



# Ex-situ training of large memristor crossbars for neural network applications

Raqibul Hasan<sup>1</sup> · Chris Yakopcic<sup>1</sup> · Tarek M. Taha<sup>1</sup>

Received: 23 February 2018 / Accepted: 10 August 2018 / Published online: 29 August 2018  
© Springer Science+Business Media, LLC, part of Springer Nature 2018

## Abstract

Memristor crossbar arrays carry out multiply–add operations in parallel in the analog domain, and can enable neuromorphic systems with high throughput at low energy and area consumption. Neural networks need to be trained prior to use. This work considers ex-situ training where the weights pre-trained by a software implementation are then programmed into the hardware. Existing ex-situ training approaches for memristor crossbars do not consider sneak path currents, and they may work only for neural networks implemented using small crossbars. Ex-situ training in large crossbars, without considering sneak paths, reduces the application recognition accuracy significantly due to the increased number of sneak current paths. This paper proposes ex-situ training approaches for both 0T1M and 1T1M crossbars that account for crossbar sneak paths and the stochasticity inherent in memristor switching. To carry out the simulation of these training approaches, a framework for fast and accurate simulation of large memristor crossbars was developed. The results in this work show that 0T1M crossbar based systems can be 17–83% smaller in area than 1T1M crossbar based systems.

**Keywords** Memristor crossbars · Neural networks · Training · Ex-situ training · Sneak paths · Low power pattern recognition system

## 1 Introduction

Device variability and power consumption are among the main obstacles for continued performance improvement of future computing systems. Neural network applications have diverse uses in multiple areas including pattern recognition, signal processing, image processing, image compression, and classification of remote sensing data. Recently deep neural networks are becoming increasingly popular for object recognition and computer vision applications [1]. Interest in specialized architectures for accelerating neural networks has increased significantly because

of their ability to reduce power, increase performance, and allow fault tolerant computing.

Memristors [2] have received significant attention as a potential building block for neuromorphic systems [3]. Memristor devices implemented in a crossbar structure [3, 4] can evaluate many multiply–add operations in parallel in the analog domain very efficiently (these are the dominant operations in neural networks). Memristor crossbar based neural network systems can be orders of magnitude more area, power, and throughput efficient compared to the traditional computing systems [4].

Neural networks need to be trained prior to use. There are several training algorithms that can be used, all of which adjust synaptic weights within the network to achieve the desired functionality. In memristor crossbars, the training process would essentially set the conductance of individual memristors. There are two different approaches commonly used for training memristor crossbar based neural hardware: in-situ training and ex-situ training. The in-situ approach requires the circuit implementing the training algorithm to be on the chip. The training data would be applied to the system and the memristor

---

✉ Raqibul Hasan  
hasanm1@udayton.edu

Chris Yakopcic  
cyakopcic1@udayton.edu

Tarek M. Taha  
tarek.taha@udayton.edu

<sup>1</sup> Department of Electrical and Computer Engineering,  
University of Dayton, Dayton, OH, USA

conductivity values (representing synaptic weights) would be updated iteratively [5].

In ex-situ training, a neural network is trained in software, and the final sets of weights are then downloaded into the physical memristor crossbars. This approach does not require training circuits to be on the memristor chip, and thus offloads training complexity. Although, ex-situ training is less tolerant to crossbar defects such as stuck faults and device variations [3, 6].

Programming a memristor crossbar is challenging because a significant amount of variation is present between devices [7]. This means that identical voltage pulses may not induce identical amounts of resistance change in different memristors within a crossbar. As a result, an iterative feed-back process is necessary for programming memristor devices. Upon each iteration, a new device state is measured after each adjustment pulse is applied. This process is repeated until the target memristor is successfully programmed.

The two different memristor crossbar layouts that are studied in this work are 0T1M (0 Transistor, 1 Memristor) and 1T1M (1 Transistor, 1 Memristor) crossbars. In a 1T1M crossbar, there is an isolation transistor at each cross-point allowing any individual memristor to be accessed and read. In a 0T1M crossbar, the lack of isolation devices means that a read from the crossbar would be based on multiple memristors due to sneak path currents as explained in [16]. Sneak paths in 0T1M crossbars make reading the resistance of an individual memristor challenging (detailed in Sect. 5.3). Therefore ex-situ training of 0T1M crossbar based neural networks becomes challenging.

Ex-situ training in large crossbars, without considering sneak paths, reduces the application recognition accuracy significantly. Most of the existing works [3, 6] examined only neural networks built with small memristor crossbars. Works in [10, 11] examined ex-situ training on large crossbars, but did not consider crossbar wire resistance and sneak paths. To the best of our knowledge, no existing work demonstrate ex-situ training on large memristor crossbar based neural network considering crossbar wire resistance and sneak paths. We believe, main reason behind this is a lack of suitable simulation tool. The SPICE simulation of a large memristor crossbar (size over  $100 \times 100$ ) is very time consuming (about a day per iteration). We proposed a MATLAB framework for large memristor crossbar simulation which is very fast (less than a minute per iteration) compared to SPICE.

Ex-situ training approaches for both 0T1M, and 1T1M crossbars considering crossbar sneak paths have been developed in this work. These approaches can be very effective for deep networks which require large memristor crossbars. The proposed ex-situ training approach is able to

tolerate stochasticity in memristor devices, and we have examined the impact of device variation on the proposed ex-situ training approach.

The following sections are organized as follows: Sect. 2 describes memristor based neuron circuit and neural network designs. Section 3 demonstrates the challenges in programming large memristor crossbar based neural networks. The proposed ex-situ training process is elaborated in Sect. 4. Section 5 demonstrates experimental results and finally in Sect. 6 we summarize our work.

