Design Space Evaluation of a Memristor Crossbar Based Multilayer Perceptron for Image Processing

Chris Yakopcic*, B. Rasitha Fernando, and Tarek M. Taha
Dept. Of Electrical and Computer Engineering
University of Dayton
Dayton, OH, USA
*cyakopcic1@udayton.edu

***Abstract*—This paper describes a simulated memristor-based neuromorphic system that can be used for ex-situ training of a multi-layer perceptron algorithm. The presented programming technique can be used to map the weights required of a neural algorithm directly onto the grid of resistances in a memristor crossbar. Using this weight-to-crossbar mapping approach along with the dot product calculation circuit, neural algorithms can be easily implemented using this system. To show the effectiveness of this circuit, a Multilayer Perceptron is trained to perform Sobel edge detection. Following these simulations, an analysis was presented that shows how memristor programming accuracy and network size are related to output error; the results show that network size can be increased to reduce testing error. In some cases, the memristors in the circuit may be capable of operating with at lower precision if the network size is increased. This means that less precise (or lower resolution) memristor devices may be used to implement the proposed system. Furthermore, a power, timing, and energy analysis shows that this circuit has a computation throughput that allows it to process 4K UHD video in real time at approximately 337mW.**

Keywords—memristor, memristive, perceptron, dot-product

I. INTRODUCTION

The main obstacles for continued performance improvement in future computing systems are reliability and power consumption. Embedded neuromorphic processing systems have significant advantages to offer, such as the ability to solve complex problems while consuming very little power and area [1,2]. These neural network accelerators can be used for a broad number of applications [3]. Furthermore, Chen et al. [4] have shown that Recognition, Mining, and Synthesis (RMS) applications (described by Intel as the key application drivers of future [3]) can be performed using neural networks.

The memristor [5] has received significant attention as a potential building block for neuromorphic systems. The physical realization of the memristor [6] has produced a nanoscale non-volatile device with a large varying resistance range. Voltage pulses can be applied to memristors to alter their conductivity and tune them to a specific resistance state [7-9]. Physical memristors [10-13] can be laid out in a high density grid known as a crossbar [14-16]. Using this layout, memristors have the potential to be fabricated with a synaptic density greater than that of brain tissue [17]. Systems based on these circuits will produce high density, extreme low-power, neuromorphic hardware that is capable of performing many multiply-add operations in parallel in the analog domain [18,19]. Thus, neuromorphic systems based on memristor crossbars [20-25] have potential to perform at a power efficiency of 6 to 8 orders of magnitude greater than that of traditional RISC processors [26].

Both memristor-based [27-45] and non-memristor based [46-48] wafer-scale hardware implementations of neural networks have been proposed. Some of these systems are capable of performing in-situ training, or on-chip learning [35]. Other systems are based on an ex-situ approach where weights are pre-trained in software [29]. Furthermore, groups have studied the impact of using memristors to implement convolution [49,50] or develop a Convolutional Neural Network [50-52]. Thus, a variety of different memristor architectures have been proposed that each implement different aspects of different neural network algorithms.

This paper presents a circuit that uses a weight-to-crossbar mapping scheme that produces a dot-product output that is comparable to what is obtained using traditional software. In this system, the memristor conductance values are predetermined using an ex-situ training process. Thus, the best weights for this system can be pre-determined off chip. We use a feedback process to program these crossbars to ensure that memristor devices (which commonly exhibit some degree of stochastic behavior when switching [53,54]) are correctly programmed. We also use a weight programming scheme that uses a D-to-A for programming, and we study the minimum required bit-width and precision of the D-to-A circuits using a detailed study.

In this work we use the proposed circuit to implement a multilayer perceptron trained to perform Sobel edge detection. From this example, we show how neural network size may impact mean squared error (MSE) in this system, and we show that increasing the number of neurons in the circuit allows this circuit to utilize lower resolution memristors to obtain the same MSE. Furthermore, we perform an energy, power, and timing analysis that brings forth a relationship between neural network evaluation accuracy and power consumed for real time operation. In general, this work presents a circuit that is capable of learning digital image processing functionality. Thus, due to the highly parallel memristor crossbar design, the circuit provides high speed and low power operation.

Related works [31] have used summing amplifier techniques to carry out analog multiply-adds in a memristor crossbar, but this paper goes one step further and uses a method for weight conversion that allows us to obtain the expected dot product values (between the inputs and the weight matrix) precisely and reliably. Others have proposed alternative transformation strategies for memristor crossbars [40], but not to achieve versatile functionality based on accurate dot product calculations. We are also not aware of any other work that presents a parameter analysis that is able to provide a succinct relationship between neural network size, training time, MSE, memristor resolution and programming accuracy, energy, timing, and power.

Work in this paper builds on a design that has been presented in several iterations. Work in [29] describes the first version of this circuit and required a 1T1M (1-Transistor 1-Memristor) system (as opposed to the 0T1M (0-Transistor 1-Memristor) system in this work). Next [18,55] describe the first instances of the circuit framework described in this paper, and detail how this single circuit can be used to implement several different neural network algorithms. Work in [56] shows how this circuit can be used to train a network capable of performing network intrusion detection. Lastly, [19,50] show how the presented circuit can be altered to implement a convolutional neural network. The unique aspects presented in the following work are parameter analyses that allow the user to form a relationship between network size, MSE, training time, and memristor resolution and programming accuracy. Furthermore, we uniquely present an energy, power, and timing analysis showing real-time digital image processing is feasible with memristor based neural network systems.

II. Memristor Based Dot-Product Calculator

The circuit in Fig. 1 is used to implement neurons that are capable of producing a true dot product calculation. This circuit will be used as the building block in the presented memristor crossbar system. In this circuit, two memristors are used to represent a single weight so that negative numbers can be represented accurately. To achieve this, the weight matrix from the neural algorithm W must first be transformed into two different matrices [18]. The matrix W^+ contains positive non-zero elements where $W_{ij} > 0$ and zeroes in all other positions. Therefore, $W_{ij}^+ = W_{ij}$ when $W_{ij} > 0$ and $W_{ij}^+=0$ when $W_{ij} \leq 0$. Likewise, the matrix W^- contains positive non-zero elements in each position where $W_{ij} < 0$. Therefore, $W_{ij}^- = |W_{ij}|$ when $W_{ij} < 0$, and $W_{ij}^- = 0$ where $W_{ij} \geq 0$. Using this method, W can always be reproduced by subtracting W^- from W^+.

