Energy Efficient Learning With Low Resolution Stochastic Domain Wall Synapse for Deep Neural Networks

Walid Al Misba, Mark Lozano, D. Querlioz, Jayasimha Atulasimha

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 6 July 2022, accepted 23 July 2022, date of publication 5 August 2022, date of current version 17 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3196688

RESEARCH ARTICLE

Energy Efficient Learning With Low Resolution Stochastic Domain Wall Synapse for Deep Neural Networks

WALID AL MISBA[1], MARK LOZANO[1], DAMIEN QUERLIOZ[2], (Senior Member, IEEE), AND JAYASIMHA ATULASIMHA[1,3], (Senior Member, IEEE)

[1]Mechanical and Nuclear Engineering Department, Virginia Commonwealth University, Richmond, VA 23284, USA
[2]Université Paris-Saclay, CNRS, Centre de Nanosciences et de Nanotechnologies, 91120 Palaiseau, France
[3]Electrical and Computer Engineering Department, Virginia Commonwealth University, Richmond, VA 23284, USA

Corresponding authors: Walid Al Misba (misbawa@vcu.edu) and Jayasimha Atulasimha (jatulasimha@vcu.edu)

This work was supported in part by the National Science Foundation (NSF) under Grant ECCS 1954589 and Grant CCF 1815033 and in part by the Virginia Commonwealth Cyber Initiative (CCI) CCI Cybersecurity Research Collaboration Grant.

ABSTRACT We demonstrate extremely low resolution quantized (nominally 5-state) synapses with large stochastic variations in synaptic weights can be energy efficient and achieve reasonably high testing accuracies compared to Deep Neural Networks (DNNs) of similar sizes using floating-point precision synaptic weights. Specifically, voltage-controlled domain wall (DW) devices demonstrate stochastic behavior and can only encode limited states; however, they are extremely energy efficient during both training and inference. In this study, we propose both in-situ and ex-situ training algorithms, based on modification of the algorithm proposed by Hubara *et al.*, 2017 which works well with quantization of synaptic weights, and train several 5-layer DNNs on MNIST dataset using 2-, 3- and 5-state DW devices as a synapse. For in-situ training, a separate high precision memory unit preserves and accumulates the weight gradients which prevents accuracy loss due to weight quantization. For ex-situ training, a precursor DNN is first trained based on weight quantization and DW device model. Moreover, a noise tolerance margin is included in both of the training methods to account for the intrinsic device noise. The highest inference accuracies we obtain after in-situ and ex-situ training are $\sim$ 96.67% and $\sim$96.63%, respectively, which is very close to the baseline accuracy of $\sim$97.1% obtained from a similar topology DNN having floating-point precision weights with no stochasticity. Large inter-state intervals due to quantized weights and noise tolerance margin enables in-situ training with significantly lower number of programming attempts. Our proposed approach demonstrates a possibility of at least *two orders of magnitud*e energy savings compared to the floating-point approach implemented in CMOS. This approach is specifically attractive for low power intelligent edge devices where the ex-situ learning can be utilized for energy efficient non-adaptive tasks and the in-situ learning can provide the opportunity to adapt and learn in a dynamically evolving environment.

INDEX TERMS Domain wall, synapse, quantized weight, deep neural network, energy efficient, neuromorphic, in-memory computing.

I. INTRODUCTION

Deep neural networks (DNNs) have proven to be successful in image recognition and other big data driven classification tasks. However, implementing a DNN with traditional von-Neumann computing is time consuming [1] as it requires shuttling a large number of synaptic weight data stored in the memory to the processing unit to perform matrix-vector multiplication during the forward propagation and backward propagation stages. Moreover, shuttling data between the

The associate editor coordinating the review of this manuscript and approving it for publication was Juan Wang.

computational unit and memory unit is energy intensive [2], which hinders the implementation of such DNNs in edge devices where energy is at a premium [3], [4].

In-memory computing has been widely explored to reduce the physical separation between computation and memory units. In-memory computing is a non-von-Neumann computing paradigm where the computational memory units are arranged in a way that certain computational tasks take place in the memory itself [5], [6]. Matrix vector multiplication, the most computationally intensive part of a DNN [7], has been demonstrated with in-memory computing [8], [9]. When the computational memory units are connected in a crossbar and programmed to provide conductances equivalent to the DNN weights [10], [11], the matrix-vector multiplication operation can be implemented in single time step [5], [7] and with minimal data movement. Computational memory such as phase change random access memory (PCM) [12], [13], resistive random-access memory (RRAM) [14], [15], arranged in a crossbar array have been shown to classify handwritten digits [10], [16] and recognize human faces [17]. However, these analog memory devices have stochastic and non-linear responses and provide limited resolution for synaptic weights. To achieve higher classification accuracy, these issues should be addressed with appropriate training algorithms.

Recently spintronic memory devices have been widely explored for in-memory DNN implementation because of their non-volatility, high endurance, high speed of access, high scalability and compatibility with CMOS technology [1], [18]–[23]. Among these spintronic devices, domain wall (DW) based computational memory [18], [19] is promising and these devices can be programmed with a low energy budget [24]. However, similar to other analog devices, DW devices have limitations such as their stochastic behavior [25]–[28] and low resolution due to the relatively small on/off ratio of magnetic tunnel junctions (MTJs) which are 7:1, at best, at room temperature [29].

With recent advances in computing, researchers have shown fast and energy efficient implementation of DNNs with low resolution synaptic weights [30], [31]–[34] where the weights' values can only be binary (1-bit or 2-state) [31]. However, for updating weights, gradients are calculated in full precision to achieve high accuracy [30]. This idea of keeping full precision gradient information for training a network with limited precision synaptic weights can be useful for a DNN that is built from energy efficient DWs or other analog low-resolution devices.

Apart from the low resolution, stochasticity and non-linear response of the analog devices should be addressed during training to achieve higher classification accuracy [10]. To address the stochasticity of the analog devices, both online (in-situ) and offline (ex-situ) training of the DNN are proposed. For online training, multiple devices per synapse have been proposed with [35] or without 'periodic carry' [36] to address device variability and noise. In another work, a 3T1C module (consists of 3 CMOS transistors and 1 capacitor) is used in conjunction with a stochastic PCM device to accumulate small linear updates and then periodically transfer them to the non-volatile PCM [37]. However, with online training, using the techniques mentioned above during the weight update stage, each of the synaptic weights in the crossbar array are updated. This has great implications for the endurance of the devices as well as the energy consumed in training the device. Recently, a mixed precision framework [38], [39] has been proposed where large computational loads, such as weighted sum operation (matrix-vector multiplication) along with conductance updates, are performed in a low precision computational memory unit and the weight updates are accumulated in a high precision unit. Using this framework, a large variety of DNNs have been shown to achieve high classification accuracy with significantly smaller number of weight updates [38].

In contrast to the online training, for offline training the DNN is trained in software and the actual devices are programmed based on the learned weights from software. In this case, hardware nonidealities are characterized first and then included in the training process. To address stochasticity of the devices, Gaussian noise injection for the DNN weights has been proposed [40] and has shown excellent accuracy. Random Gaussian noise is also added to the ternary weights (3-state weight) of a DNN [41]. Variation aware offline training algorithm is reported in [42], [43] where the variation in device conductances and device defects are first characterized and then incorporated during the training of the DNN in software. In another case involving a deep convolutional neural network, the optimal weights for convolution layers and fully-connected layers are learned via offline training before the fully connected layers' weights being updated by online training [44].

Although a significant amount of research has focused on addressing the device variability and non-linearity, a largely unexplored area is quantized (low resolution) learning with these non-volatile devices. Even a high-resolution device (or a low granularity device) can behave as a low-resolution device, when device variability is taken into account. The limited numbers of synaptic states provided by low-resolution devices can strongly impact accuracy. At the same time, in neuromorphic computing applications, these devices offer advantages. An inherent benefit of the low-resolution devices can be their large inter-state interval. It provides an opportunity to train a neural network with significantly lower number of the weight updates with a standard learning rate. However, accuracy metrics need to be acceptable while using these devices for in-situ learning.

In this study, we demonstrate that such low resolution and stochastic non-volatile DW devices can perform highly accurate in-situ classification tasks while taking advantage of significantly lower number of device updates (higher energy efficiency). In our proposed algorithm, weight gradients are accumulated in high precision (digital domain) before quantizing this information to program the low-resolution

devices in analog domain. Using rigorous device model and simulations, we show that such in-situ training can achieve comparable accuracy to that of a 32-bit precision synapse. This demonstrates that there is a possible low-resolution weight space, which can provide an optimal solution to the classification problem with highly energy-efficient non-volatile devices. In addition, we have shown ex-situ training strategies to achieve high classification accuracy for DNN implemented with these highly stochastic (non-Gaussian) and extremely low resolution (nominally 2-state, 3-state, and 5-state for synaptic weights) analog DW based computational memory devices.

The rest of the paper is organized as follows. In the methods section, we detail the architecture of the DW device that can work as a synapse in the DNN and discuss the in-situ and ex-situ learning algorithms of such DW device based DNNs. For both of these learning algorithms we adapt quantized neural network learning algorithm [30], [31] with several modifications including the weight deviation tolerance from target weight to account for the programming noise intrinsic to such stochastic DW devices. For ex-situ training, we also incorporate the statistical distribution of the DW device conductance during the training, which helps to achieve higher test accuracy. This is followed by the results and discussion section, and then a conclusion.

II. METHODOLOGY

A. DW BASED NANO-SYNAPSE AND MICROMAGNETIC MODELING FOR DEVICE STOCHASTICITY

We model our synapse on a magnetic DW based nanodevice, which is non-volatile in nature. Once the memory state (here the synaptic weight) is written, the information is retained for a long time. For the nano-synapse device, we simulated a thin ferromagnetic racetrack having a dimension of 600 nm $\times$ 60 nm $\times$ 1 nm with a DW initialized and stabilized in a notch at one end. In addition, we assume several engineered notches at regular intervals along the racetrack. The racetrack dimension and notch intervals are shown in Fig. 1a. Moreover, we consider edge irregularities (rms roughness of $\sim$ 2 nm) in the racetrack to mimic the effect of lithographic imperfections by randomly removing or adding some finite difference cells from the edges [45], [46]. We assume the racetrack is on top of a heavy metal layer that is patterned on top of the piezoelectric layer (see Fig. 1b). An insulator (MgO layer) and a reference ferromagnetic layer (one could also add a synthetic antiferromagnet (SAF) layer to cancel dipole coupling from this fixed layer) are stacked on top of the racetrack, these two layers combined with the racetrack ferromagnetic layer (free layer) forms a magnetic tunnel junction (MTJ) (see Fig. 1b), which facilitates the readout of the device. With this configuration, a combination of fixed amplitude and fixed time current pulse ''or clocking signal'' injected in the heavy metal layer and a varying amplitude ''control'' voltage pulse applied across the piezoelectric translates the domain wall to different distances along the racetrack. Different positions of the DW lead to different conductances of the MTJ thus forming a voltage programmable non-volatile synapse.