## 2 Memristor crossbar based neuron circuit and neural network

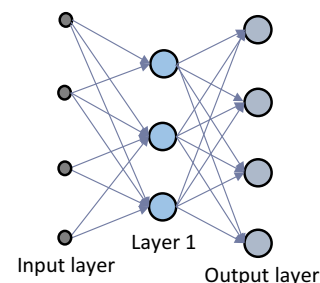
Figure 1 shows a block diagram of a two layer feed-forward neural network. Each neuron in this network performs two types of operations: a dot product of its inputs and weights (see Eq. 1), and an evaluation of a nonlinear function (see Eq. 2). If the inputs to a neuron,  $j$  are  $x_i$  and the corresponding synaptic weights are  $w_{i,j}$  then the dot product of the inputs and weights is given by:

$$dp_j = \sum_i x_i w_{i,j} \quad (1)$$

while the neuron output is given by:

$$y_j = f(dp_j) \quad (2)$$

In Eq. 2,  $f(dp_j)$  is the activation function of the neuron. In a multi-layer feed forward neural network, a non-linear differentiable activation function (such as  $f(dp_j) = [1/(1 + e^{-x})] - 0.5$ ) is desired. Deep neural networks [1] are variants of multi-layer neural networks which contain a large number of layers with a large number of neurons per layer. Neural networks with a large number of layers and a high neuron count per layer enable construction of complex nonlinear separators that allow for the classification of complex datasets.



**Fig. 1** Diagram showing a two layer neural network with four inputs, three hidden neurons, and four output neurons

### 2.1 Neuron circuit

The conductance of a memristor in a crossbar can be altered by dynamically applying an appropriate voltage across it. The voltage across a memristor typically needs to surpass a threshold voltage ( $V_{wt}$ ) in order to change the device conductance (for some devices  $V_{wt} = 1.3$  V [7]). If a voltage less than  $V_{wt}$  is applied across a memristor, it will behave like a regular resistor. This paper utilizes memristors as neuron synaptic weights.

The circuit in Fig. 2(a) shows the memristor based neuron circuit design utilized in this paper. 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 the memristor connected to the first column ( $\sigma_{A+}$ ) is higher than the conductance of the memristor connected to the second column ( $\sigma_{A-}$ ), 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. As shown below, the two op-amps can be configured to approximate a sigmoid activation function.

In Fig. 2(a), currents through the first and second columns are  $A\sigma_{A+} + \dots + \beta\sigma_{\beta+}$  and  $A\sigma_{A-} + \dots + \beta\sigma_{\beta-}$  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

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

Assume that

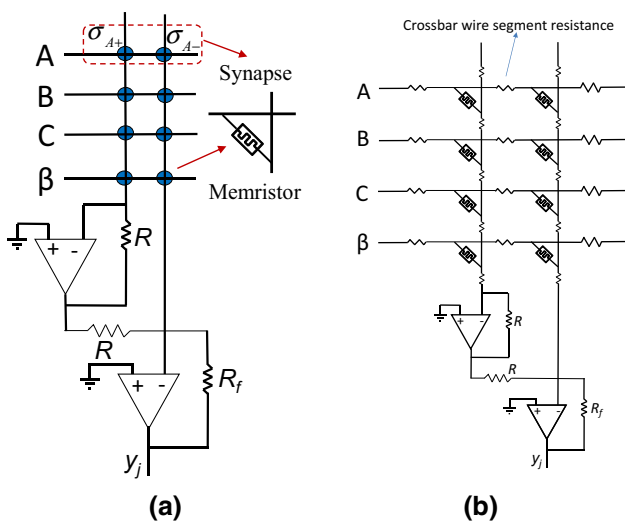


Fig. 2 Memristor based neuron circuit. A, B, C are the inputs and  $y_j$  is the output

$$DP_j = 4R_f [A(\sigma_{A+} - \sigma_{A-}) + \dots + \beta(\sigma_{\beta+} - \sigma_{\beta-})] \\ \text{(here } 4R_f \text{ is a constant)}$$

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

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

Figure 3 shows that  $h(x)$  closely approximates the sigmoid activation function,  $f(x) = 1/(1 + e^{-x}) - 0.5$ . 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. 2(b).

### 2.2 Memristor based neural network implementation

Figure 4 shows a memristor crossbar circuit used to evaluate the neural network in Fig. 1. There are two memristor crossbars in this circuit, each representing a layer of neurons. Each crossbar utilizes the neuron circuit shown in Fig. 2.

In Fig. 4, the first layer of the neurons is implemented using a  $5 \times 6$  memristor crossbar. The second layer of four neurons is implemented using a  $4 \times 8$  memristor crossbar, where three of the inputs are coming from the three output neurons of the first crossbar. One additional input is used as a bias. A key benefit of this circuit is that by applying all the inputs to a crossbar, an entire layer of neurons can be processed in parallel within one cycle. Implementation of a memristor based deep network is similar. As deep networks typically have large numbers of neurons (each with a large number of inputs), large memristor crossbars would be required for such neural networks.