Memristor conductivities are used to represent the weight values in the neural network. Therefore, the value of each weight in the matrices W^+ and W^- must be converted to a bounded number within a conductance range between σ_{min} and σ_{max}. Equations (1) and (2) are used to convert the weight matrices W^+ and W^- into matrices of conductivity values (σ^+ and σ^-) that can be programmed into a memristor crossbar. Using this weight organization structure, both positive and negative weights can be represented because each weight value is represented by a pair of memristors, each tied to a voltage input that differs only in polarity. Weight polarity is determined by the polarity of the voltage input connected to the memristor in a given pair with the higher conductivity.

Adding σ_{min} to all values ensures that all zeros in the W^+ and W^- matrices will be at the minimum conductance level of the memristors. The op-amp circuit at the crossbar column output in Fig. 1 is used to both scale the column output voltage and implement the neuron activation function. This is an alternative approach to the systems that require a sigmoid activation circuit following an op-amp output such as [31]. It is desirable to have a non-binary activation function (such as a sigmoid) at each neuron output to allow for non-binary outputs and improved classification potential. To accomplish this in our work, the amplifier at the output of the circuit is used to create a pseudo sigmoid function as shown in Fig. 2. A linear amplifier transfer function (bounded by upper and lower voltage rails) matches the sigmoid relatively closely (see eq. (3)), and the simulation results in the following sections show that this is an effective alternative. To obtain the optimal linear fit of the sigmoid function m = 1/4 and b = 1/2.

$$\sigma^+ = \frac{(\sigma_{\max} - \sigma_{\min})}{\max(|W|)} W^+ + \sigma_{\min} \tag{1}$$

$$\sigma^- = \frac{(\sigma_{\max} - \sigma_{\min})}{\max(|W|)} W^- + \sigma_{\min} \tag{2}$$

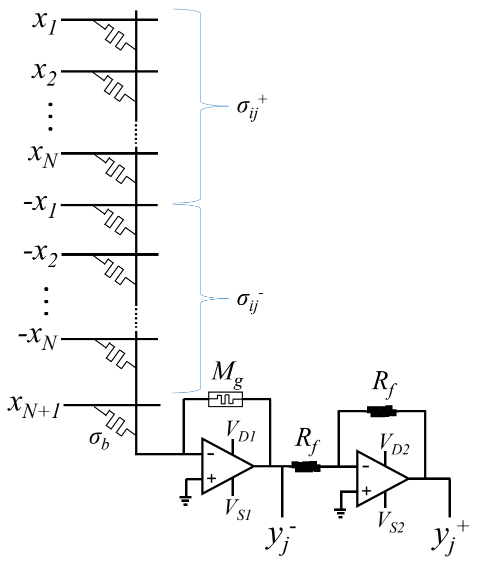


Fig. 1. Memristor based neuron circuit that can be replicated across crossbar columns where V_{D1} = 0, V_{S1} = -1, V_{D2}=1, and V_{S2}=0.

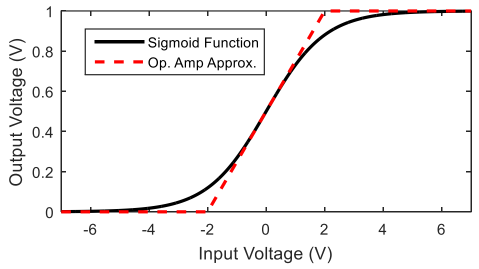


Fig. 2. Comparison of the sigmoid transfer function and the amplifier alternative (the first op-amp stage will actually produce an inverted result, but this will be the eventual outcome seen at the y_j^+ output).

The complete circuit in Fig. 1 will operate according to equations (4) through (6). The op-amp producing the y_j^- output acts as a summing amplifier that also accounts for the slope and bias required of the approximate sigmoid activation function. The y_j^- output is then fed into a unity gain inverting amplifier to obtain the y_j^+ output (see eq. (6)).

$$y_j^+ = \begin{cases} 1, & v > 2 \\ mv + b, & |v| \le 2 \\ 0, & v < -2 \end{cases} \approx \frac{1}{1+e^{-v}} \tag{3}$$

In eq. (4) the output of the summing amplifier is determined. The voltage $x_{N+1} = 1$ and is used to drive the bias value b for the sigmoid function. The resistance R_g is the resistance of the programmable gain memristor M_g. A memristor is used to hold this resistance so that M_g can be set according to equation (5) during evaluation and set to σ_{MAX} during programming. The value σ_b is the conductance of the memristor M_{N+1} and $\sigma_b=b/R_g$. In eq. (5), R_g is set so that the summation of conductance and voltage pairs is multiplied by the inverse of the scaling factor in equations (1) and (2), as well as the slope of the activation function m. Using this method, the resulting outputs y_j^+ and y_j^- will now have a value equal to that of the typical dot product performed in a software implementation after the activation function has been applied. Using this method, the outputs are already formatted for the next layer of the network where y_j^+ becomes the set of inputs for σ_{ij}^+ and y_j^- becomes the set of inputs for σ_{ij}^-.

Fig. 3 shows how repetitions of the circuit in Fig. 1 can be implemented within a memristor crossbar. The output of each crossbar column shows the simplified depiction of the op-amp circuit capable of producing the values y_j^+ and y_j^-.

$$y_j^- = -R_g\left(\sum_{i=1}^{N}\left[x_i\sigma_{ij}^+ - x_i\sigma_{ij}^-\right] + x_{N+1}\sigma_b\right) \tag{4}$$

$$R_g = m\frac{\max(W)}{(\sigma_{\max} - \sigma_{\min})} \tag{5}$$

$$y_j^+ = -\frac{R_f}{R_f}y_j^- = -y_j^- \tag{6}$$

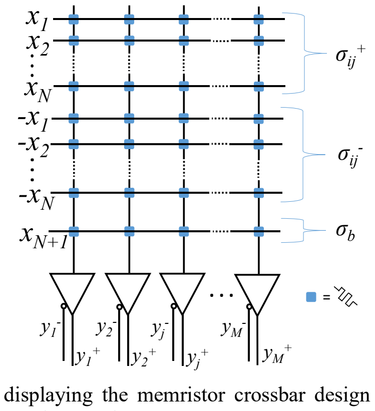


Fig. 3. Diagram displaying the memristor crossbar design for processing a single layer of a neural network.

