Unsupervised Digit Recognition Using Cosine Similarity In A Neuromemristive Competitive Learning System

BON WOONG KU, Synopsys, Inc.
CATHERINE D. SCHUMAN, Oak Ridge National Laboratory
MD MUSABBIR ADNAN, The University of Tennessee, Knoxville
TIFFANY M. MINTZ, RAPHAEL POOSER, and KATHLEEN E. HAMILTON,
Oak Ridge National Laboratory
GARRETT S. ROSE, The University of Tennessee, Knoxville
SUNG KYU LIM, Georgia Institute of Technology

This work addresses how to naturally adopt the l^2-norm cosine similarity in the neuromemristive system and studies the unsupervised learning performance on handwritten digit image recognition. Proposed architecture is a two-layer fully connected neural network with a hard winner-take-all (WTA) learning module. For input layer, we propose single-spike temporal code that transforms input stimuli into the set of single spikes with different latencies and voltage levels. For a synapse model, we employ a compound memristor where stochastically switching binary-state memristors connected in parallel, which offers a reliable and scalable multi-state solution for synaptic weight storage. Hardware-friendly synaptic adaptation mechanism is proposed to realize spike-timing-dependent plasticity learning. Input spikes are sent out through those memristive synapses to each and every integrate-and-fire neuron in the fully connected output layer, where the hard WTA network motif introduces the competition based on cosine similarity for the given input stimuli. Finally, we present 92.64% accuracy performance on unsupervised digit recognition with only single-epoch MNIST dataset training via high-level simulations, including extensive analysis on the impact of system parameters.

CCS Concepts: • **Computing methodologies → Unsupervised learning**; **Bio-inspired approaches;**

Additional Key Words and Phrases: Unsupervised learning, MNIST, digit recognition, single-spike temporal code, spiking neural network, compound memristor, winner-take-all, neuromorphic computing

ACM Reference format:
Bon Woong Ku, Catherine D. Schuman, Md Musabbir Adnan, Tiffany M. Mintz, Raphael Pooser, Kathleen E. Hamilton, Garrett S. Rose, and Sung Kyu Lim. 2022. Unsupervised Digit Recognition Using Cosine Similarity In A Neuromemristive Competitive Learning System. *J. Emerg. Technol. Comput. Syst.* 18, 2, Article 38 (March 2022), 20 pages.
https://doi.org/10.1145/3473036

Authors' addresses: B. W. Ku, Synopsys, Inc., 800 N Mary Ave, Sunnyvale, CA, 94085; email: bon.ku@synopsys.com; C. D. Schuman, T. M. Mintz, R. Pooser, and K. E. Hamilton, Oak Ridge National Laboratory, 1 Bethel Valley Rd, Oak Ridge, TN, 37830; emails: schumancd@ornl.gov, mintztm@ornl.gov, pooserrc@ornl.gov, hamiltonke@ornl.gov; Md M. Adnan and G. S. Rose, The University of Tennessee, Knoxville, 1520 Middle Dr, Knoxville, TN, 37996; emails: madnan@vols.utk.edu, garose@utk.edu; S. K. Lim, Georgia Institute of Technology, 266 Ferst Dr, Atlanta, GA, 30332; email: limsk@ece.gatech.edu.

1550-4832/2022/03-ART38 $15.00
https://doi.org/10.1145/3473036

1 INTRODUCTION

In a neuromorphic system, synaptic weights are updated by **spike-timing-dependent plasticity (STDP)**, a biological learning mechanism directly correlated to the brain-oriented functions [10, 18, 48]. Over the past few years, memristor-based emerging **non-volatile memory (NVM)** technologies [8, 9, 60, 66] have drawn huge attention of the field of neuromorphic computing to the fact that NVMs naturally realize STDP thanks to their efficient in-memory processing capability [19, 21, 26, 33]. The superior fabrication density of NVMs over the traditional memory technologies allows us to expect a large-capacity, compact-size, low-power, and hopefully affordable synaptic weight storage [1, 5, 31, 37, 53, 55]. Furthermore, engineering breakthroughs in NVMs enable us to explore a large-scale memristive neuromorphic (neuromemristive) system architecture that targets brain-level parallelism [28, 38].

Most studies on neuromemristive system architecture [3, 39, 50, 52, 59, 62, 64] have used the dot-product as a similarity metric between an input feature vector and a synaptic weight vector during training. Although a memristor crossbar realizes the dot-product operation very efficiently, it is unbounded unless the normalization followed, which tends to degrade the learning performance [17, 34, 41, 46, 58]. Recently, Reference [20] experimentally showed that subsequent forward and backward processing of the memristor crossbar provides the l^2-norm value of the weight vector very efficiently. This motivates us to demonstrate system-level unsupervised learning performance while actively employing cosine similarity (dot-product between l^2-normalized input feature and synaptic weight vectors) as the similarity metric.

In this work, we propose a neuromemristive competitive learning system architecture that finds the best cosine similarity in the hard **winner-take-all (WTA)** [24] network motif. Shown in Figure 1, our neuromemristive system is a fully connected two-layer spiking neural network with a competitive learning module. Each layer has been optimized to manipulate cosine similarity. Pre-synaptic input encoding neurons in the input layer perform single-spike temporal coding. This encoding scheme allows not only efficient dot-product along the memristor crossbar but also hardware-friendly STDP. Input spikes are sent out to the receptive field of fully connected post-synaptic output integrate-and-fire neurons. The WTA logic introduces the competition among output neurons based on cosine similarity between input and weight vectors. The sole winner either triggers the STDP learning during training or chooses the final label for inference. Via abstract high-level software simulations, we analyze the impact of system parameters extensively and finally present 92.64% accuracy performance on unsupervised digit recognition [30] with only single-epoch training.

2 INPUT SPIKE ENCODING

The learning efficiency of a neuromemristive system is directly affected by a spike encoding scheme [7, 42]. The encoding scheme decides the amount of feature information delivered to the system. Also it has a significant impact on the synaptic and neuronal signal processing complexity to deal with the incoming spike sequence. In biological nervous systems, various neuronal spike patterns have been observed in response to different types of external sensory stimuli [2, 14]. Simply put, when higher input intensity causes more frequent spikes, it is called rate code. However, when the temporal structure of a spike sequence encodes the sensory input features, we call it temporal code.

Rate code has been a traditional input encoding scheme in neuromorphic studies [11, 16, 33, 47, 49, 51], since neurons are known to be so sensitive to the internal fluctuations that noisy responses deform the temporal structure of a spike sequence. However, rate code results in long network simulation runtime, since a neuron should collect enough number of spikes to capture

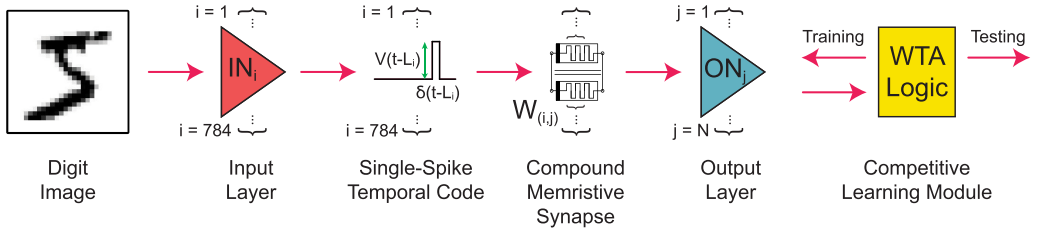

Fig. 1. Our neuromemristive competitive learning system. There exists 784 (28×28) pre-synaptic input encoding neurons (IN_i) that transform an input image into 784 single-spike signals. These spikes are sent out through compound memristive synapses to each and every post-synaptic output integrate-and-fire neuron (ON_j) in the fully connected output layer, where the hard WTA network motif introduces the competition among them.

the precise firing rate. Also, **long-term depression (LTD)** might occur right after **long-term potentiation (LTP)**, since a high-intensity stimulus causes pre-synaptic neurons to fire right after a post-synaptic neuron fires, which complicates an STDP implementation. To resolve this issue, some adopt neuronal refractory period to ignore the incoming spikes after post-synaptic firing, others do not implement LTD but decrease the synaptic weights exponentially unless pre-synaptic firing happens, and yet others utilize more parameters to tune the impact of LTD and LTP differently. These all enlarge the parameter space to optimize, leading to difficulty in achieving near-ideal performance and fast runtime. Moreover, a spike generation circuit for rate code commonly utilizes a random number generator (Poisson-distributed in general), which turns out to be a huge overhead for the hardware implementation [12, 54].

However, temporal code has been known to offer a fast and efficient spike encoding solution for the given visual stimuli [45, 56]. This is especially true when the temporal pattern of a input spike sequence are so distinctive that post-synaptic neurons do not need to collect the whole spike train redundantly. Previously, References [25, 35] utilized a Gabor filter to preprocess an image and turned the resulting edge information of an image into spikes using temporal code. However, the latency is reversely proportional to the input intensity, which leads to the distortion of input features around high intensity pixels.