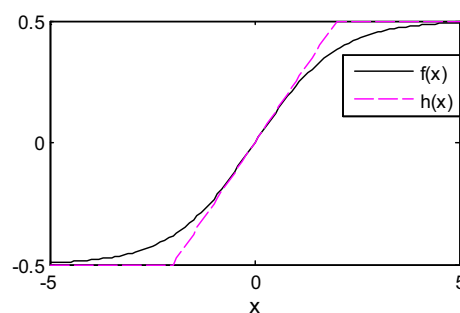


Fig. 3 Plot of functions  $f(x)$  and  $h(x)$

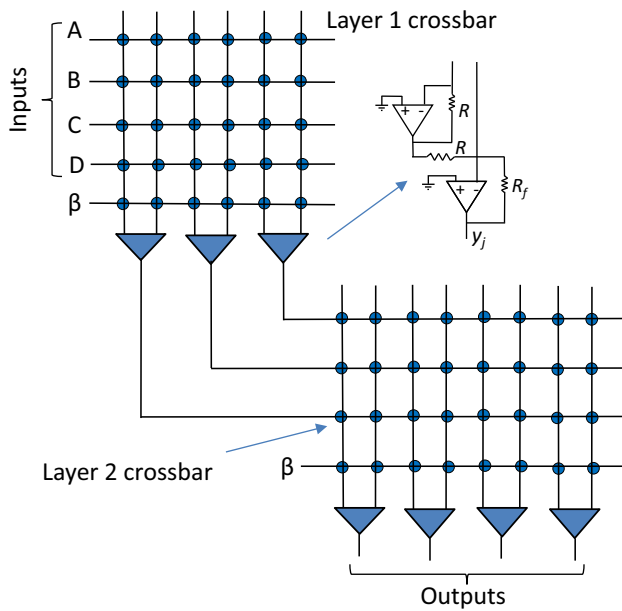


Fig. 4 Schematic of the neural network shown in Fig. 1

### 2.3 Traditional ex-situ training approach

The objective of the ex-situ training process is to set each memristor in the crossbar to a specific resistance. Because the memristor devices have stochastic behavior, multiple write pulses may be required to set a memristor to a target state, and reads will have to be performed after each write pulse to ensure correct programmability. This is essentially a feedback write process that requires the ability to read the resistance of each individual memristor in a crossbar.

Writing to a single memristor in a crossbar is relatively straight forward compared to reading. To write, a voltage drop  $V_w$  should be applied to a target memristor, while a voltage drop less than the threshold  $V_{th}$  should be applied to all other memristors [8]. Figure 5 shows how to change the conductance of a single memristor while keeping the conductance of the remaining memristors in the crossbar unchanged.

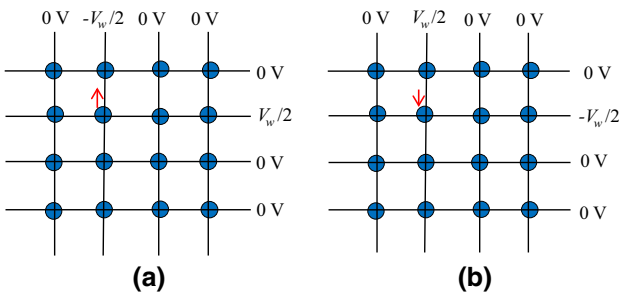


Fig. 5 Demonstration of the write operation in a crossbar: **a** increasing memristor conductance (upward arrow) and **b** decreasing memristor conductance (downward arrow)

*Programming a 1T1M crossbar:* In the ex-situ training process, reading an individual memristor resistance levels in a crossbar is challenging due to sneak paths. Placing a transistor at each cross-point (Fig. 6) will ensure that only the resistance of the target memristor is impacting the column current during the read process as in a 1T1M crossbar [9]. In this process, only the *select* line ( $S_{ri}$ ) corresponding to the target memristor is enabled.

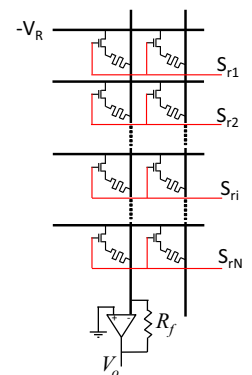
*Programming a 0T1M crossbar:* Ex-situ training in 0T1M crossbars has a significant area benefit over 1T1M crossbars due to the elimination of the cross-point transistors. Figure 7 shows the circuit used to read a particular memristor from a 0T1M crossbar. An op-amp is utilized at the bottom of the corresponding crossbar column and a read voltage,  $-V_R$  is applied to the corresponding row. The remaining rows and columns in the crossbar are set to 0 V. In an ideal circuit, the op-amp output can be expressed as  $V_o = V_R \times (R_f/R_1)$ , which is a function of only  $R_1$  as  $R_f$  and  $V_R$  are constant. But in a real circuit, the voltage  $V_o$  is essentially a function of all the memristors in the crossbar due to sneak path currents. Works in [3, 6, 10, 11] examine ex-situ training in 0T1M crossbar without considering crossbar parasitic resistance. Those approaches may work for small crossbar based implementations but would not perform desirably for large crossbar based implementations (detailed in Sect. 3.3).

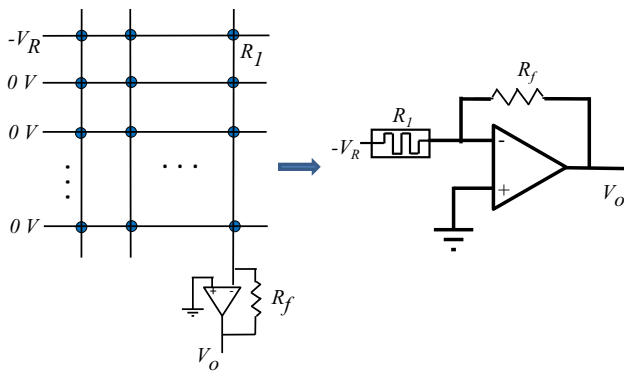
## 3 Large crossbar ex-situ training challenges

### 3.1 Large crossbar simulations

Recall that large deep networks require large memristor crossbars. The SPICE simulation of large memristor crossbars (size over  $100 \times 100$ ) is very time consuming (about a day per iteration). This section presents a MATLAB framework for large memristor crossbar simulation which is very fast (less than a minute per iteration) compared to SPICE, and is as accurate as SPICE when

