Effect of Asymmetric Nonlinearity Dynamics in RRAMs on Spiking Neural Network Performance

Mohammed E. Fouda[1], E. Neftci[2], A. Eltawil[1,3], and F. Kurdahi[1]
[1]Electrical Engineering and Computer Science Dept., University of California, Irvine, CA 92697-2625 USA.
[2]Cognitive Sciences Dept., University of California, Irvine, CA 92697-2625 USA.
[3]King Abdullah University of Science and Technology, Thuwal, Saudi Arabia.
email:{foudam@uci.edu}

***Abstract*—Crossbar-based Resistive Random Access Memory (RRAM) array is a promising candidate for fast and efficient implementation of the vector-matrix multiplication, an essential step in a wide variety of workloads. However, several RRAM devices, demonstrating promising synaptic behaviors, are characterized by nonlinear and asymmetric update dynamics, which is a major obstacle for large-scale deployment in neural networks, especially for online learning tasks. In this work, we first introduce a memristive Spiking Neural Network (SNN) with local learning. Then, we study the effect of this asymmetric and nonlinear behavior on the spiking neural network performance and propose a method to overcome the performance degradation without extra nonlinearity cancellation hardware and read cycles. The performance of the proposed method approaches the baseline performance with $1 \sim 2\%$ drop in recognition accuracy.**

***Index Terms*—Spiking Neural Networks, Online Learning, RRAMs, Memristor, Nonidealities, Asymmetric Nonlinearity.**

I. Introduction

The last decade witnessed significant progress in developing neuromorphic hardware that can solve large scale tasks much faster and with online learning capabilities unlike Von-Neumann based processor architectures [1]–[4]. Von-Neumann architectures suffer from the memory-wall bottleneck, especially when performing neuromorphic tasks where massive data blocks are transferred from the memory to processing nodes [5]. Recently developed solid-state devices have shown non-volatile storage capabilities such as RRAM (i.e memristors), phase-change memory, and spin-transfer torque-RAM [6]. A major advantage of these devices is the ability to be assembled as a crossbar array that enables the Vector-Matrix Multiplication (VMM) operation to be completed in a single step [7]. This is unlike other hardware solutions that require $N \times M$ steps, where N and M are the dimensions of the weight matrix.

Several RRAM devices demonstrating promising synaptic behaviors are characterized by nonlinear and asymmetric update dynamics, which is a major obstacle for large-scale deployment in neural networks [8], especially for learning tasks. Applying the vanilla backpropagation algorithms without taking into consideration the device non-idealities does not guarantee the convergence of the network. Thus, a closed-form model for the device non-linearity must be derived based on the device dynamics and added to the neural network training algorithm to guarantee convergence to the optimal point (minimal error).

Recently, different supervised learning techniques were proposed which are more energy and data-efficient such as random backpropagation, direct and indirect feedback alignment and surrogate gradient learning [9]–[11]. These techniques have not been utilized to enable more efficient hardware learning yet. The existing emerging hardware accelerators rely on stochastic gradient propagation [12]–[15]. In this work, we discuss the local learning on memristive Spiking Neural Network for efficient write energy solutions.

This paper is organized as follows: Section II introduces the memristive SNN architecture with local learning. Then, Section III discusses the asymmetric nonlinearity update dynamics of RRAM devices. While in Section IV, we discuss the training with this nonideality and introduce a linearized model to get rid of the extra hardware needed for nonlinearity cancellation. Besides, results and discussion are shown. Finally, the conclusion is given.

II. Spiking Neural Network with Local Learning

A. Spiking Neural Network Model

The SNN model consists of spiking integrate-and-fire neurons without temperal dynamics to have a sparse and simple communications between the SNN layers to overcome the need for analog to digital and digital to analog converters. The discrete-time model can be formulated as

$$U_i^l[n] = \sum_j W_{ij}^l S_j^{l-1}[n] \tag{1a}$$

$$S_i^l[n] = \Theta(U_i^l[n]) \tag{1b}$$

where j and i are the indices of the input and the output signals, respectively, $U_i^l[n]$ is the membrane potential of the neuron i at layer l at time step n, W^l is the synaptic weight matrix between layer $l-1$ and l, and S_i^l is the binary output of this neuron i. The step function Θ is the spiking mechanism, *i.e.* $S_i^l[n] = 1$ for $U_i^l[n] > 0$. It is worth to highlight that in this SNN, the reset mechanism is omitted for simple implementation without loss in the performance [11]. Thus, it can be seen as spiking version of the artificial neural networks.

