8-2017

On-Chip Training of Memristor Crossbar Based Multi-Layer Neural Networks

Raqibul Hasan

Tarek M. Taha

Christopher Yakopcic

Microelectronics Journal 66 (2017) 31–40

Contents lists available at ScienceDirect

Microelectronics Journal

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

On-chip training of memristor crossbar based multi-layer neural networks

Raqibul Hasan*, Tarek M. Taha, Chris Yakopcic

Department of Electrical and Computer Engineering, University of Dayton, OH, USA

ARTICLE INFO

Keywords:
Neural networks
Memristor crossbars
Training
On-chip training

ABSTRACT

Memristor crossbar arrays carry out multiply-add operations in parallel in the analog domain, and so can enable neuromorphic systems with high throughput at low energy and area consumption. On-chip training of these systems have the significant advantage of being able to get around device variability and faults. This paper presents on-chip training circuits for multi-layer neural networks implemented using a single crossbar per layer and two memristors per synapse. Using two memristors per synapse provides double the synaptic weight precision when compared to a design that uses only one memristor per synapse. Proposed on-chip training system utilizes the back propagation (BP) algorithm for synaptic weight update. Due to the use of two memristors per synapse, we utilize a novel technique for error back propagation. We evaluated the training of the system with some nonlinearly separable datasets through detailed SPICE simulations which take crossbar wire resistance and sneak-paths into consideration. Our results show that in the proposed design, the crossbars consume about 9× less power than single memristor per synapse design.

1. Introduction

Specialized neural network based processing systems have significant advantages to offer, such as the ability to provide high throughput while consuming very little power and area [1,2]. This is important as reliability and power consumption are among the main obstacles for continued performance improvement in future computing systems. These systems have wide applications in the areas including signal processing and pattern recognition.

Memristors [3,4] have received significant attention as a potential building block for neuromorphic systems [5,6]. In these systems memristors are used in a crossbar structure. Memristor devices in a crossbar structure can evaluate many multiply-add operations in parallel in the analog domain very efficiently (these are the dominant operations in neural networks). This enables highly dense neuromorphic system with great computational efficiency [1].

It is necessary to have an efficient training system for memristor neural network based systems. Two approaches for training are off-chip training and on-chip training. The key benefit of off-chip training is that any training algorithm can be implemented in software and run on powerful computer clusters. Memristor crossbars are prone to device variations and faults [7,8]. These variations can occur between individual devices within a crossbar and in crossbars between different chips. These are difficult to model in software, thus making off-chip training challenging for these analog circuits. On-chip training has the advantage that it can take into account variations between devices and can use the full analog range of the device (as opposed to a set of discrete resistances that off-chip training will likely need to target).

This paper presents circuits for on-chip training of memristor crossbars that utilize two memristors per synapse. Most recent memristor crossbar circuit fabrications for neuromorphic computing have been using two memristors per synapse [9,10]. Use of two memristors per synapse provides double the synaptic weight precision when compared to a design that uses only one memristor per synapse. This can enable better training of the neural networks [11] consuming relatively less power. We design a novel circuit for error back propagation which is utilized for back propagation algorithm based training.

Existing works regarding on-chip training circuits for memristor crossbars include [12,13]. Soudry et al. [12] examined on-chip gradient descent based training of memristor crossbars with a single memristor per synapse. They do not consider the training of systems with two memristors per synapse. Boxun et al. [13] examined on-chip training of crossbar systems with two memristors per synapse. But they utilized a pair of crossbars per layer, with one for the forward pass, and the second for the backward pass. In their design, the second crossbar needs to be an exact transposed copy of the first. The key limitation of this design is that the variability and stochastic switching characteristics of memristors would make it difficult to create an exact copy of a memristor crossbar without a complex feedback write mechanism.

The rest of the paper is organized as follows: Section 2 describes related work in the area. Section 3 describes memristor devices and our

* Corresponding author.
E-mail addresses: hasanm1@udayton.edu (R. Hasan), tarek.taha@udayton.edu (T.M. Taha), cyakopcic1@udayton.edu (C. Yakopcic).

http://dx.doi.org/10.1016/j.mejo.2017.05.005
Received 7 June 2016; Received in revised form 22 March 2017; Accepted 11 May 2017
Available online 06 June 2017

memristor crossbar based neuron circuit design. Section 4 describes the proposed memristor based training approach. Sections 5 and 6 demonstrate the experimental setup and results respectively. Finally Section 7 concludes the paper.

2. Related work

2.1. Transistor based systems

Specialized architectures can significantly reduce the power consumption for neural network applications and provide high performance [14–17]. Several research efforts examined neural accelerators consisting of SRAM based synaptic arrays [2,11,15–17]. These systems utilize ex-situ (off-chip) training of the neural networks. IBM's TrueNorth chip [5] consists of 5.4 billion transistors. It has 4096 neurosynaptic cores interconnected via an intra-chip network that integrates one million programmable spiking neurons and 256 million configurable synapses. The basic building block is a core, a self-contained neural network with 256 input lines (axons), and 256 outputs (neurons) connected via 256×256 directed, programmable synaptic connections. DaDianNao [6] is an accelerator for deep neural network (DNN) and convolutional neural network (CNN). This is a digital system and neuron synaptic weights are stores in an on-chip eDRAM.

2.2. Memristor based learning systems

Zamarreño-Ramos et al. [18] examined how a memristor grid can implement a highly dense spiking neural network and used it for visual image processing. They examined STDP training to implement spiking neural networks. Several research efforts examined memristor based linear separator design and training [9,19–21]. Nonlinearly separable problems were not studied in these works.

Memristor bridge circuits have been proposed [22,23] where small groups of memristors (either 4 or 5) are used to store a synaptic weight. One of the advantages of these bridge circuits is that either a positive or negative weight can be stored based on the sensed voltage. Adhikari et al. [24] examined multi-layer neural network training using memristor bridge circuits. They utilized random weight update rule which does not require error back propagation. Their results showed that the training using random weight update rule converges more slowly than the training using the BP algorithm.

Soudry et al. [12] proposed gradient descent based learning on a memristor crossbar neural network. This system utilizes two transistors and one memristor per synapse. Synaptic weight precision of the proposed implementations are two time more than this design. Moreover, proposed design is more power efficient compared to a design based on single memristor per synapse.

Work in [13] proposed using two crossbars for the same weight values. One would be used to propagate forward and another transposed version would be used to propagate errors backward. Since the switching characteristic of a memristor often contains some degree of noise, it would be difficult to store an exact copy of a memristor crossbar without a complex feedback write mechanism. Our proposed work is able to apply a variable pulse width during the weight update which is not possible in the system in [13].

Training a multi-layer neural network requires the output layer error to be back propagated to the hidden layer neurons. Research efforts [25,26] examined training of multi-layer neural networks using a training algorithm named “Manhattan Rule”. They did not detail the error back propagation step and design of the training pulse generation circuitry. Table 1 summarizes the contributions and drawbacks of the proposed work and the related works.

