Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Stochastic learning in deep neural networks based on nanoscale PCMO device characteristics

Anakha V Babu[a], Sandip Lashkare[b], Udayan Ganguly[b], Bipin Rajendran[a,*]

[a] Department of Electrical and Computer Engineering, New Jersey Institute of Technology, NJ 07102, USA
[b] Department of Electrical Engineering, Indian Institute of Technology, Bombay, India

ARTICLE INFO

Article history:
Received 27 February 2018
Revised 10 July 2018
Accepted 6 September 2018
Available online 20 September 2018

Communicated by Dr Chenchen Liu

Keywords:
Deep Neural Networks (DNN)
Artificial Neural Networks (ANN)
Non-Volatile memory (NVM)
Stochastic Computing
Memristor

ABSTRACT

Deep Neural Networks (DNN) have proven to be highly effective in extracting high level abstractions of input data using multiple neural network layers. However, the huge training times for DNNs in traditional von-Neumann machines have hindered their ubiquitous adoption in IoT and other mobile computing platforms. Recently, acceleration of DNN with a time complexity of $\mathcal{O}(1)$ was proposed using the idea of stochastic weight update with resistive processing units (RPU). However, it has been projected that RPU devices require more than 1000 reliable conductance levels, which is a stringent requirement to realize in memristive devices. Here, we study the optimization of stochastic learning for DNNs for the hand-written digit classification benchmark using the characteristics of non-filamentary $Pr_{0.7}Ca_{0.3}MnO_3$ (PCMO) devices that are fabricated using a standard lithography process. The electrical characteristics of these devices exhibit a linear conductance response with an on-off ratio of 1.8 with 26 levels and significant programming variability. We captured these non-ideal behaviors of experimental PCMO device in the simulations to demonstrate stochastic learning with $\mathcal{O}(1)$ time complexity, achieving a test accuracy of 88.1% for the hand-written digit recognition benchmark. While the linearity, dynamic range, bit resolution, programming variability and the reset rate of the device conductance to account for its limited dynamic range have to be co-optimized for improving the training efficiency, we show that programming variability has the paramount role in determining the network performance. We also show that if devices with reduced programming variability (5x smaller compared to our experimental device) can be developed keeping all other parameters constant, it is possible to boost the network performance as high as 95%. We also observe that the performance of stochastic DNNs with memristive synapses is independent of the on-off ratio of the devices for a fixed programming variability. Thus, programming variability represents a new optimization corner for on-chip learning of stochastic DNNs. Further, we also evaluate the performance of stochastic inference engines to noise corrupted input test data as a function of the variability in the memristive devices. We demonstrate that noise-resilient inference engines can be achieved if 100 bits are used for stochastic encoding during inference even though the expensive network training can be done with as few as 10 bits. Thus, our studies emphasize the need for optimization of learning strategies for DNNs based on the non-ideal characteristics of experimental nanoscale devices.

1. Introduction

The human brain, with its exceptional structural organization and efficiency has inspired the development of today's modern microprocessors [1,2]. Human brain employs a massively parallel architecture based on a large interconnected network of neurons and synapses (local memory elements) for efficient computation. In contrast, the performance of conventional microprocessors is limited by the "von Neumann bottle-neck" for applications involving large data movement between processor and memory [3,4]. In an effort to mimic the brain's power efficient and fault tolerant computation, neural networks have been explored since 1940s [5–7]. Among these, the *second* generation Artificial Neural Networks (ANN) have shown unprecedented success recently. However, training of deep neural networks is computationally intensive and requires large training times with a time complexity of $\mathcal{O}(N^2)$ (N is the number of neurons in any layer) in von-Neumann machines; hence several hardware approaches have been proposed for accelerating DNN training [8,9]. However, none of these approaches have mitigated the limitations with respect to power, area and training time, and hence there are several proposals to employ

* Corresponding author.
E-mail address: bipin@njit.edu (B. Rajendran).

https://doi.org/10.1016/j.neucom.2018.09.019
0925-2312/© 2018 Elsevier B.V. All rights reserved.

non-volatile memory (NVM) based synapses for efficient acceleration of neural network training and inference [10–13].

Two-terminal memristive devices are an ideal choice for implementing electrical synapses due to its small size, enabling them to be densely packed in crossbar arrays. Further, thanks to their low power programming and read characteristics and the ability to store and retain multiple bits in a single device [14,15], they offer one possible way to emulate the brain's connectivity in hardware. Moreover, it has been numerically estimated that NVM based on-chip learning systems promise upto $25\times$ speed up and $3000\times$ improvement in power compared to GPU (Graphics Processing Unit) based implementations [16]. However, most of the memristive devices being explored today also have non-ideal characteristics such as finite on-off ratio, finite conductance resolution and has temporal and spatial conductance variability during programming and read [17,18]. Therefore, new architectural and device level optimizations are necessary to obtain the projected performance enhancements in hardware.

It has recently been proposed that DNNs can be implemented using tiled arrays of 2D crossbars of resistive processing units (RPU), which are memristive devices that can store multiple analog states and also adjust its conductivity based on identical sequence of voltage pulses [19]. If such crossbar arrays can be designed and all the weights in the array can be updated in parallel, the training time can be accelerated by replacing the vector cross-product operation with AND operation of stochastic bit streams representing neuronal signals. One of the most challenging requirements to be satisfied by an ideal RPU device is that it must be possible to incrementally program it to nearly 1000 reliable conductance states within a dynamic range of 10 by the application of identical sequence of voltage pulses. This is a stringent requirement and has not been demonstrated so far on experimental devices. $Pr_{0.7}Ca_{0.3}MnO_3$ (PCMO) based RRAM devices have been explored for neuromorphic hardware due to its analog conductance response by previous authors. However, most of these schemes use complex programming methods which cannot support parallel synaptic communication or updates, which is crucial for obtaining $\mathcal{O}(1)$ time complexity operation and hardware acceleration. In [20], numerical simulations of a 3-layer network with experimental PCMO characteristics having an on-off ratio of 5 and 256 conductance states showed a recognition accuracy of 90.55% for hand-written digit classification, although pulses with variable amplitudes were used for device programming. Linear and symmetric conductance response are shown to improve network performance [20] and several programming strategies have been explored to compensate for the non-linear and asymmetric conductance response at the cost of higher power and chip area [21–23].

Towards the goal of attaining parallel synaptic communication and weight update, we fabricated and characterized PCMO devices specifically optimized for analog and incremental programming upon the application of identical programming pulses. Using the measured characteristics, we study the performance of DNNs trained in a stochastic fashion for the exemplary hand-written image recognition task. We then conduct several numerical studies to determine the crucial device parameters for improving network accuracy and training times. While it is necessary to optimize the linearity, dynamic range, bit resolution, programming variability, and the frequency of conductance reset to account for the limited dynamic range of the synaptic device for improving the training time and convergence rate of hardware deep networks, we show that programming variability has the paramount role in determining the network performance. We also show that if devices with reduced programming variability can be developed keeping all other parameters constant, it is possible to attain close to baseline accuracies in PCMO based synaptic hardware.

This paper is organized as follows: We first discuss the basic ideas of stochastic computing and how it can be used for accelerating DNN training. A 4-layer network for hand-written digit classification is then analyzed as a benchmark for our studies based on conventional floating point implementation using stochastic weight updates. The basic 4-layer network is then extended to a crossbar compatible implementation with PCMO device conductances as the synaptic weights at the crossbar. The performance of stochastic DNNs is then studied for different optimizations of the PCMO device characteristics such as programming variability and dynamic range. Finally, we demonstrate the robustness of stochastic inference engines using low on-off ratio devices to noise corrupted test data by using higher precision for network encoding for inference as compared to training.

