networks trained by PHAX and LATIM approaches, respectively. The corresponding accuracies in the case of the BCW dataset were 95.6%, 96.2%, 95.7%, and 96.2%, respectively. These results indicate that the networks trained by LATIM had higher accuracies compared to those trained by PHAX due to the higher modeling accuracy of LATIM.

V. Conclusion

In this brief, we presented a technique for improving the training accuracy as well as reducing power consumption of *IM*-NNs. The effective load of the memristive crossbar was approximated and used in predicting the VTC of the inverter with high accuracy. The impact of the inverter size on the VTC also was considered. In the technique, which was called LATIM, two NNs were trained to model the inverter VTC coefficients based on the inverter size as well as its output load. For the MNIST benchmark, the proposed method improved the network accuracy from 31.7% to 92.3% compared to PHAX while also reduced the power consumption by 75% compared to RIM, which are recently proposed offline training methods for *IM*-NNs.

References

[1] A. Castiglione, P. Vijayakumar, M. Nappi, S. Sadiq, and M. Umer, "COVID-19: Automatic detection of the novel coronavirus disease from CT images using an optimized convolutional neural network," *IEEE Trans. Ind. Informat.*, early access, Feb. 5, 2021, doi: 10.1109/TII.2021.3057524.

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

[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, M. Huizi, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015. [Online]. Available: arXiv:1510.00149.

[5] 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.

[6] P. Yao *et al.*, "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, pp. 641–646, Jan. 2020.

[7] Y. Zhang, X. Wang, and E. G. Friedman, "Memristor-based circuit design for multilayer neural networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 2, pp. 677–686, Feb. 2018.

[8] 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.

[9] R. Hasan, T. M. Taha, and C. Yakopcic, "A fast training method for memristor crossbar based multi-layer neural networks," *Analog Integr. Circuits Signal Process.*, vol. 93, no. 3, pp. 443–454, 2017.

[10] 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.

[11] 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 Autom. Test Europe Conf. Exhibition (DATE)*, Lausanne, Switzerland, 2017, pp. 440–445.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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. Des. Integr. Circuits Syst.*, vol. 32, no. 8, pp. 1201–1214, Aug. 2013.

[19] 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.