3. Memristor crossbar based neuron circuit and linear separator design

3.1. Memristor devices

The memristor device was first theorized in 1971 by Dr. Leon Chua [3]. Several research groups have demonstrated memristive behavior using several different materials. One such device, composed of layers of HfO_x and AlO_x [7], has a high on state resistance ($R_{ON} \approx 50$ kΩ) and a very high resistance ratio ($R_{OFF}/R_{ON} \approx 1000$). In general, a certain energy (or threshold voltage, V_{th}) is required to enable the state change in a memristive device [7,27]. When the electrical excitation through a memristor exceeds the threshold, i.e., $V(t) > V_{th}$, the resistance of the device changes. Otherwise, a memristor behaves like a resistor. The device characterized in [7] has a threshold voltage of about 1.3 V. Physical memristors can be laid out in a high density grid known as a crossbar. The schematic and layout of a memristor crossbar can each be seen in Fig. 1. A memristor in the crossbar structure occupies $4F^2$ area (where F is the device feature size). This is 36 times smaller than a SRAM memory cell. A memristor crossbar can perform many multiply-add operations in parallel and the conductance of multiple memristors in a crossbar can be updated in parallel [12]. Multiply-add operations are the dominant operations in neural networks and training of neural networks require update of synaptic weights iteratively. As a consequence, memristors have a great potential as a synaptic element in a neural network based system design.

3.2. A neuron in a neural network

Fig. 2 shows a block diagram of a neuron. A neuron in a neural network performs two types of operations, (i) a dot product of the inputs $x_1,\ldots,x_n$ and the weights $w_1,\ldots,w_n$, and (ii) the evaluation of an activation function. The dot product operation can be seen in Eq. (1). The activation function of the neuron is shown in Eq. (2). In a multi-layer feed forward neural network, a nonlinear differentiable activation function is desired (e.g. $tan^{-1}(x)$).

$$DP_j = \sum_{i=1}^{n} x_i w_{ij} \tag{1}$$

$$y_j = f(DP_j) \tag{2}$$

3.3. Neuron circuit

Fig. 3(a) shows a memristor based neuron circuit having three inputs and one bias input (β). A synapse in the circuit is represented by a pair of memristors. In this circuit, each data input is connected to two virtually grounded op-amps (operational amplifiers) through a pair of memristors. For a given row, if the conductance of a memristor connected to the first column (σ_A^+) is higher than the conductance of the memristor connected to the second column (σ_A^-), then the pair of memristors represents a positive synaptic weight. In the inverse situation, when $\sigma_A^+ < \sigma_A^-$, the memristor pair represents a negative synaptic weight.

In Fig. 3(a) currents through the first and second columns are $A\sigma_{A+}+\ldots+\beta\sigma_{\beta+}$ and $A\sigma_{A-}+\ldots+\beta\sigma_{\beta-}$ respectively. The output of the op-amp, connected directly with the second column, represents the neuron output. In the non-saturating region of the second op-amp, the output y_j of the neuron circuit is given by

$$y_j = R_f[\{A\sigma_{A+}+\ldots+\beta\sigma_{\beta+}\}-\{A\sigma_{A-}+\ldots+\beta\sigma_{\beta-}\}]$$

$$= R_f[A(\sigma_{A+}-\sigma_{A-})+\ldots+\beta(\sigma_{\beta+}-\sigma_{\beta-})]$$

Assume that

$$DP_j = 4R_f[A(\sigma_{A+}-\sigma_{A-})+\ldots+\beta(\sigma_{\beta+}-\sigma_{\beta-})]$$

(here $4R_f$ is a constant).

Table 1
Major contributions and drawbacks of the proposed work and related works.

	Contributions/drawbacks
Proposed work	– Nonlinear classifier training using BP algorithm. – Used two memristors/synapse for higher precision of weights. – Designed a novel circuit for error back propagation which utilize the same crossbar as used in the forward pass. – Demonstrated the power benefit over a single memristor/synapse design.
[12]	BP algorithm based training using single memristor, two transistors per synapse.
[13]	Used a variant of the BP algorithm and two memristor crossbars for each layer.
[9,19–21]	Examined only training of linear separators. Training of multilayer neural network was not examined.
[24]	Used random weight update rule which does not require error back propagation. Converges more slowly than the training using the BP algorithm.
[18]	Spiking neural network, used STDP learning rule.
[25,26]	Manhattan Rule based training. Did not examine the error back propagation step and the design of the training pulse generation circuitry.
[5,6]	Digital, only for recognition task.

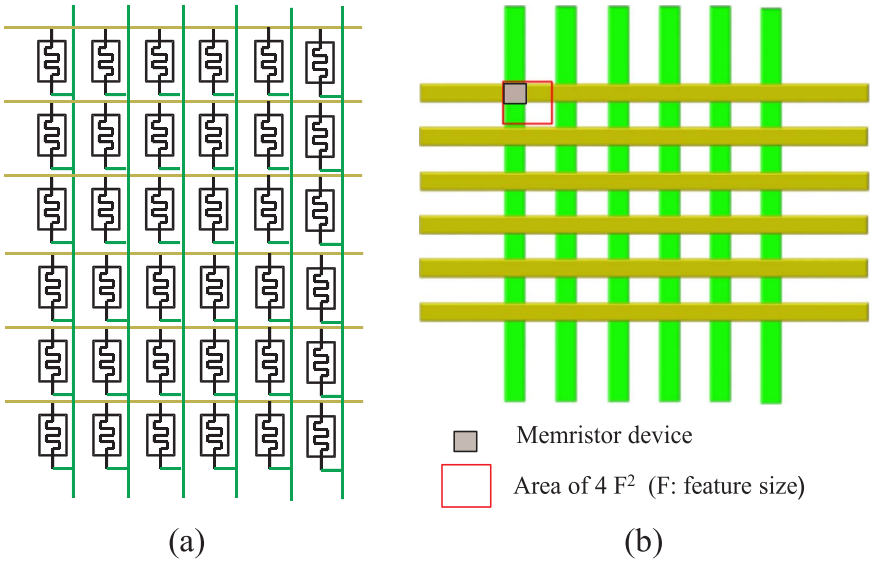

Fig. 1. (a) Memristor crossbar schematic and (b) memristor crossbar layout.

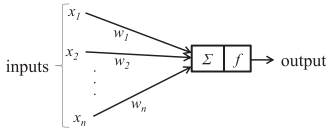

Fig. 2. Neuron block diagram.

When the power rails of the op-amps, V_{DD} and V_{SS} are set to 0.5 V and −0.5 V respectively, the neuron circuit implements the activation function $h(x)$ as in Eq. (3) where.

$x = 4R_f[A(\sigma_{A+} - \sigma_{A-}) + \ldots + \beta(\sigma_{\beta+} - \sigma_{\beta-})]$. This implies, the neuron output can be expressed as $h(DP_j)$.

$$h(x) = \begin{cases} 0.5 & if \quad x > 2 \\ \frac{x}{4} & if \quad |x| < 2 \\ -0.5 & if \quad x < -2 \end{cases} \tag{3}$$

Fig. 4 shows that $h(x)$ closely approximates the sigmoid activation function, $f(x) = \frac{1}{1+e^{-x}} - 0.5$. The values of V_{DD} and V_{SS} are chosen such that no memristor gets a voltage greater than V_{th} across it during evaluation. Our experimental evaluations consider memristor crossbar wire resistance. The schematic of a memristor based neuron circuit considering wire resistance is shown in Fig. 3(b).