III. Ex-Situ Training Process

To use the circuit described in the previous section, a memristor crossbar must be capable of being programmed to a set of precise conductance values. This section describes how programming precision will be defined, and will also present the programming circuit designed to perform this task.

A. Memristor Programmability

The idealized memristor device has a continuous bounded programmable resistance range. However, physical memristor device characterizations show stricter limitations in device programmability. Specifically, there appears to be a limit to how many discrete states can be programmed into a memristor device, and there is a certain amount of stochastic resistance change present when attempting to program a device to a specific resistance [18]. For example, work in [57] shows that 128 states can be programmed into a memristor device, and this supports the programming assumptions that we have made in the presented work. We go beyond the work in [57] to show how feedback programming techniques can be used to implement neural network algorithms in crossbar circuits.

For the system described in this paper we assume that there are only a set number of states in the memristor that can be accessed. This keeps the size of the D-to-As required for ex-situ programming at a minimum. There is also a programming precision (α) associated with each memristor state to account for stochastic switching noise. In the studies in this work, the programming precision is set to 0.01 V (this was previously determined as the optimal value for α in this circuit in [18]). This will make programming much easier because a memristor will only have to be programmed to within a set of bounds (instead of an exact resistance value).

B. Crossbar Programming Circuit

The circuit designed to implement the programming method is capable of programming one device at a time. An entire row of devices in the crossbar could be programmed simultaneously, but this would increase the chip area devoted to the programming circuit. We chose the single device approach because the object of this system is to program the crossbar once before it is used in a read mode extensively, so less importance was placed on programming speed.

To program each of the weights in this memristor crossbar, only one row input x_i (where i=1,…,N+1) is active (set to 1V) and all other inputs are set to 0V. This will reduce the summing amplifier to a simple inverting amplifier. Also, the second inverting amplifier that produces y_j^+ can be ignored during the write process. In this circuit, the feedback memristor M_g is held at a constant value σ_{MAX} (or R_{MIN}). Using this approach, when the target memristor being programmed (M_i) is set to σ_{MAX}, the output y_j^-=-1 V. Likewise, when the target memristor M_i is set to σ_{MIN}, the output $y_j^-\approx$0 V.

The schematic in Fig. 4 displays the entire programming circuit in addition to the active part of the memristor crossbar when programming a single memristor. This circuit utilizes two D-to-A converters to implement a bounded voltage range for successful programming. It is assumed that these D-to-A converters have access to the weight values produced using the desired learning algorithm in software. Also, it is assumed that software is able to convert each floating point weight to a value within a set of predefined conductivity states, where the size of this set corresponds to the width of the D-to-A used. This will essentially convert the weights from floating point to much lower resolution values. For the programming process to determine that a memristor is programmed, y_j^- must be greater than τ-α and less than τ+α. The logic at the output of the comparators determines whether the memristor conductivity should be increased or decreased based the output voltage of the two comparators. If resistance is to be decreased, a positive write pulse will be applied to the memristor M_i, and if the resistance is to be increased, a negative pulse will be applied. This process will be repeated until the XOR gate in Fig. 4 has a high output, signifying a successfully programmed device. This iterative process is what forms the feedback programming system. A pulse width known to be much smaller than that required to fully switch the memristor is used to program the memristors to these specific resistances to induce very small amounts of resistance change [7,58].

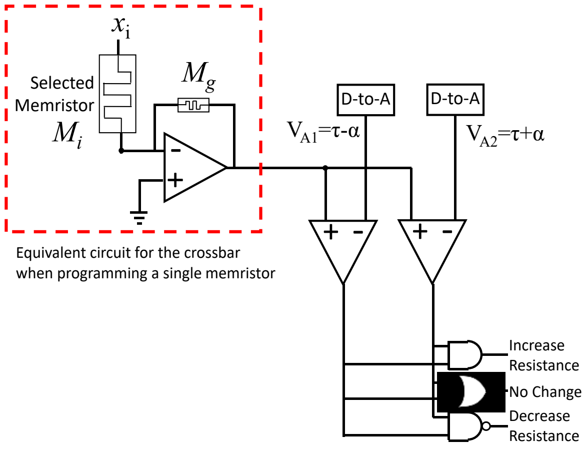


Fig. 4. Circuit used to program a single memristor to a target resistance.

The plot in Fig. 5 shows how the feedback programming process operates. The memristor switching characteristic was implemented with some switching noise added so that it would operate closer to a physical device (such as [59]). Once the op-amp output voltage y_j^- falls within the dashed lines portraying the programming precision bounds, the XOR output becomes high and no more programming pulses are applied to this memristor. In these studies, memristors were assumed to behave according to the model in [60].

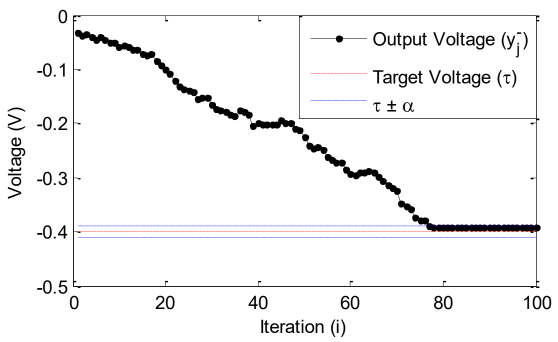


Fig. 5. Example of how the feedback programming circuit will drive a memristor resistance to a value within the programming precision bounds.

IV. MLP Sobel Edge Detection

In this section, the proposed circuit design was used to implement a multilayer perceptron architecture (see Fig. 6 for network layout) to perform Sobel edge detection.

