UNIVERSITY OF CALIFORNIA SAN DIEGO

Energy Efficient Hardware Implementation of Neural Networks Using Emerging Non-Volatile Memory Devices

A Dissertation submitted in partial satisfaction of the requirements
for the degree Doctor of Philosophy

in

Electrical Engineering (Applied Physics)

by

Sangheon Oh

Committee in charge:

Professor Duygu Kuzum, Chair
Professor Gert Cauwenberghs
Professor Tse Nga Ng
Professor Ivan K Schuller
Professor Yuan Taur

2023

The Dissertation of Sangheon Oh is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest appreciation to my advisor Professor Duygu Kuzum for her invaluable guidance and support throughout my whole phD study. I am also deeply indebted to Professor Ivan K Schuller and Professor Gert Cauwenberghs for their irreplaceable support for our collaborative projects. Additionally, I would like to extend my sincere thanks to Professor Tse Nga (Tina) Ng and Professor Yuan Taur for their valuable feedback and suggestions as committee members for my dissertation.

I am also extremely grateful to Yuhan Shi for her help on all my research projects. Special thanks to Yucheng Zhou and Dr. Jaeseoung Park for always being a company for tiresome cleanroom fabrications. I also would like to mention that I had the pleasure of working with my colleagues, Yichen Lu, Xin Liu, Zhisheng Huang, Jeonghoon Kim, Madison Wilson, Mehrdad Ramezani, Dr. Akshay Ananthakrishnan, Shawn Fisher, and Dr. Ashwani Kumar. I am deeply indebted to Dr. Javier del Valle, Dr. Pavel Salev, and Dr. Yoav Kalcheim. They generously provided their knowledge and expertise to expedite the progress of our collaborative work. Many thanks to Soumil Jain and Gopabandhu Hota. The collaborative project with them could have not been advanced without them.

I would be remiss in not mentioning my family. Without their unwavering supports and encouragement, I could not go through challenging times during my PhD study. My spouse Minah Yoo's selfless support has been truly invaluable and made me unswerving in taking journey thus far. My mother and my little brother always provided unconditional support all through my studies even during their sadness time. Lastly, my endless gratitude to my father for his unwavering guidance and support. I always feel your presence even though you're gone.

Chapter 1, in full, is a reprint of the material as it appears in IEEE Electron Device Letters. Oh, Sangheon; Shi, Yuhan; Liu, Xin; Song, Jungwoo; Kuzum, Duygu, Drift-Enhanced Unsupervised Learning of Handwritten Digits in Spiking Neural Network with PCM Synapses, 2018. The dissertation author was the primary author of this paper.

Chapter 2, in full, is a reprint of the material as it appears in IEEE Electron Device Letters. Oh, Sangheon; Huang, Zhisheng; Shi, Yuhan; Kuzum, Duygu, The Impact of Resistance Drift of Phase Change Memory (PCM) Synaptic Devices on Artificial Neural Network Performance, 2019. The dissertation author was the primary researcher and author of this paper.

Chapter 3, in full, is a reprint of the material as it appears in Nature Communications. Shi, Yuhan; Nguyen, Leon; Oh, Sangheon; Liu, Xin; Koushan, Foroozan; Jameson, John, R.; Kuzum, Duygu, Neuroinspired unsupervised learning and pruning with subquantum CBRAM arrays, 2019. The dissertation author was a co-author of this paper.

Chapter 4, in full, has been submitted for publication of the material as it may appear in IOP Neuromorphic Computing, Shi, Yuhan; Oh, Sangheon; Valle, Javier del; Salev, Pavel; Schuller, Ivan K; Kuzum, Duygu, Integration of Ag-CBRAM Crossbars and Mott-Relu Neurons for Efficient Implementation of Deep Neural Networks in Hardware, 2023. The dissertation author was a co-author of this paper.

Chapter 5, in full, is a reprint of the material as it appears in IEEE Transactions on Electron Devices, Shi, Yuhan; Ananthakrishnan, Akshay, Oh, Sangheon; Liu, Xin; Hota, Gopabandhu; Cauwenberghs, Gert; Kuzum, Duygu, A Neuromorphic Brain Interface Based on RRAM Crossbar Arrays for High Throughput Real-Time Spike Sorting, 2021. The dissertation author was a co-author of this paper.

Chapter 6, in full, is a reprint of the material as it appears in Nature Nanotechnology, Oh, Sangheon; Shi, Yuhan; Valle, Javier Del; Salev, Pavel; Lu, Yichen; Huang, Zhisheng; Kalcheim; Yoav; Schuller; Ivan K; Kuzum, Duygu, Energy-efficient Mott activation neuron for full-hardware implementation of neural networks, 2021. The dissertation author was the primary author of this paper.

VITA

2015 Bachelor of Science in Electrical and Computer Engineering, University of Seoul

2018 Master of Science in Electrical and Computer Engineering, University of Seoul

2023 Doctor of Philosophy in Electrical Engineering (Applied Physics), University of California San Diego

ABSTRACT OF THE DISSERTATION

Energy Efficient Hardware Implementation of Neural Networks Using Emerging Non-Volatile Memory Devices

by

Sangheon Oh

Doctor of Philosophy in Electrical Engineering (Applied Physics)

University of California San Diego, 2023

Professor Duygu Kuzum, Chair

Deep learning based on neural networks emerged as a robust solution to various complex problems such as speech recognition and visual recognition. Deep learning relies on a great amount of iterative computation on a huge dataset. As we need to transfer a large amount of data and program between the CPU and the memory unit, the data transfer rate

through a bus becomes a limiting factor for computing speed, which is known as Von Neumann bottleneck. Moreover, the data transfer between memory and computation spends a large amount of energy and cause significant delay. To overcome the limitation of Von Neumann bottleneck, neuromorphic computing with emerging nonvolatile memory (eNVM) devices has been proposed to perform iterative calculations in memory without transferring data to a processor. This dissertation presents energy efficient hardware implementation of neuromorphic computing applications using phase change memory (PCM), subquantum conductive bridge random access memory (CBRAM), Ag-based CBRAM, and CuO_x-based resistive random access memory (RRAM).

Although substantial progress has been made towards in-memory computing with synaptic devices, compact nanodevices implementing non-linear activation functions for efficient full-hardware implementation of deep neural networks is still missing. Since DNNs need to have a very large number of activations to achieve high accuracy, it is critical to develop energy and area efficient implementations of activation functions, which can be integrated on the periphery of the synaptic arrays. In this dissertation, we demonstrate a Mott activation neuron that implements the rectified linear unit function in the analogue domain. The integration of Mott activation neurons with a CBRAM crossbar array is also demonstrated in this dissertation.

INTRODUCTION

Neural networks have become a popular tool for machine learning and artificial intelligence, due to their capability to learn complex patterns from excessively large dataset and make predictions or decisions[1-5]. However, their widespread adoption is hindered by the high energy consumption and latency of the hardware when they are implemented on conventional Von Neumann architecture. The conventional computing architecture has excessive delay and energy consumption arise from data transfer between memory and processor, so-called Von Neumann bottleneck (i.e., the energy consumption for data transfer is 170 times larger than that for the computation of the same data)[6,7]. Due to this Von Neumann bottleneck, integrating data-intensive neural networks with the conventional computer architecture leads to extremely large energy consumption and delay on the hardware.

One promising solution to this problem is the use of in-memory computing architecture with emerging non-volatile memory (eNVM) devices[7,8], such as phase-change memory (PCM)[9-12], resistive random access memory (ReRAM)[13], conductive bridge random access memory (CBRAM)[14,15], and metal oxide based resistive random access memory (RRAM)[16,17]. The in-memory computing architecture uses the large and high-speed memory to store and process data in-place, without the need to transfer data between memory and storage[18,19]. Data is stored in arrays of eNVM devices and the computations (e.g. weighted sum) are performed using Kirchhoff's current law[20]. Since the slow and energy consuming data transfer can be minimized in the in-memory computing architecture, neural networks on the in-memory computing architecture can achieve high energy efficiency with low latency[21].

In this dissertation, we explore the potential of using eNVM devices for the hardware implementation of neural networks. We evaluate our approach using both simulation and hardware experiments and show that it can significantly reduce the energy consumption and latency of neural network hardware while maintaining high performance.

In Chapter 1, we investigate the impact of resistance drift and the variations in resistance drift parameters during unsupervised online learning of a spiking neural network (SNN). We use the resistance drift characteristics measured from experiments and incorporate them into the SNN for MNIST handwritten digits classification.

In Chapter 2, we employ experimentally measured resistance drift characteristics into the artificial neural network models, multilayer perceptron (MLP) and convolutional neural network (CNN), to accurately model weight updates represented by PCM synaptic devices.

In chapter 3, using a subquantum conductive bridge random access memory (CBRAM) array, we experimentally demonstrate high recognition accuracy on the MNIST dataset for digital implementation of unsupervised learning.

In Chapter 4, we present integration of Ag-based CBRAM crossbar arrays with Mott-ReLU activation neurons for scalable, energy and area efficient hardware implementation of deep neural networks.

In Chapter 5, we present a high-density CuO_x resistive crossbars to implement real-time, low latency spike sorting processor that utilizes in-memory computations in a massively parallel manner.

In Chapter 6, we present an energy-efficient and compact Mott activation neuron based on vanadium dioxide (VO_2) and its integration with a CBRAM array in hardware.

References

1 LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. nature 521, 436 (2015).

2 Krizhevsky, A., Sutskever, I. & Hinton, G. E. in Advances in neural information processing systems. 1097-1105.

3 Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A. & Bernstein, M. Imagenet large scale visual recognition challenge. International Journal of Computer Vision 115, 211-252 (2015).

4 Collobert, R. & Weston, J. in Proceedings of the 25th international conference on Machine learning. 160-167 (ACM).

5 Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P. & Sainath, T. N. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine 29, 82-97 (2012).

6 Wong, H. S. P., Lee, H.-Y., Yu, S., Chen, Y.-S., Wu, Y., Chen, P.-S., Lee, B., Chen, F. T. & Tsai, M.-J. Metal–Oxide RRAM. Proceedings of the IEEE 100, 1951-1970, doi:10.1109/jproc.2012.2190369 (2012).

7 Zidan, M. A., Strachan, J. P. & Lu, W. D. The future of electronics based on memristive systems. Nature Electronics 1, 22-29, doi:10.1038/s41928-017-0006-8 (2018).

8 Yu, S. Neuro-Inspired Computing With Emerging Nonvolatile Memorys. Proceedings of the IEEE 106, 260-285, doi:10.1109/jproc.2018.2790840 (2018).

9 Wong, H. S. P., Raoux, S., Kim, S., Liang, J., Reifenberg, J. P., Rajendran, B., Asheghi, M. & Goodson, K. E. Phase Change Memory. Proceedings of the IEEE 98, 2201-2227, doi:10.1109/JPROC.2010.2070050 (2010).

10 Kuzum, D., Jeyasingh, R. G., Lee, B. & Wong, H. S. Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing. Nano Lett 12, 2179-2186, doi:10.1021/nl201040y (2012).

11 Bichler, O., Suri, M., Querlioz, D., Vuillaume, D., DeSalvo, B. & Gamrat, C. Visual Pattern Extraction Using Energy-Efficient "2-PCM Synapse" Neuromorphic Architecture. IEEE Transactions on Electron Devices 59, 2206-2214, doi:10.1109/TED.2012.2197951 (2012).

12 Eryilmaz, S. B., Kuzum, D., Jeyasingh, R. G. D., Kim, S., BrightSky, M., Lam, C. & Wong, H. S. P. in 2013 IEEE International Electron Devices Meeting. 25.25.21-25.25.24.

13 Chen, Y. ReRAM: History, Status, and Future. IEEE Transactions on Electron Devices 67, 1420-1433, doi:10.1109/ted.2019.2961505 (2020).

14 Shi, Y., Nguyen, L., Oh, S., Liu, X., Koushan, F., Jameson, J. R. & Kuzum, D. Neuroinspired unsupervised learning and pruning with subquantum CBRAM arrays. Nature Communications 9, 5312, doi:10.1038/s41467-018-07682-0 (2018).

15 Suri, M., Querlioz, D., Bichler, O., Palma, G., Vianello, E., Vuillaume, D., Gamrat, C. & DeSalvo, B. Bio-Inspired Stochastic Computing Using Binary CBRAM Synapses. IEEE Transactions on Electron Devices 60, 2402-2409, doi:10.1109/ted.2013.2263000 (2013).

16 Grenouillet, L., Castellani, N., Persico, A., Meli, V., Martin, S., Billoint, O., Segaud, R., Bernasconi, S., Pellissier, C., Jahan, C., Charpin-Nicolle, C., Dezest, P., Carabasse, C., Besombes, P., Ricavy, S., Tran, N. P., Magalhaes-Lucas, A., Roman, A., Boixaderas, C., Magis, T., Bedjaoui, M., Tessaire, M., Seignard, A., Mazen, F., Landis, S., Vianello, E., Molas, G., Gaillard, F., Arcamone, J. & Nowak, E. in 2021 IEEE International Memory Workshop (IMW) 1-4 (2021).

17 Yuhan Shi, A. A., Sangheon Oh, Xin Liu, Gopabandhu Hota,Gert Cauwenberghs,Duygu Kuzum. High Throughput Neuromorphic Brain Interface with CuOx Resistive Crossbars for Real-time Spike Sorting. International Electron Devices Meeting In press (2021).

18 Cai, F., Correll, J. M., Lee, S. H., Lim, Y., Bothra, V., Zhang, Z., Flynn, M. P. & Lu, W. D. A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations. Nature Electronics 2, 290-299, doi:10.1038/s41928-019-0270-x (2019).

19 Ielmini, D. & Wong, H.-S. P. In-memory computing with resistive switching devices. Nature electronics 1, 333-343 (2018).

20 Ambrogio, S., Narayanan, P., Tsai, H., Shelby, R. M., Boybat, I., di Nolfo, C., Sidler, S., Giordano, M., Bodini, M., Farinha, N. C. P., Killeen, B., Cheng, C., Jaoudi, Y. & Burr, G. W. Equivalent-accuracy accelerated neural-network training using analogue memory. Nature 558, 60-67, doi:10.1038/s41586-018-0180-5 (2018).

21 Wan, W., Kubendran, R., Gao, B., Josbi, S., Raina, P., Wu, H., Cauwenberghs, G. & Wong, H. S. P. in 2020 IEEE Symposium on VLSI Technology. 1-2.

CHAPTER 1. DRIFT-ENHANCED UNSUPERVISED LEARNING OF HANDWRITTEN DIGITS IN SPIKING NEURAL NETWORK WITH PCM SYNAPSES

Phase change memory (PCM), one of the most mature emerging non-volatile memories, has gained considerable attention over the years for use as electronic synapses in biologically inspired neuromorphic systems. The resistance drift of PCM devices, nonetheless, has long been identified as one of the biggest challenges toward realizing many areas of applications. Although this drawback has been extensively studied for memory development and many methods were proposed to mitigate the drift effect, its impact, if any, on online learning has not been fully explored yet. In this chapter, we investigate the impact of resistance drift and variations in resistance drift parameters during unsupervised online learning. We use the resistance drift characteristics measured from experiments and incorporate them into the spiking neural network (SNN) for MNIST handwritten digits classification. Our results show that resistance drift, considered as a non-ideality for PCM devices, can be exploited to boost accuracy for online learning of handwritten digits in the SNN.

1. Introduction

Neural Networks (NNs) have provided the best solutions in various tasks such as image classification and speech recognition. As training NNs require iterative updates of a massive number of parameters, it is inefficient to use the traditional von Neumann architecture in which data continuously transfer between processor and main memory via bus. To overcome the limitation of the conventional architecture, crossbar array of resistive synaptic devices[1,2] have been developed as a promising alternative to implement NNs in an energy-efficient manner.

Among the resistive synaptic devices, phase change memory (PCM), which stores the information using gradual switching between amorphous (high resistivity) and polycrystalline (low resistivity) states of phase change material[3], has been extensively studied at both device[4] and array[5,6] levels. However, PCM has a so-called low-field resistance drift phenomenon, which refers to the increase of the resistance in amorphous state over time due to structural relaxation[7-9]. Although this drawback has been extensively investigated for memory applications and there are attempts to overcome it via clever device-level ideas[9] or drift resilient architectures[10], its impact on on-line learning has not been fully explored yet. In this chapter, we study how PCM drift affects on-line learning using an unsupervised spiking neural network (SNN). We characterize and model device drift behavior from experimental data and incorporate it in the SNN training on MNIST dataset. We demonstrate that resistance drift in PCM is not necessarily detrimental but can be utilized to assist weight evolution during unsupervised learning to enhance the classification accuracy.

2. PCM Synapse and Drift Model

A. Device Structure and Characteristics

In this chapter, we use Ge2Sb2Te5 (GST), a phase change material, to implement PCM devices as electrical synapses. GST (200 nm thick) is deposited between a bottom electrode (TiN) with a small contact area (75 nm diameter) and a top electrode (TiN) (See Fig. 1.1(a)). Figure 1.1(b) and (c) show that PCM exhibits gradual conductance responses through a few thousand cycles in both of high and low conductance (G) regime. In high G regime (See Fig. 1.1(b)), gradual set (increasing conductance) programming of the PCM devices is achieved by using staircase pulses (i.e. 20 pulses per each voltage step of 0.1 V starting from 0.5 V to 0.9 V).

Gradual reset (decreasing conductance) is achieved using pulses with increasing amplitude from 2 V to 4 V with 20 mV voltage steps. In low G regime (See Fig. 1.1(c)), gradual set is performed by stair-case pulses with an increasing step of 50 mV in the range of 1 V to 1.7 V (four pulses for each step). Gradual reset is performed by pulses with increasing amplitude in the 5.7 V − 7.3 V voltage range with 25 mV voltage steps.

B. Resistance Drift Model

After the PCM is programmed to high-resistance amorphous state (Reset), the resistance of the PCM tends to gradually increase with time[7-9] due to the resistance drift. The resistance gradually increases with time to a higher value following the power-law[8],

$$R(t) = R_0(t/t_0)^{\nu}, \quad (1)$$

where R_0 and t_0 are initial resistance and time, t is wait time, and ν is the drift coefficient. Figure 1.2(a) presents the relationship of resistance drift and wait time measured from the devices (ν = 0.1, R_0 = 2 M, and $t_0 = 10^{-3}$ s)[11]. Another important aspect of the resistance drift is that the drift is more evident in high resistance state than in low resistance state[12]. In order to accurately characterize this property, we use a model which describes the dependency of drift coefficient on resistance [12],

$$v(R_2) - v(R_1) = k \cdot log(R_2/R_1), \quad (2)$$

where k represents the dependency of drift coefficient. In this work, we use k = 0.2 based on the experimental data[11] (see Fig. 1.2(b)).

3. Neural Network Architecture

To investigate the impact of resistance drift of PCM on unsupervised learning, we implement a SNN for MNIST digit classification. We utilize a network consists of 398 input neurons (397 for image pixels + 1 for bias), 500 output neurons, and 199,000 synapses to classify MNIST handwritten digits (Fig. 1.2(c)). We retain 397 of the original 784 pixels of MNIST images by removing background pixels to reduce the complexity of the SNN. We use the unsupervised learning algorithm from[13]. The input images are converted to Poisson pre-spike trains based on the pixel intensity. Then, the output neurons integrate all the input spikes to generate output spike trains based on probabilistic winner-take-all mechanism. The synaptic weights (W) are updated via a simplified spiking time dependent plasticity (STDP) rule. If the time difference between the post-spike and pre-spike (Δt) is within a 10 ms window, the synaptic weight is updated via the long-term potentiation (LTP) rule,

$$\Delta W_{LTP} = \alpha \cdot \exp\big(-\beta(W + 1)\big), \tag{3}$$

where α and β are parameters that control the scale of the exponential and W is the current weight value. If Δt is not within the 10 ms window, it is updated via the long-term depression (LTD) rule,

$$\Delta W_{LTP} = -\gamma. \tag{4}$$

The parameters used in our simulations are α = 0.05, β = 0.5, and γ = 0.15 and the synaptic weights of the SNN are limited in the range [−1, 1]. After training the NN with the MNIST dataset (10 classes, 60,000 images), we fix the trained weights and present the training images again for labeling. We assign labels to each output neuron based on the highest average firing rate over all the training images. After the labeling, we perform inference (i.e. classification) with the MNIST test set of 10,000 images. In these simulations, we assume that

the peripheral programming circuit can apply proper programming pulses (i.e. the number of pulse and its amplitude) based on device's conductance and the weight update rules.

4. Device-Network Mapping for Simulation

In order to properly incorporate device characteristics into the SNN model, we map synaptic weights of the SNN to the PCM conductance values. Since the network has W ranging from −1 to 1 while the PCM device conductance data ranges from 10 to 230 μS under the high G regime (0.4 ~ 5.5 μS under low G regime operation), we use linear transformation to map the device conductance to the range of W (i.e. [−1, 1]) as follows: $G_{NORM} = G - (G_{max} + G_{min})/2$ $(G_{max} - G_{min})/2$ (5) where GNORM is the normalized conductance (= W), G is device conductance, and Gmax (Gmin) is the maximum (minimum) conductance of device data. For each G_{NORM} (= W), we calculate the ideal ΔW using eq. (3) and (4) and find the ΔG_{NORM} for each G_{NORM} which is closest to the ideal ΔW corresponding to the G_{NORM}.

5. Resistance Drift During On-Line Training

A. Estimation of Wait Time During Training

To analyze the impact of resistance drift on online learning with the SNN, we need to properly estimate the wait time of each PCM synapse during the training. To this end, we propose an algorithm (see Fig. 1.3(a)) which keeps track of the wait time for each synapse between weight updates during the training. Then, based on the estimated wait time, we calculate the resistance drift using eq. (1) and (2). Next, we update the weight based on the drifted resistance and the learning rule described in Section 3. During the training, synaptic devices are updated only when the corresponding output neuron generates an output spike. Therefore,

devices in the array remain idle and wait until the connected output neuron spikes. Thus, as shown in Fig. 3(b), the wait time of synaptic devices of each output neuron can be estimated by measuring the time interval between two consecutive output spikes (See Fig. 1.3(c)). Please note that we reset resistance drift of the devices (i.e. set $t = t_0$) after the devices are programmed. This resistance drift recovery phenomenon is reported by A. Pirovano et al[14].

B. Simulation Results

Figure 1.4 shows the resistance and weight distributions of the SNN consisted of low G PCMs after training for both of no-drift and drifted case. The distribution of the trained weights can be divided into two distinct parts, namely the foreground (FG) pixels (blue) and background (BG) pixels (green, yellow, and red) of the resistance map shown in Fig. 1.4(c). The BG (FG) pixels are negative (positive) weights in Fig. 1.4(a). High resistance PCM synapses representing the BG pixels of the digits drift toward the negative weight range. The weight distribution histogram shows that the number of cells with weight −1 is increased due to drift, as indicated by the red rectangular in Fig. 1.4(a). Fig. 1.4(b) shows the effect of drift on resistance distribution at the end of the training. As shown in Fig. 1.4(b), most of resistance are below ∼2 M with no drift. However, drift shifts high resistance weights into 2∼6 M resistance ranges. The effect of drift can be also understood by comparing the color of the BG pixels in the representative weight visualizations shown in Fig. 4(c). It means that the contrast between FG and BG pixels is enhanced due to the drift.

The inference results of our SNN are summarized in Fig. 1.5(a). Our ideal SNN can achieve 94.05 % and 92.02 % accuracy with 64-bit and 8-bit precision weights, respectively. SNN based on PCM device data achieves 89.38 % with high G data and 85.12 % with low G data in no-drift case. This is mainly because the weights represented by device data have limited

number of levels[15-17] (i.e. High G: ~106; Low G: ~55). The low G result has lower accuracy than the high G result because it has fewer number of levels. However, the low G regime (Gmax ~5.5 μS) is more desirable than the high G regime (Gmax ~230 μS) to minimize IR drop due to the finite resistance of word line and bit line particularly for large scale crossbar array applications (i.e. Gmax < 100 μS to prevent accuracy drop due to IR drop)[15,18].

Since the resistance drift in high G regime is negligible, the accuracy remains unchanged. For low G case, the classification accuracy is improved by 1.63 % as a result of the drift (Fig. 1.5(a) drifted case). The accuracy boost for the drift case can be understood from two perspectives: First, the resistance drift does not distort the weight distribution of FG pixels because most of these pixels lie in no-drift region (i.e. below ~0.5 M). Since the shape of the FG weight distribution is essentially preserved, the resistance drift does not show detrimental effects on NN learning. Second, the more the BG pixels are programmed to −1 weight state, the more effectively the membrane potential of an output neuron is decreased. This improves the weight representation of the learned digit by enhancing the degree of contrast between the FG and the BG pixels. As a result, false positive spiking for non-matching classes can be prevented, thereby leading to an improvement in the classification accuracy. Figure 1.5(b) shows that additional wait time between training and testing does not affect accuracy. Since all the devices get drifted enough during the training (i.e. all BG pixels drifted to W = −1), the contrast between FG and BG pixels is not enhanced further by the additional wait time. Hence, the additional wait time after training does not affect accuracy.

The impact of variation, stochasticity, linearity and asymmetry of PCM conductance change have been extensively studied in the literature[19]. Here we specifically focus on variations in drift parameters and investigate their impact on classification accuracy. The impact of cycle-

to-cycle (C2C) and deviceto-device (D2D) variations in drift parameters (v and k) on our SNN is shown in Fig. 1.5(c). For D2D variation, drift parameters of each device are fixed during the training. D2D variation only drifts a limited number of FG pixels, which have high drift coefficients. SNN can easily adapt and compensate the variation in drift characteristics during the training. As a result, the weight distribution of FG pixels is well-maintained (See Fig. 1.6(a) and (b)) and the accuracy does not get affected. In contrast, if there is C2C variation in drift parameters, some of the FG pixels with relatively high resistance end up having different drift coefficients after every update. This random change in drift behavior of the FG pixels results in distortion of the weight distribution (See Fig. 1.6(c)) and information loss leading to the accuracy decrease beyond 30-40% C2C variation.

6. Conclusions

In this chapter, we investigate the impact of PCM resistance drift and variations in resistance drift parameters on unsupervised learning of MNIST handwritten digits in spiking neural network for on-line learning applications. We model the device drift characteristics from the experimental data and incorporate it into our neural network simulation. We demonstrate that PCM drift does not degrade network performance. Instead, it can act as a mechanism boosting the classification accuracy by improving the weight representations of the learned digits.

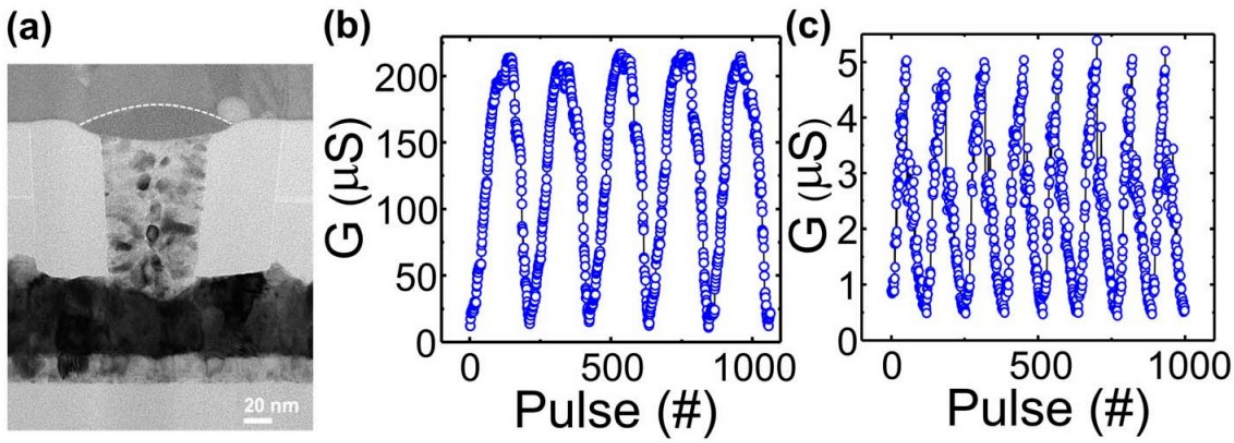


Figure 1.1: (a) A cross-section TEM image of an electronic synapse made of GST. (b) and (c) are the gradual switching characteristics of the device in high G and low G regimes, respectively. For both high G and low G regimes, we use pulses with 10 ns of pulse width, 5 ns of rise time, and 5 ns of fall time for gradual set and 20 ns of pulse width, 5 ns of rise time, and 5 ns of fall time for gradual reset.

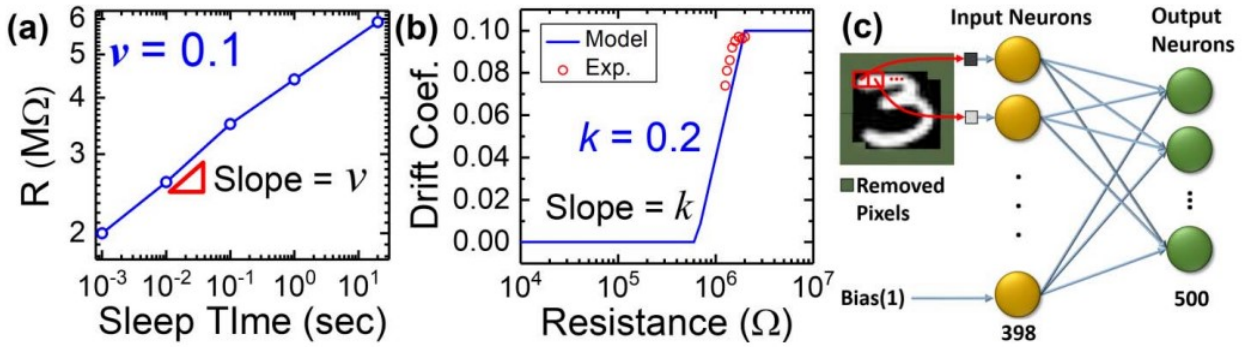


Figure 1.2: (a) Resistance drift characteristics of GST PCM. The initial resistance is 2 MΩ and it gets drift to higher resistance as wait time increases. The slope of the curve is the drift coefficient $\nu = 0.1$. (b) The drift coefficients for different initial resistances. Line represents the model described in eq. (1.2) and circle symbols represent experimental results. (c) NN architecture with 398 input neurons and 500 output neurons with fully connected structure. Each pixel of a MNIST image is corresponding to one of the input neurons.

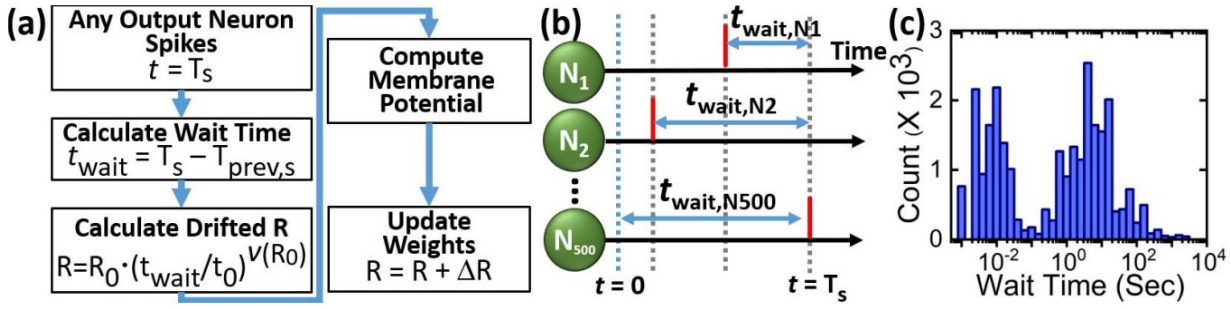


Figure 1.3: (a) Algorithm to compute wait time of each synapse. We estimate wait time first and then calculate resistance drift using eq. (1.1) and (1.2). (b) Visualization of wait time of each output neuron. When one of output neurons generates a spike, the neuron's corresponding weights are updated. After weight update, the updated devices remain idle until output spike is generated again.(c) A representative distribution of the estimated wait time of synapses connected to Neuron 1 (N_1) during the training.

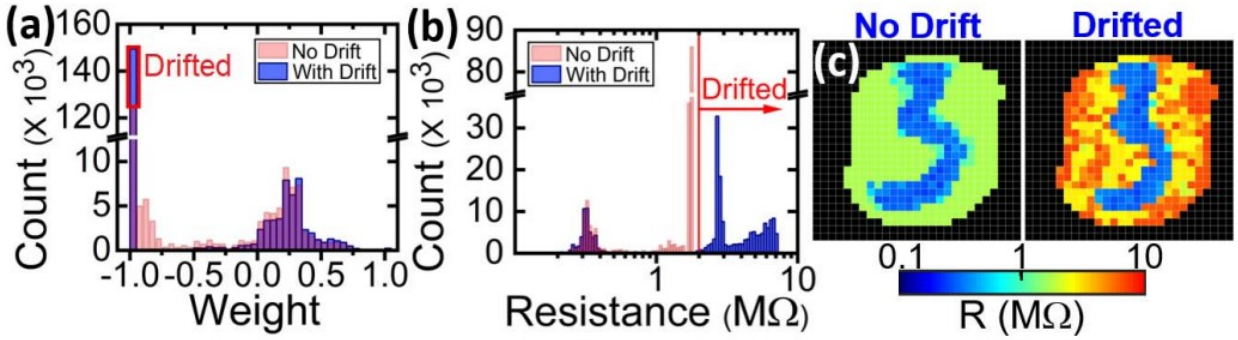


Figure 1.4: (a) Weight and (b) resistance distribution after training for no-drift and drifted cases. Weight distribution shows that the number of cells with –1 weight is increased due to drift. Resistance distribution shows high resistance synapses are drifted to MΩ range during the training. The red vertical line is a reference for eye to show the drift in high resistance ranges. (c) Resistance map of a representative output neuron for no-drift and drifted cases. With drift, the resistance values of BG pixels are increased.

(a)	No Drift	Drifted
Ideal (64-bit)	94.05 %	-
Ideal (8-bit)	92.02 %	-
Device (High G)	89.38 %	89.38 %
Device (Low G)	85.12 %	86.75 %

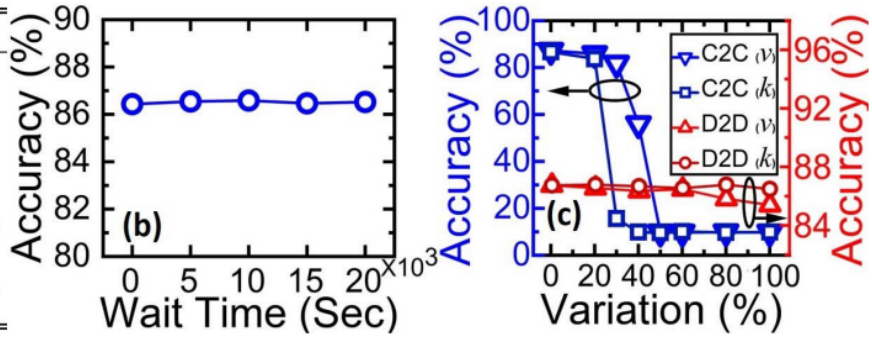


Figure 1.5: (a) MNIST classification accuracy of our SNN with ideal weights (64-bit and 8-bit precision) and device data (See Fig. 1.2(b) and (c)). (b) Classification accuracy for different additional wait time after training. (c) Resistance drift variation results. Our SNN exhibits great tolerance against D2D variations. The accuracy drops when C2C variation is greater than 30-40%.

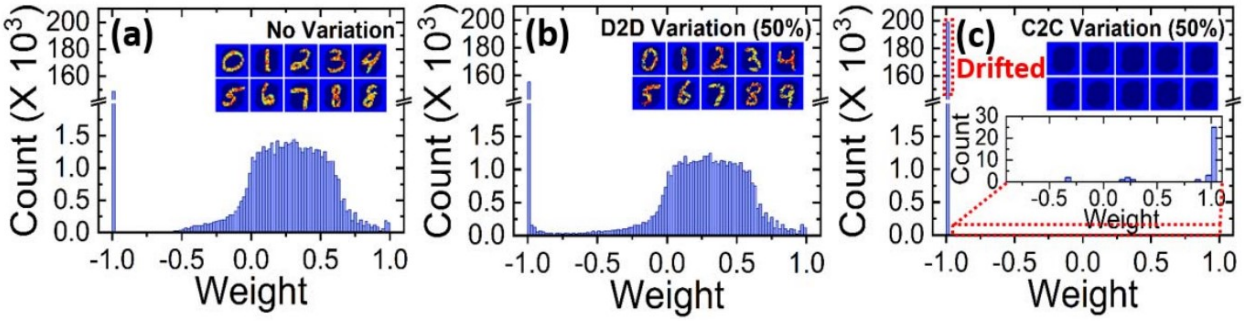


Figure 1.6: The weight distribution after training (a) without variation, (b) with 50 % of D2D variation, and (c) with 50 % of C2C variation. The inset images are the visualization of weights of 10 output neurons as representative examples. With 50 % of D2D variation, the weight distribution is well-maintained. However, 50 % of C2C variation drifts almost all of FG pixels to the minimum value (red box outline in (c)) and totally distorts weight distribution. The inset histogram in (c) shows few weights larger than the minimum.

7. References

1 Yu, S. Neuro-Inspired Computing With Emerging Nonvolatile Memorys. Proceedings of the IEEE 106, 260-285, doi:10.1109/jproc.2018.2790840 (2018).

2 Zidan, M. A., Strachan, J. P. & Lu, W. D. The future of electronics based on memristive systems. Nature Electronics 1, 22-29, doi:10.1038/s41928-017-0006-8 (2018).

3 Wong, H. S. P., Raoux, S., Kim, S., Liang, J., Reifenberg, J. P., Rajendran, B., Asheghi, M. & Goodson, K. E. Phase Change Memory. Proceedings of the IEEE 98, 2201-2227, doi:10.1109/JPROC.2010.2070050 (2010).

4 Kuzum, D., Jeyasingh, R. G., Lee, B. & Wong, H. S. Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing. Nano Lett 12, 2179-2186, doi:10.1021/nl201040y (2012).

5 Bichler, O., Suri, M., Querlioz, D., Vuillaume, D., DeSalvo, B. & Gamrat, C. Visual Pattern Extraction Using Energy-Efficient "2-PCM Synapse" Neuromorphic Architecture. IEEE Transactions on Electron Devices 59, 2206-2214, doi:10.1109/TED.2012.2197951 (2012).

6 Eryilmaz, S. B., Kuzum, D., Jeyasingh, R. G. D., Kim, S., BrightSky, M., Lam, C. & Wong, H. S. P. in 2013 IEEE International Electron Devices Meeting. 25.25.21-25.25.24.

7 Ielmini, D., Lavizzari, S., Sharma, D. & Lacaita, A. L. in 2007 IEEE International Electron Devices Meeting. 939-942.

8 Boniardi, M. & Ielmini, D. Physical origin of the resistance drift exponent in amorphous phase change materials. Applied Physics Letters 98, doi:10.1063/1.3599559 (2011).

9 Koelmans, W. W., Sebastian, A., Jonnalagadda, V. P., Krebs, D., Dellmann, L. & Eleftheriou, E. Projected phase-change memory devices. Nature Communications 6, 8181, doi:10.1038/ncomms9181 (2015).

10 Zhang, W. & Li, T. in 2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN). 197-208.

11 Kim, S., Lee, B., Asheghi, M., Hurkx, F., Reifenberg, J. P., Goodson, K. E. & Wong, H. S. P. Resistance and Threshold Switching Voltage Drift Behavior in Phase-Change Memory and Their Temperature Dependence at Microsecond Time Scales Studied Using a Micro-Thermal Stage. IEEE Transactions on Electron Devices 58, 584-592, doi:10.1109/ted.2010.2095502 (2011).

12 Braga, S., Cabrini, A. & Torelli, G. Dependence of resistance drift on the amorphous cap size in phase change memory arrays. Applied Physics Letters 94, doi:10.1063/1.3088859 (2009).

13 Nessler, B., Pfeiffer, M. & Maass, W. STDP enables spiking neurons to detect hidden causes of their inputs. Vol. 22 (Curran Associates, Inc., 2009).

14 Pirovano, A., Lacaita, A. L., Pellizzer, F., Kostylev, S. A., Benvenuti, A. & Bez, R. Low-field amorphous state resistance and threshold voltage drift in chalcogenide materials. IEEE Transactions on Electron Devices 51, 714-719, doi:10.1109/ted.2004.825805 (2004).

15 Chen, P.-Y., Peng, X. & Yu, S. NeuroSim: A Circuit-Level Macro Model for Benchmarking Neuro-Inspired Architectures in Online Learning. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 37, 3067-3080, doi:10.1109/tcad.2018.2789723 (2018).

16 Mallik, A., Garbin, D., Fantini, A., Rodopoulos, D., Degraeve, R., Stuijt, J., Das, A. K., Schaafsma, S., Debacker, P., Donadio, G., Hody, H., Goux, L., Kar, G. S., Furnemont, A., Mocuta, A. & Raghavan, P. in 2017 Symposium on VLSI Technology. T178-T179.

17 Yu, S., Li, Z., Chen, P. Y., Wu, H., Gao, B., Wang, D., Wu, W. & Qian, H. in 2016 IEEE International Electron Devices Meeting (IEDM). 16.12.11-16.12.14.

18 Liu, B., Li, H., Chen, Y., Li, X., Huang, T., Wu, Q. & Barnell, M. in 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). 63-70.

19 Burr, G. W., Shelby, R. M., Sidler, S., di Nolfo, C., Jang, J., Boybat, I., Shenoy, R. S., Narayanan, P., Virwani, K., Giacometti, E. U., Kurdi, B. N. & Hwang, H. Experimental Demonstration and Tolerancing of a Large-Scale Neural Network (165 000 Synapses) Using Phase-Change Memory as the Synaptic Weight Element. IEEE Transactions on Electron Devices 62, 3498-3507, doi:10.1109/ted.2015.2439635 (2015).

8. Acknowledgements

We acknowledge Office of Naval Research (N000141612531) and National Science Foundation (ECCS-1752241, ECCS-1734940) for funding. We acknowledge Prof. H.-S. Philip Wong for insightful discussions.

Chapter 1, in full, is a reprint of the material as it appears in IEEE Electron Device Letters. Oh, Sangheon; Shi, Yuhan; Liu, Xin; Song, Jungwoo; Kuzum, Duygu, Drift-Enhanced Unsupervised Learning of Handwritten Digits in Spiking Neural Network with PCM Synapses, 2018. The dissertation author was the primary author of this paper.

CHAPTER 2. THE IMPACT OF RESISTANCE DRIFT OF PHASE CHANGE MEMORY (PCM) SYNAPTIC DEVICES ON ARTIFICIAL NEURAL NETWORK PERFORMANCE

Phase change memory (PCM) has been considered as one of the most promising emerging non-volatile memories for in-memory computing of neural networks. In this chapter, we investigate the impact of resistance drift and its statistical variations on two widely-used artificial neural network (ANN) models, multi-layer perceptron (MLP), and convolutional neural network (CNN). We employ experimentally measured resistance drift characteristics into the ANN models to accurately model weight updates represented by PCM synaptic devices. Our results suggest that the resistance drift in PCM causes minor accuracy degradation (only ~1%) for both MLP and CNN models. However, classification accuracy can be significantly reduced if the PCM drift characteristics exhibit high device-to-device and cycle-to-cycle variations in the drift coefficients.

1. Introduction

Advancements in artificial neural networks (ANNs) have enabled huge performance improvements for a wide range of applications including image recognition[1,2] and speech processing[3,4]. However, the limitations due to Von Neumann architecture, known as Von Neumann bottleneck, have started to impede the development of more advanced models[5]. To circumvent this challenge, non-Von Neumann approaches such as in-memory computing systems based on emerging non-volatile memory (eNVM) devices have been proposed[6-12]. By performing data processing in memory, in-memory computing systems minimize the need for data transfer between the processor and the off-chip memory, and accelerate matrix

multiplication[13], which is a very energy and latency-intensive process[14] and the culprit of Von Neumann bottleneck. Among various eNVM devices, which are considered for in-memory computing systems, phase change memory (PCM) has been one of the promising candidates[15,16] due to its fast and gradual switching, energy efficiency, and scalability[17,18].