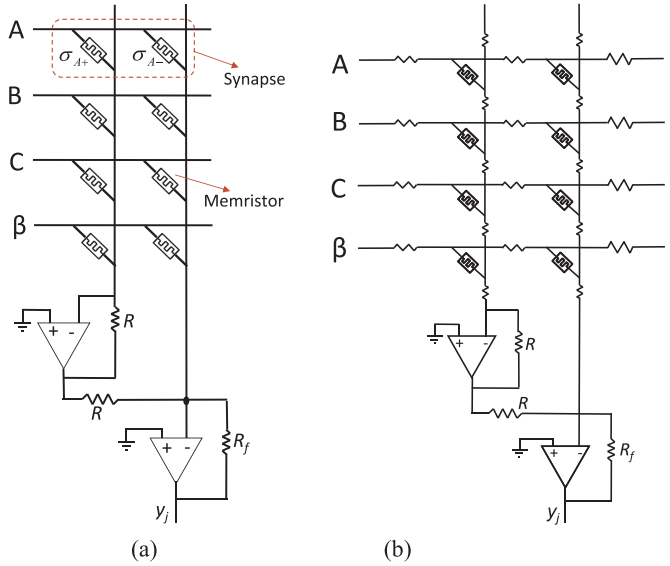

Fig. 3. Memristor-based neuron circuit. *A, B, C* are the inputs and y_j is the output.

3.4. Synaptic weight precision

The precision of memristor based synaptic weights depends on the number of memristors used for each synapse and the resistance (or conductance) range of the memristor device. Assume that the maximum conductance of the memristor device is σ_{max} and the minimum conductance is σ_{min}. For a design using only a single memristor per

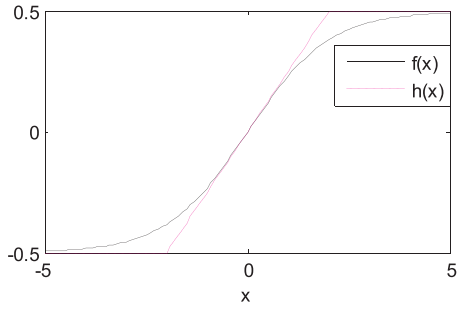

Fig. 4. Plot of functions *f(x)* and *h(x)* which show *h(x)* is approximating *f(x)* closely.

Table 2
Synaptic weight precision.

	Two memristors per synapse	One memristor per synapse
Maximum weight	σ_{max} - σ_{min}	σ_{max} - $\bar{\sigma}$
Minimum weight	σ_{min} - σ_{max}	σ_{min} - $\bar{\sigma}$
Range	$2(\sigma_{max}$ - $\sigma_{min})$	σ_{max} - σ_{min}

synapse, $\bar{\sigma}$ defines the separation between positive and negative weights [12]. Table 2 shows that the range of synaptic weights when using two memristors per synapse is two times that of a single memristor per synapse design.

3.5. Linearly separable classifier design

A neuron can be trained to work as a linear separator. This subsection demonstrates the functionality of the neuron circuit in Fig. 3 by implementing a set of linearly separable functions. We have considered the implementation of the three input Boolean functions using the proposed neuron circuit. There are 256 three input Boolean functions and among them 104 are linearly separable. It is cumbersome to examine the implementation of all 104 of these linearly separable functions. Therefore, we have implemented the 8 minterms that exist within the set of all 3 input logic functions. These functions are listed in Table 3. Each of the 8 minterms was implemented by a separate neuron and we utilized a 4×16 memristor crossbar (see Fig. 5) to implement and train all the minterms simultaneously.

Conductance of the memristors in the crossbar was randomly initialized. We have utilized the single layer perceptron learning algorithm [28] for training the memristor crossbar. The process used to apply the training pulses to the memristor crossbar is explained in Section 4, subsection D. A detailed SPICE simulation of the crossbar that considered crossbar wire resistance showed that the circuit was able to correctly classify each of the linearly separable functions. Fig. 6 shows the training graph indicating successful training. After training, each neuron in the crossbar recognizes a minterm in Table 3 exclusively.

4. Memristor crossbar based multi-layer neural network training

4.1. Multi-layer circuit design

The implementation of a nonlinear classifier requires a multi-layer neural network. Fig. 7 shows a simple two layer feed forward neural network with four inputs, four outputs, and three hidden layer neurons. Fig. 8 shows a memristor crossbar based implementation of the neural network in Fig. 7, utilizing the neuron circuit shown in Fig. 3(a). There

Table 3
8 Three input minterms used for training.

m_0 = A'B'C'	m_1 = A'B'C	m_2 = A'BC'	m_3 = A'BC
m_4 = AB'C'	m_5 = AB'C	m_6 = ABC'	m_7 = ABC

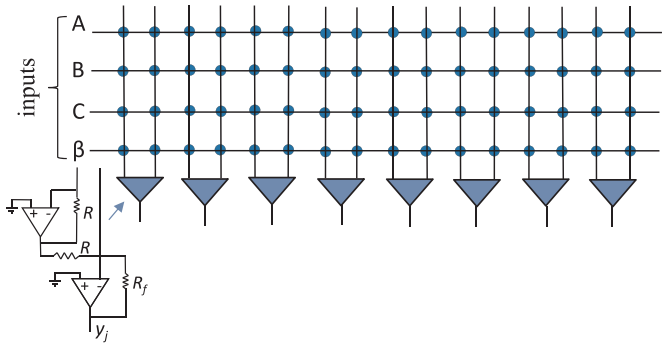

Fig. 5. Memristor crossbar implementing 8 three input minterms.

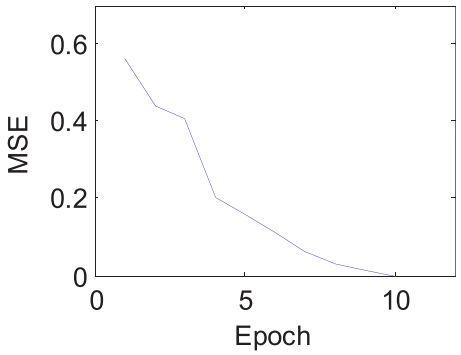

Fig. 6. Training graph of 8 three input minterms utilizing neuron circuit shown in Fig. 3(a).

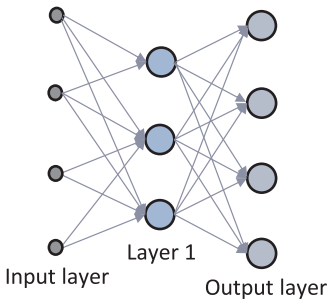

Fig. 7. Two layer neural network having four inputs, three hidden neurons and four output neurons.

are two memristor crossbars in this circuit, each representing a layer of neurons.

In Fig. 8, the first layer of neurons is implemented using a 5×6 crossbar. The second layer of two neurons is implemented using a 4×8 memristor crossbar, where 3 of the inputs are coming from the 3 neuron outputs of the first crossbar. The additional input is used as a bias. When the inputs are applied to a crossbar, the entire crossbar is processed in parallel within one cycle.

4.2. Training algorithm

In order to provide proper functionality, a multi-layer neural network needs to be trained using a training algorithm. Back propagation (BP) [29] and the variants of the BP algorithm are widely used for training such networks. The stochastic BP algorithm was used to train the memristor based multi-layer neural network which is described below:

1) Initialize the memristors with high random resistances.
2) For each input pattern *x:*

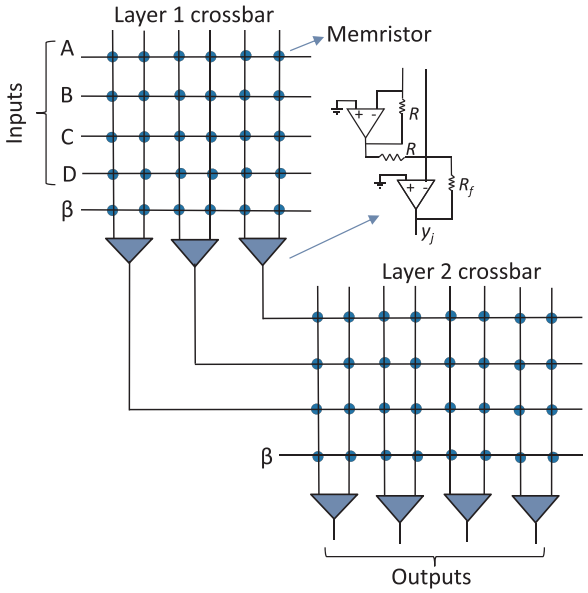

Fig. 8. Schematic of the neural network shown in Fig. 7 for forward pass utilizing neuron circuit in Fig. 3(a).