Fig. 6 Accessing a memristor in a 1T1M crossbar. Each of the select lines ( $S_{r1}, S_{r2}, \dots, S_{rN}$ ) enables a row of memristors in the crossbar



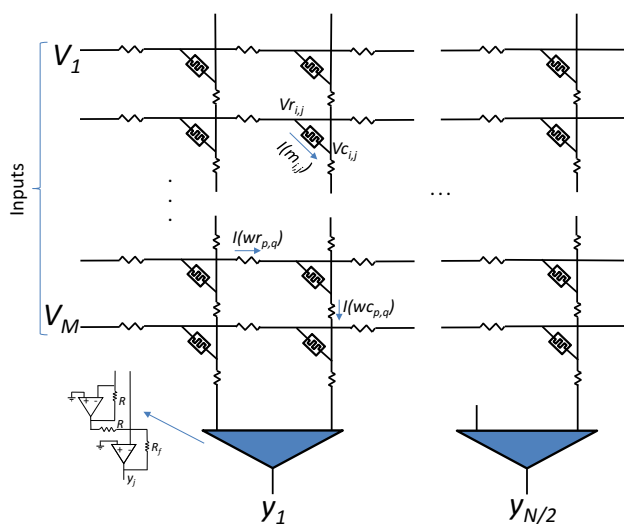


**Fig. 7** Reading a particular memristor from a 0T1M crossbar. In an ideal case, the circuit at right is functionally equivalent to the circuit at left

accounting for crossbar wire resistances and driver resistances.

Consider the  $M \times N$  crossbar in Fig. 8. There are  $MN$  memristors in this crossbar circuit,  $2MN$  wire segments, and  $M$  input drivers. For any given set of crossbar input voltages,  $2MN$  node voltages need to be determined. The Jacobi method of solving systems of linear equations is used to determine these unknown voltages. All the nodes within the crossbar rows ( $Vr_{i,j}$  for  $i = 1, 2, \dots, M$  and  $j = 1, 2, \dots, N$ ) are initialized to the applied input voltages, while each of the nodes on the crossbar columns ( $Vc_{i,j}$  for  $i = 1, 2, \dots, M$  and  $j = 1, 2, \dots, N$ ) are initialized to 0 V (due to the virtually grounded op-amp circuits in Fig. 2(a)).

Then currents through each of the memristors ( $I(m_{i,j})$  for  $i = 1, 2, \dots, M$  and  $j = 1, 2, \dots, N$ ) are determined based on the initial values of  $Vr_{i,j}$ ,  $Vc_{i,j}$  and the known memristor resistances. The currents through the horizontal and vertical wire resistors ( $I(wr_{i,j})$  and  $I(wc_{i,j})$  for  $i = 1, 2, \dots, M$  and



**Fig. 8** Schematic of a  $M \times N$  crossbar implementing a layer of  $N/2$  neurons

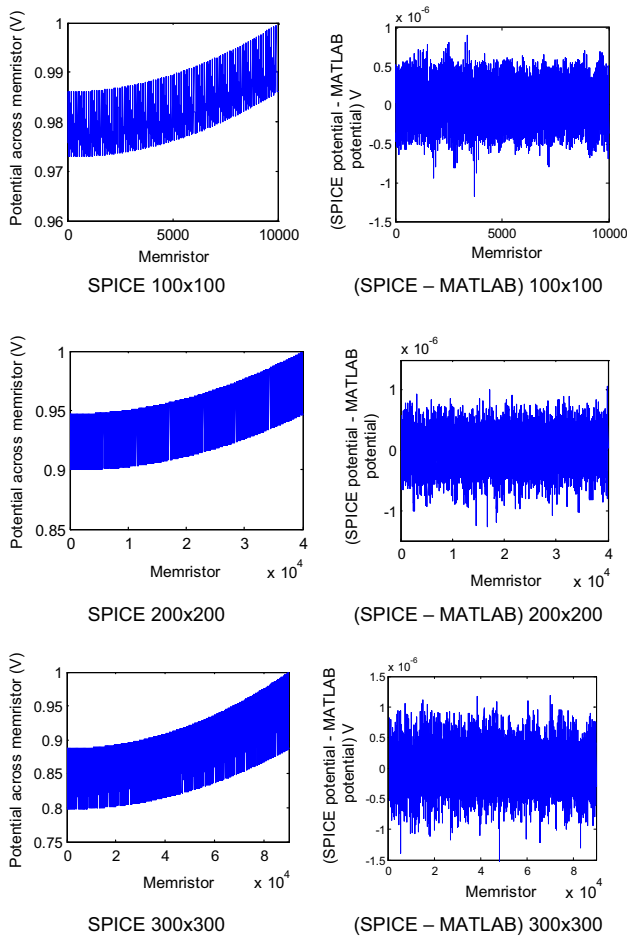
$j = 1, 2, \dots, N$ ) are determined based on the measured currents through the memristors using the Kirchoff’s current law. The voltage drops across the wire resistances are determined based on the measured currents through them. Then the crossbar node voltages ( $Vr_{i,j}$  and  $Vc_{i,j}$  for  $i = 1, 2, \dots, M$  and  $j = 1, 2, \dots, N$ ) are adjusted (based on the voltage drops across the wire resistances and the applied input voltages). This process is repeated until the node voltages reach a stable solution.

