Dot-Product Engine for Neuromorphic Computing: Programming 1T1M Crossbar to Accelerate Matrix-Vector Multiplication

Miao Hu[*1], John Paul Strachan[1], Zhiyong Li[1], Emmanuelle M. Grafals[1], Noraica Davila[1], Catherine Graves[1], Sity Lam[1], Ning Ge[2],

Jianhua (Joshua) Yang[3], and R. Stanley Williams [1]

[1]Hewlett Packard Laboratories, Palo Alto, CA 94304, USA. *Email: miao.hu@hpe.com
[2]HP Inc, Palo Alto, CA 94304, USA.
[3]Department of Electrical and Computer Enginerring, University of Massachusetts, Amherst, MA 01003, USA.

ABSTRACT

Vector-matrix multiplication dominates the computation time and energy for many workloads, particularly neural network algorithms and linear transforms (e.g, the Discrete Fourier Transform). Utilizing the natural current accumulation feature of memristor crossbar, we developed the Dot-Product Engine (DPE) as a high density, high power efficiency accelerator for approximate matrix-vector multiplication. We firstly invented a conversion algorithm to map arbitrary matrix values appropriately to memristor conductances in a realistic crossbar array, accounting for device physics and circuit issues to reduce computational errors. The accurate device resistance programming in large arrays is enabled by close-loop pulse tuning and access transistors. To validate our approach, we simulated and benchmarked one of the state-of-the-art neural networks for pattern recognition on the DPEs. The result shows no accuracy degradation compared to software approach (99 % pattern recognition accuracy for MNIST data set) with only 4 Bit DAC/ADC requirement, while the DPE can achieve a speed-efficiency product of $1,000\times$ to $10,000\times$ compared to a custom digital ASIC.

1. INTRODUCTION

An ideal nanoscale memristor crossbar array can naturally carry out vector-matrix multiplication–a computationally expensive task for many important applications–in a single constant time step [2]. By applying a vector of voltage signals to the rows of a memristor crossbar, multiplication by each memristor element's conductance is carried out by the KCL rule and the current is summed across each column [3]. This "analog" method of vector-matrix multiplication can be orders of magnitude more efficient than any digital ASIC [4], particularly as the crossbar array size is highly scalable.

However, many circuit issues in analog domain are not trivial, as one needs to account for finite wire resistance, input/output stage resistance, memristor device nonlinearity in current-voltage relationship, and all sources of noise. A naïve linear mapping from matrix values to crossbar conductance and input/output values to voltage signals leads rapidly to low accuracy as array size grows[5]. Some researchers use hardware training schemes to minimize errors [6, 7, 8, 9]. However, so far hardware training approaches are
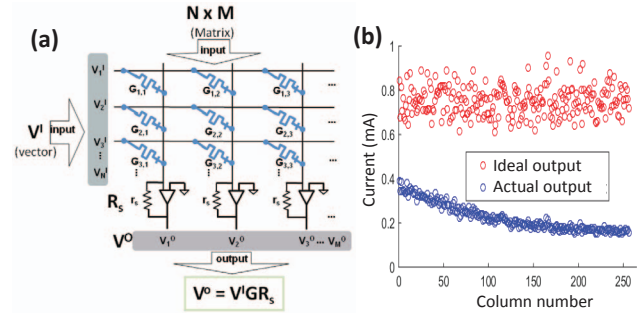
Figure 1: Memristor crossbar for matrix multiplication. (a) Basic concept; (b) Difference between ideal and actual output.

slow, iterative processes with extensive reading and writing of all devices, with limited performance/energy efficiency comparing to software [10] and potential device wear out [11].

To overcome these issues, we developed the Dot-Product Engine (DPE), together with a fast conversion algorithm, as a realistic solution to accelerate matrix-vector multiplication in robust applications which can tolerate lower computing accuracy such as neural network algorithms. Our contributions are summarized as follows:

- A general conversion algorithm accounting for device and circuit issues has been developed to map arbitrary matrix values to memristor conductances. This conversion algorithm can be extended to any other crossbar structures or cross-point devices by just replacing circuit or device models.

- The simulation of DPE is based extensively on circuit and device models calibrated from real fabricated devices. The accuracy, speed and energy efficiency of DPE is quantitatively analyzed compared to a custom digital ASIC [1]. With conservative assumptions of DPE parameters, a 1,000 to 10,000 times better speed-efficiency product can be achieved.

- To evaluate the DPE in neural network applications, we implement one of the state-of-the-art neural networks on DPEs [10]. Simulation result shows that with 4 bits accuracy DAC/ADC interfaces, the hardware can achieve the same recognition accuracy(99%) as the software approach, but with much better speed and energy efficiency.