B. MAGNETIZATION DYNAMICS

The magnetization dynamics of the domain wall (DW) in the racetrack ferromagnetic layer which governs the conductance (synaptic state) evolution of the nano-synapse is simulated in MUMAX3 [47] using the Landau–Lifshitz–Gilbert-Slonczewski equation. The simulation parameters are listed in Table 1. The simulation details can be found in [28].

TABLE 1. Simulation parameter.

Parameters	Values
DMI constant (D)	$0.0006\ Jm^{-2}$
Gilbert damping (ψ)	0.03
Saturation magnetization (M_s)	$10^6\ Am^{-1}$
Exchange constant (A_{ex})	$2 \times 10^{-11} Jm^{-1}$
Saturation magnetostriction (λ_s)	$250\ ppm$
Perpendicular Magnetic Anisotropy (K_u)	$7.5 \times 10^5\ Jm^{-3}$

C. MAPPING DOMAIN WALL POSITION TO CONDUCTIVITY

The distribution of equilibrium DW positions for five different programming voltages, represented by different perpendicular magnetic anisotropy (PMA) coefficient, K_u, in addition to fixed amplitude and fixed time spin orbit torque (SOT) generating current pulse ($35 \times 10^{10} \mathrm{A/m^2}$ applied for 1 ns) in the presence of room temperature thermal noise are shown in Fig. 1c. The mean equilibrium DW positions vary for different K_u, which implies that different programming voltages can be chosen for different synaptic states. For example, one can select five, three, or two different programming voltages to implement a 5-state, 3-state or 2-state synapse. The number of states that can be obtained from the device is limited due to the fact that with the presence of Dzyaloshinskii–Moriya interaction (DMI), SOT current causes DW titling long after the current stimulus is withdrawn [48]. Thus, in the presence of room temperature thermal noise and structural irregularities (edge roughness) DW motion becomes significantly stochastic. As a result, an average variance of $\sim$ 90 nm can be seen for different DW mean positions in our modelled racetrack. Considering all of these factors contributing to the stochasticity, we choose a maximum 5-state for our modelled device as higher number of states can cause larger overlaps between the states, which is detrimental for DNN performance. While it is possible to increase the number of states by increasing the racetrack dimension (increase area footprint) and/or increase the notch depth (increases energy as operating current increases), the

main contributions of our study is to show that we can use extremely low precision (5-state, 3-state, etc.) non-volatile synapses for in-situ (and ex-situ) DNN training and achieve classification accuracy that are comparable to that of full precision (32-bit) DNNs.

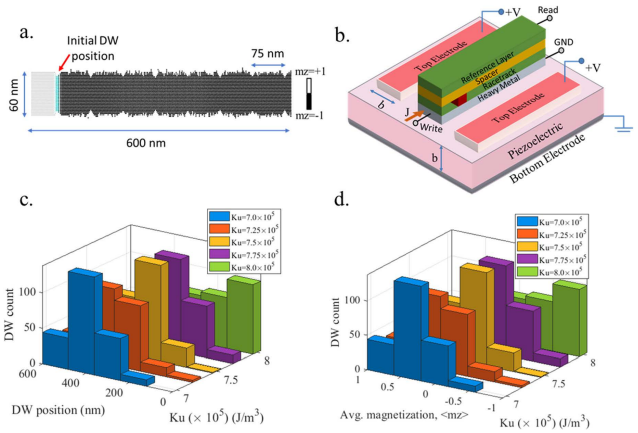


FIGURE 1. a. Micromagnetic configuration of a ∼2 nm rms rough edge racetrack with perpendicular magnetic anisotropy (PMA). Engineered notches are placed regularly at 75 nm interval. A DW is initialized at a notch 60 nm from the left of the racetrack. b. DW based nano-synapse device: racetrack ferromagnet/insulator/reference ferromagnet (MTJ) on top of a heavy metal layer on a piezoelectric substrate. A fixed amplitude current pulse, J in the heavy metal layer along with different amplitude voltage pulse, V across the piezoelectric changes the perpendicular anisotropy (PMA or K_u constant) of the racetrack and translates the DW (shown in red rectangle) to different longitudinal positions along the racetrack. c. Distribution of equilibrium DW positions in the racetrack (shown in Fig. 1a) at room temperature T = 300 K for a fixed SOT generating current pulse of $J = 35 \times 10^{10} A/m^2$ applied for 1 ns and five different PMA coefficients, K_u (corresponds to five different programming voltages). Different mean positions for different K_u implies that 5-state, 3-state or 2-state stochastic synapses can be implemented by choosing 5,3 or 2 different programming voltages. d. distribution of average perpendicular magnetization, $< m_z >$ (which is equivalent to DNN weights according to Eq. 4) derived directly from DW positions.

Equilibrium DW positions shown in Fig. 1c can be linearly mapped to a conductance value by means of the following equations:

$$G^{synapse} = \frac{G_{max} + G_{min}}{2} + \frac{G_{max} - G_{min}}{2} < m_z > \quad (1)$$

where, $< m_z >$ is the average magnetization moment of ferromagnetic racetrack along z-direction and the reference ferromagnetic layer magnetization is assumed to point upward, +z-direction. The distribution of $< m_z >$ is shown in Fig. 1d, which can be derived directly from DW position. G_{max} and G_{min} are the maximum and minimum conductance of the synaptic device which occur when the racetrack and reference layer magnetizations are parallel and anti-parallel respectively.

III. LEARNING OF FULLY CONNECTED DNN WITH DW NANO-SYNAPSE

A. CROSSBAR WITH DW DEVICES

We assume a crossbar architecture for the DW devices (Fig. 2b) that implements a fully connected DNN (Fig. 2a). The task of the DNN studied here is classification of handwritten digits from the MNIST database [49]. The network is trained with the MNIST training images each having 28 × 28 pixels or a total of 784 pixels with intensity values ranging from 0-255. The pixel intensities of the image converted to binary values acts as input to the DNN and the corresponding linearly mapped voltages are supplied to the crossbar using peripheral circuits. We have considered 3 hidden layers for the DNN and the numbers of neurons for input layer, hidden layers and output layer are chosen to be 784-392-196-98-10. The reason for the choice is discussed in the results section. The conductance of the devices can be scaled linearly to represent the weights W_{ij} of the DNN [18].

$$G_{ij}^{synapse} = \frac{G_{max} + G_{min}}{2} + \frac{(G_{max} - G_{min})\, W_{ij}}{2W_{max}} \quad (2)$$

Here, W_{max} is the maximum absolute value for the weights of the DNN. DNN weights, W_{ij} can be both positive and negative; however, the DW devices can only provide positive conductance values. To address the issue, one can add a parallel conductance, $G_P = \frac{G_{max}+G_{min}}{2}$ to each of the cross-points in the crossbar and feed this parallel conductance with a voltage that is of opposite polarity to the voltage applied to the DW device [18]. This parallel reference conductance, G_P can be achieved using a similar dimension DW device as the nano-synapse with the DW being driven and pinned at the center of the racetrack (in this case, $< m_z > \sim 0$ in Eq. 1). An engineered notch placed at the center of the racetrack can provide further pining strength to the DW in addition to the demagnetization potential minima, which acts near the center of the racetrack [28]. Stochasticity that could arise in programming the parallel conductances is not considered in our study.

Two separate rows supplied with opposite polarity voltages connects the synaptic devices and parallel conductances to a single column of the crossbar (bit line (BL)) as shown in Fig. 2b. The additional read line (RL), write word line (WWL) and source line (SL) shown in Fig. 2b are required to enable read and write operation. The WWL for the parallel conductances are not shown for the sake of simplicity. To read the column sum, RL is activated and WWL is deactivated, whereas a specific device can be read by activating RL and supplying read voltage to the corresponding input line. To program a device, WWL is activated and RL is deactivated and SL and BL are made high or low depending on the direction of the current in addition to a write voltage being supplied across the piezoelectric thickness. The effective conductance, G_{ij}, at each cross-point would be,

$$G_{ij} = \frac{(G_{max} - G_{min})\, W_{ij}}{2W_{max}} \quad (3)$$

Combining Eq. 1 and 2 and considering $W_{max} = 1$, one can get,

$$W_{ij} = < m_z > \quad (4)$$

From the above equation it is clear that if we train the DNN shown in Fig. 2a with weights (both positive and

negative) that are derived from micro-magnetics (see Fig. 1d) for different programming conditions, we are effectively implementing a hardware DNN with DW devices shown in Fig. 2b given that the peripheral circuitry is designed to provide the appropriate scaling.

During backward propagation, linearly scaled voltages corresponding to the output layer, L, error signal, $\delta_i^L = y_i^L - d_i^L$, are supplied to the crossbar, where y_i^L and d_i^L are the predicted and desired outcomes of the output layer's neuron i.