2. Stochastic computing

In the stochastic framework for computation, a number $x \in [0,1]$ can be represented as a Bernoulli sequence $X=[x_1, x_2, x_3, \ldots, x_N]$ such that the random variable x has a probability of $P(x_i=1)=x,$ and N is the length of the Bernoulli sequence [24–26]. In order to compute the product of two scalar quantities a and b, that are appropriately scaled in the range [0,1], two uncorrelated N bit long Bernoulli sequences representing a, b are created such that $P(a_i=1)=a,\ \ P(b_i=1)=b$. Let C represent the bit-wise logical AND operation of sequences A and B of variables a and b. Therefore,

$$P(c_i=1)=P(a_i=1)P(b_i=1)=ab \tag{1}$$

$$P(c_i=0)=1-P(a_i=1)P(b_i=1)=1-ab \tag{2}$$

Thus the product $c=a\times b,$ can be approximately obtained by the bit-wise logical AND or coincidence detection of the Bernoulli sequences A, B. The expectation and variance of the binary random variable C_i is $\mathrm{E}(C_i)=ab$, $\mathrm{Var}(C_i)=ab(1-ab)$. Let the number represented by the Bernoulli sequence $c=C_1, C_2, \ldots, C_N$ be obtained by averaging the N independent random variables C_i,

$$c=\frac{1}{N}\sum_{i=1}^{N}C_i \tag{3}$$

$$\Longrightarrow E(c)=ab\ ,\ Var(c)=\frac{ab(1-ab)}{N} \tag{4}$$

The error in the estimated average decreases with the length of the Bernoulli sequence N. Hence, a key advantage of stochastic representation is that multiplication can be implemented efficiently by using simple logic gates or coincidence detection [27–29]. Stochastic computing is suitable for applications that can tolerate certain level of inexactness such as image processing [30], error correcting codes [31] as well as in artificial neural networks [32,33].

3. DNN training and acceleration

Deep Neural Network (DNN) training is done in two steps - a forward pass to calculate the activations or the output of all the neurons in the network, and a backward pass to determine the weight update required for all the synapses in the network. In forward pass, the input y to neuron j in layer $(l+1)$ is calculated based on the inputs in the previous layer and the synaptic weights between these two layers, given by

$$y_j^{(l+1)}=\sigma\left(\sum_{i=1}^{N}w_{ij}^{(l)}x_i^{(l)}\right) \tag{5}$$

where σ is the non-linear sigmoid activation function. The time complexity involved in forward pass given in Eq. (5) of DNN training is $\mathcal{O}(N^2)$, with N being the number of neurons in each layer. In the backward pass, the error terms for each layer (l) is calculated based on the error in the succeeding layer $(l+1)$ using back propagation algorithm [34].

$$\delta_j^{(l+1)} \propto \sum_{k=1}^{N} w_{jk}^{(l+1)} \delta_k^{(l+2)} \tag{6}$$

$$w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} \pm \eta x_i^{(l)} \delta_j^{(l+1)} \tag{7}$$

As before, matrix multiplication in Eqs. (6) and (7) has a time complexity of $\mathcal{O}(N^2)$. Each of the weights in layer (l) is updated by multiplying the error in the succeeding layer $(l+1)$, the input to layer (l) and learning rate (η) as shown in Eq. (7). Hence, network training for data intensive applications such as audio and video analysis can have training times exceeding several days or months even when implemented in parallel architectures such as GPUs [35]. In order to accelerate DNN training in hardware, forward pass, backward pass and the weight update operations have to be executed in parallel. The vector-matrix multiplications involved in the forward and backward pass can be implemented using crossbar arrays with memristive devices at the cross-point [36–39].

Recently, a scheme to implement parallel weight update has been proposed using memristive resistive processing units (RPU) at the crossbar under the framework of stochastic computing [19]. In the parallel weight update scheme, stochastic pulses of constant amplitude (and opposite polarity) representing x_j and δ_j are fed to the crossbar from the row and column respectively. The pulse amplitude is chosen such that when there is no overlap between the pulses from the row and column, the device conductance remains un-perturbed, and only during coincidence of these pulses does the device conductance change. Ideally, the RPU conductance will change by an integer multiple of minimum allowed conductance change (Δw_{min}) per coincident pulse pair. Thus DNN training can be done with a time complexity of $\mathcal{O}(1)$ using RPU devices at the cross-point [19].

To perform DNN training with $\mathcal{O}(1)$ time complexity using stochastic computing, simulations suggest that the proposed RPU device should have more than 1000 reliable conductance states which is a stringent requirement to meet experimentally. Therefore the challenge is to achieve DNN training with real device specifications such as - limited on-off ratio, programming variability and limited conductance resolution (levels). This paper realizes a 4-layer stochastic DNN for hand-written digit classification using memristive synapses, whose synaptic behavior is derived from the electrical characteristics of the fabricated $Pr_{0.7}Ca_{0.3}MnO_3$ device. The electrical characteristics of $Pr_{0.7}Ca_{0.3}MnO_3$ devices are highly non-ideal, with a dynamic range in conductance (Gmax/Gmin) of 1.8, 26 discernible levels in this range and significant variability during programming. We then simulate and analyze the performance of stochastic DNNs to various device parameters-such as on-off ratios, sensitivity to conductance variations and tolerance to noisy input for inference.

4. Network architecture

A 4-layer deep network is selected for hand-written digit classification with 784-256-128-10 fully connected neurons in each layer as shown in Fig. 1 [19,40]. This network is trained with a standard hand-written digits database (MNIST - Modified National Institute of Standards and Technology) with 50,000 hand-written digits for training, 10,000 images for validation and a different set of 10,000 images for testing. Each of the input image has 784 pixel values in the grey scale and is preprocessed by normalization such that the input lies in the range [0,1] [41]. The 784 input neurons in the first layer takes these 784 pre-processed pixel inputs for training. Network learning is done by using multi-class cross entropy objective function with sigmoid activation function for the hidden layers and softmax function for the output layer. Batch size of unity is used for training with the weights updated after every image and the presentation of entire 50,000 images constitutes 1 epoch. A variable learning rate scheme is used for weight updates and all the network simulations for training and testing is done using MATLAB.

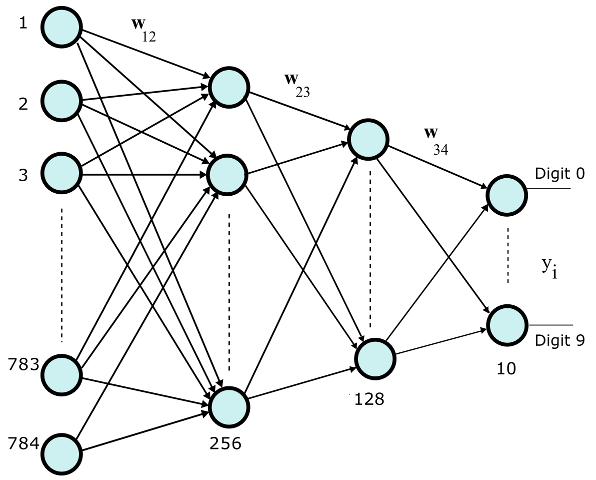


Fig. 1. A 4-layer deep neural network with 784-256-128-10 neurons in each layer used for hand-written digit classification (Simulated using MATLAB) [40].

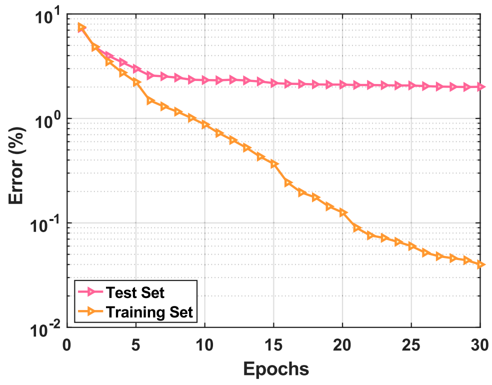
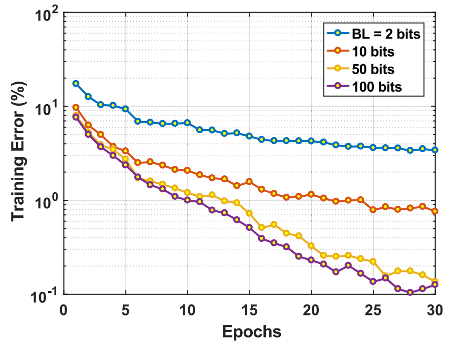


Fig. 2. Training and test error for floating point DNN with weight update using Eq. (7). Training and test error decreases with epoch and reaches a maximum test accuracy of 98%. This network is used as the baseline to determine the bit length required for stochastic DNNs.

4.1. Floating point DNN

As a baseline, the network which is trained by the floating point multiplication (without stochastic encoding) for weight update shown in Eq. (7) is referred to as 'Floating Point DNN'. After 30 epochs of training, we obtain state-of-the-art accuracies for fully connected networks of similar complexity (98%) (http://yann.lecun.com/exdb/mnist/). The corresponding test and training error is shown in Fig. 2. This network is used as the base line for optimizing stochastic DNNs.