2. PRELIMINARY
2.1 Memristor crossbar array

Memristor crossbar arrays show great application potential for next generation non-volatile memories. For memory applications, a large on/off ratio is desired, along with fast switching, high endurance, and especially high nonlinearity to suppress the leakage
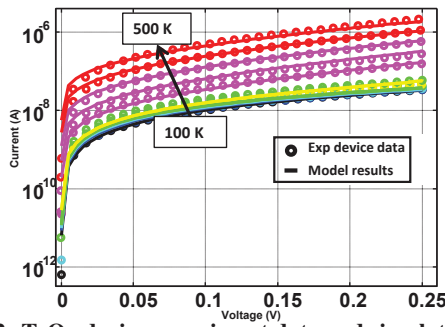
Figure 2: TaOx device experiment data and simulation result

Figure 3: (a) Actual wafer image; (b) 4×4 **crossbar array; (c), Circuit diagram and noise sources for simulation.**

current when read as well as the sneak current during write operations in larger crossbar arrays. [12].

Memristor crossbar arrays also raise great interest for computing applications because they can naturally carry out array-size vector matrix multiplication in one constant time step. As in Fig.1 (a), in an ideal crossbar where memristors are linear and all circuit pararistics are ignored, applying an input vector voltage $\mathbf{V_{in}}$ to rows of the crossbar and sensing the output voltage $\mathbf{V_{out}}$ with Trans-Impedance Amplifiers (TIA) at columns, we get $\mathbf{V_{out}} = \mathbf{V_{in}GR_S}$, where $\mathbf{R_S}$ is the feedback resistance, and $\mathbf{G}$ is the conductance matrix of each cross-point device. For analog computing applications, memristors are desired to have stable, continuous linear conductance states for representing matrix values [13]. The high on/off ratio, high endurance and switching speed are still desired but not as strictly required as for memories[2].

However, naïve mapping leads to poor computing accuracy in real crossbars. Fig.1 (b) shows an example of all voltages across devices in a 256×256 crossbar array, positive matrix values are linearly mapped to memristor conductance, but the actual output values are significantly different from the ideal output values due to non-ideal devices and circuit issues.

2.2 Mapping algorithms and hardware training schemes

There are many research work on using memristor crossbar for effective computations, like matrix multiplications or synaptic operation in a neural network. Generally, they can be categorized into two approaches and still at infant stages. One approach is to find mapping schemes and crossbar designs to tolerate device and circuit imperfections for accurate vector-matrix multiplication, current work either have high restriction on the matrix to map [3, 14] and/or high requirement on the crossbar parameters [5]. A general solution to map arbitrary matrices onto realistic crossbars is still missing, and none of work use basic matrix-multiplication accuracy to evaluate their performance or compare to digital ASICs.

Another approach is to embed learning algorithms on peripheral circuits to automatically tune memristor conductance as synapse weight to achieve logic operations [12] or pattern recognition/ classification functions [6, 7, 8, 9, 13]. Instead of pursuing computing accuracy of memristor crossbar, they try to improve pattern recognition/ classification accuracy of crossbar-based neural networks. Some impressive work have been done to realize basic supervised [8, 6] or unsupervised learning schemes [15] on crossbar to handle small scale pattern recognition/classification tasks. However, these hardware learning chips are still far behind existing software algorithms on accuracy, speed as well as power efficiency due to high cost and uncertainty on memristor tuning operations [6].

3. METHODOLOGY

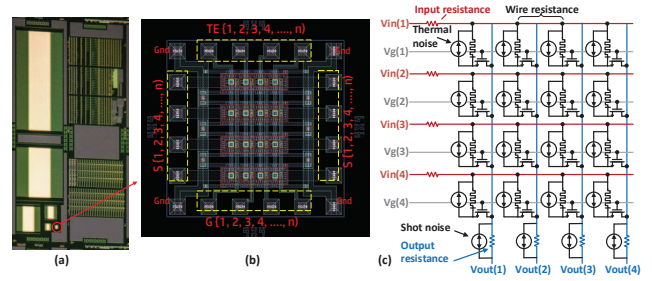The Dot product Engine (DPE) is developed to execute vector

matrix multiplication on memristor crossbars. It includes memristor device and crossbar designs and, most importantly, a conversion algorithm to overcome non-idealities and ensure optimized computing accuracy and robustness.

3.1 Device and circuit design

Fig.2 (a) shows data from a fabricated TaOx memristor device. These devices can be repeatedly programmed to different target resistance states from 2k to 3M Ω and shows the needed linearity at sufficiently low voltages <=0.3 V. Thus, it is a good candidate for implementing multiplication operations. We built a compact memristor model to capture the measured electron transport for these devices. The model matches well the memristor's current-voltage (I-V) behavior with varying ambient temperatures.

