Representable Matrices: Enabling High Accuracy Analog Computation for Inference of DNNs using Memristors

Baogang Zhang*, Necati Uysal*, Deliang Fan[†] and Rickard Ewetz*
*University of Central Florida, Orlando, FL, 32816, USA
[†]Arizona State University, Tempe, AZ, 85281, USA
baogang.zhang@knights.ucf.edu, necati@knights.ucf.edu, dfan@asu.edu, rickard.ewetz@ucf.edu

Abstract—Analog computing based on memristor technology is a promising solution to accelerating the inference phase of deep neural networks (DNNs). A fundamental problem is to map an arbitrary matrix to a memristor crossbar array (MCA) while maximizing the resulting computational accuracy. The state-of-the-art mapping technique is based on a heuristic that only guarantees to produce the correct output for two input vectors. In this paper, a technique that aims to produce the correct output for every input vector is proposed, which involves specifying the memristor conductance values and a scaling factor realized by the peripheral circuitry. The key insight of the paper is that the conductance matrix realized by an MCA is only required to be proportional to the target matrix. The selection of the scaling factor between the two regulates the utilization of the programmable memristor conductance range and the representability of the target matrix. Consequently, the scaling factor is set to balance precision and value range errors. Moreover, a technique of converting conductance values into state variables and vice versa is proposed to handle memristors with non-ideal device characteristics. Compared with the state-of-the-art technique, the proposed mapping results in 4X-9X smaller errors. The improvements translate into that the classification accuracy of a seven-layer convolutional neural network (CNN) on CIFAR-10 is improved from 20.5% to 71.8%.

I. INTRODUCTION

Deep neural networks (DNNs) have in recent years achieved remarkable results in terms of image, audio, and video recognition [6]. The arising solution to enable the computationally heavy DNNs to be deployed on edge-devices in the Internet of Things is to leverage memristor-based technology. Memristor crossbar arrays (MCAs) can perform matrix-vector multiplication in the analog domain with orders of magnitude smaller power and latency than in the digital domain [5], [4]. Moreover, the use of MCAs allow matrices to be stored in-place, which reduces data fetching and communication costs that fundamentally bounds the performance of any computing system that processes large amounts of data [8], [2], [9].

Matrix-vector multiplication is performed using an MCA with access transistors by first programming the conductance values of the memristors to realize a conductance matrix G, which is illustrated in Figure 1(a) [3]. Next, an input vector of voltages (v_{in}) are applied to the vertical columns and a vector of output voltages (v_{out}) are measured from the horizontal rows. The inputs are provided to the MCA using digital-to-analog converters (DACs) and the outputs are converted
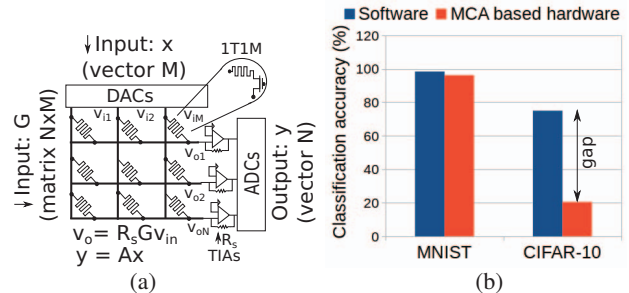
Fig. 1. (a) An MCA used for matrix-vector multiplication. (b) Classification accuracy in software and MCA based hardware on MNIST and CIFAR-10 using the mapping in [5].

into digital values using transimpedance amplifiers (TIAs) and analog-to-digital converters (ADCs). The output voltages v_o are equal to $R_s G v_i$, where R_s is the feedback resistances of the TIAs [3], [5]. Next, the output voltages are scaled into digital values. However, if a weight matrix is mapped to an MCA without considering effects as IR-drop, programming errors, and non-ideal device characteristics, the computational accuracy will be degraded into noise [5]. In particular, the accuracy is degraded by the IR-drop across the non-zero input, output and wire resistance in the MCA.