Although PCM devices have desirable switching characteristics for in-memory computing of ANNs, they still suffer from a non-ideal device behavior called resistance drift, which increases the resistance under amorphous (high resistivity) phase over time[19-21]. As a result, the data stored in PCM devices can change over time even though the devices remain idle. Many researchers have proposed ideas to mitigate the effects of the resistance drift for digital memory applications[22,23] and neuromorphic architectures[24]. In our previous work[25], we investigated the effect of resistance drift on the classification accuracy of unsupervised learning in spiking neural networks trained by biologically inspired spike timing dependent plasticity. However, most of the deep neural network models are supervised and trained using backpropagation. Understanding the impact of drift on the training of ANNs using backpropagation is critical for designing in-memory computing platforms based on PCM arrays. Therefore, in this chapter, we present a thorough study on how the resistance drift impacts training of commonly used ANN models. We model resistance drift and statistical variations in the drift parameters and investigate how the evolution of weights is impacted for two neural network models, multi-layer perceptron (MLP)[26] and convolutional neural network (CNN)[2]. Our simulation results demonstrate that MLP and CNN can tolerate resistance drift unless the drift parameters exhibit severe statistical variations.

2. Resistance Drift of PCM

To implement neural network weight updates using PCM device data, we used PCM devices with $Ge_2Sb_2Te_5$ (GST) phase change material[25]. The PCM devices have a 200 nm thick GST layer which is sandwiched by the TiN top and bottom electrodes. Gradual conductance changes are shown in **Fig. 2.1(a)**, which are achieved with a non-identical pulse scheme: Gradual set is achieved by stair-case pulses with an increasing amplitude (50 mV step) from 1 V to 1.7 V (4 pulses/step), and gradual reset is demonstrated by pulses with increasing amplitude from 5.7 V to 7.3 V with 25 mV voltage steps.

The resistance drift characteristics of the PCM device are shown in **Fig. 2.1(b)** and **(c)**. **Figure 2.1(b)** shows that the resistance of PCM at high-resistive amorphous state (~2 MΩ) gradually increases over time. The drift coefficients (v) for various resistance values shown in **Fig. 2.1 (c)** indicate that resistance drift becomes more severe as the resistance increases. The resistance drift behaviors can be characterized with two models: The first model[20], shown in (2.1), represents how much the resistance is increased for a given wait time of a PCM device (See **Fig. 2.1(b)**). In the model, drift coefficient (v) controls how fast the resistance is increased from the initial resistance (R_0), which is measured after initial read time (t_0) from a write operation, as wait time (t) increases.

$$R = R_0(t/t_0)^v \tag{2.1}$$

The second model[21], presented in (2.2), describes the dependency of drift coefficient (v) on the resistance of a PCM device (See **Fig. 2.1(c)**). The parameter k controls the dependency of drift coefficient on resistance. If k is larger, the drift coefficient increases faster as the resistance increases until it reaches its maximum value.

$$v(R) - v_0 = k \cdot \log(R/R_0) \tag{2.2}$$

In this chapter, we use $v_0 = 0.1$, $R_0 = 2$ MΩ, $t_0 = 1$ ms, and $k = 0.2$ which are extracted from the device data[25] shown in **Fig. 2.1**. Note that the maximum value of v is set to 0.1 as shown in **Fig. 2.1(c)**.

3. Neural Networks with PCM Devices

We constructed MLP and CNN for supervised online learning of MNIST handwritten digit dataset (28×28 grayscale images)[2] using PyTorch. The MLP model has 784 input neurons, 128 neurons in the hidden layer, and 10 output neurons as shown in **Fig. 2.2(a)**. Both the input layer and hidden layer have bias term (i.e., one additional neuron per each layer whose input is always 1). The other network, CNN (See **Fig. 2.2(b)**), has two 2D convolutional layers (5×5 kernel) and each of them is followed by a max polling layer (2×2 kernel). It also has one fully connected (FC) layer with 120 neurons, another FC layer with 84 neurons, and an output layer with 10 output neurons. We also added batch normalization layers[27] in-between FC layers to allow the use of larger learning rate and improve the accuracy of the network. For both MLP and CNN, the weights are quantized based on our PCM device data (~55 levels), and stochastic gradient descent optimizer (learning rate = 1 for MLP and 0.05 (with momentum = 0.9) for CNN) with cross-entropy loss and ReLU activation function are used for training.

In our previous work[25], we mapped network weights onto the device conductance using equation (2.3) and applied the two resistance drift models to the conductance values (i.e. (2.1) and (2.2)).

$$W = W_{max} + \frac{W_{max} - W_{min}}{G_{max} - G_{min}}(G - G_{max}) \qquad (2.3)$$

However, this method is not appropriate for directly using in neural network simulators, since all the calculations are done in the weight domain. Therefore, we converted the two resistance drift

models, which are in the resistance domain, into models in weight domain using the mapping equation (2.3) as follows:

$$W = W_i \left(\frac{t}{t_0}\right)^{-v} + \left(G_{max}\frac{W_{max}-W_{min}}{G_{max}-G_{min}} - W_{max}\right)\left\{\left(\frac{t}{t_0}\right)^{-v} - 1\right\} \qquad (2.4)$$

$$v(W_i) = v(W_0) + k \cdot \log\left(\frac{G_0/G_{max}}{\frac{W_i-W_{max}}{W_{max}-W_{min}}\left(1-\frac{G_{min}}{G_{max}}\right)+1}\right) \qquad (2.5)$$

where G_{max} (G_{min}) is the maximum(minimum) resistance that devices can achieve (our PCM devices exhibit G_{max} = 4.762 μS and G_{min} = 0.554 μS), W_{max} (W_{min}) is the maximum(minimum) weight of networks (we used W_{max} = 1 and W_{min} = -1), W_i is weight without drift, G_0 is initial conductance ($G_0 = 1/R_0$ = 0.5 μS), and W_0 is weight corresponding to G_0.

To implement resistance drift in network simulations, we need to develop a scheme to estimate the wait time of PCM devices during the network simulations. If we store weights and biases of networks using PCM devices, we will read the devices in feedforward step and write weights to the devices in the backpropagation step. The main contribution to the wait time comes from the calculation of weight updates. Therefore, we can assume that the delay between weighted sum read ($t_{RF,n}$) and individual weight read ($t_{RB,n}$) is negligible for the wait time estimation. Then, we can use the time interval between the current read time ($t_{RF,n} \approx t_{RB,n}$) and the initial drift time (t_{init}), the time when the drift is started, as the wait time ($t_{wait,n}$) of PCM device as shown in **Fig. 2.2(c).** Our wait time estimation scheme also addresses the resistance drift recovery phenomenon[28] which resets the resistance drift of devices when they are programmed (i.e. $t_{init} = t_{W,n}$ when $W_{n+1} \neq W_n$).

By using the resistance drift models with experimentally derived device parameters and the wait time estimation scheme, we performed network simulations for MLP and CNN (See **Fig. 2.3(a)**). For each network, we simulated the network with ideal software weights and PCM

device data with and without resistance drift. The results show that MLP can achieve 97.53 % accuracy for ideal (64-bit, floating point) software simulation. However, when weights are implemented with PCM device data, the accuracy is reduced by 5.93 % due to low bit precision as a result of quantization into the limited number of conductance levels (~55 levels) of PCM devices (shown as PCM w/o Drift) and it is further decreased by 1.18 % when the resistance drift is included (PCM w/ Drift). Meanwhile, CNN with ideal 64-bit precision achieves 98.96 % accuracy while the accuracy with PCM device data is decreased by 2.40 % due to quantization and 0.79% as a result of drift. The simulation results show that the accuracy degradation due to resistance drift is not significant compared to the accuracy drop because of low-precision of PCM devices (Note that the accuracy can be severely degraded if $\nu > 0.15$ (See **Fig. 2.3(b)**), however PCM with $\nu > 0.15$ will not reflect empirical results because most of PCM devices have $\nu \sim 0.1$ or less[29]). The slight accuracy degradation due to resistance drift indicates that the neural network models can successfully adapt to the resistance drift if the devices exhibit deterministic drift characteristics. Our results are consistent with prior works, which demonstrated that ANNs could adapt to the drift[30] and the drift can be mitigated by using proper synaptic array architectures[24]. It is noteworthy that the accuracy degradation due to quantization can be minimized with advanced quantization techniques[31-33].

4. Variations on Resistance Drift

While neural network models can adapt to deterministic weight changes due to drift, it is crucial to explore how variations in drift characteristics affect neural network performance. The variations in drift characteristics (e.g., device-to-device (D2D) and cycle-to-cycle (C2C) variation) have been experimentally investigated by many researchers[34-37] (e.g., J. L. M. Oosthoek et al.[34] reported ~16 % of C2C and M. Boniardi et al.[35] reported ~25% of D2D). We

simulated both D2D and C2C on both resistance drift parameters (v and k) for MLP and CNN as shown in **Fig. 2.4(a)**. The variations are characterized as Gaussian variation whose standard deviation is a given percentage of its mean. The results demonstrate that the classification accuracy of MLP with resistance drift is not degraded until D2D on v or k is larger than 20 %, C2C on v is larger than 15 %, or C2C on k is greater than 5 %. Similarly, CNN with resistance drift (See **Fig. 2.4 (b)**) shows high accuracy when D2D on v or k is less than 20 %, C2C on v is smaller than 10 %, or C2C on k is less than 5 %. We also show simulation results with experimental measurement results[35-37], in good agreement with our simulations. It is noteworthy that C2C variations are more detrimental for the accuracy of both MLP and CNN.

To explain the difference between C2C and D2D, we chose 20% of C2C and D2D for plotting the distribution of weights (Hidden-to-Output (HO) layer for MLP and FC2-Output layer for CNN as representative examples) in **Fig. 5**, where the distortion is severe with C2C while it is not severe yet with D2D. The main difference between C2C and D2D is that the cells exhibiting very high v are fixed in D2D while it keeps changing for C2C in every cycle. If the weights exhibiting high v are fixed, the distortion is minimal and the network can compensate it over the training. As a result, only a small number of weights are drifted and stuck at −1 over the training with D2D (Compare **Fig. 2.5 (a)** and **(b)** vs. **Fig. 2.5 (c)** and **(d)**). However, if the weights exhibiting high v are changed for every cycle, the variation can affect the whole array over the training and the network cannot compensate for the variation. Hence, much larger number of the weights are drifted and stuck at −1 over the training with C2C as shown in **Fig. 5 (e)** and **(f)**. The distortion of weights due to resistance drift (i.e. weights get stuck at −1) represents that the weight change during training is dominated by not the weight update (ΔW) but the resistance drift. It means that the weights were not changed in the direction of error

minimization of the network. Since the number of weights affected by the resistance drift with variation is larger with C2C than D2D, the accuracy degradation due to the C2C is more severe. As a result, accuracy degradation starts around 15% variation for C2C, while it starts around 20% for D2D (**Fig. 2.4**).

5. Conclusion

We demonstrated that resistance drift does not cause severe accuracy degradation for MLP and CNN. This implies that the ANNs can adapt to the weight changes due to resistance drift. We also explored resistance drift variation effects and found out that the networks cannot tolerate high statistical device-to-device and cycle-to-cycle variations in resistance drift characteristics. Our findings suggest that improving device uniformity and reducing cycle-to-cycle variations in drift characteristics are important to achieve high accuracy.

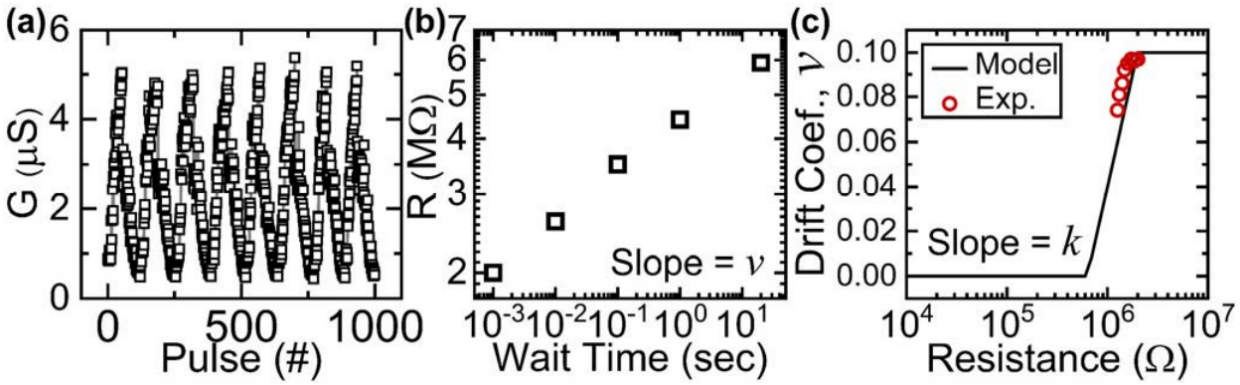


Figure 2.1: (a) Conductance response of our PCM device shows gradual switching behavior. (b) The resistance of PCM is measured after various wait time. It shows that resistance increases as wait time increases. The slope of the plot is the drift coefficient(ν). (c) The dependency of ν on resistance. The model shown in (2) (solid line) is fitted to measurement results (red circle symbols).

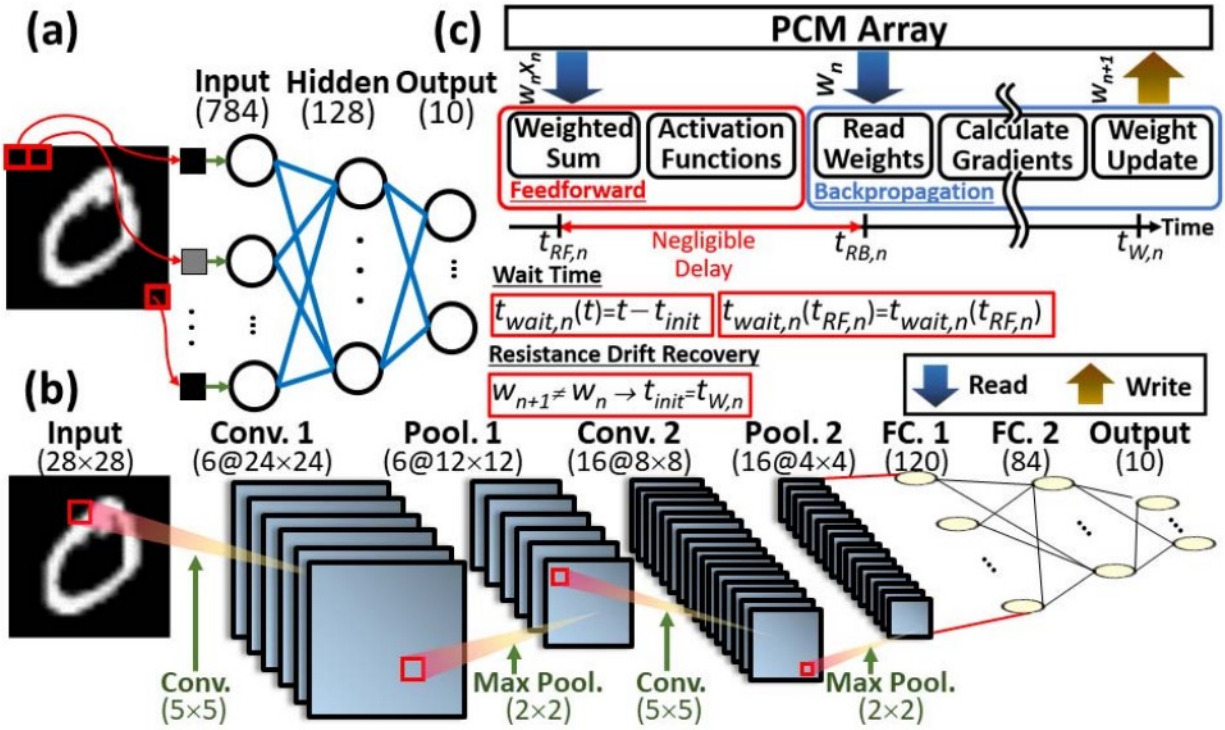


Figure 2.2: The schematics of two ANNs used for PCM drift simulations: (a) MLP and (b) CNN. (c) The illustration shows how the wait time of PCM cells is estimated during training. The wait time ($t_{wait,n}$) is estimated as a time interval between the initial drift time (t_{init}) and the current read time($t_{R,n+1}$) (Individual weights are read once at the beginning of the backpropagation step).

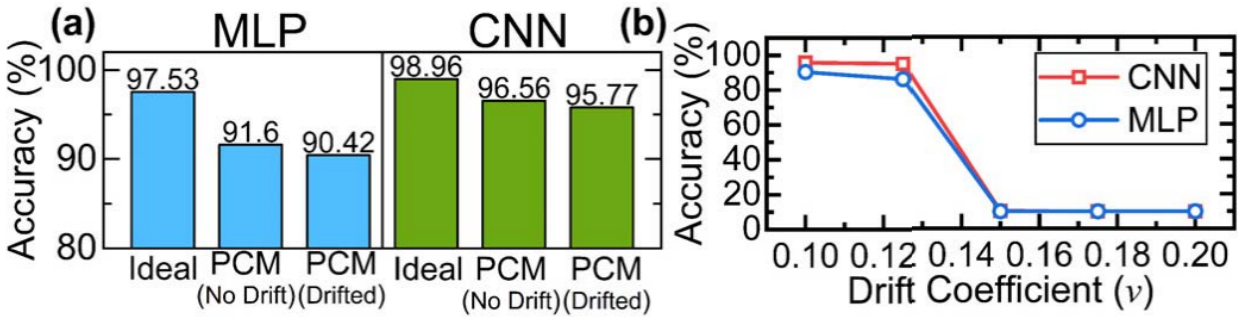


Figure 2.3: (a) The classification accuracy of MLP (left three bars) and CNN (right three bars) with ideal software weights and PCM device data without and with drift. (b) The accuracy of MLP and CNN with different drift coefficient ν at R_0.

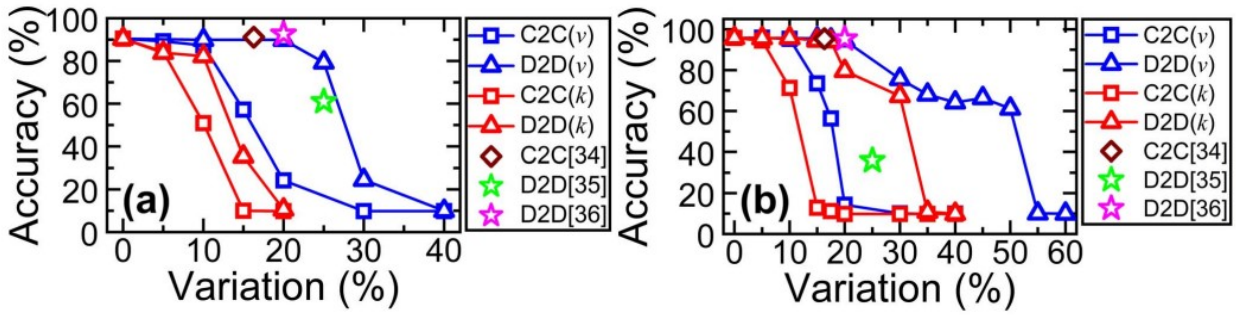


Figure 2.4: The classification accuracy of (a) MLP and (b) CNN with resistance drift variations (C2C and D2D) on drift coefficients (i.e., *v* and *k*). For both networks, the accuracy degradation due to C2C or variation on *k* is severer than that due to D2D or variation on v. Star symbols (D2D) and a diamond symbol (C2C) represent the results with the device data[34-36].

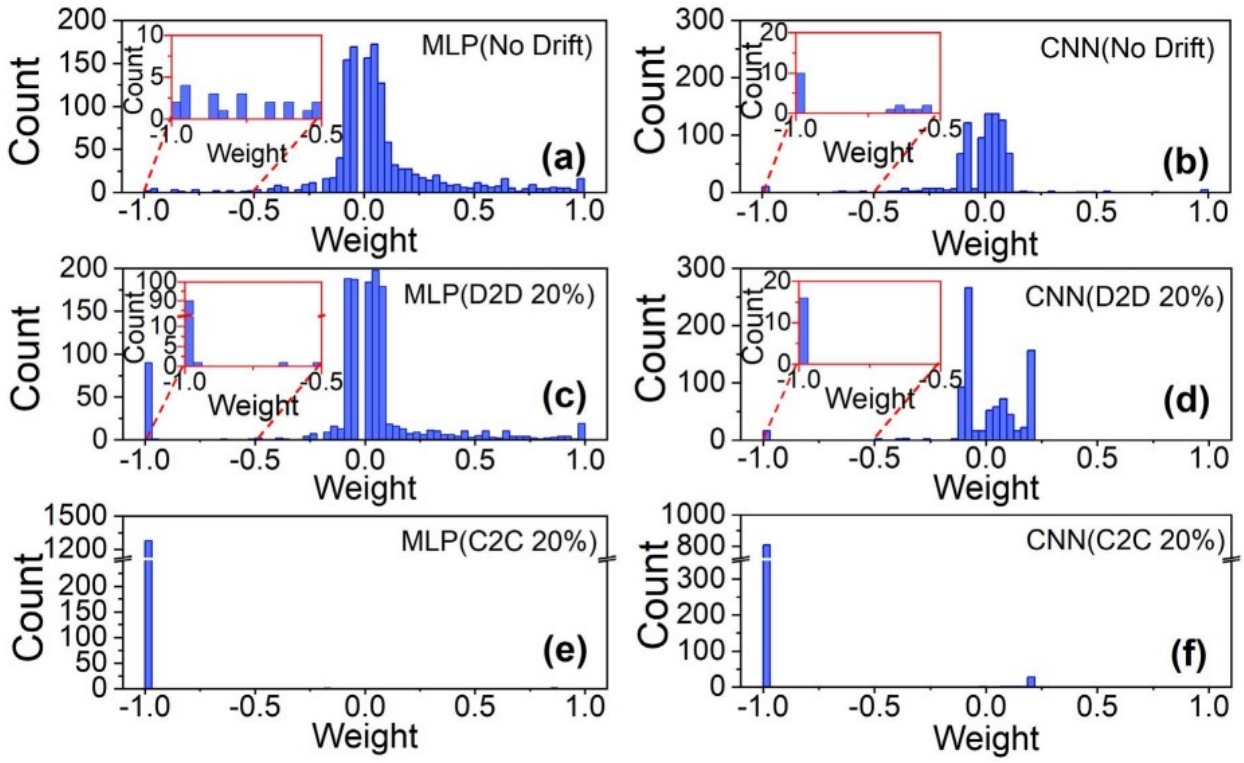


Figure 2.5: The weight distribution of (a) MLP (HO layer) and (b) CNN (FC2-Output layer) without resistance drift. With 20 % of D2D variation on v, most of the weights in −1 to −0.5 are drifted to −1 while the distribution of weights larger than −0.5 is well preserved for both (c) MLP and (d) CNN. However, with 20 % of C2C variation on v, almost all of weights are drifted to −1 for both of (e) MLP and (f) CNN.

6. References

1 He, K., Zhang, X., Ren, S. & Sun, J. in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 770-778 (2016).

2 LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278-2324 (1998).

3 Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N. & Kingsbury, B. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. IEEE Signal Processing Magazine 29, 82-97, doi:10.1109/MSP.2012.2205597 (2012).

4 Graves, A., Mohamed, A. r. & Hinton, G. in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. 6645-6649.

5 Backus, J. Can programming be liberated from the von Neumann style?: a functional style and its algebra of programs. (ACM, 2007).

6 Sun, J., Oh, S., Choi, Y., Seo, S., Oh, M. J., Lee, M., Lee, W. B., Yoo, P. J., Cho, J. H. & Park, J.-H. Optoelectronic Synapse Based on IGZO-Alkylated Graphene Oxide Hybrid Structure. Advanced Functional Materials 28, 1804397, doi:https://doi.org/10.1002/adfm.201804397 (2018).

7 Shukla, A. & Ganguly, U. An On-Chip Trainable and the Clock-Less Spiking Neural Network With 1R Memristive Synapses. IEEE Transactions on Biomedical Circuits and Systems 12, 884-893, doi:10.1109/TBCAS.2018.2831618 (2018).

8 Eryilmaz, S. B., Joshi, S., Neftci, E., Wan, W., Cauwenberghs, G. & Wong, H. S. P. in 2016 17th International Symposium on Quality Electronic Design (ISQED). 118-123.

9 Jo, S. H., Chang, T., Ebong, I., Bhadviya, B. B., Mazumder, P. & Lu, W. Nanoscale Memristor Device as Synapse in Neuromorphic Systems. Nano Letters 10, 1297-1301, doi:10.1021/nl904092h (2010).

10 Sun, X., Liu, R., Peng, X. & Yu, S. in 2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT). 1-4.

11 Shi, Y., Nguyen, L., Oh, S., Liu, X., Koushan, F., Jameson, J. R. & Kuzum, D. Neuroinspired unsupervised learning and pruning with subquantum CBRAM arrays. Nature Communications 9, 5312, doi:10.1038/s41467-018-07682-0 (2018).

12 Kuzum, D., Jeyasingh, R. G., Lee, B. & Wong, H. S. Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing. Nano Lett 12, 2179-2186, doi:10.1021/nl201040y (2012).

13 Shafiee, A., Nag, A., Muralimanohar, N., Balasubramonian, R., Strachan, J. P., Hu, M., Williams, R. S. & Srikumar, V. in 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA). 14-26.

14 Hameed, R., Qadeer, W., Wachs, M., Azizi, O., Solomatnikov, A., Lee, B. C., Richardson, S., Kozyrakis, C. & Horowitz, M. Understanding sources of inefficiency in general-purpose chips. SIGARCH Comput. Archit. News 38, 37–47, doi:10.1145/1816038.1815968 (2010).

15 Boybat, I., Le Gallo, M., Nandakumar, S. R., Moraitis, T., Parnell, T., Tuma, T., Rajendran, B., Leblebici, Y., Sebastian, A. & Eleftheriou, E. Neuromorphic computing with multi-memristive synapses. Nat Commun 9, 2514, doi:10.1038/s41467-018-04933-y (2018).

16 Ambrogio, S., Narayanan, P., Tsai, H., Shelby, R. M., Boybat, I., di Nolfo, C., Sidler, S., Giordano, M., Bodini, M., Farinha, N. C. P., Killeen, B., Cheng, C., Jaoudi, Y. & Burr, G. W. Equivalent-accuracy accelerated neural-network training using analogue memory. Nature 558, 60-67, doi:10.1038/s41586-018-0180-5 (2018).

17 Kuzum, D., Jeyasingh, R. G. D. & Wong, H. S. P. in 2011 International Electron Devices Meeting. 30.33.31-30.33.34.

18 Lee, B. C., Ipek, E., Mutlu, O. & Burger, D. Phase change memory architecture and the quest for scalability. Communications of the ACM 53, doi:10.1145/1785414.1785441 (2010).

19 Ielmini, D., Lavizzari, S., Sharma, D. & Lacaita, A. L. in 2007 IEEE International Electron Devices Meeting. 939-942.

20 Boniardi, M. & Ielmini, D. Physical origin of the resistance drift exponent in amorphous phase change materials. Applied Physics Letters 98, doi:10.1063/1.3599559 (2011).

21 Braga, S., Cabrini, A. & Torelli, G. Dependence of resistance drift on the amorphous cap size in phase change memory arrays. Applied Physics Letters 94, doi:10.1063/1.3088859 (2009).

22 Koelmans, W. W., Sebastian, A., Jonnalagadda, V. P., Krebs, D., Dellmann, L. & Eleftheriou, E. Projected phase-change memory devices. Nature Communications 6, 8181, doi:10.1038/ncomms9181 (2015).

23 Zhang, W. & Li, T. in 2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN). 197-208.

24 Suri, M., Garbin, D., Bichler, O., Querlioz, D., Vuillaume, D., Gamrat, C. & DeSalvo, B. in 2013 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH). 140-145.

25 Oh, S., Shi, Y., Liu, X., Song, J. & Kuzum, D. Drift-Enhanced Unsupervised Learning of Handwritten Digits in Spiking Neural Network With PCM Synapses. IEEE Electron Device Letters 39, 1768-1771, doi:10.1109/led.2018.2872434 (2018).

26 Jain, A. K., Jianchang, M. & Mohiuddin, K. M. Artificial neural networks: a tutorial. Computer 29, 31-44, doi:10.1109/2.485891 (1996).

27 Ioffe, S. & Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv e-prints (2015).

28 Pirovano, A., Lacaita, A. L., Pellizzer, F., Kostylev, S. A., Benvenuti, A. & Bez, R. Low-field amorphous state resistance and threshold voltage drift in chalcogenide materials. IEEE Transactions on Electron Devices 51, 714-719, doi:10.1109/ted.2004.825805 (2004).

29 Wimmer, M., Kaes, M., Dellen, C. & Salinga, M. Role of activation energy in resistance drift of amorphous phase change materials. Frontiers in Physics 2, doi:10.3389/fphy.2014.00075 (2014).

30 Chen, P. Y. & Yu, S. in 2018 IEEE International Reliability Physics Symposium (IRPS). 5C.4-1-5C.4-4.

31 Shi, Y., Huang, Z., Oh, S., Kaslan, N., Song, J. & Kuzum, D. Adaptive Quantization as a Device-Algorithm Co-Design Approach to Improve the Performance of In-Memory Unsupervised Learning With SNNs. IEEE Transactions on Electron Devices 66, 1722-1728, doi:10.1109/TED.2019.2898402 (2019).

32 Louizos, C., Reisser, M., Blankevoort, T., Gavves, E. & Welling, M. Relaxed Quantization for Discretized Neural Networks. arXiv e-prints (2018).

33 Kwon, D., Lim, S., Bae, J.-H., Lee, S.-T., Kim, H., Kim, C.-H., Park, B.-G. & Lee, J.-H. Adaptive Weight Quantization Method for Nonlinear Synaptic Devices. IEEE Transactions on Electron Devices 66, 395-401, doi:10.1109/ted.2018.2879821 (2019).

34 Oosthoek, J. L. M., Attenborough, K., Hurkx, G. A. M., Jedema, F. J., Gravesteijn, D. J. & Kooi, B. J. Evolution of cell resistance, threshold voltage and crystallization temperature during cycling of line-cell phase-change random access memory. Journal of Applied Physics 110, doi:10.1063/1.3603025 (2011).

35 Boniardi, M., Ielmini, D., Lavizzari, S., Lacaita, A. L., Redael, A. & Pirovano, A. Statistics of Resistance Drift Due to Structural Relaxation in Phase-Change Memory Arrays. IEEE Transactions on Electron Devices 57, 2690-2696, doi:10.1109/ted.2010.2058771 (2010).

36 Ielmini, D., Sharma, D., Lavizzari, S. & Lacaita, A. L. Reliability Impact of Chalcogenide-Structure Relaxation in Phase-Change Memory (PCM) Cells—Part I: Experimental Study. IEEE Transactions on Electron Devices 56, 1070-1077, doi:10.1109/ted.2009.2016397 (2009).

37 Betti Beneventi, G., Ferro, M., Calderoni, A. & Fantini, P. Physics-Based Statistical Modeling of PCM Current Drift Including Negative-Drift-Coefficients. IEEE Electron Device Letters 34, 879-881, doi:10.1109/led.2013.2261892 (2013).

7. Acknowledgements

This work was supported in part by the National Science Foundation under Grant ECCS-1752241 and Grant ECCS-1734940, in part by Samsung Electronics, and in part by Qualcomm FMA Fellowship.

Chapter 2, in full, is a reprint of the material as it appears in IEEE Electron Device Letters. Oh, Sangheon; Huang, Zhisheng; Shi, Yuhan; Kuzum, Duygu, The Impact of Resistance Drift of Phase Change Memory (PCM) Synaptic Devices on Artificial Neural Network Performance, 2019. The dissertation author was the primary researcher and author of this paper.

CHAPTER 3. NEUROINSPIRED UNSUPERVISED LEARNING AND PRUNING WITH SUBQUANTUM CBRAM ARRAYS

Resistive RAM crossbar arrays offer an attractive solution to minimize off-chip data transfer and parallelize on-chip computations for neural networks. Here, we report a hardware/ software co-design approach based on low energy subquantum conductive bridging RAM (CBRAM®) devices and a network pruning technique to reduce network level energy consumption. First, we demonstrate low energy subquantum CBRAM devices exhibiting gradual switching characteristics important for implementing weight updates in hardware during unsupervised learning. Then we develop a network pruning algorithm that can be employed during training, different from previous network pruning approaches applied for inference only. Using a 512 kbit subquantum CBRAM array, we experimentally demonstrate high recognition accuracy on the MNIST dataset for digital implementation of unsupervised learning. Our hardware/software co-design approach can pave the way towards resistive memory based neuro-inspired systems that can autonomously learn and process information in power-limited settings.

1. Introduction

Inspired by the biological neural networks giving rise to human intelligence, artificial neural networks[1] have revolutionized numerous computer vision[2,3] and speech recognition[4,5] tasks. Their near-human performance has been widely leveraged in various applications, including automated systems[6], aerospace and defense[7], health care[8], and home assistance devices[9]. However, training of neural networks requires substantial computing power and time due to the iterative updates of massive number of network parameters. For example, today's

advanced neural network algorithms require training times ranging from days to weeks and use carefully organized datasets consisting of millions of images to recognize objects such as animals or vehicles[10-12], while it only takes a few repetitions for a two-year-old toddler to identify these accurately and effortlessly[13]. Another example is AlphaGo, an advanced neural network trained for playing the board game Go against world champions, requiring 1920 CPUs and 280 GPUs and consuming hundreds of kilowatts per game[14]. The human brain, which can perform the exact same task, is 30,000 times more efficient, only consuming power on the order of 10W[13,15]. High energy consumption and extensive training time have been the major limitations for widespread adoption of neural networks at every scale – from mobile devices to data centers. The need for back and forth data transfer between the memory and processor in conventional computing systems based on von Neumann architecture is one of the major causes of high energy consumption during neural network computations. To address this major architectural drawback, on-chip memory storage and in-memory computing solutions using resistive switching memory arrays have been proposed to perform storage and computing at the same location. Non-volatile memory-based synaptic devices such as phase change synapses (PCM)[16,17], Ag-based conductive bridging synapses (CBRAM)[18], and resistive RAM synapses (RRAM)[19-21] have been investigated for implementing synaptic weight updates during neural network operation. The synaptic arrays using memristors have also been widely used in energy efficient implementation of unsupervised learning[22-25] and MNIST classification[26-34] in the past.

On a separate front, the pruning algorithm[35,36] inspired from neuroscience[37] has been suggested towards reducing network level energy consumption and time by settings the low valued weights to zero. However, these methods were mostly applied on the trained networks[35,36]. Pruning during training by backpropagation was previously employed in literature to prevent

overfittings[38,39]. Yet, there is no systematic study showing how pruning can address the energy consumption and excessive training time problems during the training in hardware.

In order to overcome the energy consumption challenge, incremental improvements in devices or algorithms alone will not be sufficient. Therefore, in this work, we focus on a hardware/software co-design approach that combines the advances in low-power device technologies with algorithmic methods to reduce the energy consumption during neural network training. First, we experimentally investigate and characterize the gradual conductance change characteristics of subquantum CBRAM devices, targeting implementation of neural network training in hardware. We show that the subquantum CBRAM devices can achieve gradual switching using stepwise programming and they can be directly programmed into any arbitrary level by controlling wordline (WL) voltage. Then we develop a spiking neural network (SNN) model for unsupervised learning and evaluate its performance by simulations for both analog and digital hardware implementations. In order to improve network level efficiency, we introduce a pruning algorithm carried out during the training and investigate its limits and performance through software simulations. Different from previous algorithmic approaches employing pruning on already trained networks[35,36], our neuro-inspired pruning method is applied during the network training to minimize the energy consumption and training time. Combining the energy-efficient subquantum CBRAM devices and the pruning technique, we experimentally demonstrate highly energy efficient unsupervised learning using a large-scale (512kbit) subquantum CBRAM array. The hardware/software codesign approach presented in this work can open up new avenues for applications of unsupervised learning on low-power and memory-limited hardware platforms.

2. Subquantum synaptic device characteristics.

In this section, we investigate device characteristics of subquantum CBRAM relevant to the general context of neural network operation. We explore gradual switching capability of subquantum CBRAM for implementation of different biological or non-biological weight update rules. For CBRAM devices, the 1-atom conductance (G_{1atom}), which corresponds to the conductance (G) of a filament just one atom "wide" at its thinnest point, is a critical parameter affecting energy consumption and filament stability (retention)[40]. G_{1atom} is on the order of the fundamental conductance $G_0 = 2e^2/h \approx 80\mu S$ for CBRAM cells based on filament metals such as Ag and Cu, so typical programming voltages of about 1-3V yield a minimum programming current (i.e., to form a filament just 1 atom "wide") of $I_{prog} \approx G_0$ (1V – 3V) = 80-240μA, resulting in high energy consumption in the range from about 1pJ to 100pJ for commonly used programming pulse durations (10ns to 100ns) (**Table 3.S1**). Subquantum CBRAM cells reduce programming energy and improve filament stability (**Fig. 3.1a**) by utilizing filaments comprising a semiconductor or semimetal (at least at their thinnest spot, which dominates the resistance)[40]. A subquantum CBRAM memory cell utilizing tellurium (Te), an elemental semiconductor with a band gap of 0.3eV[41], which has a 1-atom conductance deduced[40] to be $G_{1atom} = 0.03G_0$, is shown in **Fig. 1b**. With a much lower G_{1atom} than Ag or Cu and with write/erase speeds as low as about 10ns (**Figure 3.S1**), such subquantum CBRAM cells can consume as little as about 0.2pJ ($I_{prog} \approx 0.03G_0 \times$ (1V – 3V) ≈ 2.4-7μA and $E = I_{prog} \times V_{prog} \times pulse\ duration$ = 7μA × 3V × 10ns = 0.2pJ) when programmed to their 1-atom limit. This is an order of magnitude lower than for metal filament-based devices programmed to their corresponding 1-atom limit **Table 3.S1**). The retention of the subquantum CBRAM device is shown in **Figure 3.S2** and is discussed in **Supplementary Note 1**.

Figure 3.1b shows a cross-section TEM of a subquantum CBRAM cell, fabricated using Ta as the cathode material, sputtered amorphous Al_2O_3 as the insulating layer, and sputtered amorphous ZrTe as the anode material. The array **(Fig. 3.1b)** containing the subquantum CBRAM device has one-transistor one-resistor (1T1R) structure, which provides access to individual cells. I-V characteristics of subquantum CBRAM cells measured by a typical double DC sweep exhibit bipolar characteristics **(Fig. 3.1c)**. In the positive regime, a voltage bias is applied to the anode and swept from 0V to +3V with step size 5mV. The resistance of the cell was switched from a high resistance state to a low resistance ON-state. This process is suggested[40] as inducing an electrochemical replacement reaction wherein Te is liberated from the anode by O from the oxide layer. In the negative regime, reversing the polarity of the voltage will break the filament and switch the cell back to a high resistance OFF-state. The resistance can be read without disturbing the state of the cell by applying a small voltage (~100mV) of either polarity. These two distinct states are utilized in memory applications to store binary information. On the other hand, a gradual, analog-like conductance change has been suggested as a requirement for implementation of synaptic plasticity and learning[42]. Gradually increasing and decreasing device conductance is equivalent to long-term potentiation (LTP) and long-term depression (LTD) of synapses in the brain, which are two major forms of synaptic plasticity. LTP and LTD allow for fine synaptic weight updates during network training. Subquantum CBRAM cells can potentially provide more gradual changes in conductance than metal filament-based cells since during programming G tends to increase in increments of $\sim G_{1atom}$, which for Te is an order of magnitude smaller than for metals.

We investigate general gradual programming characteristics of subquantum CBRAM cells using two different methods. Controlling WL voltage allows to change programming

current values to program the CBRAM devices to different conductance levels, as this property of resistive memories has been studied before. **Fig. 3.2a** shows gradual switching of a subquantum CBRAM cell by application of stepwise voltage pulses applied to the WL with an increasing step of 10mV for conductance increase and 4mV for conductance decrease over many cycles. Subquantum CBRAM cells can provide linear weight tuning for both LTP and LTD (**Fig. 2a,** as shown by linear trend lines). The linearity of the weight tuning was previously reported to be important for implementation of various operations and achieving high accuracy in artificial neural network implementations with resistive memory devices[43,44]. Stepwise gradual programming of subquantum CBRAM synapses (**Fig. 3.2a**) can be used to implement various forms of learning and plasticity. As representative examples, **Supplementary Figure 3** shows two different forms of biological spike-timing-dependent plasticity (STDP)[16,42,45] implemented with subquantum CBRAM synapses. Symmetric plasticity **Supplementary Figure 3a** can be employed for associative learning and recall[16], and asymmetric plasticity (**Figure 3.S3b**) can be used to transform temporal information into spatial information for sequence learning[16]. The STDP implementation is discussed in **Supplementary Note 2**.

Alternative to stepwise programming, the subquantum CBRAM cells can also be directly programmed into an arbitrary conductance state by controlling the WL voltage without being bound to a particular sequence of states. **Fig. 3.2b** shows a sequence of programming operations in which the WL voltage increases with step size 20mV followed each time by an erase operation. This offers flexibility for implementing weight update rules of greater complexity. **Figure 3.4** shows that the nonlinear weight update rule we used can be greatly represented by the device conductance change using this WL voltage modulation.

In order to implement neural network training with 1T1R resistive memory arrays,

synaptic weights can be represented in either binary (digital) or analog manners[46]. For digital implementation, N binary 1T1R cells are grouped to represent one synaptic weight (**Fig. 3.2c**) and each cell is programmed to high or low conductance states, providing N-bit weight precision in a binary format. For analog implementation, the cells can be arranged into a pseudo-crossbar array and synaptic weights are stored in the form of multi-level conductances (**Fig. 3.2d**)[46]. As shown in the measurement results presented in this section, the subquantum CBRAM devices are capable of both digital and analog implementations. The tradeoff between analog and digital implementations in terms of energy consumption, latency and area will be further discussed in the context of our neural network model in the **Methods** section.

3. Neural network algorithm for unsupervised learning.

Here, we investigate neuro-inspired spiking neural network (SNN) configurations and implement unsupervised learning on 1T1R CBRAM synaptic arrays to classify MNIST handwritten digits, which consists of 60,000 training samples and 10,000 test samples. Different from other neural networks trained using back propagation, neuro-inspired SNNs use event-based and data-driven updates to reduce redundant information processing to gain efficiency and minimize energy consumption, making them ideal for hardware implementations[47-49]. Neuromorphic hardware platforms based on SNNs have already been demonstrated and employed in various applications of neural networks[48-50]. To reduce the network size, we crop some black background pixels from the full image of 784 (28 × 28) pixels. Therefore, our network contains 397 input neurons with a bias term and 500 output neurons, resulting in 199,000 synaptic weights (**Fig. 3.3a**). SNNs encode information between input and output neurons using spike trains. The firing frequency of the Poisson spike trains generated by the

input neurons scales linearly with respect to the pixel intensity (0 Hz for intensity value of 0 and 200 Hz for intensity value of 1). The output neurons integrate all the inputs to generate output spike trains based on a probabilistic winner-take-all (WTA) mechanism (See **Methods** section for more details)[51,52]. The synaptic weights of the firing output neuron are updated by a simplified STDP rule shown in **Fig. 3.3b** during training. STDP rule that modulates weights based on the timing of input and output spikes: if the time difference between the post-spike and pre-spike is less than 10ms, the synaptic weight is updated via the LTP rule, otherwise, it is updated via the LTD rule. Here, the LTD update is a constant weight decrease and the LTP update depends on the current weight state of the synapse with an exponentially decaying function shown in **Fig. 3.3c**. Exponential LTP updates will guarantee that the weights converge to the upper bound of 1. For LTD updates, the lower bound of the weight is clipped to -1. Overall, these rules result in weight values that are in the range of -1 to 1, allowing for a feasible and practical hardware implementation. During the training, the weights are adjusted incrementally based on the STDP rule so that output neurons fire selectively for a certain class in the dataset. Before training, output neurons exhibit random spiking response to the presented digits (**Fig. 3.3a**). However, after training, output neurons fire selectively during the presentation of specific samples learned during the training (**Fig. 3.3a**). **Figure 3.3d** and **e** show MNIST digit classification accuracy as a function of training epoch and neuron number. Training more than 3 epochs (**Fig. 3.3d**) or increasing the output neuron number beyond 500 (**Fig. 3.3e**) do not result in noticeable increase in accuracy, similar to what has been reported for single layer spiking neural networks in literature[53]. Therefore, we choose to use 500 neurons and 3 epochs for the training in our analysis. The algorithm we used for unsupervised learning is summarized in **Figure 3.S5**. After training is complete, the training dataset is presented again to assign neuron

labels to the output neurons by determining which digits provoked the highest average firing rate for each of the output neurons[53]. We predict the labels from the test set, which consists of 10,000 new samples from the MNIST test set, based on the same framework used during training to find the output neuron with the highest average firing rate for each sample (See **Methods** section for more details). We simulate our network for the ideal software (64-bit), and our proposed digital (**Fig. 3.2c**) and analog implementations (**Fig. 3.2d**). **Table 3.1** summarizes classification accuracy for all three cases. For the ideal software implementation, it is important to point out that ~94% accuracy is already very high for unsupervised learning with SNN[53]. Increasing the accuracy further to the levels of deep neural networks will definitely require introducing supervision to the SNN[54-56]. For digital implementation, we use 8-bit digital synapses and the weights are quantized to 256 levels distributed evenly between [-1, 1-2/256]. For analog implementation, we directly use conductance values (**Fig. 3.2a**) from device characteristic in our simulation to perform weight update during training. Neural network weights in the range of [-1, 1] can be mapped to device conductance using a linear transformation, as explained in the **Methods** section. Our results suggest that 8-bit digital implementation achieves comparable recognition accuracies with ideal software case and analog implementation has slightly lower accuracy due to the limited conductance states exhibited by each CBRAM synapse.