B. BACKPROPAGATION AND LEARNING ALGORITHM

For the training of the DNN, we update the weights by calculating the gradient of a cost function with respect to the weights. We considered mean square error, $C = \frac{1}{2}\sum (y_i^L - d_i^L)^2$ as our cost function where the gradient of the cost function with respect to the output of the output layer neuron i is expressed as, $\delta_i^L = (y_i^L - d_i^L)$ (we also call it error). Once the output layer's errors are determined, the preceding layer's errors can be calculated using the backpropagation equation, $\delta_i^l = W_{ij}\delta_j^{l+1}$, which is different from the backpropagation equation, $\delta_i^l = W_{ij}\delta_j^{l+1}f'_{l+1}$ reported in [50] where f'_{l+1} is the gradient of the activation function of layer $l+1$ neuron (we use sigmoid function as activation in our simulation). In other words, we do not backpropagate the gradients of activation function as it does not achieve high testing accuracy with quantized weights. Finally, the derivative of the cost function with respect to the weights is calculated, which determines the weight update signal, ΔW_{ij} for the weights connected between layer l neuron i and layer $l+1$ neuron j,

$$\Delta W_{ij} = \eta x_i^l \delta_j^{l+1} f'_{l+1} \tag{5}$$

Here, η denotes the learning rate. For our learning algorithm we propose to store the updated weights in a separate high precision memory unit. That way, the gradients with respect to the weights can be calculated accurately [30]. We note that these high precision weights are different from the actual synaptic weights (or equivalent conductances) provided by the DW device that are quantized and of low precision. However, we use these high precision weights to update the DW device weights (conductances). As we apply stochastic gradient decent for optimization, these high precision weights are updated at each forward pass with an input image.

As DW devices can only provide limited resolution in their synaptic weights we adopt weight quantization in our training algorithm. For that, the high precision weights are quantized at each forward pass during the training. For weight quantization, we use the following sets of functions in the manner of [51]:

$$clip\,(m, a, b) = min\,(max\,(m, a)\,, b)$$
$$\Delta = \frac{b-a}{n-1}$$
$$q = \left[round\left(\frac{clip\,(m, a, b) - a}{\Delta}\right)\right] \times \Delta + a \tag{6}$$

where, q is the quantized value of the real valued number m, $[a; b]$ is the quantization range and n is the level of quantization. The level of quantization depends on the number of distinct states (without significant overlap) the device that is used to implement the DNN crossbar arrays is capable of providing. After quantization, a programming pulse is generated to update the DW device weights to the quantized value, a target that is similar to the quantized neural network learning algorithm [30]. We note that, the cost gradients with respect to the prior quantization quantities are zero, so to backpropagate gradients through weight quantization we apply "straight through estimator" approach similar to that used in [30]. Typically, two types of training are possible for a DNN implemented with DW nano-synapse device: in-situ and ex-situ. In in-situ training the DNN is trained and tested in hardware. In contrast, in ex-situ training, a precursor DNN is first trained in software and then the DW devices are programmed to provide the equivalent learned weights prior to testing.

1) IN-SITU TRAINING

Here, we describe in detail the step-by-step in-situ training algorithm presented in Algorithm 1 and shown in Fig. 2b. This Algorithm 1 is based on the modification of quantized neural network algorithms presented in [30]. For each DW device in crossbar arrays there is a corresponding high precision weight that is stored in a separate digital memory unit to accumulate the weight gradients in full precision. Initially, these high (full) precision weights are chosen at random from a Gaussian distribution. After each forward and backward pass in the analog crossbar array, these weights are updated according to Eq. 5. Then, these weights are clipped and quantized so that they lie between -1 to 1. After that, a programming pulse is sent to the DW device to update its synaptic weight value to the quantized value. For example, in 5-level quantization (5-state for the synaptic device) the quantized weights can be of any value from the set $W_q \in (-1, -0.5, 0, 0.5, 1)$. Five different programming voltages can be applied to the device, which results in K_u = 8, 7.75, 7.5, 7.25 and 7.0 ($\times 10^5$) J/m^3 to achieve five different quantized weights of -1, -0.5, 0, 0.5 and -1 respectively as seen from Fig. 1d (DW device weights, $W_{ij} = < m_z >$ according to Eq. 4). Because of the significant spread that exists in the DW device weights (or the $< m_z >$ distribution) due to the stochastic nature of the device we introduce a noise tolerance hyperparameter called alpha, α (real valued) during training, as after applying a programming pulse (fixed current + control voltage) the device weights can be of a value other than the desired quantized weight. For instance, if we want to program a DW device to a quantized weight of W_q then we would allow any values for the device weights that satisfy the condition, $W_q - \alpha \leq W_{ij} \leq W_q + \alpha$. Therefore, at each iteration following quantization, we read the states of the DW device (costs read energy but that is

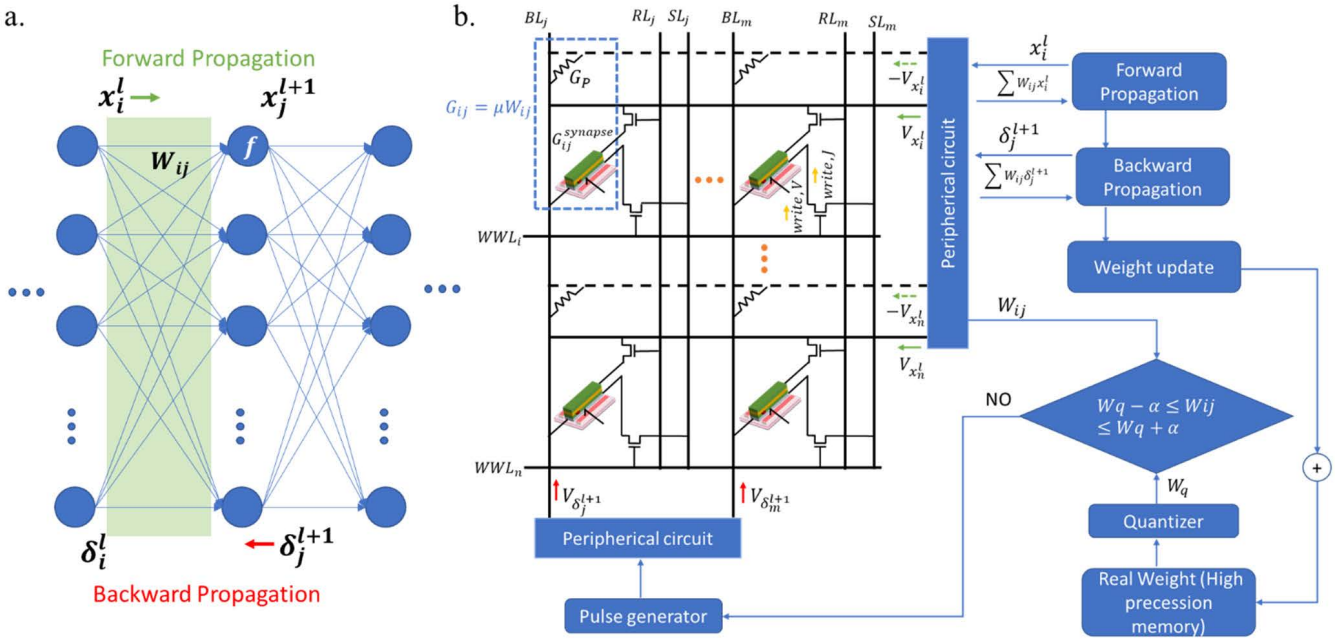


FIGURE 2. a. Architecture of a fully connected deep neural network (DNN). Any neuron i in layer l is connected to neuron j in layer $l+1$ with synaptic weight W_{ij}. At forward propagation, inputs to neuron j are summed and passed through an activation function f to generate its output, x_j^{l+1}. At backward propagation, errors of layer $l+1$ neurons are back propagated to calculate the error δ_i^l of neuron i in layer l b. Implementation of the DNN in crossbar with DW devices. The peripheral circuit and the crossbar shown here implements DNN functionalities of only one layer ("l") and the next layer ("$l+1$") and the number of rows in the crossbar are determined by the number of neurons in layer, l and the number of columns by the number of neurons in layer, $l+1$ (shadowed region in Fig. 2a). At each cross point of the crossbar there is a DW device with conductance $G_{ij}^{synapse}$ and a parallel conductance G_P. The effective conductance at each cross point is equivalent to the DNN weights W_{ij} such that $G_{ij} = \mu W_{ij}$. Inputs and errors of neurons are scaled to voltages before feeding them into the crossbar. The flow of the training algorithm is shown at the right-hand side of the crossbar. For each of the DW devices there is a corresponding high precision weight (real weight) that is stored in a separate high precision memory unit. These high precision weights are updated after a forward and backward pass before passing it through a quantizer (i.e., 2, 3 or 5-level quantization, depending on the number of states of the device). The DW device conductances, G_{ij} (or the corresponding device weights, $W_{ij} = \frac{G_{ij}}{\mu}$) are updated when they fall outside the prescribed range of the target quantized weights, W_q. Figure idea adopted from [38].

typically much lower than write energy) and if it falls outside the noise tolerance margin, a programming pulse is sent to the device to write the corresponding quantized weight. However, due to the large inter-state interval in quantized learning, a quantized weight does not change at each forward pass (the backpropagated errors update the weights slowly due to low learning rate). Instead, it typically changes only after several passes. Therefore, the noise tolerance condition need not be satisfied strictly at each iteration. Furthermore, if a DW device weight is programmed outside the tolerance margin, it is not rectified in the current iteration, as it has a chance to satisfy the window in the next several iterations. This relaxation over noise tolerance condition speeds up the training process without losing accuracy. Again, the DW device, which already satisfies the tolerance margin, need not be programmed for the next several iterations due to same reason of the quantized weights not being updated frequently. Introduction of noise tolerance hyperparameter, α, is critical during the training of this stochastic device based DNN. Without α, the DW device needs to be programmed a significantly large number of times to achieve a particular quantized weight. On the other hand, a high value of α allows more imprecise weight update or higher variation of the DW device weights from the target values, which will degrade the accuracy. Thus, a proper balance needs to be found for selecting the value of α so that it not only ensures high classification accuracy but also low programming energy. Once the DNN is trained, the learned DW device's weights (or conductances) remains the same during testing, as these devices are non-volatile.

We have chosen two representative noise tolerance limits for our study which are $\alpha = 0.15$ and $\alpha = 0.25$. Studies have shown that during training a Gaussian noise of standard deviation, σ that is up to 7.5% of the maximum magnitude of DNN weights does not degrade test accuracies significantly when no inference noise is assumed [40]. This motivates us to consider a noise tolerance limit of $\alpha = 0.15$ that is 15% of the maximum DNN weights (most of the weights in Gaussian distribution lies within $2\sigma \sim 15\%$). However, the DW device we studied here does have inference noise due to the device stochasticity. Furthermore, we choose a maximum noise tolerance of $\alpha = 0.25$ that is 25% of the maximum DNN weights so that the state overlaps between two adjacent states can be restricted for 5-state networks (half of the interstate interval for 5-state is 0.25).