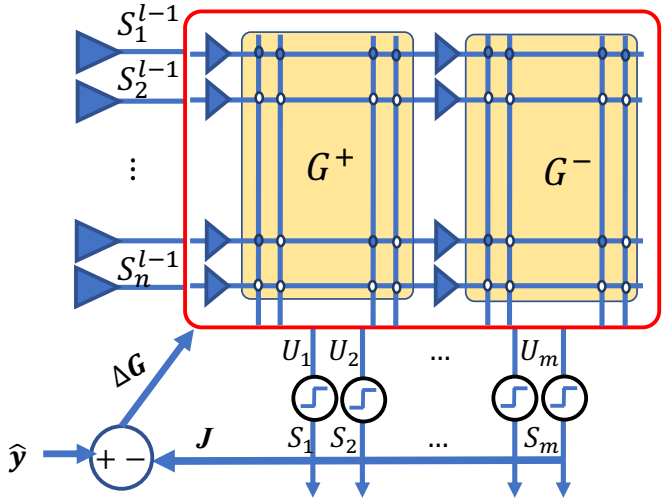


Fig. 1: Single memristive spiking neural network layer with online local learning cababilities.

B. Local Learning with Crossbar Arrays

RRAMs are used to implement the synaptic weights. In practice, two RRAMs are needed to realize each weight to have positive and negative values. In this work, we use a balanced realization where two RRAMs are used for each weight such that $w = G^+ - G^-$. If the G^+ is greater/less than G^-, it represents positive/negative weight, respectively. Thus, the memristive SNN, shown in Fig. 1, can be written as

$$U_i^l[n] = \sum_j \left(G_{ij}^{+l} - G_{ij}^{-l}\right) S_j^{l-1}[n] \tag{2}$$

Assuming a local cost function $\mathcal{L}$ for each layer, the gradients with respect to each device's conductance are formulated as three factors

$$\frac{\partial \mathcal{L}}{\partial G_{ij}^{\pm l}} = \frac{\partial \mathcal{L}}{\partial S_i^l} \frac{\partial S_i^l}{\partial U_i^l} \frac{\partial U_i^l}{\partial G_{ij}^{\pm l}} \tag{3}$$

The term $\partial U_i^l / \partial G_{ij}^{\pm l}$ is equal to $\pm S_j^l[n]$ in the integrate and fire neuron. The middle factor, $\partial S_i^l / \partial U_i^l$ is the derivative of the step function Θ which is not differentiable. In practice, the step function is replaced by a piecewise linear function [11]. The derivative of the piecewise linear function is the box function $\partial S_i^l / \partial U_i^l = 1$ if $u_- < U_i < u_+$ and 0 otherwise. The term $\partial \mathcal{L} / \partial S_i^l$ describes the change in the spiking state affects in the loss is called the local error denoted as δ_i^l and computed by gradient backpropagation. The positive and negative conductances become

$$\Delta G_{ij}^{\pm l} = -\eta \frac{\partial \mathcal{L}}{\partial G_{ij}^{\pm l}} = \mp \eta \delta_i^l S_j^{l-1}, \text{ if } u_- < U_i < u_+, \tag{4}$$

where η is the learning rate.

In the local learning, the local losses are local classifiers (using same output labels) [16]. Local losses may outperform the regular backpropagation in some tasks [17]. The local classifier is a random linear transformation J that reduce the dimension of each layer output S^l to the number of classes. This comparison with class labels can be performed using either mean-squared error loss or cross-entropy or any other loss function. For example, mean-squared error loss is

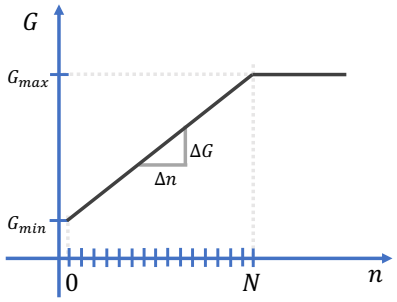


Fig. 2: RRAM's conductance change relation with number of pulses .