In this work, we propose rapid and effective spike encoding scheme termed **single-spike temporal code (STC)**. STC transforms input stimuli into the set of single spikes with different latencies and voltage levels; higher pixel intensity leads to earlier spike firing and larger spike voltage. The sparsity derived from using single spikes allows cheap neuronal input spike processing. The latency is calculated by the additive inverse of the pixel value with respect to the maximum pixel intensity, so that the temporal structure of input spikes does not have any distortion. This deterministic delay calculation removes the need for a random generator making the architecture hardware friendly. Also the temporal feature makes easy to implement efficient STDP scheme for memristive synapses that will be discussed in the next section. Note that an input stimulus turns into voltage and temporal feature vectors. What makes STC distinctive from existing works [15, 32, 63] on temporal coding is that the voltage feature in STC naturally allows us to utilize the fast and energy-efficient dot-product operation through a memristor crossbar array, since the amplitude of spike voltage delivers the pixel value information directly.

Figure 2 depicts how STC transforms a handwritten digit image into spikes. There are 784 (28×28) pixels for an image in the MNIST dataset [30], and single-channel pixel value ranges from 0 to 255. For ease of explanation, the encoding timesteps are assumed to be ranged from 0 to 255, and the voltage level ranges from V_{min} to V_{max}. Now, if the pixel value is at its maximum (255), then

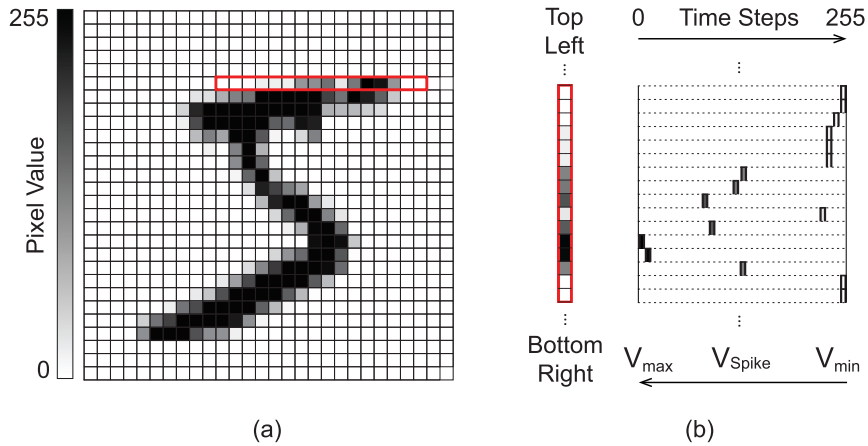

Fig. 2. (a) In the MNIST dataset [30], there are 784 (28×28) pixels for a handwritten digit image, whose single-channel intensity values range from 0 to 255. (b) Image is flattened and STC rapidly encodes the image into 784 single spikes with different latencies and voltage levels. Depending on the pixel intensity, the firing time and the voltage level of a spike are decided; higher pixel intensity leads to earlier spike firing and larger spike voltage.

the single spike fires at 0 timestep with maximum voltage level (V_{max}), reaching the post-synaptic neurons immediately and strongly. If the pixel value is at its minimum (0), then the single spike fires at 255 timestep with the minimum voltage level (V_{min}). Note that spikes from higher intensity pixels contribute to the post-synaptic neuron firing earlier and more.

Moreover, STC offers a flexible solution to boost the simulation runtime by reducing the number of encoding timesteps at the expense of the temporal and voltage precision of input spikes. Figure 3 shows the spike encoding result based on STC with only 4 timesteps. The single spikes placed at their respective pixel locations are colored in black at the timestep when they are fired. Spike voltage level for each timestep is presented at the bottom. Due to the reduced number of encoding timesteps, a certain range of pixel intensity shares the same firing time and spike voltage level, which are calculated by the following equations:

$$T_i = (N_{step} - 1) - \lfloor N_{step} \times (P_i/(P_{max} + 1)) \rfloor, \tag{1}$$

$$V_t = (V_{max} - V_{min}) \times \frac{(N_{step} - 1 - t)}{(N_{step} - 1)} + V_{min}, \tag{2}$$

where T_i is the spike timing for pixel i, V_t denotes the spike voltage level for timestep t, and V_{max} and V_{min} denotes the maximum and minimum voltage level, respectively. N_{step} denotes the number of encoding timesteps, P_i the intensity value of pixel i, and P_{max} the maximum pixel intensity (255 in this work).

Note that the reduced number of encoding timesteps results in the altered temporal structure and voltage level of spikes. Therefore, the runtime savings makes a tradeoff with reduction in the input feature precision.

3 SYNAPSE MODEL AND SPIKE-TIMING-DEPENDENT PLASTICITY

We employ a compound memristor model as a synaptic device in our neuromemristive system [6, 44]. The compound memristive synapse takes advantage of the stochastic resistive switching of a memristor [40, 61]; when multiple binary-state memristors are connected in parallel to

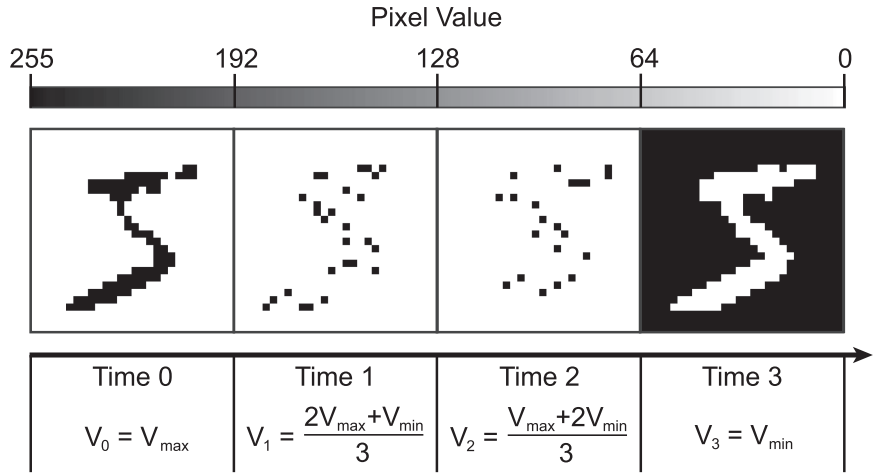

Fig. 3. The spike encoding result based on STC with four timesteps. The spike timing (Time t) and voltage level (V_t) are determined by Equations (1) and (2), respectively. Spikes placed at their respective pixel locations are colored in black when they are fired.

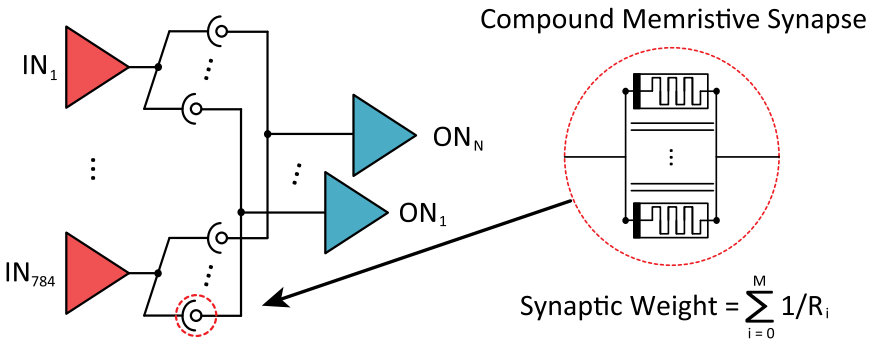

Fig. 4. Compound memristive synapses that connect pre-synaptic input encoding neurons (IN) and post-synaptic integrate-and-fire output neurons (ON). R_i denotes either R_{off} (high-resistance state, HRS) or R_{on} (low-resistance state, LRS) of the binary-state memristor i. The synaptic weight is represented by the effective memductance of parallelly connected M binary-state memristor constituents.

implement a synapse, the synaptic weight state is determined by the binary resistance states of individual memristors, which is stochastically updated during synaptic adaptation. Note that the compound memristor offers fully scalable $(M + 1)$ different evenly distributed synaptic weight states, and the synaptic weight is controlled by the population of resistance states. As shown in Figure 4, consider that R_{off} and R_{on} is the resistance for **high-resistance state (HRS)** and the **-low-resistance state (LRS)** of a binary-state memristor, respectively. Given M binary-state memristor constituents connected in parallel, assume that $x \in [0, M]$ memristors are in LRS. The weight of the compound memristive synapse (W) is calculated by

$$W = x/R_{on} + (M - x)/R_{off}. \tag{3}$$

STDP adjusts synaptic weights based on the correlation of a pre- and a post-synaptic neuron spike firing events. If the pre-synaptic neuron fires before the post-synaptic neuron fires, then

STDP increases the synaptic weights (LTP). When the pre-synaptic neuron fires after the post-synaptic neuron fires, STDP weakens the synaptic connections (LTD). Our STDP implementation with compound memristive synapses controls the number of STDP events (N_{stdp}) based on the time difference between a pre-synaptic neuron spike and a post-synaptic neuron spike; the shorter the time difference is, the more STDP events occur. The state transition of individual memristors with resistive switching probability (P) is performed by exposing them to the weak programming condition [40, 61]. When LTP occurs, memristors in HRS stochastically switch their resistance states to LRS. For LTD, the transition goes from LRS to HRS. Then the expected change of x within a single LTP / LTD event is

$$< \Delta x >_{LTP} = (M - x)P, \tag{4}$$

$$< \Delta x >_{LTD} = -xP. \tag{5}$$

Solving Equations (3), (4), and (5) together, Figure 5 shows the change of synaptic weight per LTP and LTD events, respectively. The amount of expected synaptic weight change in a single STDP event depends on the current synaptic weight, and it exponentially decreases when STDP events consecutively happen.