i) Apply the input pattern x to the crossbar circuit and evaluate the DP_j values and outputs (y_j) of all neurons (hidden neurons and output neurons).

ii) For each output layer neuron j, calculate the error, δ_j, based on the neuron output (y_j) and the target output (t_j). Here f is the neuron activation function and f' is the derivative of f.

$$\delta_j = (t_j - y_j) f'(DP_j) \tag{4}$$

iii) Back propagate the errors for each hidden layer neuron j.

$$\delta_j = \left(\sum_k \delta_k w_{k,j}\right) \times f'(DP_j) \tag{5}$$

where neuron k is connected to the previous layer neuron j and $w_{k,j}$ is the corresponding synaptic weight.

iv) Determine the amount, Δw, that each neuron's synapses should be changed (2η is the learning rate):

$$\Delta w_j = 2\eta \times \delta_j \times x \tag{6}$$

3) If the error in the output layer has not converged to a sufficiently small value, goto step 2.

4.3. Circuit implementation of the back propagation training algorithm

Without loss of generality we will describe the circuit implementation of the back propagation training algorithm for the neural network shown in Fig. 7. The implementation of the training circuit can be broken down into the following major steps:

1. Apply inputs to layer 1 and record the layer 2 neuron outputs and errors.
2. Back propagate layer 2 errors through the second layer weights and record the layer 1 errors.
3. Update the synaptic weights.

The circuit implementations of these steps are detailed below:

Step 1: A set of inputs is applied to the layer 1 neurons, and both layer 1, and layer 2 neurons are processed. In Eqs. (4) and (5) we need to evaluate the derivative of the activation function for the dot product of the neuron inputs and weights (DP_j). The DP_j value of neuron j is essentially the difference of the currents through the two columns implementing the neuron (this can be approximated based on y_j in Fig.. 3(a)). The DP_j value of each neuron j is discretized and $f'(DP_j)$ is evaluated using a lookup table. The $f'(DP_j)$ value of each neuron is stored in a buffer. The layer 2 neuron errors are evaluated based on the neuron outputs (y_j), the corresponding targets (t_j) and $f'(DP_j)$ as shown in Fig. 9. First (t_j-y_j) is evaluated and discretized using an ADC (analog to digital converter). Then (t_j-y_j), and $f'(DP_j)$ are multiplied using a digital multiplier and the evaluated δ_j value is stored in a register.

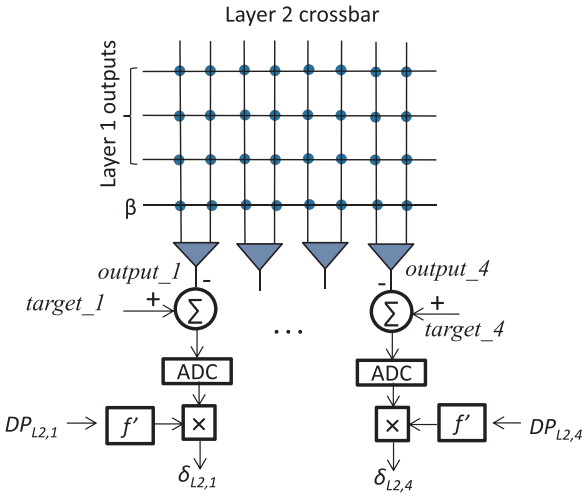

Fig. 9. Output layer error generation circuits which take neuron outputs, corresponding targets, and DP_j values as input.

Step 2: Error back propagation operation in a single memristor per synapse design is straight forward. If inputs are applied at the crossbar rows in the forward pass, error inputs are applied at the crossbar columns in the backward pass (or vice versa). Recall that the proposed system utilize two memristors per synapse and same crossbar is utilized for forward and backward passes for a layer. Our error back propagation step is significantly different form the existing on-chip training systems [12,13]. In Fig. 10 the layer 2 errors ($\delta_{L2,1}$,.., $\delta_{L2,4}$) are applied to the layer 2 weights after conversion from digital to analog form to generate the layer 1 errors ($\delta_{L1,1}$ to $\delta_{L1,3}$). The memristor crossbar in Fig. 10 is the same as the layer 2 crossbar in Fig. 8. Assume that the synaptic weight associated with input i, neuron j (second layer neuron) is $w_{ij} = \sigma_{ij}^{+} - \sigma_{ij}^{-}$ for $i=1,2,3$ and $j=1,2,..,4$. In the backward phase, we want to evaluate the layer 1 error

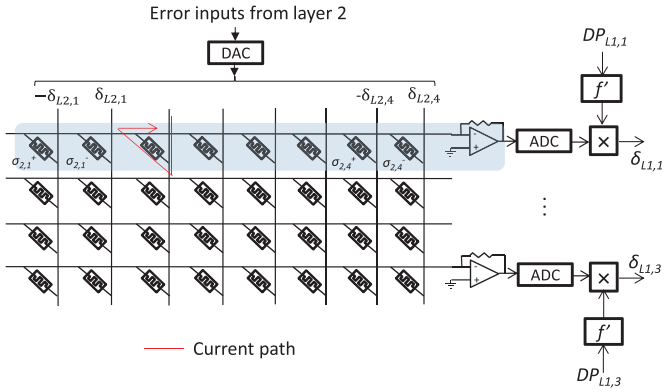

Fig. 10. Schematic of the neural network shown in Fig. 8 for back propagating errors to layer 1.

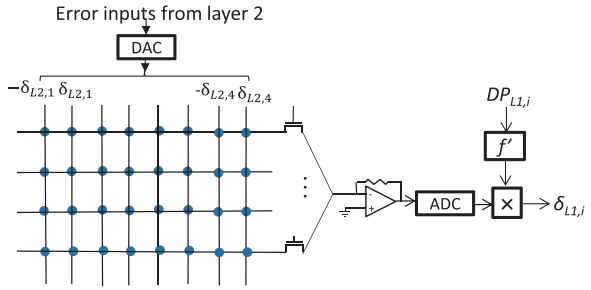

Fig. 11. Implementing back propagation phase multiplexing error generation circuit.

$$\begin{aligned}\delta_{L1,i} &= (\Sigma_j w_{ij}\delta_{L2,j})f'(DP_{L1,i}) \text{ for } i = 1, 2, 3 \text{ and } j = 1, 2, ..,4.\\ &= (\Sigma_j(\sigma_{ij}^{+} - \sigma_{ij}^{-})\delta_{L2,j})f'(DP_{L1,i})\\ &= (\Sigma_j\sigma_{ij}^{+}\delta_{L2,j} - \Sigma_j\sigma_{ij}^{-}\delta_{L2,j})f'(DP_{L1,i})\end{aligned} \quad (7)$$

The circuit in Fig. 10 is essentially evaluating the same operations as Eq. (7), applying both $\delta_{L2,j}$ and $-\delta_{L2,j}$ to the crossbar columns for $j=1,2,..,4$. The back propagated (layer 1) errors are stored in buffers for updating crossbar weights in step 3. To reduce the training circuit overhead, we can multiplex the back propagated error generation circuit as shown in Fig. 11. In this circuit, by enabling the appropriate pass transistor, back propagated errors are sequentially generated and stored in buffers. Access to the pass transistors will be controlled by a shift register. In this approach we need a single ADC for each crossbar. Same multiplexing approach could also be used for the layer 2 error generation in step 1. In this approach the time complexity of the back propagation step will be $O(m)$ where m is the number of inputs in a layer of neurons.