Techniques to map an arbitrary matrix W to an MCA have been studied in [3], [12], [7], [5]. The conductance matrix G realized by a set of memristor conductance values g can be determined analytically using Modified Nodal Analysis (MNA) [7]. Next, the effective matrix realized by an MCA (W^r) is obtained by scaling G with a factor ($1/\alpha$), which is realized by the peripheral circuitry. In [7], the conductance values g were determined by minimizing the square of the Frobenius norm of $(W - W^r)$ using steepest gradient decent. However, the method is unable to consistently converge for arbitrary matrices. In [5], a technique of tunning conductance values (or state variables of non-ideal memristor devices) to minimize $||(W - W^r) \cdot v_{cal}||^2$ using Newton's method was proposed, where v_{cal} is a calibration vector. The main limitation of these works is that the scaling factor α was not explicitly optimized. Nevertheless, the technique in [5] enabled a five-layer feed-forward neural network to be mapped to a memristor based platform while achieving software level accuracy on the MNIST dataset. However, when the technique is used to map a seven-layer convolutional neural network (CNN) to a MCA based platform, the classification accuracy drops from 75.2% to 20.5%, which is shown in Figure 1.

In this paper, a technique is proposed to map an arbitrary matrix W into a scaling factor α and memristor state variables s. The main innovations of the paper are summarized, as follows:

- The problem of specifying (s, α) is converted into a problem of specifying (g, α). The technique is based on replacing each series connected memristor and access transistor with an ideal conductor. After the conductance values g have been determined, Newton's method is used to obtain the equivalent state variables s.

- The memristor conductance values g are only required to be specified to realize a conductance matrix G that is proportional to the matrix W. Next, the conductance matrix is effectively scaled with $1/\alpha$ such that W is effectively realized. Nevertheless, the utilization of the memristor conductance range is also regulated by the scaling factor α. If α is set too small, the errors will be dominated by the limited *precision* of the memristors. If α is set too large, the errors will be dominated by *value errors* introduced by IR-drop. In particular, it is impossible to represent small and large values in the far-end of an MCA because of IR-drop. We refer to the insights of the matrices that can be realized at different locations in an MCA as defining the space of representable matrices. Consequently, α is specified while balancing precision and value range errors.

- Given α, the conductance values g are specified minimizing $||(W - W^r)||_F^2$ using steepest gradient decent, where $||(W - W^r)||^2$ is called the total errors. After it is impossible to further reduce the total errors, the matrix W is updated into a new target matrix W^t in order to ensure that $||(W - W^r) \cdot v_{cal}||^2 = 0$, where v_{cal} is a calibration vector selected from the input vector space.

- The experimental results show that the proposed mapping technique results matrix-vector multiplication with $4X$-$9X$ higher computational accuracy compared with in [5]. The improvements translate into that the classification accuracy of a seven-layer CNN is improved from 20.5% to 71.8% on CIFAR-10, which is close to the software accuracy of 75.2%.

II. PRELIMINARIES

A. Circuit model of MCAs [7], [5]

Figure 2 shows two circuit models for the MCAs in Figure 1. In Figure 2(a), the memristors and the access transistors are modeled using non-linear equations. The current $i_m(s, v_m)$ through a memristor is a non-linear function of the state variable s and the voltage v_m across the device. The current through each access transistor $i_t(v_s, v_d, v_g)$ is a non-linear function of the source, drain, and gate voltage. A simplified circuit model of the MCA when both the memristors and the access transistors are treated as ideal devices is shown in Figure 2(b), which allows them to be replaced with a single ideal memristor g (or an conductor with lower and upper bounds).

B. Matrix realized by an MCA

The matrix W^r realized by an MCA is a function of g and α, i.e., $W^r = f(g, \alpha)$. For the circuit model in Figure 2(b), the
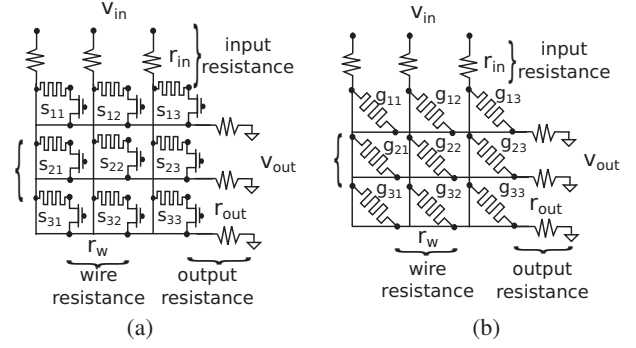

Fig. 2. MCA with (a) non-ideal and (b) ideal devices.