Figure 6 depicts that STC makes STDP implementation straightforward. The time window for the STDP rule fully covers the entire encoding timesteps (N_{step}) for both LTP and LTD cases symmetrically, so that every single input spike contributes to the synaptic adaptation. The maximum number of STDP events is constrained to be the same as N_{step} in this work. When a post-synaptic neuron fires a spike, we calculate the respective spike timestep difference from each of the pre-synaptic neuron spikes that contribute to the post-synaptic neuron firing ($\Delta T = T_{pre} - T_{post} < 0$ where T_{pre} and T_{post} denote the timestep when the pre-/post-synaptic neuron fires the spike). Then ΔT decides the number of LTP events based on the linear STDP rule ($N_{STDP} = N_{step} - |\Delta T|$), and individual compound memristive synapses are repetitively exposed to the weak programming condition up to the decided number of LTP events. Note that LTP happens right after the post-synaptic neuron fires. Also, synapses connected to pre-synaptic neurons that encode the highest-intensity pixels do not take the maximum number of LTP events, but the ones who contribute to the post-synaptic neuron firing at the closest timing take the full synaptic weight updates.

Synapses connected to pre-synaptic neurons that fire after the post-synaptic neuron firing take LTD. Again, the spike timestep difference ($\Delta T = T_{pre} - T_{post} > 0$) produces the number of LTD events based on the linear STDP rule, and weak programming is followed. LTD happens whenever the pre-synaptic neuron fires after the post-synaptic neuron firing. It is noteworthy that LTD is as important as LTP, since it captures the background contrast in the image effectively.

4 NEURON MODEL AND NETWORK ARCHITECTURE

Under the competitive learning principle, experts compete against each other with their own neuronal expertise. If a neuron is more specialized to process a specific stimulus than others by increasing its own expertise, then it becomes a winner and either takes a learning chance or chooses the final label. In this work, the neuronal expertise is defined as the cosine similarity between an input feature vector and a synaptic weight vector. The larger cosine similarity, the better neuronal expertise matching with the given input. To calculate the cosine similarity, recall that the receptive field of a post-synaptic neuron constructs a synaptic weight vector ($\vec{W}$). Also, note that STC turns an input image into spikes with different voltage levels, which construct an input feature vector ($\vec{I}$). During encoding timesteps, input spikes are applied to the memristor crossbar, while post-synaptic neurons hold virtual ground at the other end. Then, the total current sum flowing into a post-synaptic neuron is equal to the dot-product between the synaptic weight vector and

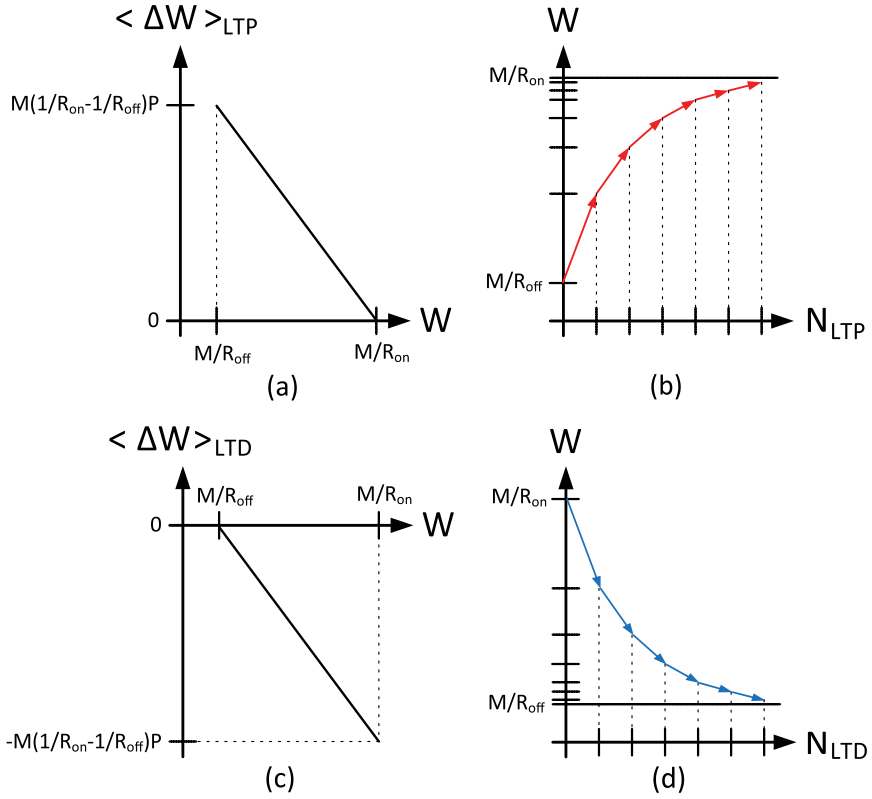

Fig. 5. The change of synaptic weight in a compound memristive synapse. R_{off} and R_{on} are HRS and LRS, respectively, and M denotes the number of binary-state memristor constituents in a compound memristive synapse. (a) The expected synaptic weight change ($< \Delta W >_{LTP}$) vs. the current synaptic weight (W) for a single LTP event. (b) The impact of the number of LTP events (N_{LTP}) on the synaptic weight. (c) The expected synaptic weight change ($< \Delta W >_{LTD}$) vs. the current synaptic weight (W) for a single LTD event. (d) The impact of the number of LTD events (N_{LTD}) on the synaptic weight.

the input feature vector ($\langle \vec{I}, \vec{W} \rangle$) by Ohm's law and Kirchhoff's current law [19, 65]. An analog-to-digital converter turns the current value into the digital membrane potential for the post-synaptic neuron [4, 27]. Given that the forward and backward processing of a memristor crossbar gives us the sum of squared $\vec{W}$ values ($\langle \vec{W}, \vec{W} \rangle$) [20], we can calculate the l^2-norm of $\vec{W}$ ($\|\vec{W}\|$) easily for each neuron. Also note that the l^2-norm of $\vec{I}$ ($\|\vec{I}\|$) is an external constant for each input image. Dividing membrane potential by $\|\vec{W}\|$ and $\|\vec{I}\|$ naturally gives us the cosine similarity, which is a measure of similarity between $\vec{W}$ and $\vec{I}$ in the dot-product space.

It is important to note that the spike firing event of the winner neuron out of competition is essential for STDP learning mechanism. Therefore, the integrate-and-fire operation of a neuron should be well defined for the best learning performance. For training, each input image has to be presented to the system twice, and the winner neuron from the first round triggers the STDP learning in the second round. To be more specific, at the end of first round, we pick the largest cosine similarity from the entire output neurons. Then we decide the amplifying factor by the ceiling of the reciprocal of the largest cosine similarity. The amplifying factor scales up the increase of membrane potential ($\langle \vec{I}, \vec{W} \rangle$) during the second round of input image presentation. When the

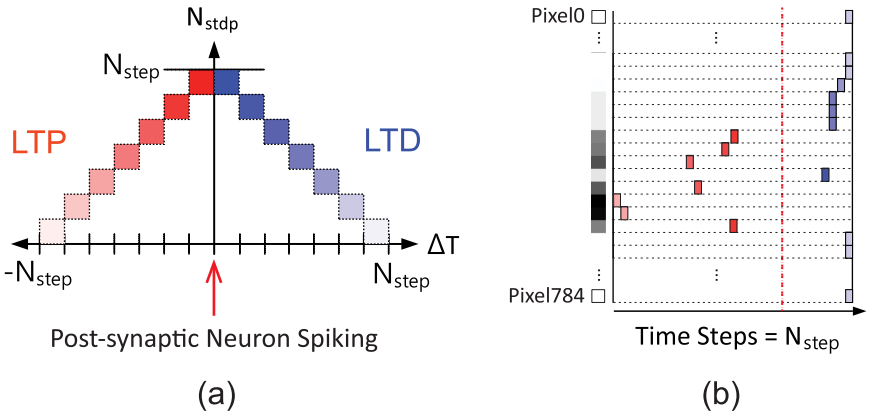

Fig. 6. (a) Our symmetric STDP rule to determine the number of STDP events (N_{stdp}). The time window for STDP rule covers the entire spike encoding timesteps (N_{step}) for both LTP and LTD cases, so that every single input spike contributes to synaptic adaptation. The maximum number of STDP events is assumed to be (N_{step}) in this work. The timestep difference between a pre-synaptic neuron spike and a post-synaptic neuron spike ($\Delta T = T_{pre} - T_{post}$) decides N_{stdp}. (b) STDP implementation with single-spike temporal code. The red dotted line is when post-synaptic neuron spike fires. Synapses connected to pre-synaptic neurons who fired before the post-synaptic neuron fires take LTP; otherwise they take LTD updates. For synaptic weight update, the compound memristive synapse is repetitively exposed to the weak programming condition up to the decided N_{stdp}.

membrane potential is over the threshold ($\|\vec{W}\| \times \|\vec{I}\|$), a neuron fires a spike. Note that the threshold of an output neuron is not affected by the amplifying factor.