Network	BL	Test Accuracy
Stochastic	2	95.95%
Stochastic	10	97.74%
Stochastic	50	98.09%
Stochastic	100	98.14%
Floating Point		98%

Fig. 3. Training error for stochastic DNN with $BL = 2, 10, 50, 100$ bits. $BL = 10$ bits is a reasonable choice for stochastic DNN (left); Table showing the maximum test accuracy of floating point and stochastic DNNs for hand-written digit classification. $BL = 10$ bits is a reasonable choice for stochastic DNN. Shown here is the maximum test accuracy across 5 different MATLAB iterations (right).

4.2. Stochastic DNN

The network which uses stochastic pulses for forward pass, backward pass and weight update is referred to as 'Stochastic DNN'. During forward pass, the input to each layer is converted to a stochastic pulse stream of length BL to compute the neuronal activations. For the backward pass given by Eq. (6), the error in the succeeding layer $(l+2)$ is converted to a stochastic pulse stream of length BL for calculating the error in the layer $(l+1)$. For the weight update, if x_i and $\delta_j \in [0,1]$ is represented by Bernoulli sequences, then the multiplication of x_i and δ_j can be implemented by coincidence detection as explained in section 2. The modified weight update rule is

$$w_{ij}^{(l)} = w_{ij}^{(l)} \pm B\left(\sum_{n=1}^{BL} x_{i,n}^{(l)} \wedge \delta_{j,n}^{(l+1)}\right) \tag{8}$$

where BL is the bit length that is used to approximately represent the real values in x_i, δ_j, and B is the minimum jump in conductance that can be programmed into the device and corresponds to one coincidence event of the sequences $x_{i,n}$, $\delta_{j,n}$. Based on the number of coincidences in the two sequences, the network weights get updated by an integer multiple of B. We first study the optimization of stochastic training of the network as a function of BL and is shown in Fig. 3.

The learning rates for stochastic DNN and floating point DNN are kept the same (as explained in [19]) and the corresponding simulated test accuracies for $BL = 2, 10, 50, 100$ bits is shown in the table in Fig. 3. The table lists the maximum test accuracy of 5 different seeds for the random generator in MATLAB. Since there is only marginal improvement in test accuracy for $BL > 10$, $BL = 10$ bits is used in the rest of the paper.

5. PCMO device as synapse

PCMO based RRAM devices are non-filamentary in nature and hence exhibit high endurance and low variability compared to the filamentary switching devices such as those based on HfO_2 [42]. PCMO devices are favored for neuromorphic hardware due to its simple structure, fast switching speed and area scalability [43,44]. Therefore, here we choose PCMO device as the memristive synapse for accelerating stochastic DNNs.

5.1. Device fabrication

The $Pr_{0.7}Ca_{0.3}MnO_3$ based RRAM devices were fabricated on 4" Si wafer using a 2 mask lithography process. To isolate the device from substrate, 300 nm thick SiO_2 was grown by thermal wet oxidation. Ti (20 nm)/ Pt (70 nm) was then deposited on

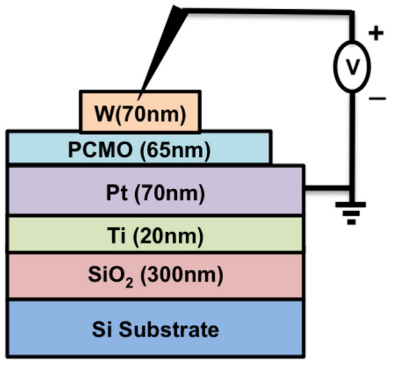


Fig. 4. $Pr_{0.7}Ca_{0.3}MnO_3$ device structure with Tungsten (W) as the top contact and Platinum (Pt) as the bottom contact.

SiO_2 by DC sputtering. Ti acts as an adhesion layer between SiO_2 and Pt with Pt as the bottom contact for the device. This was followed by deposition of 65 nm thick PCMO alloy using RF sputtering. Different sizes of devices were obtained by defining via-holes of 1 µm in SiO_2 by electron beam lithography (EBL). Finally, Tungsten (W) top contact pads were created using EBL followed by liftoff of W. The device schematic is shown in Fig. 4. All the electrical measurements were done using Agilent B1500A/B1530A semiconductor analyzer at room temperature.

5.2. Device characterization

The fabricated $Pr_{0.7}Ca_{0.3}MnO_3$ device is characterized and the corresponding resistive switching behavior is shown in Fig. 5(a). As can be seen from the current-voltage characteristics, significant non-linearity is observed in the conductance of the device as a function of voltage. For instance, the ratio $\eta = g(V)/g(V/2) > 50$ for the operating region of the device, where $g(V)$ is the conductivity of the device measured at voltage V. This allows PCMO based crossbar arrays to be used for synaptic and memory applications without a current limiting diode or access device at every cross-point, based on a $V/2$ programming scheme, as explained in Section 6.1.2 [45,46]. The selectivity of PCMO devices during programming greatly depends on the non-linearity in the I-V characteristics and is in-turn determined by the PCMO composition [44].

On the application of positive and negative voltage polarity, the PCMO RRAM shows SET (i.e. low to high conductance state) and RESET (i.e. high to low conductance state) switching respectively. The device is initialized to low conductance state by applying a RESET pulse with amplitude 2 V lasting 1 ms [46]. After initializing, WRITE pulses of amplitude 2.2 V and duration 100 ns were applied. A READ pulse (-0.5 V, 5 µs) follows each WRITE pulse to mea-

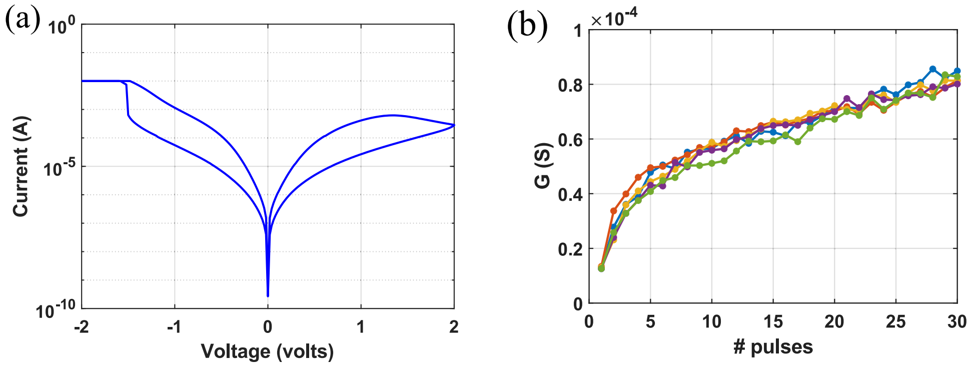


Fig. 5. (a) Current-Voltage characteristics of the fabricated PCMO device exhibiting significant non-linearity. (b) The conductance response of the PCMO device showing the variability across 5 different measurements on the same device. The approximately linear region of the conductance response from pulse #5 to pulse # 30, with an on-off ratio of 1.8 and a resolution of 26 states is used for training stochastic DNNs.

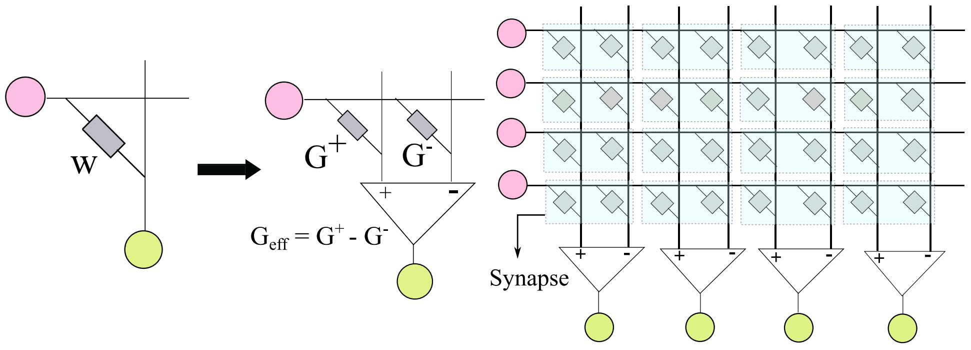


Fig. 6. Every synapse is represented using 2 PCMO devices scheme such that the effective synaptic weight, $G_{eff}= G^+ - G^-$ (left); Illustration of a fully connected crossbar network with 4 input neurons and 4 output neurons (right).

sure conductance change. The conductance response of PCMO device across 5 different measurements on the same device is shown in the Fig. 5(b). The approximate linear region in the conductance characteristics ranging from pulse number 5 to 30 is utilized for training stochastic DNNs. The conductance response clearly shows a variability for repeated measurements and the average programming variability is captured in the simulation of stochastic DNNs.