realized matrix W^r can be obtained analytically, as follows:

$$W^r = G/\alpha \tag{1}$$

where G is the conductance matrix realized by the resistive network and and α is an arbitrary scaling factor realized by the peripheral circuitry. The conductance matrix G is a non-linear function of the memristor conductance values g. G is obtained by formulating a system on linear equations that capture the resistive network using MNA, as follows:

$$Y(g) \begin{bmatrix} v \\ v_{dac} \end{bmatrix} = \begin{bmatrix} 0 \\ v_{in} \end{bmatrix}, \tag{2}$$

where $Y(g)$ is a matrix with dimensions $(2NM + M) \times (2NM + M)$ that is a function of g. M and N are the number of inputs and outputs, respectively. v and v_{dac} are respectively the node voltages of the MCA and DACs; v_{in} is the input voltages. Next, G is obtained analytically, as follows:

$$G = SY^{-1}(g)B, \tag{3}$$

where $B = [0, 0, I]^T$ and I is an $M \times M$ identity matrix. S is an $N \times (2 \cdot N \cdot M + M)$ matrix that selects the output voltages from $Y^{-1}B$.

C. Problem formulation

This paper considers the problem of specifying the memristor state variables s and the scaling factor α that maximizes the computational accuracy when performing matrix-vector multiplication using an MCA. If the memristor devices are ideal, the problem consists of specifying the conductance values g and the scaling factor α. The computational accuracy is evaluated while accounting for that memristors only can be programmed to a limited number of distinguishable states [1], [4]. In the experimental results, the impact of the proposed mapping technique is also evaluated in terms of classification accuracy when DNNs trained in software are mapped to an MCA based platform for inference.

The problem is approached by first converting problem of specifying state variables and a scaling factor (s, α) into a problem of specifying memristor conductance values and a scaling factor (g, α). This enables g to be specified while minimizing $||(W - W^r)||^2$ or $||(W - W^r) \cdot v_{cal}||$. Lastly, the conductance values g are converted back into state variables s.
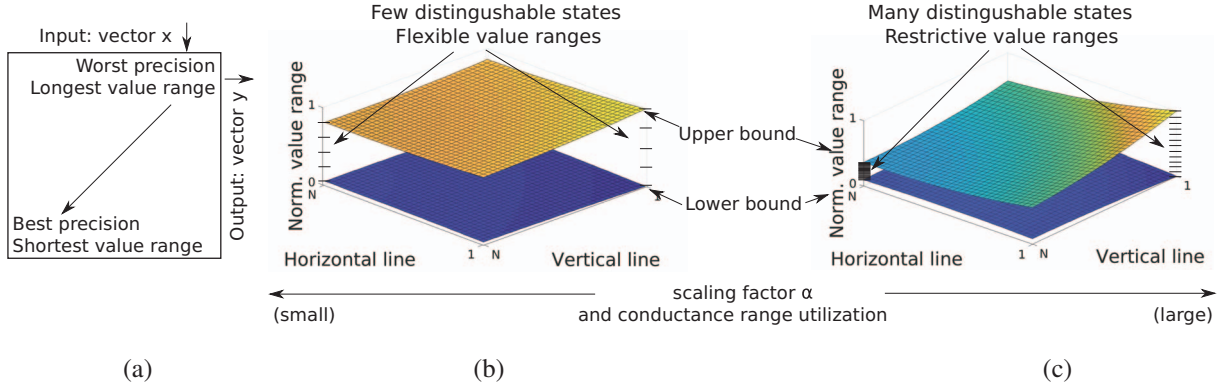
Fig. 3. (a) The value range and precision of an matrix element is dependent on the location in the MCA. (b) and (c) shows a trade-off between the value ranges and the precision based on the scaling factor α.

III. PREVIOUS WORK

A. Specification of conductance values g in [7]

In [7], α was fixed and the conductance values g were determined by formulating an optimization problem, where the square of the Frobenius norm of $(W - W^r)$ was minimized, as follows:

$$\min F(g, \alpha) = ||W - W^r||^2 = \sum_{i=1}^{N} \sum_{j=1}^{M} (w_{ij} - w_{ij}^r)^2, \quad (4)$$

where $||.||^2$ is the square of the Frobenius norm. w_{ij} and w_{ij}^r are the elements in row i and column j in the weight matrix W and the realized matrix W^r, respectively. The function F is minimized using steepest gradient decent, as follows:

$$g_{k+1} = g_k + tF(g)$$
$$= g_k + t \cdot \sum_{i=1}^{N} \sum_{j=1}^{M} 2 \cdot (w_{ij} - w_{ij}^r) \cdot \frac{\partial w_{ij}^r}{\partial g}, \quad (5)$$

where $\frac{\partial w_{ij}^r}{\partial g}$ is the derivative of w_{ij} with respect to g. t is the step size, which is determined using a linear search. g_0 is equal to W linearly mapped into the memristor conductance range. Iterative tuning is performed to compensate for IR drop in the MCA. The main limitations is that the method is unable to consistently converge to solutions with high accuracy because α was fixed.