We note that for 3-level quantization (3-state device), DW device can be programmed with control voltages to generate PMA of $K_u = 8$, 7.5, and 7.0 $(\times 10^5)\mathrm{J}/\mathrm{m}^3$ that can achieve quantized weights of -1, 0, and 1 respectively. For 2-level quantization (2-state device), the devices can be programmed to $K_u = 8$ and 7.0 $(\times 10^5)\mathrm{J}/\mathrm{m}^3$ to achieve weights of -1 and 1 respectively. During in-situ training, the device weights are selected randomly from the $< m_z >$ distribution of corresponding K_u to program the DW device to a target quantized value. Although we have computed 250 instances for each of the programming conditions (in Fig. 1c-d) due to the limitation in computational resources, we note that there are dominant pinning sites in the racetrack because of the notches. As a result, the DWs tend to be stuck in or close to those pinning sites in most cases rather than the pinning sites offered by the rough edges of the racetrack (see supplementary Fig. S2). Hence, generating more instances will likely follow the probability distribution, which already exists in the current distribution.

2) EX-SITU TRAINING

In this section, we discuss the steps of ex-situ training algorithm. The goal of ex-situ training is to achieve high testing accuracy in hardware although a precursor DNN is first trained in software. For this training, we also adopt weight quantization and allocate a separate memory in software where we store the high precision weights (similar to in-situ training) in addition to the DNN weights. The training algorithm shown in Fig. 2b remains the same for ex-situ training. After each iteration (forward and backward pass), high precision weights are updated and then quantized. Ideally, these quantized weights should be used as DNN weights for the next iteration in case of deterministic quantized neural network learning [30]. However, as we are dealing with a stochastic device for our inference engine, we include stochastic behavior of synaptic weights during learning. This stochasticity is obtained from a statistical distribution of the device (shown in Fig. 1d) rather than from uniform random distribution [30] or Gaussian distribution [40], [52]. For example, in 5-level quantization if the quantized weight is 0 then the DNN weight can be of any values selected randomly from the $< m_z >$ distribution of $K_u = 7.5$ $(\times 10^5)\mathrm{J}/\mathrm{m}^3$ which is responsible for generating quantized weight of 0 (see Fig. 1d). The noise tolerance margin α is also used during ex-situ training. This will relax the stringent requirement of programming a stochastic DW to a predetermined learned weight value and potentially save a large number of programming attempts. More importantly, if the DNN becomes aware of the statistical distribution of the device during training it can perform well during inference as the same device based DNN is used for inference.

Once the ex-situ training is accomplished, the DNN weights (or the high precision weights) are quantized and transferred to the DW devices by suitable programming. Here, the learned weights and the programmed weights may not be the same due to the programming noise. During the programming, we allow the same noise tolerance margin, α that is used during training. Thus, any programmed device weight, W_{ij} need to satisfy, $W_q - \alpha \leq W_{ij} \leq W_q + \alpha$ for a target-quantized weight of W_q. The devices can be programmed by repeated programming or performing read-verify-write operation in a loop, which is called "Open loop off device" method [53]. As we have already trained our network with stochastic distribution of weights by introducing finite α, the network is expected to perform well during testing when we allow the same noise tolerance level for programming the device.

Algorithm 1 In-Situ Training of a Quantized Neural Network With Crossbar Array of DW Devices. L is the Number of Layers, C is the Cost Function, λ is the Learning Rate Decay and α is the Noise Tolerance Margin for Writing the DW Devices. Quantize () Specifies How to Quantize a Weight With n-Level Quantization and Clip () Specifies How to Clip the Weights Based on Eq. 6. Update () Specifies How to Update Weights Once Their Gradients are Calculated Using Stochastic Gradient Decent. These Updated Weights are Accumulated in Full Precision (32-bit) in High Precision Memory Unit. Program () Specifies Sending a Voltage and Current Pulse to the DW Device to Write Its Conductance to a Target Quantized Weight

Require: a set of inputs and desired label (a^0, d^L), previous DW device weights $W_{ij,device}$ and corresponding full precision weights $W_{ij,fp}$, previous learning rate η.
Ensure: updated full precision weights $W_{ij,fp}^{t+1}$, corresponding DW device weights $W_{ij,device}^{t+1}$ and updated learning rate η^{t+1}.
{1. Forward propagation in analog DW device crossbar:}
for $k = 1$ to L **do**
$a^k \leftarrow a^{k-1} W_{ij,device}^k$
end for
{2. Backward propagation in analog DW device crossbar:}
{Gradients are computed in digital unit built from CMOS devices}
Compute gradient $g_{a^L} = \frac{\partial C}{\partial a^L}$ from a^L and d^L
for $k = L$ to 1 **do**
$g_{a^{k-1}} \leftarrow g_{a^k} W_{ij,device}^k$
$g_{W_{ij}^k} \leftarrow g_{a^k}^T a^{k-1}$
end for
{3. Accumulating the gradients in full precision in digital unit and update DW devices:}
for $k = 1$ to L **do**
$W_{ij,fp}^{k,t+1} \leftarrow \text{Update}\left(W_{ij,fp}^{k,t}, \eta, g_{W_{ij}^k}\right)$
$Level \leftarrow n//n$ represents maximum number of states of DW device
$W_{ij,q}^{k,t+1} \leftarrow \text{Quantize}\left(Clip\left(W_{ij,fp}^{k,t+1}, -1, 1\right), Level\right)$
if $\left|W_{ij,device}^{k,t} - W_{ij,q}^{k,t+1}\right| > \alpha$ **do**
$W_{ij,device}^{k,t+1} \leftarrow \text{Program}\left(W_{ij,device}^{k,t}, W_{ij,q}^{k,t+1}\right)$
end if
$\eta^{t+1} \leftarrow \lambda \eta^t$
end for

C. TESTING THE DNN

During the testing stage, we computed the predicted class for all the image samples from the MNIST test dataset using the trained DNN and compared it to the desired class. The percentage accuracy is calculated by dividing the total number of accurate predictions to the total number of test samples. During the testing stage, we consider two scenarios depending on in-situ or ex-situ training. When both the training and testing is performed on simulated hardware, the testing accuracy we record is termed online testing accuracy. In contrast, when the training is performed offline (ex-situ) in software and we program the hardware (simulated device in this case) prior to testing according to the learned weights then the testing accuracy we record is termed offline testing accuracy.

IV. RESULTS AND DISCUSSIONS

A. DNN CONFIGURATION SELECTION

The focus of our paper is to demonstrate the ability to classify images using a DW device based DNN and benchmark its performance against a DNN with floating-point precision (32-bit) weights. The topography of the benchmark DNN can be arbitrary as the inference accuracy varies across the spectrum of the parameters such as hidden layer number, layer size ratio (ratio of neurons between a layer and the previous layer) and learning rate constant (see Fig. S1 in supplementary). Thus, one can select multiple configurations for the DNN and achieve good accuracy. We select a benchmark DNN architecture consisted of a network with three hidden layers, an initial learning rate of 0.007 and a layer size ratio of $\frac{1}{2}$. Also, we assume a learning rate decay of 10 % after each epoch and use stochastic gradient decent method as the optimizer. The selection criteria are detailed in the supplementary section S1. After training the selected benchmark DNN for 10 epochs, the test accuracy we achieve is 97.1 %. We note that there are opportunities to improve the accuracy further in terms of topography, batch normalization, dropout layer and selection of different optimizers. However, the main goal of this study is to show how well a stochastic and low precision DW based DNN can perform in comparison to a similar architecture floating-point precision DNN. The selected topography mentioned above is used throughout the study to implement the DW device based DNN.

B. ONLINE (IN-SITU) TRAINING

After determining the DNN topography we investigate the test accuracies of the DNNs that are built from 2-state, 3-state and 5-state DW devices and trained with the proposed online training algorithm. For simplicity, we did not consider additional hardware non-idealities that could arise from peripheral circuits or unresponsive devices as these factors would automatically be included as constraints during the online training [40] and would not result in a significant degradation in performance compared to our current work.

The effectiveness of the proposed in-situ training algorithm is evident from Fig. 3a and Fig. 3b which plots the in-situ training accuracies for different state devices for low ($\alpha = 0.15$, 15% of maximum possible absolute weight) and high ($\alpha = 0.25$, 25% of maximum possible absolute weight) noise tolerance margin respectively. The results are also compared with baseline accuracy (accuracy of a same topography DNN with floating-point precision weights and no stochasticity). The training accuracies for DW device based DNNs increase with the number of device states and almost reach the baseline accuracy of $\sim$ 99.6 % for low noise tolerance of $\alpha = 0.15$ as can be seen from Fig. 3a. However, the training accuracies with high noise tolerance, $\alpha = 0.25$ become slightly lower (see fig. 3b) as these networks allow higher deviation from the target quantized weights. Nonetheless, competitive training accuracies are achieved for both 3- and 5- state devices with high noise tolerance margin.

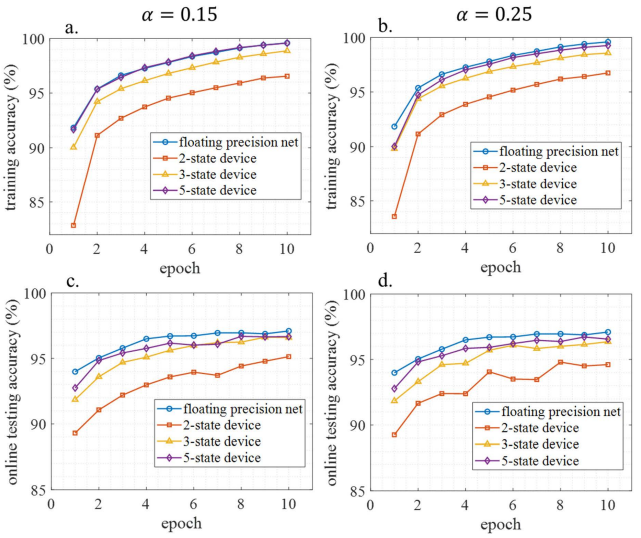


FIGURE 3. Online training accuracy and online testing accuracy for DNNs with different state DW devices for two different noise tolerance margins. These accuracies are compared with a DNN trained and tested with full precision weights and no stochasticity (baseline accuracy) a. and b. show the online training accuracies with the numbers of epochs for α = 0.15 and 0.25 respectively. c. and d. show online testing accuracies with numbers of epochs for α = 0.15 and 0.25 respectively.