6. Crossbar compatible implementation

This section describes the details of the crossbar implementation of the 4-layer network for stochastic DNN training (Fig. 1).

6.1. Two-devices per synapse

As proposed in [47], 2 PCMO devices are used per synapse, so that both positive and negative weights can be realized in the network. The synaptic weight is encoded as the difference of the PCMO device conductances, i.e, $G_{eff} = G^+ - G^-$ as shown in Fig. 6.

If the minimum voltage required for increasing the device conductance is V_p, a $V_p/2$ pulsing scheme is used, where pulses of $+V_p/2$ amplitude are used to encode the 1s in the stochastic streams of the x_i in the rows and pulses of amplitude $-V_p/2$ are used to encode the 1s in the stochastic streams of the δ_j in the columns. Thus, when there is a coincidence of 1s in the two pulse streams, V_p voltage drops across the device, perturbing its conductance. For this pulsing scheme to work accurately, the synaptic device conductance is un-perturbed for pulses of amplitude $\pm V_p/2$, but undergoes conductance transitions for pulses of amplitude V_p. This condition is satisfied in PCMO devices as they exhibit a strong non-linearity as described earlier.

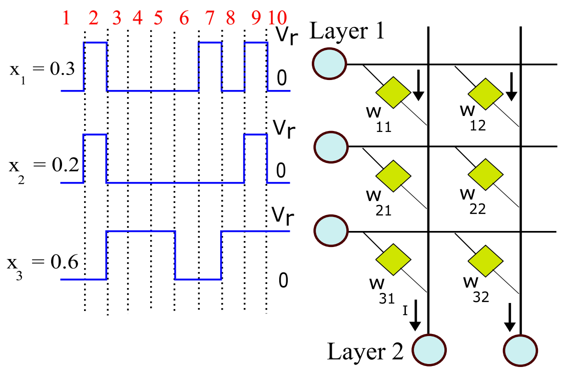


Fig. 7. Crossbar compatible implementation of forward pass (Read mode). The real valued input to each neuron in any layer is converted to a stochastic pulse stream and depending on the presence or absence of the pulse, current flows downstream to layer 2. This configuration is used for inference as it requires only forward pass.

6.1.1. Forward pass

In forward pass, the devices will be operated in read mode and a voltage smaller than the programming voltage (denoted by V_r) can be used (Fig. 7). The real valued input that lie in the range [0,1] is converted to a stochastic pulse stream with $BL = 10$ and applied to the crossbar wires across the rows. The forward pass is illustrated in Fig. 7 with 3 neurons in layer 1 and 2 neurons in layer 2. The stochastic pulse stream has 10 slots (since $BL = 10$), and depending on the presence or absence of pulse a current flows downstream and gets sensed by the sense amplifier. Thus the forward activation functions can be calculated across all layers in parallel achieving a time complexity of $\mathcal{O}(1)$.

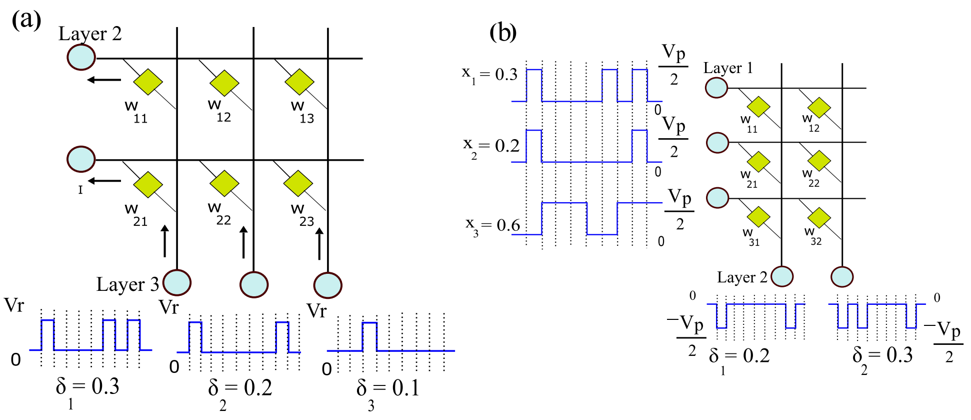


Fig. 8. Illustration of backward pass. (a) First step of backward pass (Read mode): Matrix multiplication of the error term with the transpose of the weight matrix is done by feeding the error voltage as stochastic pulses to the column such that the total current is summed across the neurons in layer 2. (b) Second step of backward pass (Write mode): For updating the weights in layer 1, the error obtained at layer 2 and input to layer 1 is scaled by the learning rate and applied as stochastic pulses through the crossbar wires. A $V_p/2$ pulsing scheme is used such that the conductance changes only when each device experiences a minimum voltage of V_p. In the case of inference, the configurations shown in (a) and (b) will not be used as no training is involved.

6.1.2. Backward pass

The backward pass involves two steps - illustrated by Eq. (6), (7) and is demonstrated in Fig. 8 (a), (b). In the first step, matrix multiplication of transpose of the weight matrix with the error term is achieved by passing the stochastic error voltage pulses as input to the columns (layer 3) and the currents get summed up across the rows in layer 2. Since the devices are operated in read mode, a voltage of V_r is used to encode the stochastic pulse stream. In the second step of back propagation, voltages corresponding to these current values are then adequately scaled to incorporate the learning rate and presented across the column as stochastic pulses as shown in Fig. 8 (b). The input to layer 1 will also be scaled and subsequently converted to stochastic pulse streams and applied along the rows. Based on the number of coincidences in the two pulse streams (across the column and row), the device will experience an effective voltage of $V_p/2 - (-V_p/2) = V_p$, thereby changing the conductance by an integral multiple of the minimum resolution B (which in real devices could exhibit significant variability). As a result, by effectively utilizing the locality and parallelism of the back propagation algorithm, weight updates can happen in parallel across all devices in the crossbar achieving a time complexity of $\mathcal{O}(1)$ [37–39].

Since most of the memristive devices exhibit gradual conductance change in one direction, we assume a unidirectional weight update scheme [48]. In this scheme, conductance of the device will be always increased and the device representing G^+ (G^-) will be selectively programmed to increase (decrease) the weight. The weight update schematic is illustrated in Fig 9. If the calculated incremental weight update (ΔW) using the back propagation algorithm is positive, then the conductance of G^+ (shown in green in Fig. 9) is increased so that G_{eff} is increased. On the other hand, if the weight update required is negative, then the conductance of G^- is increased to effectively reduce G_{eff}.

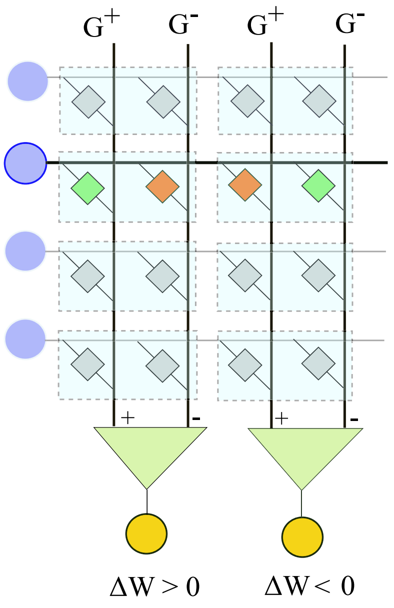


Fig. 9. Unidirectional weight update scheme in 2 PCMO devices per synapse in the crossbar. When the required weight update (ΔW) for a synapse > 0, G^+ will get selected and the conductance increases such that $G_{eff} = G^+ - G^-$ is effectively increased. On the other hand, if $\Delta W < 0$ then G^- is selected for weight increment and $G_{eff} = G^+ - G^-$ is reduced. The device marked green is selected for weight increment and the unselected is indicated in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6.2. Modifications to the 4-layer network

All the PCMO devices in the network are initialized randomly between the minimum and maximum conductance values. As the actual device conductance values are not compatible for network learning, they are appropriately scaled during training. The scaling factor is absorbed in the calculation of activation functions. Due to the limited on-off ratio and conductance resolution, the devices could easily saturate and learning could stop. To avoid this, the devices are reset periodically (data refresh) based on the conductance values [47]. During reset, if the effective synaptic weight (G_{eff}) is positive, then G^+ is set to $(G_{min} + G_{eff})$ while G^- is kept to G_{min} such that G_{eff} is unperturbed. Similarly, when G_{eff} is negative, G^+ is set to G_{min} and G^- is set to $(G_{min} + G_{eff})$. Therefore the effective weight $G_{eff} = G^+ - G^-$, at any synapse will remain unaffected after a reset operation.