We also fabricated a wafer with different sizes of 1T1M crossbar arrays for testing, as shown in Fig. 3 (a) and (b). Controlling electronics for DPE operations are implemented using peripheral circuits built on separate printed circuit boards. The crossbar arrays are accessed through high bandwidth multi-pin connectors to provide sequential device programing and parallel reading for computing. Fig. 3 (c) shows the crossbar simulation model including all of the circuit parasitics and noise referenced above. The model also includes a temperature dependence which is important for most memristor devices. The current driving capability and current sensing sensitivity of the peripheral circuits are also included. Additionally, we model the random telegraph noise (RTN), or two-level fluctuations , frequently present in memristor devices, which are treated as binary noise [17]. The RC delay of the crossbar is not considered here since, for the geometries considered here even in large crossbar arrays, is expected to be sub-ns [5] and can be ignored for DPE operations running here at 10 MHz. Retention issues are also not expected to be of concern for these devices and timescales allowing for many operations.

3.2 Conversion algorithm

We utilized our knowledge of the device and circuit physics to develop a conversion algorithm that transforms matrix values into memristor conductances that minimize inaccuracies in matrix multiplication due to the known circuit issues.

To be of practical value, our conversion algorithm must also be efficiently computed. We developed a solver in MATLAB that includes a crossbar simulation down to the device level based on experimental data. Unlike SPICE which uses approximations for general purpose circuit simulation, our crossbar simulator solves analytically and is 2~3 orders of magnitude faster than SPICE simulators on crossbar array simulation, with no accuracy loss. Fig.4(a) shows the overall sequence of the DPE, including the basic flow for the conversion algorithm. A matrix is first linearly mapped to an ideal memristor crossbar to get the ideal crossbar behavior, it assumes the ideal crossbar has zero wire resistance, zero input/output
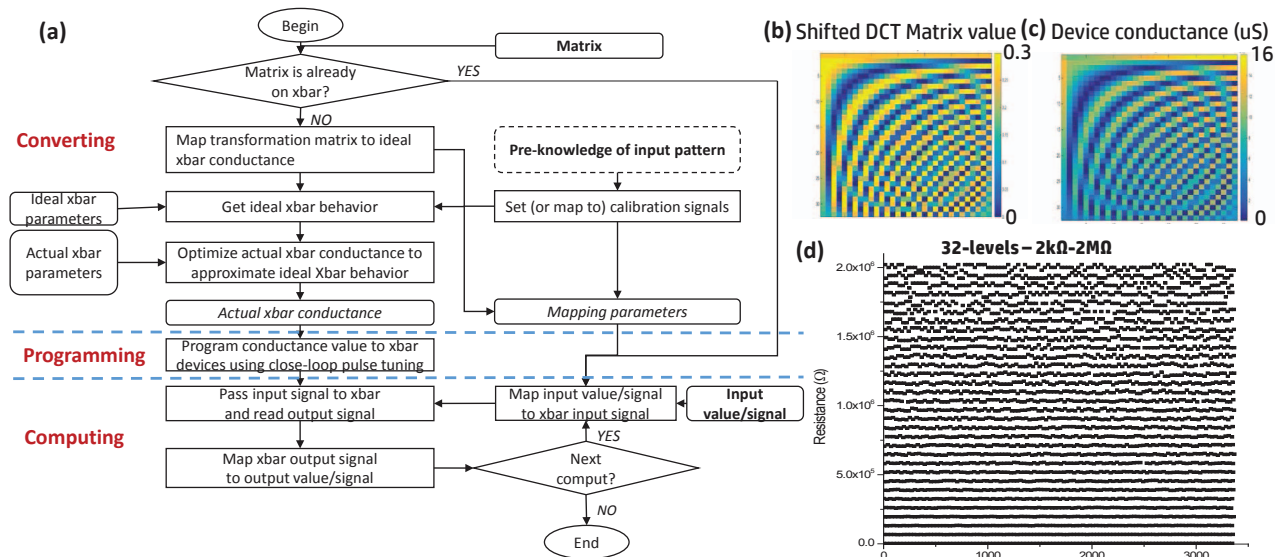
Figure 4: The DPE work flow and result example. (a) DPE work flow; (b) shifted DCT matrix value; (c) converted memristor conductance value; (d)Tune memristor to consistently change among 32 arbitrary pre-set levels with 1% error tolerance;

stage resistances, perfectly linear I-V relationship in the cross-point devices, zero noise, etc. Our conversion algorithm then simulates the actual (non-ideal) current and voltages across the realistic crossbar array, and tunes the device conductances to match the current that should pass through each cross-point device in an ideal crossbar. Computation of the device physics is sped-up by using the pre-calculated Jacobian matrix. The resulting process is efficient and converting arbitrary matrices onto a 128×128 1T1M crossbar takes less than 3 seconds on a normal desktop workstation.

After the conversion, we use close-loop tuning scheme [16]to program memristors to the desired conductance values. With the help of access transistors, close-loop tuning scheme can be crossbar-compatible. Fig.4(b) and (c) illustrate converting a 32×32 Discrete Cosine Transformation (DCT) matrix onto a 32×32 crossbar. Fig.4(d) shows that our device can be consistently tuned from one resistance state to any other states, and it demonstrates a resolution of up to 32 levels with 1 % error tolerance.