After each epoch of the in-situ training we test the DNN with test images form MNIST dataset and compute the online test accuracy. Fig 3c and 3d plots online testing accuracies for low and high level of noise tolerance margin respectively. The baseline (DNN with floating-point precision weights and no stochasticity) test accuracies are plotted for comparison. For low noise tolerance margin of $\alpha = 0.15$, the test accuracy is highest for 5-state device and reaches $\sim$ 96.67% after 10 epochs of training. This accuracy is very close to the baseline test accuracy of $\sim$ 97.1 %. It is important to note that, the 3-state device based DNN achieves a test accuracy of $\sim$ 96.6 % after 10 epochs of training, which is similar to a 5-state device. When the noise tolerance margin is increased to $\alpha = 0.25$, the test accuracies for 5-state and 3-state devices

are ∼ 96.56% and ∼ 96.36% after 10 epochs of training. Thus, a maximum decrease of accuracy of ∼ 0.74 % from 32-bit precision weight is recorded for a 3-state stochastic weight. We note that, the test accuracies for 2-state device are ∼ 95.14% and ∼94.64% for low and high noise tolerance margin respectively. Thus, the same topography networks for 2-state does not achieve comparable test accuracies. Changing the topography, such as increasing the number of neurons in hidden layers, can increase the accuracy of binary DNN [54].

Next, we analyze the total number of programming pulses that are applied to the DW devices during the course of the online training at various epochs. Because the network updates the high-precision weights, a single weight may have its high precision value updated many times before crossing the threshold to update the DW device weight. As the number of device updates is dependent on the number of times a high precision weight crosses the threshold; the larger the threshold the fewer the updates. Between the 2, 3 and 5 state networks the 5-state has the smallest threshold, which increases the number of DW device updates as seen in Fig. 4. These DNNs are also compared with a DNN trained with floating-point precision weights (and no stochasticity). In floating-point precision DNN, all the weights are updated at each time a training image is passed to the network. Thus, although the network is better trained with increasing number of epochs, the weight update count remains almost constant as seen in Fig. 4. In contrast, for DNNs with limited state DW devices with the proposed training method, the programming instances decrease significantly with the number of epochs. As expected, with low noise tolerance margin the DNNs with DW devices become more selective and require higher number of weight updates during the course of training (though this is much smaller than the case of floating-point precision weights).

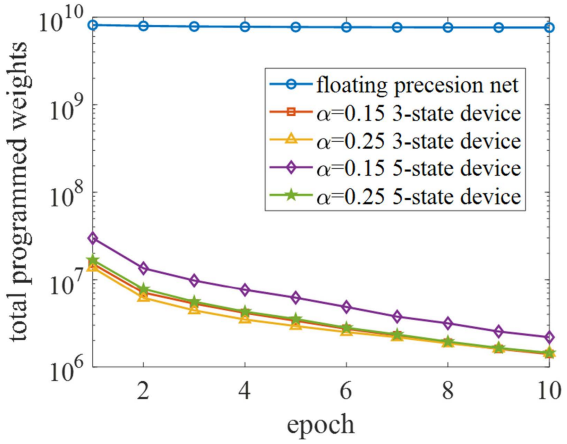


FIGURE 4. Comparison of the total number of programmed weights with the number of training epochs for different networks. A significantly lower number of weights are updated during the proposed online training compared to the floating precision weight network of the same architecture.

In Fig. 5, we show the convergence of DW device weights during the training. DNN weights whose noise tolerance are higher will converge to a value quicker, on an average, than a weight with a lower noise tolerance. In Fig. 5a and 5b, the DW device weights fall within $\pm\alpha$ of the quantized weight value. In both cases, the DW device weight is closer to the high precision value than the quantized weight, which tends to provide a higher accuracy for our DW based DNN.

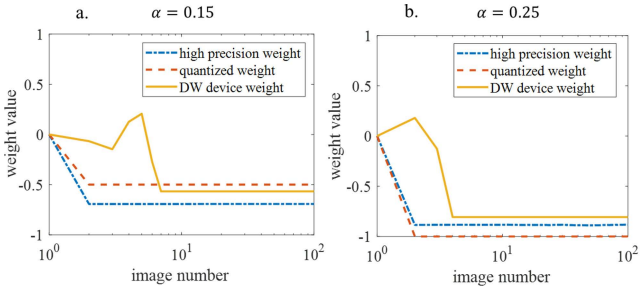


FIGURE 5. Weight evolution of high precision weight, quantized weight and the DW device weight during the first few training images for two different noise tolerance margin a. $\alpha = 0.15$ 0.15 b. $\alpha = 0.25$. The synaptic weight shown here is connected between the neurons located in hidden layer 2 and 3.

C. OFFLINE (EX-SITU) TRAINING

In this section, we first analyze the effectiveness of our proposed ex-situ training by comparing it with other techniques. For that, we train several precursor DNNs in software using different offline training algorithms (Fig. 6) and then test the DNNs, which are built from DW synaptic devices (3- state and 5-state hardware). Each of the DNNs are trained offline with a total of 10 epochs (train with entire training dataset 10 times) and prior to the testing the DW devices are programmed according to the weights that are learned offline. These results are shown in Fig. 6a and 6b when we consider a low ($\alpha = 0.15$) and high ($\alpha = 0.25$) value of noise tolerance margin to program the devices. In both Fig. 6a and 6b, for each of the hardware test accuracies, a corresponding software accuracy is presented side by side with green and yellow bar. When the exact learned weights (no programming noise is considered while transferring the learned weights to the device) are used to test the DNNs we call it software accuracy.

When offline training is performed with both the floating-point precision and quantized weights cases, the test accuracies are low for low noise tolerance margin, as can be seen from Fig. 6a. After floating-point precision weight training, the learned weights need to be converted to 3- or 5-state to program the DW devices. Thus, for both of the 3- and 5-state hardware the test accuracies degrade compared to software accuracy of ∼ 97.1%. Converting floating-point precision learned weights to 5-state compatible weights (5-level quantization) generates smaller deviations compared to the 3-state weight (3-level quantization). Thus the 5-state device provides higher test accuracy which is ∼ 87% compared to the 3-state which is only ∼10%.

Training with quantized weights (as proposed in [30]) improves the test accuracies to ∼ 90% for 5-state device

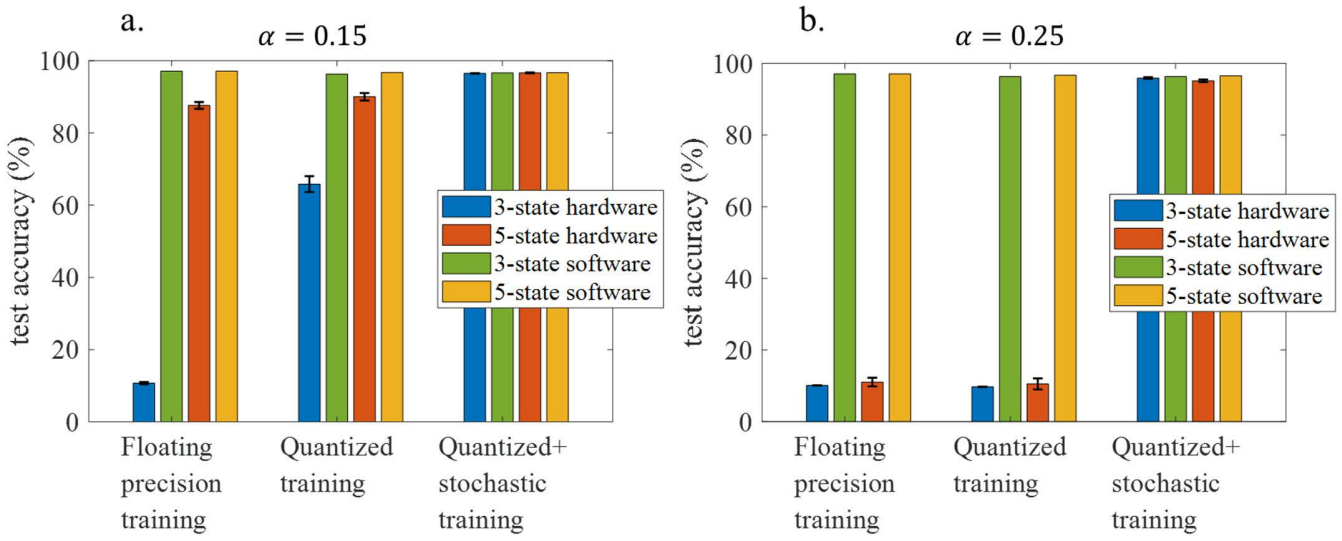


FIGURE 6. Testing accuracy comparison of 3-state and 5-state DW device based DNNs for different ex-situ training algorithms with a programming noise tolerance margin of a. $\alpha = 0.15$ b. $\alpha = 0.25$. The networks are trained offline with floating precision weights, quantized weights, and stochastic quantized weights derived from micromagnetic simulation. Each of the networks is trained with a total number of 10 epochs. Once training is done, the 3-state and 5-state DW devices are programmed based on the quantized value of trained weights prior to testing. For different training algorithms and for each of the test accuracies of DNNs built from 3- and 5-state hardware, a corresponding software test accuracy (no programming noise is considered and exact trained weights are used for testing the DNN) is plotted side by side with green and yellow bar. Error bar seen in the figure is calculated from 10 different test trials. For both noise tolerance margins, the test accuracy is highest when the DNNs are trained with proposed training algorithm (quantized + stochastic).

(see Fig. 6a) as the network becomes aware of the limited states of the weights during the training period. However, the test accuracy remains low (software accuracy is $\sim 96.74\%$). The accuracy loss is mainly due to the deviation of the programmed weights from the learned weights. We note that, with floating precision training, weight deviations occur in two ways: converting the floating-point precision weights to quantized weights and during the programming of the device where the target quantized weights are not achieved deterministically. However, with quantized training only the latter deviation occurs during the testing stage.