We study two possible reset mechanisms - conditional reset and global reset. In conditional reset, the devices are reset whenever one of the two devices of the synapse reach the maximum conductance value and no more increment in conductance is possible. However, conditional reset requires constant monitoring of conductance at every synapse. A cost effective solution is hence to use a global reset in which all the devices are reset after a fixed number of training steps [49]. We study the optimization of reset intervals in the subsequent Section 7.1.

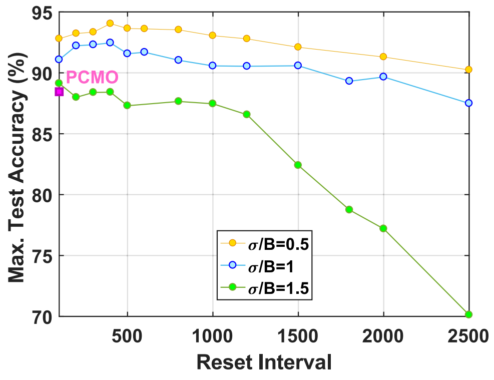


Fig. 10. Simulated maximum test accuracies for stochastic DNNs with different choices of reset intervals. An optimum reset interval of 400 is chosen for devices having a variability ratio of σ/B=0.5, 1 and 100 is used for devices with variability ratio of σ/B=1.5. The accuracy of the experimental PCMO device based network from Fig. 11 is also shown.

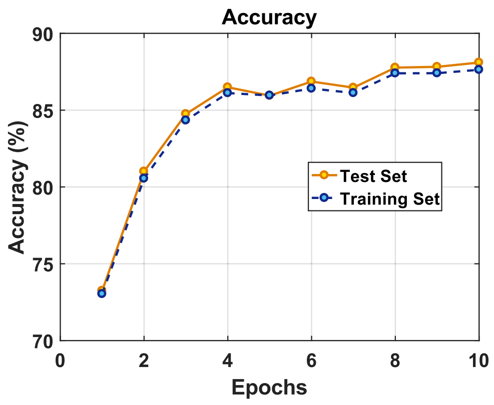


Fig. 11. Simulated test and training accuracy of stochastic DNNs with experimental PCMO device at the synapse. Shown here is the average response of 3 different MATLAB iterations. The test accuracy reaches a maximum value of 88.1% after 10 epochs.

7. Network response with PCMO device

The baseline stochastic network is extended to a crossbar compatible implementation as mentioned in the previous section with 2 PCMO devices per synapse. Here we simulate stochastic DNNs using $Pr_{0.7}Ca_{0.3}MnO_3$ devices, whose synaptic behavior is determined from the electrical measurements as explained in Section 5.2. These PCMO devices has an on-off ratio of 1.8 and approximately only 26 discernible levels including the minimum and maximum conductance states. The conductance of the device varies linearly with pulse number and has a programming variability associated for each incremental change. The impact of programming variability on network performance is studied using the parameter σ/B, where σ is the standard deviation of the Gaussian noise and B is the minimum change in conductance obtainable under the chosen programming conditions. As discussed, the devices have to be periodically reset to facilitate continuous learning and the reset interval in turn depends on the amount of programming variability (σ/B).

7.1. Optimization of reset interval

During network training, the devices are periodically reset after presenting a fixed number of input images to the network. Here we study the optimization of reset interval for $Pr_{0.7}Ca_{0.3}MnO_3$ devices with different programming variability (captured by the parameter σ/B). The optimization of reset interval for σ/B=0.5, 1, 1.5 is shown in Fig. 10.

The choice of the reset frequency is based on the network performance. In these simulations, the variability encountered by the devices during the reset process is assumed to be negligible compared to the programming variability associated with the weight update as iterative programming schemes can be used at the time of the infrequent reset operation [50]. Therefore from Fig. 10, the reset interval is chosen to be 400, 400, 100 for $\sigma/B = 0.5,\ \ 1$ and 1.5, respectively. In the case of devices with low programming variability having $\sigma/B \leq 0.5$, the network can tolerate a higher reset interval with minimum degradation in test accuracy. The conductance deviation for our experimental PCMO is approximately 1.59 × the bin-width (B) of the conductance states ($\sigma = 1.59B$). The training and test accuracy corresponding to stochastic DNN with PCMO device is shown in Fig. 11. A reset interval of 100 is used for simulating PCMO based memristive synapses to obtain a maximum test accuracy of 88.1% after 10 epochs. The inferior performance of the network from the baseline is due to the increased variability in programming, causing the device conductance levels to spill over into the neighboring levels. In order to understand the fundamental reasons for performance degradation with memristive implementations and to optimize device and network performance, we now study the role of specific device characteristics on training and inference accuracy.

7.2. Sensitivity to conductance variability

Conductance variability is a critical parameter of practical NVM devices and its effect on the learning of stochastic networks for hand-written digit classification is discussed here. Repeated pulse

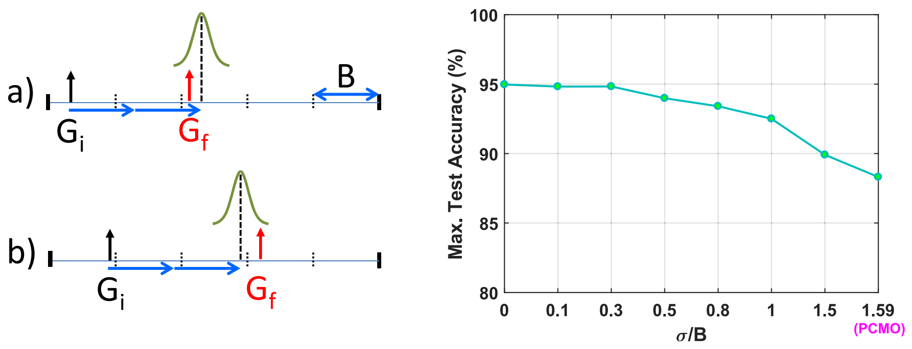


Fig. 12. Cartoon illustrating the effect of memristive programming variability with an initial conductance G_i, final conductance G_f and σ representing the standard deviation of programming noise. (a) Final Conductance (G_f) obtained by a pulse overlap of 2 from the initial conductance (G_i) exhibits an underestimate, and (b) represents an overestimate of G_f [40]; Maximum generalization (test) accuracy of stochastic DNNs ($BL = 10$) when trained with PCMO devices of different programming variability. Here σ/B is the ratio of standard deviation of the conductance variability to the bin-width of the conductance states (right).

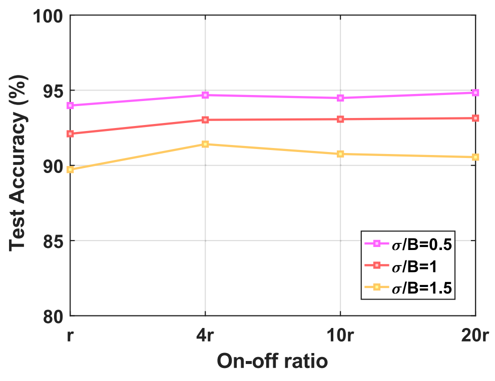


Fig. 13. Comparison of test accuracy of stochastic DNNs for different on-off ratios of PCMO device at the crossbar. Here, r denotes the experimental on-off ratio of 1.8. The average test accuracies of three different MATLAB iterations is shown here. The accuracy is almost independent of the on-off ratio of the device for a fixed σ/B ratio.