Recall that STC encodes the input spike feature in the priority order. Spikes from higher intensity pixels come earlier to output neurons and add larger values to the membrane potential. However, the largest STDP learning chance is given to the synapses that deliver the pre-synaptic spikes fired most recently. By adopting the amplifying factor in the integrate-and-fire operation, we controls the firing timing of an output neuron, tuning the impact of STDP learning. The ceiling of the reciprocal of the largest cosine similarity gives the smallest integer amplifying factor so that it minimizes the number of firing output neurons while guaranteeing at least one of the most relevant output neurons firing in the second round of image presentation. The ceiling function allows us to offer a learning chance to other immature experts, which naturally realizes balanced training.

4.1 Training

The purpose of training is to increase the neuronal expertise, which is cosine similarity ($\langle \vec{I}, \vec{W} \rangle / \|\vec{W}\| \times \|\vec{I}\|$) between an input feature vector $\vec{I}$ and a synaptic weight vector $\vec{W}$ in this work. At the beginning of training, we initialize the synaptic weights by reseting all memristors into HRS, which implies that no expertise is given for every output neuron. For each input image, every output neuron is activated before the first encoding timestep and performs integrate-and-fire operation as encoding timesteps proceed. The simulator records the firing status of output neurons and places a neuron on the candidate list if it fires. Note that the $\vec{W}$ of output neurons who are not trained once yet has no specific orientation, since we reset all the synaptic weights before training starts. When the very first competition decides the sole winner from output neurons randomly, only the winner takes a specific orientation in its $\vec{W}$ after synaptic adaptation. Therefore, there is a high chance for the first winner to win the competition again unless we give an

advantage to another output neurons for the next competition. For the fair competition, if there exists output neurons that have never fired before, then we place them on the candidate list in the first priority even though they have not fired and choose one of them as the winner. Then we force the winner to fire at the first timestep so that it develops the seed orientation in its $\vec{W}$ and produces valid cosine similarity. Once every output neuron has fired at least once, the normal competition proceeds.

When only the single candidate exists, the rest of output neurons are deactivated, and the winner takes an STDP based on the proposed synaptic adaptation method. If multiple candidates exist, then we find the winner neuron who has the smallest ratio of membrane potential to the threshold ($V_{mem}/V_{th} > 1$) at the current firing timestep. The reason why we choose the smallest V_{mem}/V_{th} is as follows. The fact that cosine similarity is 1 at its maximum implies that without controlling the amplifying factor, an output neuron fires only if the input feature vector ($\vec{I}$) is exactly on the same direction of synaptic weight vector ($\vec{W}$). In this case, multiple output neurons can fire, but they should have exactly the same expertise, so it does not matter that WTA chooses any one of them as the winner. However, if no one fires at the first image presentation, and we adjust the amplifying factor for the next iteration, then multiple candidates can exist although they have different expertise. Recall that the amplifying factor is initially determined with the largest cosine similarity. Therefore, neuron firing guarantees that the expertise of a candidate is close enough to input feature, so choosing the smallest V_{mem}/V_{th} helps to give a training chance to immature output neurons, which partially realizes homeostasis for balanced training. When the single winner is chosen, again all output neurons except for the winner become inactive, and only the winner takes an STDP.

4.2 Labeling and Inference

Labeling proceeds in the same way as training, but synaptic weights are fixed and we do not use WTA competition. We feed visual stimuli to the network and record the firing status of output neurons for the given stimuli. Since it is not a competition, multiple neurons are allowed to fire. Each output neuron has its own scoreboard for 10 labels. For the given input image and its label, if the neuron fires, then we add 1 to the score of the given label. After a single epoch of training set, the label of each output neuron is decided by the label with the maximum score. This is the only step where labels in the training set are used. For inference, again synaptic weights are fixed, but we turn on the WTA module this time. In addition, we do not use the amplifying factor but only care about choosing the best expert for the given input stimuli based on the cosine similarity to input images in the test set, which are unseen from training. Then the single winner determines the final classification label.

5 EXPERIMENTAL RESULTS

5.1 Simulation Parameters

The STDP rule is assumed to be symmetric, and the maximum number of STDP events (N_{stdp}) is assumed to be the same as the number of spike encoding timesteps (N_{step}) in this work. Our simulation parameters are referred to the experimental data [61]. We set V_{min} and V_{max} as 0.1 V and 1 V, respectively, to be within the practical memristor readout voltage range. For memristor parameters, R_{on} is assumed to be 10 kΩ, and 1 MΩ for R_{off}. Based on those assumptions, we develop a C++ high-level software simulator and run it on Titan, the flagship Cray XK7 supercomputer system located at Oak Ridge National Laboratory. The number of binary switching memristor constituents in a compound memristive synapse (M), memristor switching probability (P),

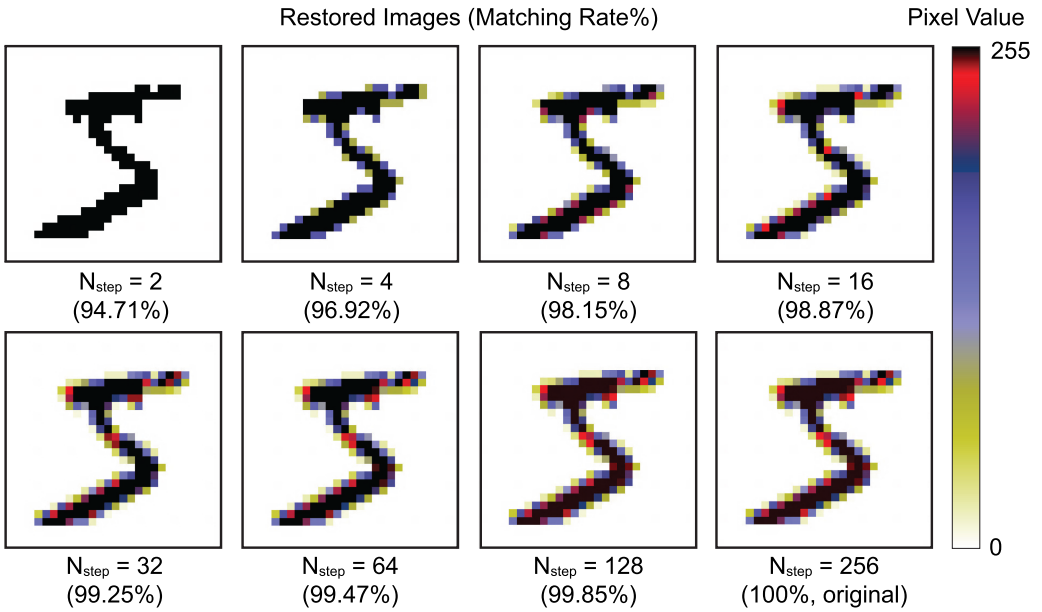

Fig. 7. Reconstructed input images based on the different number of encoding timesteps (N_{step}). The voltage level of spikes in each timestep is scaled to the range of pixel intensity (~0–255). The definition of matching rate is the projection ratio, which is the ratio of l^2-norm of the original vector to that of the restored vector after projection.

number of output neurons (N_{expert}), and number of encoding timestep (N_{step}) are parameterized to analyze the impact on the learning performance of our neuromemristive system.

5.2 Impact of the Number of Encoding Timesteps

N_{step} has a direct impact on the temporal structure and voltage level of encoded spikes. If N_{step} is equal to the maximum pixel intensity, then each pixel value is mapped to a specific timestep in one-to-one correspondence. In this case, however, spike processing at the output layer takes a huge portion of network-level simulation, resulting in an exponential increase of simulation runtime. To reduce the runtime overhead, N_{step} can be scaled down, and a range of pixel values shares the same spike timestep. However, this leads to compression loss in the feature precision as a consequence. To quantize the impact of reduction in N_{step} on the input feature, we reconstruct a digit image by transforming the voltage level of spikes into a pixel value and explore the resemblance of the reconstructed image to the original.

Figure 7 shows the reconstructed images based on the different N_{step}. The voltage level of spikes in each timestep is scaled to the range of pixel intensity (~0–255). The definition of resemblance used here is the projection ratio. We project the restored feature vector onto the original feature vector and measure the ratio of L^2-norm of the original vector to that of the projected vector. We observe that the mismatching rate increases exponentially while reducing N_{step}. When binary spike encoding ($N_{step} = 2$) is used, the mismatching rate goes up to 5.29%.

Figure 8 shows the spike processing runtime overhead for an output neuron based on different N_{step}. When $N_{step} = 256$, processing an input image takes around 10 s, which is a huge overhead for the entire network-level simulation (consider 60,000 training images for a single epoch). It is observed that this overhead is exponentially decreased with reduction in N_{step}. Therefore, we

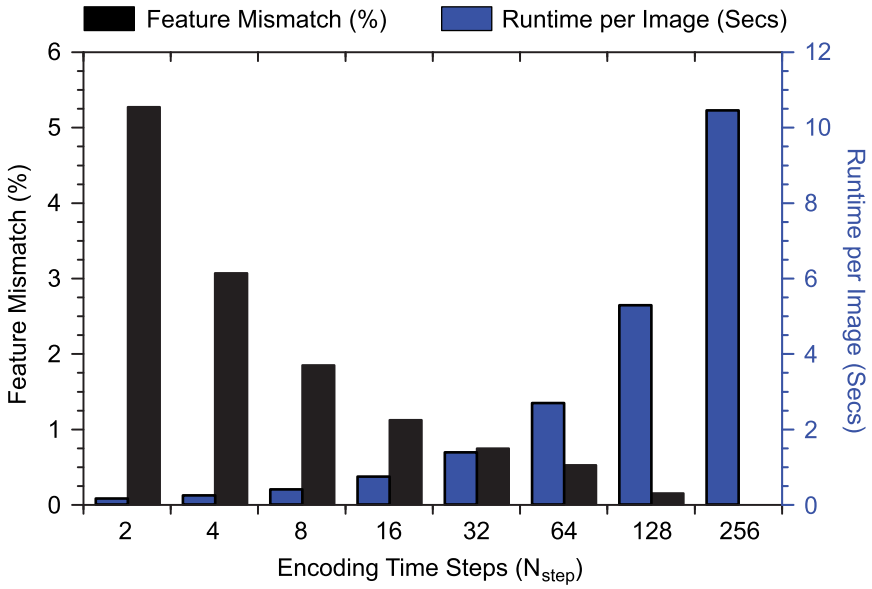

Fig. 8. A tradeoff between the feature precision and the spike processing runtime based on different N_{step}. With the multiplication of mismatching rate and simulation runtime as a cost function, we decide $N_{step} = 4$ as the sweet spot while targeting less than 2 s of spike processing runtime.