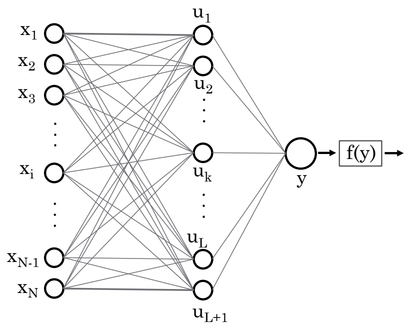


Fig. 6. Diagram for the perceptron used in this study.

A. Part 1: Off-Chip Training

The memristor based neural network is implemented to mimic the operation of a 3×3 edge detection kernel. The network topology that was used in this study was a 10→L+1→1 multi-layer perceptron (see Fig. 6). The 10 inputs to the first layer represent a 3×3 pixel square and a single bias input (therefore N=10). The number of nodes in the hidden layer (L) is varied throughout these experiments to study the impact of network size, where node L+1 is a bias used in all experiments. The output is a single pixel used in part to construct the edge detected output image. The edge detected image will be constructed as each 9-pixel section of the input image is applied to the input layer of the neural net. No zero-padding is applied to the input image, so the output will have 2 less pixels in each direction when compared to the input.

Fig. 7 displays a portion of an image that was input to a Sobel edge detector, and Fig. 7 (b) displays the corresponding output image (determined using a DSP algorithm, not a neural network). To train the neural network, the image in Fig. 7 (b) was input to the MLP system and the error was minimized to fall below a set threshold (either 0.5 or 0.4 in these experiments). In an example case, Fig. 8 shows that the MSE was minimized until MSE was less than 0.5 between Fig. 7 (b) and the neural network output.

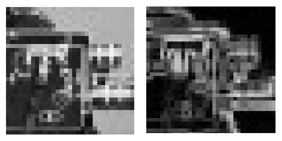


Fig. 7. Portion of an image (30×30 pixels) that was used the train the neural network. Images show (a) the input and (b) the output for the Sobel Edge Detector (based on the DSP algorithm, not neural network approximation).

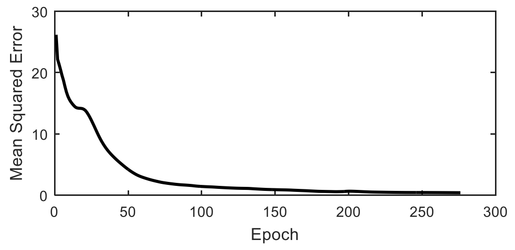


Fig. 8. MSE when training the MLP based on the image in Fig. 7 (b).

B. Part 2: Memristor Circuit Evaluation

After the MLP was trained, the weights were programmed into the simulated memristor crossbar circuit, and the entire image was used to test the network. Fig. 9 displays the output comparisons via example image. The entire input image is displayed in Fig. 9 (a), and the resulting image due to the DSP Sobel filter (not the neural network) can be seen in Fig. 9 (b).

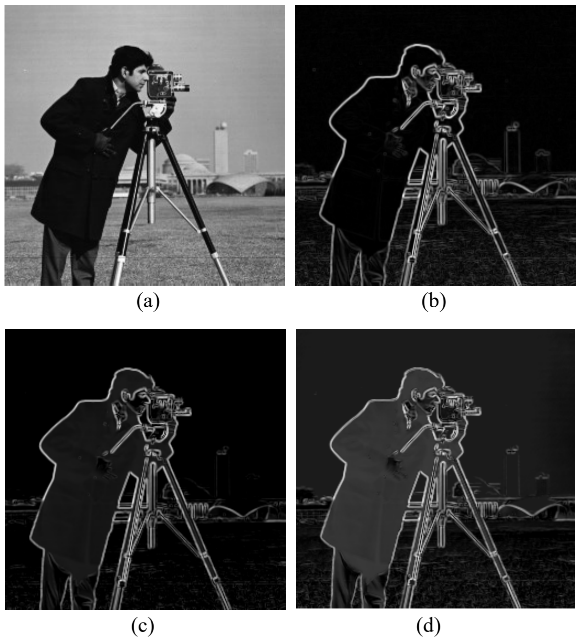


Fig. 9. The images displayed are (a) the original input image, (b) the output image using the Sobel edge detection algorithm, (c) the output image using the memristor-based neural network with a 3-bit D-to-A for programming, and (d) the output image using the memristor-based neural network with a 2-bit D-to-A for programming. In both (c) and (d), α=0.01 V.

Figs. 9 (c) and (d) show the image outputs generated using the memristor neural network. Reducing the bit-width of the D-to-A circuits used during programming reduces the number of possible conductance values to which the memristors can be programmed. As the bit width of the D-to-As is reduced, the MSE in the system increases. Fig. 9 (c) shows the output image obtained from the memristor based neural network after it was programmed using a 3-bit D-to-A. The MSE between Figs. 9 (b) and (c) is 0.0033. Fig. 9 (d) shows the resulting output image when a 2-bit D-to-A is used for programming. Since a fewer number of resistance states can be programmed, the error goes up significantly. The MSE between Figs. 9 (b) and (d) is 0.0112.

It should be noted that when comparing the results from training and evaluation, the MSE during evaluation is much lower. This is because MSE is averaged over all of the pixels in the image, and in this case most of the pixels were solid black and easily reproduced by the hardware. This large amount of dark data brings down the average error when compared to the visually complex training image segment.

V. Design Space Analysis

Now that the initial implementation has been tested, several follow-on simulations were performed using this system with varying perceptron layouts. The studies in this section evaluate the interplay between memristor programming bit width, MSE, energy, power, and timing. This is an extension of work presented in [18].