$$\mathcal{L}^l = \frac{1}{C} \sum_{k=1}^{C} \mathcal{L}_k^l = \| \sum_{i=1}^{N^l} J_{ki}^l S_i^l - \hat{y}_k \|_2, \tag{5}$$

where C is the number of classes, N^l is the layer width and J_{ki}^l is an element in a random, fixed matrix, and $\hat{y}_k$ are one-hot encoded labels. Thus, $\delta_i^l = \sum_{k=1}^{C} J_{ik}^l err_k^l$ where $err_k^l = \sum_{i=1} J_{ki}^l S_i^l - \hat{y}_k$. For simplicity, it can be written in matrix form as $\boldsymbol{\delta}^l = \mathbf{J}^{l^T}(\mathbf{J}^l \mathbf{S}^l - \hat{\mathbf{y}})$.

The common practice to potentiate or depress the conductance value is through applying positive or negative voltage pulses, respectively. Ideally, the conductance change should have a linear relation with the update pulses, as follows:

$$G = G_{min} + \frac{G_{max} - G_{min}}{N} n, \quad G \in [G_{min}, G_{max}] \tag{6}$$

where G_{max} and G_{min} are the maximum and minimum achievable conductances, respectively, n is the pulse index, and N is the total number of pulses to fully potentiate or depress the device. According to this modeling equation, the required number of pulses to cause a certain change in conductance ΔG can be written as

$$\Delta n = \frac{N}{G_{max} - G_{min}} \Delta G \tag{7}$$

which should be rounded to the nearest integer. The previous equation can be rewritten in normalized form as $\Delta n = \frac{N}{1-K} \Delta w$ where $K = G_{min}/G_{max}$ and $\Delta w = \Delta G / G_{max}$.

C. Experimental Setup

A three-layer fully connected spiking network is implemented using Pytorch with local learning using random local classifiers. The layer connections are as follows: 784 → 500 → 500 → 250. In this work, we used Adam optimizer with a cross-entropy loss function. The back-propagation calculation of each layer is calculated using autograd functionality [18].

Figure 3 shows the local test accuracy of each layer of a three-layer spiking neural network with limited weight value $w \in [K-1, 1-K]$ and with stochastically rounded updates.

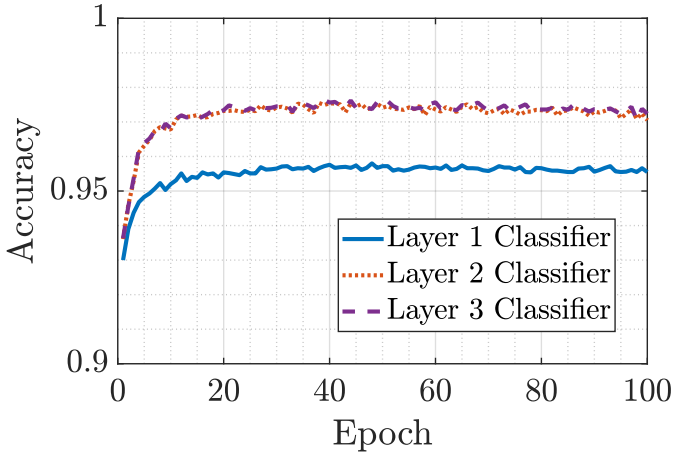


Fig. 3: Ideal validation with limited weight range and with stochastic rounding at $K = 1/20$ and $N = 100$.

The stochastic rounding is common practice for limited numerical precision calculation where it is an unbiased rounding and has zero expected rounding error [19]. Without the stochastic rounding, the network fails to learn.

III. Asymmetric nonlinearity Update Dynamics

Most of the potentiation and depression behaviors have exponential dynamics versus the programming time or the number of programming pulses. In practice, the depression curve has higher slope compared to the potentiation curve, which causes the asymmetric programming. Updating the RRAM conductance is commonly performed through positive negative programming pulses for Long Term Potentiation (LTP)/ Long Term Depression (LTD) with pulse width T_P/T_D and constant programming voltage V_{pP}/V_{pD}, respectively. The asymmetric non-linearity of the RRAM's conductance update can be fitted to the following model

$$G(t) = \begin{cases} G_{max} - \beta_P e^{-\alpha_P n} & LTP \\ G_{min} + \beta_D e^{-\alpha_D n} & LTD \end{cases} \tag{8}$$

where $\alpha_P, \alpha_D, \beta_P$ and β_D are fitting coefficients and function of the programming pulse voltages and periods.