In contrast, with our proposed training which we call quantized + stochastic training, the test accuracy increases and reaches up to $\sim$ 96.63% for 5-state device, which is very close to the software accuracy of $\sim$96.67%. The accuracy improvement can be attributed to even smaller deviation of the programmed weights from the learned weights. Unlike quantized training, in our proposed training the weight quantization is also accompanied by training the DNN weights according to the statistical distribution of the device. As a result, during back propagation, the high precision weights are updated depending on the weighted sum performed over the imprecise DNN weights (which are mapped from the stochastic distribution of the device as in Fig. 1d). In other words, the high precision weights are being tuned based on the stochastic signature of the device. Thus, the statistical distribution of the device is embedded in the learning. When the same devices are used for testing, the distribution matches better and this plays an important role for improving the test accuracy. This finding is also supported by other works [40], [52]. Reference [40] shows that the DNN trained with Gaussian distributed weights of a certain standard deviation performs better when a weight distribution of same standard deviation is used for inference.

With high programming noise, for both floating precision and quantized training, the programmed weights deviate more from the learned weights because of the higher noise tolerance. Thus, the test accuracies for 3- or 5-state hardware degrade significantly compare to the software-based accuracies as seen from Fig. 6b. In contrast, with our proposed training method, the DNNs are made aware about the statistical distribution of the device thus resulting in significantly higher test accuracies compare than other offline training methods. (Note that as the device statistics are not Gaussian and instead heavily dominated by the pinning positions, training with Gaussian distributed weights does not improve accuracy and was not employed).

We also studied the evolution of offline test accuracies with the number of epochs for different state devices, which are presented in supplementary Fig. S3. The influence of noise tolerance margin α on training accuracy, online testing accuracy and offline testing accuracy for DNN with limited state device (5-state) is shown in supplementary Fig. S4, which shows that offline testing accuracy is affected most by the choice of different α.

D. ENERGY DISSIPATION

The energy required to program a DW synapse is determined from charging the piezoelectric layer with a voltage pulse, $\frac{1}{2}CV^2$ and I^2R loss due to the SOT current in the heavy metal layer. The maximum change in PMA is

$\Delta PMA = 0.5 \times 10^5 \mathrm{J/m^3}$. For magnetic racetrack of CoFe the saturation magnetostriction is, $\lambda_s = 250$ ppm. Thus, the maximum required stress, σ is, $\frac{\Delta PMA}{\frac{3\lambda_s}{2}} = 133$ MPa and the strain is, $\frac{133MPa}{200GPa} \sim 10^{-3}$, considering the Young's Modulus of CoFe to be 200 GPa. When the electrode dimensions are in the same order as the piezoelectric thickness, previous study [55] demonstrated that 10^{-3} strain is possible in Lead Zirconate Titanate (PZT) with an applied electric field of $E = 3$ MV/m. If we consider PZT layer to be $b = 60$ nm thick (same as top electrode or racetrack width as illustrated in Fig. 1(b)) then a voltage of, $Eb = 0.18$ V applied between the top electrode pair and the bottom electrode can generate the required strain. For a top electrode of length $L = 600$ nm (same as racetrack length 600 nm) and width $b = 60$ nm and a relative permittivity of PZT $\epsilon_r = 3000$, the effective capacitance is calculated to be $\frac{\epsilon_0 \epsilon_r (Lb)}{b} \sim 16$ fF. This predicts a $\frac{1}{2}CV^2$ loss of ~0.5 fJ, considering two top electrodes on each sides of the racetrack.

The heavy metal layer is considered to be Pt and for $600 \times 60 \times 5$ nm^3 dimension Pt layer the resistance is calculated to be 200 Ω assuming the resistivity of Pt to be 100 Ωnm. The heat loss in the heavy metal layer is calculated to be 2.2 fJ for a fixed SOT generating current pulse of magnitude $35 \times 10^{10} \mathrm{A/m^2}$ applied for 1 ns. Thus, the maximum energy dissipation to program a synapse is calculated to be 2.7 fJ.

1) IN-SITU TRAINING

With in-situ training, highest inference accuracy is achieved for a 5-state device when a low noise margin is considered during the training. However, with higher noise tolerance margin similar test accuracy is obtained with fewer device updates as can be seen from Fig. 4. For 5-state device, if we consider a noise tolerance margin of $\alpha = 0.25$, the total number of weight updates are calculated to be ~ 48 million after running the training for 10 epochs. *Thus, the energy dissipation to program the DNN synapses is calculated to be ~13* pJ *for one inference event followed by the weight updates* (10000 test images in MNIST).

2) EX-SITU TRAINING

With ex-situ training, highest inference accuracy is achieved for 5-state device when the noise margin to program the DW devices is considered to be low. Fig. 7 shows the cumulative probability of the DW device weights for different programming condition for a 5-state device. The solid black line represents the target quantized weights of 1, 0.5, 0, −0.5 and −1 (in this case −0.833) which can be achieved by a combination of fixed SOT current pulse and a varying amplitude voltage pulse which modulates the anisotropy of the racetrack to $K_u = 7, 7.25, 7.5, 7.75$ and 8.0 $(\times 10^5) \mathrm{J/m^3}$ respectively. The adjacent red dotted lines in the figure shows the noise margin ($\alpha = 0.15$) that is allowed while programming the DW device to a specific quantized state. From Fig. 7 it can be seen that the probability of programming the DW device weight to a quantized value of 1 is the lowest which is ~ 6 % meaning a number of ~ 20 attempt is required to program the device. If we consider the worst-case scenario, then after ex-situ training prior to the inference, we need 20 programing pulses to program each of the DW devices implementing the DNN weights. *Thus, for our network topology of 784-392-196-98-10 neurons, the energy dissipation to program the DW synapse is 2.8* pJ *per inference event.*

The energy dissipation to program the DW devices in in-situ training is found to be 5× the dissipation incurred in ex-situ training, which is moderately low provided that the training is performed over the entire 60000 training images for 10 epochs. This low dissipation in-situ training is possible due to distinct features of proposed training algorithm that benefits from weight quantization and noise tolerance margin. Large inter-state interval in quantized learning helps to reduce the number of weight updates. Moreover, once the device is programmed within the noise tolerance margin, further write operation is avoided with a simple low cost read operation. We note that onsite learning is attractive in power constraint edge devices, where the learning itself needs to adapt and respond to a continuously evolving environment. Embedded medical systems [56], real time intrusion detection [57], and dialect specific speech recognition systems can be benefitted from such onsite learning. Ex-situ learning can perform inference tasks in edge devices with energy efficient manner (given the training is performed over cloud server), however the benefit can only apply to non-adaptive tasks.

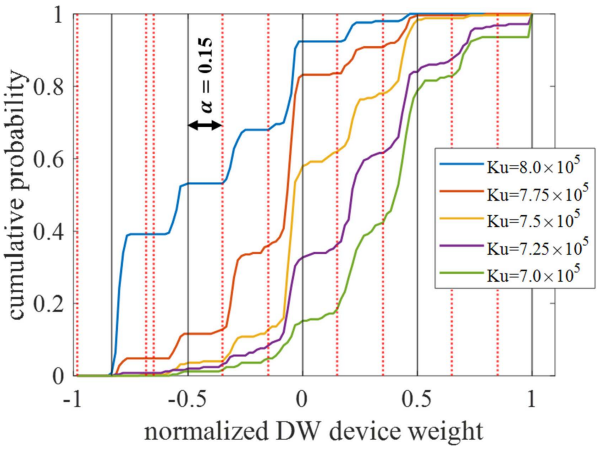


FIGURE 7. Cumulative probability of normalized DW device weights for 5-state device under different programming conditions denoted by different K_u. Black solid line represents the target quantized weights and the adjacent dotted red lines represent the programming noise tolerance margin of $\alpha = 0.15$.

Finally, the accuracy and energy consumption of our proposed DW based approach is compared with state-of-the-art techniques in the literature. The accuracies that we achieve for 5-state DW are comparable to the RRAM [54] and PCM [38] and better than the DW approach presented in [48] that can provide 32-states.

For energy comparison purpose, we first calculate the energy consumption of our proposed in-situ approach, including the energy expenditure for performing forward and backward propagation in the analog domain (crossbar devices) and weight gradient accumulation in the digital domain (high precision memory update). The details about energy calculation can be found in supplementary section S5. In addition, we estimate the energy consumption of a similar architecture deep neural network (DNN) with 32-bit precision weights (see supplementary section S5). Our proposed approach demonstrates a possibility of $\sim 165\times$ more energy saving compared to the 32-bit precision DNN implemented with on-chip CMOS static random access memory (SRAM).

The estimated energy consumption of $\sim$ 26 nJ per inference is comparable with state-of-the-art non-volatile technologies such as RRAM [54] and PCM [38]. Moreover, our proposed 5-state DW based DNN consumes less power compared to 32-state DW based DNN [48] for each synaptic weight update event as a 50 μA and 1 ns duration current pulse is used to program the 32-state synapses as opposed to our synapse that requires 21μA and 1 ns duration current pulse (Note that SOT clock dominates the energy consumed in our case). Further, the DW-based approach presented in [24] consumes an energy $\sim$ 8.64 fJ to program the synapse from one extreme conductance to the other, compared to our $\sim$ 2.7 fJ. However, Ref [24] does not take thermal noise and edge irregularities into consideration that could significantly reduce the number of distinguishable states due to device stochasticity. Finally, our algorithm ensures the number of times the weights are programmed are also significantly lower making the training cost very small.

V. CONCLUSION

We have shown that DNNs with extremely low resolution and stochastic DW device-based synapses can achieve high classification accuracy when trained with appropriate learning algorithms. In this study, both in-situ and ex-situ training algorithms are presented for DNNs that are implemented with 2-state, 3-state and 5-state DW devices. For in-situ training, a high precision memory unit is employed to preserve and accumulate the weight gradients, which are quantized to obtain target conductance for updating the low precision DW devices. A noise tolerance margin further allows for random deviations of the programmed conductances from the target conductance values. For ex-situ training, a precursor DNN is first trained in software by performing weight quantization and considering a noise tolerance margin from the quantized weight and later tested with an equivalent DNN of DW devices programmed with the same noise margin. While the energy dissipation statistics for programming the DNN synapses shows that ex-situ method is energy efficient, however, the in-situ training comes with an opportunity to learn and adapt to the changing environment with only $5\times$ more dissipation (despite the fact that the in-situ training is performed over a vast number of training images for many epochs). This technology is specifically attractive for low power intelligent edge devices of future IoT where energy requirement is at a premium. In future we are planning to extend our quantization aware stochastic DW device based DNN learning to convolutional, recurrent, long-short term memory and transformer based neural networks.