In order to compare the digital (**Fig. 3.2c**) and analog synaptic core (**Fig. 3.2d**), we develop a SNN platform for NeuroSim[46] (SNN+NeuroSim). NeuroSim is a C++ based simulator with hierarchical organization starting from experimental device data and extending to array architectures with peripheral circuit modules and algorithm-level neural network models[46]. We use SNN+NeuroSim to perform circuit-level simulations (**Table 3.2**) to estimate the energy, latency and area for the digital and analog implementations using the experimental data

measured from subquantum CBRAM devices (**Fig. 3.2**). The left two columns of **Table 3.2** show benchmarking results for analog synaptic core and 6-bit digital synaptic core. 6-bit precision is chosen to match the number of levels that can be achieved by gradual programing of subquantum CBRAM devices for the analog implementation. However, in order to achieve a recognition accuracy above 90%, 8-bit precision is required. Therefore, we include the third column, showing the results for 8-bit digital case, which is also used in the hardware demonstration (**Methods** section). The best performing metrics are highlighted in yellow. As shown in the table, the 6-bit digital scheme has better accuracy, shorter latency and lower energy consumption. On the other hand, the analog scheme occupies smaller chip area. Therefore, the benchmarking results suggest that digital implementation could be more advantageous in terms of energy consumption and latency for hardware implementation of on-line learning using subquantum CBRAM array.

4. Pruning during the training.

Neural network pruning algorithms have been very effective to reduce the time and energy consumption during inference by removing unimportant weights. Conventional pruning methods[35,36], which we also refer to as pruning in this work, set the low valued weights to zero. However, these methods are not suitable to be directly applied to the network learning algorithms that can produce non-zero centered weight distributions. In such situations, zero-valued weights are also important so that arbitrarily setting pruned weights to zero may affect accuracy. Additionally, conventional pruning mostly targets the networks which have already been trained. Therefore, the issues of excessive time and energy consumption during training remain unaddressed. To address both of these, we develop a method as an extension of pruning,

which we refer to as soft-pruning[57]. Instead of completely removing the weights from network by setting them to zero, soft-pruning sets the values of pruned weights to a constant non-zero value and prevents them from being updated during the rest of the training while allowing them to still participate in the inference step after the training. Therefore, pruning weights during training helps to significantly reduce the number of weight updates, minimizing computation and energy consumption. To decide when to prune weights during the training, we determine if the output neurons are trained enough to recognize a class from the dataset. We quantify this by counting the occurrences of consecutive output spikes (**Figure 3.S6**) from a single output neuron. The corresponding time interval between consecutive output spikes follows a Poisson distribution. Once an output neuron sees p occurrences of consecutive spikes during the training, a certain percentage of its weights are pruned to their lowest possible value (in our case, W_{min} = -1). The pruning algorithm is summarized in **Figure 3.S7.** Potential hardware implementations of this pruning algorithm are discussed in **Supplementary Note 3** and associated overheads estimation in area, energy and latency via simulation (SNN+NeuroSim) are shown in **Figure 3.S8** and **Table 3.2**. We investigate the distribution of weights in the SNN before and after soft-pruning along with a baseline control case, where pruning is not employed (no pruning) (**Fig. 3.4a**). Simulation of recognition accuracy for different p values in **Fig. 3.4b** suggests that p = 10 provides the highest accuracy even for very large pruning percentages (up to 80%). Visualization of weights from ten representative output neurons (bottom row of **Fig. 3.4a**) shows that foreground pixels (the digits) correspond to higher weight values on the distributions, and background pixels (background of the digits) correspond to lower weight values for no pruning case (weights visualization for all output neurons can be found in **Figure 3.s9**). Before pruning, the distributions indicate that the weight updates have been the same for both cases. **Fig. 3.4c**

compares recognition accuracy for as a function of pruning percentage for soft-pruning and pruning during the training, in comparison to pruning at the end of training for both cases. The recognition accuracy for pruning falls below ~90% for ~40% pruning percentage. In contrast, soft-pruning maintains high classification accuracy (~90%) even up to ~75% pruning percentage (**Fig. 3.4c**) The accuracy improvement achieved by the soft-pruning algorithm can be understood from the following two perspectives. First, since the pruned weights are set to -1 instead of being completely removed from the network, they still participate in the inference. Pruning the unimportant weight to -1 effectively decreases the membrane potential of output neurons, which helps to prevent false positive spikes. Second, the soft-pruning algorithm preserves the original weight distribution. As shown in **Fig. 3.4a**, the final distribution of learned weights clearly consists of two distinct parts which correspond to the foreground and background pixels of the image. The weights concentrated at -1 are associated with the background pixels, while the remaining weights centered around zero accounts for the foreground pixels. Soft-pruning sets pruned weights to -1, grouping them with the background pixels. On the contrary, pruning sets pruned weights to 0, which is in the range of weights that are associated with foreground pixels; this significantly changes the shape of foreground weight distributions, which leads to the accuracy degradation. Our soft-pruning method achieves high recognition accuracy for extensively pruned networks, offering superior energy efficiency during training for hardware implementations of unsupervised learning.

5. Hardware demonstration of pruning during training.

In order to implement unsupervised learning and pruning during the training on the hardware, we used a 512kbit subquantum CBRAM chip fabricated in a 130nm Cu back end of

line (BEOL) process (**Fig. 3.1b**). The array has a 1T1R architecture, which provides access to individual cells. Although each individual cell in our array has gradual conductance switching capabilities as demonstrated in **Fig. 3.2 a** and **b**, the digital implementation offers smaller energy consumption and shorter latency which is important for online learning as shown in **Table 3.2**. Furthermore, analog approach with varying amplitude pulses requires peripheral neuron circuits to produce non-identical pulses with fine grained duration[58,59]. Therefore, we choose to use digital implementation for hardware demonstration. We uniformly quantize the weights and map them onto the CBRAM array using an 8-bit digital representation between W_{min} = -1 and W_{max} = 1 (Details are explained in the **Methods** section), as our simulations have shown high recognition accuracy for 8-bit representation. Each weight is approximated to its closest quantized level when updating. Using our proposed network size to implement 10-digits MNIST classification requires at least 199,000 × 8 = 1.5 Mbit array. Given our array size limitation of 512kbit, we reduce the network size to 395 input and 10 output neurons to classify three classes ("0", "3", and "4") from MNIST. **Fig. 3.5a** shows recognition accuracy as a function of bit precision in the range of 5 to 12 bits, corresponding to quantization to 2^5 and 2^{12} discrete levels. The recognition accuracy stays relatively constant down to 8 bits but shows a steep decrease for bit precisions less than 7 bits. For hardware implementation of online unsupervised learning, the weights are updated on the subquantum CBRAM array at run-time. **Fig. 3.5b** shows experimentally obtained weight maps from the subquantum CBRAM array for the 10 output neurons for the no pruning and 50% soft-pruning cases after unsupervised online training with 1,000 MNIST samples. Weight update history during the online training process is investigated. **Figures 3.S10a** and **b** show the number of switching cycles of every bit in CBRAM cells for no pruning and 50% soft-pruning, respectively. Least significant bits (LSB) update more frequently

than the most significant bits (MSB) in both cases. For the no pruning case, all bits are constantly updated throughout training, causing extensive energy consumption through programming and erasing of the subquantum CBRAM devices. In contrast, pruning reduces the number of switching cycles for all of the individual bits and the number of cumulative switching cycles as shown in **Figure 3.S10b** and **Figure 3.S10c**, respectively. **Fig. 3.5c** shows the accuracy for the pruning and no pruning cases for the experimental results obtained with the subquantum CBRAM array as a function of training set size. This hardware implementation achieves 93.19% accuracy, which is very close to the accuracy for no pruning (93.68%) and the 8-bit and 64-bit ideal software implementations. **Fig. 3.5d** shows the number of bit updates by device updates vs. training set size, where the data for the first 1,000 samples are obtained from the hardware implementation, and the rest is computed using software simulations. The number of bit updates for both cases is identical until pruning starts. After all output neurons are pruned, the 50% pruned network has around twofold reduction in the number of bit updates compared to the no pruning case. Although our hardware demonstration focuses on 50% pruning, our simulations suggest that pruning percentages up to 80% can be implemented to further increase energy savings.

6. Discussion

The performance of our hardware implementation for unsupervised learning is far superior to the previous state-of-the-art unsupervised learning of MNIST dataset with synaptic devices in terms of recognition accuracy, energy consumption per programming, number of weight updates in training, and network size (**Table 3.S3**). For energy consumption per programming event, subquantum CBRAM is two to three orders of magnitude more efficient

than transistor-based devices (**Table 3.S1**) and shows the lowest energy consumption among RRAM based synaptic devices (**Table 3.S3**). Our pruning algorithm can reduce the number of parameter updates significantly and lead to ~20× less number of parameter updates compared to previous reports (**Table 3.S3**). Combining device level energy savings provided by subquantum CBRAM with network level energy savings by pruning may lead up to two orders of magnitude reduction in total energy consumption for hardware implementation of weight updates during unsupervised learning.

Compared to other software simulations in the literature (**Table 3.S4**), our network achieves a high classification accuracy on MNIST dataset using the lowest number of neurons and synapses and a low-complexity one-layer architecture that can be easily mapped onto 1T1R or crossbar arrays. **Table 3.S5** compares hardware demonstration of our pruning method with other software approaches of pruning in terms of energy savings and accuracy loss. Our method provides comparable energy savings with minimal accuracy loss, while being the only method, which can be applied during the training. Last but not least, our work presents the demonstration of mapping of pruning onto a hardware platform.

We demonstrate unsupervised learning using an energy efficient subquantum CBRAM array. Synaptic pruning is implemented during the training and mapped onto hardware to reduce energy consumption while maintaining a classification accuracy close to ideal software simulations. We show that subquantum CBRAM cells are capable of gradual and linear conductance changes desirable for implementing online training in hardware and can be directly programmable into different conductance states indicating their potential for implementing a broad range of weight update rules for neuromorphic applications. Following a software/hardware co-design approach, we develop a neuro-inspired synaptic pruning method to

significantly reduce the number of parameter updates during neural network training. Low-energy subquantum CBRAM devices combined with the network-level energy savings achieved by pruning can provide a promising path towards realizing AI hardware based on spiking neural networks that can autonomously learn and handle large volumes of data. Our hardware/software co-design approach can also be adapted to other network models to reduce the energy cost in implementing network training in low-power mobile applications.

7. Methods

A. Neural network algorithm.

Here we describe the network architecture of the SNN including the input and output layers. Then, we explain our training, labeling, and classification procedure for the MNIST dataset. **Table 3.S6** summarizes the parameters used in simulations.

(A) Network architecture: Our SNN is a one-layer network defined by the number of inputs neurons m, the number of outputs neurons n, and an m by n weight matrix. Each output neuron is fully-connected to every input neuron. Our SNN has 398 input and 500 output neurons. Our output neurons do not have refractory periods and there is no lateral inhibition between them.

(B) Input layer: We crop each training sample by removing pixels that represent the background in at least 95% of the training samples. Because the pixels have intensity values in the range [0, 1], those with a value of 0 correspond to the background and are thus candidates for removal. After this step, we have 397 input neurons in total by including an additional bias term, which has an input value of 1. The weights associated with this bias input neuron are learned via the same learning rule as the other weights. Each input neuron generates a Poisson spike train X_i

whose mean firing rate is determined linearly by the pixel intensity, where a pixel of value 0 corresponds to 0 Hz and a pixel of value 1 leads to 200 Hz. The timing of each spike that is generated by the Poisson process is rounded toward the nearest millisecond, which is the time step of the simulation.

(C) Output layer: The SNN fires an output spike from any given output neuron according to a Poisson process with the specified frequency. The output neuron that fires is chosen from a softmax distribution of the output neurons' membrane potentials as (3.1):

$$P(u_k) = \frac{e^{u_k}}{\sum_{k=1}^{N} e^{u_k}} \tag{3.1}$$

, where $P(u_k)$ is the softmax probability distribution of the membrane potentials u_k (k = 1, …, N). N is the number of output neurons. We calculate membrane potentials u_k using (3.2)

$$u_k = \sum_i W_{ki} X_i + b_k \tag{3.2}$$

W_{ki} is the weight between input neuron i and output neuron k. X_i is the spike train generated by input neuron i and b_k is the weight of the bias term.

(D) Training: The SNN displays each input sample for the first 40 ms of a 50 ms presentation period, and thus the input spikes for a given sample only occurs in this 40 ms window. **Fig. 3a** shows an example of the input spiking activity for the duration of four training samples. We use the whole training set, which contains 60,000 samples, and train for three epochs. It is important to note that 50 ms is a virtual simulation parameter along with the firing frequency chosen for generating input spikes. In the real hardware implementation, the presentation time of one image can be much shorter than 50 ms as long as enough number of input spikes are generated. The weights are updated via STDP rule shown in **Fig. 3.3b**. The LTP and LTD rules are detailed in (3.3) and (3.4) respectively,

$$\Delta W_{LTP} = a \times e^{-b\,(W+1)} \tag{3.3}$$

, where *a* and *b* are parameters that control the scale of the exponential, and W is the current weight value. The result ΔW is the amount of weight update of LTP and it is dependent on current W. LTD is a constant depression in terms of c in (3.4),

$$\Delta W_{LTD} = -c \tag{3.4}$$

(E) Labeling: After training is done, we fix the trained weights and assign a class to each neuron by the following steps: First, we present the whole training set to the SNN and record the cumulative number of output spikes N_{ij}, where i = 1, ... , N (N is number of output neurons) and j = 1, … ,M (M is number of classes). Then, for each output neuron i, we calculate its response probability Z_{ij} to each class j using (3.5). Finally, each neuron i is assigned to the class that gives the highest response probability Z_{ij}.

$$Z_{ij} = \frac{N_{ij}}{\sum_{j=1}^{M} N_{ij}} \tag{3.5}$$

(F) Classification: We use the standard test set which contains 10,000 images. We use equation (3.6) to predict the class of each sample, where S_{jk} is the number of spikes for the kth output neuron that are labeled as class j and N_j is the number of output neurons labeled as class j[53].

$$J = \underset{j}{argmax} \frac{\sum_{k=1}^{N_j} S_{jk}}{N_j} \tag{3.6}$$

(G) Weight mapping for analog synapse implementation: The network weights (W) ranging from -1 to 1 are mapped to the device conductance data range from ~1μS to 200 μS, we map the device conductance to the weight range [-1, 1] by using below linear transformation (3.7),

$$G_{NORM} = \frac{G - \frac{G_{max}+G_{min}}{2}}{\frac{G_{max}-G_{min}}{2}} \quad (3.7)$$

In Eq. (3.7), we denote this normalized conductance as G_{NORM}. G, G_{max} and G_{min} are extracted from experimental data (**Fig.3.2**).

B. Hardware Implementation.

For the hardware demonstration of unsupervised learning and pruning shown in **Fig. 3.5** CBRAM devices are employed as binary synapses. The network contains 395 input neurons (crop using the same method explained in **B. Input Layer**) and 10 output neurons to classify three classes from MNIST. In 3-digits classification, out of the ~20,000 samples that represent the digits "0", "3", or "4" in the entire MNIST dataset, we randomly sample 5,000 to create our training set. We present this training set for one epoch to train our SNN. We form the test set by drawing 10,000 samples from the remaining 15,000 samples. Neurons are implemented using a custom software to program the digital peripheral circuitry of the chip. Weight summation is performed by this program to implement the integrate-and-fire neuron. Weight update values are converted into programming pulses by the peripheral circuitry to update binary weights in the digital implementation. Fixed wordline voltages are used for binary programming of CBRAM devices. We use 8 bits to represent a synaptic weight in the network, where 1 bit is used to represent the sign of the weight value and the other 7 bits stores the absolute weight value. Bit 1 is MSB and bit 7 is LSB. The weight range [-1,1] is first uniformly divided into 256 (2^8) discrete intervals $[-1 + \frac{i}{128}, -1 + \frac{i+1}{128})$, where i = 0, …, 255. Then we map the weight whose value lies in the ith interval to the ith discrete values. For example, the weights between [-1, -0.9921875) are mapped to 00000000, whereas the weights between [-0.9921875, -0.984375) are mapped to 00000001, etc. For the boundary case where the weight takes the value of 1, we map it to

11111111. The weights are updated on the hardware at run-time. We track the weight update history during the online training process (**Figure 3.S10**).

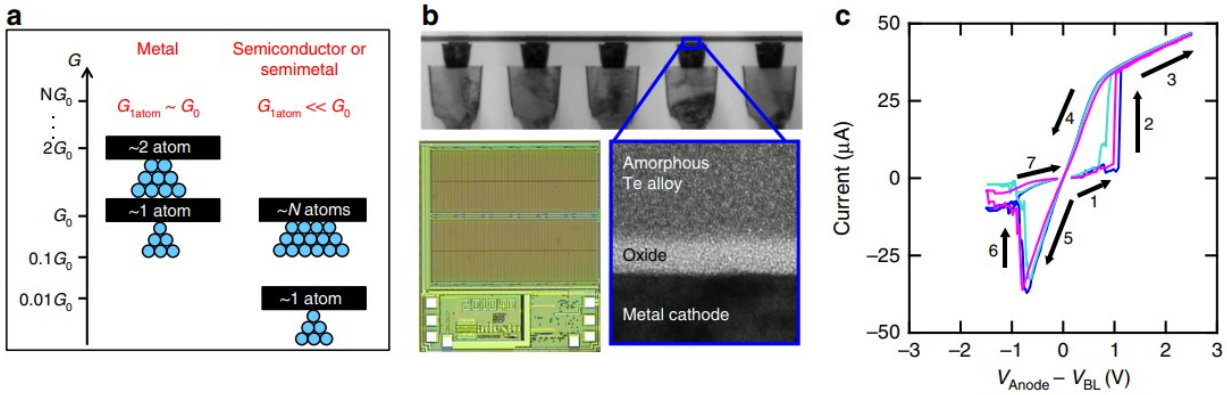


Figure 3.1: Subquantum CBRAM characteristics. **a** Semiconductor or semimetal filaments can yield lower conductance than metal filaments of comparable width. **b** Subquantum conductive bridging RAM (CBRAM) cell fabricated in a standard 130 nm logic process. Photograph shows 512 kbit subquantum CBRAM chip with one-transistor one-resistor (1T1R) array architecture. Cell cross-section shows amorphous Te alloy as anode, metal as cathode and oxide as switching layer. **c** Example of bipolar current-voltage characteristic of a subquantum CBRAM cell. Directionality of switching is shown in arrows

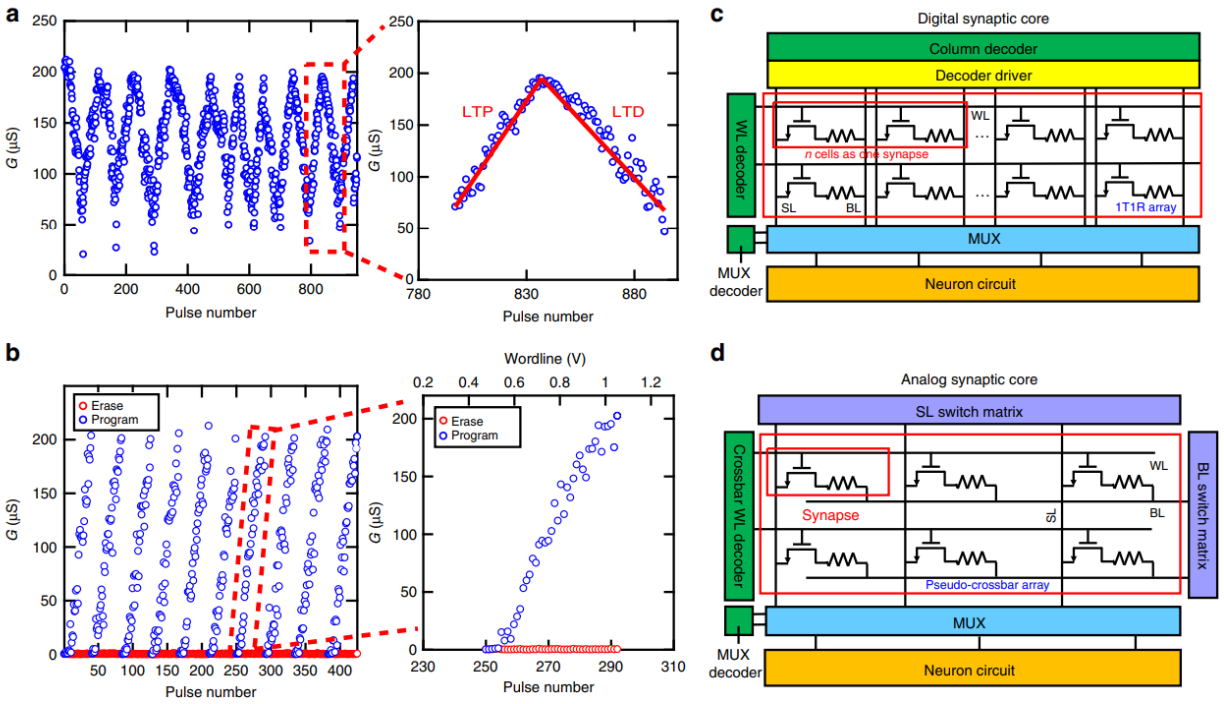


Figure 3.2: Gradual switching. **a** Gradual switching in a subquantum conductive bridging RAM (CBRAM) synapse using stepwise voltage pulses applied to the wordline (WL) (left). Callout window (right) shows one cycle of long-term potentiation (LTP) and long-term depression (LTD). Red lines are added to emphasize linearity of the conductance change. For LTP, anode (AN) = 3 V, bitline (BL) = 0, and WL stepped from 0.8 V in increments of 10 mV. For LTD, AN = 0, BL = WL, and WL stepped from 1.6 V in increments of 4 mV. **b** Gradual switching in a subquantum CBRAM synapse by WL voltage modulation. The subquantum CBRAM cells are directly programmed into the conductance state by controlling the WL voltage. The figure (left) shows a sequence of programming operations in which the WL voltage increases with step size 20 mV followed each time by an erase operation. Callout window (right) shows conductance versus pulse number and WL voltage for a representative cycle. **c** Digital synaptic core design groups multiple binary one-transistor one-resistor (1T1R) cells along the row as one synapse to represent a synaptic weight with higher precision. WL decoder is used to activate the WL in a row-by-row fashion. Column decoder can select a group of synapses to perform the weight update. The weighted sum is implemented using mux and neuron circuit. The mux is used to share the read periphery circuitry[60]. The neuron circuit which contains sense amplifier, adder and shift register can be used to read out the memory array and accumulate partial weight sum to get the final weighted sum. **d** Analog synaptic core uses a single cell with multi-level conductance states to represent one synaptic weight. The crossbar WL decoder can activate all WLs, BL read out the weighted sum results and neuron circuit contains analog-to-digital (ADC) converters convert current to digital outputs. Source line (SL) can be used to perform weight update[60].

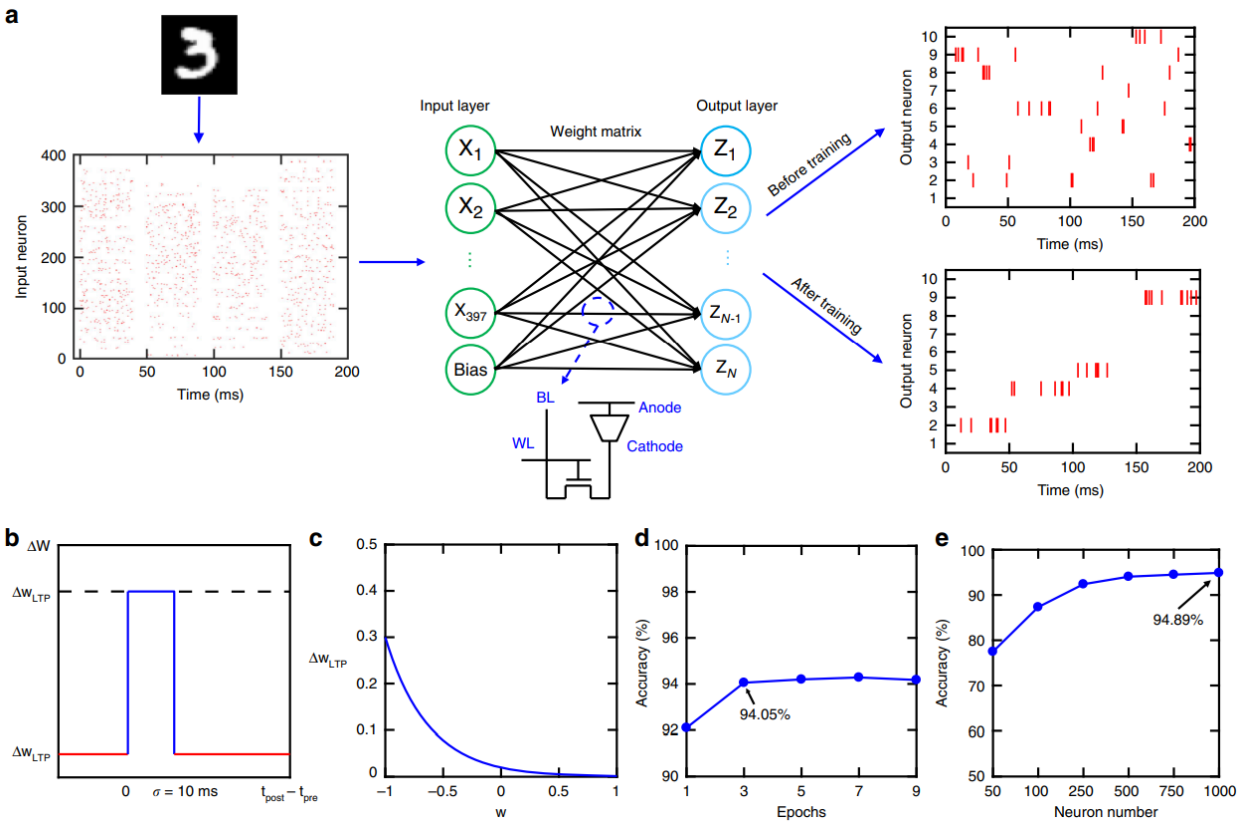


Figure 3.3: Neural Network for unsupervised learning. **a** Each input digit contains 28 × 28 = 784 pixels and has been cropped and reduced to 397 pixels. The neural network has 397 input neurons with a bias term and 500 output neurons. Input spike trains of input neurons are generated according to pixel density (from 0 to 1) and then fed to the neural network. Synaptic devices represent weights in the network. Top (before training): Random spike activity from representative 10 out of 500 output neurons before learning. Bottom (after training): Output spike trains after learning show coordinated selective firing activity as a result of unsupervised learning of digits. **b** Spike-timing-dependent plasticity (STDP) rule showing the 10 ms window for an post-pre spike time difference ($t_{post} - t_{pre}$) that determines whether a long-term potentiation (LTP) or a long-term depression (LTD) update is performed. If the firing time of an output neuron (t_{post}) is within 10 ms of the firing time of an input neuron (t_{pre}), the weight (synapse) between this input–output neuron pair is updated via LTP. Otherwise, the weight is updated via LTD. **c** The LTP update is an exponentially decaying function that depends on the current weight, and the LTD update is a constant. The exponential LTP update depending on the current weight keeps the weight values within the range [−1, 1]. **d** Recognition accuracy vs. number of training epochs. Three epochs are used in our network training. **e** Recognition accuracy vs. neuron number. Recognition accuracy does not have noticeable increase when number of output neurons is larger than 500. Therefore, 500 output neurons are used in our network model

Figure 3.4: Network pruning during training. **a** Schematics compare no pruning, soft-pruning and pruning cases. Top two row shows weight histograms of a representative output neuron. For no pruning, the spike-timing-dependent plasticity (STDP) rule results in weights ranging from −1 to 1 at the end of training. For 50% soft-pruning, it prunes weights smaller than the dashed line (weights on the left of the dashed line) to the lowest value −1. 50% Pruning prunes the weights between the two dashed lines, which represent the 50% of the weights that are centered around 0 and sets their values to 0 (red bar). Only unpruned weights continue to be updated until end of training. Bottom row shows weight visualization of all representative 10 out of 500 output neurons for no pruning, 50% soft-pruning and pruning. Soft-pruning allows for the weights to still learn the foreground and background pattern of the input samples while reducing weight update computations during training. Pruning causes the pruned weights to overwhelm the learned weights and results in inaccuracy. **b** Recognition accuracy vs. prune parameter (p) for varying pruning percentages. Prune parameter is the criterion to decide when to prune for each neuron during training. **c** Recognition accuracy vs. pruning percentage for soft-pruning and pruning performed during training. Soft-pruning during the training performs significantly better than pruning especially for high pruning percentages. The baseline accuracy (no pruning) is 94.05%. The data points are taken in steps of 10%. The parameters used in the simulation are specified in **Table 3.S6**.

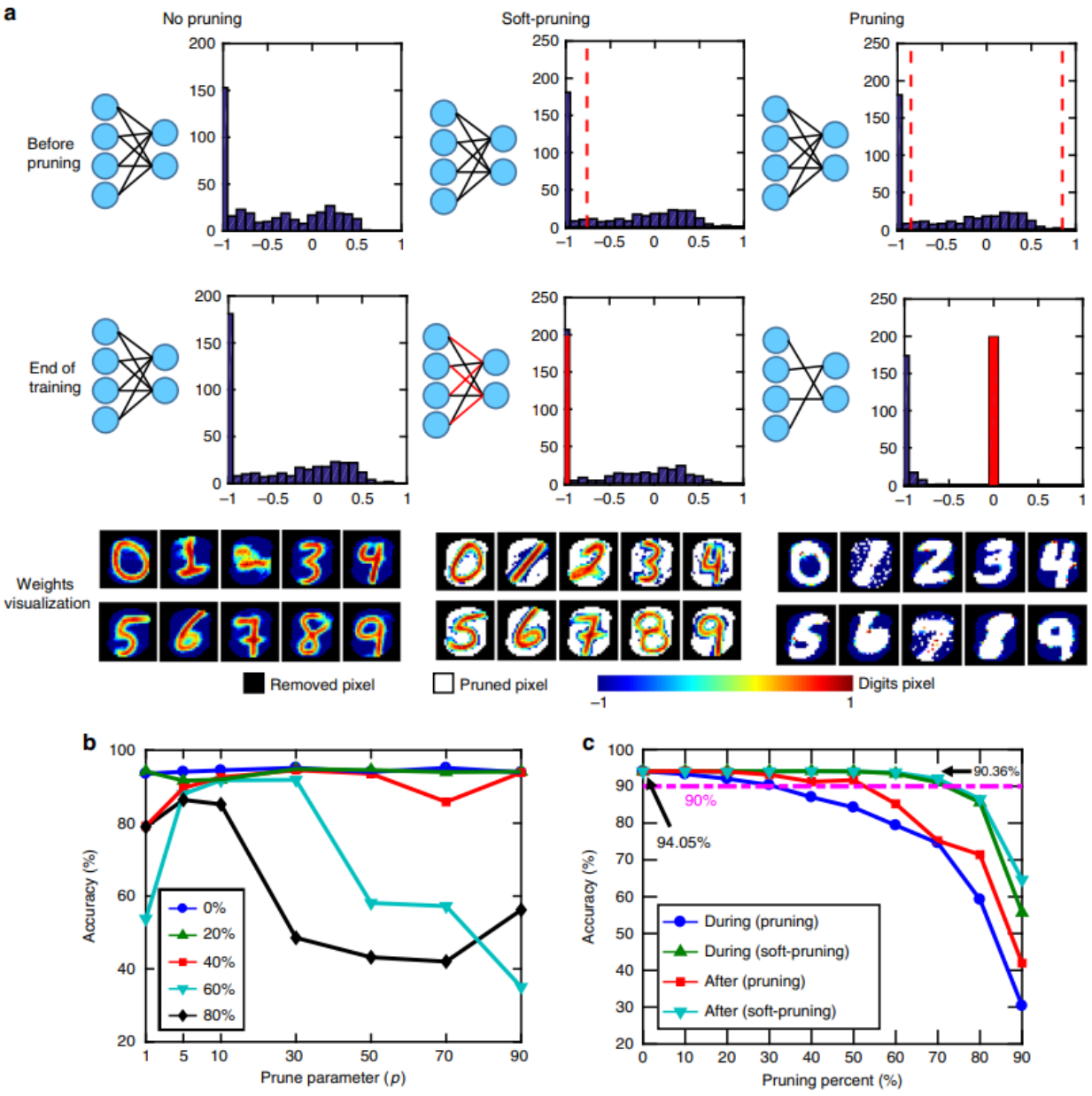

a
No pruning
Soft-pruning
Pruning
Before pruning
End of training
Weights visualization
Removed pixel
Pruned pixel
Digits pixel
b
Accuracy (%)
Prune parameter (p)
0%
20%
40%
60%
80%
c
Accuracy (%)
Pruning percent (%)
90.36%
90%
94.05%
During (pruning)
During (soft-pruning)
After (pruning)
After (soft-pruning)

Figure 3.5: Hardware implementation of unsupervised learning and pruning. **a**, Recognition accuracy vs. bit precision. The bit precision levels include 1 bit for representing the sign. 64 corresponds to 64-bit floating point. The accuracy drops below 90% after 8 bits. Test dataset has 10k images. **b** Experimentally measured binary weights from subquantum conductive bridging RAM (CBRAM) synaptic array as a result of training with 1k MNIST digits. Binary weight 1 corresponds to black pixel, which is high resistance state (~1 MΩ). Binary weight 0 corresponds to white pixel, which is low resistance state (~10 kΩ). The bit precision per weight is 8 bits with one bit used for the sign (+/−). During the training, there is a total of 8,959 weight updating events for output neurons. For no pruning (top), there were 833,889-bit updates. For training with soft-pruning at a 50% pruning rate (bottom), there were 481,921-bit updates. During the training, weights of different neurons are pruned at different times based on their learning level. At the end of training, all 10 neurons' weights have been pruned. Bits corresponding to pruned weights are marked in blue. **c** Recognition accuracy vs. training digits for 50% soft-pruning and no pruning calculated using experimental data from hardware implementation of unsupervised learning with subquantum CBRAM array. The accuracy for pruning is comparable to no pruning. Test dataset has 10k images. **d** Number of bit updates by device updates vs. training digits/timesteps with a 50% pruning rate (blue) and without pruning (red). First 1k samples are from hardware implementation of spiking neural network (SNN) using CBRAM array

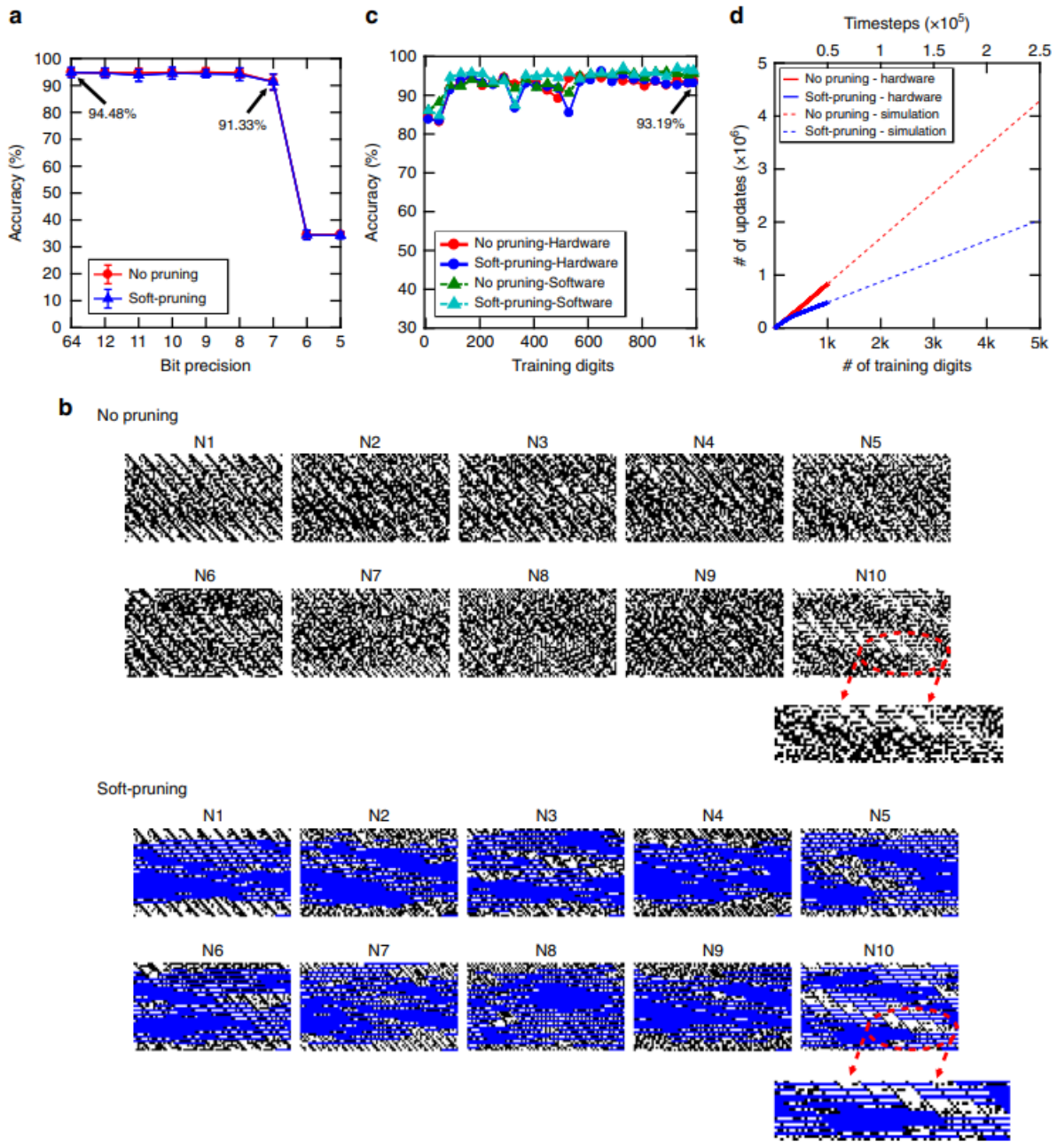

a
Accuracy (%)
Bit precision
94.48%
91.33%
No pruning
Soft-pruning
c
Accuracy (%)
Training digits
93.19%
No pruning-Hardware
Soft-pruning-Hardware
No pruning-Software
Soft-pruning-Software
d
Timesteps (×10^5)
# of updates (×10^6)
# of training digits
No pruning - hardware
Soft-pruning - hardware
No pruning - simulation
Soft-pruning - simulation
b
No pruning
N1
N2
N3
N4
N5
N6
N7
N8
N9
N10
Soft-pruning
N1
N2
N3
N4
N5
N6
N7
N8
N9
N10

Table 3.1. Network Accuracy

Precision	Accuracy
64-bit	94.05 %
CBRAM (Analog)	82 %
8-bit	92.02 %

The table summarizes the recognition accuracy of 64-bit ideal software simulation, 8-bit digital implementation and analog CBRAM synapses implementation evaluated using our network.

Table 3.2. Circuit-level Benchmark Results

	Analog	Digital (6-bit)	Digital (8-bit)
Conductance Levels	57 levels (~6 bit)	64 levels	256 levels
LTP Pulse	0.8-1.32 V/10mV/1μs	2V/1us	2V/1μs
LTD Pulse	1.6-1.84 V/4mV/10μs	2V/1us	2V/1μs
Accuracy[a]	82%	85.87%[b]	92.02%
Area (μm^2)	12,277.05[b]	35,397.34	47,233.8
Latency[a] **(s)**	516	129.72[b]	401.1
Energy[a] **(mJ)**	149.4097	62.911[b]	151.977
Leakage Power (μW)	53.78	54.14	58.99

[a] For 60,000 training images

[b] Best performing metrics The table summarizes circuit-level benchmark results using SNN+NeuroSim for analog synaptic core and digital synaptic core with 6-bit and 8-bit. The simulations are performed for 14 nm technology node

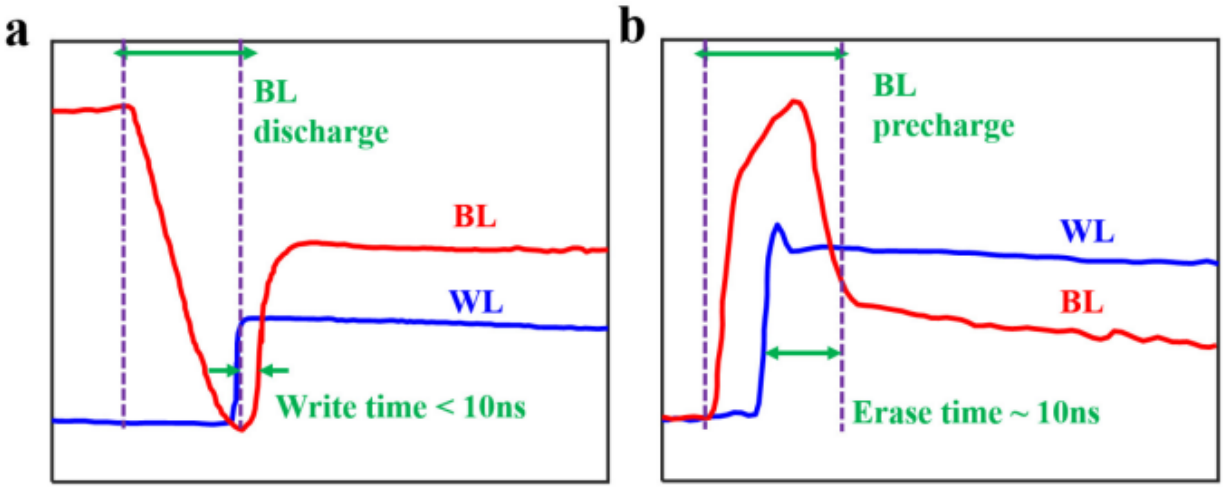


Figure 3.S1: CBRAM Write/Erase Speed. a, Bitline (BL) and wordline (WL) during a 3V program operation. The anode voltage is fixed at the 3V BL voltage. After the WL is enabled, the cell programs in <10ns. b, Bitline (BL) and wordline (WL) during an erase operation. After the WL is enabled, the cell erases in ~10ns1. For programming, the voltage to be applied to the cell is established when the BL discharges (red curve). After that, the WL (blue curve) is enabled. When the cell programs, the BL voltage increases towards the anode voltage (which is high). The programming time is the offset between the time when the WL is enabled and the time when the BL voltage is seen to increase, which is shown to be <10 ns in (a). The situation is similar for erase operation, where only the polarity is reversed (BL is high). The erase time is the offset between the time when WL is enabled and the time when the BL pulls down towards the anode (which is low). Erase time is measured as ~10ns as seen in (b).