One way to quantify the device potentiation and depression asymmetry and linearity is the asymmetric nonlinearity factor [20]. The effect of these factors is reflected in the coefficients $\alpha_P, \alpha_D, \beta_P$ and β_D which are used for the training. The potentiation asymmetric non-linearity (PANL) factor and depression asymmetric non-linearity (DANL) are defined as $PANL = G_{LTP}(N/2)/\Delta G - 0.5$ and $DANL = 0.5 - G_{LTD}(N/2)/\Delta G$, respectively, where N is the total number of pulses to fully potentiate the device. $PANL$ and $DANL$ are between $[0, 0.5]$. The sum of both potentiation and depression asymmetric non-linearities represents the total asymmetric non-linearity (ANL) which can be written as follows for the proposed RRAM model:

$$ANL = 1 - \frac{\beta_P e^{-0.5\alpha_P N} + \beta_D e^{-0.5\alpha_D N}}{\Delta G}. \tag{9}$$

IV. Asymmetric Nonlinearity Dynamics in TiOx Device

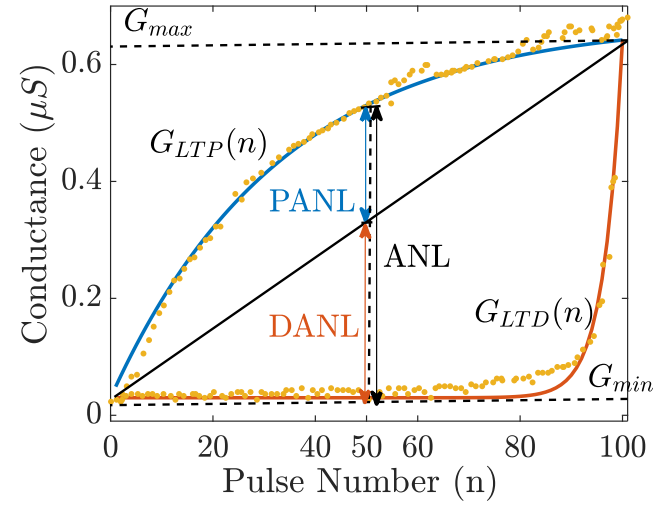
An example of a synaptic device is a non-filamentary (oxide switching) TiO_x based RRAM with a precision measured to 6 bits [21]. The $Mo/TiO_x/TiN$ device was fabricated based on a redox reaction at Mo/Tio_x interface which forms conducting MoO_x. This type of interface based switching device exhibits good switching variability across the entire wafer and guarantees reproducibility [21]. The asymmetric nonlinear behavior of this device is shown in Fig. 4a on top of the measured data of the RRAM's potentiation and depression. In our previous work [22], [23], we curve-fitted the measured data to a closed-form mathematical model to be included in a machine learning framework for realistic and accurate training. In this work, programming with $\pm 3V$ is considered since it has the widest switching range. The model parameters are $G_{max} = 674nS, \alpha_P = 0.03058, \beta_P = 626.8nS$, $G_{min} = 32.95nS, \alpha_D = 0.3534$, AN $\beta_D = 921.9nS$. Figure 4a shows the curve fitted model on the top of the measured conductance (dotted curve) for both potentiation and depression scenarios. This device has $PANL = 0.32$ and $DANL = 0.45$ with $ANL = 0.77$.

In our model, we also considered the device variations which occur due to the randomness in the device and it also differs from device to another which is called device to device variations. These variations can be mitigated with Error-Correcting (EC) techniques such as write-verify techniques where the written value is read to verify the value and corrected until the desired value is obtained [24]. Such EC techniques increase the training time and total write energy which is not desirable in online learning. The training should take care of these variations. Figure 4b shows simulated conductance variations of multiple devices during the potentiation and depression cycles with $\pm 3V$ programming pulses. The model parameters are sampled from Gaussian sources with 25% tolerance (Variance/mean) for α, and 1% and 5% tolerances for the maximum and minimum conductances, respectively.