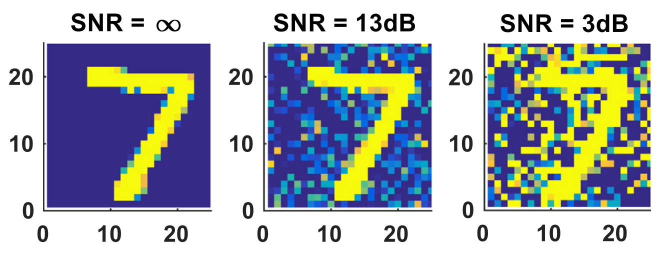


Fig. 14. A sample test image '7' showing the signal to noise ratio (SNR) after introducing Gaussian noise of different variances. The noise added images are then fed as input to stochastic DNNs to evaluate the inference response.

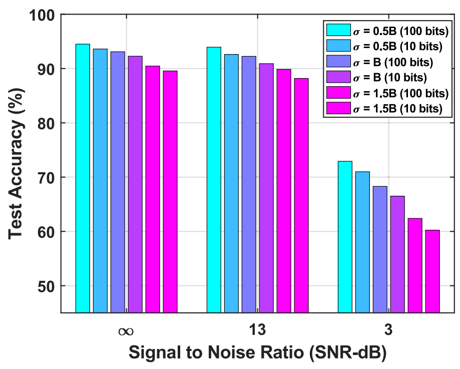
measurements on the fabricated $Pr_{0.7}Ca_{0.3}MnO_3$ device shows significant variations in the conductance values for the same pulse number (see Fig. 5). The parameter σ/B, which is the ratio of the standard deviation of the PCMO conductance variability to the bin-width of the conductance levels, is used to study the impact of programming variability on network performance. An illustration of memristive programming variability is shown in Fig. 12 [40]. The conductance variations are introduced in the simulations as a zero mean Gaussian noise with a standard deviation $\sigma/B = 0.1, 0.3$ and so on. The synaptic devices are randomly initialized to any of the 26 conductance states with programming variability. Except for the standard deviation of the conductance variations, all the other parameters such as on-off ratio and number of levels are kept the same as that of experimental PCMO device. The response of stochastic DNNs is observed for $\sigma/B = 0, 0.1, 0.3, 0.8, 1, 1.5$ as shown in Fig. 12. For $\sigma/B < 0.5$, where the standard deviation of the Gaussian is lesser than the bin-width, the obtained device conductance is close to the desired conductance resulting in close-to baseline accuracies. With standard deviations close to zero, the network achieves a test accuracy close to 95% even with a low on-off ratio of 1.8. This shows that programming variability is critical for training of stochastic DNNs when compared to on-off ratio of the memristive devices. When σ/B ratio is gradually increased from 0 to 0.3, the test accuracy remains almost constant and degrades by 2% at $\sigma/B = 1$. For $\sigma/B > 1$, where the standard deviation is more than the bin-width, the device conductance could easily jump to the neighboring conductance states thus showing inferior network performance.

7.3. Impact of on-off ratio

The performance of stochastic DNNs is analyzed for different on-off ratios of the PCMO devices while keeping the number of levels and programming variability (σ/B) invariant. The device is assumed to have a resolution of 26 states and different scaling for the device on-off ratio compared to the experimental value. The corresponding test accuracy for different on-off ratios ($4r$, $10r$ and $20r$ compared to the baseline experimental on-off ratio of $r = 1.8$) is shown in the Fig. 13. It can be noticed that the test accuracy is largely independent of the on-off ratio of the PCMO devices when the programming noise is kept fixed. This study illustrates the need for synaptic devices with minimum programming variability rather than high on-off ratio. Since programming variability is inherent to most nanoscale memristive devices, algorithm level optimizations are necessary to obtain better performance.

8. Inference in the presence of noise

One of the advantages of systems based on stochastic encoding is its inherent tolerance to noise and lesser vulnerability to variability than their deterministic counterparts [51,52]. Here, we study the stability of stochastic inference engines by observing the network response to noise-corrupted test data. The stochastic DNNs are trained with $BL = 10$ using synaptic devices and these trained weights corresponding to the best generalization accuracy is then used for inference analysis [40]. Zero mean Gaussian noise of variances $\sigma_i^2 = 0, 0.01, 0.1$ is added to the normalized input images in the MNIST test set. The noise added input is then limited in the range [0,1] for evaluating the robustness of stochastic inference engines. A noise corrupted sample image with its corresponding signal to noise ratio (SNR in dB) is shown in the Fig. 14. Noise resilience for inference is studied for PCMO devices with programming variability ratios of σ/B = 0.5, 1, 1.5. Two variants of stochastic inference engines are used in our study, one with



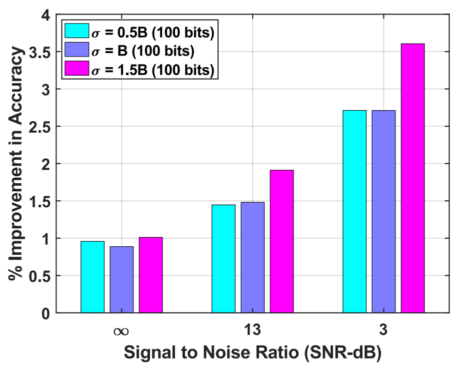


Fig. 15. Study of stochastic inference engines to noise corrupted test data using trained weights as a function of the programming variability of the device ($\sigma/B = 0.5, 1$ and 1.5) (left); Percentage improvement in test accuracy of stochastic inference engines with $BL = 100$ when compared to $BL = 10$ using devices with different conductance variability. Using inference engines with $BL = 100$, a remarkable improvement in test accuracy can be noticed for devices with higher programming variability (right).

$BL = 10$ and the other $BL = 100$ even though training is done using $BL = 10$.

The inference response of PCMO based synapses with different programming variability to noise added test set is shown in Fig. 15. The percentage improvement in test accuracy for stochastic inference engines with $BL = 100$ over $BL = 10$ is shown in Fig. 15. It is observed that as the programming variability increases, stochastic inference engines with $BL = 100$ has a superior performance compared to $BL = 10$. This study shows that even though the expensive network training is performed using as few as 10 bits, the inference accuracy of the network can be improved, especially when dealing with noisy inputs, just by increasing the bit resolution used during the relatively in-expensive forward pass operations necessary for inference.

9. Conclusion

In this paper, we discuss the need for non-von Neumann architecture and a hardware based approach for implementing neuromorphic systems with memristive synapses at the crossbar. DNNs can be trained with a time complexity of $\mathcal{O}(1)$ using RPU devices at the crossbar by representing the real-valued neuronal output and feedback terms as stochastic bit-streams. We used a non-filamentary $Pr_{0.7}Ca_{0.3}MnO_3$ device to understand the device parameters that are critical to the performance of stochastic DNNs. These devices were fabricated using a standard lithography process and electrically characterized to determine the on-off ratio, bit-resolution and programming variability. Using these measured characteristics, we studied the network performance as a function of the device parameters such as on-off ratio, frequency of conductance reset to account for the limited dynamic range and programming variability. We showed that programming variability is paramount to the network performance and demonstrated that if the conductance variability is kept minimum, network test accuracies close to 95% is obtainable using PCMO based devices. Since programming variability is inherent to most nanoscale devices, algorithms that could mitigate these non-ideal characteristics have to be developed for enhanced network performance. We also demonstrated approaches to improve network inference accuracy without incurring significant costs for realizing on-chip learning, especially for noisy real-world inputs.

Acknowledgment

This research was supported in part by the National Science Foundation grant 1710009 and CISCO Systems Inc. The authors, Sandip Lashkare and Udayan Ganguly would like to thank Nano Mission, Department of Science and Technology and the Department of Electronics and IT, Government of India for partially funding this work. Sandip Lashkare is supported by Intel India Ph.D. Fellowship. The authors would like to gratefully acknowledge the comments of the anonymous reviewers which helped improve the paper.