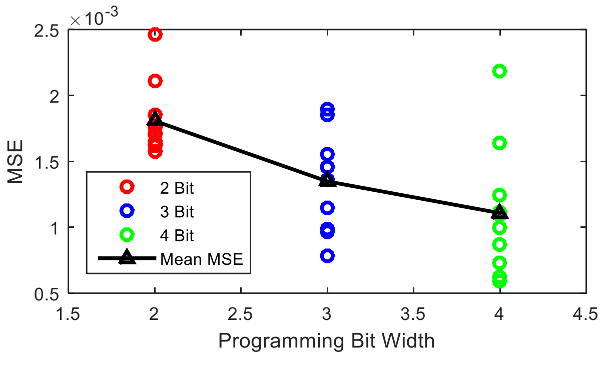
A. Network Layout Analysis

The three plots in Fig. 10 show the resulting MSE for three different network configurations: the first had 20 hidden neurons and a training error threshold of 0.5, the second had 50 hidden neurons and a training error threshold of 0.5, and the third had 50 hidden neurons and a training error threshold of 0.4. The training error threshold was pushed lower because this was observed to improve testing MSE. Alternatively, it is very difficult to achieve a training error of 0.4 with only 20 hidden nodes, so that result is not considered. Beyond this limited design space, it may be possible to further optimize the relationship between network size and training threshold to determine a stronger configuration. Since the memristor system we are proposing uses ex-situ training, all of this network layout, design, and analysis can be done in software before the final optimized weights are simply downloaded into the memristor system.

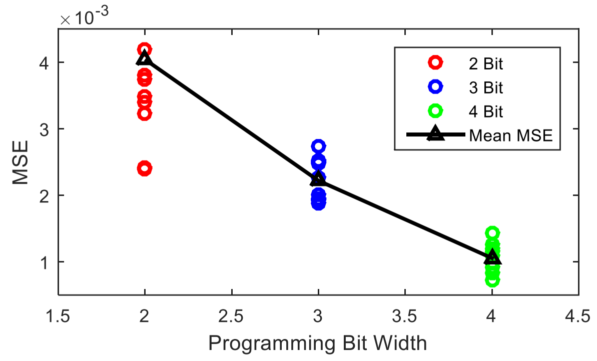
After the network layouts and weight matrices were determined, these weights were programmed into the simulated memristor crossbar system using the ex-situ programming approach in Section III. A programming bit width of either 2, 3, or 4 was used, so the memristors in this system can store either 4, 8, or 16 unique states respectively. Each of the plots in Fig. 10 present a data point designating a mean MSE for each bit width. This mean is based on 10 identical iterations of the programming process in Section III (see the red, blue, and green circles in each plot), these are not 10 different training processes. This variation is simply due to the error that occurs when programming each memristor in the crossbar. Since the memristors start at a high random resistance and they are subject to significant variability during programming, identical results after programming are rarely obtained.

B. Memristor Resolution Analysis

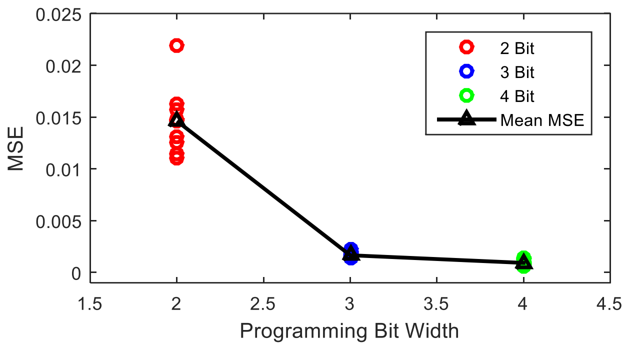
To extend this experiment, we determined how large the network would have to be before the number of programmable states within the memristor devices could be reduced. This was done using the same simulation setup and circuit model described in Section III, except that the number of hidden layers present in the network layout was varied between experiments. The result in Table I displays two different network options along with the corresponding testing MSE for both the 3 and 4-bit programming options. This result shows that when the number of hidden nodes in the system is 100 and the training error threshold is equal to 0.4, the testing error with 3-bit programming is less than the error obtained for 4-bit programming of the system with 20 hidden nodes and a training error threshold of 0.5. This also shows that the size of the system can be increased to reduce the required number of programmable memristor states in the system. In this case, if a MSE less than .0014 is required, either the number of required programmable memristor states can be increased, or the size of the network can be increased. This makes the system more general, as a lower resolution memristor could be used if network size is increased. However, five time as many hidden layer neurons are required to achieve that result.



(a)



(b)



(c)

Fig. 10. Simulation results that show resulting MSE for different programming bit widths and network arrangements. Plots show (a) a network with 20 hidden nodes and a training threshold of 0.5, (b) a network with 50 hidden nodes and a training threshold of 0.5, and (c) a network with 50 hidden nodes and a training threshold of 0.4.

Table I. Results that display the relationship between MSE, network size, and programming bit width.

Number of Hidden Nodes (L)	Training Threshold	Testing MSE: 3 Bits	Testing MSE: 4 Bits
20	0.5	0.00146	0.00124
100	0.4	0.00092	0.00110

C. *Energy, Timing, and Power Analysis*

In the previous subsection, a relationship between memristor resolution and network size has been determined. Thus, we can now use the data in this section to determine a relationship between memristor resolution and power consumption. Tables II and III display the energy and power requirements during evaluation of the two network layouts compared in Table I (power and energy analysis of the system during training will be carried out in future work).

To generate the data in Tables II and III, a small scale SPICE simulation of the circuits used to develop the neural network layouts were used. The memristors were simulated according to the model in [60,61], and the op-amps in the column amplifiers were designed starting from the circuits in [62,63] using 180 nm technology. It is assumed that the memristors used in this study have a minimum resistance of 125 kΩ and a maximum resistance of 125 MΩ. The average energy consumption of this circuit for a 9-pixel evaluation was considered to be the mean energy generated from four simulations with different sample inputs. Average energy consumption values were then determined for a single memristor, and a single column amplifier (which consists of two op-amp circuits), and multiplied to determine the energy consumption of the entire network layout. Table II shows that the large network layout capable of achieving desired MSE with lower resolution memristors requires approximately 4.2 times the energy to operate. Therefore, a tradeoff exists between memristor resolution, MSE, and energy consumption.

Table II. Energy consumption breakdown of the two memristor based MLP structures described in Table I.