B. Specification of state variables s in [5]

In [5], a technique of specifying the state variables of the memristors s was proposed. The matrix W is first linearly mapped into the programmable memristor conductance range to obtain an ideal conductance matrix G_{ideal}. The ideal current (i_{ideal}) through each memristor device is obtained using G_{ideal} and an input calibration vector v_{cal}, which also implicitly defines the scaling factor α. Next, MNA is used to formulate a system of $(4NM+2N+2M) \times (4NM+2N+2M)$ equations to capture the circuit model in Figure 2(a). NM of the equations are used to force the currents through each memristor to be equal to i_{ideal} and the remaining equations are used to capture the behavior of the circuit. The state variables s are determined by solving the system of equations using Newton's method. If Newton's algorithm does not converge, i_{ideal} is updated to ensure that the full programmable conductance range was utilized. The limitation is that only the zero input

vector ($\bar{0}$) and the calibration vector (v_{cal}) are guaranteed to produce the correct output, i.e., $||(W - W^r) \cdot \bar{0}||^2 = 0$ and $||(W - W^r) \cdot v_{cal}||^2 = 0$.

IV. SPACE OF REPRESENTABLE MATRICES

In this section, we define the space of matrices that are representable using an MCA and analyse the impact of the scaling factor α. The observations motivates our proposed mapping technique.

The *value range* and *precision* for each matrix element is dependent on the *location* in the MCA and the scaling factor α, which is illustrated in Figure 3. Based on Eq (1) and Eq (3), the scaling factor α directly regulates the utilization of the programmable conductance range, i.e., a larger α implies a utilization of larger conductance values. The value range for a matrix element consists of a lower and upper bound on the value that can be realized. The upper bound mainly stems from IR-drop. The lower bound stems from that currents may flow from an vertical line i to an horizontal line j even if the memristor device connecting vertical line i to horizontal line j is set to be non-conductive (maximum resistance), i.e., the current would flow on paths in the MCA containing more than one memristor. The length of the value range is the longest in the top-right corner and the shortest in the bottom-left corner of an MCA. Moreover, there is an equal number of distinguishable states between every lower and upper bound. The number of states is dependent on the accuracy of the closed loop programming and the selected utilization of the programmable conductance range. Consequently, the worst (best) precision is obtained for value range's with the longest (shortest) length, which is illustrated in Figure 3(a). Moreover, by reducing the utilization of the conductance range, every value range becomes more flexible at the expense of worse precision because the number of distinguishable states within each value range is reduced, which is illustrated in (b-c) of Figure 3. The explanation is that utilization of high conductive increases the currents on the paths with multiple memristors and IR-drop.

The described observations directly explain the space of matrices that are representable using an MCA, i.e., every matrix can be realized using an MCA but the computational accuracy depends on the sum of the value range errors and the precision errors (called total errors). Value range errors occur when matrix elements are attempted to be realized outside

their respective value ranges. The precision errors depend profoundly on how large portion of the programmable memristor conductance range is utilized. Consequently, a critical problem is to specify the scaling factor α to balance the value range and the precision errors, which is shown in Figure 4.
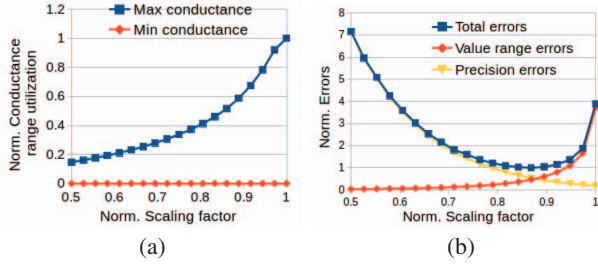

(a) (b)

Fig. 4. (a) Memristor conductance range utilization vs. scaling factor α. (b) Total errors, value range errors, precision errors vs. scaling factor α. The results are obtained for an MCA with dimensions 64x64.