We have compared the crossbar simulation results obtained from SPICE and the MATLAB framework for crossbars of sizes  $100 \times 100$ ,  $200 \times 200$ , and  $300 \times 300$ . In these three crossbars 1 V is applied as input to the leftmost point of each row, while the bottom of the crossbar columns are grounded. The conductance of the each memristor in the crossbars is set to  $0.55 \mu\text{S}$  [7] and each wire segment resistance is set to  $5 \Omega$ . In an ideal circuit (ignoring the crossbar wire resistance) the potential across the memristors will be 1 V. Figure 9 shows the voltage drop across the memristors in the crossbar from SPICE simulation and the difference between SPICE and the MATLAB framework results. In Fig. 9 (left column), the voltage across the memristors are plotted from the 1st row to the last row, and from left to right within each row.

Figure 9 (left column) shows the potential across the memristors in crossbars of different sizes obtained through SPICE simulation. Right column shows the difference of the corresponding potentials across the memristors in the crossbars, obtained form SPICE and the proposed MATLAB framework based simulations. We can observe that the corresponding results from both simulations are very close (absolute values of the difference of the corresponding values are less than  $2 \times 10^{-6}$ ). These results also imply that sneak paths were properly evaluated in the MATLAB simulations.

### 3.2 Impact of sneak paths in a large crossbar

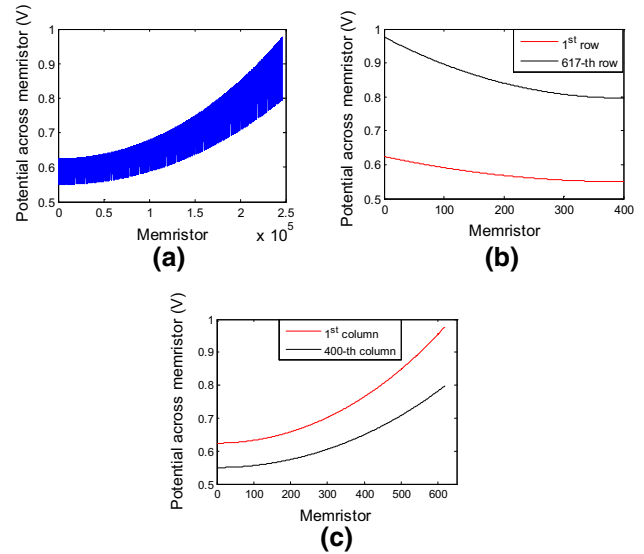
From Fig. 9 it can be seen that as we increase the crossbar size impact of sneak paths become more severe. In the  $300 \times 300$  crossbar potential across the memristors were between 0.8 V to 1 V. We also have examined the impact of sneak paths in a  $617 \times 400$  crossbar which is utilized in a neural network implementation in the following subsection. Conductance of each memristor in the crossbar is  $0.55 \mu\text{S}$  and 1 V is applied as input to the leftmost point of each row. We could not simulate such big crossbar using our version of the LTspice-IV. Figure 10 shows the resulting voltage drop across each memristor in this crossbar obtained through the MATLAB framework. In an ideal circuit, the voltage across each memristor would be 1 V. However, due to the sneak paths in the crossbar, the actual voltage drop across these memristors is less than 1 V. In



**Fig. 9** Potential across the memristors in crossbars of different sizes obtained through SPICE simulations (left column). Difference of the corresponding potentials across the memristors in the crossbars, obtained from SPICE and the proposed MATLAB framework based simulations (right column)

fact, the minimum potential across a memristor in the crossbar is 0.55 V.

Plots in Fig. 10 assume the schematic of the crossbar as shown in Fig. 8. Potential across the memristors in Fig. 10(b) are plotted from left to right in a row. In Fig. 10(c), potential across the memristors are plotted from top to bottom in a column. Figure 10(b) shows that in a row from left to right potential across the memristors gradually decreases. Reasoning behind this is in a row the rightmost memristor has relatively longer wire in the path from a driver to a sink. Figure 10(c) shows that in a column from top to bottom potential across the memristors gradually increases. Reasoning behind this is in a column the bottom memristor has relatively shorter wire in the path from a driver to a sink (see Fig. 8).



**Fig. 10** Voltage drop across the memristors in a  $617 \times 400$  crossbar: **a** for all the memristors in the crossbar **b** for the 1st and the 617-th rows of the crossbar **c** for the 1st and the 400-th columns of the crossbar

### 3.3 Impact of sneak paths in ex-situ training of 1T1M crossbars

Existing ex-situ training approaches train neural networks without considering sneak paths that may occur within a crossbar. For small scale neural networks (e.g. network for Iris dataset) traditional ex-situ training works satisfactorily [3, 6]. But for large neural networks, traditional training approach does not work well. We have examined ex-situ training for ISOLET dataset [12] using the existing approach (which does not consider crossbar sneak path currents). The ISOLET dataset has 6238 training data and 1559 test data belonging to 26 classes. Each data sample has 617 attributes. We have considered the following neural network configuration:  $617 \rightarrow 200 \rightarrow 26$  (617 inputs, 200 hidden neurons and 26 output neurons).

We programmed the weights obtained using the simplified ex-situ training approach (that ignores wire resistance) into a crossbar where each wire segment has a resistance of  $5 \Omega$ . Recognition error on test dataset in software was 9.49%. After the trained weights were programmed into 1T1M crossbars, we used the proposed MATLAB framework for the evaluation of the neural network which accounts sneak path currents. In this case the recognition error is increased to 13.08%. This is due to the presence of sneak paths in the recognition phase (as all the cross-point transistors are turned on and entire crossbar is processed in parallel).

## 4 Proposed ex-situ training process

We have examined ex-situ training approach using both 0T1M, and 1T1M crossbars that accounts for crossbar sneak paths. The neural networks trained in software essentially account for the wire resistance within memristor crossbars (using the framework described in Sect. 3.1). After training we have the trained memristor conductances which need to be programmed in the crossbars. We quantized the conductances into 8 bit representations before programming the memristors.