Attribute	Energy Consumption System 1 20 Hidden Neurons 442 Memristors 21 Amplifiers	Energy Consumption System 2 100 Hidden Neurons 1202 Memristors 101 Amplifiers
Avg. Memristor Energy	400 fJ	400 fJ
Avg. Column Amplifier Energy	23.81 pJ	23.81 pJ
Total Memristor Energy	176.98 pJ	481.28 pJ
Total Amplifier Energy	500.04 pJ	2404.97 pJ
Energy to Generate One Pixel	677.02 pJ	2886.25 pJ
Energy to Process Image in Fig. 9	43.68 µJ	186.21 µJ

Given that the column amplifier circuit has a resolve time of about 100 ns, we assume the maximum operating frequency of this system is 10 MHz. Therefore, if real-time video processing is required, a single circuit can process at most 10 million pixels per second. Using this limit, we can determine how many copies of the memristor based neuromorphic image processing circuit are required to process real-time video of different sizes and speeds (see Table III). The power estimates in Table III only account for the processing circuit described in this work and do not account for clocking, bit-shifting, or data splitting. Power consumption is likely to increase if this circuit is implemented as part of a larger system.

Table III. Power consumption and throughput breakdown of the two memristor based MLP structures described in Table I.

Application	Circuits Required for Real Time Processing	Power (mW) System 1 20 Hidden Neurons 442 Memristors 21 Amplifiers	Power (mW) System 2 100 Hidden Neurons 1202 Memristors 101 Amplifiers
Low Resolution Video (640×480 Pixels 24 FPS)	1	4.99	21.28
HD Ready Video (1280×720 Pixels 30 FPS)	3	18.72	79.80
Full HD Video (1920×1080 Pixels 30 FPS)	7	42.12	179.55
4K UHD Video (3840×2160 Pixels 60 FPS)	50	336.93	1436.38

VI. Conclusion

This ex-situ system is capable of programming a memristor crossbar based on a set of quantized weights generated using a traditional software implementation of a given learning algorithm. Each memristor crossbar is capable of reproducing accurate dot product outputs, so values can be transmitted between network layers correctly. Using this system, we were able to show successful neural network based digital image processing at high throughput and low power. The design space analysis in this work details the relationships between network size, MSE, memristor resolution, energy, power, and timing. Results show that 4K UHD video can be processed in real time at a power consumption of approximately 337mW if 50 replicas of the neural network evaluation circuit are used simultaneously. Likewise, utilizing a memristor device that is only capable of storing 8 states instead of 16 is likely to cause a 4.2× increase in the processing energy required.

Future work includes scaling the system to see how an *N*-layer network will perform. Furthermore, the results presented in this work are based on the edge detection of a single image. Moving forward, we want to compare these results to neural implementations of several other DSP applications to see if any other patterns emerge when tuning programming parameters. We also plan to study how this design performs on a physical memristor crossbar.

References

[1] T. M. Taha, R. Hasan, C. Yakopcic, and M. R. McLean, "Exploring the Design Space of Specialized Multicore Neural Processors," IEEE International Joint Conference on Neural Networks (IJCNN), pp. 1-8, Dallas, TX, Aug. 2013.

[2] B. Belhadj, A. J. L. Zheng, R. Héliot, and O. Temam. "Continuous real-world inputs can open up alternative accelerator designs," ACM/IEEE International Symposium on Computer Architecture (ISCA), Tel-Aviv, Israel, June, 2013.

[3] P. Dubey, "Recognition, mining and synthesis moves computers to the era of tera," Technology@Intel Magazine, Feb. 2005.

[4] T. Chen, Y. Chen, M. Duranton, Q. Guo, A. Hashmi, M. Lipasti, A. Nere, S. Qiu, M. Sebag, O. Temam, "BenchNN: On the Broad Potential Application Scope of Hardware Neural Network Accelerators," IEEE International Symposium on Workload Characterization (IISWC), pp. 36-45, San Diego, CA, November 2012.

[5] L. O. Chua, "Memristor—The Missing Circuit Element," IEEE Transactions on Circuit Theory, vol. 18, no. 5, pp. 507–519, Jan. 1971.

[6] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing Memristor found," Nature, 453, pp. 80–83, Oct. 2008.

[7] C. Yakopcic and T. M. Taha, "Determining optimal switching speed for memristors in a neuromorphic system," Electronics Letters, vol. 51 no. 21, pp. 1637-1639, October, 2015.

[8] C. Yakopcic, T. M. Taha, and M. R. McLean, "Method for ex-situ training in a memristor-based neuromorphic circuit using a robust weight programming method," Electronics Letters, vol. 51, no. 12, pp. 899-900, June, 2015.

[9] C. Yakopcic, T. M. Taha, G. Subramanyam, and R. E. Pino, "Impact of Memristor Switching Noise in a Neuromorphic Crossbar" IEEE National Aerospace and Electronics Conference (NAECON), pp. 320-326, Dayton, OH, June, 2015.

[10] C. Yakopcic, S. Wang, W. Wang, E. Shin J. Boeckl, G. Subramanyam, and T. M. Taha, "Filament Formation in Lithium Niobate Memristors Supports Neuromorphic Programming Capability" Neural Computing and Applications (Forthcoming).

[11] S. Wang, W. Wang, C. Yakopcic, E. Shin, G. Subramanyam and T. M. Taha, "Experimental study of LiNbO3 memristors for use in neuromorphic computing," Microelectronic Engineering, vol. 168, pp. 37–40, Jan. 2017.

[12] S. Wang, W. Wang, C. Yakopcic, E. Shin, G. Subramanyam and T. M. Taha, "Reconfigurable Neuromorphic Crossbars Based on Titanium Oxide Memristors," Electronics Letters, vol. 52, no. 20, pp. 1673-1675, Sept. 2016.

[13] C. Yakopcic, S. Wang, W. Wang, E. Shin, G. Subramanyam and T. Taha, "Methods for High Resolution Programming in Lithuim Niobate Memristors for Neuromorphic Hardware," IEEE/INNS International Joint Conference on Neural Networks (IJCNN), pp. 1704-1708, Anchorage, AK, May 2017.

[14] S. H. Jo, K.-H. Kim, and W. Lu, "High-Density Crossbar Arrays Based on a Si Memristive System" Nano Letters, vol. 9, no. 2, pp. 870-874, Jan. 2009.

[15] C. Yakopcic and T. M. Taha, "Model for maximum crossbar size based on input driver impedance," Electronics Letters, vol. 52 no. 1, pp. 25-27, Jan. 2016.