In Figure 4(a), it is shown that the utilization of the programmable memristor conductance range is dependent on α. In Figure 4(b), the trade-off between value range errors and precision errors is shown based on α. If α is selected too small, the value range errors will neglectable but large precision errors will be introduced. If α is selected to large, the precision errors will be neglectable but large value range errors will be introduced. Hence, α should be selected so only a few values are slightly outside the value ranges such that the algorithm in [7] can be used to specify the conductance values by minimizing $||W - W^r||$.

V. PROPOSED METHODOLOGY

We propose a five step flow to map an arbitrary target matrix W to an MCA which is shown in Figure 5. The first step consists of replacing each non-ideal memristor and access transistor with an equivalent ideal memristor. As mentioned earlier, the conversion is performed to allow W^r to be computed using Eq (1). The details are provided in Section V-A. The second step is to determine the scaling factor α that minimizes the total errors $||W - W^r||^2$, which is explained in Section V-B. The third step is to specify the conductance values g that minimize the total errors while guaranteeing that, $||(W - W^r) \cdot v_{cal}||$, is close to zero, which is outlined in Section V-C. The motivation is to leverage the known properties of the input vector space. Fourth, the state variables s of the non-ideal memristors are determined by solving a system of non-linear equations using Newton's method, as explained in Section V-D. Lastly, closed loop programming is applied to program the memristors on-chip to the desired states s using the techniques in [1], [4].
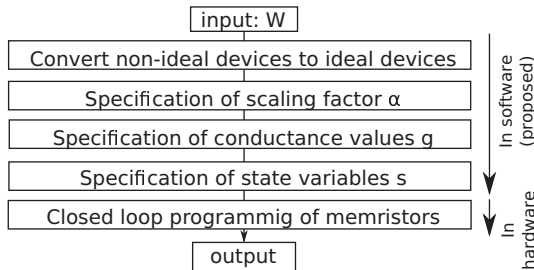

Fig. 5. Proposed flow for mapping W to an MCA.

A. Convert non-ideal devices to ideal devices

The first step is to convert the non-ideal memristors and access transistor to ideal memristor with a conductance of g, i.e., converting the circuit in Figure 2(a) to the circuit in Figure 2(b). g is bounded within $[g_{min}, g_{max}]$, where g_{min} and g_{max} are respectively the conservatively estimated minimum and maximum conductance of the series connection of each memristor and access transistor. These conductance values are different from the programmable memristor conductance range because the estimated conductance of the access transistor is included.

B. Specification of scaling factor α

In this section, a technique of specifying the scaling factor α_{opt} that minimizes the total errors $||(W - W^r)||^2$ is proposed.

The method used in this paper is based on first guessing an scaling factor α_0. Given α_0, the technique in [7] is utilized to specify the memristor conductance values g by minimizing $||W - W^r||$. Next, α_k is updated to α_{k+1} based on the relation between the value range errors and the conversion errors to minimize the total errors. If the value range errors are larger than the precision errors, α_k is updated to $\alpha_{k+1} = \alpha_k \cdot (1 - \beta)$. If the precision errors are larger than the value range errors, $\alpha_{k+1} = \alpha_k \cdot (1 + \beta)$. (Experimentally we have observed that the total errors are close to the minimum when the value range errors are equal to the precision errors.) The process is repeated until no further improvements in terms of total errors are achieved over t iterations. The parameters β and t are set to balance a trade-off between errors and run-time.

Specifically, the total errors are obtained by first quantizing g (based on the bit-accuracy of the closed loop programming). Next, W^r is obtained using Eq (1) and the total errors are computed as $||W - W^r||^2$. The value range errors are obtained as $||W - W^r||^2$ without first quantizing the conductance values g. The precision errors are set to the difference between the total errors and the value range errors.

C. Specification of conductance values g

In this section, the conductance values g are specified given W, α_{opt}, and a calibration vector v_{cal} from the input vector space. The objective is to minimize $||W - W^r||^2$ while guaranteeing that $||(W - W^r) \cdot v_{cal}|| = 0$. The motivation for minimizing $||(W - W^r) \cdot v_{cal}||$ is to exploit that the outputs from neurons in DNNs are non-negative due to the activation functions.

This step is performed by updating W to a new target matrix W^t. Next, given W^t and α_{opt} the conductance values g are specified using the approach based on minimizing $||W^t - W^r||^2$ using the method in [7]. Unfortunately, it is impossible eliminate the errors by updating elements in W that are realized too small (or too large) because the corresponding memristors are already tuned to the lower (or upper) bound of the programmable memristor conductance range. Hence, the errors are distributed to the matrix elements in the same row to ensure that $||(W - W^r) \cdot v_{cal}|| = 0$.