Table 1. The Impact of N_{step} on the Number of Training Repetition Required to Have at Least 1 of 784 Synapses Fully Matured, Where All Memristor Constituents Are in LRS

Encoding Time Steps	Repetition Count			Encoding Time Steps	Repetition Count		
	Min	Avg	Max		Min	Avg	Max
4	94	99.3	103	16	23	25.8	28
8	48	51.4	56	32	12	13	14

find a tradeoff between feature precision and the simulation runtime overhead. With the multiplication of mismatching rate and simulation runtime as a cost function, we decide $N_{step} = 4$ as the sweet spot while targeting less than 2 s of spike processing runtime ($N_{step} \leq 32$).

Recall that the maximum number of STDP events (N_{stdp}) is the same as N_{step}. Since N_{stdp} decides the number of weak programming events for synaptic adaptation, changing N_{step} affects the speed of synaptic weight update as well. Table 1 shows the impact of N_{step} on the number of training repetition required to have at least one of 784 synapses fully matured, where all memristor constituents are in LRS. Two hundred fifty-six memristor constituents for a synapse, and 0.01 resistive switching probability is used for this experiment. Due to the stochasticity of resistive switching, experiments are done 10 times each, and the first training image is used. We observe that increasing N_{step} inversely decreases the required repetition count for a neuron to become a matured expert on a specific input image. If $N_{step} = 32$, then only 13 repetition in average allows full synaptic adaptation, which implies that too-large N_{step} makes a neuron easily forget its previous expertise and adapt to a new input feature.

Table 2. The Impact of Synapse Parameters on the Number of Training Repetition Required to Have at Least 1 of 784 Synapses Fully Matured

Memr. Count	Swt. Prob.	Repetition Count			Memr. Count	Swt. Prob.	Repetition Count		
		Min	Avg	Max			Min	Avg	Max
4	0.001	53	79.9	130	64	0.001	572	666.9	720
4	0.01	7	10.1	14	64	0.01	54	63.9	75
4	0.1	1	1.3	2	64	0.1	6	6.9	8
16	0.001	227	337	388	256	0.001	856	984.9	1061
16	0.01	30	37	45	256	0.01	94	99.3	103
16	0.1	3	3.3	5	256	0.1	9	10	11

5.3 Impact of Synapse Parameters

Synapse parameters have a direct impact on synaptic adaptation. The switching probability (P) decides the speed of weight update, and the larger number of binary-state memristor constituents in a synapse (M) increases the synaptic precision while decreasing the speed of synaptic adaptation. Similarly to Tables 1 and 2 tabulates the impact of synapse parameters on the number of training repetition required to have at least 1 of 784 synapses fully matured; $N_{step} = 4$ is used, and experiments are done 10 times each. We observe that P affects the speed of synaptic adaptation directly, and it corresponds to the learning efficiency in the standard neural network training model. Too-small P leads to huge required repetition especially when $M = 256$. However, large M helps to increase the weight precision by offering a higher number of synaptic weight states, but a synapse needs to take more training repetition to be matured. To show the impact of synaptic weight precision more clearly, Figure 9 shows the receptive fields after full repetition. When $M = 256$, a relatively large repetition count allows us to capture the most important feature of an input image. However, when $M = 4$, small number of repetition allows fast learning, but it does not fully capture the important features, which would degrade the learning performance.

Figure 10 shows how P affects the learning speed. For this experiment, $M = 256$ and $N_{step} = 4$ is set. After the full repetition, we feed the next training image only once and compare the receptive field before and after the presentation of the new image. When $P = 0.001$, single presentation of the new image does not affect the previous expertise. However, when $P = 0.1$, the neuron starts to adapt to the new feature very quickly. Too-large P, therefore, does not help to converge the learning performance, since it makes a neuron a fast learner but a fast forgetter at the same time. Note that P can be affected by the device imperfection including manufacturing defects, aging, and thermal impacts. Given that the stochasticity of resistive switching is controllable by updating the weak programming condition, such microscopic device variations can be addressed by hardware optimization to preserve the optimal P value experimentally.

5.4 Impact of the Number of Output Neurons

Table 3 shows the impact of the number of output neurons (N_{expert}) on the simulation runtime and testing accuracy with $M = 256$, $P = 0.01$, $N_{step} = 4$ setup. Since the core usage increases along with N_{expert} in the same ratio, the training, labeling, and testing time are consistent. Note that the testing accuracy increases when we use more experts. Using 1,600 experts, we achieve 8.3% accuracy enhancement compared to the testing result with 100 experts. If there are too few a number of experts, then they might take frequent training overriding the existing expertise even though the incoming feature comes from a different label. Therefore, a large number of experts is favorable to preserve the previously learned expertise, and to deposit new features safely.

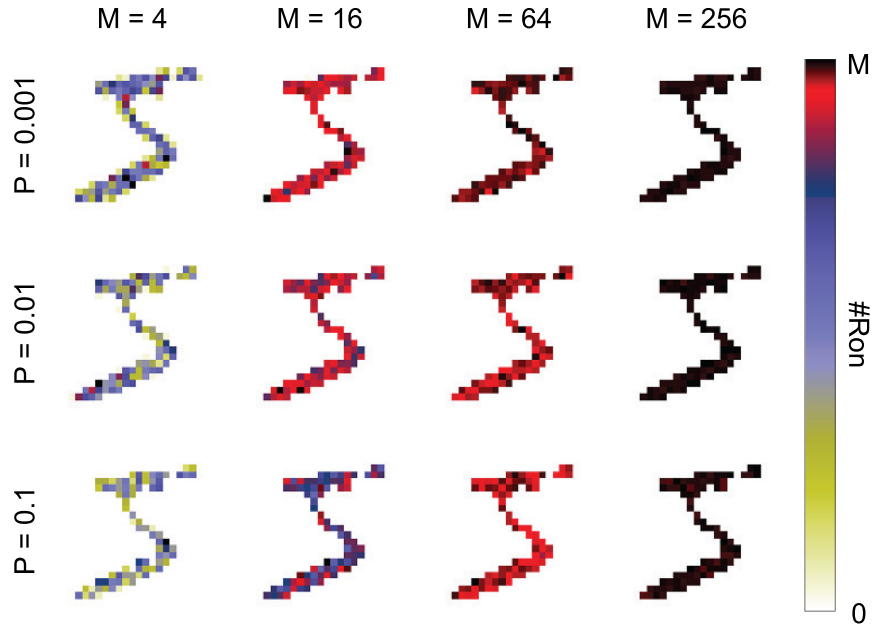

Fig. 9. The receptive fields after full repetition. P denotes the switching probability and M the number of binary-state memristor constituents in a synapse. We count the number of memristors in LRS ($\#R_{on}$) to show the synaptic precision. When $M = 256$, relatively large repetition count is required but it captures the most important feature of an input image. However, when $M = 4$, small number of repetition allows fast learning, but it does not fully capture the important features, which would degrade the learning performance.

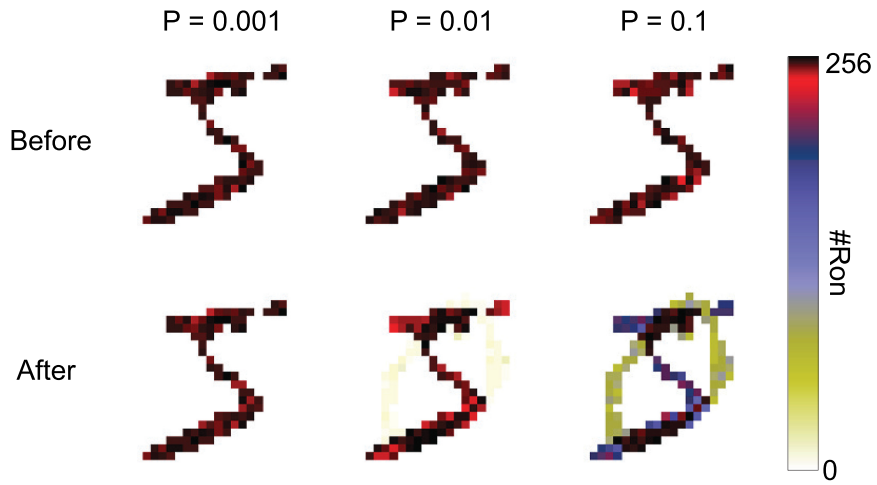

Fig. 10. After the full repetition, we feed the next training image only once and compare the receptive field before and after the presentation of the new image. When $P = 0.001$, single presentation of the new image does not affect the previous synaptic states. However, when $P = 0.1$, the neuron starts to adapt to the new feature very quickly. Too-large P, therefore, does not help to converge the learning performance, since it makes a neuron s fast learner but a fast forgetter at the same time.