[16] C. Yakopcic, R. Hasan, and T. M. Taha, "Hybrid Crossbar Architecture for a Memristor Based Cache," Microelectronics Journal, vol. 46, no. 11, pp. 1020-1032, November, 2015.

[17] G. S. Snider, "Cortical computing with memristive nanodevices," SciDAC Review, Winter 2008, pp. 58–65. Available at http://www.scidacreview.org/0804/pdf/hardware.pdf

[18] C. Yakopcic, R. Hasan, and T. M. Taha, "Flexible Memristor Based Neuromorphic System for Implementing Multi-layer Neural Network Algorithms," International Journal of Parallel, Emergent and Distributed Systems, vol. 33, no. 4, pp. 408-429, Aug. 2018.

[19] C. Yakopcic, Z. Alom and T. Taha, "Extremely Parallel Memristor Crossbar Architecture for Convolutional Neural Network Implementation," IEEE/INNS International Joint Conference on Neural Networks (IJCNN), pp. 1696-1703, Anchorage, AK, May 2017.

[20] R. Hasan, T. M. Taha, and C. Yakopcic, "On-chip Training of Memristor Crossbar Based Multi-layer Neural Networks," Microelectronics Journal, vol. 66, pp. 31-40, Aug. 2017.

[21] R. Hasan, T. Taha and C. Yakopcic, "On-chip Training of Memristor Based Deep Neural Networks," IEEE/INNS International Joint Conference on Neural Networks (IJCNN), pp. 3527-3534, Anchorage, AK, May 2017.

[22] R. Hasan, T. Taha, C. Yakopcic, and D. Mountain, "High Throughput Neural Network based Embedded Streaming Multicore Processors," IEEE International Conference on Rebooting Computing (ICRC), pp. 1-8, San Diego, CA, Oct. 2016.

[23] R. Uppala, C. Yakopcic, and T. M. Taha, "Methods for Reducing Memristor Crossbar Simulation Time" IEEE National Aerospace and Electronics Conference (NAECON), pp. 312-319, Dayton, OH, June, 2015.

[24] R. Hasan, C. Yakopcic, and T. M. Taha, "Ex-situ Training of Dense Memristor Crossbar for Neuromorphic Applications," IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), pp. 75-81, Boston, MA, July 2015.

[25] C. Yakopcic and T. M. Taha, "Ex-Situ Programming in a Neuromorphic Memristor Based Crossbar Circuit" IEEE National Aerospace and Electronics Conference (NAECON), pp. 300-304, Dayton, OH, June, 2015.

[26] T. M. Taha, R. Hasan, and C. Yakopcic, "Memristor Crossbar Based Multicore Neuromorphic Processors," IEEE International SOCC, pp. 383-389, Las Vegas, NV, 2014.

[27] C. Yakopcic, R. Hasan, T. M. Taha, M. McLean, and D. Palmer, "Memristor-based neuron circuit and method for applying a learning algorithm in SPICE," IET Electronics Letters, vol. 50, no. 7, pp. 492-494, April 2014.

[28] C. Yakopcic and T. M. Taha, "Energy efficient perceptron pattern recognition using segmented memristor crossbar arrays," IEEE International Joint Conference on Neural Networks (IJCNN), pp. 1–8, Dallas, TX, Aug. 2013.

[29] C. Yakopcic, R. Hasan, and T. M. Taha, "Memristor Based Neuromorphic Circuit for Ex-Situ Training of Multi-Layer Neural

Network Algorithms," IEEE International Joint Conference on Neural Networks (IJCNN), pp. 1-7, Killarney, Ireland, July 2015.

[30] D. Soudry, D. D. Castro, A. Gal, A. Kolodny, and S. Kvatinsky, "Memristor-Based Multilayer Neural Networks With Online Gradient Descent Training," IEEE Trans. on Neural Networks and Learning Systems, vol. 26, no. 10, pp. 2408-2421, Oct. 2015.

[31] Boxun Li; Yuzhi Wang; Yu Wang; Chen, Y.; Huazhong Yang, "Training itself: Mixed-signal training acceleration for memristor-based neural network," 19th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 361-366, Makuhari, Japan, Jan. 2014.

[32] D. Chabi, W. Zhao, D. Querlioz, J.-O. Klein, "Robust Neural Logic Block (NLB) Based on Memristor Crossbar Array" IEEE/ACM International Symposium on Nanoscale Architectures, pp.137-143, San Diego, CA, June, 2011.

[33] C. Zamarreño-Ramos, L. A. Camuñas-Mesa, J. A. Pérez-Carrasco, T. Masquelier, T. Serrano-Gotarredona, and B. Linares-Barranco, "On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex," Frontiers in Neuroscience, Neuromorphic Engineering, vol. 5, Article 26, pp. 1-22, Mar. 2011.

[34] F. Alibart, E. Zamanidoost, and D.B. Strukov, "Pattern classification by memristive crossbar circuits with ex-situ and in-situ training", Nature Communications, 4(2072), pp. 1-7, June, 2013.

[35] R. Hasan and T. M. Taha, "Enabling Back Propagation Training of Memristor Crossbar Neuromorphic Processors," IEEE International Joint Conference on Neural Networks, pp. 21-28,Beijing, China, July, 2014.

[36] M. Hu, H. Li, Q. Wu, G.S. Rose, and Y. Chen, "Memristor crossbar based hardware realization of BSB recall function," International Joint Conference on Neural Networks (IJCNN), pp. 1-7, Brisbane, Australia, June, 2012

[37] J. H. Lee and K. K. Likharev, "Defect-tolerant nanoelectronic pattern classifiers," International Journal of Circuit Theory and Applications, vol. 35, no. 3, pp. 239–264, May, 2007.

[38] D. Chabi, D. Querlioz, W. Zhao, and J. Klein, "Robust learning approach for neuro-inspired nanoscale crossbar architecture," ACM Journal on Emerging Technologies in Computing Systems (JETC), vol. 10, no. 1, pp. 1-20, Jan. 2014.