Let $R = (W^r - W)$ be the difference between the realized matrix W^r and the matrix W. Next, let r be a vector containing the sum of the elements in each row of R and let c be a vector containing the number of memristors with a

conductance not equal to g_{min} in each row of the MCA. Next, let u be equal to r element-wise divided by c. Subsequently, W is updated to W^t by adding $u(i)$ to each element in row i where the conductance of the corresponding memristor is not equal to g_{min}. Next, W^t is mapped into conductance values using the technique in [7].

D. Specification of state variables s

In this section, the state variables of the memristors are determined from the ideal conductance values g and a calibration signal v_{cal}, which is illustrated in Figure 6. First, the node voltages v_c and v_r and the currents through the conductors i_g are computed using g and v_{cal}, which is shown in Figure 6(a). Next, the state variables s are computed using v_c, v_r, and i_g, which is illustrated in Figure 6(b).

Computation of v_c, v_r, i_g: First, the node voltages v_c and v_r in the MCA are computed with respect to a calibration signal v_{cal} by solving Eq (1) with $v_{in}=v_{cal}$. We use $v_{cal}=v_{max}/2$ in our implementation (v_{cal} can also be set based on prior knowledge of the input vectors of a specific application). Next, the currents though each conductor g is obtained using $i_g = g \cdot (v_c - v_r)$.
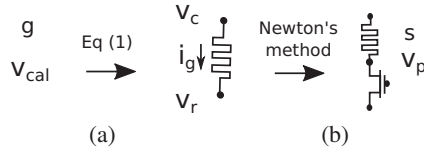

Fig. 6. Specification of s from g and v_{cal}.

Computation of state variables s: The state variables s are found by solving a non-linear system of two equations, as follows:

$$X = \begin{bmatrix} s \\ v_p \end{bmatrix}, \quad F(X) = i_g - \begin{bmatrix} i_m(s, v_c\text{-}v_p) \\ i_t(v_p, v_r, v_g) \end{bmatrix}, \quad (6)$$

where v_p is the node voltages between the memristors and the access transistors. Next, Newton's method is used to solve for $F(X) = 0$, as follows:

$$X_{k+1} = X_k - J^{-1}F(X_k), \quad (7)$$

where J^{-1} is the inverse of the Jacobian of F. Note that Newtons method can be applied independently for each memristor and access transistor pair.

VI. EXPERIMENTAL EVALUATION

The experimental results are obtained using a quad core 3.4 GHz Linux machine with 32GB of memory. The proposed techniques are implemented in MATLAB. The default MCA in the evaluation has dimensions 128x128, a wire resistance $r_w=1\Omega$, and both the input and output resistance are 100Ω. The programmable memristor conductance range is $2k\Omega$ to $3M\Omega$ [5]. We use the same non-ideal device models for the memristors and access transistors as in [5], which is available in [11]. The bit-accuracy for the memristors is set to 8 bits. The programming errors are modeled using quantization, where it is assumed that the distinguishable states in memristor conductance range (or state space) are equidistant. The maximum input voltage is set to $0.2V$. The determined state variables s and scaling factors α are evaluated using circuit simulation

with SPICE accuracy using the circuit model in Figure 2(a). In Section VI-A, the proposed mapping is evaluated in terms of matrix-vector multiplication. In Section VI-B, the proposed mapping technique is evaluated in an DNN application. We compare our results with the technique proposed in [5]. No direct comparison is provided with in [7], since that work considered a subproblem of our problem formulations.