Table 3. The Impact of the Number of Output Neurons on the Simulation Runtime and Testing Accuracy ($M = 256$, $P = 0.01$, $N_{step} = 4$)

Expert Count	Core Usage	Training (H:M:S)	Labeling (H:M:S)	Testing (H:M:S)	Accuracy (%)
100	20	10:12:12	10:33:32	0:54:41	85.56
400	80	10:16:16	10:38:53	0:54:49	89.24
900	180	10:20:05	10:42:12	0:54:46	92.05
1600	320	10:26:09	10:54:24	0:54:59	92.64

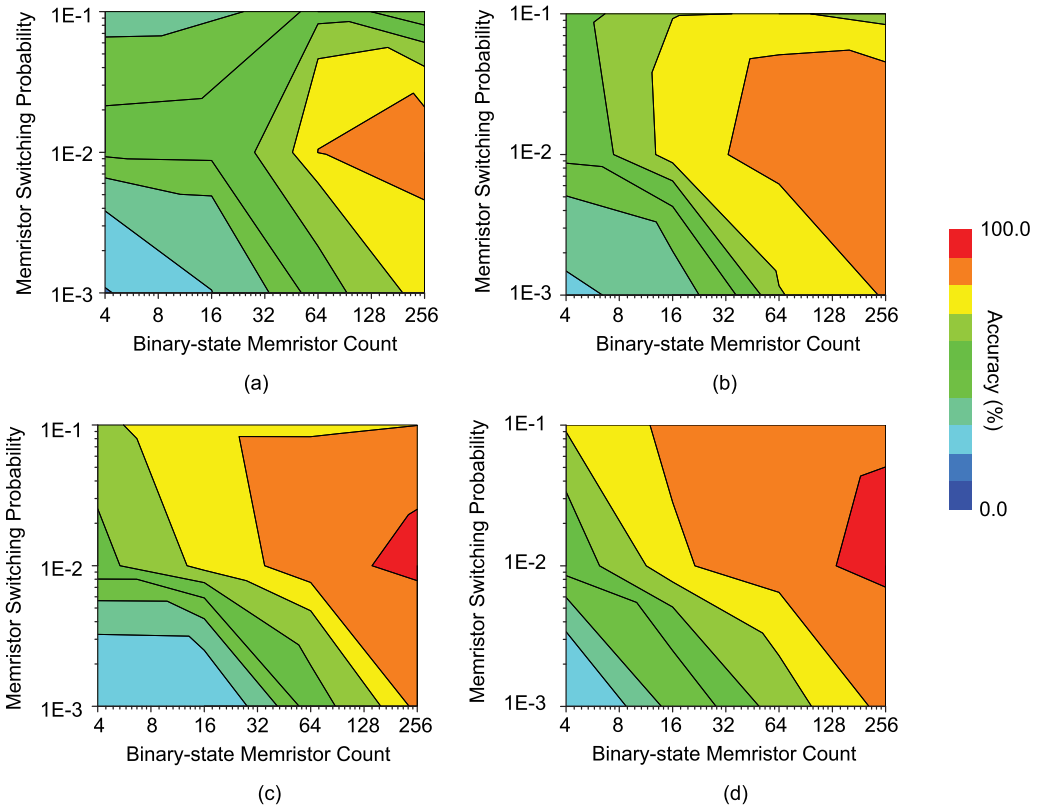

Fig. 11. Accuracy vs. Synapse Parameters: (a) 100 neurons, (b) 400 neurons, (c) 900 neurons, and (d) 1,600 neurons.

The impact of synapse parameters changes when we increase N_{expert}. Figure 11 shows the impact of P and M on the testing accuracy. We observe that using a small number of binary-state memristors in a synapse limits the precision of expertise and directly degrades the testing accuracy. Therefore, the synaptic precision turns out to be the critical factor to the learning performance. However, the switching probability of the memristor has a dynamic impact on the testing accuracy. Regardless of how we change the number of output neurons, we observe the sweet spot around $P = 0.01$. However, it is found that increasing N_{expert} makes the optimal switching probability region larger. This is because when more experts exist, there are more places to store the expertise so that fast learning does not affect the previously learned expertise.

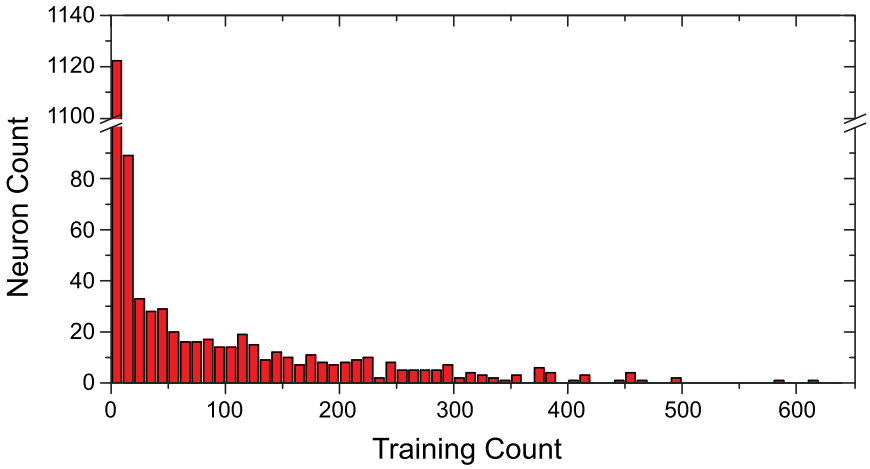

Fig. 12. Training count histogram. Bin size is 10. The X axis is training count, and the Y axis is the number of output neurons that take the respective training count. A total of 1,120 output neurons takes fewer than 10 training chances.

Based on the extensive network simulations and parameter tuning, we find that 1,600 output neurons, 256 memristor count per synapse, 4 spike encoding timesteps, and 0.01 memristor switching probability provides the 92.64% testing accuracy at its best. Now, we take a close look at the training result with 1,600 output neurons. We count the training history of each expert as shown in Figure 12. We find that 755 experts take only a single training and 2 for 153 experts and 2 for 73 experts. A total of 1,120 output neurons takes fewer than 10 trainings. The maximum training is done as many as 646 on a single expert, whose expertise for the straight-up 1 digit. Figure 13 shows a respective receptive fields of 1,600 experts after a single epoch of training. The background for each cell is colored by its training count. Although many experts take only a single training, these immature expertises are still found important, since the exceptional and new testing image could be close to those minor expertises. With 10,000 testing images, Figure 14 presents the confusion matrix, showing that 5 and 8 are commonly inferred by 3 and 4 and 7 by 9 easily. It also summarizes 736 failed testing images.

6 STATE-OF-THE-ART COMPARISON AND FUTURE WORKS

We tabulate unsupervised handwritten digit recognition learning performance of state-of-the-art neuromemristive systems in Table 4. All studies listed up have trained and tested the entire MNIST [30] dataset without any preprocessing. Most of previous works [33, 44, 47, 51] employed rate code, which adds lots of complexity/runtime overheads for the hardware/software implementation of their architectures. Also they used multi-state memristors, whose stochastic switching becomes the system noise to suppress and again increases hardware complexity. Furthermore, they trained their networks with three epochs, which helps improve the learning performance but at the expense of system resources.

Interestingly, Zhou et al. [63] presented a hardware-friendly neuromemristive architecture that utilizes temporal code for spike encoding. Their neuron model was surprisingly simplified, such as integrate-and-fire without leakage, single spike communication, no lateral inhibition, fixed threshold. In addition, the network architecture was minimal (shallow two-layer WTA without homeostasis) while achieving 94.6% learning performance with 6,400 neurons. However, their temporal code

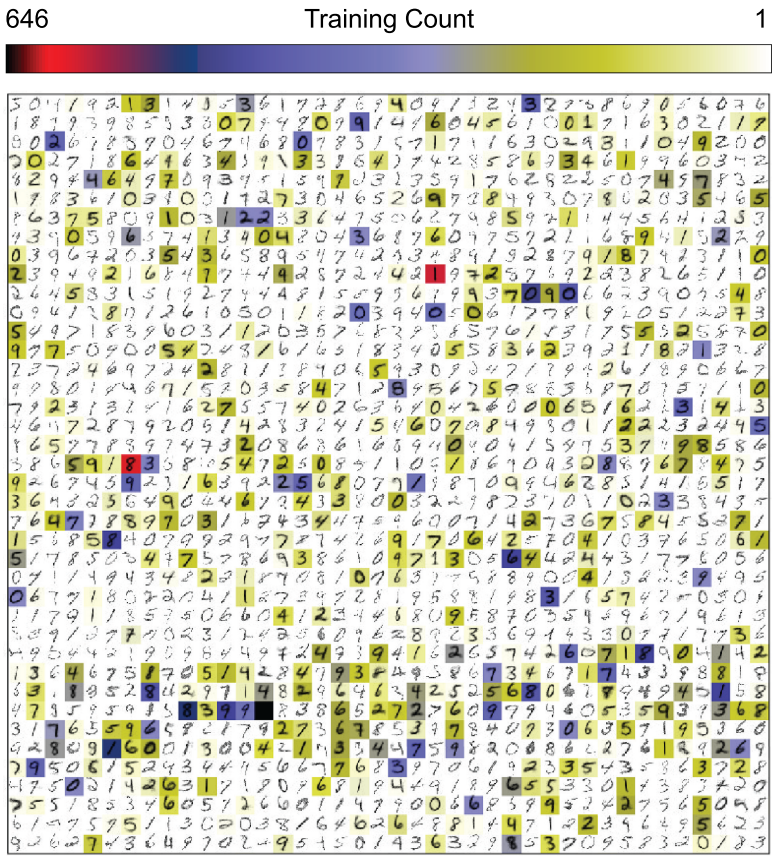

Fig. 13. Receptive fields of 1,600 output neurons after a single epoch of training. The background for each cell is colored by its training count.

turns an input image into the time-domain feature only, and the time frame is still assumed to be continuous while targeting digital implementation. Multi-state memristors are also used there, which is susceptible to weight variation. Moreover, they assumed bounded exponential STDP, which adds more parameters to set and does not sound hardware friendly. Last, their majority voting process is not clear to determine the winner at ties.