4. OPTIMIZATION

Devices/crossbars can be optimized to pursue high linearity/low interconnect resistance, but it is important to optimize the array mapping itself to improve the performance. In this section we go through the optimization process of the conversion algorithm and show that accurate computing can be achieved on 1T1M crossbars with current material systems instead of requiring further improvements at the device/circuit level. For this work, crossbar settings are fixed to 10Ω wire resistance, and $100 \ \Omega$ input/output resistance.

With fixed crossbar parameters, the DPE conversion algorithm function have three input variable set: the initial conductance of crossbar array, the calibration signal and the temperature. These variables should be optimized to achieve the best computing accuracy, as well as keeping input/output signal amplitudes within the peripheral circuit capability.

4.1 Optimize initial conductance

The conductance $\mathbf{G_{IDEAL}}$ is the matrix of cross-point device conductances for an ideal crossbar array. Mapping an arbitrary matrix $\mathbf{A}$ to $\mathbf{G_{IDEAL}}$ involves first performing a uniform shifting to handle any negative values, since negative conductances are not possible. If $\mathbf{A}$ contains negative values, it is shifted to positive by the constant A_{SHIFT} added to all entries, with this contribution to

the vector-matrix multiplication removed at the final step by subtracting $A_{SHIFT} \cdot sum(\mathbf{X})$, where $sum(\mathbf{X}$ is the summation of the input vector. Two linear mapping coefficients are generated:

$$a = (G_{on} - G_{off})/(A_{max} - A_{min})); b = G_{on} - a \cdot (A_{max})). \tag{1}$$

G_{on}/G_{off} are maximum/minimum memristor conductance values and A_{max}/A_{min} are the maximum/minimum matrix values. Finally, $\mathbf{G_{IDEAL}}$ can be calculated by:

$$\mathbf{G_{IDEAL}} = a \cdot \mathbf{A} + b \tag{2}$$

To optimize initial conductance, there is a general trend that as the crossbar size increases, it is necessary to map the matrix to a higher resistance range. This is due to larger crossbar arrays having longer wire paths and causing more signal degradation. Devices, particularly those on the far side of the crossbar, need to be tuned to lower resistance to compensate. Increasing initial resistance range can lower the expectation of ideal signal at each cross-point and makes conversion process easier. Moreover, if the initial resistive range is too low, some devices may not be able to be tuned more conductive to compensate signal degradation and conversion process halts. Despite such tuning required by our mapping algorithm, we show that high accuracy for the final computation is achieved with a calibration input signal. Additionally, it is necessary that the device can accommodate the required lower resistances for large array sizes for convergence of the algorithm.

4.2 Optimize calibration signal

Besides the initial conductance, a calibration input signal is used to produce the ideal crossbar behavior. For neuromorphic and data analysis where the input data has patterns, a calibration signal can be chosen from the input data pattern to minimize the computing error. For general purpose vector-matrix multiplications, we use a uniform calibration signal vector. For simplicity, the input vector is normalized to the range [0,1]. Converting the input vector $\mathbf{X}$ to an input voltage signal $\mathbf{V_{in}}$ follows:

$$\mathbf{V_{in}} = \mathbf{X} \cdot V_{max} \tag{3}$$

V_{max} is the maximum voltage of the DAC. For the current DPE design, the voltage inputs need to be positive to keep the transistor working in the positive linear region.
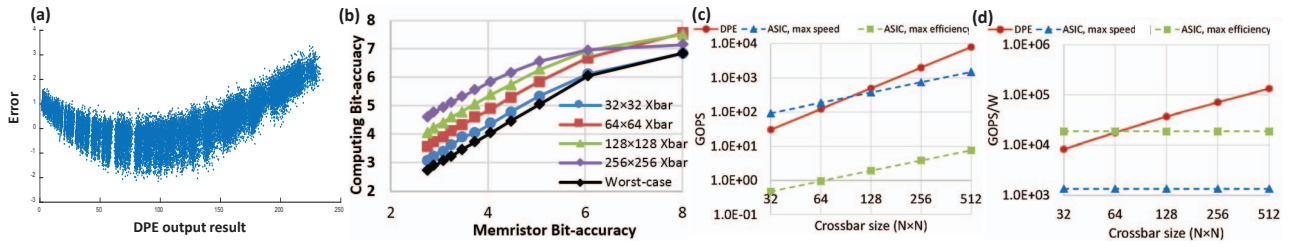
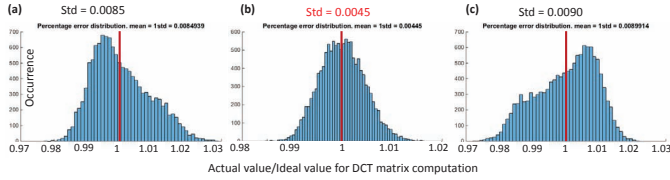**Figure 5: DPE results. (a) error vs. output result; (b) DPE accuracy; (c) DPE speed; (d) DPE power efficiency**
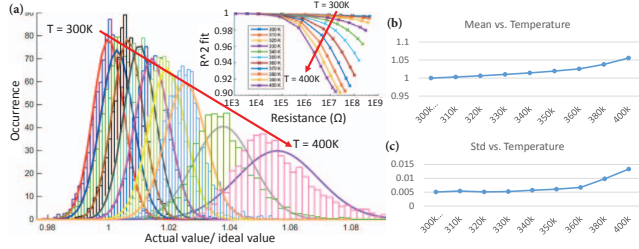