A. Evaluation of matrix-vector multiplication

In this section, we compare the proposed mapping technique with the state-of-the-art mapping technique in [5] using full analog simulation using state variables s. The evaluation is performed with respect to the maximum output error for various input vectors and weight matrices. In Figure 7, it is demonstrated that the proposed mapping results in $4X$ to $9X$ smaller errors based on the wire resistance, crossbar size, number of memristors used per matrix element/weight, and memristor device model. It is not surprising that significant smaller maximum output errors are obtained because [5] is a heuristic and the proposed mapping technique specifies both g and α by leveraging the insights provided by the space of representable matrices. Since the benefits are obtained using only parameter optimization, the power and area is expected to be extremely similar to in [5], [4], i.e., the benefits are obtained with no overhead. In general, we find that when square MCAs of size 32/64/128 are used, $64/60/48\%$ of the programmable memristor conductance range is utilized. The average run-time is $0.5/1.5/5$ min per matrix with a dimension of 32/64/128, respectively. Note that for MCAs of dimension 128x128, only a few α values were evaluated in order to control the run-time.
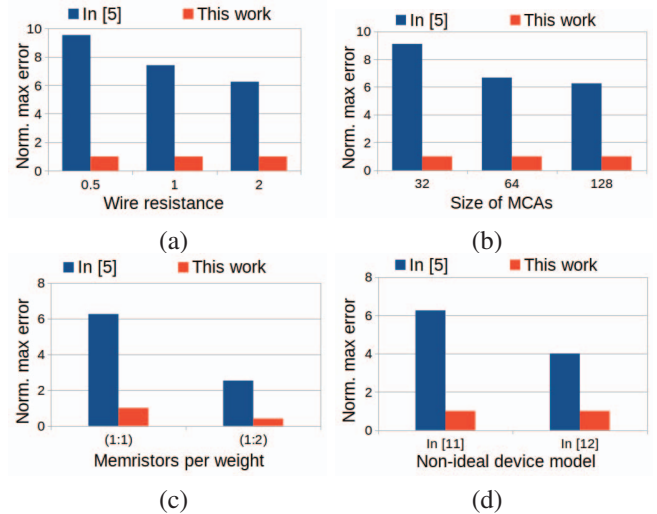

Fig. 7. Comparison with [5] using different (a) wire resistance, (b) crossbar sizes, (c) number of memristors per weight, and (d) non-ideal device models.

B. Evaluation of DNN applications

In this section, the proposed mapping and the method in [5] are used to map DNNs trained in software using GPUs to MCA based platforms for inference. The networks are trained using Keras combined with TensorFlow. In particular, we evaluate a four-layer feed-forward network trained on the MNIST dataset and a seven-layer CNN trained on the CIFAR-10 dataset. The feed-forward network has dimensions

784x500x300x10 and the properties of the CNN is shown in Figure 8.

(a)

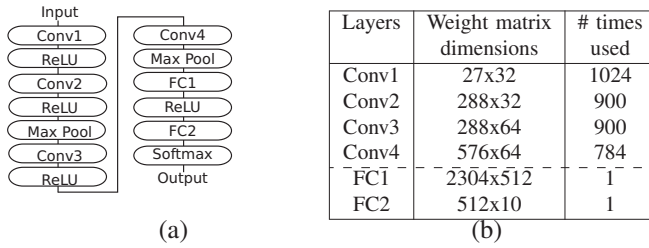| Layers	Weight matrix dimensions	# times used
Conv1	27x32	1024
Conv2	288x32	900
Conv3	288x64	900
Conv4	576x64	784
FC1	2304x512	1
FC2	512x10	1

(b)

Fig. 8. (a) Layers of CNN. (b) Weight matrices in convolutional (Conv) and fully-connected layers (FC).

The feed-forward network is mapped to an MCA based platform by partitioning each weight matrix onto a grid of 128x128 MCAs. The CNN is mapped to an MCA based platform using the kernel mapping in [9], where each convolutional layer and fully-connected layer is partitioned onto a grid of 128x128 MCAs. The default settings for the MCA are used with one memristor per weight. The classification accuracy is computed using one thousand randomly selected input images and SPICE level circuit simulation.
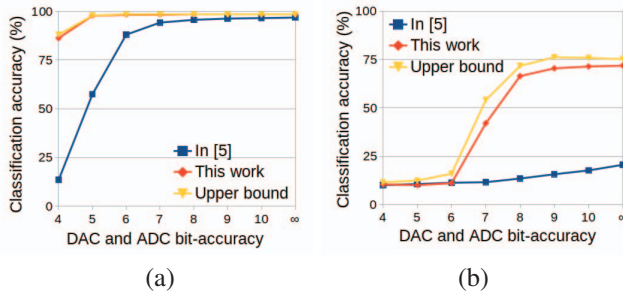

(a)   (b)

Fig. 9. Classification accuracy for different DAC/ADC bit-accuracies (a) MNIST (b) CIFAR-10.