REFERENCES

[1] H.-S.-P. Wong and S. Salahuddin, "Memory leads the way to better computing," *Nature Nanotechnol.*, vol. 10, no. 3, pp. 191–194, Mar. 2015, doi: 10.1038/nnano.2015.29.

[2] A. Pedram, S. Richardson, M. Horowitz, S. Galal, and S. Kvatinsky, "Dark memory and accelerator-rich system optimization in the dark silicon era," *IEEE Design Test*, vol. 34, no. 2, pp. 39–50, Apr. 2017, doi: 10.1109/MDAT.2016.2573586.

[3] F. Jiang, K. Wang, L. Dong, C. Pan, W. Xu, and K. Yang, "Deep-learning-based joint resource scheduling algorithms for hybrid MEC networks," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6252–6265, Jul. 2020, doi: 10.1109/JIOT.2019.2954503.

[4] F. Jiang, L. Dong, K. Wang, K. Yang, and C. Pan, "Distributed resource scheduling for large-scale MEC systems: A multiagent ensemble deep reinforcement learning with imitation acceleration," *IEEE Internet Things J.*, vol. 9, no. 9, pp. 6597–6610, May 2022, doi: 10.1109/JIOT.2021.3113872.

[5] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nat. Nanotechnol.*, vol. 15, pp. 529–544, Mar. 2020, doi: 10.1038/s41565-020-0655-z.

[6] D. Ielmini and H. S. P. Wong, "In-memory computing with resistive switching devices," *Nature Electron.*, vol. 1, no. 6, pp. 333–343, Jun. 2018, doi: 10.1038/s41928-018-0092-2.

[7] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," in *Proc. 53rd Annu. Design Autom. Conf.*, Austin, TX, USA, Jun. 2016, pp. 1–6.

[8] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves, Z. Li, J. P. Strachan, P. Lin, Z. Wang, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, and Q. Xia, "Analogue signal and image processing with large memristor crossbars," *Nature Electron.*, vol. 1, pp. 52–59, Dec. 2017, doi: 10.1038/s41928-017-0002-z.

[9] G. W. Burr, R. M. Shelby, A. Sebastian, S. Kim, S. Kim, S. Sidler, K. Virwani, M. Ishii, P. Narayanan, A. Fumarola, L. L. Sanches, I. Boybat, M. Le Gallo, K. Moon, J. Woo, H. Hwang, and Y. Leblebici, "Neuromorphic computing using non-volatile memory," *Adv. Phys., X*, vol. 2, no. 1, pp. 89–124, Dec. 2016, doi: 10.1080/23746149.2016.1259585.

[10] G. W. Burr, R. M. Shelby, S. Sidler, C. Nolfo, J. Jang, I. Boybat, R. S. Shenoy, P. Narayanan, K. Virwani, E. U. Giacometti, B. N. Kurdi, and H. Hwang, "Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element," *IEEE Trans. Electron Devices*, vol. 62, no. 11, pp. 3498–3507, Nov. 2015, doi: 10.1109/TED.2015.2439635.

[11] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, pp. 61–64, May 2015, doi: 10.1038/nature14441.

[12] M. Suri, O. Bichler, D. Querlioz, O. Cueto, L. Perniola, V. Sousa, D. Vuillaume, C. Gamrat, and B. DeSalvo, "Phase change memory as synapse for ultra-dense neuromorphic systems: Application to complex visual pattern extraction," in *Proc. Int. Electron Devices Meeting*, Dec. 2011, p. 4, doi: 10.1109/IEDM.2011.6131488.

[13] T. H. Lee, D. Loke, K.-J. Huang, W.-J. Wang, and S. R. Elliott, "Tailoring transient-amorphous states: Towards fast and power-efficient phase-change memory and neuromorphic computing," *Adv. Mater.*, vol. 26, no. 44, pp. 7493–7498, Nov. 2014, doi: 10.1002/adma.201402696.

[14] S. Yu, Y. Wu, R. Jeyasingh, D. Kuzum, and H.-S. P. Wong, "An electronic synapse device based on metal oxide resistive switching memory for neuromorphic computation," *IEEE Trans. Electron Devices*, vol. 58, no. 8, pp. 2729–2737, Aug. 2011, doi: 10.1109/TED.2011.2147791.

[15] J. Woo, K. Moon, J. Song, S. Lee, M. Kwak, J. Park, and H. Hwang, "Improved synaptic behavior under identical pulses using AlO_x /HfO_2 bilayer RRAM array for neuromorphic systems," *IEEE Electron Device Lett.*, vol. 37, no. 8, pp. 994–997, Aug. 2016, doi: 10.1109/LED.2016.2582859.

[16] C. Li, D. Belkin, Y. Li, P. Yan, M. Hu, N. Ge, H. Jiang, E. Montgomery, P. Lin, Z. Wang, W. Song, J. P. Strachan, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, and Q. Xia, "Efficient and self-adaptive *in-situ* learning in multilayer memristor neural networks," *Nature Commun.*, vol. 9, no. 1, pp. 1–8, Jun. 2018, doi: 10.1038/s41467-018-04484-2.

[17] P. Yao, H. Wu, B. Gao, S. B. Eryilmaz, X. Huang, W. Zhang, Q. Zhang, N. Deng, L. Shi, H.-S. P. Wong, and H. Qian, "Face classification using electronic synapses," *Nature Commun.*, vol. 8, no. 1, pp. 1–8, May 2017, doi: 10.1038/ncomms15199.

[18] D. Bhowmik, U. Saxena, A. Dankar, A. Verma, D. Kaushik, S. Chatterjee, and U. Singh, "On-chip learning for domain wall synapse based fully connected neural network," *J. Magn. Magn. Mater.*, vol. 498, Nov. 2019, Art. no. 1654342, doi: 10.1016/j.jmmm.2019.165434.

[19] A. Sengupta, Y. Shim, and K. Roy, "Proposal for an all-spin artificial neural network: Emulating neural and synaptic functionalities through domain wall motion in ferromagnets," *IEEE Trans. Biomed. Circuits Syst.*, vol. 10, no. 6, pp. 1152–1160, Dec. 2016, doi: 10.1109/TBCAS.2016.2525823.

[20] D. Zhang, Y. Hou, L. Zeng, and W. Zhao, "Hardware acceleration implementation of sparse coding algorithm with spintronic devices," *IEEE Trans. Nanotechnol.*, vol. 18, pp. 518–531, 2019, doi: 10.1109/TNANO.2019.2916149.

[21] A. F. Vincent, J. Larroque, N. Locatelli, N. B. Romdhane, O. Bichler, C. Gamrat, W. S. Zhao, J.-O. Klein, S. Galdin-Retailleau, and D. Querlioz, "Spin-transfer torque magnetic memory as a stochastic memristive synapse for neuromorphic systems," *IEEE Trans. Biomed. Circuits Syst.*, vol. 9, no. 2, pp. 166–174, Apr. 2015, doi: 10.1109/TBCAS.2015.2414423.

[22] M. Alamdar, T. Leonard, C. Cui, B. P. Rimal, L. Xue, O. G. Akinola, T. P. Xiao, J. S. Friedman, C. H. Bennett, M. J. Marinella, and J. A. C. Incorvia, "Domain wall-magnetic tunnel junction spin-orbit torque devices and circuits for in-memory computing," *Appl. Phys. Lett.*, vol. 118, Mar. 2021, Art. no. 112401, doi: 10.1063/5.0038521.

[23] M.-C. Chen, A. Sengupta, and K. Roy, "Magnetic skyrmion as a spintronic deep learning spiking neuron processor," *IEEE Trans. Magn.*, vol. 54, no. 8, Aug. 2018, Art. no. 1500207, doi: 10.1109/TMAG.2018.2845890.

[24] D. Kaushik, U. Singh, U. Sahu, I. Sreedevi, and D. Bhowmik, "Comparing domain wall synapse with other non volatile memory devices for on chip learning in analog hardware neural network," *AIP Adv.*, vol. 10, no. 2, pp. 1–7, Feb. 2020, Art. no. 025111, doi: 10.1063/1.5128344.

[25] V. Uhlíř, S. Pizzini, N. Rougemaille, J. Novotný, V. Cros, E. Jiménez, G. Faini, L. Heyne, F. Sirotti, C. Tieg, A. Bendounan, F. Maccherozzi, R. Belkhou, J. Grollier, A. Anane, and J. Vogel, "Current-induced motion and pinning of domain walls in spin-valve nanowires studied by XMCD-PEEM," *Phys. Rev. B, Condens. Matter*, vol. 81, no. 22, pp. 1–10, Jun. 2010, doi: 10.1103/PhysRevB.81.224418.

[26] X. Jiang, L. Thomas, R. Moriya, M. Hayashi, B. Bergman, C. Rettner, and S. S. P. Parkin, "Enhanced stochasticity of domain wall motion in magnetic racetracks due to dynamic pinning," *Nature Commun.*, vol. 1, no. 1, pp. 1–5, Jun. 2010, doi: 10.1038/ncomms1024.

[27] J. P. Attané, D. Ravelosona, A. Marty, Y. Samson, and C. Chappert, "Thermally activated depinning of a narrow domain wall from a single defect," *Phys. Rev. Lett.*, vol. 96, no. 14, pp. 1–4, Apr. 2006, doi: 10.1103/PhysRevLett.96.147204.

[28] W. A. Misba, T. Kaisar, D. Bhattacharya, and J. Atulasimha, "Voltage-controlled energy-efficient domain wall synapses with stochastic distribution of quantized weights in the presence of thermal noise and edge roughness," *IEEE Trans. Electron Devices*, vol. 69, no. 4, pp. 1658–1666, Apr. 2022, doi: 10.1109/TED.2021.3111846.