Step 3: The weight update procedures for both layers are similar. They take neuron inputs, and errors to generate a set of training pulses. The training unit essentially implements Eq. (6). To update a synaptic weight by an amount Δw, the conductance of the memristor connected to the first column of the corresponding neuron is updated by amount $\Delta w/2$ (where $\Delta w/2 = \eta \times \delta_j \times x_i$) and the memristor connected to the second column of the corresponding neuron by amount $-\Delta w/2$. We will describe the weight update procedure for the first columns of the neurons (odd crossbar columns in Fig. 8). The weight update procedure for the second columns of the neurons (even columns) is similar to the procedure for the first columns, except that the neuron errors (δ_j) need to be multiplied by -1.

For training, pulses of variable amplitude and variable duration are produced. The amplitude of the training signal is modulated by the neuron input (x_i) and is applied to the row wire connecting the desired memristor (see Fig. 12(a)). The duration of the training pulse is

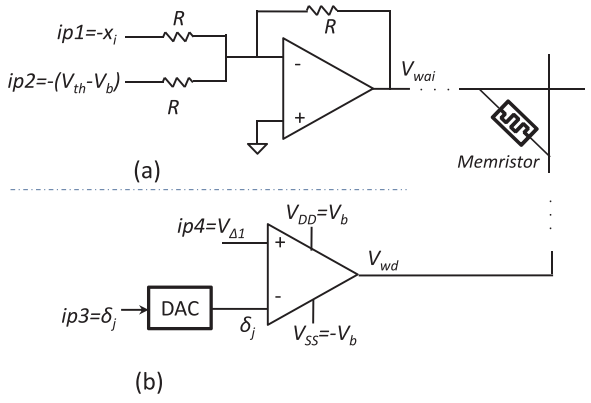

Fig. 12. Training pulse generation module. Inputs to the circuit are mentioned for the scenario shown in the first row of Table 4.

Table 4
Input to the training module for different scenarios.

Sign of δ_j	Sign of x_i	Weight update	ip1	ip2	ip3	ip4
+	+	increase	$-x_i$	$-(v_{th}-v_b)$	δ_j	$V_{\Delta 1}$
+	–	decrease	$-x_i$	$-(-v_{th}+v_b)$	$-\delta_j$	$-V_{\Delta 1}$
–	+	decrease	x_i	$-(-v_{th}+v_b)$	δ_j	$-V_{\Delta 1}$
–	–	increase	x_i	$-(v_{th}-v_b)$	$-\delta_j$	$V_{\Delta 1}$

modulated by $\eta \times \delta_j$ and is applied to the column wire connecting the desired memristor (see Fig. 12(b)). The combined effect of the two voltages applied across the memristor will update the conductance by an amount proportional to $\eta \times \delta_j \times x_i$.

Fig. 12 shows the training circuit for the case when $\delta_j > 0$ and $x_i > 0$. Table 4 shows the inputs to the training module for each of the combinations of $sign(x_i)$ and $sign(\delta_j)$. The inputs are taken such that during weight increase $V_{wai} > 0$, $V_{wd} = V_{SS}$ for the training period (so that potential across the memristor is greater than V_{th}). During weight decrease, we want to have $V_{wai} < 0$, $V_{wd} = V_{DD}$ for the training period (so that potential across the memristor is less than $-V_{th}$). The training circuit utilize a triangular wave $V_{\Delta 1}$ (see Eq. (8)) to modulate the training pulse duration based on $\eta \times \delta_j$. Duration of $V_{\Delta 1}$, $T_{\Delta 1}$ determines the learning rate. Detail description on the training circuit is given in Appendix.

$$V_{\Delta 1}(t) = \begin{cases} 1 - \frac{2t}{T_{\Delta 1}} if \;\; 0 \le t \le T_{\Delta 1}/2 \\ \frac{2t}{T_{\Delta 1}} - 1 \;\; if \;\; \frac{T_{\Delta 1}}{2} < t \le T_{\Delta 1} \\ 1 \;\; otherwise \end{cases} \quad (8)$$

4.4. Writing to memristor crossbars

Recall that the voltage across a memristor needs to surpass a threshold voltage (V_{th}) in order to change the conductance of a given memristor [14] (for the device considered $V_{th} = 1.3$ V [7]). Several studies [7,10,30] examined the change in memristor conductance for positive and negative voltage pulses for variable pulse height and pulse width (duration). They reported a strong correlation between change in conductance, the width of the applied pulse, and the applied voltage amplitude. It is reasonable to assume that a generated training pulse, determined according to the training rule, will be able to update memristor conductance values accordingly.

The physical layout of the memristor devices is assumed as in Fig. 12. When V_{wai}-$V_{wd} > V_{th}$ the conductance of the memristor is increased. Alternatively when V_{wai}-$V_{wd} < -V_{th}$ conductance of the memristor is decreased. In the weight update process, two memristors in the same row or same column of a crossbar cannot have their conductance changes in different directions (one increase, another decrease) simultaneously. As a result, the conductance of the memristors in the crossbar will be updated column by column. For a crossbar column, the conductance of the memristors will be updated in two steps: first the memristors requiring a conductivity increase will be updated, then the memristors requiring a conductivity decrease will be updated. In this process, the circuit shown in Fig. 12(a) will be required for each row of the crossbar. For an entire crossbar one circuit shown in Fig. 12(b) is required which will be used for each column one by one.

For the scenarios mentioned in the first and fourth rows of Table 4, the conductance of the memristors will be updated in the increasing phase. Fig. 13(a) shows the weight update operation on the first crossbar column in the increasing phase. In this phase, voltages V_{wa1} and V_{wa4}, produced by the circuits similar to Fig. 12(a), will be applied to the first and fourth rows respectively where we want to increase conductance. The training pulse V_{wd}, generated by the circuit in Fig. 12(b), will be applied to the column of the memristor we want to update (in this case the first column). The remaining rows and columns

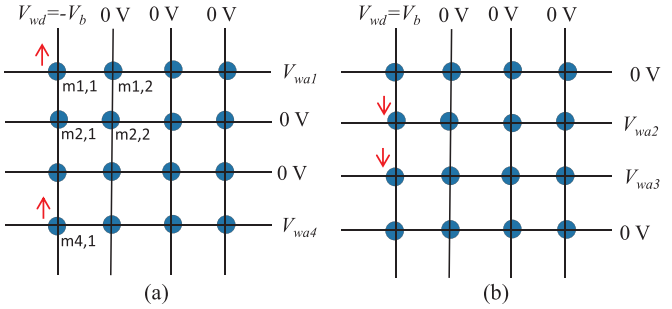

Fig. 13. Demonstration of the weight update operation in the first column of a crossbar: (a) increasing phase and (b) decreasing phase. Upward arrow indicates increase of conductance and downward arrow indicates decrease of conductance.

of the crossbar will be set to 0 V.