Compared to Reference [63], our neuromemristive architecture presents single-spike temporal code that transforms input stimuli into both time and voltage-domain features. This encoding scheme makes STDP implementation and cosine similarity calculation straightforward. Also we adopt the compound memristor model in our simulation so that we actively utilize the stochastic switching characteristic of memristor devices. Our STDP implementation is more hardware-friendly in that we only use step-wise learning curve, rather than using exponential continuum. Last, our network size is only 25% of them, while achieving comparable learning performance.

Considering great classification accuracy achieved by conventional deep learning algorithms over the past few years [43, 57], it is true that learning algorithms for the neuromemristive architecture are a much less mature yet active field of research. Recall that one of the critical motivations for the rebirth of the neuromorphic computing paradigm [13, 29, 36] is the great energy efficiency

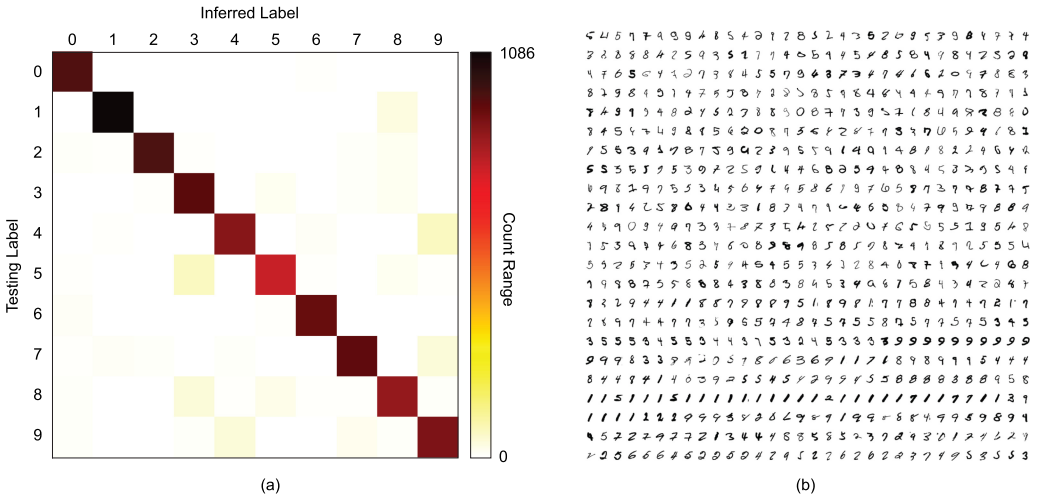

Fig. 14. (a) Confusion matrix (Testing Label vs. Inferred Label). Of 10,000 testing images, we observe that 5 and 8 are confused with 3, and 4 and 7 with 9 easily. (b) Seven hundred thirty-six incorrectly inferred testing images.

Table 4. State-of-the-art Unsupervised Handwritten Digit Recognition Learning Performance of Neuromemristive Architectures

Works	Encoding	Spike Format	Memristor	Network	Neuron#	Epochs	Accuracy
[33]	Rate	Poisson Train	Multi-state	Two-layer FCNN	100	3	82.0%
[47]	Rate	Poisson Train	Multi-state	RBM	300	3	89.4%
[44]	Rate	Poisson Train	Multi-state	Two-layer FCNN	300	3	93.5%
[51]	Rate	Poisson Train	Multi-state	Two-layer FCNN	500	3	94.0%
[63]	Temporal	Single Spike	Multi-state	Two-layer FCNN	6400	1	94.6%
This work	Temporal	Single Spike	Compound	Two-layer FCNN	1600	1	92.6%

on the learning tasks [22, 23]. Although we have focused on the network-level simulation to prove the effectiveness of proposed neuromorphic architecture in this work, exploring both learning performance and energy efficiency metrics based on actual hardware implementation will enable fair justification of neuromemristive architecture over the convenctional deep learning systems in the future work. In addition, hardware simulations along with advanced compact models of various memristor devices will allow us to study the impact of practical system parameters on the learning performance of neuromemristive systems.

7 CONCLUSION

In this work, we built a two-layer fully connected compound neuromemristive spiking neural network with hard WTA network motif. For the spike encoding scheme, we presented single-spike temporal code that turns the input image into the set of single spikes with different latencies and voltage levels. We did not perform any pre-processing on the dataset and pixel values in an image are directly transformed into the set of sparse spikes, resulting in specific sequence of spike arrival time representing the input feature naturally. For the synaptic device, we adopted the compound memristors, which connects the binary-state memristors in parallel to provide full scalability on the weight precision. We proposed the integrate-and-fire neuron model while adopting the WTA competition based on cosine similarity. By merging all these proposed models, our high-level C++

network simulation results showed the impact of model parameters on the learning performance in detail and finally offered successful unsupervised classification performance on the MNIST dataset with up to 92.64% accuracy.

REFERENCES

[1] Gina C. Adam et al. 2017. 3-D memristor crossbars for analog and neuromorphic computing applications. *IEEE Trans. Electr. Devices* 64, 1 (2017), 312–318.

[2] Edgar D. Adrian. 1926. The impulses produced by sensory nerve endings. *J. Physiol.* 61, 1 (1926), 49–72.

[3] Maruan Al-Shedivat, Rawan Naous, Gert Cauwenberghs, and Khaled Nabil Salama. 2015. Memristors empower spiking neurons with stochasticity. *IEEE J. Emerg. Select. Top. Circ. Syst.* 5, 2 (2015), 242–253.

[4] Amir Babaie-Fishani and Pieter Rombouts. 2016. Highly linear VCO for use in VCO-ADCs. *Electr. Lett.* 52, 4 (2016), 268–270.

[5] I. G. Baek et al. 2011. Realization of vertical resistive memory (VRRAM) using cost effective 3D process. In *Proceedings of the IEEE International Electron Devices Meeting (IEDM'11)*. IEEE, 31–8.

[6] Johannes Bill and Robert Legenstein. 2014. A compound memristive synapse model for statistical learning through STDP in spiking neural networks. *Front. Neurosci.* 8 (2014), 412.

[7] Romain Brette. 2015. Philosophy of the spike: Rate-based vs. spike-based theories of the brain. *Front. Syst. Neurosci.* 9 (2015), 151.

[8] Geoffrey W. Burr et al. 2017. Neuromorphic computing using non-volatile memory. *Adv. Phys.: X* 2, 1 (2017), 89–124.

[9] An Chen. 2016. A review of emerging non-volatile memory (NVM) technologies and applications. *Solid-State Electr.* 125 (2016), 25–38.

[10] J. Anthony Deutsch. 1971. The cholinergic synapse and the site of memory. *Science* 174, 4011 (1971), 788–794.

[11] Peter U. Diehl and Matthew Cook. 2015. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci.* 9 (2015), 99.

[12] Michael Epstein et al. 2003. Design and implementation of a true random number generator based on digital circuit artifacts. In *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 152–165.

[13] Steve Furber and Steve Temple. 2008. Neural systems engineering. In *Computational Intelligence: A Compendium*. Springer, 763–796.

[14] Tim Gollisch and Markus Meister. 2008. Rapid neural coding in the retina with relative spike latencies. *Science* 319, 5866 (2008), 1108–1111.

[15] J. Göltz et al. 2020. Fast and deep neuromorphic learning with first-spike coding. In *Proceedings of the Neuro-Inspired Computational Elements Workshop (NICE'20)*. 1–3.

[16] Stefan Habenschuss, Johannes Bill, and Bernhard Nessler. 2012. Homeostatic plasticity in Bayesian spiking networks as expectation maximization with posterior constraints. In *Advances in Neural Information Processing Systems*, Vol. 25. 773–781.

[17] Raqibul Hasan and Tarek M. Taha. 2017. Memristor crossbar based winner take all circuit design for self-organization. In *Proceedings of the Neuromorphic Computing Symposium*. 1–4.

[18] Donald Olding Hebb. 2005. *The Organization of Behavior: A Neuropsychological Theory*. Psychology Press.

[19] Miao Hu et al. 2016. Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication. In *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 19.

[20] YeonJoo Jeong, Jihang Lee, John Moon, Jong Hoon Shin, and Wei D Lu. 2018. K-means data clustering with memristor networks. *Nano Lett.* 18, 7 (2018), 4447–4453.