[29] S. Ikeda, J. Hayakawa, Y. Ashizawa, Y. M. Lee, K. Miura, H. Hasegawa, M. Tsunoda, F. Matsukura, and H. Ohno, "Tunnel magnetoresistance of 604% at 300 K by suppression of Ta diffusion in CoFeB/MgO/CoFeB pseudo-spin-valves annealed at high temperature," *Appl. Phys. Lett.*, vol. 93, Aug. 2008, Art. no. 082508, doi: 10.1063/1.2976435.

[30] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 187, pp. 1–30, Apr. 2017.

[31] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, Montreal, BC, Canada, vol. 2, Dec. 2015, pp. 3123–3131.

[32] H. Zhang, J. Li, K. Kara, D. Alistarh, J. Liu, and C. Zhang, "ZipML: Training linear models with end-to-end low precision, and a little bit of deep learning," in *Proc. 34th Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, vol. 70, Aug. 2017, pp. 4035–4043.

[33] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2016, *arXiv:1606.06160*.

[34] D. Miyashita, E. H. Lee, and B. Murmann, "Convolutional neural networks using logarithmic data representation," 2016, *arXiv:1603.01025*.

[35] S. Agarwal, R. B. Jacobs Gedrim, A. H. Hsia, D. R. Hughart, E. J. Fuller, A. A. Talin, C. D. James, S. J. Plimpton, and M. J. Marinella, "Achieving ideal accuracies in analog neuromorphic computing using periodic carry," in *Proc. Symp. VLSI Technol.*, Kyoto, Japan, Jun. 2017, pp. T174–T175.

[36] I. Boybat, M. Le Gallo, S. R. Nandakumar, T. Moraitis, T. Parnell, T. Tuma, B. Rajendran, Y. Leblebici, A. Sebastian, and E. Eleftheriou, "Neuromorphic computing with multi-memristive synapses," *Nature Commun.*, vol. 9, no. 1, pp. 1–12, Jun. 2018, doi: 10.1038/s41467-018-04933-y.

[37] S. Ambrogio, P. Narayanan, H. Tsai, R. M. Shelby, I. Boybat, C. Di Nolfo, S. Sidler, M. Giordano, M. Bodini, N. C. P. Farinha, B. Killeen, C. Cheng, Y. Jaoudi, and G. W. Burr, "Equivalent-accuracy accelerated neural-network training using analogue memory," *Nature*, vol. 558, no. 7708, pp. 60–67, Jun. 2018, doi: 10.1038/s41586-018-0180-5.

[38] S. R. Nandakumar, M. Le Gallo, C. Piveteau, V. Joshi, G. Mariani, I. Boybat, G. Karunaratne, R. Khaddam-Aljameh, U. Egger, A. Petropoulos, T. Antonakopoulos, B. Rajendran, A. Sebastian, and E. Eleftheriou, "Mixed-precision deep learning based on computational memory," *Frontiers Neurosci.*, vol. 14, pp. 1–17, May 2020, doi: 10.3389/fnins.2020.00406.

[39] M. L. Gallo, A. Sebastian, R. Mathis, M. Manica, H. Giefers, T. Tuma, C. Bekas, A. Curioni, and E. Eleftheriou, "Mixed-precision in-memory computing," *Nature Electron.*, vol. 1, no. 4, pp. 246–253, Apr. 2018, doi: 10.1038/s41928-018-0054-8.

[40] V. Joshi, M. Le Gallo, S. Haefeli, I. Boybat, S. R. Nandakumar, C. Piveteau, M. Dazzi, B. Rajendran, A. Sebastian, and E. Eleftheriou, "Accurate deep neural network inference using computational phase-change memory," *Nature Commun.*, vol. 11, no. 1, pp. 1–13, May 2020, doi: 10.1038/s41467-020-16108-9.

[41] G. Boquet, E. Macias, A. Morell, J. Serrano, E. Miranda, and J. L. Vicario, "Offline training for memristor-based neural networks," in *Proc. 28th Eur. Signal Process. Conf. (EUSIPCO)*, Amsterdam, The Netherlands, Jan. 2021, pp. 1547–1551, doi: 10.23919/Eusipco47968.2020.9287574.

[42] L. Chen, J. Li, Y. Chen, Q. Deng, J. Shen, X. Liang, and L. Jiang, "Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Lausanne, Switzerland, Mar. 2017, pp. 19–24, doi: 10.23919/DATE.2017.7926952.

[43] B. Liu, H. Li, Y. Chen, X. Li, Q. Wu, and T. Huang, "Vortex: Variation-aware training for memristor X-bar," in *Proc. 52nd Annu. Design Autom. Conf.*, San Francisco, CA, USA, Jun. 2015, pp. 1–6, doi: 10.1145/2744769.2744930.

[44] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian, "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, Jan. 2020, doi: 10.1038/s41586-020-1942-4.

[45] E. Martinez, L. Lopez-Diaz, L. Torres, C. Tristan, and O. Alejos, "Thermal effects in domain wall motion: Micromagnetic simulations and analytical model," *Phys. Rev. B, Condens. Matter*, vol. 75, no. 17, pp. 1–11, May 2007, doi: 10.1103/PhysRevB.75.174409.

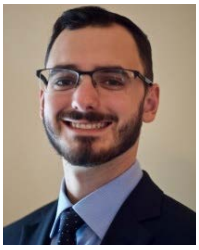
[46] S. Dutta, S. A. Siddiqui, J. A. Currivan-Incorvia, C. A. Ross, and M. A. Baldo, "Micromagnetic modeling of domain wall motion in sub-100-nm-wide wires with individual and periodic edge defects," *AIP Adv.*, vol. 5, Aug. 2015, Art. no. 127206, doi: 10.1063/1.4937557.

[47] A. Vansteenkiste, J. Leliaert, M. Dvornik, M. Helsen, F. Garcia-Sanchez, and B. V. Waeyenberge, "The design and verification of MuMax3," *AIP Adv.*, vol. 4, no. 10, Oct. 2014, Art. no. 107133, doi: 10.1063/1.4899186.

[48] S. Liu, T. P. Xiao, C. Cui, J. A. C. Incorvia, C. H. Bennett, and M. J. Marinella, "A domain wall-magnetic tunnel junction artificial synapse with notched geometry for accurate and efficient training of deep neural networks," *Appl. Phys. Lett.*, vol. 118, May 2021, Art. no. 202405, doi: 10.1063/5.0046032.

[49] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.
[50] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986, doi: 10.1038/323533a0.
[51] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," 2017, *arXiv:1712.05877*.
[52] T. Hirtzlin, M. Bocquet, J.-O. Klein, E. Nowak, E. Vianello, J.-M. Portal, and D. Querlioz, "Outstanding bit error tolerance of resistive RAM-based binarized neural networks," 2019, *arXiv:1904.03652*.
[53] B. Liu, H. Li, Y. Chen, X. Li, T. Huang, Q. Wu, and M. Barnell, "Reduction and IR-drop compensations techniques for reliable neuromorphic computing systems," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Jan. 2015, pp. 63–70, doi: 10.1109/ICCAD.2014.7001330.
[54] T. Hirtzlin, M. Bocquet, B. Penkovsky, J.-O. Klein, E. Nowak, E. Vianello, J.-M. Portal, and D. Querlioz, "Digital biologically plausible implementation of binarized neural networks with differential hafnium oxide resistive memory arrays," *Frontiers Neurosci.*, vol. 13, pp. 1–14, Jan. 2020, doi: 10.3389/fnins.2019.01383.
[55] J. Cui, J. L. Hockel, P. K. Nordeen, D. M. Pisani, C.-Y. Liang, G. P. Carman, and C. S. Lynch, "A method to control magnetism in individual strain-mediated magnetoelectric islands," *Appl. Phys. Lett.*, vol. 103, no. 23, Dec. 2013, Art. no. 232905, doi: 10.1063/1.4838216.
[56] T. Dalgaty, N. Castellani, C. Turck, K.-E. Harabi, D. Querlioz, and E. Vianello, "*In situ* learning using intrinsic memristor variability via Markov chain Monte Carlo sampling," *Nature Electron.*, vol. 4, pp. 151–161, Jan. 2021, doi: 10.1038/s41928-020-00523-3.
[57] M. S. Alam, B. R. Fernando, Y. Jaoudi, C. Yakopcic, R. Hasan, T. M. Taha, and G. Subramanyam, "Memristor based autoencoder for unsupervised real-time network intrusion and anomaly detection," in *Proc. Int. Conf. Neuromorphic Syst.*, Jul. 2019, pp. 1–8, doi: 10.1145/3354265.3354267.

WALID AL MISBA received the B.Sc. degree in electrical and electronic engineering from the Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2013, and the M.S. degree in electrical engineering from Tuskegee University, AL, USA. He is currently pursuing the Ph.D. degree in mechanical and nuclear engineering with Virginia Commonwealth University, Richmond, VA, USA.



MARK LOZANO received the B.S. degree in mechanical engineering from Virginia Commonwealth University, Richmond, VA, USA, in 2020, and the M.S. degree in computer science. His research interest includes designing machine learning algorithms for varying applications. After fulfilling his ongoing contract in Chantilly, VA, USA, he plans on going back to school to further research in neuromorphic computing.

DAMIEN QUERLIOZ (Senior Member, IEEE) received the Graduate degree from the Ecole Normale Supérieure, Paris, and the Ph.D. degree from Université Paris-Sud, in 2008. After post-doctoral appointments at Stanford University and CEA, he became a Permanent Researcher with the Centre for Nanoscience and Nanotechnology, Université Paris-Sud. He focuses on novel usages of emerging non-volatile memory, in particular relying on inspirations from biology and machine learning. He coordinates the INTEGNANO Interdisciplinary Research Group. He is currently a CNRS Research Scientist with Univeristé Paris-Sud. In 2016, he was a recipient of the European Research Council Starting Grant to develop the concept of natively intelligent memory. In 2017, he received the CNRS Bronze Medal.

JAYASIMHA ATULASIMHA (Senior Member, IEEE) received the M.S. and Ph.D. degrees in aerospace engineering from the University of Maryland, College Park, MD, USA, in 2003 and 2006, respectively. He is currently a Professor in mechanical and nuclear engineering and electrical and computer engineering with Virginia Commonwealth University, Richmond, VA, USA. His current research interests include magnetostrictive materials, nanoscale magnetization dynamics, and multiferroic nanomagnet-based computing architectures.