The potential difference across the memristors to be updated will be V_{wai}-V_{wd} for $i=1,4$. For the training duration those potential differences will be V_{th}-$V_b+|x_i|$-(-V_b) or $V_{th}+|x_i|$. As $V_{th}+|x_i| > V_{th}$ the conductance of the selected memristors will be changed. The potential difference across other memristors will be 0 V, V_b, V_{th}-$V_b+|x_i|$, or V_{th}-$2V_b+|x_i|$ for $i=1,4$. To ensure that other memristors are not changing their conductance, we need to select V_b such that $V_b < V_{th}$, V_{th}-$V_b+max\{x_i\} < V_{th}$, and V_{th}-$2V_b+max\{x_i\} > -V_{th}$. We used the value $V_b=1.2$ V. Fig. 14 shows the plot of training signals and potential difference across some memristors for this weight update step. Only the memristors *m1,1* and *m4,1* get potential across them greater than V_{th} for the training duration (2–8 ns). Except that no memristor gets potential across it greater than V_{th} in this step.

For the scenarios mentioned in the second and third rows of Table 4, the conductance of the memristors will be updated in the decreasing phase in a procedure similar to the one used for the increasing phase (see Fig. 13(b)). The complexity of the weight update operations for a layer of neurons is *O(n)* where *n* is the number of neurons in the layer.

5. Experimental setup

The memristor crossbar circuits were simulated in SPICE so that the

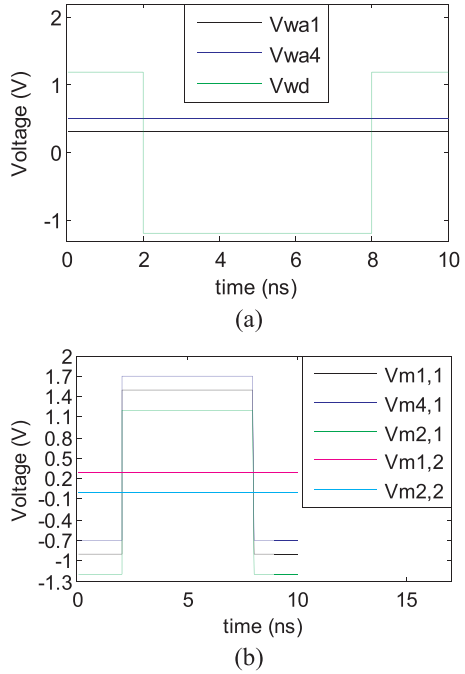

Fig. 14. (a) Training signals while $x_1=0.2$ V and $x_4=0.4$ V and (b) voltage across some memristors for the operation in Fig. 13(a). *Vmi,j* is the voltage difference across the memristor *mi,j*.

Table 5
Simulation parameters.

Memristor R_{ON}	50 kΩ
Memristor R_{OFF}	10 MΩ
Maximum read voltage, V_{read}	0.5 V
Threshold voltage	1.3 V
Memristor switching time for write voltage 2.5 V	20 μs
Crossbar each wire segment resistance	5 Ω
Maximum deviation of memristor device response due to device variation & stochasticity	30%
Value of R_f in Fig. 3(a)	14 MΩ
Learning rate	5 ns to −8 ns
ADC precision (to discretize DP_j values)	8 bits

memristor grid could be evaluated very accurately considering the crossbar sneak-paths and wire resistances. A wire resistance of 5 Ω between memristors in the crossbar is considered in these simulations. Each attribute of the input was mapped within [-V_{read}, V_{read}] voltage range. As mentioned in Section 4, duration of $V_{\Delta 1}$ determines the learning rate. Table 5 shows the simulation parameters. The resistance of the memristors in the crossbars were randomly initialized between 0.909 MΩ and 10 MΩ.

Two sets of crossbar simulations were carried out: one considering memristor device variation and stochasticity, and the other not considering these. We assumed a maximum deviation of 30% in memristor device responses due to device variations and stochasticity. That is, when we want to update a memristor conductance by amount *x*, the corresponding training pulse would update the conductance by an arbitrary value randomly taken from the interval [0.7*x*, 1.3*x*].

Simulation of the memristor device used an accurate model of the device published in [31]. The memristor device simulated in this paper was published in [7] and the switching characteristics for the model are displayed in Fig. 15. This device was chosen for its high minimum resistance value and large resistance ratio. According to the data presented in [7] this device has a minimum resistance of 50 kΩ, a resistance ratio of 10^3, and the full resistance range of the device can be switched in 20 μs by applying 2.5 V across the device.

MATLAB (R2014a) and SPICE (LTspice IV) were used to develop a simulation framework for applying the training algorithm to a multi-layer neural network circuit. SPICE was mainly used for detailed analog simulation of the memristor crossbar array and MATLAB was used to simulate the rest of the system. The circuits were trained by applying input patterns one by one until the errors were below the desired levels.

We have examined the training of the memristor crossbar arrays for five nonlinearly separable datasets: (a) 2 input XOR function, (b) 3 input odd parity function, (c) 4 input odd parity function, (d) Wine [32], and (e) Iris [33]. The Wine dataset is consisting of 178 instances (samples) belonging to three classes where each instance has 13 attributes. For this dataset, we used 118 samples for training and 60 samples for testing. The Iris dataset consists of 150 samples (99 training and 51 test samples) belonging to three classes: Iris Setosa, Iris Versicolour, and Iris Virginica. Each pattern consists of 4 attributes/features. A multi-layer neural network is required to learn a nonlinearly classifier. Table 6 shows the network configurations used in these experiments. A neural network configuration descried by $x \rightarrow y \rightarrow z$ means the network has x inputs, y hidden neurons, and z output neurons.

6. Results

Fig. 16 shows the training graphs obtained from the SPICE simulations for different datasets utilizing the training process described in Section 4. The results show that the neural networks were able to learn each classification application in both cases: without considering device variation, stochasticity and considering device variation, stochasticity.

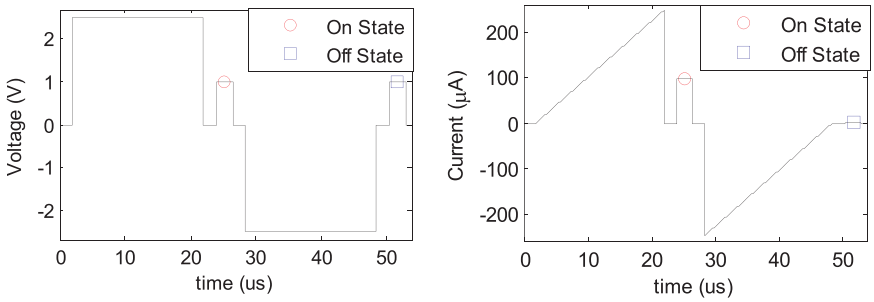

Fig. 15. Simulation results displaying the input voltage and current waveforms for the memristor model [31] that was based on the device in [7]. The following parameter values were used in the model to obtain this result: $V_p = 1.3$ V, $V_n = 1.3$ V, $A_p = 5800$, $A_n = 5800$, $x_p = 0.9995$, $x_n = 0.9995$, $\alpha_p = 3$, $\alpha_n = 3$, $a_1 = 0.002$, $a_2 = 0.002$, $b = 0.05$, $x_o = 0.001$.

Table 6
Neural network configurations.

Dataset	Neural network configurations	Number of training data
2 input XOR	2 → 5 → 1	4
3 input odd parity	3 → 7 → 1	8
4 input odd parity	4 → 12 → 1	16
Wine	13 → 20 → 3	118
Iris	4 → 15 → 3	99

Table 7
Recognition error on test data for different datasets and SPICE training approaches.