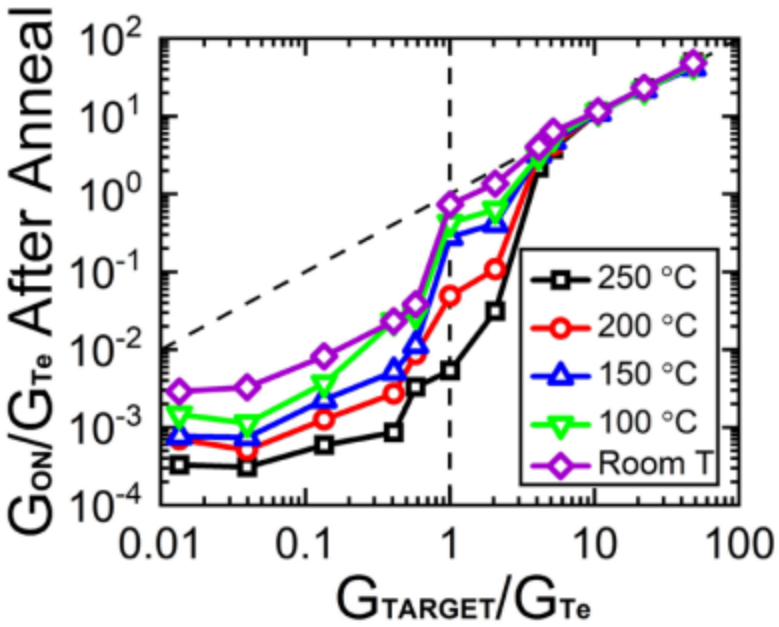


Figure 3.S2: CBRAM Retention Characteristic. Excellent retention is achieved by the subquantum cells for 10 min annealing at high temperature with the ON-state conductance a few times greater than G_{Te} are targeted[2].

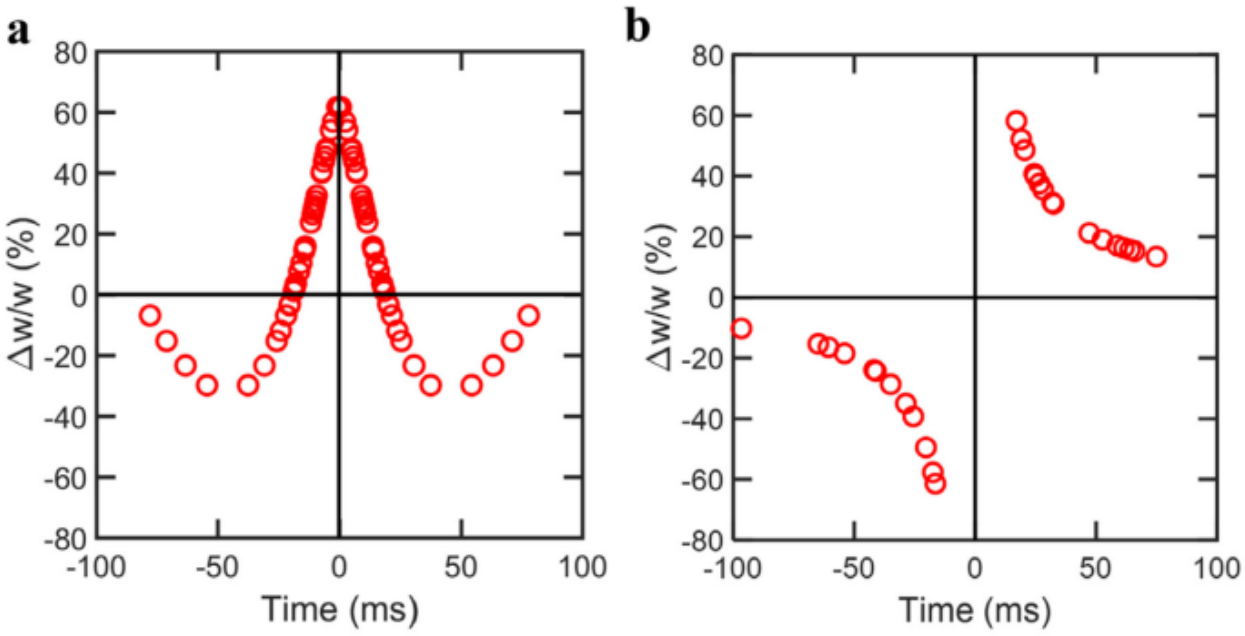


Figure 3.S3: STDP. a, Symmetric spike-timing-dependent plasticity (STDP) and b, Asymmetric STDP learning rules modeled using the gradual programming data of 1T1R subquantum CBRAM cells in Fig 2a.

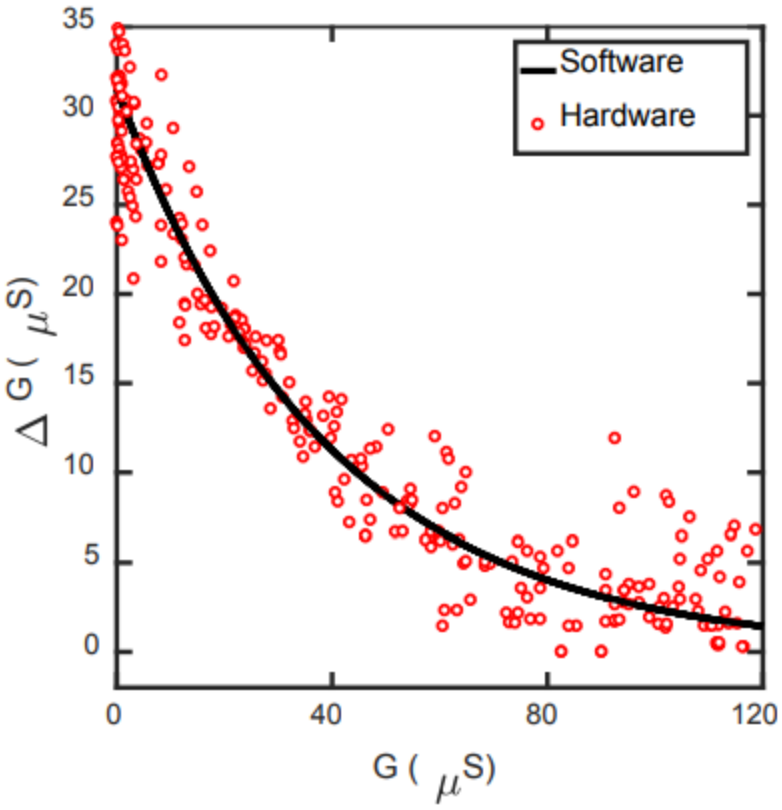


Figure 3.S4: STDP Fitting. The measured data from Fig 3.2b is fitted into the neural network weight updating rule (Fig. 3.3c).

Algorithm 1: Unsupervised SNN Training

Input: Training digits

Initialization: STDP parameters, synaptic weights

for Each simulation time step t = 1ms **do**

STEP 1 Present a training sample for 50ms and generate input Poisson spike train X_i from 0 – 200 Hz according to pixel intensity P_i

STEP 2 Compute membrane potential for each output neuron

$U_k(t) = \sum_i W_{ki} X_i(t) + b_k$

$X_i(t) = 1$ **if** X_i fired within past 10 ms; **else** $X_i(t) = 0$

STEP 3 Generate output Poisson spike train via probabilistic firing model

STEP 4 Update corresponding neuron's weights using STDP

$W_{ki} = W_{ki} + \Delta W_{ki}$

ΔW_{ki} is updated via LTP rule **if** X_i fired within past $\sigma = 10$ ms;

else ΔW_{ki} is updated via LTD rule

end for

Figure 3.S5: Unsupervised Learning Algorithm. Unsupervised spiking neural network (SNN) learning algorithm used in software neural network simulation and hardware demonstration.

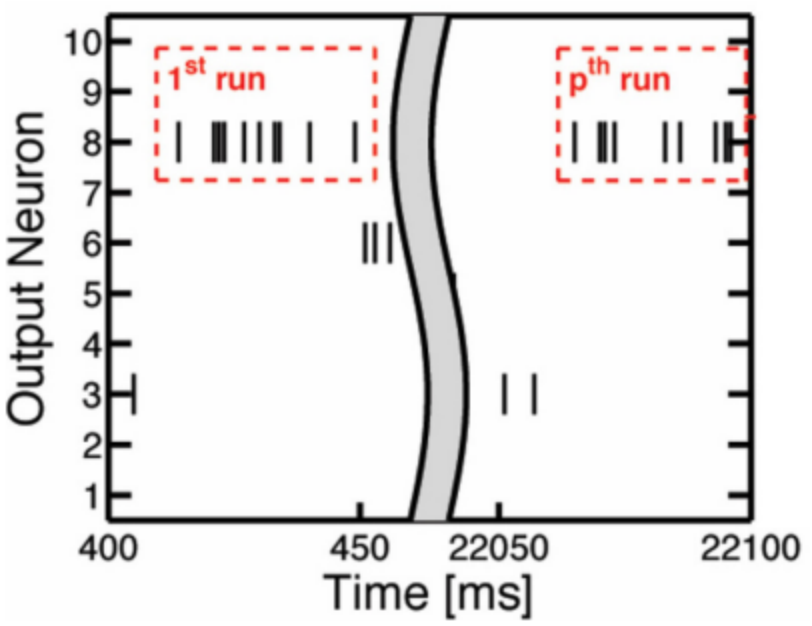


Figure 3.S6: Consecutive Spikes. The illustration of consecutive output spikes of 10 output neurons as a representative example. The consecutive output spikes of Neuron 8 are boxed in red. The consecutive spikes can be measured using integrate-and-fire neuron circuits which contain capacitors[3] or memristor[4] to store the information about how many spikes they received within a time interval representing using charge or resistance.

Algorithm 2: Soft-pruning during unsupervised learning

Input: pruning percentage **r**, valid runs **p**, spike count **c**, number of weights **N**

if there is an output spike **then**

if this neuron has yet to be pruned **then**

STEP 1 Count **p** for each neuron

STEP 2 Check each sample present time window

1. if an output neuron has accumulated **p** runs

2. if each of those run has at least **c** consecutive output spikes

STEP 3 If both criteria in **STEP 2** are met

1. Set the weight threshold to the pre-determined threshold which represents $\approx$ **r**th percentile of AAthe neuron's weights

2. Set all weights below the threshold to the lowest possible values $W_{min} = -1$.

end if

end if

Figure 3.S7: Pruning Algorithm. Soft-pruning during the training algorithm for hardware implementation.

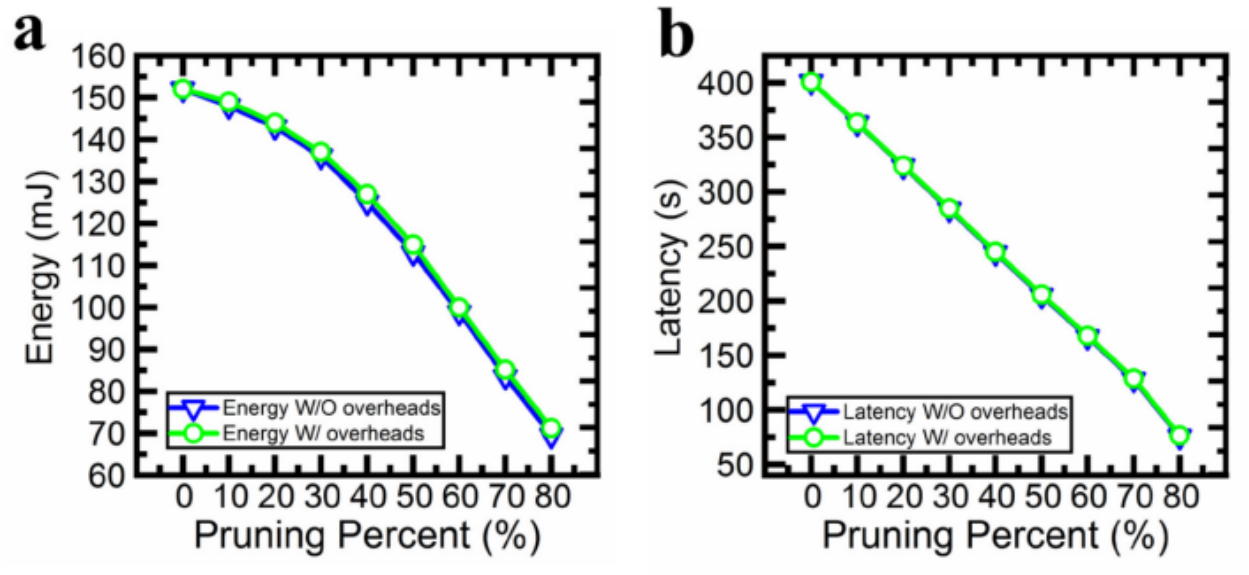


Figure 3.S8: Pruning Overheads. a, Energy and b, Latency without and with overheads estimation for soft-pruning from 10% to 80% with a step of 10% using SNN+NeuroSim. Overheads include hardware flag and setting pruned weights to -1. Without overheads (W/O overheads) results mean that flagging mechanism is implemented in software and overhead associated5 with setting pruned weights to -1 is not considered. With overheads (W/ overheads) results mean that flagging mechanism is implemented in hardware and overhead associated with setting pruned weights to -1 is considered.

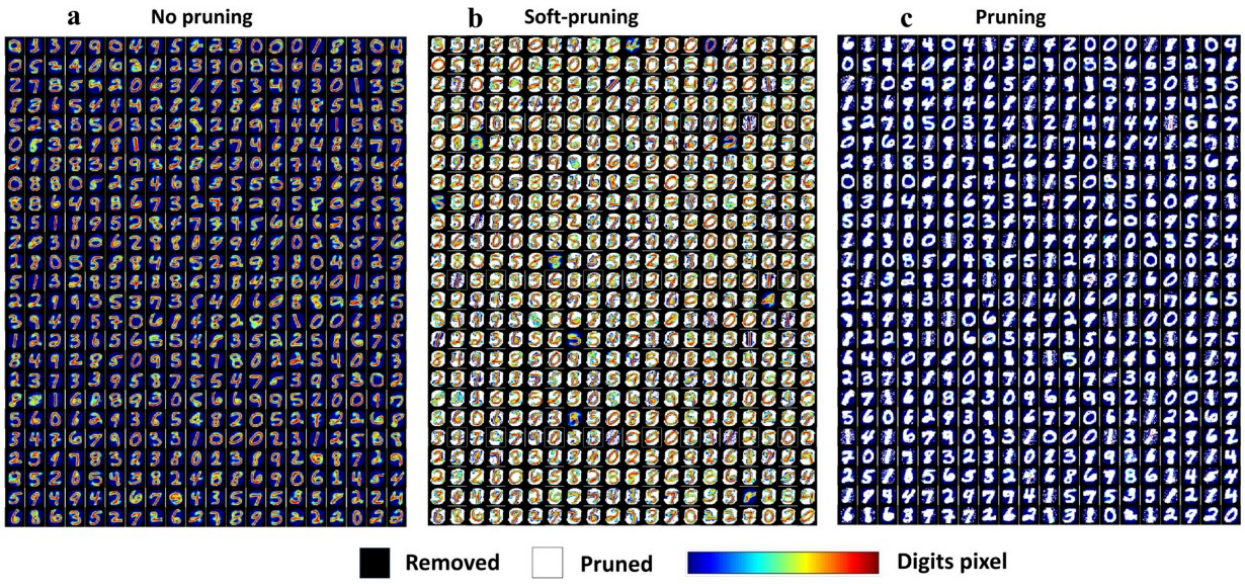


Figure 3.S9: Classification and Pruning Visualization. Weights visualization of all 500 output neurons for a, no pruning, b, 50% soft-pruning and c, pruning after training.

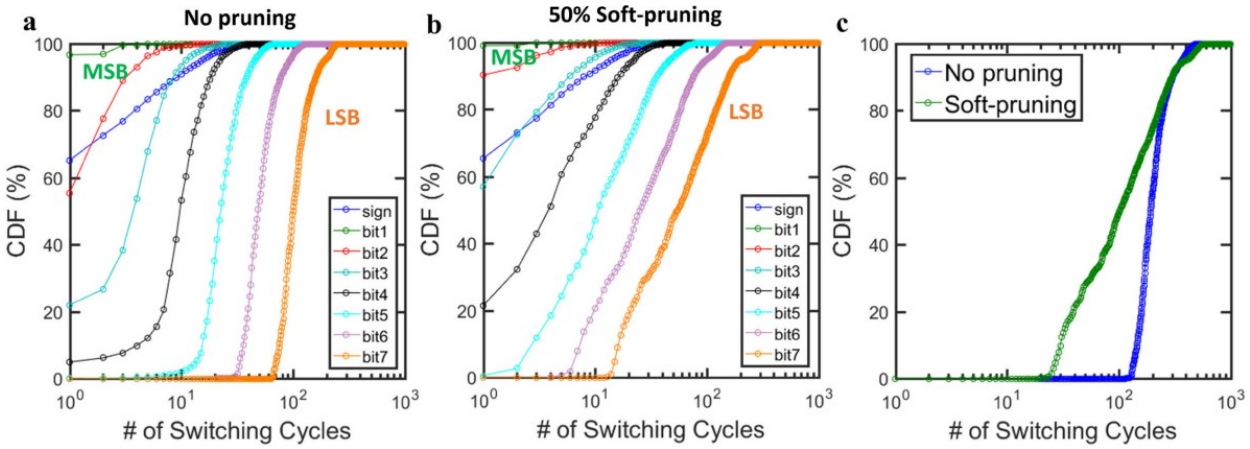


Figure 3.S10: Device Switching Cycles during Training. a, b, Empirical cumulative distribution of the switching cycles of each bit in the weight matrix during training (a) no pruning and (b) with 50% soft-pruning. We use one bit for the sign. Bit 1 is MSB and bit 7 is LSB. LSB updates more frequently than MSB in both cases. 50% pruning method effectively reduces the weight updates in every bit. c, Cumulative distribution of the switching cycles of all bits. Pruning significantly reduces the number of switching cycles for all the bits during training.

Table 3.S1: Device Energy Profile. Energy consumption in subquantum conductive bridging RAM (CBRAM), metal filament-based CBRAM cells and floating gate flash[5]. Subquantum CBRAM is 10× more energy efficient than metal filament CBRAM and 100× more energy efficient than floating gate flash, even for the maximum energy consumption cases.

	Subquantum CBRAM Synapses	**Metal Filament CBRAM Synapses**	**Floating Gate Flash**
G_{1atom}	0.03 G_0	1 G_0	-
Read voltage	1V (for WL)	1V (for WL)	3-5V (for WL)
Program voltage	1-3 V	1-3 V	|6|-|9|V
Program time	0.01-0.1 μs/cell	0.01-0.1 μs/cell	1-10 μs/cell
Program energy	0.1-10 pJ/cell	1-100 pJ/cell	1000 pJ/cell
Erase voltage	1-3 V	1-3 V	|6|-|9| V
Erase time	0.01-0.1 μs/cell	0.01-0.1 μs/cell	1 ms/cell
Erase energy	0.1-10 pJ/cell	1-100 pJ/cell	1000 pJ/cell

Table 3.S2: Pruning Overheads Estimation. Area, energy and latency estimation of no pruning, 80% soft-pruning without and with overheads. Without overheads (W/O overheads) results mean that flagging mechanism is implemented in software and overhead associated with setting pruned weights to -1 is not taken into account. With overheads (W/ overheads) results mean that flagging mechanism is implemented in hardware and overhead associated with setting pruned weights to -1 is taken into account. The numbers inside of the parentheses show the energy and latency increase due to overheads associated with [a] hardware flag and [b] setting pruned weights to -1, respectively.

Accuracy (%)	**No pruning (8-bit)**	**80% soft-pruning w/o overheads (8-bit)**	**80% soft-pruning w/ overheads[*] (8-bit + 1-bit flag)**
Area (μm^2)	47233.8	47233.8	53218.5
Energy (mJ)	151.9	69.5	71.3 (+1.09[a], +0.68[b])
Latency (s)	401.1	75.6	(+0.42[a], +0.50[b])

[*]Overheads include [a]hardware flag and [b]setting pruned weights to -1.

Table 3.S3: State-of-the-Art Unsupervised Learning Demonstration with Synaptic Devices on MNIST. The table compares the overall performance of this work with the state-of-the-art unsupervised learning demonstration with synaptic device on MNIST dataset. All the references report recognition performance simulated using single device data, while this work reports recognition accuracy for hardware implementation. The synaptic device energy consumption per programming is calculated by multiplying the pulse amplitude with the current flowing across the device and the programming pulse width. The number of updates is calculated by multiplying number of iterations in training with number of weights needed to be updated per iterations. The numbers of neurons are counted by summing up input and output neurons. If the first cell of a row contains multiple citations, subsequent values may have been taken from any one of the cited works, which are written or cited by the same authors.

Architecture	**Preprocessing**	**Learning-rule**	**# Neurons/# Plastic synapses**	**Performance**	**Hardware Type**	**Energy Consumption**	**# of updates (10^4)**
Two layer network[61-63]	None	Exponential STDP	1,184/313,600	91.6%	HfO_x based RRAM (simulation)	~0.85-24 pJ	~141
One layer network[64,65]	None	Exponential STDP	1,084/235,200	87.4%	STT-MRAM (simulation)	~0.09-96.9 pJ	~47
One layer network[66,67]	None	Probabilistic prespike rule	834/39,200	84%	WO_x based RRAM (simulation)	~1.68 nJ	~47
One layer network[68]	None	Exponential STDP	846/62,720	70%	CNT synaptic transistor (simulation)	~40-800 nJ	~47
One layer network[69]	None	Exponential STDP	794/7,840	60%	CNT synaptic transistor (simulation)	~50 nJ	~47
One layer network[70]	Yes	Exponential STDP	206/~1,980	59.8%	TiO_x based RRAM (simulation)	~25-750 nJ	~8.8
This work	Yes	Exponential STDP	405/3,950	93.19%	Subquantum CBRAM (Hardware Implementation)	~0.1-10 pJ	No prune: ~4.2 50% prune: ~2

Table 3.S4: State-of-the-Art Software Demonstration of Unsupervised Learning Demonstration on MNIST. The table compares the performance (recognition accuracy) of this work with the state-of-the-art software demonstrations of unsupervised learning on MNIST dataset. The numbers of neurons are counted by summing up input and output neurons.

Architecture	**Preprocessing**	**Learning-rule**	**# Neurons/# Plastic synapses**	**Performance**	**Hardware Type**
Spiking Deep neural network[71]	None	Simplified STDP	N/A	98.4%	No
Two layer network[63]	None	Exponential STDP	7,184/5,017,600	95%	No
Two layer network[72]	Yes	Exponential STDP	~600/~50,000	80.14%	No
This work	yes	Exponential STDP	~898/~199,000	94.05%	Subquantum CBRAM

Table 3.S5: State-of-the-Art Pruning Techniques. This table summarizes pruning methods. The first four can only be applied after training as reported by the references. The method described in this work is the first to be implemented in hardware and also can be applied either during or after training. Energy savings are relative to the result with no pruning. Energy savings for this work is obtained from Supplementary Table 2. Accuracy loss is based on the result with 50% pruning.

Pruning Method	**Machine Learning Tasks**	**Network Structure**	**During Training**	**After Training**	**Hardware Implementation of Pruning**	**Simulation System**	**Energy Savings**	**Accuracy Loss (50% pruning)**
Deep Compression[73]	ImageNet pattern Recognition	AlexNet	No	Yes	No	EIE (SRAM)	53%	~ -0.1% - + 0.1%
SIMD-Aware Pruning[74]	ImageNet pattern Recognition	AlexNet	No	Yes	No	CPU,GPU	~53%	~ -1%-+1%
Energy-Aware Pruning[75]	ImageNet pattern Recognition	AlexNet	No	Yes	No	CPU	73%	<1%
Structural Pruning[76]	CIFAR-10 Pattern Recognition	CNN	No	Yes	No	CPU	N/A	~-1%-+2%
This work	MNIST Pattern recognition	Fully Connected	Yes	Yes	Yes	CBRAM synaptic array	54.2%	During: +0.9% After: -0.5%

Table 3.S6: Simulation Parameters. All the parameters used for the simulations are listed above.

Parameters		10-Digits		
		Training	Labeling	Testing
# of Neuron	Input		398	
	Output		500	
Firing Rate (Hz)	Input	200	200	200
	Output	200	200	600
Image presenting Time (ms)		50	50	200
Pruning Threshold	Prune Parameter (p)	10	-	-
	Spike Count	8	-	-
STDP		a = 0.0667 b = 2.5 c = 0.0167	-	-

8. Supplementary Notes

A. Supplementary Note 1

Supplementary Figure 2 shows an experimentally quantified stability (retention) as a function of conductance. It can be seen that the subquantum CBRAM has robust thermal stability. In this figure, the x-axis is the target cell conductance during a program operation. Note that the x-axis is normalized so that the value 1 corresponds to the conductance of a filament with a 1-atom constriction. The y-axis is the actual conductance measured after annealing at temperatures ranging from room temperature to 250 °C. The plot shows that stability (retention) is poor if the targeted conductance level is lower than the conductance of an incomplete filament whose thinnest spot is less than 1 atom thick. However, once the target conductance is above the value of the 1-atom thick filament, the post-anneal conductance quickly approaches the targeted value. Therefore, the filament is increasingly stable as it becomes thicker. Note that filaments can be stable even at the highest temperature used in the study (250 °C).

B. Supplementary Note 2

Asymmetric and symmetric STDP are implemented using the same spike scheme described by *Kuzum. et al.*[77]. Pre and post spikes are implemented to the word line and bit line of the 1T1R array. Time overlap of the pre and post spikes allows programming of the CBRAM synapse. The spike timing differences between the pre and post spikes are translated to the amplitude of voltage pulses applied to the word line. Integrate-and-fire neurons are implemented using a computer program and pulse generators, and the pulses are applied to the WL and BL of the device to modulate the conductance change.

C. Supplementary Note 3

The overhead costs of the pruning algorithm can be estimated using our SNN platform for NeuroSim. The first overhead is that the pruned weights need to be flagged to prevent them from further updating. This can be implemented by adding an additional bit with an initial value of 0 to serve as a hardware flag for pruning. We update the pruning flags of an output neuron's weights to '1' when they have been pruned during the training. Note that since the weights are only pruned once during the entire training, each hardware flag is just written once. Before weight update, we read the hardware flag of the winner neuron's weights and the weight will not be updated if its flag is '1'. Another overhead is setting the pruned weights to -1. We take these two overheads into account in our simulation for pruning using digital hardware implementation (**Fig.3.2c**). In **Table 3.S2**, both overhead costs are estimated in terms of area, energy and latency based on the peripheral programming circuitry shown in **Fig. 3.2c** for no pruning, 80% soft-pruning without (W/O overheads) and with overheads (W/ overheads). As can be seen from this table, the area is increased by ~12.7% because the flag only takes up one extra bit for each synapse. The hardware flag increases energy and latency by ~1.6% (1.09mJ) and ~0.6% (0.42s), respectively. Setting pruned weights to -1 increases energy and latency by ~0.98% (0.68mJ) and ~0.66% (0.5s), respectively. In summary, total energy and latency are increased by ~2.5% (~1.7mJ) and ~1.2% (~0.92s) due to the overheads of pruning implementation. This is significantly smaller and hence negligible compared to the energy and latency gains from pruning.

9. References

1 LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *nature* **521**, 436 (2015).

2 Krizhevsky, A., Sutskever, I. & Hinton, G. E. in *Advances in neural information processing systems.* 1097-1105.

3 Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A. & Bernstein, M. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* **115**, 211-252 (2015).

4 Collobert, R. & Weston, J. in *Proceedings of the 25th international conference on Machine learning.* 160-167 (ACM).

5 Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P. & Sainath, T. N. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine* **29**, 82-97 (2012).

6 Chen, C., Seff, A., Kornhauser, A. & Xiao, J. in *Proceedings of the IEEE International Conference on Computer Vision.* 2722-2730.

7 Vishwakarma, S. & Agrawal, A. A survey on activity recognition and behavior understanding in video surveillance. *The Visual Computer* **29**, 983-1009 (2013).

8 Cireşan, D. C., Giusti, A., Gambardella, L. M. & Schmidhuber, J. in *International Conference on Medical Image Computing and Computer-assisted Intervention.* 411-418 (Springer).

9 Lane, N. D., Bhattacharya, S., Georgiev, P., Forlivesi, C. & Kawsar, F. in *Proceedings of the 2015 International Workshop on Internet of Things towards Applications.* 7-12 (ACM).

10 Krizhevsky, A. & Hinton, G. Learning multiple layers of features from tiny images. *Technical report, University of Toronto* **1**, 7 (2009).

11 Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. & Fei-Fei, L. in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.* 248-255 (IEEE).

12 Asuncion, A. & Newman, D. UCI machine learning repository. (2007).

13 Salelanonda, G. *Learning how to learn: Toddlers vs. neural networks*, <https://www.eetimes.com/author.asp?section_id=36&doc_id=1330538> (2016).

14 Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V. & Lanctot, M. Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484-489 (2016).

15 Kandel, E. R., Schwartz, J. H., Jessell, T. M., Siegelbaum, S. A. & Hudspeth, A. J. *Principles of neural science*. Vol. 4 (McGraw-hill New York, 2000).

16 Kuzum, D., Jeyasingh, R. G. & Wong, H.-S. P. in *Electron Devices Meeting (IEDM), 2011 IEEE International.* 30.33. 31-30.33. 34 (IEEE).

17 Eryilmaz, S. B., Kuzum, D., Jeyasingh, R. G., Kim, S., BrightSky, M., Lam, C. & Wong, H.-S. P. in *Electron Devices Meeting (IEDM), 2013 IEEE International.* 25.25. 21-25.25. 24 (IEEE).

18 Mahalanabis, D., Sivaraj, M., Chen, W., Shah, S., Barnaby, H. J., Kozicki, M. N., Christen, J. B. & Vrudhula, S. in *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on.* 2314-2317 (IEEE).

19 Yu, S., Li, Z., Chen, P.-Y., Wu, H., Gao, B., Wang, D., Wu, W. & Qian, H. in *Electron Devices Meeting (IEDM), 2016 IEEE International.* 16.12. 11-16.12. 14 (IEEE).

20 Milo, V., Pedretti, G., Carboni, R., Calderoni, A., Ramaswamy, N., Ambrogio, S. & Ielmini, D. in *Electron Devices Meeting (IEDM), 2016 IEEE International.* 16.18. 11-16.18. 14 (IEEE).

21 Park, S., Kim, H., Choo, M., Noh, J., Sheri, A., Jung, S., Seo, K., Park, J., Kim, S., Lee, W., Shin, J., Lee, D., CHoi, G., Woo, J., Cha, E., Jang, J., Park, C., Jeon, M., Lee, B., Lee, B. H. & Hwang, H. in *Electron Devices Meeting (IEDM), 2012 IEEE International.* 10.12. 11-10.12. 14 (IEEE).

22 Serb, A., Bill, J., Khiat, A., Berdan, R., Legenstein, R. & Prodromakis, T. Unsupervised learning in probabilistic neural networks with multi-state metal-oxide memristive synapses. *Nature communications* **7**, 12611 (2016).

23 Choi, S., Sheridan, P. & Lu, W. D. Data clustering using memristor networks. *Scientific reports* **5**, 10492 (2015).

24 Wang, Z., Joshi, S., Savel'ev, S., Song, W., Midya, R., Li, Y., Rao, M., Yan, P., Asapu, S., Zhuo, Y., Jiang, H., Lin, P., Li, C., Yoon, J. H., Upadhyay, N. K., Zhang, J., Hu, M., Strachan, P. J., Barnell, M., Wu, Q., Wu, H., Williams, R. S., Xia, Q. & Yang, J. J. Fully memristive neural networks for pattern classification with unsupervised learning. *Nature Electronics* **1**, 137 (2018).

25 Jeong, Y., Lee, J., Moon, J., Shin, J. H. & Lu, W. D. K-means data clustering with memristor networks. *Nano letters* (2018).

26 Li, C., Belkin, D., Li, Y., Yan, P., Hu, M., Ge, N., Jiang, H., Montgomery, E., Lin, P., Wang, Z., Song, W., Strachan, J. P., Barnell, M., Wu Qing, Williams, R. S., Yang, J. J. & Xia, Q. Efficient and self-adaptive in-situ learning in multilayer memristor neural networks. *Nature Communications* **9**, 2385 (2018).

27 Hu, M., Graves, C. E., Li, C., Li, Y., Ge, N., Montgomery, E., Davila, N., Jiang, H., Williams, R. S., Yang, J. J., Xia, Q. & Strachan, J. P. Memristor-Based Analog Computation and Neural Network Classification with a Dot Product Engine. *Advanced Materials* **30**, 1705914 (2018).

28 Ambrogio, S., Narayanan, P., Tsai, H., Shelby, R. M., Boybat, I., Nolfo, C., Sidler, S., Giordano, M., Bodini, M., Farinha, N. C., Killeen, B., Cheng, C., Jaoudi, Y. & Burr, G. W. Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature* **558**, 60 (2018).

29 Kataeva, I., Merrikh-Bayat, F., Zamanidoost, E. & Strukov, D. in *Neural Networks (IJCNN), 2015 International Joint Conference on.* 1-8 (IEEE).

30 Bayat, F. M., Prezioso, M., Chakrabarti, B., Nili, H., Kataeva, I. & Strukov, D. Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits. *Nature communications* **9**, 2331 (2018).

31 Du, C., Cai, F., Zidan, M. A., Ma, W., Lee, S. H. & Lu, W. D. Reservoir computing using dynamic memristors for temporal information processing. *Nature communications* **8**, 2204 (2017).

32 Boybat, I., Le Gallo, M., Nandakumar, S., Moraitis, T., Parnell, T., Tuma, T., Rajendran, B., Leblebici, Y., Sebastian, A. & Eleftheriou, E. Neuromorphic computing with multi-memristive synapses. *Nature communications* **9**, 2514 (2018).

33 Nandakumar, S., Le Gallo, M., Boybat, I., Rajendran, B., Sebastian, A. & Eleftheriou, E. in *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on.* 1-5 (IEEE).

34 Liu, C., Yang, Q., Yan, B., Yang, J., Du, X., Zhu, W., Jiang, H., Wu, Q., Barnell, M. & Li, H. in *VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on.* 110-115 (IEEE).

35 Han, S., Mao, H. & Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* (2015).

36 Yang, T.-J., Chen, Y.-H. & Sze, V. Designing energy-efficient convolutional neural networks using energy-aware pruning. *arXiv preprint arXiv:1611.05128* (2016).

37 Graham, J. Children and brain development: What we know about how children learn. *Cooperative Extension Publication* (2011).

38 Reed, R. Pruning algorithms-a survey. *IEEE transactions on Neural Networks* **4**, 740-747 (1993).

39 Goh, Y.-S. & Tan, E.-C. in *TENCON'94. IEEE Region 10's Ninth Annual International Conference. Theme: Frontiers of Computer Technology. Proceedings of 1994.* 805-808 (IEEE).

40 Jameson, J. R. & Kamalanathan, D. Subquantum conductive-bridge memory. *Applied Physics Letters* **108**, 053505 (2016).

41 Vis, V. A. Photoconductivity in Single-Crystal Tellurium. *Journal of Applied Physics* **35**, 360-364 (1964).

42 Kuzum, D., Jeyasingh, R. G., Lee, B. & Wong, H.-S. P. Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing. *Nano letters* **12**, 2179-2186 (2011).

43 Yu, S., Chen, P.-Y., Cao, Y., Xia, L., Wang, Y. & Wu, H. in *Electron Devices Meeting (IEDM), 2015 IEEE International.* 17.13. 11-17.13. 14 (IEEE).

44 Chen, P.-Y. & Yu, S. *Impact of Nonideal Resistive Synaptic Device Behaviors on Implementation of Sparse Coding Algorithm*. (Springer, 2017).

45 Kuzum, D., Jeyasingh, R. G. D., Yu, S. & Wong, H.-S. P. Low-energy robust neuromorphic computation using synaptic devices. *IEEE Transactions on Electron Devices* **59**, 3489-3494 (2012).

46 Chen, P.-Y., Peng, X. & Yu, S. NeuroSim: A Circuit-Level Macro Model for Benchmarking Neuro-Inspired Architectures in Online Learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2018).

47 Lee, J. H., Delbruck, T. & Pfeiffer, M. Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience* **10** (2016).

48 Neftci, E., Das, S., Pedroni, B., Kreutz-Delgado, K. & Cauwenberghs, G. Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in neuroscience* **7**, 272 (2014).

49 Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., Jackson, B. L., Imam, N., Guo, C. & Nakamura, Y. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* **345**, 668-673 (2014).

50 Indiveri, G., Linares-Barranco, B., Hamilton, T. J., Van Schaik, A., Etienne-Cummings, R., Delbruck, T., Liu, S.-C., Dudek, P., Häfliger, P. & Renaud, S. Neuromorphic silicon neuron circuits. *Frontiers in neuroscience* **5** (2011).

51 Nessler, B., Pfeiffer, M., Buesing, L. & Maass, W. Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS computational biology* **9**, e1003037 (2013).

52 Nessler, B., Pfeiffer, M. & Maass, W. in *Advances in neural information processing systems.* 1357-1365.

53 Diehl, P. U. & Cook, M. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience* **9**, 99 (2015).

54 Kulkarni, S. R. & Rajendran, B. Spiking neural networks for handwritten digit recognition—Supervised learning and network optimization. *Neural Networks* **103**, 118-127 (2018).

55 Lee, J. H., Delbruck, T. & Pfeiffer, M. Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience* **10**, 508 (2016).

56 Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S.-C. & Pfeiffer, M. in *Neural Networks (IJCNN), 2015 International Joint Conference on.* 1-8 (IEEE).

57 Kijsirikul, B. & Chongkasemwongse, K. in *Proceedings of IEEE Int. Conf. on Neural Networks.* 1876-1880.

58 Chen, P.-Y., Lin, B., Wang, I., Hou, T.-H., Ye, J., Vrudhula, S., Seo, J.-s., Cao, Y. & Yu, S. in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design.* 194-199 (IEEE Press).

59 Zhang, J., Wang, Z. & Verma, N. In-Memory Computation of a Machine-Learning Classifier in a Standard 6T SRAM Array. *J. Solid-State Circuits* **52**, 915-924 (2017).

60 Chen, P.-Y., Peng, X. & Yu, S. NeuroSim: A Circuit-Level Macro Model for Benchmarking Neuro-Inspired Architectures in Online Learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **37**, 3067-3080, doi:10.1109/tcad.2018.2789723 (2018).

61 Tosson, A. M. S., Yu, S., Anis, M. H. & Wei, L. A Study of the Effect of RRAM Reliability Soft Errors on the Performance of RRAM-Based Neuromorphic Systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **25**, 3125-3137, doi:10.1109/tvlsi.2017.2734819 (2017).

62 Kuzum, D., Yu, S. & Wong, H. S. Synaptic electronics: materials, devices and applications. *Nanotechnology* **24**, 382001, doi:10.1088/0957-4484/24/38/382001 (2013).

63 Diehl, P. U. & Cook, M. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front Comput Neurosci* **9**, 99, doi:10.3389/fncom.2015.00099 (2015).

64 Zhang, D., Zeng, L., Zhang, Y., Zhao, W. & Klein, J. O. in *2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH).* 173-178.

65 Vincent, A. F., Larroque, J., Locatelli, N., Ben Romdhane, N., Bichler, O., Gamrat, C., Zhao, W. S., Klein, J. O., Galdin-Retailleau, S. & Querlioz, D. Spin-transfer torque magnetic memory as a stochastic memristive synapse for neuromorphic systems. *IEEE Trans Biomed Circuits Syst* **9**, 166-174, doi:10.1109/TBCAS.2015.2414423 (2015).

66 Sheridan, P., Ma, W. & Lu, W. in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1078-1081.

67 Chang, T., Jo, S.-H., Kim, K.-H., Sheridan, P., Gaba, S. & Lu, W. Synaptic behaviors and modeling of a metal oxide memristive device. *Applied Physics A* **102**, 857-863, doi:10.1007/s00339-011-6296-1 (2011).

68 Kim, S., Choi, B., Lim, M., Yoon, J., Lee, J., Kim, H. D. & Choi, S. J. Pattern Recognition Using Carbon Nanotube Synaptic Transistors with an Adjustable Weight Update Protocol. *ACS Nano* **11**, 2814-2822, doi:10.1021/acsnano.6b07894 (2017).

69 Kim, S., Yoon, J., Kim, H.-D. & Choi, S.-J. Carbon Nanotube Synaptic Transistor Network for Pattern Recognition. *ACS Applied Materials & Interfaces* **7**, 25479-25486, doi:10.1021/acsami.5b08541 (2015).

70 Zahari, F., Hansen, M., Mussenbrock, T., Ziegler, M. & Kohlstedt, H. Pattern recognition with TiO x-based memristive devices. *AIMS Materials Science* **2**, 203-216 (2015).

71 Kheradpisheh, S. R., Ganjtabesh, M., Thorpe, S. J. & Masquelier, T. STDP-based spiking deep convolutional neural networks for object recognition. *Neural Netw* **99**, 56-67, doi:10.1016/j.neunet.2017.12.005 (2018).

72 Nessler, B., Pfeiffer, M., Buesing, L. & Maass, W. Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS Comput Biol* **9**, e1003037, doi:10.1371/journal.pcbi.1003037 (2013).

73 Han, S., Pool, J., Tran, J. & Dally, W. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems* **28** (2015).

74 Yu, J., Lukefahr, A., Palframan, D., Dasika, G., Das, R. & Mahlke, S. in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*. 548-560.

75 Yang, T.-J., Chen, Y.-H. & Sze, V. 5687-5695.

76 Anwar, S., Hwang, K. & Sung, W. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)* **13**, 1-18 (2017).

77 Kuzum, D., Jeyasingh, R. G. D., Yu, S. & Wong, H. S. P. Low-Energy Robust Neuromorphic Computation Using Synaptic Devices. *IEEE Transactions on Electron Devices* **59**, 3489-3494, doi:10.1109/ted.2012.2217146 (2012).

10. Acknowledgements

The authors acknowledge support from the Office of Naval Research Young Investigator Award (N00014161253), National Science Foundation (ECCS-1752241, ECCS-1734940) and the UC San Diego Frontiers of Innovation Scholars Program for funding this research. Adesto and CBRAM are Trademarks of Adesto Technologies Corp.

Chapter 3, in full, is a reprint of the material as it appears in Nature Communications. Shi, Yuhan; Nguyen, Leon; Oh, Sangheon; Liu, Xin; Koushan, Foroozan; Jameson, John, R.; Kuzum, Duygu, Neuroinspired unsupervised learning and pruning with subquantum CBRAM arrays, 2019. The dissertation author was a co-author of this paper.

CHAPTER 4. INTEGRATION OF AG-CBRAM CROSSBARS AND MOTT-RELU NEURONS FOR EFFICIENT IMPLEMENTATION OF DEEP NEURAL NETWORKS IN HARDWARE

In-memory computing with emerging non-volatile memory devices (eNVMs) has shown promising results in accelerating matrix-vector multiplications (MVMs). However, activation function calculations are still being implemented with general processors or large and complex neuron peripheral circuits. Here, we present integration of Ag-based conductive bridge random access memory (Ag-CBRAM) crossbar arrays with Mott-ReLU activation neurons for scalable, energy and area efficient hardware implementation of deep neural networks (DNNs). We develop Ag-CBRAM devices that can achieve high ON/OFF ratio and multi-level programmability. Compact and energy efficient Mott-ReLU neuron devices implementing rectified linear unit (ReLU) activation function are directly connected to the columns of Ag-CBRAM crossbars to compute the output from the weighted sum current. We implement convolution filters and activations for VGG-16 using our integrated hardware and demonstrate successful generation of feature maps for CIFAR-10 images in hardware. Our approach paves a new way towards building highly compact and energy efficient eNVMs based in-memory computing system.