Figure 6: Impact of calibration signal.

Figure 7: Impact of temperature. (a) error distribution vs. working temperature; (b) mean of the error distribution; (c) std. of the error distribution.

Fig.6 shows the impact of calibration signal in a 128×128 crossbar array. The calibration signal was set to 0.25V and the DPE accuracy tested with random inputs ranging from either 0∼0.5V, 0∼0.25V, or 0∼0.125V. For 0∼0.5V inputs, the results show less compensation is provided by the algorithm. This is because TaOx memristor devices, become more conductive and nonlinear under high voltage excitations and a low calibration signal is not enough to compensate for errors in large inputs. On the opposite side, for 0∼0.125V inputs, the results show over-compensation because the conversion algorithm over-estimates the device nonlinearity, which leads to more error for small inputs. It appears that for our device, it is best to choose the maximum inputs as the calibration signal. In this way, the device nonlinearity is correctly compensated by the conversion algorithm, and a symmetric Gaussian-like distribution can be observed with minimal standard deviation.

4.3 Optimize calibration temperature

Our memristor device model was developed including the impact of a varying ambient temperature, and we are the first to analyze the impact of temperature on memristor crossbar for vector-matrix multiplication. It is clearly the best to use the working temperature for calibration, but it is still important to analyze the robustness of the DPE when operating temperatures exceed calibration.

Fig.7 shows the impact of temperature. The calibration temperature is fixed at 300K and the DPE operation is simulated for temperatures of 300K to 400K. The device nonlinearity as a function of temperature is shown in the inset of Fig.7. The result shows the DPE has high resilience to changing environmental temperature. The Gaussian-like distribution is always preserved, and the DPE computing result shows only linear shifting of the mean with no obvious increase in the standard deviation until 360K. This shift can be restored in the post-processing of the ADC sensing data.
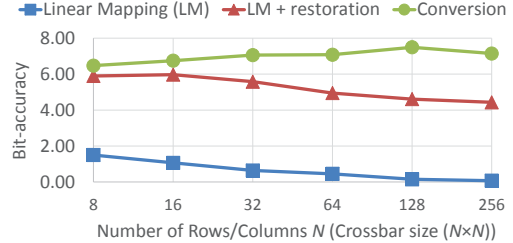

Figure 8: Comparing different mapping schemes

5. RESULTS

Using the conversion algorithm, Fig.5 shows the error pattern, accuracy, speed and power efficiency of the DPE. The error pattern (panel (a)) is measured by gradually increasing input vector amplitude from 0 (minimum value) to 1 (maximum value) for a 128×128 crossbar. Here, we did not include memristor resistance drift or fluctuations, while thermal and shot noises are included in the simulation. The error pattern shows a belly shape since the conversion algorithm is optimized to reach ∼0 average error.

The memristor accuracy (number of repeatable and precise resistance levels) is a larger concern compared to thermal or shot noise in the system. An error here can come from both inaccurate device programming or the resistance changing after programming (due to noise, random telegraph noise or drift). We added different levels of RTN in the memristor conductance from 0% to 30% [17]. Accuracy (panel (b)) is measured in "bits", or log[number of distinguishable levels]. As expected, the final computation accuracy, in bits, will depend on the number of bits of accuracy present in the memristor programming (called "Memristor bit-accuracy"). The worst-case scenario refers to a sparse matrix with only one value per column. In such a case, the overall DPE computation accuracy equals the memristor accuracy; in contrast, for dense matrices, over 7-bit computing accuracy can be achieved even with memristors of only 6-bit accuracy. This is because the memristor noise (limiting memristor bit accuracy) is assumed to be un-correlated and has a reduced effect for large array sizes. Moreover, it is also observed that the computing accuracy begins to saturate near 8-bit even with increased device accuracy. It is because the intrinsic current-voltage nonlinearity of devices becomes the dominating factor. Excluding the impact of RTN, our result shows device nonlinearity contributes ∼80% of the total noise, whereas shot noise contributes ∼15% and thermal noise contributes the remaining ∼5%.