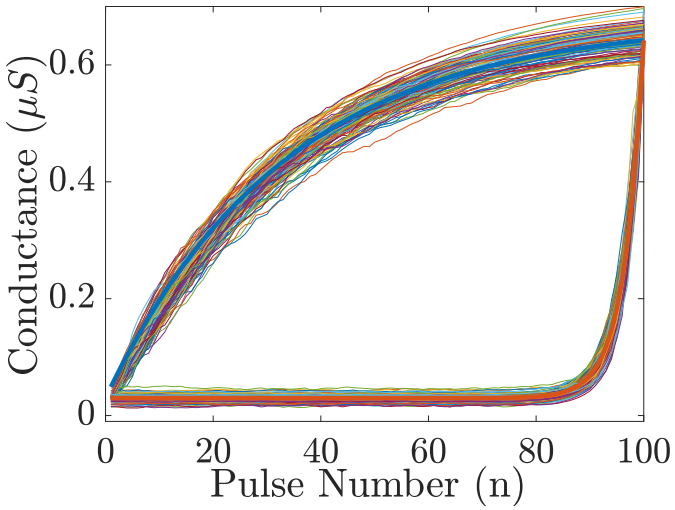
The effect of the variation in the parameter β is considered inside the variations of α. β is modeled as a lognormal variable to have a monotonic increasing or decreasing conductance update. Thus, the second term of the conductance update has a logarithmic Gaussian variable, which is e^z, multiplied by $e^{\alpha n}$ where z and α are Gaussian variables. Since the sum of two Gaussian random variables is a Gaussian random variable, the variation of β and α can be included in either one of them.

A. RRAM Updates during Training

In [22], we proposed a method to have the resistive devices behave exactly like the learning rule where the change in each weight must be proportional to the change in the RRAM's conductance, $\Delta \mathbf{G} \propto \Delta \mathbf{w}$. To achieve this, the asymmetric nonlinear behavior of potentiation and depression are included in the learning rule. We first calculate the change in the weights for both potentiation and depression cases taking into effect the asymmetric nonlinearity of the RRAM model. In general, the



(a)



(b)

Fig. 4: Non-idealities of the RRAM:(a) asymmetric nonlinear weight update (b) device Variations

change in the LTP's and LTD's conductance due to applying Δn is

$$\begin{aligned} \Delta G_{LTP} &= (G_{max} - G)\left(1 - e^{-\alpha_P \Delta n}\right), \text{ and} \\ \Delta G_{LTD} &= (G - G_{min})(e^{-\alpha_D \Delta n} - 1), \end{aligned} \tag{10}$$

respectively where G is the previous conductance. Clearly, the relation between the rate of change in conductance and Δn is an injective function. Thus, the number of pulses to cause ΔG_{LTP} and ΔG_{LTD} are

$$\Delta n_P = -\frac{1}{\alpha_P} \ln\left(1 - \frac{\Delta G_{LTP}}{G_{max} - G(n)}\right), \text{ and} \tag{11}$$

$$\Delta n_D = -\frac{1}{\alpha_D} \ln\left(\frac{\Delta G_{LTD}}{G(n) - G_{min}} + 1\right), \tag{12}$$

respectively. After learning, $\Delta \mathbf{G}$ goes to $\mathbf{0}$. As a result, $\Delta \mathbf{n}$ goes to zero as well.

The update equations ((11) and (12)) require the knowledge of the weight value, meaning a read operation is needed to calculate the required number of pulses to update. This nonlinear cancellation can be realized either in analog domain using logarithmic amplifier circuits or in digital domain using lookup tables. However, they are area and latency expensive.

One way to overcome the need for extra hardware for the nonlinearity cancellation is by linearizing the updated model as discussed in [22]. Both equations 11 and 12 can be linearity as $\ln(1-x) \approx -x(1+0.5x) \approx -x$ and $\ln(1+x) \approx x(1-0.5x) \approx x$ for $x \ll 1$. This leads to that the potentiation and depression updates are given by

$$\Delta n_P = \frac{1}{\alpha_P} \frac{\Delta G_{LTP}}{G_{max} - G(n)}, \ \Delta G_{LTP} \ll G_{max} - G(n), \text{ and} \tag{13a}$$

$$\Delta n_D = \frac{1}{\alpha_D} \frac{|\Delta G_{LTD}|}{G(n) - G_{min}}, \ |\Delta G_{LTD}| \ll G(n) - G_{min}, \tag{13b}$$

respectively. As previously discussed, it is needed to make $\Delta \mathbf{G} \propto \Delta \mathbf{w}$, these two conditions can be satisfied by using smaller learning rates. But, the weight read cycle is still needed even in this linearized model. The rounding of the integer number of update pulses does not guarantee the convergence to the optimal point since the rounding is done after nonlinearity cancellation and gradient calculations. The device dynamics should be included directly to the learning rule which is out of the scope of this paper.