1. Introduction

Deep neural networks (DNNs) have been widely successful in solving difficult problems in computer vision, speech recognition, machine translation, playing board and video games and

medical diagnosis. DNNs have been constantly making breakthroughs in improving the state-of-the-art computational accuracy[1]. Large-scale DNNs require a very large number of matrix vector multiplication (MVM) operations in each layer followed by non-linear neuron activations between the layers (**Fig.4.1a**). By introducing non-linear transformation to the input, activation function plays an important role in solving vanishing gradient problem and making the network capable to learn and perform more complex tasks[2]. Although in-memory computing with emerging non-volatile memory arrays (eNVMs) considerably accelerate computation of MVMs[3,4], current approaches still require external processors or complex peripheral circuits to implement neuron activations. Analogue-to-digital converters (ADC) is typically used in computing activation function and propagate data through eNVM layers. However, it has been shown that the power of 9-bit SAR-ADCs is roughly 1W compared to 0.3W dissipated on a 4096 × 4096 eNVM array for MVM operations[5]. The energy and latency overheads associated with separate implementation of the weights and activations significantly increase the energy consumption and constitute a major bottleneck for scalability of the hardware with ever evolving neural network architectures. Recent advances have explored using analogue CMOS circuits[6] or an ADC with reconfigurable function mapping[7] to implement activation functions in hardware. Although these approaches improve processing speed, they are difficult to be directly integrated as part of the eNVM array due to area mismatch compared to the compact array[8]. To overcome this limitation, we have previously developed a Mott activation neuron that implements the rectified linear unit function in the analogue domain[8]. Integration of resistive memory synaptic arrays with Mott-ReLU neurons could enable full hardware implementation of DNNs through direct combination of MVM operations with activation functions. To that end, in this work, we experimentally investigate integration of Ag-CBRAM synaptic crossbars for MVMs and

compact Mott neuron devices for activation functions to implement a VGG-16 inference task. Our hardware demonstration concentrates on convolutional and activation layers, which are main building blocks of VGG-16. Our Ag-CBRAM device exhibits high ON/OFF ratio (~10^{10}) and 4-bit multi-level switching, which are suitable for performing large scale MVMs in DNNs. Mott activation neurons integrated with CBRAM arrays emulate characteristics of ReLU activation function, which is the most frequently used activation functions in DNNs[9]. As shown in Fig.1b, a crossbar comprises Ag-CBRAM devices implementing synaptic layer accept inputs in the wordlines (WLs) and generates weighted sum current in the bitlines (BLs). Each column of Ag-CBRAM crossbars is connected to a nano-scale Mott neuron device for direct computation of ReLU activation using the weighted sum. The outputs of Mott ReLU devices can be directly fed to the WLs of the following Ag-CBRAM layers (**Fig.4.1c**). The rest of the chapter is organized as follows. First, we present characterization of Ag-CBRAM devices including DC switching behavior, variation, retention, endurance, and multi-level switching. Then, we share our results on the volatile four-terminal Mott activation neuron device based on vanadium dioxide (VO_2) and experimentally measured input-output characteristics for implementation of ReLU activation function. Transient response of the device is also measured to validate its low energy consumption. Lastly, we demonstrate hardware implementation of a CIFAR-10 image classification task using VGG-16 by integrating Ag-CBRAM crossbars and Mott-ReLU neurons using a custom PCB board. Our results based on integration of Ag-CBRAM crossbars and Mott-ReLU neurons suggest that the small size and energy efficiency of the Mott activation neuron can replace power-hungry CMOS circuits for ReLU activation and allow direct stacking of multiple synaptic layers.

2. Ag-based CBRAM

The simplicity of fabrication makes lateral eNVM devices desirable for direct integration on the back end of line (BEOL) CMOS circuitry. For implementing synaptic layers, we first developed a lateral Ag-based CBRAM crossbar that can be fabricated at the wafer scale as shown in **Fig.4.2a**. The 4-inch wafer contains 16×16 and 32×32 crossbar arrays as well as single devices for electrical characterization (**Fig.4.2b**). For crossbar fabrication, we started with 300nm SiO_2/Si wafer and deposited 50nm-thick Ag layer via DC sputtering. Then a 5 μm × 20 μm Ag channels along with its BL were patterned via photolithography and wet-etching. Next, 250 nm SiO_2 was deposited as an insulating layer with PECVD method. After patterning SiO_2 to open via holes, 200nm Cr/Au was deposited and patterned for WLs as well as defining contact pads for BLs and WLs. Single test devices were fabricated in a similar way. The Ag-CBRAM devices (**Fig. 4.2c**) exhibit an initial low resistance as fabricated and need to undergo an oxidation step whereby the device is transformed from its highly conductive "pristine" state (**Fig.4.2c**) to an oxidized high-resistance state (HRS) to initialize the subsequent switching (**Fig.4.2d** and **e**) using a low amplitude voltage sweep. Figure.3a demonstrates this forming process. The forming process here is different from the conventional forming process involving formation of a conductive filament in metal oxide based RRAM devices. Instead, here the forming step transforms the conductive metal layer into an oxidized state which exhibits resistive switching. As the voltage input was swept from 0V to 1V, the device current increased nearly proportionately up to ~45 mA. However, when the input bias reached ~0.8V, the resistance of the Ag channel suddenly increased from its $R_{initial}$ = 14.4 **Ω** to R_{formed} = ~10^{12}**Ω**. By comparing the optical images of the pre-formed (**Fig.4.2c**) and post-formed (**Fig.4.2d**) device, we noticed that the left part of the Ag channel became visibly darker shown in **Fig.4.2e**, likely due to the

formation of resistive silver oxide. This observation explains the forming-induced transition to HRS.

Thereafter, the device operates as a resistive switching memory. **Fig. 4.3b** shows the bipolar I-V characteristics of the Ag-CBRAM as measured from a DC double sweep cycle. Here, the applied voltage was increased in 5mV steps for the positive (0V to 2V) and negative (0V to -1V) voltage ramps while enforcing compliance currents of 500μA and 10mA to achieve SET and RESET respectively. To investigate the consistency of switching operations, we characterized the statistical distribution of switching voltages and device resistances by performing 50 DC switching experiments (Fig. 3c). The average switching voltage for the SET was ~1.77V whereas that for RESET was ~ -0.35V. **Figure 3d** records the average low resistance state (LRS) and HRS resistances of ~340 **Ω** and ~3×10^{13} **Ω** with good uniformity. These values translate to an ultra-high ~10^{10} ON/OFF ratio of the device which distinctly favors its flexibility in mapping a wide range of neural network weights[10]. It is noteworthy that the low resistance of the Ag-CBRAM promises low latency operation whereas the high HRS can help lower the static power consumption of the device by suppressing leakage currents.

Device reliability is essential for implementing network training and inference that requires frequent switching and long-term storage of the weights[10]. **Figure 4.4** shows that our device can retain the HRS and LRS states for more than 10^4 s at room temperature. In addition to being non-volatile with long retention, our Ag-CBRAM devices exhibit high endurance. Figure 4b shows that the device can be switched between LRS (~10^4**Ω**) and HRS (~10^{12}**Ω**) for at least 10^4 cycles without any observable degradation. The HRS and LRS over 10^4 cycle switching is presented in **Fig.4.4c**, which indicates our device maintains low variations in both states. These

results confirm the capability of Ag-CBRAM for achieving reliable crossbar operation with long-term stability.

While we demonstrated conventional memory application for Ag-CBRAM device, gradual resistance switching is a key to achieve higher storage density by mapping multi-bit weights to a single device. By controlling the current compliance levels from 100pA to 1mA, the device can reliably switch between 16 states spanning 7 orders of magnitude in resistance as shown in **Fig.4.5a**. Moreover, we characterized the mean (**Fig.4.5b**) and standard deviation (Fig.5c) of each distinct resistance level versus compliance current. Our results validate that the Ag-CBRAM device has multi-level programmability with minimal overlap between different levels.

3. Mott-ReLU Activation Neuron

We have previously developed array of Mott-ReLU neuron devices to implement activation function layer[8]. Each Mott ReLU neuron device has four terminals that allow exploiting of a thermal driven Mott transition of VO_2, which emulates ReLU activation function in a single device (**Fig. 4.6a**). Mott ReLU devices were fabricated by depositing 70nm VO_2 film via reactive sputtering. Then device switching area was defined by two Ti (20 nm)/Au (30 nm) electrodes with 50 nm gap using e-beam lithography and evaporation. Then, 70 nm Al_2O_3 was deposited as the electrical insulation layer. Lastly, a local heater was defined by patterning Ti (20 nm)/Au (30 nm) nanowire on the VO_2 gap using e-beam lithography and evaporation. A SEM image of a fabricated device is shown in **Fig.4.6b**. **Figure.4.6c** explains the operation of the device. The resistance of the heater is ~30 Ω and the initial resistance the VO_2 gap is ~10kΩ. ReLU activation function can be emulated by applying current bias to the nanowire that induce

thermal gradual resistivity switching on the VO_2 gap. The VO_2 gap formed a voltage divider circuit with a load resistor to generate voltage output (V_{OUT}) that can be directly fed as an input to the synaptic layer. By flowing current through the heater, the temperature of the VO_2 gap is precisely modulated to induce thermal-driven gradual resistive switching. As a result, the resistance change of the VO_2 gap modulates Vout through Vload and successfully emulates ReLU function as shown in **Fig. 4.6c**. The device shows precision higher than 4-bit. As shown in **Fig. 4.6d**, Mott-ReLU neurons show low latency of ~61.4 ns, while consuming 199.5pJ per operation.

To further understand how to achieve the optimal energy efficiency for our device, we developed an empirical thermal model in SPICE to project energy consumption of the device. This compact thermal model consists of Joule-heating model of the heater, thermal model of VO_2 and coupling model between heater and VO_2 gap (**Fig.4.7a**). **Figure.4.7b** lists model parameters and equations (4.1) and (4.2) govern the heater current and latency estimation in the model. By varying heater thermal resistance, our model indicates that heater current can be reduced by 3.4× when the thermal resistance of the nanowire heater increased by 10× (**Fig.4.7c**). Therefore, replacing the heater material with a higher thermal resistance material such as Ti can significantly improve thermal coupling and allow generated heat to be more confined within the VO_2 gap. In addition, the latency can be further reduced to ~3.8ns by minimizing the parasitic capacitance of the Mott ReLU below 10^{-11} F as shown in **Fig.4.7d**. As a result, our model estimates the energy consumption of Mott-ReLU neurons can be minimized down to ~0.638pJ at single device level by careful engineering the heater material to enhance the thermal coupling and reduce parasitic capacitance.

$$C_{th\_H}\frac{dT_H(t)}{dt} = \frac{(T(t)-T_H(t))}{Rox} + {I_{IN}}^2 R_H - \frac{(T_H(t)-T_0)}{R_{th\_H}} \qquad (4.1)$$

$$C_{th\_VO2}\frac{dT(t)}{dt} = \frac{(T_H(t)-T(t))}{Rox} + V_{VO2}I_{VO2} - \frac{(T(t)-T_0)}{R_{th\_VO2}} \quad (4.2)$$

Table1 summarizes the energy, latency, area and leakage performance of Mott ReLU activation devices against other activation devices or circuits at single ReLU level. Our Mott ReLU device can already achieve ~17× energy reduction compared with Analogue CMOS circuits [6]. With optimized thermal coupling, the device is projected to achieve ~30× energy reduction compared with digital ADC implementation[7]. The device can also provide 450-1500× improvement in area and 1.5-3× improvement in latency. These substantial performance gains of activation layers motivate our integration with Ag-CBRAM array that can achieve more efficient DNN implementation in hardware.

4. Ag-CBRAM and Mott-ReLU Integration for a DNN Application

To demonstrate core operations of DNN inference with our hardware, we focused on VGG-16 for CIFAR10 image classification task in a hardware. First, we designed a custom PCB to integrate the Ag-CBRAM crossbar arrays with Mott-ReLU arrays (**Fig.4.8a**). The board is capable of monitoring two arrays simultaneously and verifying weighted sum and activation results. We used 16×16 Ag-CBRAM crossbar for this demonstration (**Fig. 4.8b**). The callout window of Fig. 8b shows a representative device in the crossbar. **Figure. 4.8c** shows an array that contains 44 Mott-ReLU devices that can be individually connected to the BLs of the crossbar via PCB.

Then we investigated how to efficiently map VGG-16 to our hardware. VGG-16 is a convolutional neural network that is 16 layers deep, which was widely used for computer vision applications. **Figure4.9a** shows the representative CIFAR-10 images from 10 classes and network architecture. In VGG-16, there are 13 convolutional layers in which each layer is

followed by ReLU activation layers, 5 max pooling layers, and 3 fully connected layers in order. For the hardware demonstration, we focused on convolutional layers. Max pooling layers and fully connected layers were implemented in software.

Before we map full-precision (64-bit) weights in VGG-16 into hardware, we performed post-training uniform quantization of both weights and activation function with various precision and investigated its impact on inference accuracy. **Figure. 4.9b** shows that 5-bit weights precision and 4-bit activation precision are the minimal bit precision allowed to ensure there is no significant accuracy degradation. Although each Ag-CBRAM cell in our crossbar array has gradual resistive switching capabilities as shown in **Fig. 4.5a**, analogue approach requires custom peripheral neuron circuits to precisely vary current compliance and realize fine control of resistance levels. Therefore, we chose to use digital implementation for this array level demonstration to ensure better controllability of the resistance states.

Figure 4.9c explains the mapping of the network to the Ag-CBRAM crossbar arrays using binary weights (HRS~1012 Ω, LRS~20 kΩ). In this illustration, N of 3×3 convolutional filters are unrolled to N of 9×1 vectors and mapped to columns of the crossbar. We quantized the filter weights into 5-bit binary representation to minimize memory size while maintaining high accuracy. As a result, five columns are used to represent MSB to LSB of the weights. As the filter slides across the input image, the part of the input (W×W) overlaps with the filter is also unrolled to a 9×1 vector and feed into the WLs of the crossbar. The crossbar performs MVM and the weighted sum current is accumulated at the end of each column. Activation layers are implemented by connecting a Mott-ReLU to each column. Mott-ReLU neurons rectify weighted sum and produce final pixel values in output feature maps (OFM).

Before implementing CIFAR-10 classification task in our hardware, we first tested whether Ag-CBRAM crossbars can drive Mott-ReLU neurons (**Fig. 4.10a**). We varied the input voltage (V_{in}) to a column of the CBRAM array by sweeping it from –250 mV to 250 mV when ~2/3 of devices on a column of the CBRAM array are set to a LRS while the others are set to HRS (**Fig. 4.10b**). Moreover, we varied the number of LRS in the column of Ag-CBRAM array from 0%-100% (**Fig. 4.10c**). For both cases, 1.1 V is applied as V_{DD} to the VO_2 gap of Mott-ReLU with a 3.3-kΩ-load resistor connected in series, and 7 mA of offset current is applied to the heater. As can be seen in **Fig. 4.10b** and **c**, the Mott-ReLU neuron shows ReLU inputoutput characteristics.

After verifying Mott-ReLU neuron can be driven by Ag CBRAM crossbar, we then converted CIFAR10 images into 8-bit pulse trains and fed into the crossbar to generate weighted sum current, I_{sum}, which is a result of application of convolution filters to the images. Mott-ReLUs rectify I_{sum} and generate ReLU output (V_{out}). **Figure 4.11** shows representative experimental results from network operations performed for first (layer1) and last (layer13) convolution layer of VGG-16 on a 32×32 input image from the dog class. For each layer, we presented both software (SW) simulated result and measured result in hardware (HW) side by side for comparisons. **Figure 4.11a** and **g** shows 3×3 quantized convolution filters. After mapping these filters using the approach described previously to the Ag-CBRAM array, Isum is measured at the end of each BLs. **Figure 4.11b** and **h** show measured Isum in real time for 4 representative patches (each patch contains 5×5 pixels) highlighted as red boxes in **Fig. 4.11c** and **i**. Slide number represents the position of the filter as it slides across each patch. Isum from each BLs drives individual Mott activation neurons in the PCB and output voltage of the neuron device is shown in **Fig. 4.11d** and **j**. These results indicate that our hardware implementation of

convolution filters and activations can reliably generate OFMs and ReLU output without additional driver circuits and achieve close to ideal software results (**Fig. 4.11e**, **f**, **k**, and **l**). The learned OFMs represent abstract features of the dog class in layer 1 and 13. Based on the measured results in hardware, the estimated classification accuracy for the entire CIFAR-10 dataset using our hardware is 93.04%, approaching ideal software accuracy (~94%). Energy efficiency is estimated as 25.7 TOPS/W.

5. Conclusion

In this work, a direct integration of Ag-CBRAM array with Mott-ReLU activation neurons are successfully demonstrated in hardware. Our Ag-CBRAM device shows ultra-high ON/OFF ratio, low variation, reliable endurance and retention. In addition, Ag-CBRAM has multi-level switching capability with 16 states, making it an ideal synaptic device for neural network operation. The simplicity of fabrication for lateral Ag-CBRAM array makes it easy to be integrated with the back end of the line (BEOL) of CMOS chips. The four-terminal Mott-ReLU device embodies ReLU characteristics and can be directly driven by weighted sum currents generated in Ag CBRAM array. The small footprint of the device allows stacking in between synaptic layers for scalable in-memory computing system. The hardware demonstration shows that Ag CBRAM arrays integrated with Mott ReLU devices offer a compact and scalable solution for accelerating DNNs with close to software accuracy. Our approach opens new avenues in implementing deeper and more complex network architecture with higher area and energy efficiency using eNVM based synaptic arrays and Mott-ReLU activation devices.

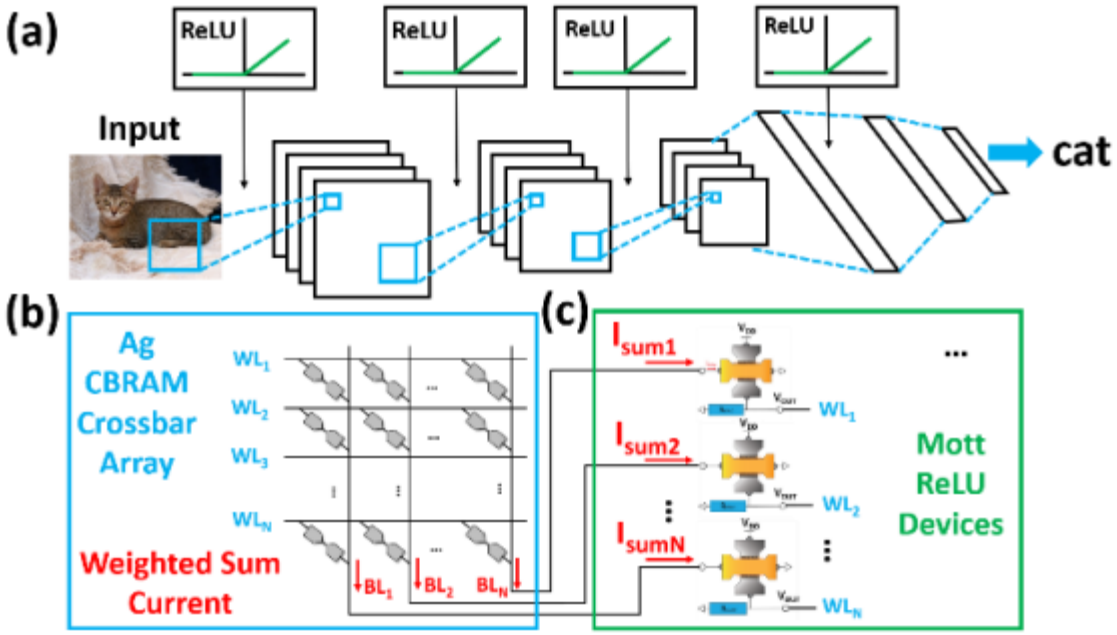


Figure 4.1: (a) Representative DNN consisting of convolutional layers and ReLU activations. ReLU activations are applied after each convolutional layer. (b) Convolutional layers are implemented with Ag-CBRAM crossbar arrays. Ag-CBRAM devices are arranged in a crossbar fashion with inputs feed into the WLs and weighted sum current accumulated in BLs. (c) ReLU activation layers are implemented with Mott-ReLU devices. The weighted sum current in BLs drive inputs of Mott devices and outputs of the devices are fed to the subsequent Ag-CBRAM layers.

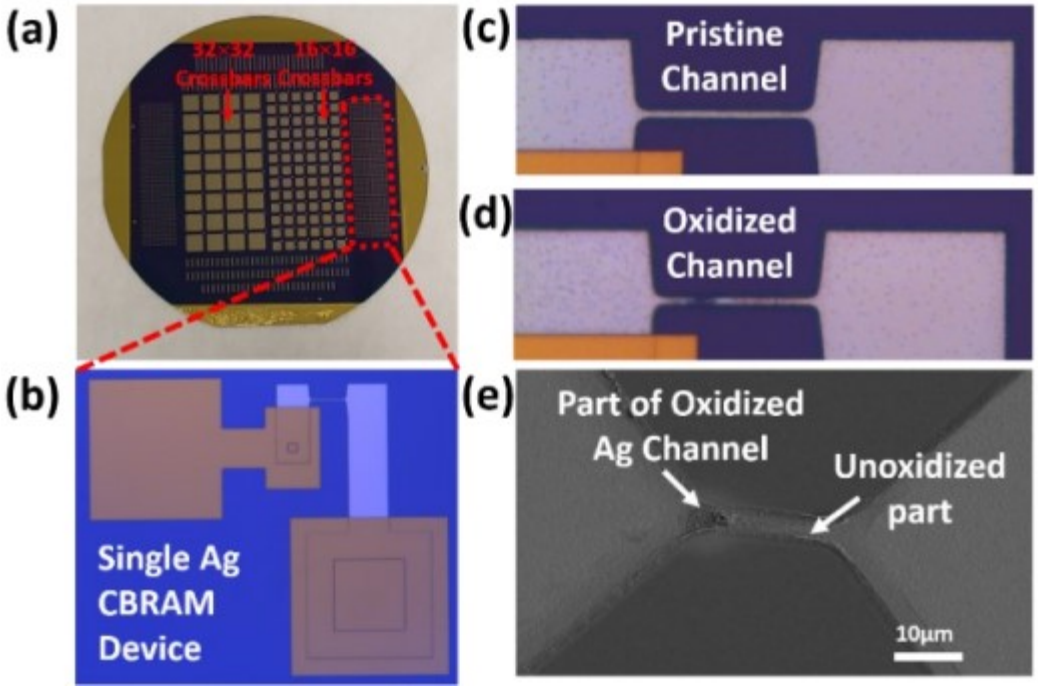


Figure 4.2: (a) Wafer-scale image of Ag-CBRAM crossbar arrays (16×16 and 32×32) and (b) single devices with 5μm×20μm channel. Microscope images of (c) Pristine and (d) Oxidized Ag channel. The lateral Ag-CBRAM devices have BL and WL pads defined with the narrow channel. (e) SEM for oxidized channel which shows darker color compared with unoxidized part.

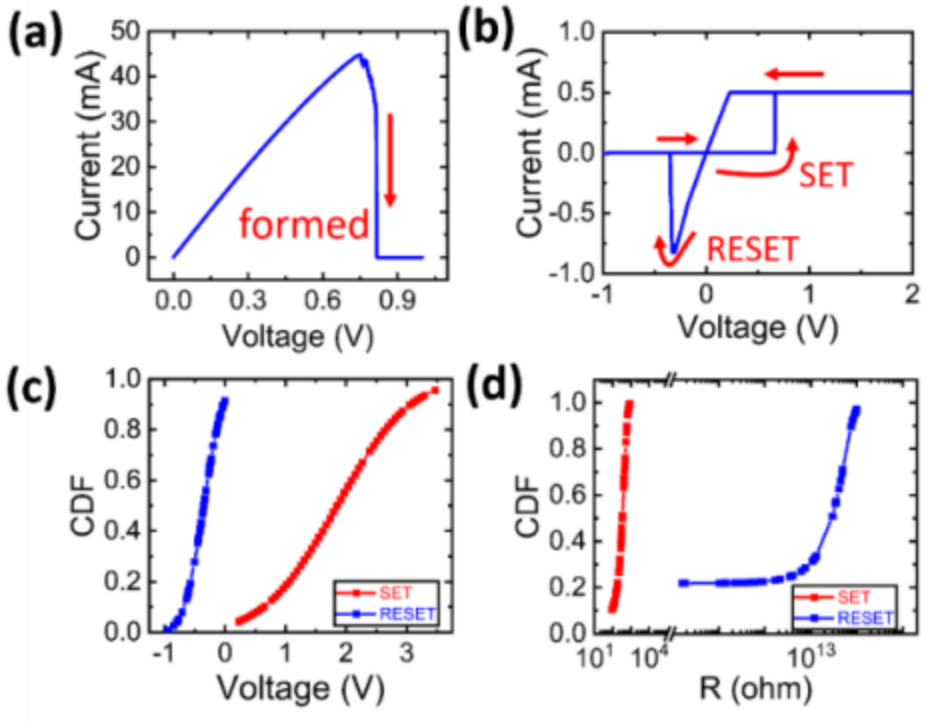


Figure 4.3: (a) Device initialization by applying a voltage sweep to oxidized pristine Ag-channel. (b) Bipolar I-V switching characteristic measured by a typical DC double sweep. Binary states are obtained. The cycle to cycle (C2C) variation of (c) switching voltages and (d) resistance represented using cumulative distribution function (CDF). C2C variations are obtained by applying 50 DC double sweeps to a Ag-CBRAM device.

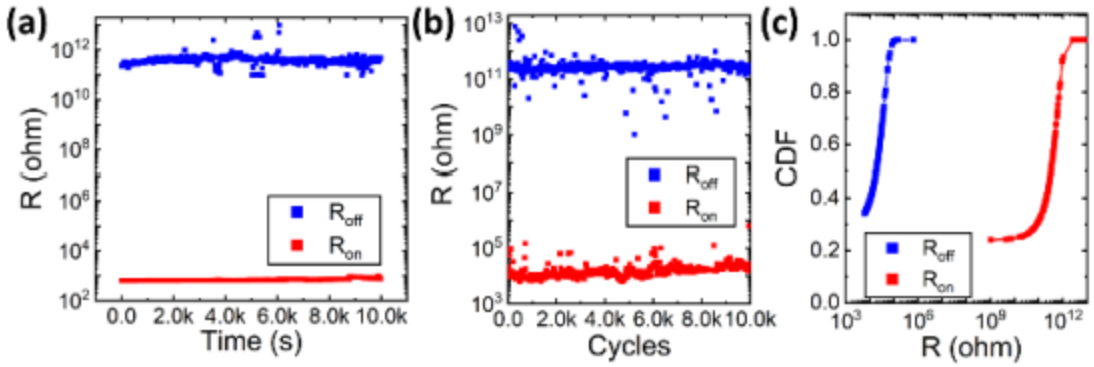


Figure 4.4: (a) Retention behavior of Ag-CBRAMs. The device is first SET to LRS. A sampling measurement that lasts 10^4 s is performed to constantly monitor the device resistance. The device is then switched to HRS and same measurement is performed. (b) Endurance of Ag-CBRAMs. 10^4 cycles are achieved by alternatively applying SET and RESET pulses to the device while monitoring the device resistance. During the test, the RESET and SET transitions were achieved using - 4V/100µs and 3V/5ms voltage pulses, respectively. (c) CDF of resistance in pulse programming, which is extracted from the endurance measurements in (b).

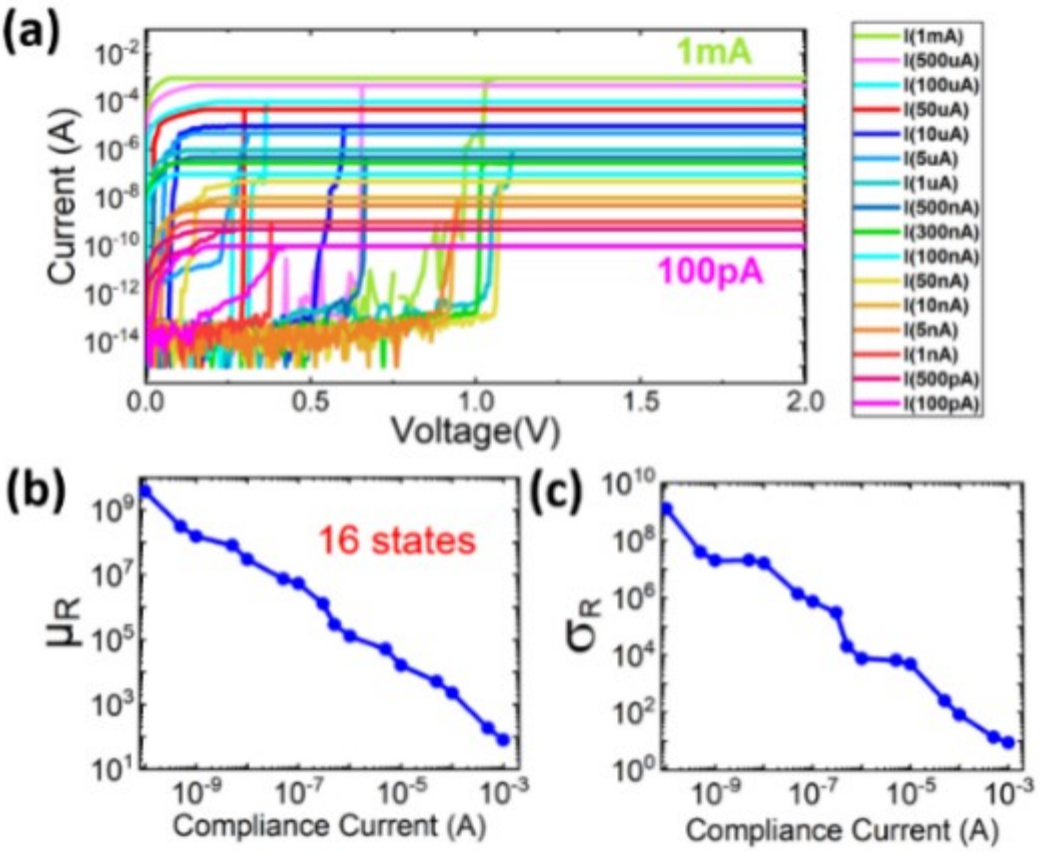


Figure 4.5: (a) Multi-level switching characteristics using different SET current compliance from 100pA to 1mA. The SET sweep is applied from 0V to 2V with 100mV steps for all compliance current conditions. (b) mean and (c) standard deviation of 16 (4-bit) resistance states as a function of compliance current.

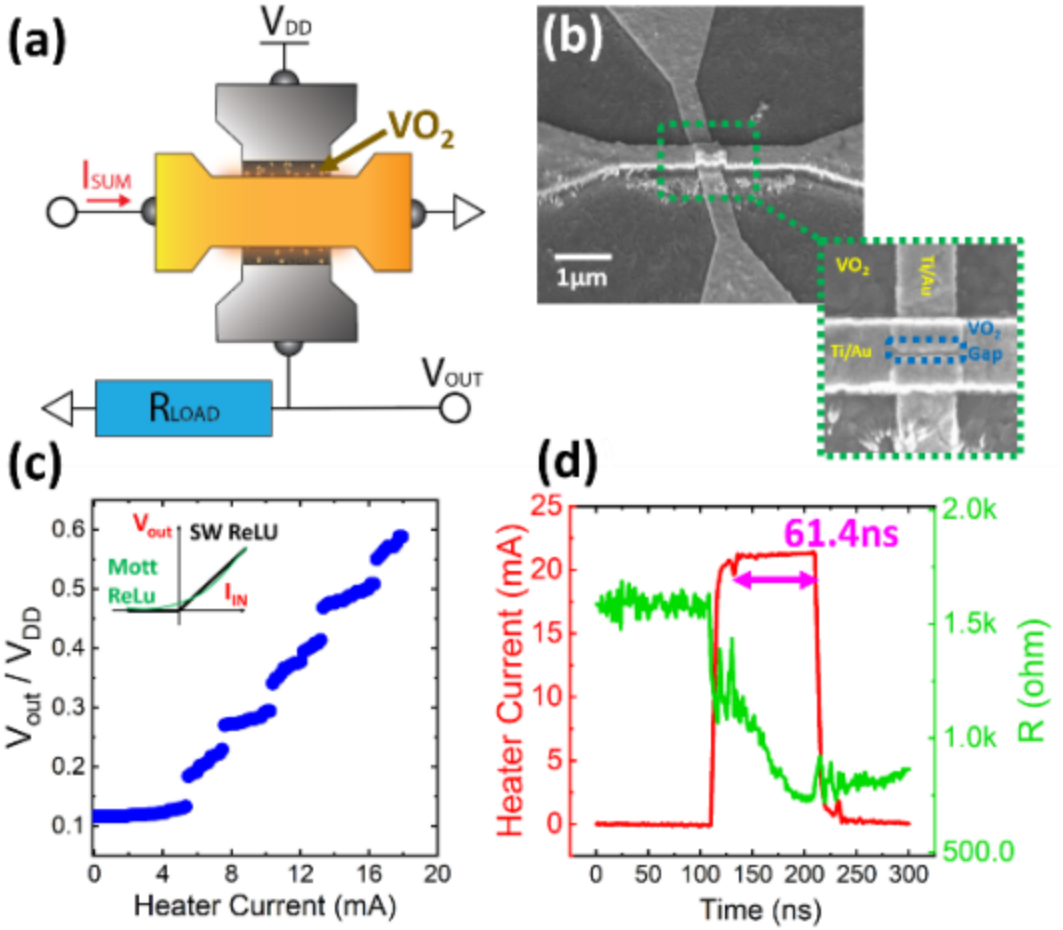
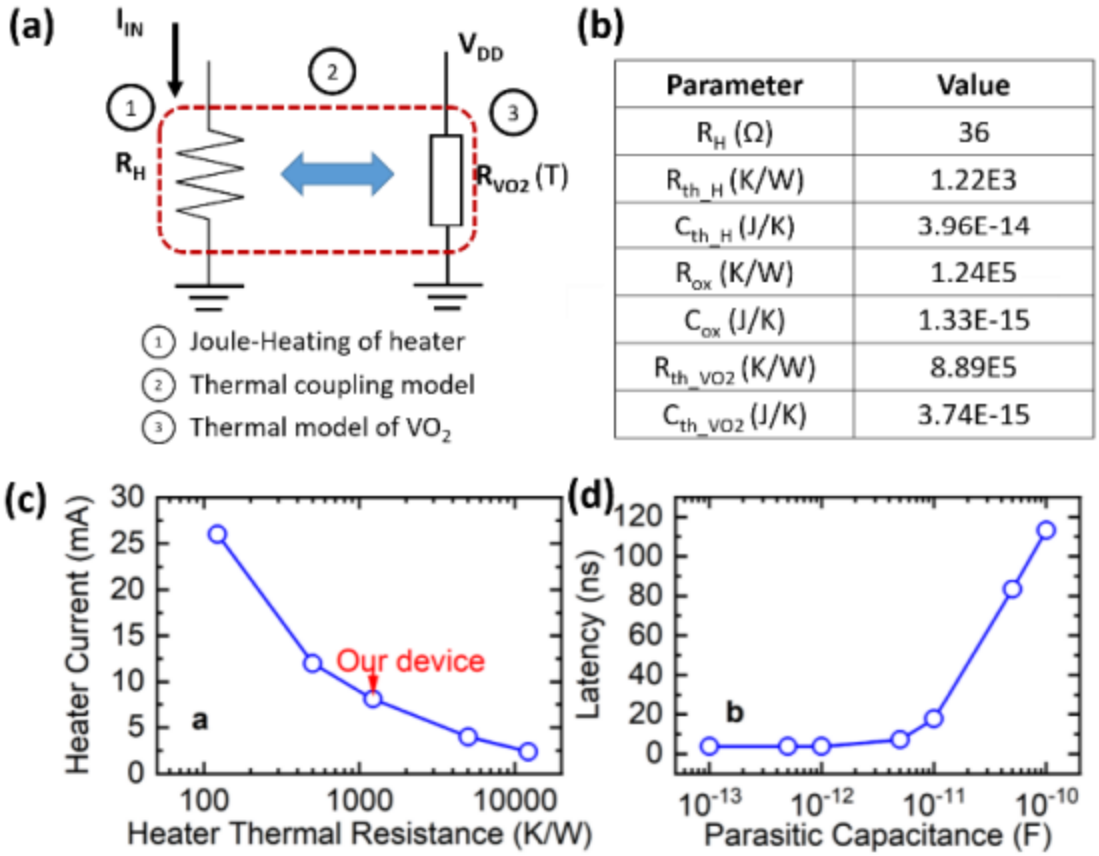


Figure 4.6: (a) Schematic and (b) SEM of Mott-ReLU device. The heater driven by weighted sum current (I_{sum}) enables the change of VO_2 resistance to emulate ReLU characteristics. (c) Measured Mott-ReLU activation output characteristic. The inset illustrates software ReLU in black in compare with Mott ReLU in green. (d) Pulse measurement for latency. Output of the Mott device becomes stable after input applied to heater for ~61.4 ns.



Parameter	Value
R_H (Ω)	36
R_{th_H} (K/W)	1.22E3
C_{th_H} (J/K)	3.96E-14
R_{ox} (K/W)	1.24E5
C_{ox} (J/K)	1.33E-15
R_{th_VO2} (K/W)	8.89E5
C_{th_VO2} (J/K)	3.74E-15

Figure 4.7: (a) A schematic shows the compact thermal model for the Mott ReLU device built in SPICE. The model has three parts: Joule-heating of heater, thermal coupling model for heater and VO_2 and thermal model of VO_2. (b) SPICE model parameters. (c) Heater current as heater thermal resistance of nanowire heater increases while keeping the resistance of the VO_2 gap to 1 kΩ. (d) Latency of the Mott ReLU device as parasitic capacitance increases. ~3.8ns latency can be achieved by reducing parasitic capacitance < 10^{-11}F.

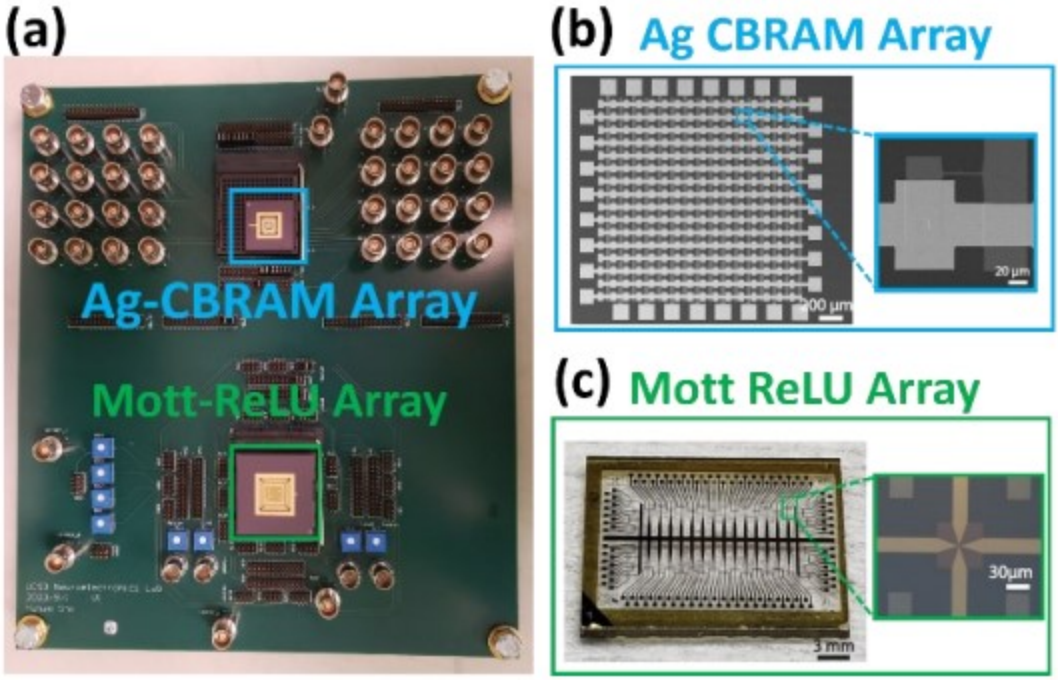


Figure 4.8: (a) Custom PCB integrating Ag-CBRAM crossbar (top) and Mott-ReLU array (bottom). (b) SEM image of 16×16 Ag-CBRAM crossbar array. Call out window shows single Ag-CBRAM device. (c) Mott-ReLU array including 44 MottReLU neurons. Call out window shows single Mott-ReLU device.

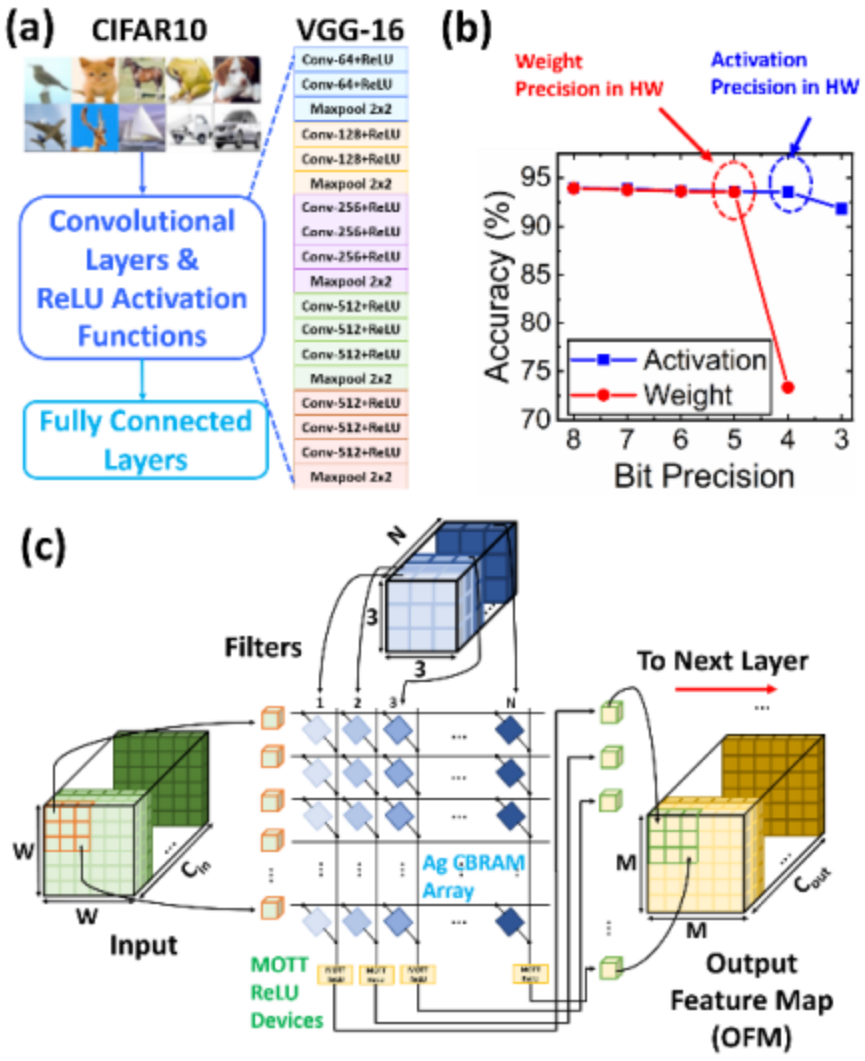


Figure 4.9: (a) VGG-16 network architecture for CIFAR-10 used for hardware implementation. (b) Post-training quantization using trained VGG-16 weights. 5-bit and 4-bit for weights and activation are use in hardware. (c) The network is mapped to the Ag-CBRAM crossbar by unrolling the filters. The weighted sum current in each BLs is fed into the Mott-ReLU neurons at the end of each column.