References

[1] J. von Neumann, First draft of a report on the EDVAC, IEEE Ann. Hist. Comput. 15 (4) (1993) 27–75, doi:10.1109/85.238389.
[2] M. Di Ventra, Y.V. Pershin, The parallel approach, Nat. Phys. 9 (4) (2013) 200–202, doi:10.1038/nphys2566.
[3] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, K. Yelick, A case for intelligent RAM, Micro IEEE 17 (2) (1997) 34–44, doi:10.1109/40.592312.
[4] G.W. Burr, B.N. Kurdi, J.C. Scott, C.H. Lam, K. Gopalakrishnan, R.S. Shenoy, Overview of candidate device technologies for storage-class memory, IBM J. Res. Dev. 52 (4.5) (2008) 449–464, doi:10.1147/rd.524.0449.
[5] W.S. McCulloch, W. Pitts, in: Neurocomputing: Foundations of Research, MIT Press, Cambridge, MA, USA, 1988, pp. 15–27. http://dl.acm.org/citation.cfm?id=65669.104377.
[6] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain., Psychol. Rev. 65 (6) (1958) 386–408, doi:10.1037/h0042519.
[7] W. Maas, Networks of spiking neurons: the third generation of neural network models, Trans. Soc. Comput. Simul. Int. 14 (4) (1997) 1659–1671. http://dl.acm.org/citation.cfm?id=281543.281637.
[8] A. Coates, B. Huval, T. Wang, D.J. Wu, A.Y. Ng, B. Catanzaro, Deep learning with COTS HPC systems, in: Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28'13, JMLR.org, 2013. http://dl.acm.org/citation.cfm?id=3042817.3043086.
[9] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, O. Temam, DaDianNao:a machine-learning supercomputer, in: Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-47, IEEE Computer Society, Washington, DC, USA, 2014, pp. 609–622, doi:10.1109/MICRO.2014.58.
[10] B. Rajendran, Y. Liu, J.S. Seo, K. Gopalakrishnan, L. Chang, D.J. Friedman, M.B. Ritter, Specifications of nanoscale devices and circuits for neuromorphic computational systems, IEEE Trans. Electron. Dev. 60 (1) (2013) 246–253, doi:10.1109/TED.2012.2227969.
[11] D. Kuzum, S. Yu, H.-S.P. Wong, Synaptic electronics: materials, devices and applications, Nanotechnology 24 (38) (2013) 382001. http://stacks.iop.org/0957-4484/24/i=38/a=382001.
[12] S.H. Jo, T. Chang, I. Ebong, B.B. Bhadviya, P. Mazumder, W. Lu, Nanoscale memristor device as synapse in neuromorphic systems, Nano Lett. 10 (4) (2010) 1297–1301, doi:10.1021/nl904092h.
[13] B.L. Jackson, B. Rajendran, G.S. Corrado, M. Breitwisch, G.W. Burr, R. Cheek, K. Gopalakrishnan, S. Raoux, C.T. Rettner, A. Padilla, A.G. Schrott, R.S. Shenoy, B.N. Kurdi, C.H. Lam, D.S. Modha, Nanoscale electronic synapses using phase change devices, J. Emerg. Technol. Comput. Syst. 9 (2) (2013) 12:1–12:20, doi:10.1145/2463585.2463588.
[14] G. Snider, Instar and outstar learning with memristive nanodevices, Nanotechnology 22 (1) (2011) 015201. http://stacks.iop.org/0957-4484/22/i=1/a=015201.
[15] G. Indiveri, B. Linares-Barranco, R. Legenstein, G. Deligeorgis, T. Prodromakis, Integration of nanoscale memristor synapses in neuromorphic computing architectures, Nanotechnology 24 (38) (2013) 384010. http://stacks.iop.org/0957-4484/24/i=38/a=384010.
[16] G.W. Burr, P. Narayanan, R.M. Shelby, S. Sidler, I. Boybat, C.d. Nolfo, Y. Leblebici, Large-scale neural networks implemented with non-volatile memory as the synaptic weight element: comparative performance analysis (accuracy, speed, and power), in: Proceedings of the 2015 IEEE international electron devices meeting (IEDM), 2015, pp. 4.4.1–4.4.4, doi:10.1109/IEDM.2015.7409625.
[17] 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.L. Gallo, K. Moon, J. Woo, H. Hwang, Y. Leblebici, Neuromorphic computing using non-volatile memory, Adv. Phys.: X 2 (1) (2017) 89–124, doi:10.1080/23746149.2016.1259585.
[18] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, H.S.P. Wong, A low energy oxide-based electronic synaptic device for neuromorphic visual systems with tolerance to device variation, Adv. Mater. 25 (12) (2013) 1774–1779, doi:10.1002/adma.201203680.
[19] T. Gokmen, Y. Vlasov, Acceleration of deep neural network training with resistive cross-point devices: design considerations, Front. Neurosci. 10 (2016) 333, doi:10.3389/fnins.2016.00333.
[20] J.W. Jang, S. Park, G.W. Burr, H. Hwang, Y.H. Jeong, Optimization of conductance change in Pr_{1-x} Ca_xMnO_3-based synaptic devices for neuromorphic systems, IEEE Electron. Dev. Lett. 36 (5) (2015) 457–459, doi:10.1109/LED.2015.2418342.
[21] P.Y. Chen, B. Lin, I.T. Wang, T.H. Hou, J. Ye, S. Vrudhula, J.s. Seo, Y. Cao, S. Yu, Mitigating effects of non-ideal synaptic device characteristics for on-chip learning, in: Proceedings of the 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2015, pp. 194–199, doi:10.1109/ICCAD.2015.7372570.
[22] A. Fumarola, P. Narayanan, L.L. Sanches, S. Sidler, J. Jang, K. Moon, R.M. Shelby, H. Hwang, G.W. Burr, Accelerating machine learning with Non-volatile memory: exploring device and circuit tradeoffs, in: Proceedings of the 2016 IEEE International Conference on Rebooting Computing (ICRC), 2016, pp. 1–8, doi:10.1109/ICRC.2016.7738684.
[23] S. Sidler, I. Boybat, R.M. Shelby, P. Narayanan, J. Jang, A. Fumarola, K. Moon, Y. Leblebici, H. Hwang, G.W. Burr, Large-scale neural networks implemented with non-volatile memory as the synaptic weight element: impact of conductance response, in: Proceedings of the 2016 46th European Solid-State Device Research Conference (ESSDERC), 2016, pp. 440–443, doi:10.1109/ESSDERC.2016.7599680.
[24] W.J. Poppelbaum, C. Afuso, J.W. Esch, Stochastic computing elements and systems, in: Proceedings of the Fall Joint Computer Conference November 14–16, 1967,AFIPS '67 (Fall), ACM, New York, NY, USA, 1967, pp. 635–644, doi:10.1145/1465611.1465696.
[25] B.R. Gaines, Springer US, Boston, MA, 1969, pp. 37–172, doi:10.1007/978-1-4899-5841-9_2.
[26] S. Gupta, V. Sindhwani, K. Gopalakrishnan, Learning Machines Implemented on Non-Deterministic Hardware, CoRR abs/1409.2620(2014). arxiv.org/abs/1409.2620.
[27] A. Alaghi, J.P. Hayes, Fast and accurate computation using stochastic circuits, in: Proceedings of the 2014 Design, Automation Test in Europe Conference Exhibition (DATE), 2014, pp. 1–4, doi:10.7873/DATE.2014.089.