Fig.8 compares conversion algorithm to other mapping schemes. "LM + restoration" improves LM method by assuming memristors' resistance in crossbar follows Gaussian distribution, and use specific filters to restore the signal. However, the matrix it can apply is very limited and restoration also cause large overhead. All in all, our conversion algorithm provides the best accuracy as well as high robustness, and it has near zero overhead.

For the speed and power efficiency (panels (c) and (d)), we carefully compared DPE with a state-of-the-art ASIC [1]. Included in our simulation for the DPE is the peripheral circuitry, including
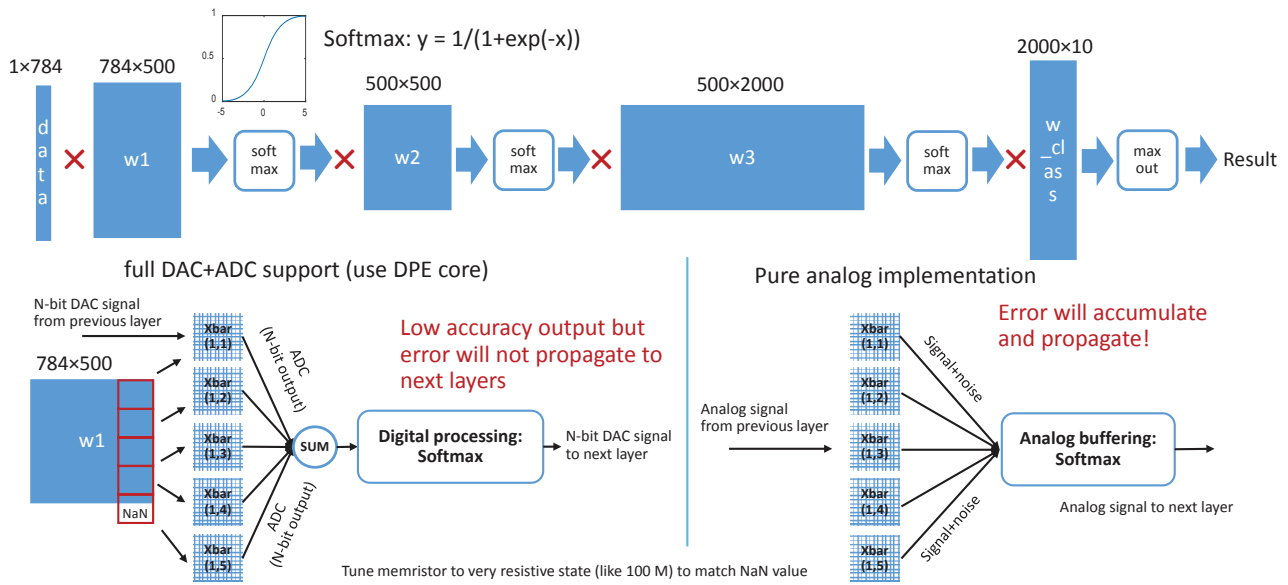
Figure 9: (a) Algorithm flow of Auto-encoder classifier for pattern recognition on MNIST data set. (b) Implementation with DPEs of digital interfaces such as DAC and DAC; (c) Implementation with DPEs of analog interfaces such as analog buffer and amplifier.

the DACs, TIAs, and ADCs. Each channel is assumed to be 8-bit, operate at 10MHz and consume $100\mu W$ [1]. The peripheral circuit consumes >90% of the total power and >95% of the total chip area. The major speed bottleneck also comes from the DAC and ADC.This result calls for more efficient and compact DAC and ADC design for low speed parallel signal processing. However, the DPE can still achieve the same speed-efficiency product at a crossbar size of 32×32, and becomes more than $1,000\times$ higher (when ASIC is in "maximum speed mode"), or more than $10,000\times$ (ASIC in "maximum energy efficiency mode"), when crossbar size is scaled up to 512×512. Such crossbar sizes are conservative estimations of what can be fabricated [18, 19].

6. NEURAL NETWORK IMPLEMENTATION
6.1 Auto-encoder classifier on DPEs

To demonstrate the capability of the DPE as a key accelerator for real-world workloads, we implemented and simulated one of the state-of-the-art neural networks, the auto-encoder classifier on pattern recognition for MNIST data sets [10].

Fig. 9 (a) shows the full neural network implemented in the algorithm, which contains 4 layers and large scale matrices (e.g., 784×500). Fig.9 (b) and (c) shows two different ways to implement the DPE for each layer of the neural network, using either digital (DAC + ADC) or analog interfaces (analog buffer + amplifiers). To enable large scale neural networks, we partitioned the large matrices into 125×125 sub-matrices so they can be converted onto 128×128 crossbar arrays, then the partitioned outputs are collected to construct the final output signal to the next layer. The reason behind using larger crossbar for matrix mapping is first, to fight the yield problem, and second, to evenly distribute the weight of biasing signal into multiple arrays. Since the weight of the biasing signal is usually much larger than the weight matrix values, directly putting them in a single crossbar causes that row of memristors to be significantly more conductive than all others, resulting in degradation of the biasing accuracy. By evenly distributing the biasing weight across multiple crossbars, each part of the biasing weight will be comparable to other weight values. Comparing the two designs, digital interfaces consume more area and power than analog interfaces and the signals are quantized resulting in accu-

racy loss. But signal errors do not accumulate from same-layer crossbars or propagate to the next layers of the neural network.