In Figure 9, the classification accuracy achieved in MCA based hardware is shown for DACs and ADCs with various bit accuracies on the MNIST and CIFAR-10 datasets. We also plot the upper bound on the classification accuracy that can be achieved using DACs and ADCs, i.e., errors are only introduced by the DACs and ADCs. The DACs and ADCs use a fixed and dynamic reference voltage, respectively. In Figure 9(a), it can be observed that when no DAC/ADC quantization is performed (indicated with ∞) the classification accuracy on the MNIST dataset using the mapping this work and in [5] is 98.3% and 96.3%, respectively. The upper bound or software accuracy is 98.4%. Moreover, the proposed mapping technique follows the upper bound closely and outperforms the mapping in [5] when DACs and ADCs with smaller bit-accuracies are used. In Figure 9(b), it can be observed that the proposed mapping achieves a classification accuracy of 71.8%, whereas the mapping in [5] results in an accuracy of 20.5% on CIFAR-10, which is close to the software classification accuracy of 75.2%. Moreover, the proposed technique follows the upper bound on CIFAR-10 closely for DACs and ADCs with different bit-accuracies. It is easy to understand that the proposed mapping technique outperforms the method in [4] because each matrix-vector multiplication is performed with $6X$ smaller errors.
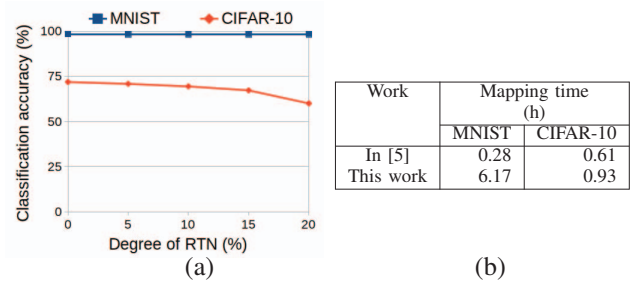

(a)   (b)

Fig. 10. (a) Classification accuracy with RTN. (b) Run-time of mapping DNNs to MCAs.

When moderate to severe (up to 20%) random telegraph noise is included [10], the classification accuracy is gracefully reduced (not impacted) on CIFAR-10 (MNIST), which is shown in Figure 10(a). In Figure 10(b), we show the run-time of our framework and the techniques in [5]. To limit the run-time of mapping the weight matrices in the CNN to state variables, we used the technique in [5] to map FC1, which is not sensitive to errors. The run-time reported in the table is based on our implementation of [5], where performance was prioritized over run-time in the implementation.

VII. SUMMARY AND FUTURE WORK

In this paper, a technique for mapping arbitrary weight matrices to MCAs is proposed. The technique improves the computational accuracy of the state-of-the-art with $4X$ to $9X$ and achieves close to software level accuracy on CIFAR-10 dataset when a CNN trained in software is mapped to an MCA based platform for inference. We plan to reduce the run-time of the algorithm in the future.

REFERENCES

[1] F. Alibart, L. Gao, B. D. Hoskins, and D. B. Strukov. High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm. *Nanotechnology*, 23(7):075201, 2012.

[2] P. Chi et al. PRIME: a novel processing-in-memory architecture for neural network computation in reram-based main memory. ISCA'16, pages 27–39, 2016.

[3] M. Hu et al. Memristor crossbar-based neuromorphic computing system: A case study. *IEEE Transactions on Neural Networks and Learning Systems*, 25:1864–1878, 2014.

[4] M. Hu et al. Memristor-based analog computation and neural network classification with a DPE. *Adv. Materials*, 30, 2018.

[5] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams. Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication. DAC'16, pages 1–6, 2016.

[6] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. In *Nature*, pages 436–444, 2015.

[7] B. Liu et al. Reduction and IR-drop compensations techniques for reliable neuromorphic computing systems. ICCAD'2014, pages 63–70, 2014.

[8] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar. ISAAC: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars. ISCA'16, pages 14–26, 2016.

[9] L. Song, X. Qian, H. Li, and Y. Chen. Pipelayer: A pipelined reram-based accelerator for deep learning. HPCA'17, pages 541–552, 2017.

[10] J. P. Strachan. DPE: Exploring high efficiency analog multiplication with memristor arrays. In *Int. Conf.on Rebooting Computing*, 2015.

[11] J. P. Strachan, A. C. Torrezan, F. Miao, M. D. Pickett, J. J. Yang, W. Yi, G. Medeiros-Ribeiro, and R. S. Williams. State dynamics and modeling of tantalum oxide memristors. *IEEE Transactions on Electron Devices*, 60(7):2194–2202, 2013.

[12] L. Xia et al. Technological exploration of RRAM crossbar array for matrix-vector multiplication. *Journal of Computer Science and Technology*, 31(1):3–19, 2016.