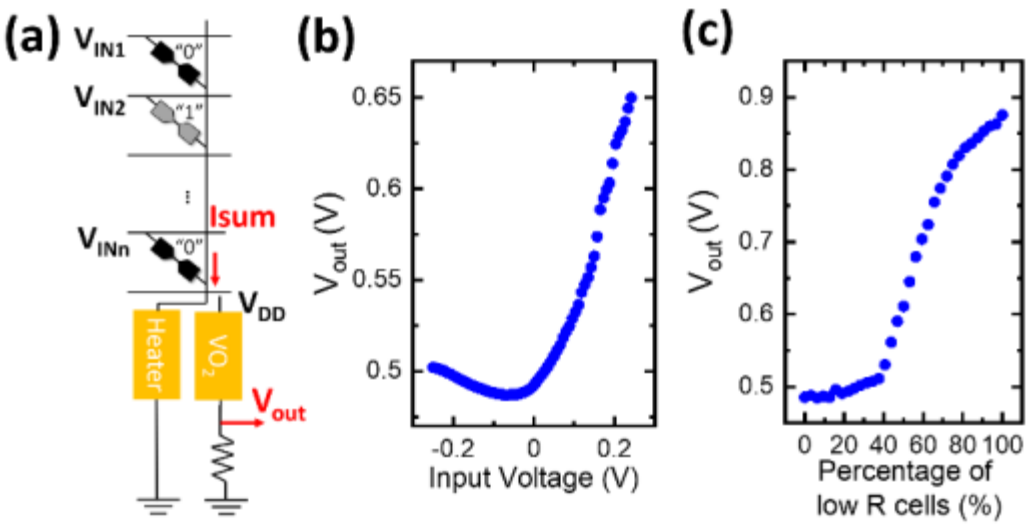


Figure 4.10: (a) A column of Ag-CBRAM array with Mott-ReLU neuron connected at the end. Ag-CBRAM colored in black represented LRS while colored in grey represented HRS. ReLU input-output characteristics measured using PCB by (b) varying the input to the crossbar from −0.25V to 0.25V or (c) changing the number of LRS cells while fixing input voltage to 130mV.

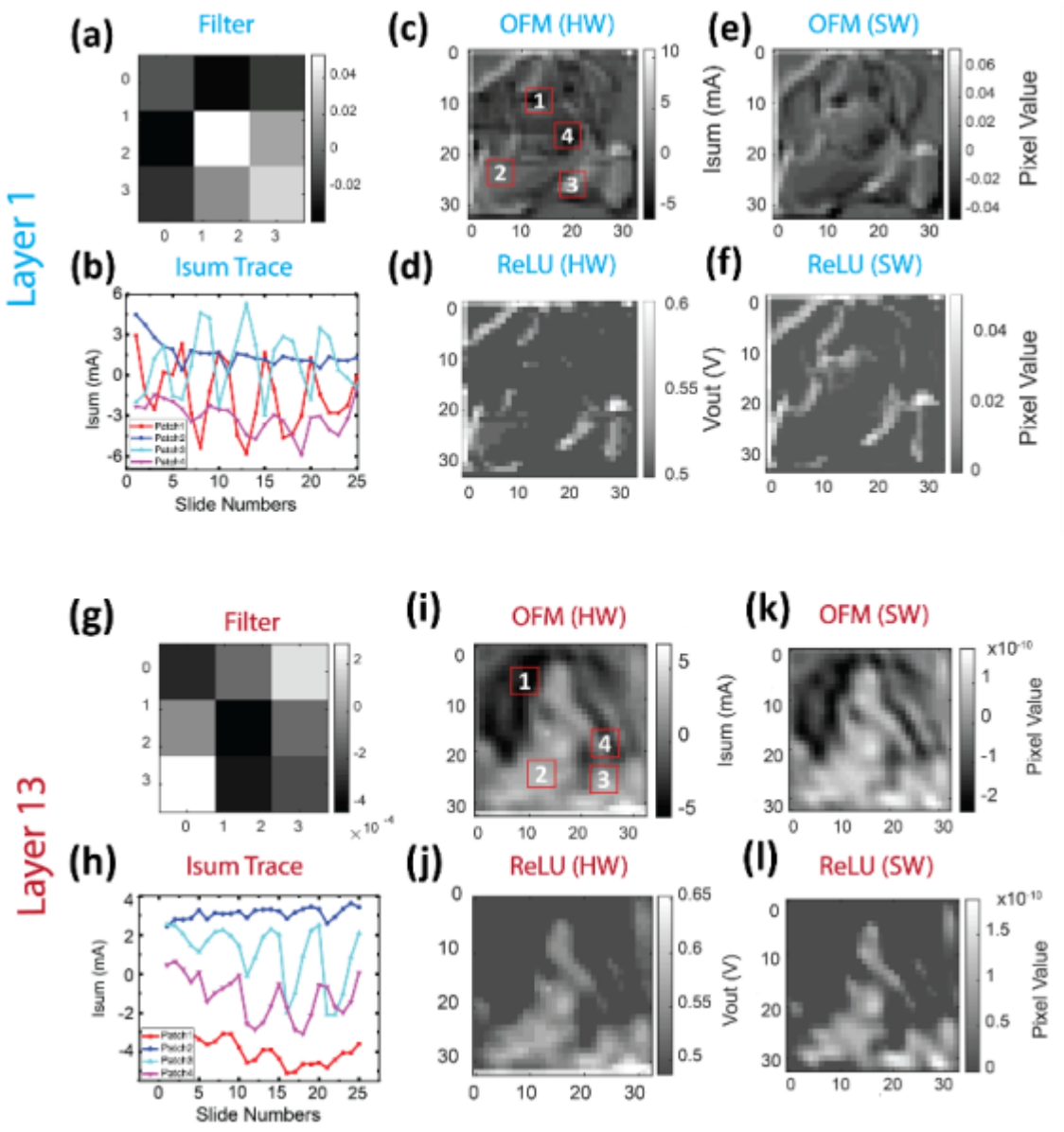


Figure 4.11: Hardware implementation of convolution and ReLU activation layers of VGG-16 (Fig. 9a) using Ag-CBRAM array and Mott-ReLU neurons on a CIFAR10 image. Representative results for layer 1 (a-f) and layer 13 (g-l) are shown. (a) and (g) are representative 3×3 convolutional filters in layer1 and layer13 respectively. The filter weights are quantized to 5-bit. (b) and (h) are measured weighted sum current trace from 4 patches shown in (c) and (i). (c) and (i) are OFMs obtained using Ag CBRAM synaptic array and in compared with OFMs generated in software ((d) and (j)). (e) and (k) are final OFMs after passing Mott-ReLU activation layers and in compared with ReLU results in software((f) and (l)).

Table 4.1: Performance Comparison of Activation Device/Circuit

Parameters	Mott	Analogue CMOS[6]	Digital ADC[7]
Energy (Exp./Optimal, pJ)	199.5/0.638*	3410	19.4
Latency (Exp./Optimal, ns)	61.4/3.8*	91.91	207
Area (μm2)	0.64	951.06	289**
Leakage (μW)	27.0	11060	-

*Shows projected optimal energy and latency when the thermal resistance of the heater is increased by 10× and the parasitic capacitance of a Mott ReLU is $< 10^{-11}$F.
**The area is only the area per neuron circuit.

6. References

1 LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. nature 521, 436-444 (2015).

2 Tan, H. H. & Lim, K. H. in 2019 7th international conference on smart computing & communications (ICSCC). 1-4 (IEEE).

3 Ielmini, D. & Wong, H.-S. P. In-memory computing with resistive switching devices. Nature electronics 1, 333-343 (2018).

4 Xue, C. J., Zhang, Y., Chen, Y., Sun, G., Yang, J. J. & Li, H. in Proceedings of the seventh IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis. 325-334.

5 Yang, T.-J. & Sze, V. in 2019 IEEE International Electron Devices Meeting (IEDM). 22.21. 21-22.21. 24 (IEEE).

6 Krestinskaya, O., Salama, K. N. & James, A. P. Learning in memristive neural network architectures using analog backpropagation circuits. IEEE Transactions on Circuits and Systems I: Regular Papers 66, 719-732 (2018).

7 Giordano, M., Cristiano, G., Ishibashi, K., Ambrogio, S., Tsai, H., Burr, G. W. & Narayanan, P. Analog-to-digital conversion with reconfigurable function mapping for neural networks activation function acceleration. IEEE Journal on Emerging and Selected Topics in Circuits and Systems 9, 367-376 (2019).

8 Oh, S., Shi, Y., Del Valle, J., Salev, P., Lu, Y., Huang, Z., Kalcheim, Y., Schuller, I. K. & Kuzum, D. Energy-efficient Mott activation neuron for full-hardware implementation of neural networks. Nature nanotechnology 16, 680-687 (2021).

9 Eckle, K. & Schmidt-Hieber, J. A comparison of deep networks with ReLU activation function and linear spline-type methods. Neural Networks 110, 232-242 (2019).

10 Zhang, Y., Huang, P., Gao, B., Kang, J. & Wu, H. Oxide-based filamentary RRAM for deep learning. Journal of Physics D: Applied Physics 54, 083002 (2020).

7. Acknowledgements

This work was supported by Office of Naval Research (N000142012405), the National Science Foundation (ECCS-1752241, ECCS-2024776), and the National Institutes of Health (DP2 EB030992). The fabrication of the devices was performed at the San Diego Nanotechnology Infrastructure (SDNI) of the University of California San Diego, supported by the National Science Foundation (ECCS-1542148).

Chapter 4, in full, has been submitted for publication of the material as it may appear in IOP Neuromorphic Computing, 2023, Shi, Yuhan; Oh, Sangheon; Valle, Javier del; Salev, Pavel; Schuller, Ivan K; Kuzum, Duygu, Integration of Ag-CBRAM Crossbars and Mott-Relu Neurons for Efficient Implementation of Deep Neural Networks in Hardware, 2023. The dissertation author was a co-author of this paper.

CHAPTER 5. A NEUROMORPHIC BRAIN INTERFACE BASED ON RRAM CROSSBAR ARRAYS FOR HIGH THROUGHPUT REAL-TIME SPIKE SORTING

Real-time spike sorting and processing are crucial for closed-loop brain–machine interfaces and neural prosthetics. Recent developments in high-density multielectrode arrays with hundreds of electrodes have enabled simultaneous recordings of spikes from a large number of neurons. However, the high channel count imposes stringent demands on real-time spike sorting hardware (HW) regarding data transmission bandwidth and computation complexity. Thus, it is necessary to develop a specialized real-time HW that can sort neural spikes on the fly with high throughputs while consuming minimal power. Here, we present a real-time, low latency spike sorting processor that utilizes high-density CuOx resistive crossbars to implement in-memory spike sorting in a massively parallel manner. We developed a fabrication process that is compatible with CMOS back end of line (BEOL) integration. We extensively characterized switching characteristics and statistical variations of the CuOx memory devices. In order to implement spike sorting with crossbar arrays, we developed a template matching-based spike sorting algorithm that can be directly mapped onto resistive random-access memory (RRAM) crossbars. By using synthetic and in vivo recordings of extracellular spikes, we experimentally demonstrated energy-efficient spike sorting with high accuracy. Our neuromorphic interface offers substantial improvements in area (∼1000× less area), power (∼200× less power), and latency (4.8μs latency for sorting 100 channels) for real-time spike sorting compared to other HW implementations based on field-programmable gate arrays (FPGAs) and microcontrollers.

1. Introduction

Extracellular recordings of neuronal spikes using microelectrode arrays have been widely used in studying neural circuits involved in sensory [1], motor [2], and navigation [3] functions in the brain [4]. The recorded signals are a mix of activities from multiple neurons and a crucial processing step, called spike sorting, is required to separate the firing activities and assign the recorded spikes to individual neurons from the recordings. Spike sorting is an indispensable tool in neuroscience for studying neural circuit[5], connectivity, causality, and decoding brain activities[6,7]. It is also fundamental in decoding intentions from neural activity in brain–machine interfaces (BMIs) [8] and neural prosthetics [9]. Conventionally, spike sorting is performed offline by transmitting raw digitized signals recorded by neural electrodes to a nearby computer. However, the off-line processing approach becomes impractical for sorting neural recordings generated from advanced high-density microelectrode arrays (HDMEAs) that comprise hundreds or thousands of recording sites in a single probe, such as recently developed Neuropixels probe [10]. Transmitting vast amounts of neural recording data from HDMEAs to an off-line spike sorter leads to excessive power dissipation which poses a serious risk of damage for the surrounding tissues[11].. For example, a 100-channel microelectrode array with a 16-bit analog-to-digital converter (ADC) operating at 30 kHz sampling frequency generates 3 MSamples/s and dissipates milliwatt-level power to nearby tissues. More importantly, to enable the closed-loop BMIs for prosthetics with multiple degrees of freedom, hundreds of neurons distributed in multiple cortical areas need to be monitored in real-time with minimal delay[12]. An 8-h recording experiment using a 100-channel microelectrode array would accumulate ~200 GB of data [13], demanding at least a few hours to sort the recorded spikes off-line [14]. The high latency associated with spike sorting becomes a limiting factor for closed-loop applications requiring rapid feedback.

These drawbacks highlight the need for developing compact, low-power, and high throughput hardware (HW) that can be integrated with high-density implantable microelectrode arrays to perform on-chip spike sorting in real-time. Although there have been sustained efforts to develop real-time spike sorting in field-programmable gate arrays (FPGAs), most implementations are inefficient in terms of area and power consumption. Want et al. demonstrated a single channel real-time spike sorting while using >90% FPGA resources[15]. Schäffer et al.[16] implemented the "Osort" algorithm in FPGA for sorting 128 recording channels, using hundreds of block RAM and DSP units. However, this approach does not scale well with channel count[16]. On the other hand, resistive random-access memory (RRAM) has been considered as a promising next-generation memory technology due to its low switching energy, nonvolatility, high switching speed, and small footprint[17]. In-memory computing based on RRAM arrays has been widely used in accelerating data-intensive applications such as neural network inferences, computer vision, and compressed sensing [18]. A crossbar array consisting of thousands of RRAM devices offers large nonvolatile memory storage and facilitates massive parallelization of matrix-vector multiplications. These advantages make RRAM crossbars uniquely poised to implement a large number of dot products in real-time with high energy efficiencies. However, to the best of our knowledge, no studies have yet shown RRAM-based brain interfaces for real-time spike sorting. In this article, we designed a compact, energy-efficient, and high throughput neuromorphic brain interface based on CuO_x crossbar arrays that can perform spike sorting for extracellular neural recordings. On the HW front, we developed a low-temperature fabrication process that is compatible with back end of line (BEOL) CMOS integration to fabricate high-density CuO_x crossbars. We developed a template matching-based spike sorting algorithm that is HW-friendly and scalable for mapping onto crossbars. In our

neuromorphic brain interface, low amplitude neural signals (a few microvolts) from an implanted neural probe were amplified and digitized using an Intan amplifier. The neural templates were encoded into device conductances and stored in columns of CuO_x crossbars (**Fig. 5.1**). Template matching was achieved by feeding neural signals to the wordlines (WLs) and using the crossbar architecture to compute their dot products with corresponding neural templates in each column. The sorting results were obtained parallelly by processing the weighted sum currents in the bitlines (BLs). We experimentally demonstrated the ability of our CuO_x crossbar arrays to sort simulated synthetic spikes as well as extracellular recordings from in vivo animal experiments with high accuracy, i.e., close to ideal software (SW) implementation. Based on experimental results, we also performed a system-level simulation and estimated that our approach can sort 100-channel recordings within 4.8 μs with ∼1000× reduction in chip area, ∼200× reduction in power, and ∼50× less energy per channel compared to the state-of-the-art FPGA and microcontroller implementations. The rest of this article is organized as follows. Section 2 presents device characterization results for CuO_x devices, including DC switching, transient pulse responses, cycle-to-cycle and device-to-device variations, and retention. Section 3 describes the template matching algorithm and two datasets used in the HW demonstration. Section 4 explains how the algorithm is mapped to the HW and the spike sorting in the crossbar. The system-level benchmarking results of our approach in comparison to other HW implementations are also discussed. Section 5 summarizes this article.

2. CuO_x Resistive Crossbars

We developed a wafer-scale process for fabricating 16 × 16 crossbar arrays of Au/CuO_x /Au resistive switching devices [**Fig. 5.2(a)**]. The SEM image of the crossbar array and the cross

section schematic are shown in **Fig. 5.2(b)** and **(d)**. The fabrication flow is illustrated in **Fig. 5.2(c)**. First, Au with Cr adhesion layer (100 nm) is sputtered and patterned via photolithography and lift-off for bottom electrodes (or WLs) with 1 μm linewidth and a 2-μm pitch. Then, 70 nm of CuO_x switching layer is deposited and patterned with reactive sputtering of Cu and Ar/O_2 (95%/5%) gas. After that, top electrodes are deposited and patterned following the same fabrication steps as the bottom electrodes. Last, 300 nm of SiO2 layer is deposited and patterned to passivate the device active region to ensure long-term stability. Since all the processes for the CuO_x crossbars are low-temperature process, it can be built directly on the BEOL of CMOS circuits. After fabricating $Au/CuO_x/Au$ resistive switching devices, we extensively characterized them (**Fig. 5.3** and **5.4**). The $Au/CuO_x/Au$ devices displayed consistent bipolar switching in response to 30 dc voltage sweeps [Fig. 3(a)]. They could be set to a low resistance state (LRS) of ∼100 at VSET = ∼ 1.5 V whereas applying VRESET = ∼−0.7 V increased device resistances to as high as ∼1 G with low cycle-to-cycle variations [Fig. 3(b)]. The high ON-/OFF-ratio ($\sim 10^7$) of the device resistances [**Fig. 5.3(c)**] provides a sufficiently large window for implementing the neuromorphic brain interface. Furthermore, the relatively low SET and RESET voltages [**Fig. 5.3(b)**] are desirable for future integration with peripheral CMOS circuitry.

Low device-to-device variations are important to ensure accurate mapping templates to the crossbar. To quantify this, we randomly selected 120 Au/CuO_x /Au devices from different regions of the wafer. Cumulative distribution function (CDF) of switching voltage and resistance is shown in **Fig. 5.4(a) and (b)**, respectively. The measured SET and RESET latencies are presented in our previous work[19]. The RESET transition (∼80 μs) was significantly faster than the SET process, highlighting the scope for further device optimization. Nonvolatility of LRS and high resistance state (HRS) was characterized by reading the device (V_{read} = 0.1 V) at regular

time intervals immediately after a successful SET or RESET process. The Au/CuO_x/Au devices could retain their LRS and HRS for >10,000 s, indicating these devices can faithfully store the neuron templates needed for real-time spike sorting and periodic refresh operations could be utilized if experiments take longer than this time period [**Fig. 5.4(c)**].

3. Template Matching Algorithm

A. Algorithm Overview

Spike sorting is a challenging clustering problem and many algorithms have been developed over the past years such as principal component analysis[20], template matching[21], Bayesian statistical frameworks[22], and hidden Markov models[23]. Among these, template matching is the most efficient approach to sort neural spikes[24]. It assumes a preexisting database of neuron templates; the goal is to assign the best-fit templates to the detected spike waveform, hence, clustering the spikes to specific neuron units. Motivated by this, we developed a template matching algorithm that can be directly mapped to the crossbars to achieve real-time spike sorting. **Figure 5.5** outlines the algorithm [**Fig. 5.5(a)–(d)**] by showing a simplified example for classifying two neurons (n = 2) from three-channel recordings (m = 3). The same methodology can be used to classify a larger number of neurons recorded across hundreds of channels. Each neuron had a template matrix $T_n = [T_{n,1}, T_{n,2}, \ldots, T_{n,m}]$, where column $T_{i,j}$ represented the template for neuron i corresponding to channel j [**Fig. 5.5(a)**]. The $T_{i,j}$ is a vector with S samples with S = fs × k, where fs is the sampling frequency and k is the user-define window that determines the duration of templates. In this example, fs = 30 kHz and k = 3 ms. Tn is built by horizontally concatenating these templates across m electrodes (m = 3). The template matrix Tn was normalized by its Frobenius Norm ($T_n/\|T_n\|_F$) to maintain the amplitude of the spikes in the

same range [**Fig. 5.5(a)**]. Similarly, we defined the neural signal $V(t) = [V_1(t), V_2(t)\ldots, V_m(t)]$, where $V_j(t)$ is the recorded signal from channel j. **Figure 5(b)** shows an example of recording in three channels at 30 kHz. To perform the template matching, we first computed the waveform similarity $C_{n,m}(t)$, which is the convolution between signals from channel m and the template of neuron n on channel m measured at time t. The convolution can be expressed as $C_{n,m}(t)=V_m(t)*T_{n,m}$, which is simply a sliding dot product between the signal and template. Then, the resulting waveform similarities from all m channels were summed up for each neuron [**Fig. 5.5(c)**] to give $C_n(t) = \sum_1^m C_{n,m}(t)$,, the overall activation of neuron n at time t. In the final step [**Fig. 5.5(d)**], we applied a threshold, which is ~3 standard deviation of the $C_n(t)$ to identify the spike times. After that, we assigned the spikes to the neuron having the largest $C_n(t)$. Note that these templates are typically obtained offline through a semiautomatic algorithm with human curation to ensure accuracy. The details of mapping templates to the HW are discussed in Section 4.

B. Datasets

We implemented the aforementioned template matching algorithm on two neural recordings: 1) synthetic "NeuroNexus-32" data [25] and 2) "real" spikes from in vivo animal experiments recorded with the NeuroFITM probe[6] for validating our spiking sorting HW with different neural electrode technologies. In the "NeuroNexus-32" dataset, the extracellular spiking activities with ground truth were generated using MEArec [25]. MEArec generated data in two phases. In the template generation phase, biophysically realistic neuron models were positioned at different locations of the NeuroNexus-32 probe model to produce extracellular potentials to form a template library. In the recording generation phase, it convolved the templates selected from the library with randomly generated spike trains. Additive Gaussian noise was added to the

convolution results to obtain the final recording data. Typically, a channel can record activities of ∼1–3 neurons nearby. Our synthetic dataset contains extracellular recordings of 12 neurons from 32 channels sampled at 30 kHz[19].. The "real" dataset contains 1 h recordings sampled at 32 kHz from an in vivo animal experiment recorded with the 32-channel NeuroFITM probe [**Fig. 5.6(a)**] [6], where spike sorting results from offline Kilosort algorithm[14] were considered as the ground truth. **Figure 5.6(b)** and **(c)** shows representative neuron templates and the recordings in Ch4. Top of **Fig. 5.6(c)** shows the predicted spike train as square symbols and the clustered neuron spike waveforms are presented in **Fig. 5.6(d)**. As can be seen, for each neuron, the shape of the clustered spike waveforms closely matched their respective templates. A similar waveform example of NeuroNeuxus-32 and the complete template libraries of both probes can be found in our previous work [19].

C. Sorting Performance

The sorting outcome of our algorithm is determined against the ground truth spikes by comparing the spike time. To quantify the sorting performance, we employed the commonly used F1 score (in percentage) given by 2TP/(2TP + FP + FN), where TP, FP, and FN denote the true-positive, false-positive, and false-negative outcomes. A TP is defined as a spike that has been classified correctly by the algorithm. An FP is defined as a spike that is classified as spiking activity but does not exist in ground truth data. An FN is defined as a spike that exists in the ground truth data but is not detected by our algorithm. The spike predictions from our algorithm agree with the ground truth well. Eleven out of 12 neurons in the NeuroNexus-32 dataset have F1 score >90% [**Fig. 5.7(a)**], whereas all the two neurons in the NeuroFITM "real" dataset have F1 score >85% [**Fig. 5.7(b)**]. The F1 score of the "real" dataset is slightly less than the synthetic dataset due to higher noise and probe drifting[26] during the recording, making the classification

more difficult. To map the templates to the HW, we investigated how quantization impacts the F1 score. The template was quantized to 2N discrete levels between the min and max amplitude range of the normalized template library. After quantization, we followed the same sorting pipeline to obtain the F1 score. **Fig. 5.7(c)** shows that the performance could be retained if the templates are quantized to at least four-bit resolution for NeuroNeuxus-32 dataset, which is also applied for NeuroFITM dataset.

4. HW Implementation of Spike Sorting

A. HW Mapping

To process hundreds of spikes per second, it would be necessary to adopt a multicore architecture [**Fig. 5.8(a)**] where each core consists of a crossbar that stores the templates for a specific set of neurons [**Fig. 5.8(b)**]. **Figure 5.8(c)** illustrates how a set of templates could be mapped on to a crossbar core. In the illustration, we assume that three channels (m = 3) record spike activities of two neurons (n = 2), resulting in a total of six templates. The templates from the same channel are mapped to the adjacent columns in the crossbar. The devices in the crossbar can store the templates using multilevel for analog implementation or binary (HRS or LRS) conductance states for digital implementation [27]. A column of devices with 16 (4-bit) multilevel can be used to map a template directly as shown in this example (i.e., templates of N1–Ch1 and N2–Ch2 are mapped to the first two columns of the crossbar, respectively). Similarly, templates from other channels are mapped to the rest of the columns to achieve the maximum usage of the array [**Fig. 5.8(c)**]. If binary conductance state is used, four columns are required to map a template from MSB to LSB. Although devices with multilevel states can achieve maximum area efficiency, it has been shown that these multilevel states may exhibit high device-to-device

variations, nonlinearity, and resistance drift due to unstable filament formation[18]. In contrast, digital implementation is more robust against variations[28], which makes it a better approach to realize high sorting accuracy for template matching tasks. In addition to conductance states, differential pair scheme is commonly used to represent both negative and positive values of the templates[29].

After all templates are mapped on a core, the voltage spike inputs on WLs (V_{WLi}) are convolved with the templates stored as cross point conductances (G_{ij}). The columns of the crossbar can perform template matching (BL currents $I_{BLj}=\sum G_{ij}V_{WLi}$) in parallel. Since a set of templates from each channel need to convolve with neural signals from the corresponding channel, recordings from Ch1 to Ch3 are processed in a time-multiplexed manner, the matching results ($I_{BLn,j}$) for each channel are collected from the corresponding BLs in parallel (n: neuron; j: channel number). The final classification result is obtained by adding the BL currents for each neuron, i.e., $I_n = \sum_1^m I_{BLn,j}$ from all m channels and then assigning the spike to the neuron with the maximum In. For the sake of illustration, we show all templates mapped to a single crossbar. For practical applications involving large channel counts, a multicore architecture can be adopted, where each core is dedicated to a channel and stores all templates belonging to the assigned channel. As a result, all channels can be processed at the same time to achieve higher parallelism.

B. HW Demonstration

A custom printed circuit board (PCB) was used to access the WLs and BLs of the wire-bonded CuO_x crossbar [**Fig. 5.9(a)** and **(b)**]. Before mapping the templates, array read was performed to confirm the initial states of the crossbar. To read a single device, the selected WL was biased to V_{read} = 0.25 V while all other lines were grounded. The as-fabricated devices had

initial resistances greater than 500 k [**Fig. 5.9(c)**]. As explained in Section IV-A. HW Mapping, digital implementation was adopted in our demonstration. Neuron templates were quantized, binarized, and mapped onto crossbar columns using differential pair scheme. To program the devices to different states, we used $V_{dd}/2$ write scheme, where the selected WL and BL were biased to $V_{dd}/2$ and $-V_{dd}/2$, and all other unselected lines were grounded to prevent sneak paths (SET: V_{dd} = 4 V and RESET: V_{dd} = 3 V).

Figure 5.9(d)–(f) shows four representative templates (F1–F4) of NeuroFITM implemented in the crossbar. The templates were quantized to four-bit and then binarized to two levels ("0"-black or "1"-white) off-line [**Fig. 5.9(d)**]. "0" was mapped to HRS and "1" was mapped to LRS of the device, respectively [**Fig. 5.9(e)**]. Since the crossbar was initially off, only "1" needs to be programmed accordingly. The patterns of the HW templates match well with SW templates, indicating precise write operation. To validate the accuracy of crossbar convolutions, we biased all WLs to high (VWLs = 0.25 V) and measured the BL currents. As shown in **Fig. 5.9(f)**, the weighted-sum BL currents (I_{sum}) increased proportionately with the number of LRS devices in the columns. Templates from NeuroNexus-32 dataset are mapped in the same way[19].

Using the programmed templates, we performed spike sorting on NeuroNexus-32 and NeuroFITM recordings. Neural data encoded as eight-bit voltage pulse trains were fed into the WLs and Isum were measured on the BLs. **Figure 5.10(a)** and **11(a)** show the NeuroNexus-32 and NeuroFITM recordings and the HW spike sorting results implemented to sort representative three neurons (N1–N3) from the NeuroNexus-32 data and two neurons (N1, N2) from the NeuroFITM data. The neural voltage traces from the recording channels (Ch1–Ch3 in NeuroNexus-32 and Ch1–Ch4 in NeuroFITM) are shown at the bottom. HW convolution trace generated by CuO_x crossbar represents final current $I_n = \sum_{j=1}^{m} I_{BL\,n,j}$ by adding weighted sum

currents measured in each $I_{BLn,j}$ for "m" channels and "n" neurons (NeuroNexus-32: m = 3, n = 3; NeuroFITM: m = 4, n = 2). The raster plots on the top of **Fig. 5.10(a)** and **11(a)** show the spike train predicted in HW compared with the ground truth spikes for Neuronexus-32 and NeuroFITM dataset, respectively. **Figure 5.10(b)** and **11(b)** show the callouts for the spikes highlighted in rectangular boxes [**Fig. 5.10(a)** and **11(a)**]. Inside the boxes, the snippet spike waveform of each neuron is shown in the left. Channels are coded in different colors that match with the signal traces above. The template matching results in SW and HW are shown as convolution traces in the middle (SW) and right (HW), respectively. Different colors represent N1–N3 of NeuroNexus-32 and N1–N2 of NeuroFITM. The SW convolution traces are shown as arbitrary units while HW traces are shown as measured weighted sum currents. For each spike, the neuron with the highest peak in the convolution trace was assigned to the spike. The shapes of convolution traces produced by the CuO_x crossbars matched closely with SW, thereby confirming our HW can reliably sort neural spikes. Note that the off-peak regions of the HW convolution traces are slightly noisy compared with SW mainly due to variations in the programmed device conductances across crossbar columns. This issue can be alleviated by adopting a more robust "program and verify" scheme[30].

C. System-Level Performance Benchmarking

Based on the HW spike sorting results obtained over a 100-ms time window (**Fig. 5.10** and **11**), we evaluated F1 scores on the entire 30 s-wide recordings in both neural data and compared them with SW predictions. The HW F1 scores were calculated by performing template matching between neural signals with HW templates that contain measured device resistances. To evaluate sorting performance across multiple neurons, we averaged F1 score based on neuron number. **Table 5.1** shows neurons could be sorted with high mean accuracy (~92.5% for

NeuroNexus-32, ~94.6% for NeuroFITM). To project the sorting performance of multicore architecture [**Fig. 5.8(a)**] with our crossbar-based spike sorting HW, we performed a system-level benchmarking to estimate area, power, and latency, and compared it with the state-of-the-art FPGA and microcontroller implementations. All implementations included in **Table 5.2** use in vivo experimental datasets and template matching based approach for a fair comparison. Our work and microcontroller implementation[31] demonstrated sorting for 32-channel probe while FPGA[15] implemented sorting for a single channel. The area per channel was estimated by the number of columns used in mapping a neuron template of a channel (i.e., ~8 columns are used for a channel template and it occupies 40 μm × 20 μm = 8×10^{-4} mm^2). Power per channel was calculated by averaging power consumption $P_{avg} = \sum_1^N I_{sum} \times V_{read}$ across a representative spike waveform snippet during template matching. Here, N is the number of samples in the spike waveform (N = 30), I_{sum} is the weighted sum current for processing a sample measured crossbar.

Overall, our crossbar-based spike sorting hardware promises ~1000× smaller (area/channel) [15] and ~200× reduction in power consumption [31] compared to state-of-the-art spike sorting HW implementations that rely on FPGAs (**Table 5.2**). To better understand the sorting latency in the multicore architecture, we assume one crossbar core can have size up to 256 × 256 and 10 ns read latency. Unlike previous works that rely on sequential processing, each crossbar core in the multicore architecture can process multiple recording channels in a highly parallelized manner. We estimated 12 CuO_x crossbar (256 × 256) cores can sort 100 channel recordings within 4.8 μs using the same mapping scheme of our HW demonstration. As a result, it consumes ~30–50× less energy (energy = power × latency) [15,31]. These performance gains make real-time spike sorting possible using our crossbars for high throughput BMI applications.

5. Conclusions

We presented a high throughput neuromorphic brain interface for real-time spike sorting based on resistive crossbar arrays. We fabricated CuO_x crossbars using a simple low temperature process enabling easy 3-D BEOL integration with underlying CMOS circuits. In order to realize real-time spike sorting, we developed an HW compatible template matching algorithm and developed methods for mapping onto crossbar arrays. We demonstrated that HW implementation of template matching using CuO_x crossbars can accurately classify spikes from individual neurons recorded in vivo. Our neuromorphic approach offers substantial performance gains in area, power, latency, and energy for spike sorting HW designed for processing recordings from neural probes with high channel counts. Our work paves the way toward in-memory computing-based real-time spike sorting and processing HW for next-generation closed-loop brain interfaces.

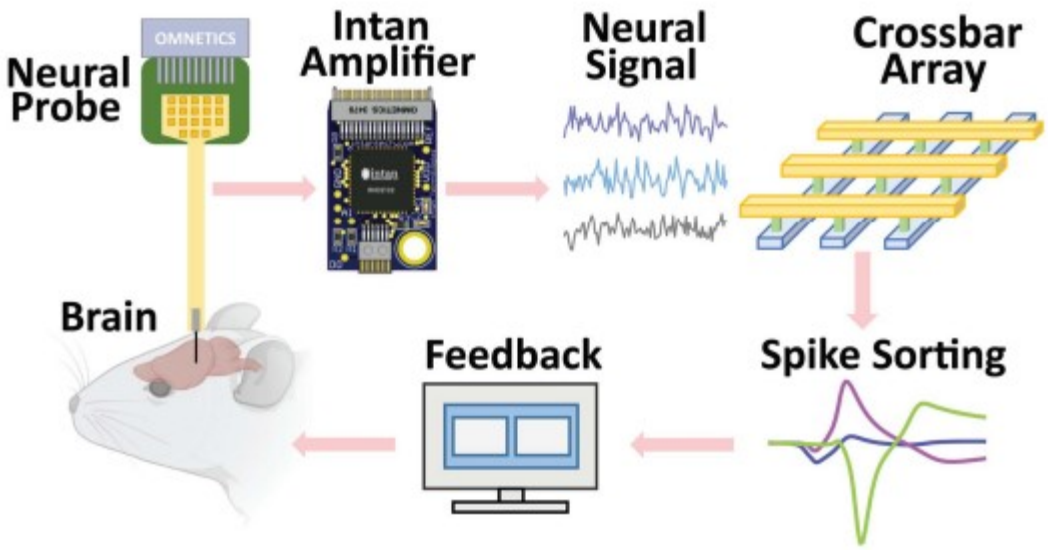


Figure 5.1: Proposed neuromorphic brain interface based on CuO_x crossbar array for spike sorting. Neural signals recorded by the multichannel neural probe are amplified and digitized using an Intan amplifier and ADC, respectively. CuO_x crossbar array performs spike sorting in real-time. That can be used as real-time feedback for a closed-loop neural interface.

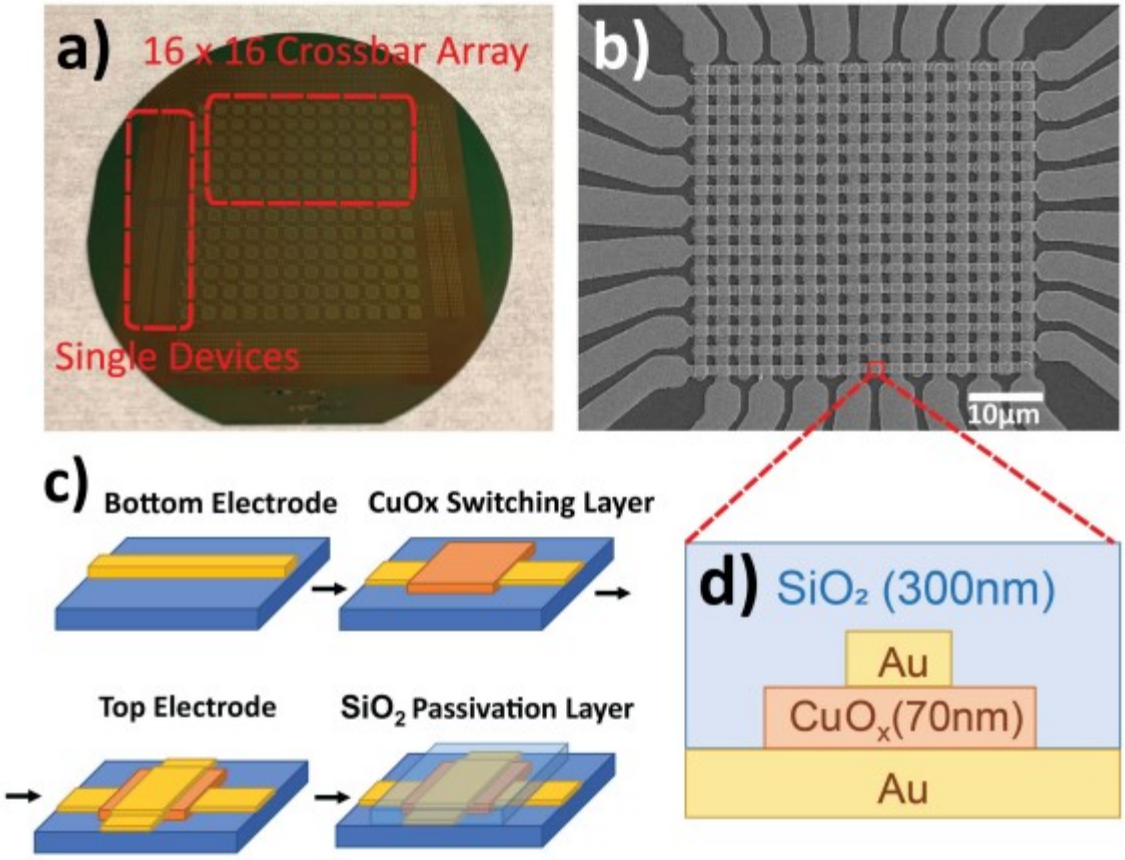


Figure 5.2: (a) Image of a wafer including fabricated 16 × 16 CuO_x crossbar arrays and single devices for testing. (b) SEM images of 16 × 16 crossbar with 4 μm^2 cross point. Scale bar: 10 µm. (c) Fabrication process for CuO_x-based single devices and 16 × 16 crossbar. (d) Device cross section (callout window) highlighting the 70-nm CuO_x resistive switching layer sandwiched between 100 nm Au electrodes. A 300-nm SiO_2 passivation layer is deposited on top of the stack.

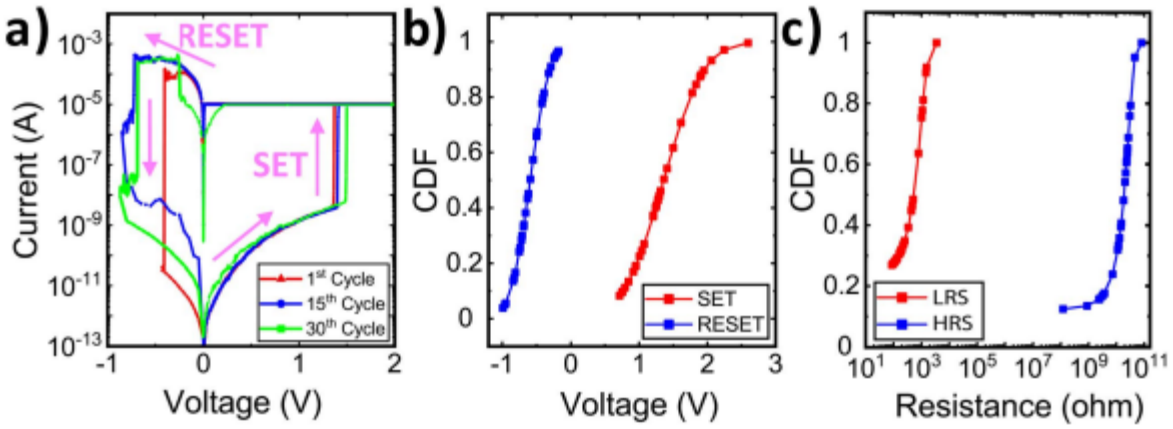


Figure 5.3: (a) DC switching characteristics of single devices for 30 cycles. (b) CDF of SET (1–2.5 V) and RESET (−1 to −0.2 V) voltages. (c) CDF of HRS (100 MΩ–100 GΩ) and LRS (100 Ω–1 kΩ) resistances.

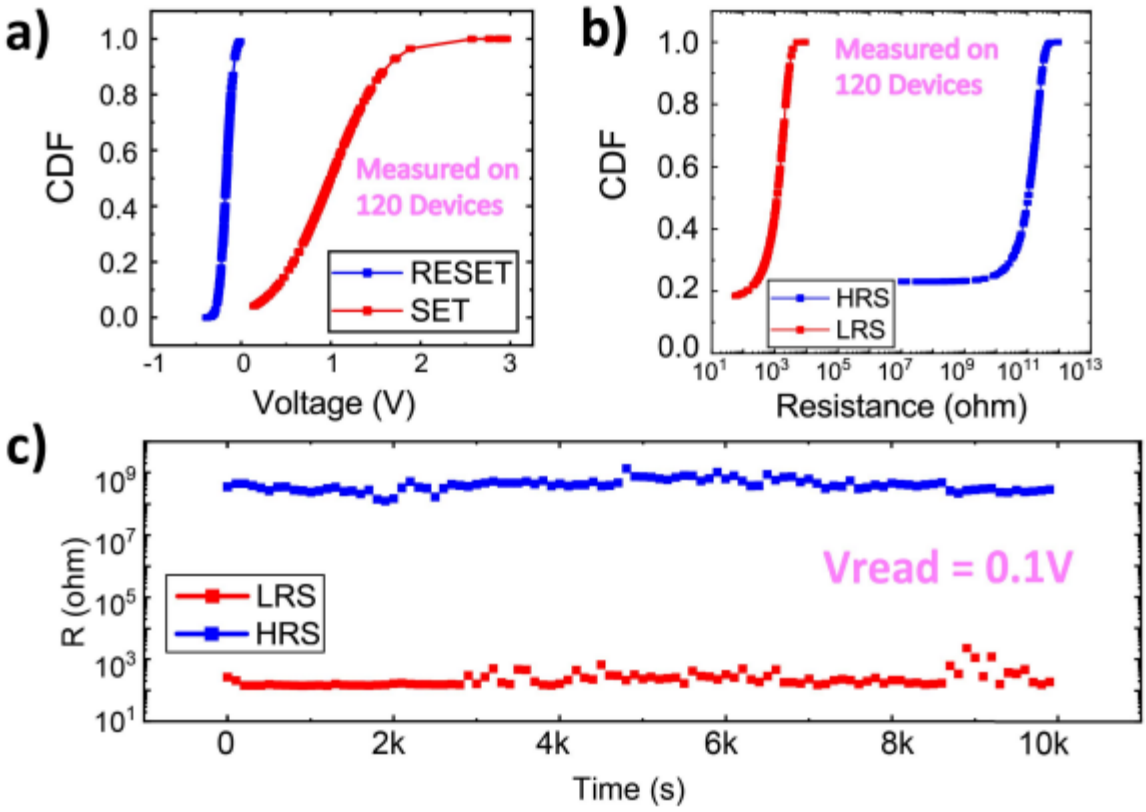


Figure 5.4: CDF of (a) switching voltages and (b) HRS/LRS resistances were measured across 120 devices randomly selected on the wafer. (c) Retention characteristics. Device resistance was monitored intermittently using 0.1 V read pulses.