6.2 Performance

Fig.10 shows the simulation result with the pattern recognition accuracy (out of 10,000 test patterns) using the DPE-based neural network. For the software approach in Fig.10(a), binary random noise is added to all weight matrices and the auto-encoder classifier neural network exhibits a high tolerance to this random noise. Even with 30% noise, the error rate only increased from 1.18% to 1.54%. An additional noise source, random telegraph noise (RTN) in which the device conductance discretely fluctuates between values, has been experimentally observed in memristors with varying amplitudes, has also been simulated here. A notable result for the DPE with a digital interface (Fig.10 (b)), is that only a 4-Bit ADC/DAC is needed and results in no performance degradation. Note that a software approach requires at least 8-10 bits of storage accuracy to ensure a similar computing accuracy. This difference is based on DPE's unique advantage that it effectively measures the most significant bits (MSB) of a result while software needs enough least significant bits (LSB) to compute the correct MSBs.

For the analog approach in Fig.10 (c), since error accumulates and propagates, the accuracy is reduced by 1.5% even with no RTN. The error at each layer of the neural network is observed in Fig.11 with 100 different testing patterns. In the ideal case, the error should be independent to input patterns and appears as white noise. However, it is observed that this dependency does exist in the analog approach, due mainly to device nonlinearity, and the error accumulates across the entire neural network. This phenomenon is clearly observed in the final stage of the neural network where 2000 inputs collapse to only 10 outputs.

As a significant improvement to the analog approach, we implemented an additional mean removal function in the system, which is simply a mapping between the known mean error and the actual output. This modification dramatically reduced the impact of error accumulation and propagation from a 1.5% to only a 0.2% recognition error increment, as shown in Fig.10 3(d). This indicates the great potential of using DPE to accurately and efficiently accelerate pattern recognition and other machine learning workloads.
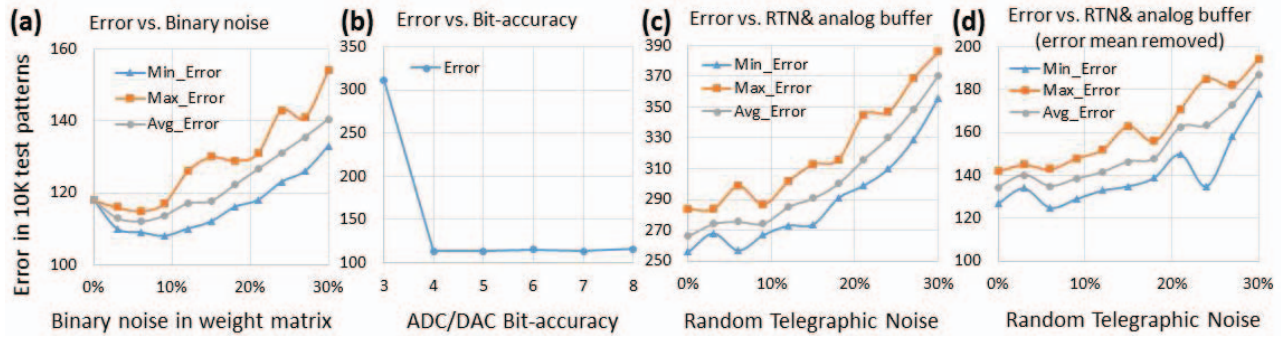
Figure 10: Recognition accuracy on 10k testing samples (a) Original software algorithm; (b) Neural network on DPEs of digital interfaces; (c) Neural network on DPEs of analog interfaces; (d) Add mean removal function to (c).
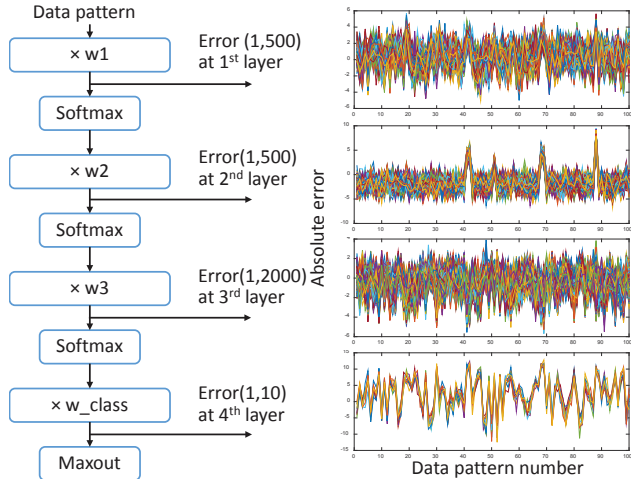

Figure 11: Error pattern accumulated from multiple crossbar arrays at each layer of the neural network.