*Programming a 1T1M crossbar:* Reading the conductivity of a particular memristor in a 1T1M crossbar is straightforward, as the crosspoint transistors are able to effectively eliminate sneak paths. As a result, programming memristor conductance values in a 1T1M crossbar is also straight forward. During evaluation/recognition of a new input, all the transistors in a 1T1M crossbar will be turned on. Thus the recognition phase is still prone to sneak path currents. However, since the memristor conductance values were determined in software considering the crossbar sneak paths (using the proposed simulation tool), this should not be a problem.

*Programming a 0T1M crossbar:* On the other hand, reading a single memristor from a 0T1M crossbar is complex due to sneak paths as stated in the previous section. In Fig. 7,  $V_o$  is essentially a function of all the memristors in the crossbar due to these unwanted sneak paths. If other memristor conductance values are known, we can determine expected value of  $V_o$  for a target value of  $R_1$  (in Fig. 7) from the software simulation of the circuit.

For programming memristors in a 0T1M crossbar we propose to initialize all the memristors in the crossbar with the lowest possible conductance value of the device. Then the memristors in the crossbar will be programmed one by one. Thus when programming a particular memristor in the crossbar, we will know the conductance of the rest of the memristors in the crossbar. During programming, for each memristor the expected read voltage will be evaluated in software, based on the initialization of the memristors (lowest conductance) and the already programmed conductances. These expected read voltages account the sneak paths. A target conductance will be programmed in the physical memristor crossbar based on the corresponding expected read voltages. Thus the ex-situ training process takes sneak paths into consideration for a 0T1M crossbar.

Recognition accuracy for both types of crossbar designs (0T1M and 1T1M) would be about the same as both approaches take sneak paths into consideration. But the 0T1M systems are more area efficient than the 1T1M systems.

*Training circuits needed:* We assumed that the ex-situ programming process will be coordinated by an off-chip system and so added only the necessary hardware needed to allow the external system to program the crossbars. These include two D-to-A converters, along with a set of buffers and control circuits per crossbar. The use of two D-to-A converters per crossbar will serialize the programming process for each crossbar. We assume this is not a problem as this system will be programmed once and then deployed for use. A proposed circuit for the training is shown in Fig. 11. Since memristors will be programmed one at a time, only one of these circuits is required per crossbar.

## 5 Experimental evaluations

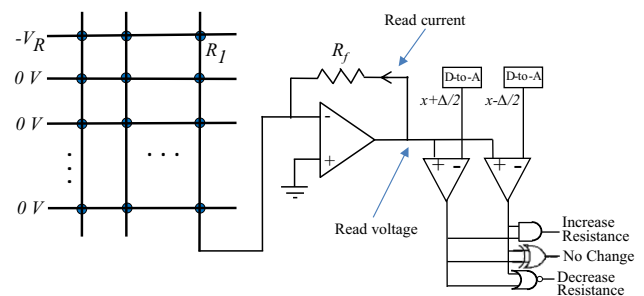
### 5.1 Training datasets

We have examined the proposed ex-situ training approach on three different datasets. Each of these datasets require a nonlinear separator for classification, and hence, each requires the use of a multi-layer neural network.

*Iris dataset:* We have utilized the Iris dataset [13] consisting of 150 patterns belonging to three different classes: Iris Setosa, Iris Versicolour, and Iris Virginica. Each pattern consists of 4 attributes/features which were normalized such that the maximum attribute magnitude is 1. We trained a two-layer neural network consisting of 15 neurons in the hidden layer and 3 in the output layer.

*KDD dataset:* We used a four-layer deep network when utilizing the KDD dataset [14]. We used 20,000 data for training and 5000 data for testing. The neural network configuration was  $41 \rightarrow 100 \rightarrow 50 \rightarrow 20 \rightarrow 1$  in this case.

*ISOLET dataset:* Described in Sect. 3.3.



**Fig. 11** Circuit used to program a single memristor to a target resistance. Here  $x$  is the target read voltage and  $\Delta$  determines the programming precision

## 5.2 Memristor model

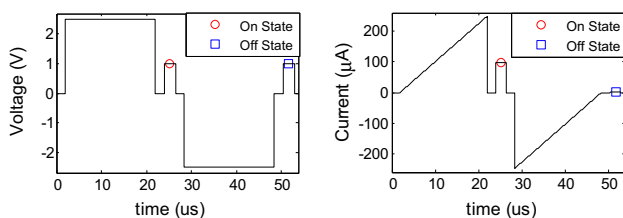
An accurate memristor model [15] of a  $\text{HfO}_x$  and  $\text{AlO}_x$  memristor device [7] was used for simulations. The switching characteristics for the device model are displayed in Fig. 12. This device was chosen for its high minimum resistance value and large resistance ratio. According to the data presented in [7] this device has a minimum resistance of 10 k $\Omega$  and a resistance ratio of  $10^3$ . The full resistance range of the device can be switched in 20  $\mu\text{s}$  by applying a 2.5 V pulse across the device.

## 5.3 Results

**Programming 1T1M crossbar:** To demonstrate the programming flow, consider the programming of the memristor in the 1st row and 398th column for the ISOLET dataset. The conductance of the memristor obtained from software simulation (using the proposed simulation tool) was 0.923  $\mu\text{S}$ . For the 1T1M system, read current would be  $V_r / [(618 + 400)R_w + 1/9.22e-7]$  or 0.918  $\mu\text{A}$ , where  $V_r = 1$  V, is the applied read voltage and  $R_w = 5$   $\Omega$ , is the crossbar wire segment resistance.