	Recognition error (%)	
	no device var.	device var.
Iris (51 test data)	2.61	3.92
Wine (60 test data)	1.34	1.93

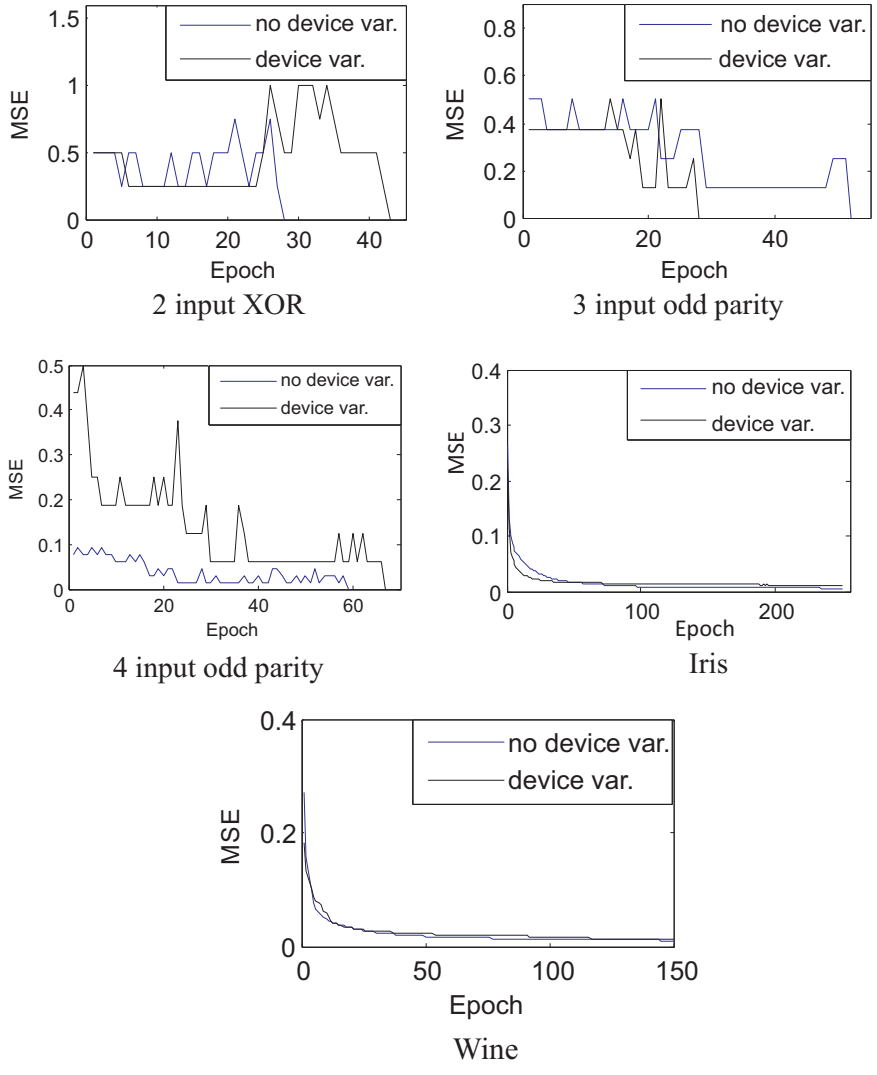

Fig. 16. SPICE training results of memristor neural networks for both cases: without considering memristor device variation, stochasticity (no device var.) and considering device variation, stochasticity (device var.).

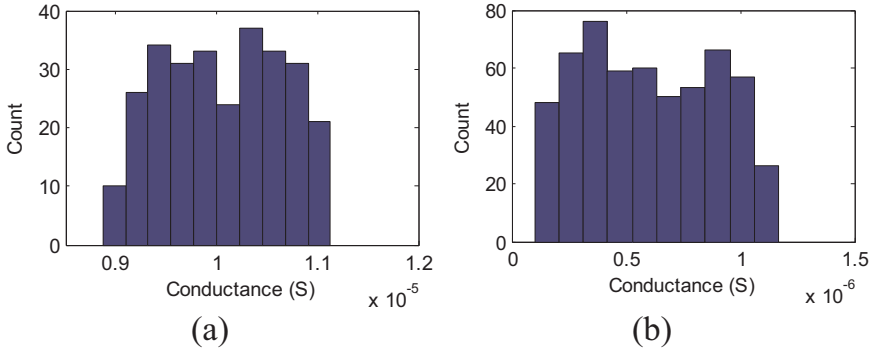

Fig. 17. Distribution of the trained weights: (a) single memristor/synapse and (b) two memristors/synapse. Trained conductances in the latter case are relatively lower.

Table 8
Crossbar power consumptions for different datasets and training approaches.

Dataset	Crossbar power (μW)	
	Single memristor per synapse	Two memristors per synapse
2 input XOR	208.9	20.9
3 input odd parity	353.3	39.8
4 input odd parity	704.5	80.9
Wine	832.4	89.7
Iris	1533.4	192.8

For the 2 input XOR, 3 input odd parity, and 4 input odd parity functions, 100% recognition accuracy was achieved for both training cases (device variation and no variation). Table 7 shows the recognition errors on test data for Iris, and Wine datasets. Test errors considering memristor device variation, stochasticity and without considering device variation, stochasticity are very close.

6.1. Power benefit over a single memristor/synapse design

In a single memristor per synapse design a conductance $\overline{\sigma}$, between the maximum and minimum device conductances, needs to be selected to define the ranges of the positive and negative weights [12]. To have equal ranges for both positive and negative weights, $\overline{\sigma}$ needs to be equal to $(\sigma_{max} - \sigma_{min})/2$. This provides range for positive weights $[0, \overline{\sigma}]$ and range for negative weights $[-\overline{\sigma}, 0)$. In such systems, trained weights are distributed symmetrically centered around $\overline{\sigma}$. Conductance of a memristor to implement a small positive (Δ) or negative weight ($-\Delta$) would be $\overline{\sigma}+\Delta$ or $\overline{\sigma}-\Delta$ respectively.

In the proposed design, before training, memristors are initialized with random high resistance values. After training, memristor conductances are concentrated near the high device resistance (or low conductance σ_{min}). In this system a small positive weight (Δ) could be implemented having $\sigma_A^+ = \sigma_{min}+\Delta$ and $\sigma_A^- = \sigma_{min}$ in Fig. 3(a). Similarly, a small negative weight ($-\Delta$) could be implemented having $\sigma_A^+ = \sigma_{min}$ and $\sigma_A^- = \sigma_{min}+\Delta$. Hence, in the proposed system, small positive or negative weights could be implemented having conductance of the corresponding memristor pair around σ_{min}. On the other hand, in a single memristor per synapse design, they are around $\overline{\sigma}$ or $(\sigma_{max} - \sigma_{min})/2$. Fig. 17 shows the distributions of the trained weights for wine dataset for the two cases: single memristor/synapse and two memristors/synapse.

In the recognition phase, the memristor crossbars consume significant amount of power. In the single memristor per synapse design, crossbar power is about 90% of the total system power. Due to the lower conductances of the trained weights in the proposed design, it has significant power benefit over the single memristor design. Table 8 shows the average (over different test data) crossbar (layer 1 + layer 2) power consumptions for the two systems. It can be observed that in the proposed system, crossbar power consumptions are about 9× less than the power consumptions in the single memristor per synapse design.

7. Conclusion

In this paper we have designed on-chip training systems for memristor based multi-layer neural networks utilizing two memristors per synapse. We utilized back propagation algorithm for training and utilized same crossbar for both forward and backward passes for a layer. We designed a novel circuit for error back propagation. We have demonstrated successful training of some nonlinearly separable datasets through detailed SPICE simulations which take crossbar wire resistance and sneak-paths into consideration. In the proposed design, the crossbars consume about 9× less power than a single memristor per synapse design.