[28] A. Alaghi, J.P. Hayes, On the functions realized by stochastic computing circuits, in: Proceedings of the 25th Edition on Great Lakes Symposium on VLSI, GLSVLSI '15, ACM, New York, NY, USA, 2015, pp. 331–336, doi:10.1145/2742060.2743758.
[29] S. Gupta, K. Gopalakrishnan, in: Revisiting Stochastic Computation: Approximate Estimation of Machine Learning Kernels, First Workshop on Approximate Computing Across the System Stack 1–2,, 2014.
[30] A. Alaghi, C. Li, J.P. Hayes, Stochastic circuits for real-time image-processing applications, in: Proceedings of the 50th Annual Design Automation Conference, DAC '13, ACM, New York, NY, USA, 2013, pp. 136:1–136:6, doi:10.1145/2463209.2488901.
[31] Q.T. Dong, M. Arzel, C. Jego, W.J. Gross, Stochastic decoding of turbo codes, IEEE Trans. Signal Process. 58 (12) (2010) 6421–6425, doi:10.1109/TSP.2010.2072924.
[32] M.L. Alomar, V. Canals, V. Martínez-Moll, J.L. Rosselló, Springer International Publishing, Cham, 2015, pp. 185–196, doi:10.1007/978-3-319-19222-2_16.
[33] J. Li, A. Ren, Z. Li, C. Ding, B. Yuan, Q. Qiu, Y. Wang, Towards acceleration of deep convolutional neural networks using stochastic computing, in: Proceedings of the 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC), 2017, pp. 115–120, doi:10.1109/ASPDAC.2017.7858306.
[34] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (1986). 533 EP – 10.1038/323533a0.
[35] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, L. Fei-Fei, Large-scale video classification with convolutional neural networks, in: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '14, IEEE Computer Society, Washington, DC, USA, 2014, pp. 1725–1732, doi:10.1109/CVPR.2014.223.
[36] K. Steinbuch, Die lernmatrix, Kybernetik 1 (1) (1961) 36–45, doi:10.1007/BF00293853.
[37] D. Niu, C. Xu, N. Muralimanohar, N.P. Jouppi, Y. Xie, Design trade-offs for high density cross-point resistive memory, in: Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design, ISLPED '12, ACM, New York, NY, USA, 2012, pp. 209–214, doi:10.1145/2333660.2333712.
[38] Y. Wang, B. Li, R. Luo, Y. Chen, N. Xu, H. Yang, Energy efficient neural networks for big data analytics, in: Proceedings of the Conference on Design, Automation & Test in Europe, DATE '14, European Design and Automation Association, 3001 Leuven, Belgium, Belgium, 2014, pp. 345:1–345:2. http://dl.acm.org/citation.cfm?id=2616606.2617095.
[39] J.J. Yang, D.B. Strukov, D. Stewart R, Memristive devices for computing, Nat. Nanotechnol. 9 (2013) 13–24. 10.1038/nnano.2012.240.
[40] A.V. Babu, B. Rajendran, Stochastic deep learning in memristive networks, in: Proceedings of the 2017 24th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2017, pp. 214–217, doi:10.1109/ICECS.2017.8292067.
[41] Y. LeCun, L. Bottou, G.B. Orr, K.-R. Müller, Effiicient BackProp, in: Proceedings of the Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop, Springer-Verlag, London, UK, UK, 1998, pp. 9–50. http://dl.acm.org/citation.cfm?id=645754.668382.
[42] N. Panwar, U. Ganguly, Variability assessment and mitigation by predictive programming in $Pr_{0.7}Ca_{0.3}MnO_3$ based RRAM, in: Proceedings of the 2015 73rd Annual Device Research Conference (DRC), 2015, pp. 141–142, doi:10.1109/DRC.2015.7175595.
[43] D.J. Seong, M. Hassan, H. Choi, J. Lee, J. Yoon, J.B. Park, W. Lee, M.S. Oh, H. Hwang, Resistive-switching characteristics of Al/ $Pr_{0.7}Ca_{0.3}MnO_3$ for nonvolatile memory applications, IEEE Electron. Dev. Lett. 30 (9) (2009) 919–921, doi:10.1109/LED.2009.2025896.
[44] P. Kumbhare, U. Ganguly, Ionic transport barrier tuning by composition in $Pr_{0.7}Ca_{0.3}MnO_3$-based selector-less RRAM and its effect on memory performance, IEEE Trans. Electron. Dev. 65 (6) (2018) 2479–2484, doi:10.1109/TED.2018.2827420.
[45] N. Panwar, D. Kumar, N.K. Upadhyay, P. Arya, U. Ganguly, B. Rajendran, Memristive synaptic plasticity in $Pr_{0.7}Ca_{0.3}MnO_3$ RRAM by bio-mimetic programming, in: Proceedings of the 72nd Device Research Conference, 2014, pp. 135–136, doi:10.1109/DRC.2014.6872334.
[46] P. Kumbhare, I. Chakraborty, A. Khanna, U. Ganguly, Memory performance of a simple $Pr_{0.7}Ca_{0.3}MnO_3$-based selectorless RRAM, IEEE Trans. Electron Dev. 64 (9) (2017) 3967–3970, doi:10.1109/TED.2017.2725900.
[47] O. Bichler, M. Suri, D. Querlioz, D. Vuillaume, B. DeSalvo, C. Gamrat, Visual efficient 2-PCM synapse neuromorphic architecture, IEEE Trans. Electron. Dev. 59 (8) (2012) 2206–2214, doi:10.1109/TED.2012.2197951.
[48] S. Lim, J.H. Bae, J.H. Eum, S. Lee, C.H. Kim, D. Kwon, B.G. Park, J.H. Lee, Adaptive Learning Rule for Hardware-based Deep Neural Networks Using Electronic Synapse Devices, ArXiv e-prints (2017).
[49] G.W. Burr, R.M. Shelby, C. di Nolfo, J.W. Jang, R.S. Shenoy, P. Narayanan, K. Virwani, E.U. Giacometti, B. Kurdi, H. Hwang, Experimental demonstration and tolerancing of a large-scale neural network (165,000 synapses), using phase-change memory as the synaptic weight element, in: Proceedings of the IEEE International Electron Devices Meeting (IEDM) 2014, 2014, pp. 29.5.1–29.5.4, doi:10.1109/IEDM.2014.7047135.
[50] I. Boybat, M.L. Gallo, S.R. Nandakumar, T. Moraitis, T.P. Parnell, T. Tuma, B. Rajendran, Y. Leblebici, A. Sebastian, E. Eleftheriou, Neuromorphic computing with multi-memristive synapses, CoRR abs/1711.06507 (2017).
[51] B. Moons, M. Verhelst, Energy-efficiency and accuracy of stochastic computing circuits in emerging technologies, IEEE J. Emerg. Sel. Top. Circuits Syst. 4 (4) (2014) 475–486, doi:10.1109/JETCAS.2014.2361070.
[52] W. Qian, X. Li, M.D. Riedel, K. Bazargan, D.J. Lilja, An architecture for fault-tolerant computation with stochastic logic, IEEE Trans. Comput. 60 (1) (2011) 93–105, doi:10.1109/TC.2010.202.

Anakha V Babu received the B.Tech degree in electronics and communication engineering from University of Kerala, India, in 2010 and M.tech degree in Microelectronics & VLSI design from the Indian Institute of Technology Bombay, Mumbai, India, in 2016. She is currently pursuing Ph.D degree in electrical engineering at the New Jersey Institute of Technology, New Jersey, USA. Her current research interests include bio-inspired computing using novel memristive devices.

Sandip Lashkare hails from Nanded, Maharashtra, India. He has received B.Tech from Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded in 2011 and M.Tech degree in electrical engineering from IIT Bombay in 2013. He worked in LSI Pvt Ltd (Now Broadcom Limited) and SILABTECH Pvt. Ltd. from 2013–2015 focussing on Post Silicon and FPGA Validation of SERDES IP's. Presently, he is pursuing the Ph.D. degree at IIT Bombay. He has authored/co-authored 7 conference and 3 journal papers. His current research interests include electrical characterization, modeling and development of brain-inspired computing using novel memories.

Udayan Ganguly received the B.Tech. degree in Metallurgical Engineering from the IIT Madras, in 2000 and the M.S. and Ph.D. degrees in Materials Science and Engineering at Cornell University, Ithaca, NY, in 2005 and 2006 respectively. In 2006, after a short Postdoctoral Scholarship position at NASA Ames Research, Udayan joined Applied Materials to serve as the technical lead for Flash Memory Applications Development at Applied Materials Front End Product Division, Sunnyvale, CA. He has joined Dept. of Electrical Engineering in 2010. He has authored/co-authored 30+ journal, 60+ conference and 20 patents (applied/granted). His research interests are in semiconductor device physics and processing technologies for advanced memory, computing, and neuromorphic systems.

Bipin Rajendran received a B. Tech degree from I.I.T. Kharagpur in 2000, and M.S. and Ph.D. degrees in Electrical Engineering from Stanford University in 2003 and 2006, respectively. He was a Master Inventor and Research Staff Member at IBM T. J. Watson Research Center in New York during 2006-'12 and a faculty member in the Electrical Engineering Department at I.I.T. Bombay during 2012-'15. His research focuses on building algorithms, devices and systems for brain-inspired computing. He has authored over 70 papers in peer-reviewed journals and conferences, and has been issued 55 U.S. patents. He is currently an Associate Professor of Electrical & Computer Engineering at New Jersey Institute of Technology.