**Programming 0T1M crossbar:** For a 0T1M crossbar we are assuming that the memristors will be programmed one by one, from the first row to the last row and from left to right within each row. When we are programming the memristor in the 1st row and 398th column of the 0T1M crossbar, the memristors in the 1st column through the 397th column of this row will have already been programmed. The conductance values of the memristors in the other rows are the lowest conductance of the device (0.1  $\mu\text{S}$ ). The desired read current for this memristor is 0.675  $\mu\text{A}$  when applied read voltage is 1 V (as evaluated by the software implementation of the crossbar).

Notice that the two read currents for the two types of crossbars are different (0.918  $\mu\text{A}$  for 1T1M crossbar and 0.675  $\mu\text{A}$  for 0T1M crossbar). Absence of isolation transistor in the 0T1M crossbar making sneak current paths



**Fig. 12** Simulation results displaying the input voltage and current waveforms for the memristor model [15] that was based on the device in [7]. The following parameter values were used in the model [15] to obtain this result:  $V_p = 1.3$  V,  $V_n = 1.3$  V,  $A_p = 5800$ ,  $A_n = 5800$ ,  $x_p = 0.9995$ ,  $x_n = 0.9995$ ,  $\alpha_p = 3$ ,  $\alpha_n = 3$ ,  $a_1 = 0.002$ ,  $a_2 = 0.002$ ,  $b = 0.05$ ,  $x_0 = 0.001$

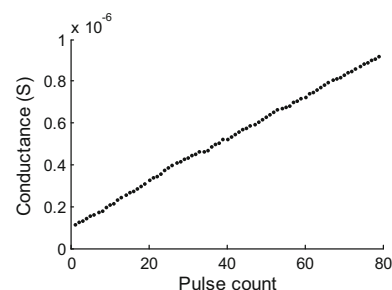
problem worse and hence the target read current is different from the target read current in the 1T1M crossbar for the same memristor. If a programming approach on 0T1M crossbar attempts to target 0.918  $\mu\text{A}$  (which does not considers sneak current paths) read current for this particular memristor would end up with very different conductance value (instead of the desired 0.923  $\mu\text{S}$ ). Therefore, the ex-situ training which does not considers crossbar sneak path currents accurately would not perform satisfactorily.

We have examined the impact of memristor device stochasticity by allowing random deviations from the programmed memristor conductance when a write pulse is applied. The maximum deviation could be  $\pm 50\%$  of the expected change in conductance. That is, when the conductance of a memristor is supposed to be updated by amount  $x$ , the corresponding programming pulse would update the conductance by an arbitrary value randomly taken from the interval  $[0.5x, 1.5x]$ . We assumed a programming pulse width of 2 ns, and thus needed several pulses to successfully program the intended final resistance level.

Figure 13 shows iterative change in conductance of the memristor (in 0T1M crossbar) during the programming process. We can observe that after about 82 iterations, the memristor was programmed to the desired conductance within the programming precision boundaries. Table 1 shows the recognition error for the different test datasets obtained from the software implementations, as well as the proposed ex-situ training approach. Recognition accuracy for both types of crossbar designs (0T1M and 1T1M) are about the same, as both approaches take sneak paths into consideration.

## 5.4 Area analysis

We have examined the area savings of 0T1M crossbar based neural network implementations over 1T1M crossbar based designs. We assumed memristors were fabricated on top of the transistor layer for both the 0T1M and 1T1M



**Fig. 13** Demonstration of memristor programming through ex-situ process



**Table 1** Recognition error on test data

	Test error (%)	
	Training of the ckt. in s/w	Ex-situ training
Iris	3.92	3.92
KDD	2.4	2.42
ISOLET	9.49	10

**Table 2** Area savings in OT1M crossbar systems over 1T1M crossbar systems for implementing neural networks of different configurations

Dataset	Area (Transistors)		Area reduction of OT1M (%)
	1T1M	OT1M	
Iris	1212	1002	17
KDD	28530	8290	71
ISOLET	268950	45578	83

designs. We have approximated the area of each system in terms of the number of transistors used. In the area calculations, we have accounted for the circuits contributing to the neural network programming and evaluation.

Table 2 shows the area savings of OT1M crossbars over 1T1M crossbars when implementing neural networks of different configurations. We can observe that a OT1M crossbar based system can provide between 17% and 83% area reduction over the 1T1M crossbar based systems. This result shows that larger crossbars benefit from greater area reduction in a OT1M based architecture.

### 5.5 Impact of device variation

*1T1M crossbar:* Due to device variations, conductance ranges of different memristors in a crossbar may not be same, and it is not possible to program a memristor to a value beyond its conductance range. Prezioso et al. [6] examined the presence of variation in minimum conductivity between devices in a crossbar. To cope with this issue, software training could be restricted such that the trained conductance values are within the conductance range of most of the memristors in a physical crossbar. Thus the impact of device variation in ex-situ training of 1T1M crossbars could be minimized.

*OT1M crossbar:* When performing ex-situ training on a OT1M crossbar we assumed all the memristor conductance values will be initialized to the minimum device conductance. In a physical crossbar not all the memristors will be at the same exact conductance value due to device variations. We have examined the impact of this issue when ex-situ training OT1M crossbars. From [6] it can be seen that

**Table 3** Impact of variations in the minimum device conductance on recognition accuracy

Standard deviation ( $\mu\text{S}$ )	Recognition error (%)
0.07	10.5
0.14	11.8
0.28	12.5

the minimum conductance values between memristors in a crossbar is normally distributed and the standard deviation of the distribution is 0.07  $\mu\text{S}$ . For the device considered in this paper, mean of minimum device conductance was 0.1  $\mu\text{S}$ . The recognition error for the ISOLET dataset, considering such device variation, when ex-situ training a OT1M crossbar was 10.5%. This is slightly greater than the error obtained when device variation is omitted, but still lower than the error obtained when ignoring the impact of sneak paths (Sect. 3.3). Greater standard deviation of the lowest device conductances would induce slightly more recognition error. Table 3 shows the recognition error for different standard deviations of the lowest device conductances.