B. Ternary Update Method

Another practical method to update the weight is that we use only directional updates where the update pulses are proportional or equal to $\Delta w = \Delta G / G_{max}$. Since ΔG is always less than G_{max}, then only positive or negative or zero update pulse is applied to each conductance which we can refer to as Ternary Update. This update method may cause slow learning. To have faster learning. scaling factors can be added to the update rule which can be generally written as

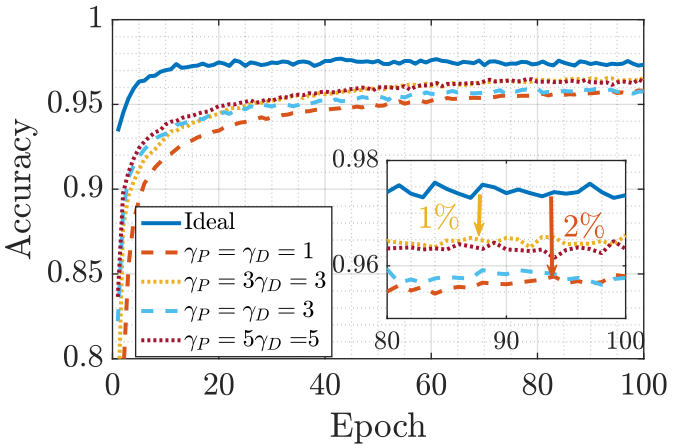
$$\Delta n_{P,D} = \gamma_{P,D} SR(\Delta w), \tag{14}$$

where $\gamma_{P,D}$ are scaling factors and SR is a stochastic rounding function. γ_P and γ_D can be equal or different based on the asymmetry in the potentiation and depression of the used device. This update model can be easily implemented without the need for any kind of nonlinearity cancellation (i.e. lookup tables) and read cycles.

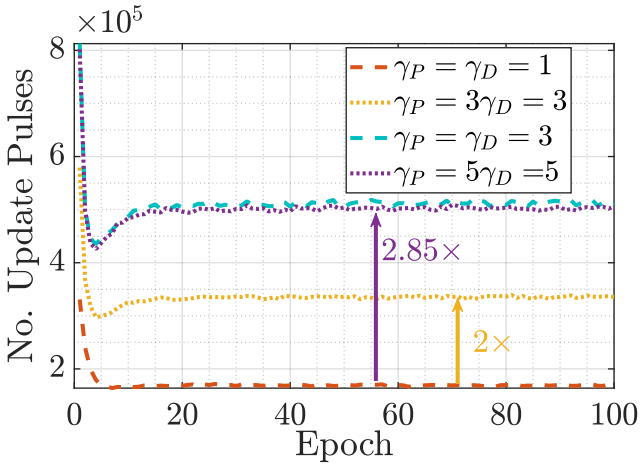
C. Results

Figure 5a shows a comparison between the baseline performance and the proposed ternary update rule for different γ values. The results show that the network can achieve almost the baseline with a $2\%p$ drop for equal γ values. In order to accelerate the training, we considered higher γ values. It is clear that when γ increased $3\times$ or $5\times$ the training is speedup by the same ratio. We can see that we asymmetric γ values that the network is able to achieve an extra $1\%p$ in the accuracy compared to symmetric or equal γ values.

Figure 5b shows the number of update pulses performed in each epoch. At the beginning of the training, the number of update pulses is high and reduces during the progress of the training. The number of update pulses settles after 10 epochs



(a)



(b)

Fig. 5: (a) Test accuracy of the outer classifier for different scaling factors of ternary update rule and (b) total number of positive and negative update pulses for each epoch.

where fine tuning of the network. Clearly, with increasing the scale factor, γ, the number of update pulses increases $2\times$ and $2.85\times$ for $\gamma_P + \gamma_P = 4$ and 6, respectively. Thus, the write energy is expected to increase with the same ratios. Clearly, the asymmetric updates is needed to achieve a better performance and faster learning.