[21] Sung Hyun Jo et al. 2010. Nanoscale memristor device as synapse in neuromorphic systems. *Nano Lett.* 10, 4 (2010), 1297–1301.

[22] Norman P. Jouppi, et al. 2017. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA'17)*. IEEE, 1–12.

[23] Eric R. Kandel et al. 2000. *Principles of Neural Science*. Vol. 4. McGraw–Hill, New York.

[24] Samuel Kaski and Teuvo Kohonen. 1994. Winner-take-all networks for physiological models of competitive learning. *Neural Netw,* 7, 6-7 (1994), 973–984.

[25] Saeed Reza Kheradpisheh et al. 2016. Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition. *Neurocomputing* 205 (2016), 382–392.

[26] Hyongsuk Kim et al. 2012. Memristor bridge synapses. *Proc. IEEE* 100, 6 (2012), 2061–2070.

[27] Yongtae Kim, Yong Zhang, and Peng Li. 2012. A digital neuromorphic VLSI architecture with memristor crossbar synaptic array for machine learning. In *Proceedings of the IEEE International SOC Conference*. IEEE, 328–333.

[28] Olga Krestinskaya, Alex Pappachen James, and Leon Ong Chua. 2019. Neuromemristive circuits for edge computing: A review. *IEEE Trans. Neural Netw. Learn. Syst.* 31, 1 (2019), 4–23.
[29] Simon B. Laughlin and Terrence J. Sejnowski. 2003. Communication in neuronal networks. *Science* 301, 5641 (2003), 1870–1874.
[30] Yann LeCun. 1998. The MNIST Database of Handwritten Digits. Retrieved from http://yann.lecun.com/exdb/mnist/.
[31] Can Li et al. 2017. Three-dimensional crossbar arrays of self-rectifying Si/SiO 2/Si memristors. *Nat. Commun.* 8 (2017), 15666.
[32] Z. Li, B. Yan, and H. Li. 2020. ReSiPE: ReRAM-based single-spiking processing-in-memory engine. In *Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC'20)*.
[33] F. Liu and C. Liu. 2018. A memristor based unsupervised neuromorphic system towards fast and energy-efficient GAN. arXiv:1806.01775. Retrieved from https://arxiv.org/abs/1806.01775.
[34] Chunjie Luo, Jianfeng Zhan, Xiaohe Xue, Lei Wang, Rui Ren, and Qiang Yang. 2018. Cosine normalization: Using cosine similarity instead of dot product in neural networks. In *Proceedings of the International Conference on Artificial Neural Networks*. Springer, 382–391.
[35] Timothée Masquelier and Simon J. Thorpe. 2007. Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS Comput. Biol.* 3, 2 (2007), e31.
[36] C. Mead. 1990. Neuromorphic electronic systems. *Proc. IEEE* 78, 10 (Oct. 1990), 1629–1636.
[37] Jagan Singh Meena et al. 2014. Overview of emerging nonvolatile memory technologies. *Nanosc. Res. Lett.* 9, 1 (2014), 526.
[38] Cory Merkel, Raqibul Hasan, Nicholas Soures, Dhireesha Kudithipudi, Tarek Taha, Sapan Agarwal, and Matthew Marinella. 2016. Neuromemristive systems: Boosting efficiency through brain-inspired computing. *Computer* 49, 10 (2016), 56–64.
[39] Cory Merkel and Dhireesha Kudithipudi. 2015. Unsupervised learning in neuromemristive systems. In *Proceedings of the National Aerospace and Electronics Conference (NAECON'15)*. IEEE, 336–338.
[40] Rawan Naous, Maruan Al-Shedivat, and Khaled Nabil Salama. 2016. Stochasticity modeling in memristors. *IEEE Trans. Nanotechnol.* 15, 1 (2016), 15–28.
[41] Joost Pieterse and Decebal Constantin Mocanu. 2019. Evolving and understanding sparse deep neural networks using cosine similarity. arXiv:1903.07138. Retrieved from https://arxiv.org/abs/1903.07138.
[42] Steven A. Prescott and Terrence J. Sejnowski. 2008. Spike-rate coding and spike-time coding are affected oppositely by different adaptation mechanisms. *J. Neurosci.* 28, 50 (2008), 13649–13661.
[43] Junfei Qiao, Gongming Wang, Wenjing Li, and Min Chen. 2018. An adaptive deep Q-learning strategy for handwritten digit recognition. *Neural Netw.* 107 (2018), 61–71.
[44] Damien Querlioz et al. 2013. Immunity to device variations in a spiking neural network with memristive nanodevices. *IEEE Trans. Nanotechnol.* 12, 3 (2013), 288–295.
[45] Pamela Reinagel and R. Clay Reid. 2000. Temporal coding of visual information in the thalamus. *J. Neurosci.* 20, 14 (2000), 5392–5400.
[46] Pau Rodríguez, Jordi Gonzalez, Guillem Cucurull, Josep M. Gonfaus, and Xavier Roca. 2016. Regularizing cnns with locally constrained decorrelations. arXiv:1611.01967. Retrieved from https://arxiv.org/abs/1611.01967.
[47] Mahyar Shahsavari, Pierre Falez, and Pierre Boulet. 2016. Combining a volatile and nonvolatile memristor in artificial synapse to improve learning in spiking neural networks. In *Proceedings of the IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH'16)*. IEEE, 67–72.
[48] Carla J. Shatz. 1992. The developing brain. *Sci. Am.* 267, 3 (1992), 60–67.
[49] Ahmad Muqeem Sheri et al. 2015. Contrastive divergence for memristor-based restricted Boltzmann machine. *Eng. Appl. Artif. Intell.* 37 (2015), 336–342.
[50] Patrick Sheridan, Wen Ma, and Wei Lu. 2014. Pattern recognition with memristor networks. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'14)*. IEEE, 1078–1081.
[51] Yuhan Shi et al. 2018. Neuroinspired unsupervised learning and pruning with subquantum CBRAM arrays. *Nat. Commun.* 9, 1 (2018), 1–11.
[52] Daniel Soudry, Dotan Di Castro, Asaf Gal, Avinoam Kolodny, and Shahar Kvatinsky. 2013. *Hebbian Learning Rules with Memristors*. Israel Institute of Technology, Haifa, Israel.
[53] Spyros Stathopoulos et al. 2017. Multibit memory operation of metal-oxide bi-layer memristors. *Sci. Rep.* 7, 1 (2017), 17532.
[54] Toni Stojanovski, Johnny Pihl, and Ljupco Kocarev. 2001. Chaos-based random number generators. Part II: practical realization. *IEEE Trans. Circ. Syst. I: Fundam. Theory Appl.* 48, 3 (2001), 382–385.
[55] Dmitri B. Strukov, Duncan R. Stewart, Julien Borghetti, Xuema Li, M. Pickett, G. Medeiros Ribeiro, Warren Robinett, G. Snider, John Paul Strachan, Wei Wu, et al. 2010. Hybrid CMOS/memristor circuits. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'10)*. IEEE, 1967–1970.

[56] Jonathan D. Victor and Keith P. Purpura. 1996. Nature and precision of temporal coding in visual cortex: A metric-space analysis. *J. Neurophysiol.* 76, 2 (1996), 1310–1326.
[57] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *Proceedings of the International Conference on Machine Learning*. PMLR, 1058–1066.
[58] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. 2017. Normface: L2 hypersphere embedding for face verification. In *Proceedings of the 25th ACM International Conference on Multimedia*. 1041–1049.
[59] Chris Yakopcic, Md Zahangir Alom, and Tarek M. Taha. 2016. Memristor crossbar deep network implementation based on a convolutional neural network. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'16)*. IEEE, 963–970.
[60] Shimeng Yu and Pai-Yu Chen. 2016. Emerging memory technologies: Recent trends and prospects. *IEEE Solid-State Circ. Mag.* 8, 2 (2016), 43–56.
[61] Shimeng Yu, Ximeng Guan, and H.-S. Philip Wong. 2011. On the stochastic nature of resistive switching in metal oxide RRAM: Physical modeling, Monte Carlo simulation, and experimental characterization. In *Proceedings of the IEEE International Conference on Electron Devices Meeting (IEDM'11)*. IEEE, 17–3.
[62] Yang Zhang, Xiaoping Wang, and Eby G. Friedman. 2017. Memristor-based circuit design for multilayer neural networks. *IEEE Trans. Circ. Syst. I: Regul. Pap.* 65, 2 (2017), 677–686.
[63] Errui Zhou, Liang Fang, and Binbin Yang. 2018. Memristive spiking neural networks trained with unsupervised STDP. *Electronics* 7, 12 (2018), 396.
[64] Ruohua Zhu, Shizhuo Ye, Zhiri Tang, Peng Lin, Qijun Huang, Hao Wang, Jin He, and Sheng Chang. 2019. Influence of compact Memristors' stability on machine learning. *IEEE Access* 7 (2019), 47472–47478.
[65] Mohammed Affan Zidan et al. 2013. Memristor-based memory: The sneak paths problem and solutions. *Microelectr. J.* 44, 2 (2013), 176–183.
[66] Mohammed A. Zidan, John Paul Strachan, and Wei D. Lu. 2018. The future of electronics based on memristive systems. *Nat. Electr.* 1, 1 (2018), 22.

Received August 2020; revised April 2021; accepted June 2021