## 6 Conclusion

Existing ex-situ training approaches for memristor crossbars do not consider sneak paths and may only work for smaller neural network implementations. This paper proposes an ex-situ training approach for both OT1M and 1T1M crossbars that accounts for crossbar sneak paths. A framework for fast and accurate simulation of large memristor crossbars was also developed so that crossbars with sneak paths can be simulated very quickly. Proposed ex-situ training approaches provide better recognition accuracies compared to the traditional approach for large neural networks. The ex-situ training approach is also able to tolerate the stochasticity in memristor devices. In regards to area, the OT1M systems are about 17–83% smaller in size when compared to the 1T1M systems (this includes programming circuit area).

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

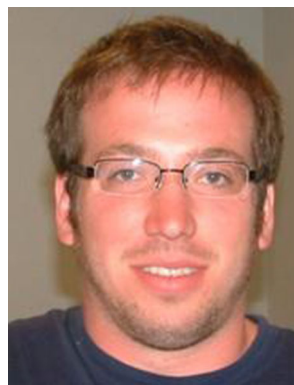
## References

- Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, 10, 1–40.
- Chua, L. O. (1971). Memristor—The missing circuit element. *IEEE Transactions on Circuit Theory*, 18(5), 507–519.

3. Alibart, F., Zamanidoost, E., & Strukov, D. B. (2013). Pattern classification by memristive crossbar circuits using ex situ and in situ training. *Nature Communications*, 4, 2072.
4. Taha, T. M., Hasan, R., & Yakopcic, C. (2014). Memristor crossbar based multicore neuromorphic processors. In *IEEE international system-on-chip conference (SOCC)* (pp. 383–389).
5. Soudry, D., Di Castro, D., Gal, A., Kolodny, A., & Kvatinsky, S. (2015). Memristor-based multilayer neural networks with online gradient descent training. *IEEE Transactions on Neural Networks and Learning Systems*, 26(10), 2408–2421.
6. Prezioso, M., Merrih-Bayat, F., Hoskins, B. D., Adam, G. C., Likharev, K. K., & Strukov, D. B. (2015). Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature*, 521(7550), 61–64.
7. Yu, S., Wu, Y., & Wong, H.-S. P. (2011). Investigating the switching dynamics and multilevel capability of bipolar metal oxide resistive switching memory. *Applied Physics Letters*, 98, 103514.
8. Dong, X., Xu, C., Member, S., Xie, Y., & Jouppi, N. P. (2012). NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 31(7), 994–1007.
9. Yakopcic, C., Hasan, R., & Taha, T. M. (2015). Memristor based neuromorphic circuit for ex-situ training of multi-layer neural network algorithms. In *IEEE IJCNN*.
10. Hasan, R., Taha, T. M., & Yakopcic, C. (2015). Ex-situ training of dense memristor crossbar for neuromorphic applications. In *The proceedings of the IEEE international symposium on nanoscale architectures*.
11. Yakopcic, C., Taha, T. M., & McLean, M. R. (2015). Method for ex situ training in a memristor-based neuromorphic circuit using a robust weight programming method. *Electronics Letters*, 51(12), 899–900.
12. <https://archive.ics.uci.edu/ml/datasets/ISOLET>.
13. <https://archive.ics.uci.edu/ml/datasets/Iris>.
14. <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>.
15. Yakopcic, C., Taha, T. M., Subramanyam, G., & Pino, R. E. (2013). Memristor SPICE model and crossbar simulation based on devices with nanosecond switching time. In *IEEE international joint conference on neural networks (IJCNN)*.
16. Zidan, M. A., Fahmy, H. A. H., Hussain, M. M., & Salama, K. N. (2012). Memristor-based memory: The sneak paths problem and solutions. *Microelectronics Journal*, 44(2), 176–183.



**Raqibul Hasan** received his B.S. degree in Computer Science and Engineering from the Bangladesh University of Engineering and Technology (BUET) in 2009. He received his M.S. and Ph.D. degrees in Electrical Engineering from the University of Dayton in 2014 and 2016 respectively. His current research includes memristor based low power circuits and systems design.



award for a paper on memristor device modeling.

**Chris Yakopcic** (IEEE S'11–M'14) received his B.S., M.S., and Ph.D. degrees in Electrical Engineering from the University of Dayton in 2009, 2011, and 2014 respectively. His current research includes memristor device modeling, analog circuit design with memristor devices, and implementing neuromorphic algorithms on memristor crossbars. In 2013 he was awarded the IEEE/INNS International Joint Conference on Neural Networks best paper



**Tarek M. Taha** (S'96–M'03) received the B.S. degree from DePauw University, Greencastle, Indiana, in 1996 and the BSEE, MSEE, and Ph.D. degrees in Electrical Engineering from the Georgia Institute of Technology, Atlanta, in 1996, 1998, and 2002, respectively. He is an Associate Professor of Electrical and Computer Engineering at the University of Dayton. His research interests include cognitive computing architectures, high performance computing, and architectural performance modeling. He received the NSF CAREER Award in 2007. His research is supported by the National Science Foundation, Air Force Office of Scientific Research, the Air Force Research Laboratory, and DARPA. He is a member of the IEEE Computer Society.