V. Conclusion

In this work, we proposed an effective method to train a memristive spiking neural network with local learning containing RRAMs with asymmetric nonlinearity update dynamics. Stochastic rounding is necessary to have a successful training. we showed with the experiments that the ternary update method was very efficient and almost achieve baseline accuracy. Other nonidealites such as IR drop (i.e sneak path problem), limited endurance will be investigated in the future work.

References

[1] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, P. Joshi, A. Lines, A. Wild, and H. Wang, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. PP, no. 99, pp. 1–1, 2018.

[2] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[3] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses," *Frontiers in neuroscience*, vol. 9, 2015.

[4] S. Friedmann, J. Schemmel, A. Grübl, A. Hartel, M. Hock, and K. Meier, "Demonstrating hybrid learning in a flexible neuromorphic hardware system," *IEEE transactions on biomedical circuits and systems*, vol. 11, no. 1, pp. 128–142, 2017.

[5] G. Cauwenberghs, "Reverse engineering the cognitive brain," *Proceedings of the national academy of sciences*, vol. 110, no. 39, pp. 15 512–15 513, 2013.

[6] Q. Xia and J. J. Yang, "Memristive crossbar arrays for brain-inspired computing," *Nature materials*, vol. 18, no. 4, pp. 309–323, 2019.

[7] ——, "Memristive crossbar arrays for brain-inspired computing," *Nature materials*, vol. 18, no. 4, pp. 309–323, 2019.

[8] S. Yu, "Neuro-inspired computing with emerging nonvolatile memorys," *Proceedings of the IEEE*, vol. 106, no. 2, pp. 260–285, 2018.

[9] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature Communications*, vol. 7, 2016.

[10] E. Neftci, C. Augustine, S. Paul, and G. Detorakis, "Event-driven random back-propagation: Enabling neuromorphic deep learning machines," *Frontiers in Neuroscience*, vol. 11, p. 324, Jun 2017.

[11] E. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks," *Signal Processing Magazine, IEEE*, Dec 2019, (accepted).

[12] C. Li and et al., "Analogue signal and image processing with large memristor crossbars," *Nature Electronics*, vol. 1, no. 1, p. 52, 2018.

[13] M. Prezioso and et al., "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, p. 61, 2015.

[14] C. Li and et al., "Efficient and self-adaptive in-situ learning in multilayer memristor neural networks," *Nature Communications*, p. 2385, 2018.

[15] S. Ambrogio, P. Narayanan, H. Tsai, R. M. Shelby, I. Boybat, C. di Nolfo, S. Sidler, M. Giordano, M. Bodini, N. C. Farinha *et al.*, "Equivalent-accuracy accelerated neural-network training using analogue memory," *Nature*, vol. 558, no. 7708, p. 60, 2018.

[16] H. Mostafa, V. Ramesh, and G. Cauwenberghs, "Deep supervised learning using local errors," *arXiv preprint arXiv:1711.06756*, 2017.

[17] A. Nøkland and L. H. Eidnes, "Training neural networks with local error signals," *arXiv preprint arXiv:1901.06656*, 2019.

[18] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[19] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *International Conference on Machine Learning*, 2015, pp. 1737–1746.

[20] J. Woo and S. Yu, "Resistive memory-based analog synapse: The pursuit for linear and symmetric weight update," *IEEE Nanotechnology Magazine*, vol. 12, no. 3, pp. 36–44, 2018.

[21] J. Park and et al., "Tio x-based rram synapse with 64-levels of conductance and symmetric conductance change by adopting a hybrid pulse scheme for neuromorphic computing," *IEEE Electron Device Letters*, vol. 37, no. 12, pp. 1559–1562, 2016.

[22] M. E. Fouda, E. Neftci, A. Eltawil, and F. Kurdahi, "Independent component analysis using rrams," *IEEE Transactions on Nanotechnology*, vol. 18, pp. 611–615, 2019.

[23] M. Fouda, F. Kurdahi, A. Eltawil, and E. Neftci, *Spiking Neural Networks for Inference and Learning: A Memristor-based Design Perspective*. Elsevier, Dec 2019, ch. 19, (in press). [Online]. Available: https://arxiv.org/abs/1909.01771

[24] F. M. Puglisi, C. Wenger, and P. Pavan, "A novel program-verify algorithm for multi-bit operation in hfo 2 rram," *IEEE Electron Device Letters*, vol. 36, no. 10, pp. 1030–1032, 2015.