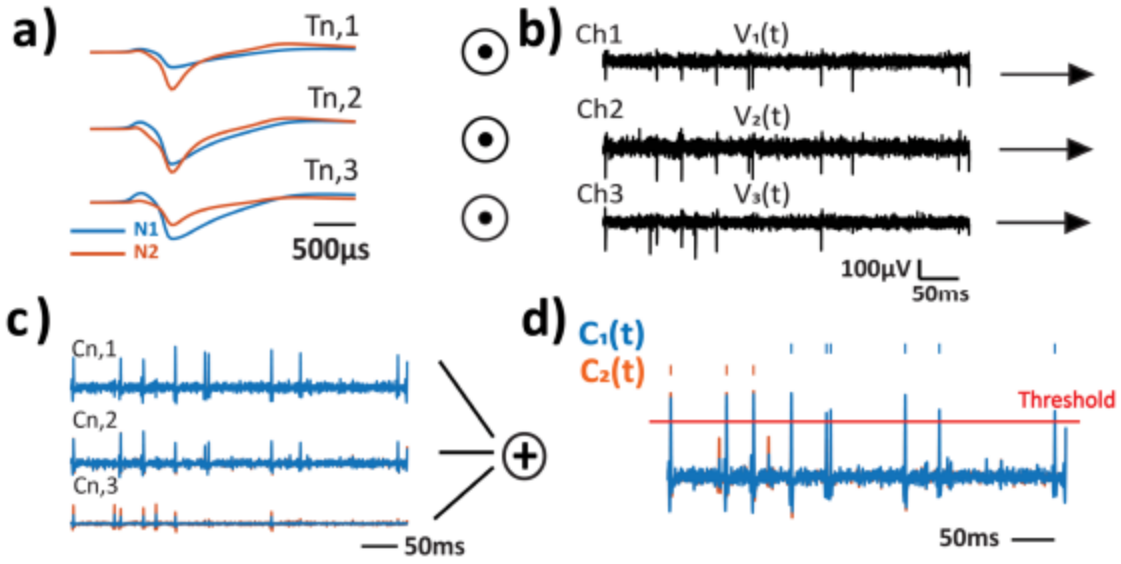


Figure 5.5: (a) Normalized templates of N1 and N2. (b) Neural recordings of three channels. (c) Computing the overall activation of neuron n neural recordings, i.e., voltage traces with normalized templates N1 and N2. Summing the convolution traces ($C_{n,m}(t)$) corresponding to each neuron. (d) Thresholding and assigning spikes to neurons N1 or N2 based on whether C1(t) > C2(t) (assign to N1) or C1(t) < C2(t) (assign to N2).

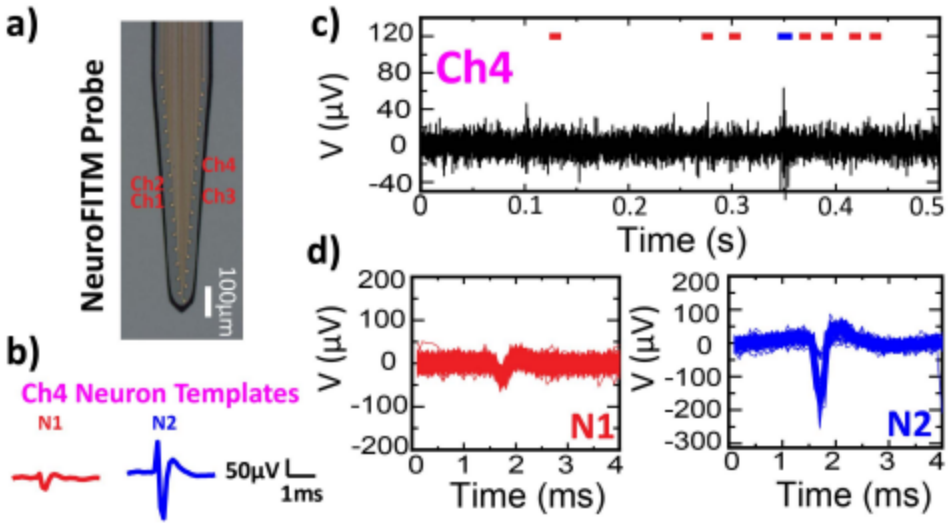


Figure 5.6: (a) Image of a 32-channel NeuroFITM probe with four representative channels highlighted as red. (b) Representative templates for the two neurons in Ch4. (c) Example 500 ms-recordings from Ch4 with predicted spike train marked in colored squares. (d) Clustered spikes for N1 and N2 for Ch4.

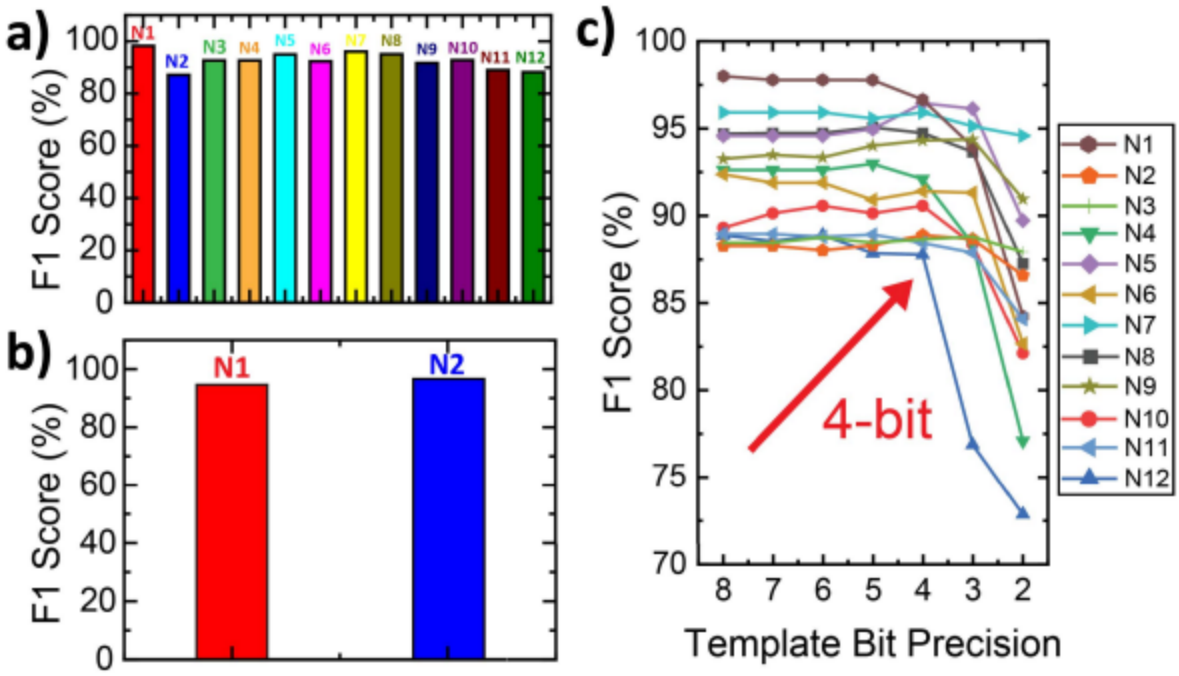


Figure 5.7: F1 scores (%) for (a) NeuroNexus-32 and (b) NeuroFITM dataset. (c) F1 score (%) as a function of template precision for 12 neurons in NeuroNexus-32 dataset. Four-bit quantized templates are used in HW experiments.

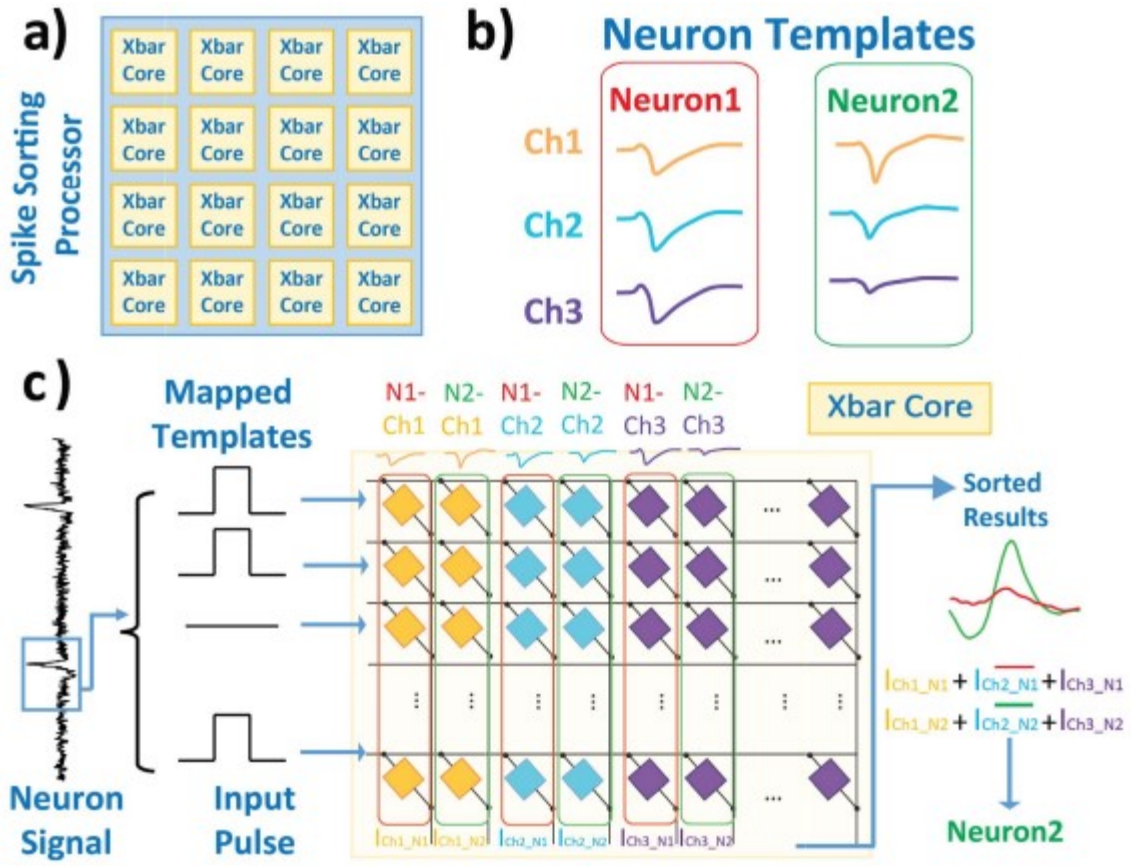


Figure 5.8: (a) Real-time spike sorting processor with multiple crossbar cores. (b) Representative templates of two neurons with three channels. (c) Crossbar spike sorting: each crossbar column stores a neuron term-bit digitized neural signals are provided as voltage inputs and weighted-sum currents from convolutions are obtained on the BLs. Neuron-wise aggregation of channel currents determines the sorting result.

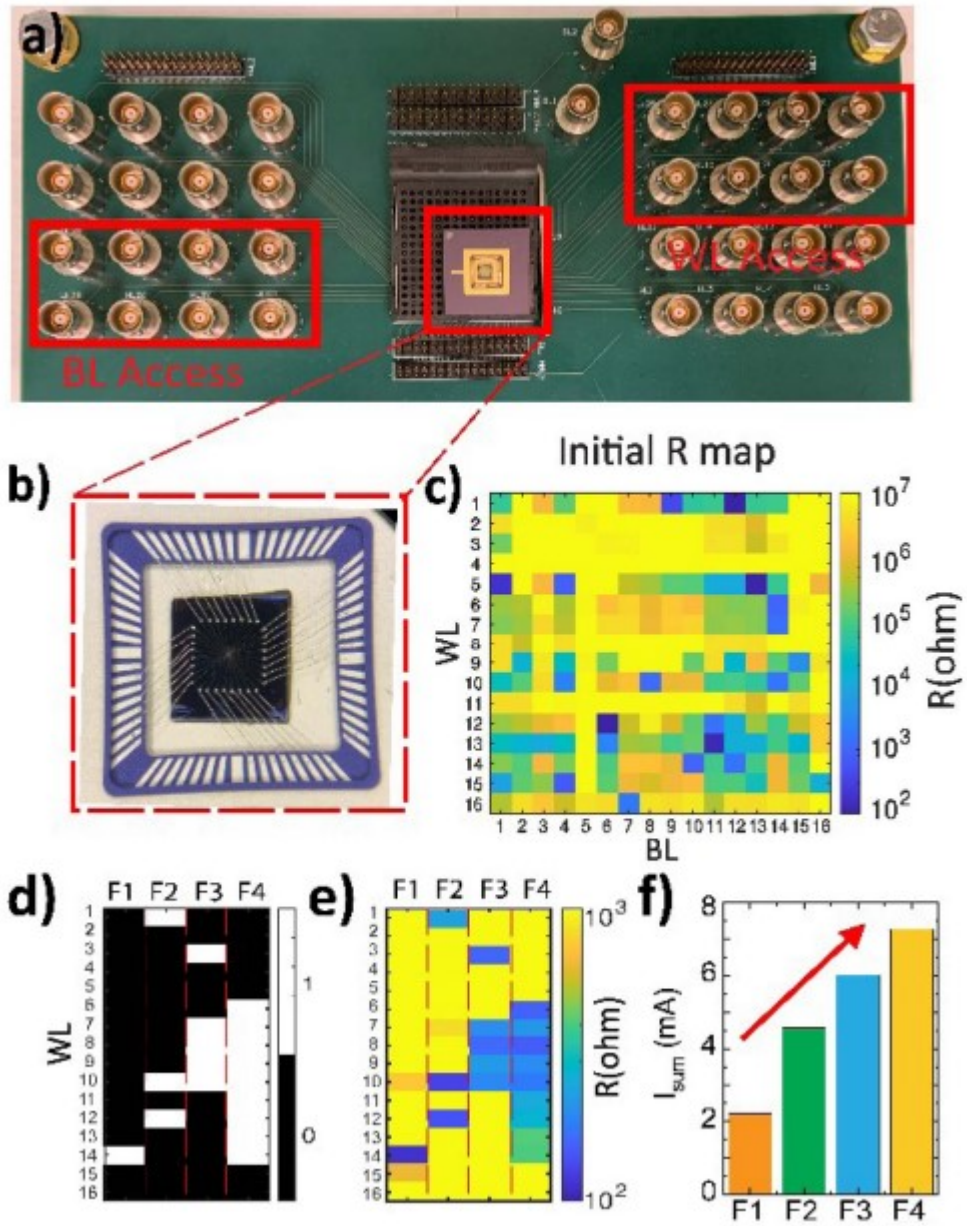


Figure 5.9: (a) Custom PCB board to access individual WLs and BLs of the CuO_x crossbar for the write and read operations. BLs can be accessed through the connectors shown in the lower left while WLs can be accessed through the connectors in the top right. (b) 16 × 16 crossbar wire-bonded onto a pin grid array (PGA) package. (c) Initial resistance map of a 16 × 16 CuO_x crossbar. (d) Four representative binarized (black = 0 and white = 1) filters (F1–F4) from NeuroFITM. (e) Programmed crossbar columns implementing these filters. (f) I_{sum} measured at VWLs = 0.25 V for four filters.

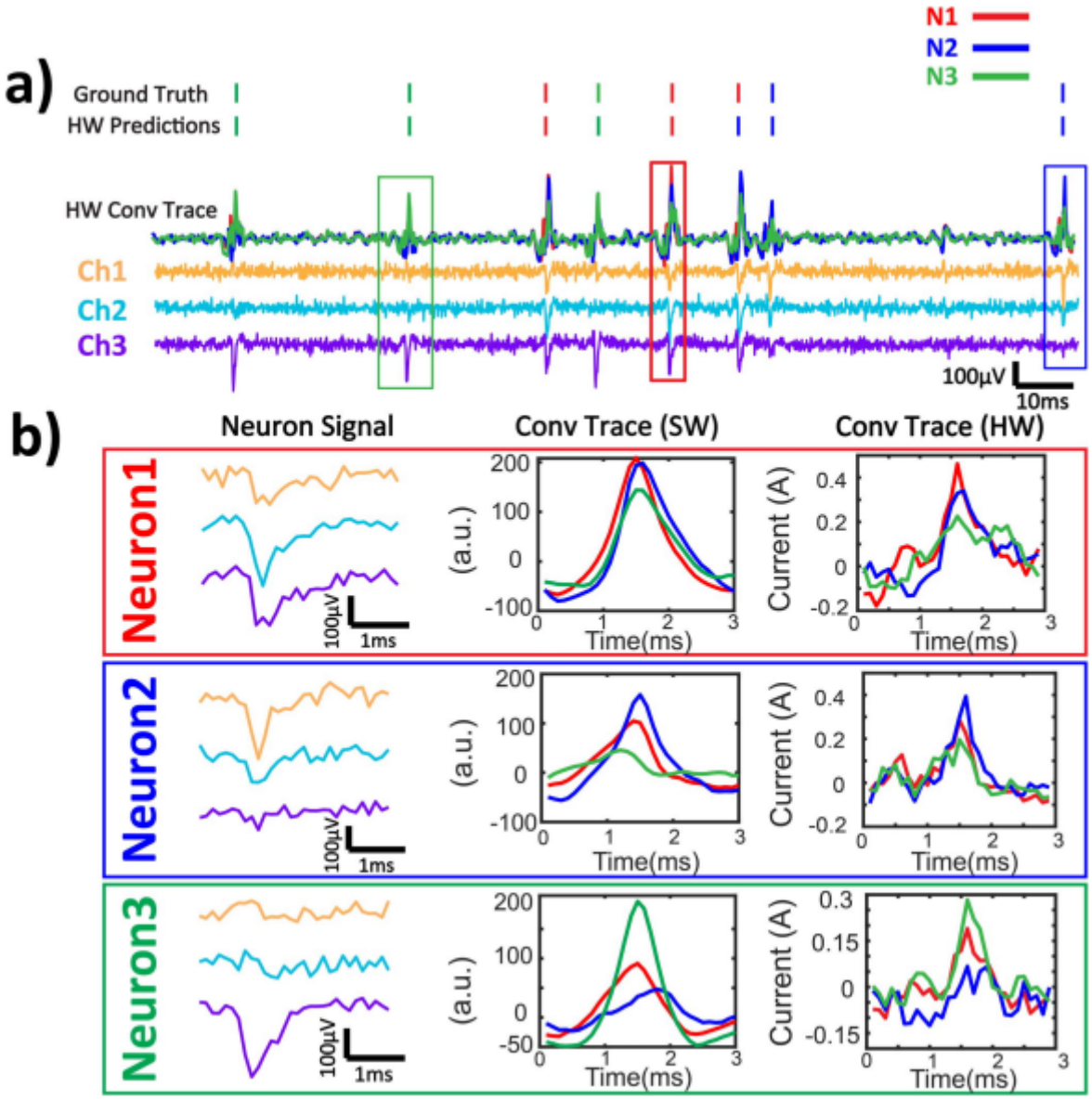


Figure 5.10: (a) NeuroNexus-32: Ch1–Ch3 are used to classify neurons N1–N3. A segment of recordings from Ch1 to Ch3 and predicted HW convolution (Conv) traces for three neurons. (b) Representative spike sorting results for N1–N3 showing convolution implemented in HW agrees with the SW implementation.

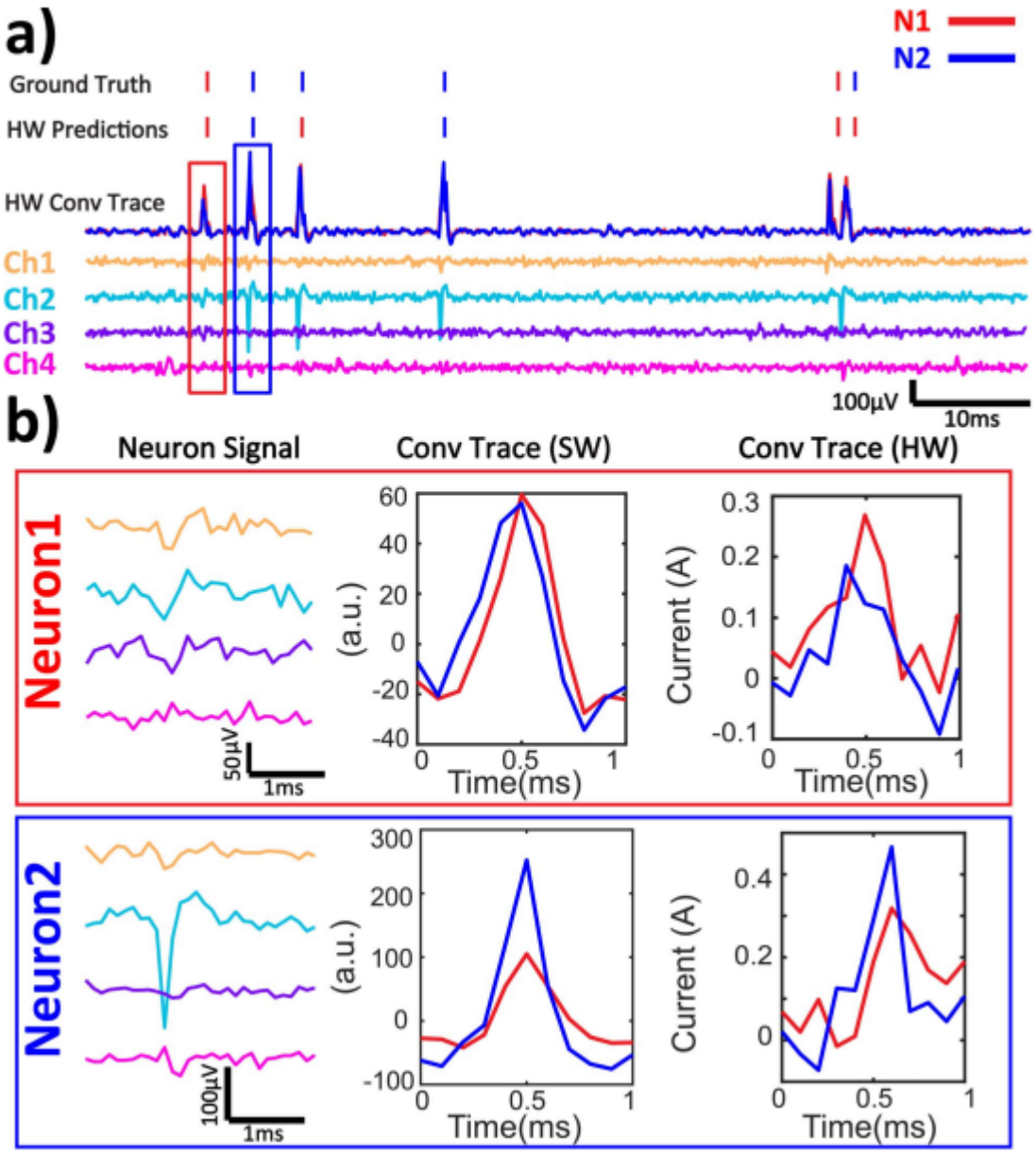


Figure 5.11: (a) NeuroFITM: Ch1–Ch4 are used to classify neurons N1 and N2. Segments of recordings from Ch1 to Ch4 and predicted HW Conv traces. (b) Representative spike sorting results for N1 and N2 implemented in HW agrees with the SW implementation.

Table 5.1: F1 Score for SW and HW Implementations.

	NeuroNexus-32 F1 Score (%)	NeuroFITM F1 Score (%)
SW	92.89 %	96.04 %
HW	92.48 %	94.62 %

Table 5.2: Benchmarking Our Results Against Previous Works[15,31] in Terms of HW Type, Recording Data Used in The Studies, Channel Count, Area/Channel, Power/Channel, Sorting Latency, and Energy/Channel. The Accuracy Obtained on Neuronexus-32 and NeuroFITM Data from SW and HW Experiments.

Reference	This Work	P. K. Wang et al.[15]	S. Luan et al.[31]
Hardware	Crossbar	FPGA	Microcontroller
Recording Data	Simulated, *in-vivo* experiments	*In-vivo* experiments	*In-vivo* experiments
No. of Channel	32	1	32
Area/Channel (mm^2)	8e-4	> 10	0.78
Power/Channel (mW)	2.15	460	3.11
Sorting Latency (μW)	4.8 per 100 channel	0.72 per channel	169 per channel
Energy/Channel (nJ)	10.3	331.2	525.6

6. References

1 Nicolelis, M. A., Ghazanfar, A. A., Stambaugh, C. R., Oliveira, L. M., Laubach, M., Chapin, J. K., Nelson, R. J. & Kaas, J. H. Simultaneous encoding of tactile information by three primate cortical areas. Nature neuroscience 1, 621-630 (1998).

2 Georgopoulos, A. P., Schwartz, A. B. & Kettner, R. E. Neuronal population coding of movement direction. Science 233, 1416-1419 (1986).

3 Moser, E. I., Kropff, E. & Moser, M.-B. Place cells, grid cells, and the brain's spatial representation system. Annu. Rev. Neurosci. 31, 69-89 (2008).

4 Quiroga, R. Q. & Panzeri, S. Extracting information from neuronal populations: information theory and decoding approaches. Nature Reviews Neuroscience 10, 173-185 (2009).

5 Luo, L., Callaway, E. M. & Svoboda, K. Genetic dissection of neural circuits: a decade of progress. Neuron 98, 256-281 (2018).

6 Liu, X., Ren, C., Lu, Y., Liu, Y., Kim, J.-H., Leutgeb, S., Komiyama, T. & Kuzum, D. Multimodal neural recordings with Neuro-FITM uncover diverse patterns of cortical–hippocampal interactions. Nature Neuroscience 24, 886-896 (2021).

7 Liu, X., Ren, C., Huang, Z., Wilson, M., Kim, J.-H., Lu, Y., Ramezani, M., Komiyama, T. & Kuzum, D. Decoding of cortex-wide brain activity from local recordings of neural potentials. Journal of Neural Engineering (2021).

8 Chaudhary, U., Birbaumer, N. & Ramos-Murguialday, A. Brain–computer interfaces for communication and rehabilitation. Nature Reviews Neurology 12, 513-525 (2016).

9 Collinger, J. L., Wodlinger, B., Downey, J. E., Wang, W., Tyler-Kabara, E. C., Weber, D. J., McMorland, A. J., Velliste, M., Boninger, M. L. & Schwartz, A. B. High-performance neuroprosthetic control by an individual with tetraplegia. The Lancet 381, 557-564 (2013).

10 Steinmetz, N. A., Aydin, C., Lebedeva, A., Okun, M., Pachitariu, M., Bauza, M., Beau, M., Bhagat, J., Böhm, C. & Broux, M. Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. Science 372 (2021).

11 Kim, S., Tathireddy, P., Normann, R. A. & Solzbacher, F. Thermal impact of an active 3-D microelectrode array implanted in the brain. IEEE Transactions on Neural Systems and Rehabilitation Engineering 15, 493-501 (2007).

12 Nicolelis, M. A. & Lebedev, M. A. Principles of neural ensemble physiology underlying the operation of brain–machine interfaces. Nature reviews neuroscience 10, 530-540 (2009).

13 Gibson, S., Judy, J. W. & Marković, D. An FPGA-based platform for accelerated offline spike sorting. Journal of neuroscience methods 215, 1-11 (2013).

14 Pachitariu, M., Steinmetz, N. A., Kadir, S. N., Carandini, M. & Harris, K. D. Fast and accurate spike sorting of high-channel count probes with KiloSort. Advances in neural information processing systems 29, 4448-4456 (2016).

15 Wang, P. K., Pun, S. H., Chen, C. H., McCullagh, E. A., Klug, A., Li, A., Vai, M. I., Mak, P. U. & Lei, T. C. Low-latency single channel real-time neural spike sorting system based on template matching. PloS one 14, e0225138 (2019).

16 Schäffer, L., Nagy, Z., Kincses, Z., Fiáth, R. & Ulbert, I. Spatial information based OSort for real-time spike sorting using FPGA. IEEE Transactions on Biomedical Engineering 68, 99-108 (2020).

17 Yin, S., Kim, Y., Han, X., Barnaby, H., Yu, S., Luo, Y., He, W., Sun, X., Kim, J.-J. & Seo, J.-s. Monolithically integrated RRAM-and CMOS-based in-memory computing optimizations for efficient deep learning. IEEE Micro 39, 54-63 (2019).

18 Sebastian, A., Le Gallo, M., Khaddam-Aljameh, R. & Eleftheriou, E. Memory devices and applications for in-memory computing. Nature nanotechnology 15, 529-544 (2020).

19 Yuhan Shi, A. A., Sangheon Oh, Xin Liu, Gopabandhu Hota,Gert Cauwenberghs,Duygu Kuzum. High Throughput Neuromorphic Brain Interface with CuOx Resistive Crossbars for Real-time Spike Sorting. International Electron Devices Meeting In press (2021).

20 Adamos, D. A., Kosmidis, E. K. & Theophilidis, G. Performance evaluation of PCA-based spike sorting algorithms. Computer methods and programs in biomedicine 91, 232-244 (2008).

21 Wouters, J., Kloosterman, F. & Bertrand, A. Towards online spike sorting for high-density neural probes using discriminative template matching with suppression of interfering spikes. Journal of neural engineering 15, 056005 (2018).

22 Bar-Hillel, A., Spiro, A. & Stark, E. Spike sorting: Bayesian clustering of non-stationary data. Journal of neuroscience methods 157, 303-316 (2006).

23 Herbst, J. A., Gammeter, S., Ferrero, D. & Hahnloser, R. H. Spike sorting with hidden Markov models. Journal of neuroscience methods 174, 126-134 (2008).

24 Rey, H. G., Pedreira, C. & Quiroga, R. Q. Past, present and future of spike sorting techniques. Brain research bulletin 119, 106-117 (2015).

25 Buccino, A. P. & Einevoll, G. T. Mearec: a fast and customizable testbench simulator for ground-truth extracellular spiking activity. Neuroinformatics 19, 185-204 (2021).

26 Rossant, C., Kadir, S. N., Goodman, D. F., Schulman, J., Hunter, M. L., Saleem, A. B., Grosmark, A., Belluscio, M., Denfield, G. H. & Ecker, A. S. Spike sorting for large, dense electrode arrays. Nature neuroscience 19, 634-641 (2016).

27 Hong, X., Loy, D. J., Dananjaya, P. A., Tan, F., Ng, C. & Lew, W. Oxide-based RRAM materials for neuromorphic computing. Journal of Materials Science 53, 8720-8746, doi:10.1007/s10853-018-2134-6 (2018).

28 Yu, S., Li, Z., Chen, P.-Y., Wu, H., Gao, B., Wang, D., Wu, W. & Qian, H. in 2016 IEEE International Electron Devices Meeting (IEDM). 16.12. 11-16.12. 14 (IEEE).

29 Yao, P., Wu, H., Gao, B., Eryilmaz, S. B., Huang, X., Zhang, W., Zhang, Q., Deng, N., Shi, L. & Wong, H.-S. P. Face classification using electronic synapses. Nature communications 8, 1-8 (2017).

30 Shim, W., Seo, J.-s. & Yu, S. Two-step write–verify scheme and impact of the read noise in multilevel RRAM-based inference engine. Semiconductor Science and Technology 35, 115026 (2020).

31 Luan, S., Williams, I., Maslik, M., Liu, Y., De Carvalho, F., Jackson, A., Quiroga, R. Q. & Constandinou, T. G. Compact standalone platform for neural recording with real-time spike sorting and data logging. Journal of neural engineering 15, 046014 (2018).

7. Acknowledgements

This work was supported in part by the Office of Naval Research under Grant N000142012405, in part by the National Science Foundation under Grant ECCS-1752241 and Grant ECCS-2024776, in part by the National Institutes of Health under Grant DP2 EB030992, and in part by the National Science Foundation under Grant ECCS-1542148.

Chapter 5, in full, is a reprint of the material as it appears in IEEE Transactions on Electron Devices, 2021, Shi, Yuhan; Ananthakrishnan, Akshay, Oh, Sangheon; Liu, Xin; Hota, Gopabandhu; Cauwenberghs, Gert; Kuzum, Duygu, A Neuromorphic Brain Interface Based on RRAM Crossbar Arrays for High Throughput Real-Time Spike Sorting, 2021. The dissertation author was a co-author of this paper.

CHAPTER 6. ENERGY-EFFICIENT MOTT ACTIVATION NEURON FOR FULL-HARDWARE IMPLEMENTATION OF NEURAL NETWORKS

To circumvent the von Neumann bottleneck, substantial progress has been made towards in-memory computing with synaptic devices. However, compact nanodevices implementing non-linear activation functions are required for efficient full-hardware implementation of deep neural networks. Here, we present an energy-efficient and compact Mott activation neuron based on vanadium dioxide and its successful integration with a conductive bridge random access memory (CBRAM) crossbar array in hardware. The Mott activation neuron implements the rectified linear unit function in the analogue domain. The neuron devices consume substantially less energy and occupy two orders of magnitude smaller area than those of analogue complementary metal–oxide semiconductor implementations. The LeNet-5 network with Mott activation neurons achieves 98.38% accuracy on the MNIST dataset, close to the ideal software accuracy. We perform large-scale image edge detection using the Mott activation neurons integrated with a CBRAM crossbar array. Our findings provide a solution towards large-scale, highly parallel and energy-efficient in-memory computing systems for neural networks.

1. Introduction

As the amount of data for computing exponentially increases, data transfer between memory and processor turns into a major bottleneck dominating the system-level energy consumption. In-memory computing has been proposed to circumvent this bottleneck arising from Von Neumann architecture by minimizing or eliminating the energy-consuming data transfer between memory and processor[1,2]. In-memory computing with emerging non-volatile

memories (eNVMs)[3-6] has shown promising results for on-chip storage of weights and computation of multiply-accumulate (MAC) operations for a single layer[7-9]. However, modern deep neural networks (DNNs) consist of hundreds of layers (e.g. ResNet has 152 layers[10]) that the outputs of each layer are individually connected to artificial neurons applying non-linear activation functions on weighted sums. Most in-memory computing approaches using eNVMs still rely on general processors to compute and propagate activation functions of each layer. However, activations that move in and out of the memory can dominate energy consumption of in-memory computing based accelerators [8,11-13]. Moreover, computation of one element of activation using analogue-to-digital converters (ADC) consumes energy comparable to the energy consumed by a whole synaptic array for a MAC operation[13]. Since DNNs need to have a very large number of activations to achieve high accuracy[13], it is critical to develop energy and area efficient implementations of activation functions, which can be integrated on the periphery of the synaptic arrays. Recent works have investigated analogue CMOS circuits[14] and ADCs with reconfigurable function mapping[15] for the implementation of nonlinear activation functions. However, a compact and energy-efficient nanodevice implementing the nonlinear activation functions has yet to be demonstrated.

Here we propose a volatile four-terminal Mott activation neuron device based on vanadium dioxide (VO_2) for compact and energy-efficient implementation of activation functions. The Mott activation neuron consists of a nanowire heater for precise control of the temperature of the VO_2 film. First, we experimentally demonstrate that the resistance of the Mott activation neuron can be switched linearly and gradually to emulate rectified linear unit (ReLU) activation function, which is the most widely used activation function. The Mott activation neuron can generate an output voltage, which follows the ReLU activation function for a given weighted sum current. Then, we

study the energy efficiency of the Mott activation neuron in comparison to activation function circuits with analogue CMOS[14] or reconfigurable digital ADC[15]. We investigate the performance of hardware neural networks implemented with the Mott activation neurons in terms of energy, latency, peripheral neuron/circuit area, and classification accuracy. Lastly, we fabricate CBRAM crossbar arrays and Mott activation neuron arrays to demonstrate edge detection using convolutional neural networks in hardware. Our results show that the small size and energy efficiency of the Mott activation neuron enable direct stacking of synaptic layers in neural networks and achieve substantial gains in energy efficiency and area while providing high accuracy.

2. Mott activation neuron

Neural networks consist of a set of neurons organized in layers, connected with synaptic weights (**Fig. 6.1a**). The inputs applied to the networks are multiplied by the corresponding weights and the multiplication results are accumulated in neurons. Then, the output of a neuron is calculated by passing the MAC results through a non-linear activation function. In-memory computing architectures map these neural network operations onto the arrays of eNVM devices. The weights are stored in arrays of eNVM devices and the weighted sum is calculated using Kirchhoff's current law[16]. While in-memory computing allows the local storage of the weights in compact and energy-efficient synaptic devices, the activation function calculations are still implemented with general processors or large and complex neuron peripheral circuits (**Fig. 6.1b**). It significantly degrades energy and area efficiency at the system-level. The activation function we target is the ReLU, which is the most widely used activation function. The output of ReLU activation function (i.e. $f(x) = \max(0, x)$) depends only on current input regardless of

previous inputs and resistance states (i.e. non-causal function). In addition, the output of the ReLU function is linear after the transition point (i.e. $x = 0$). In order to emulate the ReLU activation function, the device should exhibit volatile, linear, and gradual resistive switching. We developed a four-terminal VO_2-based activation device (illustrated in the inset of **Fig. 6.1b** on the bottom) that exploits a thermal driven Mott transition of VO_2 to embody these characteristics in a single nanodevice. The Mott ReLU device uses a nanowire heater (i.e. Ti (20 nm)/Au (30 nm)) to control the resistive switching of a lateral 50 nm VO_2 gap beneath it. The heater and the VO_2 gap are electrically insulated by 70 nm of Al_2O_3 layer. A scanning electron microscope (SEM) image of a fabricated device is shown in **Fig. 1c** and detailed fabrication procedures are discussed in **Methods**. The heater generates heat through Joule-heating depending on the magnitude of the weighted sum current generated by each column of the eNVM array. Then, the generated heat is transferred to the VO_2 film through the electrical insulator (i.e. Al_2O_3 layer) and induces the phase transition from the insulating states to the metallic states, which results in a resistivity drop. The temperature-dependent resistance of the VO_2 gap is shown in **Fig. 6.1d**. To map the gradual resistivity changes of the VO_2 gap onto the output voltage (V_{OUT}), a voltage divider circuit is implemented as illustrated in the inset of **Fig. 6.1e**. The supply voltage (V_{DD}) is divided into the voltage drop across the VO_2 gap and the load resistor depending on the resistance ratio of the VO_2 gap and the load resistor. As the resistance of the VO_2 gap decreases, the voltage drop across the VO_2 gap decreases which results in the increment of the output voltage (or the voltage drop across the load resistor). As a result, the resistive switching of the VO_2 gap allows the output voltage to emulate ReLU activation function as illustrated in **Fig. 1e**. Since the output of the Mott ReLU device is voltage, it can be directly applied to the next layer as an input voltage. Therefore, multiple synaptic layers can be directly stacked on each other for

driving the next layers by eliminating complex digital circuits and ADCs between the layers. Moreover, the small size of the Mott ReLU device allows the integration of the device for each column of the synaptic array, which eliminates the need for time-multiplexing and hence, enables fully parallel operations.

The main operating principle of the Mott ReLU device is the Mott transition (or insulator-to-metal transition) of the VO_2 gap. The Mott transition of the VO_2 gap can be induced either by electrical filamentary switching or thermal-driven domain-wise switching[17,18]. When a voltage bias above the threshold is applied across the VO_2 gap, Joule-heating due to the bias induces filament formation, and the filament is widened as the voltage increases (**Fig. 6.2a** and **b**). Since the filament formation is a cascading avalanche effect, the resistance switching is abrupt[19]. In contrast, when the transition is driven by temperature, only the domains whose critical temperature is below the device temperature transit to the metallic phase (**Fig. 6.2a** and **c**). Since the transition temperature of each domain exhibits variations[20], the number of domains switched to metallic phases gradually increases as the temperature increases. As a result, the resistance of VO_2 gradually decreases as the temperature increases (**Fig. 2d**). This gradual switching behaviour of VO_2 was confirmed by scanning microwave microscopy (SMM) imaging of VO_2 film previously[20]. The Mott ReLU device is engineered to exploit this thermal-driven linear resistive switching for emulating the linear increment of the ReLU activation function, as shown in **Fig. 6.2e**. Then, this linear resistive switching of the VO_2 gap is projected to the output voltage. The ratio between V_{OUT} and V_{DD} of the Mott ReLU device with a 1,900 Ω load resistor is demonstrated in **Fig. 6.2f** as a representative example. Two potential practical issues regarding the Mott transition are discussed in **Supplementary Note 1**.

To further assess the compatibility of the Mott ReLU device for implementing the ReLU activation function, we extensively characterized its switching characteristics. In addition to gradual switching, the resistive switching should be volatile to implement ReLU function in synaptic arrays. That is because the output of the ReLU activation function should only depend on current input regardless of previous inputs and resistance states. The volatile switching of the Mott ReLU device is experimentally verified in **Fig. 6.3a**. When 1 ms wide current pulses with various amplitudes are applied to the heater, the resistance of the device is switched and maintained only when the current pulse is high. Furthermore, the output voltage for a given input current should not exhibit a high level of variations, which could degrade neural network performance. **Figure 6.3b** demonstrates that each resistance state of the device shows only ~4% or less variation when the resistance states are iteratively measured. The impact of this small variation on the neural network performance is studied in **Neural Network Implementations** Section. Lastly, the endurance of the device should be high to allow a large number of weighted sum operations in hardware. For the inference with MNIST dataset[21], each Mott ReLU device should generate its output for 10,000 times per epoch (or whole testing set). Hence, the device should endure this large number of cycling operations. **Figure 6.3c** experimentally demonstrates that the Mott ReLU device shows no sign of ON/OFF ratio degradation up to 5,000 cycles. It has been shown that endurance larger than 10^{10} cycles can be easily achieved with VO_2 devices[22]. Furthermore, we performed pulse measurements to investigate the power consumption and the latency of the Mott ReLU device. **Figure 6.3d** shows the total power consumption as a function of heater current, as well as the power consumed by the heater and the VO_2 gap separately. The total power consumption of the Mott ReLU device is dominated by the heater. The latency of the Mott ReLU is 61.4 ns, measured as the time difference between the first saturation point of the

input and output pulse (**Figure 6.3e**). The energy consumption of the Mott ReLU is 199.5 pJ for 65 ns pulse width.

The Mott ReLU device can replace complex peripheral circuits for activation function calculation. Therefore, it is important to compare the performance of the Mott ReLU device against other implementations of activation functions (i.e. analogue CMOS[14] and digital ADC[15] circuits discussed in **Methods**). The performance benchmarking (**Supplementary Note 2**) results of the Mott ReLU device against the analogue CMOS circuit[14] and the digital ADC implementation[15] are summarized in **Table 6.1**. The energy consumption of Mott ReLU can be further reduced by optimizing the device to have more heat confinement on the VO_2 gap. As the heat generated by the heater is more confined to the VO_2 gap, the device requires less heater current to achieve the same temperature on the VO_2 gap[23]. Therefore, by replacing the heater material with a higher thermal resistance material (e.g. Ti has thermal resistance ~10× higher than that of Au), the energy consumption of the device can be lowered. To determine the energy consumption of an optimized device, we developed an empirical thermal model of our device (**Fig. 6.S1a** and **b**) as discussed in **Supplementary Note 3**, which shows good agreement with experimental data as shown in **Fig. 6.2e** and **f**. The power consumption of Mott ReLU can be reduced by ~25× down to 128 μW by increasing the thermal resistance of the nanowire heater (**Fig. 6.S2a**). Moreover, the latency can be reduced to ~3.8 ns (**Table 6.1**) by minimizing the parasitic capacitance of the Mott ReLU below 10^{-11} F (**Fig. 6.S2b**), which would result in a total reduction of ~300× in energy consumption down to 0.638 pJ (**Table 6.1**). Our experimental results show that the Mott ReLU device achieves 450−1500× improvement in area and 1.5-3× improvement in latency while achieving low energy consumption. Moreover, the optimization of the Mott ReLU device can further reduce the energy consumption and improve the latency

offering substantial gains in area, latency, and energy efficiency as a replacement to the analogue CMOS[14] and digital[15] ADC circuits.

3. Neural network implementations

We have demonstrated that the Mott ReLU neurons can provide smaller area and better energy efficiency as compared to the other circuit implementations. It is also critical to evaluate the network-level performance using the Mott ReLU devices for hardware implementation of DNNs. To compute the accuracy of neural network implementations with the Mott ReLU device, we simulated MLP (**Fig. 6.4a**) and LeNet-5[21] (**Fig. 6.4b**) (The details on the configurations of the networks are discussed in **Methods**). The schematic and transmission electron microscopy (TEM) image of the CBRAM cell (**Fig. 6.S3a**) are shown in **Fig. 6.S3b** and **c**, respectively. **Table 6.2** summarizes the accuracy results of ideal (i.e. software ReLU) and Mott ReLU cases for both MLP and LeNet-5. We investigated both online learning (i.e. training is done on the hardware) and offline classification cases (i.e. only inference is done on the hardware). When the ReLU activation functions of MLP (**Fig. 6.4c**) or LeNet-5 (**Fig. 6.4d**) are quantized, the accuracy degradation is not significant unless the precision is ~6 bit or higher. Since the precision of the Mott ReLU device is high enough (~6 bit), the accuracy degradation due to the Mott ReLU is negligible as compared to the accuracy degradation due to the synaptic devices (~10 % for MLP and ~3 % for LeNet-5). This is mainly because of the limited precision (~5-bit) of the CBRAM devices[24]. The neural networks with variations (cycle-to-cycle in **Fig. 6.4e** and **f** and device-to-device **Fig. 6.S4 a** and **b**) on Mott ReLU are also investigated and verified that there is no significant accuracy degradation due to the variations (**Supplementary Note 4**). Since the Mott ReLU achieves accuracies close to the ideal software, the accuracy will not be a limiting factor

for implementing activation functions using the Mott ReLU device.