Acknowledgment

This work was supported in part by a CAREER grant from the US National Science Foundation under Grant 1053149.

Appendix A. Supplementary material

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.mejo.2017.05.005.

References

[1] T.M. Taha, R. Hasan, C. Yakopcic, M.R. McLean, Exploring the Design Space of Specialized Multicore Neural Processors, IEEE International Joint Conference on Neural Networks (IJCNN), 2013.
[2] B. Belhadj, A.J.L. Zheng, R. Héliot, O. Temam, Continuous Real-world Inputs Can Open up Alternative Accelerator Designs, ISCA, 2013.
[3] L.O. Chua, Memristor—the missing circuit element, IEEE Trans. Circuit Theory 18 (5) (1971) 507–519.
[4] D.B. Strukov, G.S. Snider, D.R. Stewart, R.S. Williams, The missing Memristor found, Nature 453 (2008) 80–83.
[5] D. Chabi, W. Zhao, D. Querlioz, J.-O. Klein, Robust Neural LogicBlock (NLB) Based on Memristor Crossbar Array IEEE/ACMInternational Symposium on Nanoscale Architectures, pp. 137–143, 2011.
[6] T.M. Taha, R. Hasan, C. Yakopcic, Memristor crossbar based multicore neuromorphic processors, in IEEE International System-on-Chip Conference (SOCC), vol., no., pp. 383–389, 2–5 Sept, 2014.
[7] S. Yu, Y. Wu, H.-S.P. Wong, Investigating the switching dynamics and multilevel capability of bipolar metal oxide resistive switching memory, Appl. Phys. Lett. 98 (2011) 103514.
[8] G. Medeiros-Ribeiro, F. Perner, R. Carter, H. Abdalla, M.D. Pickett, R.S. Williams, Lognormal switching times for titanium dioxide bipolar memristors: origin and resolution, Nanotechnology 22 (9) (2011) 095702.
[9] F. Alibart, E. Zamanidoost, D.B. Strukov, Pattern classification by memristive crossbar circuits with ex-situ and in-situ training, Nat. Commun. (2013).
[10] M. Prezioso, F. Merrikh-Bayat, B.D. Hoskins, G.C. Adam, K.K. Likharev, D.B. Strukov, Training and operation of an integrated neuromorphic network based on metal-oxide memristors, Nature 521 (7550) (2015) 61–64.
[11] R.S. Amant, A. Yazdanbakhsh, J. Park, B. Thwaites, H. Esmaeilzadeh, A. Hassibi, L. Ceze, D. Burger, General-purpose code acceleration with limited-precision analog computation,. in: Proceeding of the 41st annual international symposium on

Computer architecuture (ISCA '14). IEEE Press, Piscataway, NJ, USA, pp. 505–516, 2014.
[12] D. Soudry, D.D. Castro, A. Gal, A. Kolodny, S. Kvatinsky, Memristor-based multi-layer neural networks with online gradient descent training, IEEE Trans. Neural Netw. Learn. Syst. (99) (2015).
[13] Boxun Li, Yuzhi Wang, Yu Wang, Y. Chen, Huazhong Yang, Training itself: Mixed-signal training acceleration for memristor-based neural network, Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific, vol., no., 20–23 Jan, 2014.
[14] S.B. Furber, S. Temple, A.D. Brown, High-PerformanceComputing for Systems of Spiking Neurons, Proceedings of AISB'06 workshop on GC5: Architecture of Brain and Mind, vol. 2, pp 29–36,Bristol, April, 2006.
[15] J. Schemmel, J. Fieres, K. Meier, Wafer-Scale Integration of Analog Neural Networks, IEEE International Joint Conference on Neural Networks (IJCNN), 2008.
[16] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, D.S. Modha, A digital neurosynaptic core using embedded crossbar memory with 45 pJ per spike in 45 nm, IEEE Custom Integrated Circuits Conference (CICC), vol., no., pp. 1–4, 19–21 Sept, 2011.
[17] P.A. Merolla, J.V. Arthur, R. Alvarez-Icaza, A.S. Cassidy, Jun Sawada, F. Akopyan, B.L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, Ivan Vo, S.K. Esser, R. Appuswamy, B. Taba, A. Amir, M.D. Flickne, W.P. Risk, R. Manohar, D.S. Modha, A million spiking-neuron integrated circuit with a scalable communication network and interface, Science 345 (6197) (2014) 668–673.
[18] C. Zamarreño-Ramos, L.A. Camuñas-Mesa, J.A. Pérez-Carrasco, T. Masquelier, T. Serrano-Gotarredona, B. Linares-Barranco, On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex, Front. Neurosci., Neuromorphic Eng. 5 (. 2011) 1–22Article 26.
[19] D. Chabi, W. Zhao, D. Querlioz, J.-.O. Klein, Robust neural logic block (NLB) based on memristor crossbar array, in Proceedings NANOARCH, pp. 137–143, 2011.
[20] J.A. Starzyk, Basawaraj, Memristor crossbar architecture for synchronous neural networks, Circuits Syst. I: Regul. Pap. IEEE Trans. 61 (8) (2014) 2390–2401.
[21] M. Prezioso, F. Merrikh-Bayat, B.D. Hoskins, G.C. Adam, K.K. Likharev, D.B. Strukov, Training and operation of an integrated neuromorphic network based on metal-oxide memristors, Nature 521 (7550) (2015) 61–64.
[22] M. Pd. Sah, C. Yang, H. Kim, L.O. Chua, Memristor Circuit for Artificial Synaptic Weighting of Pulse Inputs, IEEE ISCAS 2012.
[23] S.P. Adhikari, C. Yang, H. Kim, L.O. Chua, Memristor bridge synapse-based neural network and its learning, IEEE Trans. Neural Netw. Learn. Syst. 23 (9) (2012) 1426–1435.
[24] S.P. Adhikari, C. Yang, H. Kim, L.O. Chua, Memristor bridge synapse-based neural network and its learning, IEEE Trans. Neural Netw. Learn. Syst. 23 (9) (2012) 1426–1435.
[25] E. Zamanidoost, F.M. Bayat, D. Strukov, I. Kataeva, Manhattan Rule Training for Memristive Crossbar Circuit Pattern Classifiers, IEEE International Joint Conference on Neural Networks, 2015.
[26] E. Zamanidoost, M. Klachko, D. Strukov, I. Kataeva, Low area overhead in-situ training approach for memristor-based classifier, in Nanoscale Architectures (NANOARCH), 2015 IEEE/ACM International Symposium on, vol., no., pp. 139–142, 8-10 July, 2015.
[27] X. Dong, C. Xu, S. Member, Y. Xie, N.P. Jouppi, NVSim: a circuit-level performance, energy, and area model for emerging nonvolatile memory, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 31 (7) (2012) 994–1007.
[28] S. Theodoridis, K. Koutroumbas, Pattern Recognition, Fourth edition (4th ed.), Academic Press, 2008.
[29] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 2nd edition, Prentice Hall, 2002 (ISBN-13: 978-01379039555).
[30] 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.
[31] C. Yakopcic, T.M. Taha, G. Subramanyam, R.E. Pino, Memristor SPICE Model and Crossbar Simulation Based on Devices with Nanosecond Switching Time, IEEE International Joint Conference on Neural Networks (IJCNN), August 2013.
[32] ⟨https://archive.ics.uci.edu/ml/datasets/Wine⟩.
[33] ⟨https://archive.ics.uci.edu/ml/datasets/Iris⟩.