[39] D. Querlioz, O. Bichler, and C. Gamrat, "Simulation of a memristor based spiking neural network immune to device variations," IEEE International Joint Conference on Neural Networks, pp. 1775–1781, San Jose, CA, USA, July 2011.

[40] M. Hu, H. Li, Y. Chen, Q. Wu, G. S. Rose, and R. W. Linderman, "Memristor crossbar-based neuromorphic computing system: A case study," IEEE Trans. On Neural Networks and Learning Systems, vol. 25, no. 10, pp. 1864–1878, Oct. 2014.

[41] J. Starzyk and Basawaraj, "Memristor crossbar architecture for synchronous neural networks," IEEE Transactions on Circuits and Systems I, vol. 61, pp. 2390 – 2401, Aug. 2014.

[42] P. Sheridan, W. Ma, and W. Lu, "Pattern recognition with memristor networks," IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1078–1081, Melbourne, Austrailia, June, 2014.

[43] Y. Kim, Y. Zhang, and P. Li, "A digital neuromorphic vlsi architecture with memristor crossbar synaptic array for machine learning," IEEE International SOC Conference (SOCC), pp. 328–333, Niagra Falls, NY, Sept. 2012.

[44] A. M. Sheri, A. Rafique, W. Pedrycz, M. Jeon, "Contrastive divergence for memristor-based restricted Boltzmann machine" Engineering Applications of Artificial Intelligence, vol. 37, pp. 336-342, Jan. 2015.

[45] I. Kataeva, F. Merrikh-Bayat, E. Zamanidoost and D. Strukov, "Efficient Training Algorithms for Neural Networks Based on Memristive Crossbar Circuits", IEEE International Joint Conference on Neural Networks (IJCNN), pp. 1-8, Killarney, Ireland, July 2015.

[46] J. Schemmel, J. Fieres, K. Meier, "Wafer-Scale Integration of Analog Neural Networks", IEEE International Joint Conference on Neural Networks (IJCNN), Hong Kong, China, June, 2008.

[47] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, D. S. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm," IEEE Custom Integrated Circuits Conference (CICC), pp.1-4, San Jose, CA, Sept. 2011.

[48] J. V. Arthur, P. A. Merolla, F. Akopyan, R. Alvarez, A. Cassidy, S. Chandra, S. K. Esser, N. Imam, W. Risk, D. B. D. Rubin, R. Manohar, D. S. Modha, "Building block of a programmable neuromorphic substrate: A digital neurosynaptic core," IEEE International Joint Conference on Neural Networks (IJCNN), pp.1-8, Brisbane, Australia, June 2012.

[49] Y. Shim, A. Sengupta, K. Roy, "Low-Power Approximate Convolution Computing Unit with Domain-Wall Motion Based "Spin-Memristor" for Image Processing Applications," 53nd ACM/EDAC/IEEE Design Automation Conference (DAC), August, 2016.

[50] C. Yakopcic, M. Z. Alom, T. M. Taha, "Memristor Crossbar Deep Network Implementation Based on a Convolutional Neural Network" IEEE/INNS International Joint Conference on Neural Networks, pp. 963 – 970, July, 2016.

[51] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, V. Srikumar, "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," ACM/IEEE 43rd Annual International Symposium on Computer Architecture, pp. 14-26, June, 2016.

[52] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y, Xie, "PRIME: A Novel Processing-in-memory Architecture for Neural Network Computation in ReRAM-based Main Memory," ACM/IEEE 43rd Annual International Symposium on Computer Architecture, pp. 27-39, June, 2016.

[53] W. Yi, F. Perner, M. S. Qureshi, H. Abdalla, M. D. Pickett, J. J. Yang, M.-X. M. Zhang, G. Medeiros-Ribeiro, R. S. Williams, "Feedback write scheme for memristive switching devices," Appl Phys A (2011) 102: 973–982, DOI 10.1007/s00339-011-6279-2.

[54] S. Yu, Y. Wu, and H.-S. P. Wong, "Investigating the switching dynamics and multilevel capability of bipolar metal oxide resistive switching memory," Applied Physics Letters 98, 103514 (2011).

[55] C. Yakopcic and T. M. Taha, "Memristor Crossbar Based Implementation of a Multilayer Perceptron" IEEE National Aerospace and Electronics Conference (NAECON), pp. 38-43, Dayton, OH, June 2017.

[56] C. Yakopcic and T. M. Taha, "Analysis and Design of Memristor Crossbar Based Neuromorphic Intrusion Detection Hardware," IEEE/INNS International Joint Conference on Neural Networks (IJCNN), pp. 1-7, Rio de Janeiro, Brazil, July, 2018.

[57] F. Alibart, L. Gao, B. Hoskins, and D.B. Strukov, "High-precision tuning of state for memristive devices by adaptable variation-tolerant algorithm", Nanotechnology vol. 23, art. 075201, pp. 1-7, Jan. 2012

[58] C. Yakopcic, R. Hasan, T. M. Taha, and D. Palmer, "SPICE Analysis of Dense Memristor Crossbars for Low Power Neuromorphic Processor Designs" IEEE National Aerospace and Electronics Conference, pp. 305 – 311, Dayton, OH, June, 2015.

[59] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," Nano Letters, vol. 10, pp. 1297–1301, Mar. 2010.

[60] C. Yakopcic, T. M. Taha, G. Subramanyam, R. E. Pino, and S. Rogers, "A Memristor Device Model," IEEE Electron Device Letters, vol. 30, no. 10, pp. 1436-1438, Oct. 2011.

[61] C. Yakopcic, T. M. Taha, G. Subramanyam, and R. E. Pino, "Generalized Memristive Device SPICE Model and its Application in Circuit Design," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 32, no. 8, pp. 1201-1214, August, 2013.

[62] A. Yadav, "Design of Two-Stage CMOS Op-Amp and Analyze the Effect of Scaling," International Journal of Engineering Research and Applications, vol. 2 no. 5, pp. 647-654, Sept.-Oct. 2012.

[63] Siddarth, M. Garg, A. Gahlaut, "Comparative Study of CMOS Op-Amp in 45nm and 180nm Technology," Journal of Engineering Research and Applications, vol. 4, no. 7, pp. 64-67, July, 2014.