7. CONCLUSION

In this work, we developed the Dot-product Engine (DPE) as an accelerator for approximated vector-matrix multiplications. One key difference of DPE from previous crossbar-based computing engines is our conversion algorithm that finds the correct mapping between mathematical calculations and circuit operations to overcome the known circuit issues and other limitations to maximize computing accuracy. The mapping is currently run on an external computer but can be embedded on-chip for higher efficiency. With conservative assumptions, we built a 1T1M crossbar simulation platform calibrated to experimental data and quantitatively analyzed the performance and efficiency of the DPE compared to a state-of-the-art ASIC. Our result shows the DPE can have 1,000 to 10,000 better speed-energy efficiency product than the ASIC using 512×512 crossbars. We demonstrated the DPE's application in neuromorphic computing by implementing the auto-encoder classifier neural network. Our results show 99% recognition accuracy, which yields no degradation compared to a software approach, and using only 4-bit accuracy DAC/ADCs. The DPE is the first complete crossbar-based computing engine design that leverages and integrates existing technologies to pursue a near-term accelerator product. We show the potential of the DPE for accelerating many important applications, including machine learning, low power signal processing for Internet of Things (IoT), and linear transformations such as the Fourier transform.

8. ACKNOWLEDGMENTS

This research is based upon work supported by the Office of the

9. REFERENCES

[1] S. K. Hsu *et al.*, "A 280 mv-to-1.1 v 256b reconfigurable simd vector permutation engine with 2-dimensional shuffle in 22 nm tri-gate cmos," *IEEE JSSC*, vol. 48, no. 1, pp. 118–127, 2013.

[2] J. J. Yang *et al.*, "Memristive devices for computing," *Nature nanotechnology*, vol. 8, no. 1, pp. 13–24, 2013.

[3] M. Hu *et al.*, "Hardware realization of bsb recall function using memristor crossbar arrays," in *DAC*. ACM, 2012, pp. 498–503.

[4] K. Fatahalian *et al.*, "Understanding the efficiency of gpu algorithms for matrix-matrix multiplication," in *ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*. ACM, 2004, pp. 133–137.

[5] P. Gu *et al.*, "Technological exploration of rram crossbar array for matrix-vector multiplication," in *ASP-DAC*. IEEE, 2015, pp. 106–111.

[6] G. Burr *et al.*, "Experimental demonstration and tolerancing of a large-scale neural network (165,000 synapses), using phase-change memory as the synaptic weight element," in *IEEE IEDM*. IEEE, 2014, pp. 29–5.

[7] B. Liu *et al.*, "Vortex: variation-aware training for memristor x-bar," in *DAC*. ACM, 2015, p. 15.

[8] M. Prezioso *et al.*, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, pp. 61–64, 2015.

[9] M. Hu *et al.*, "Memristor crossbar-based neuromorphic computing system: A case study," *IEEE TNNLS*, vol. 25, no. 10, pp. 1864–1878, 2014.

[10] R. Salakhutdinov and G. E. Hinton, "Learning a nonlinear embedding by preserving class neighbourhood structure," in *ICAIS*, 2007, pp. 412–419.

[11] Y. Y. Chen *et al.*, "Endurance/retention trade-off on cap 1t1r bipolar rram," *TED*, vol. 60, no. 3, pp. 1114–1121, 2013.

[12] H.-S. P. Wong *et al.*, "Metal–oxide rram," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, 2012.

[13] S. Jo *et al.*, "Nanoscale Memristor Device as Synapse in Neuromorphic Systems," *Nano Letter*, vol. 10, no. 4, pp. 1297–1301, 2010.

[14] M. Tarkov, "Mapping weight matrix of a neural network?s layer onto memristor crossbar," *Optical Memory and Neural Networks*, vol. 24, no. 2, pp. 109–115, 2015.

[15] S. Choi *et al.*, "Data clustering using memristor networks," *Scientific Reports*, vol. 5, 2015.

[16] F. Alibart *et al.*, "High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm," *Nanotechnology*, vol. 23, no. 7, p. 075201, 2012.

[17] S. Choi *et al.*, "Random telegraph noise and resistance switching analysis of oxide based resistive memory," *Nanoscale*, vol. 6, no. 1, pp. 400–404, 2014.

[18] X. Dong *et al.*, "Pcramsim: System-level performance, energy, and area modeling for phase-change ram," in *ICCAD*. ACM, 2009, pp. 269–275.

[19] S.-S. Sheu *et al.*, "A 4mb embedded slc resistive-ram macro with 7.2 ns read-write random-access time and 160ns mlc-access capability," in *IEEE ISSCC*, 2011, pp. 200–202.