4. System-level performance benchmarking

To evaluate the performance of the hardware system for neural networks with the Mott ReLU device, we performed system-level performance benchmarking for offline classification using NeuroSim platform[25]. NeuroSim is a C++-based circuit-level macro-model for neuro-inspired architectures. We modified NeuroSim to integrate Mott ReLU peripherals with CBRAM synaptic cores. We compared the synaptic cores with the Mott ReLU peripheral against the ones with peripheral circuits implemented by analogue[14] and digital[15] CMOS ReLU circuits. For the Mott ReLU peripheral, the experimental results on energy and latency (**Fig. 6.3d** and **e**) are integrated into the NeuroSim Platform[25]. The peripheral circuits of analogue CMOS ReLU circuits for NeuroSim platform[25] is developed based on the SPICE simulations. The dynamic energy, leakage power, and latency of Mott ReLU and CMOS ReLU activation circuit shown in **Table 6.1** are integrated into the circuit modules.

The architecture of the hardware systems with conventional digital peripheral circuits, the Mott ReLU device, and analogue CMOS circuits are illustrated in **Fig. 6.5a**, **b**, and **c**, respectively. In contrast to the conventional analogue one-transistor one-resistor (1T1R) architecture with digital neuron peripheral (**Fig. 6.5a**)[25], the Mott ReLU device allows a simpler synaptic core design (**Fig. 6.5b**) by avoiding MUX sharing (**Supplementary Note 5**) and replacing complex circuits and ADCs. Before system-level benchmarking, we first investigated whether the Mott ReLU device can drive the inputs to the next synaptic array without additional circuits by performing circuit simulation with SPICE (**Supplementary Note 6**). This result (**Fig. 6.S5a** and **b**) clearly demonstrates that the Mott ReLU device can generate stable output to drive

the next synaptic layer without additional circuits. The system-level performance benchmarking results are summarized in **Table 6.S1**. The architecture with Mott ReLU (65 ns input pulse) provides substantial gains over the architectures with analogue CMOS and digital ADC implementation (**Supplementary Note 7**). Lastly, we compared the performance of synaptic cores with Mott ReLU and analogue CMOS circuits considering technology scaling (130 nm to 14 nm) as discussed in **Methods**. The results in **Fig. 6.5d** demonstrate that the experimentally measured Mott ReLU provides ~10× energy gain regardless of the CMOS technology node. Moreover, the system-level gain in energy can be further improved up to ~100× using the optimized Mott ReLU in comparison to analogue CMOS ReLU. More importantly, the Mott ReLU achieves orders of magnitude smaller peripheral circuit area in comparison to both digital ADC and analogue CMOS implementations of the activation function. The system-level performance results clearly show that the Mott ReLU device offers a promising approach to replace power-hungry and large-area activation function circuits in the neuron periphery.

5. Integration of Mott ReLU devices with crossbar arrays

To demonstrate the integration of Mott ReLU devices with synaptic arrays in hardware, we fabricated CBRAM crossbar arrays (**Fig. 6.6a**) and a Mott ReLU device array (**Fig. 6.6b**) as explained in **Methods**. We designed a custom printed circuit board (PCB) (**Fig. 6.6c**) to interface and integrate the CBRAM and the Mott ReLU chips in hardware. Each column of the crossbar array is directly connected to Mott ReLU devices (**Fig. 6.6d**) to investigate how the weighted sum current generated by the array controls the output voltage of the Mott ReLU devices. First, we varied the input voltage to the crossbar array (i.e. -250 to 250 mV) while programming the weights of ~2/3 of synaptic devices on a column to the low resistance state and the rest to the

high resistance state. **Figure 6.6e** shows that output voltage exhibits ReLU characteristics as the input voltage to the CBRAM devices are increased from –250 mV to 250 mV. Then, we varied the synaptic weights in the column while the input voltage was fixed to 130 mV. As the ratio of devices programmed to the low resistance state increases, the output voltage exhibits ReLU characteristics (**Fig. 6.6f**). These experimental results demonstrate that the weighted sum current that depends on the input voltage and the weights (resistance) of the synaptic devices can successfully drive the Mott ReLU neurons to implement ReLU activation function.

For large-scale hardware demonstration, we implemented a convolutional edge detection operation[26] with filters (**Fig. 6.S6a** and **b**) followed by a ReLU operation on a real-world image with the CBRAM crossbar and the Mott ReLU array using the custom PCB as discussed in **Methods**. The weighted sum current resulting from the convolution operation from four representative 10 × 10 input patches (**Fig. 6.6g**) with both the lateral and vertical edge detection filters (**Fig. 6.S6a** and **b**) mapped using differential pair scheme (**Supplementary Fig. 6c**) are shown in **Fig. 6.6h** and **i**, respectively. The weighted sum current generated during the convolution operation is fed to the Mott ReLU devices to perform ReLU operation on the weighted sum. The output voltages of the Mott ReLU devices as a result of weighted sum with the lateral and vertical filters for the whole input image are shown in **Fig. 6.6j** and **k**, respectively. These results show that lateral and vertical edges of the image are detected by implementing corresponding filters using the Mott ReLU devices integrated with the CBRAM crossbar array in hardware. The successful edge detection using the Mott ReLU devices integrated with the CBRAM crossbar array proves the feasibility of using Mott ReLU neurons as activation units for in-memory computing systems.

6. Conclusions

We introduced a nanoscale, Mott-transition-based device for the ReLU activation function. The device exhibits volatile, linear and gradual resistive switching of a VO_2 film controlled by the metal nanowire heater on top of it. The Mott ReLU device shows minimal cycle-to-cycle variation and long endurance, which are important for hardware implementation of neural networks. We have shown that the Mott ReLU devices generate an output voltage, which follows the ReLU activation function, with the given input current. This allows the Mott ReLU device to drive the synaptic devices on the next layer directly. We performed system-level simulations for a hardware implementation of neural networks with the Mott ReLU devices. Moreover, we experimentally demonstrated that the Mott ReLU devices can be integrated with CBRAM crossbar arrays to perform filtering operations of convolutional neural networks. Our findings suggest that the device with Mott-transition-based activation can achieve substantial gains in energy, latency and area compared to the digital or analogue circuit implementations of the activation function, while maintaining high accuracy. The small size and high energy efficiency of the Mott device provide a solution towards large-scale, highly parallel and energy-efficient in-memory computing systems for DNNs.

7. Methods

A. Mott device fabrication.

To fabricate the Mott transition based activation devices, 70nm VO_2 film is grown by reactive sputtering on top of an Al_2O_3 substrate in 4 mTorr Ar/O_2 (8% of O_2) ambient at 520°C. Then Ti (20 nm)/Au (30 nm) electrodes are patterned using e-beam lithography and e-beam evaporation to define the 50 nm VO_2 gap. 70 nm Al_2O_3 is deposited as the insulating layer. Ti

(20 nm)/Au (30 nm) nanowire heater is patterned on top of the Al_2O_3 aligning with the VO_2 gap using e-beam lithography and e-beam evaporation. To isolate each device, the VO_2 film outside the active area is etched with reactive ion etching. The resistance of the heater is ~30 Ω while the resistance of the VO_2 gap without bias to the heater is ~10 kΩ.

B. Device measurement set-up.

To measure thermal gradual resistance switching of VO_2 while preventing electrical switching, we applied 1 μA current source using Keithley 6221 to the VO_2 gap. The current is small enough not to initiate electrical switching. Then, we measured the voltage across the VO_2 gap using Keithley 2182A. The resistive switching of the VO_2 gap is solely controlled by the heat generated by the heater on the top of the VO_2 gap. The heat generation is controlled by a voltage source connected to the heater. We measured the current flow through the heater to measure the heat generation using an oscilloscope. For variability and endurance measurement, Keithley 6221 is used to apply a current pulse train to the heater. Then the resistance of the VO2 gap is extracted by measuring the voltage across the VO_2 gap using Keithley 2182A while applying constant 1 μA current through the gap using another Keithley 6221. The ambient temperature is controlled by Lake Shore TTPX Probe station for all the measurements.

C. CMOS ReLU implementation.

The analogue CMOS circuit consists of three operational amplifiers, which amplify the input current and convert the input current to the output voltage, and an analogue switch that implements the rectifying function. The digital ADC circuit is implemented using ADC with reconfigurable function mapping. In order to evaluate the energy and latency of these three different ReLU implementations as an activation function, we assume that all implementations

get an identical weighted sum result as an input to the Mott ReLU device or digital/analogue CMOS circuits. The area of each implementation is calculated from the layout of the device or circuits.

D. Neural network configuration

The MLP used for network simulations is composed of 785 input neurons (i.e. 1 input neuron for bias and the other 784 neurons for MNIST dataset inputs), 128 hidden neurons, and 10 output neurons. Each output neuron represents one of the digits (from '0' to '9'). The hidden neurons have the ReLU activation function while the output neurons have the soft-max activation function. The LeNet-5 has 6 of 5 × 5 convolutional filters for 28 × 28 MNIST input images. The outputs from the convolutional filters are fed to the ReLU activation function. Then, the outputs of ReLU activation functions are down sampled using 2 × 2 max pooling. The second convolutional layer has 16 of 8 × 8 convolutional filters with 2 × 2 max pooling. The outputs from the last max-pooling layer are fed into the FC layers, which has 120 input neurons, 80 hidden neurons, and 10 output neurons. The input neurons and hidden neurons of the FC layers have ReLU activation functions while the output neurons have soft-max activation functions. In the network simulations, the ReLU activation functions on the neuron layers (i.e. the hidden layer of the MLP and convolutional layers and fully connected (FC) layers of the LeNet-5) are implemented with the Mott ReLU based on its experimental measurement results. A 1,900 Ω load resistor is connected to the Mott ReLU, and 5 mA of offset current is applied to the Mott ReLU through an additional row on the synaptic array to shift the transition point to 0 mA. The weights are mapped onto the arrays of CBRAM devices by using the characteristics of CBRAM devices. The CBRAM cells used for the simulations exhibit ~40 conductance levels (~5-bit) and 100 ON/OFF ratio. For the network simulation, the weights of the network ranging from −1 to 1

are mapped to the minimum (~1 μS) and maximum (~100 μS) conductance of CBRAM cells. Similarly, the outputs of the ReLU activations (0 to 785) are also linearly mapped to the output voltages of Mott ReLU devices (0 to 200 mV).

LeNet-5 requires larger fanout for the FC1 layer. To address this, we incorporated a time multiplexing approach. By enabling a subset of columns of the synaptic array sequentially with the switch matrix, the number of devices connected to each Mott ReLU can be controlled. Since our architecture already has a switch matrix, this approach is directly implemented in performance benchmarking simulations with NeuroSim. It is important to note that larger-scale DNN models may require additional peripheral circuit blocks including buffers if they have many layers with large fanout. These blocks could be integrated with the synaptic arrays in the future and accounted for the performance benchmarking for different models.

E. Convolutional filtering with the Mott ReLU device integrated with CBRAM array.

To implement convolutional filtering using Mott ReLU and CBRAM array for image edge detection, the PCB is controlled by a semiconductor parameter analyser (Agilent 4155C) and a switch matrix (HP E5250A). Then, biasing and measurement are done by the semiconductor parameter analyser (Agilent 4155C). The 4 × 4 lateral and vertical filters are programmed into the columns of the crossbar array by unrolling the filters into 16 × 1 vectors on the CBRAM array. For each filter, the positive and negative weights are represented using two columns of the crossbar array to form a differential pair (i.e. $G = G^{+} - G^{-}$). The input image (180 × 270) is quantized (i.e. 16 levels) and converted into a voltage pulse train of 4 binary pulses (i.e. 250 mV for '1' and 0 mV for '0'). For the column representing negative weights, a negative voltage pulse train is applied as input to form a differential pair with the column representing positive weights (i.e. $I = I^{+} - I^{-}$). For the convolution operation, a filter slides over the input image and the

weighted sum currents from the pair are combined and fed into a Mott ReLU device. For Mott ReLU devices, 1.1 V is applied to the VO_2 Gap, load resistors are set to 3.3 kΩ, and 7 mA of offset current is applied to the heater.

8. Supplementary Notes

A. Supplementary Note 1: Potential Practical Issues of Mott Transition

Two important practical issues regarding the Mott transition must be taken into account for eventual hardware implementation. The existence of discrete domains that transit individually gives rise to discontinuities in the thermal switching behavior (**Fig. 6.2e** and **f**). However, the small jumps are not detrimental to neural network operation considering the ReLU function will be quantized. Moreover, this could also be solved by using devices larger than the typical domain size. The application of a very high electric field may lead to the formation of a metallic filament with a consequent sudden resistance change. This leads to a practical voltage limit that can be applied between electrodes, which is solvable by increasing the separation between the electrodes to lower the electric field.

B. Supplementary Note 2: The Setup for ReLU Performance Benchmarking

For performance benchmarking of single ReLU device or circuit, we assume that a multi-layer perceptron (MLP) network for MNIST dataset generates average weighted sum value which is 20, while the weighted sum can range from -785 to 785 with weights ranging from -1 to 1. We used this average weighted sum value as an input to all the implementations. For the analogue CMOS circuit, we implemented a synaptic array which has 785 columns and the analogue CMOS circuit in SPICE. The number of columns is set to 785 to match the input size of the MNIST dataset. All the synaptic devices are set to 10 kΩ. Then, 20 out of 785 devices are

biased to 450 mV (others to 0 mV), which generates the average weighted sum value for the MLP network. For the Mott ReLU device, the same amount of input current is applied as the heater current of a Mott ReLU device. Then, the energy consumption of the Mott ReLU is experimentally estimated (Dynamic power consumption of Mott ReLU for various input current is shown in **Fig. 6.3d**). Lastly, the digital ADC circuit is estimated by Spectre circuit simulations when the average weighted sum value is applied to the single digital neuron circuit[15].

C. Supplementary Note 3: Compact Thermal Model of Mott ReLU Device

To develop a compact thermal[27] Model of Mott ReLU device using the thermal switching of VO_2 film, we adopted a Mott transition model. Then, we developed a compact thermal model for the thermal dynamics of the heater and the VO_2 gap. The compact thermal model also includes the thermal coupling between the heater and the VO_2 gap. The equations of the thermal compact model are as follows:

$$C_{th\_H}\frac{dT_H(t)}{dt} = \frac{\left(T(t) - T_H(t)\right)}{R_{ox}} + {I_{IN}}^2 R_H - \frac{(T_H(t) - T_0)}{R_{th\_H}}$$

$$C_{th_{VO_2}}\frac{dT(t)}{dt} = \frac{\left(T_H(t) - T(t)\right)}{R_{ox}} + V_{VO2}I_{VO2} - \frac{(T(t) - T_0)}{R_{th_{VO_2}}}$$

D. Supplementary Note 4: Network Simulation with Variations

We performed network simulations to evaluate the impact of variations of the Mott ReLU device on the online learning accuracy of MLP (**Fig. 6.4a**) and LeNet-5 (**Fig. 6.4b**). Unless the cycle-to-cycle (or intra-device) variation (σ) on the output voltage does not exceed 50%, the accuracy with the Mott ReLU is not degraded for both MLP (**Fig. 6.4e**) and LeNet-5 (**Fig. 6.4f**). Since the cycle-to-cycle variation of the Mott ReLU device is ~4% or less (**Fig.6.3b**), the accuracy degradation due to the cycle-to-cycle variation will be negligible. We also performed simulations for device-to-device (or inter-device) variation. Similarly, the accuracy of MLP (**Fig.**

6.S4a) and LeNet-5 (**Fig. 6.S4b**) does not notably degrade up to 50% of the device-to-device variation (σ). Our Mott ReLU devices show device-to-device variation less than 50% (i.e. 25.2% for low resistance state and 40.7% for high resistance state), and hence the device-to-device variation is not expected to impact classification accuracy.

E. Supplementary Note 5: MUX Sharing for CMOS ReLU Implementations

Analog CMOS circuit[14] (**Fig. 6.5c**) could also replace the ADCs and complex digital circuits. However, the CMOS circuit would occupy ~39 columns of the synaptic array due to its large size. Therefore, each circuit would need to be shared by 39 columns using a multiplexer (MUX) and MUX decoder, leading to longer latency as one memory access splits into 39 stages. We also considered the digital ADC implementation[15] of the activation function circuit for system-level benchmarking. Similar to the analogue CMOS implementation, each circuit is shared by 22 columns using a MUX and MUX decoder due to its large size. Moreover, the digital ADC implementation would require shared external circuits consisting of lookup tables and digital-to-analogue converters (DACs), occupying an additional area of 0.086 mm^2.

F. Supplementary Note 6: SPICE Simulation for Mott ReLU Device with Synaptic Array

We performed circuit simulations in MLP configuration with 785 input neurons and 10 output neurons by incorporating our device model. A weighted sum current from 785 synaptic devices on the input-to-hidden layer is applied to the Mott ReLU neuron which then drives synaptic devices on the hidden-to-output layer (**Fig. 6.S5a**). During neural network operations, weights can be programmed into various resistance values. Depending on the magnitude of weighted sum current, the Mott ReLU neuron generates the output voltage which is fed to the next layer (**Fig. 6.S5b**). Regardless of the resistance changes on the synaptic devices, the Mott ReLU should still provide a stable output driving the next layer. Otherwise, additional circuits

(i.e. buffers or amplifiers) would be required to stabilize the output of the Mott ReLU device. To test that, we changed the resistance values of the synaptic devices from 100kΩ to 1MΩ, while monitoring the output voltage as a function of weighted sum current (**Fig. 6.S5b**). The output voltage is varied only by ~6.7 mV or less (**Fig. 6.S5b**).

G. Supplementary Note 7: System-level Benchmarking

The architecture with Mott ReLU (65 ns input pulse) provides substantial gains over the architectures with analogue CMOS in energy consumption (~7× for MLP and ~17× for LeNet-5), latency (~16× for MLP and ~10× for LeNet-5), and peripheral area (~119× for MLP and ~216× for LeNet-5) for offline classification. While the energy consumption for the Mott ReLU case approaches to the digital ADC implementation, the architecture with Mott ReLU achieves significantly lower latency (~32× for MLP and ~20× for LeNet-5), and smaller peripheral area (~885× for MLP and ~907× for LeNet-5) compared to the architecture with the digital ADC implementation.

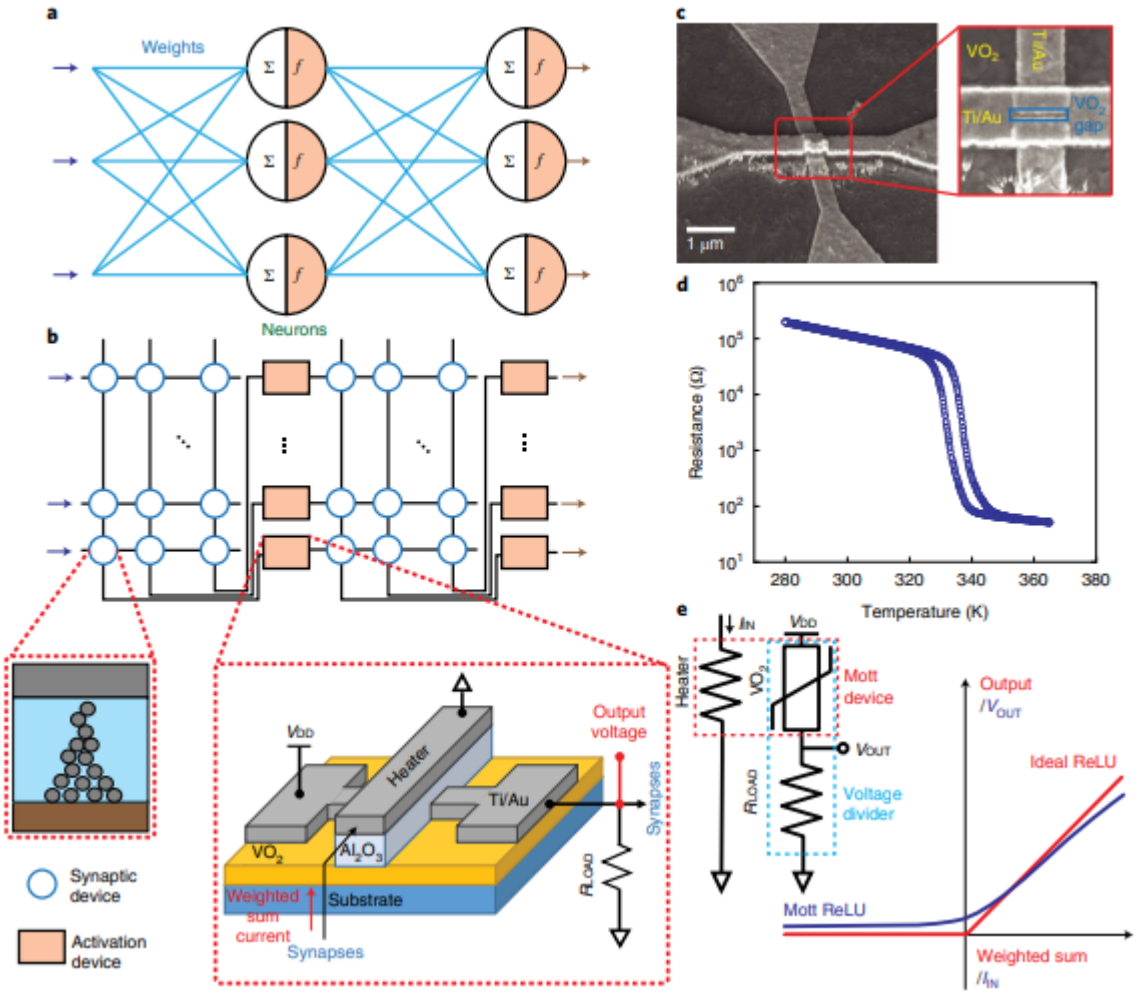


Figure 6.1: The Mott ReLU device for the hardware implementation of a neural network. a,b, An illustration shows a neural network (a) and hardware implementation (b) of the neural network with synaptic and activation (or neuron) devices. Σ represents a weighted sum while f represents the activation function. In b, the inset on the left shows a schematic of a resistive memory cell. The inset on the right shows a schematic of the Mott device with a nanowire heater. Mott activation devices allow direct stacking of multiple eNVM arrays for DNNs. The heater is connected to a column of presynaptic arrays and gets a weighted sum current. Then, one pad of the VO_2 gap is connected to V_{DD} and the other pad is connected to the next synaptic array. The pad connected to the next synaptic array is also connected to a load resistor. Weights are stored in eNVM devices, and weighted sum currents from each column are fed into the Mott ReLU. Then, the output of the Mott ReLU is applied as the input to the next layer. c, A scanning electron microscope image of the Mott device (scale bar, 1 μm). The inset shows the nanowire heater on the top of the 50 nm VO_2 gap. d, Resistance of the VO_2 gap when the temperature is swept from 280 K to 365 K. e, An illustration shows how a Mott device will be used as a ReLU activation function. The output of the ReLU activation function will be represented by VOUT of the Mott ReLU device while the weighted sum input to ReLU will be represented by the input current (I_{IN}) to the Mott ReLU device.

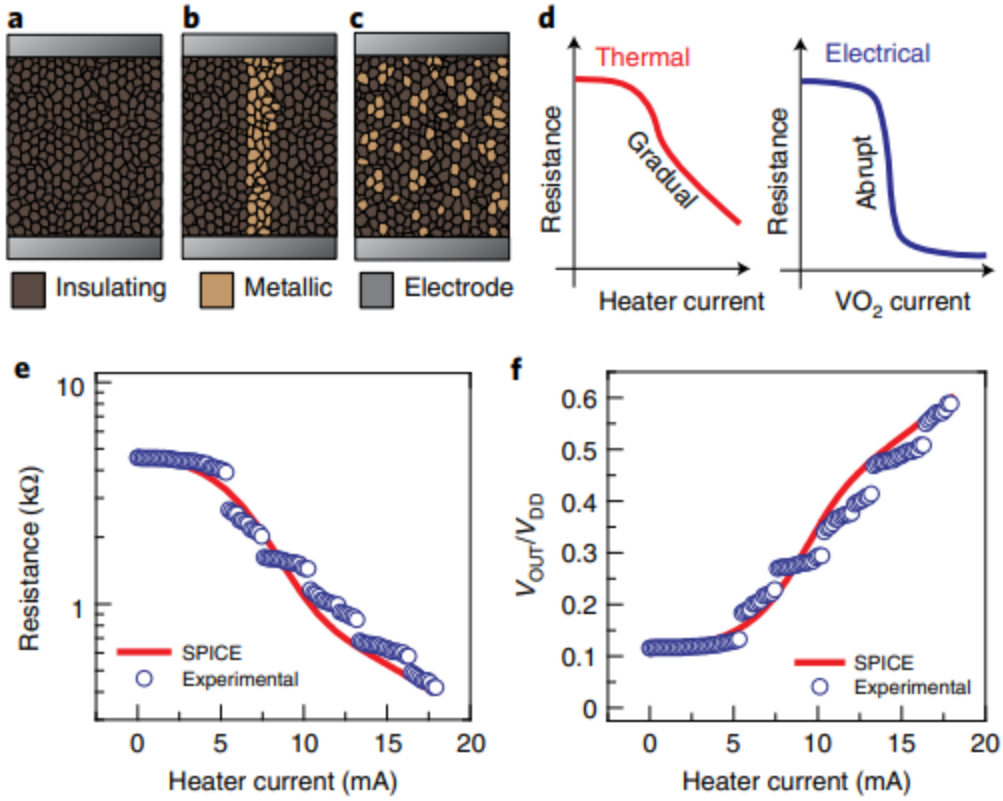


Figure 6.2: Switching mechanisms of VO_2 gap. a–c, Schematics show a VO_2 gap with no bias (a), filamentary switching (b) and thermal-driven switching (c). d, As compared to thermal-driven domain-wise switching, electrical filamentary switching shows an abrupt change in resistance. e, Resistance of the VO_2 device as a function of heater current showing ~77 levels. It shows gradual and linear resistive switching when the input current is larger than 5 mA. f, Voltage ratio between the output (V_{OUT}) and the supply voltage (V_{DD}) of the device as a function of heater current with a 1,900-Ω-load resistor as a representative example. Symbols are experimental data and lines are SPICE simulation results. The V–I characteristic is similar to the ReLU function shown in **Fig. 6.1e**.

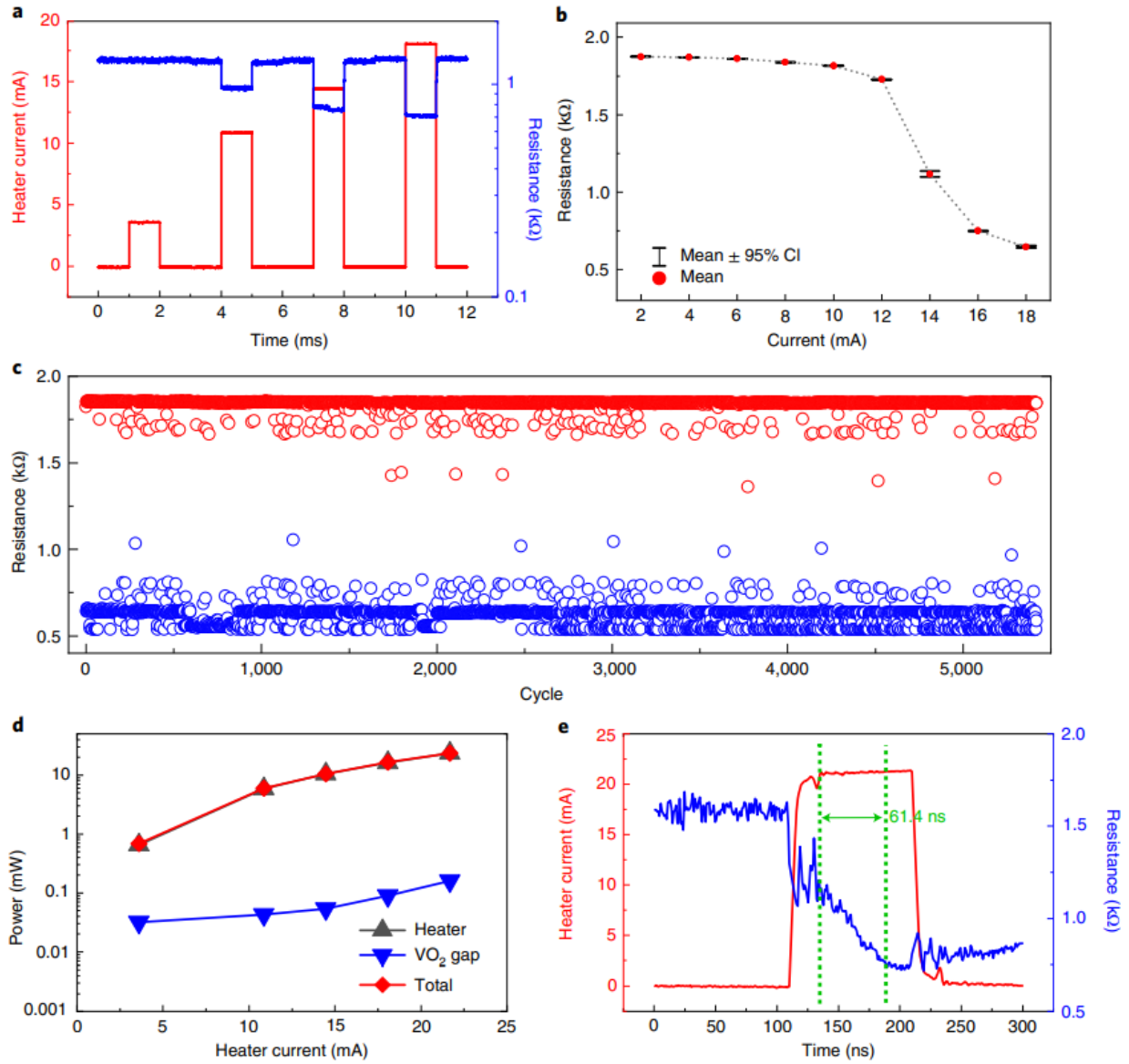


Figure 6.3: Electrical characteristics of the Mott ReLU device. a, Resistance of the VO_2 gap when a current pulse is applied to the heater. The resistance stays at a low resistance state only when the bias is applied. b, Cycle-to-cycle (or intra-device) variation of each resistance state of the Mott ReLU device. For each data point, the heater cools down to set the resistance of the VO_2 gap back to the no-bias case before applying another bias to the heater. The circle symbol represents the mean value while the error bars represent a 95% critical interval (CI). c, Endurance of the Mott ReLU device. The state of the device is alternately switched between the highest (red symbols) and lowest resistance states (blue symbols) by flowing 0 mA and 18 mA current through the heater, respectively. d, Power consumption of each component of the Mott device (that is, the heater and the VO_2 gap) with various heater currents. The power consumption of the Mott device is dominated by the heater. e, Heater current applied to the device and the resistance of the VO_2 gap as a function of time. 61.4 ns after the input to the Mott device is stabilized, the output of the Mott device becomes stable, as indicated by the green dashed lines.

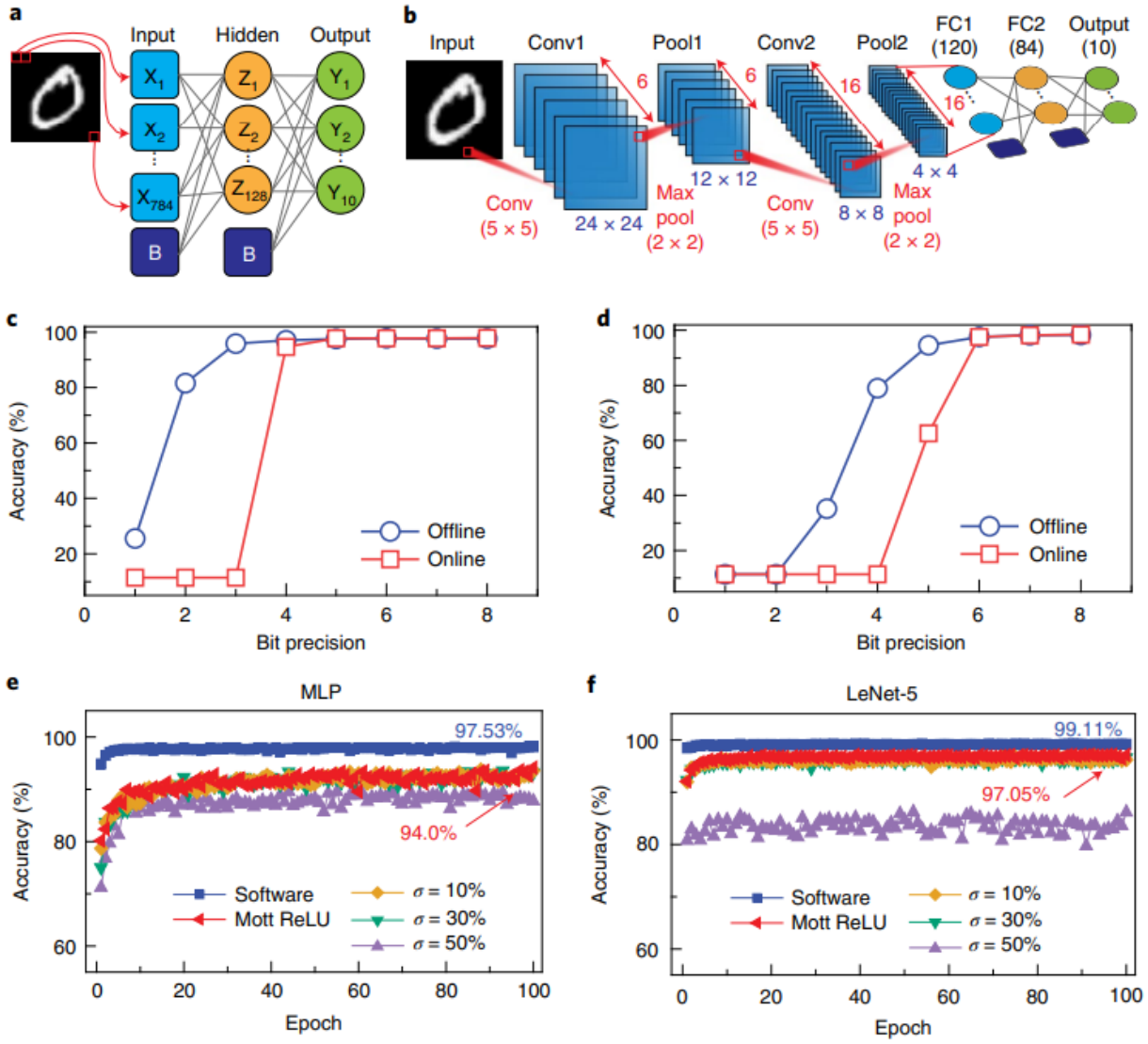


Figure 6.4: Network-level implementations. a,b, A schematic of MLP (a) and LeNet-5 (b) networks used for simulations with the Mott ReLU. MLP consists of an input layer (X), a hidden layer (Z) and an output layer (Y) with bias (B) for the input and hidden layers. MLP has one ReLU layer and LeNet-5 has four ReLU layers after convolutional (Conv) and fully connected (FC) layers. c,d, Accuracy of MLP (c) and LeNet-5 (d) with the ReLU activation function for offline classification (blue circle symbol) and online learning (red square symbol). The ReLU activation function is quantized to have 1- to 8-bit precision. MLP needs 5-bit precision while LeNet-5 requires 6-bit precision to prevent the accuracy drop. e,f, The network simulation results for MLP (e) and LeNet-5 (f) for the whole MNIST set for each epoch (60,000 images). Experimental measurement results from **Fig. 6.2e** and **f** are used for these simulations. The Mott ReLU achieves an accuracy comparable to the ideal ReLU implemented in software (blue square symbol) unless the cycle-to-cycle (or intra-device) variation of the Mott ReLU device (σ) is higher than 50%. Red triangle, yellow diamond, green triangle and purple triangle symbols represent results for the no variation, σ= 10%, σ= 30% and σ= 50% cases, respectively.

Figure 6.5: System-level benchmarking results. a–c, An illustration of a synaptic core and neuron peripheral circuits implemented with conventional digital circuits (a), Mott ReLU circuits (b) and analogue CMOS ReLU circuits (c). The Mott ReLU device can replace the ADCs, adder, shift register and neuron peripheral circuits. The CBRAM synaptic core has a wordline (WL) decoder (DEC), bitline (BL) switch matrix and source line (SL) switch matrix. In contrast to the CMOS analogue activation circuit (ACT), the Mott ReLU device can be integrated for each column due to its small size. d, Peripheral energy versus peripheral area in different technology nodes for CBRAM synaptic core with CMOS ReLU peripheral for LeNet-5 implementation. A CBRAM synaptic core with digital ADC peripherals (green square symbol) and Mott ReLU is also presented as a reference. The parameters for different technology nodes of CMOS circuits are adopted from the predictive technology model[28,29]. The Mott ReLU continues to provide substantial gains in energy and area even though the CMOS is scaled down to a 14 nm node. The star symbol shows performance results using experimentally measured Mott ReLU characteristics, while the black square symbol shows projected performance results using an optimized Mott ReLU device (that is, the thermal resistance of the heater is increased by ×10 and the parasitic capacitance is below 10^{-11} F). The system-level energy consumption using the optimized Mott ReLU can be further reduced by ~50 times.

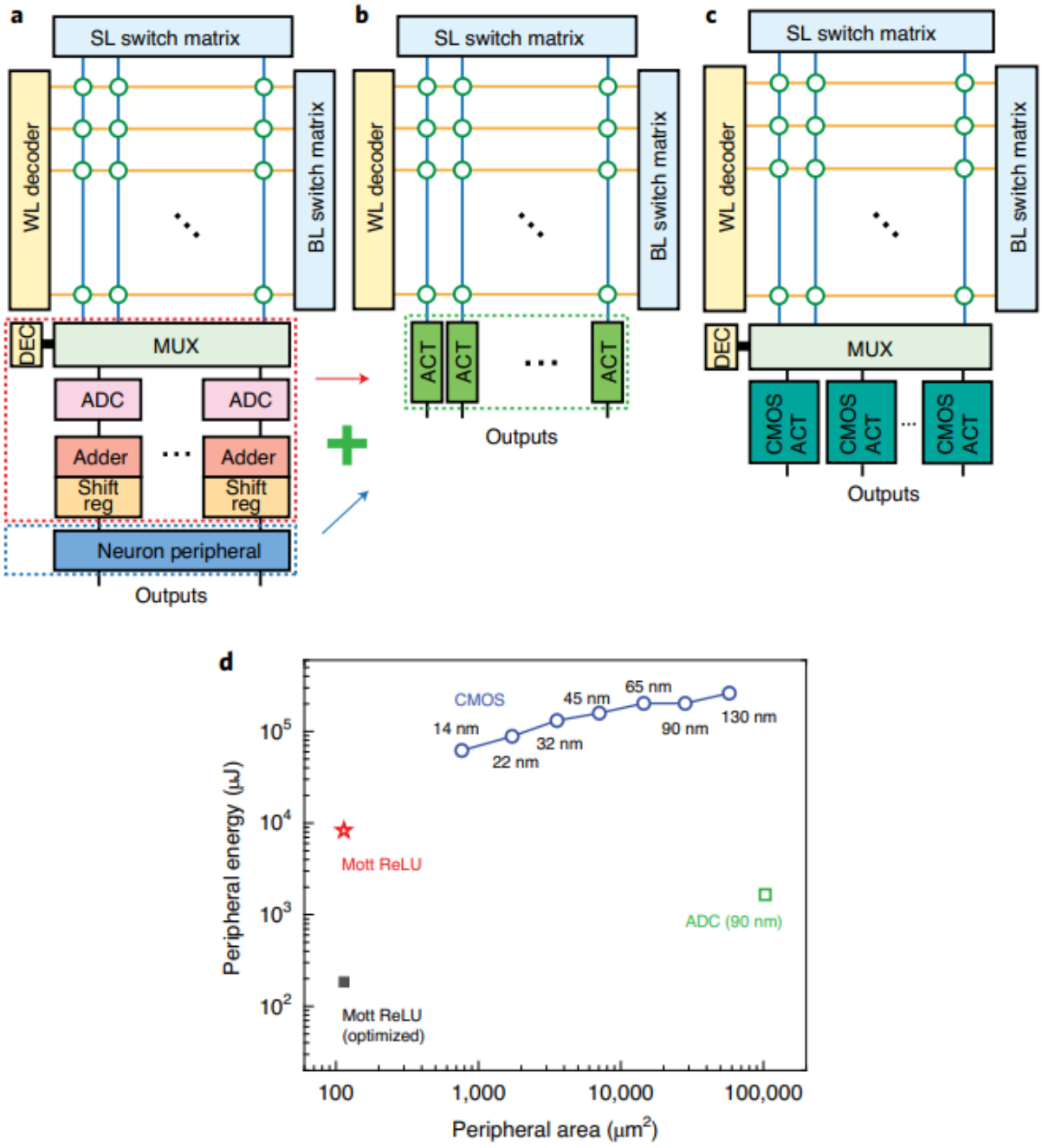

a
SL switch matrix
WL decoder
BL switch matrix
DEC
MUX
ADC
ADC
Adder
Shift reg
Adder
Shift reg
Neuron peripheral
Outputs
b
SL switch matrix
WL decoder
BL switch matrix
ACT
ACT
ACT
Outputs
c
SL switch matrix
WL decoder
BL switch matrix
DEC
MUX
CMOS ACT
CMOS ACT
CMOS ACT
Outputs
d
Peripheral energy (μJ)
10^5
10^4
10^3
10^2
CMOS
14 nm
22 nm
32 nm
45 nm
65 nm
90 nm
130 nm
Mott ReLU
ADC (90 nm)
Mott ReLU (optimized)
100
1,000
10,000
100,000
Peripheral area (μm²)

Figure 6.6: Hardware demonstration of the integration of Mott ReLU devices and a synaptic array. a, An optical image (scale bar, 150 μm) of a CBRAM crossbar array (32 × 32) and the scanning electron microscope image of a 16 × 16 CBRAM array (scale bar, 200 μm). We use a 16 × 16 array for the following hardware implementation. b, A Mott ReLU device array that contains 44 devices (scale bar, 3 mm). The insets in a and b (scale bars, 20 μm and 30 μm, respectively) show single devices, the CBRAM and Mott ReLU device, respectively. c, Image of the custom PCB board with the Mott ReLU and the CBRAM arrays wire bonded onto it to demonstrate neural network operation. d, An illustration explains how a Mott ReLU device is connected to a column of the CBRAM array with a load resistor (Load R) in hardware. e, Output voltage of the Mott ReLU device as the input voltage (V_{IN}) to the CBRAM array is swept from –250 mV to 250 mV when ~2/3 of devices on a column of the CBRAM array are set to a low resistance state while the others are set to a high resistance state. For the Mott ReLU device, 1.1 V is applied as V_{DD} to the VO_2 gap with a 3.3-kΩ-load resistor connected in series, and 7 mA of offset current is applied to the heater. f, The measured output voltage of a Mott ReLU device when the percentage of CBRAM devices at the low resistance state is varied from 0% to 100%. g, A 180 × 270 image used for edge detection. Colour bar represents the pixel intensity of the image. Four representative 10 × 10 patches and a schematic of the convolution operation are shown below. The schematic illustrates that the convolution operation is done by sliding the 4 × 4 filters on the image patches 49 times. h,i, For a lateral filter (h) and vertical filter (i), the experimentally measured weighted sum currents of the CBRAM array during the convolution operations for these four patches are shown. The weighted sum current produced by the CBRAM array during the convolution operation is fed to the Mott ReLU array to perform ReLU operation. j,k, Panels j and k show the output voltage of the Mott ReLU devices for the whole image during the convolution and ReLU operations for the lateral and vertical filters, respectively. Colour bar represents the output voltage of the Mott ReLU.

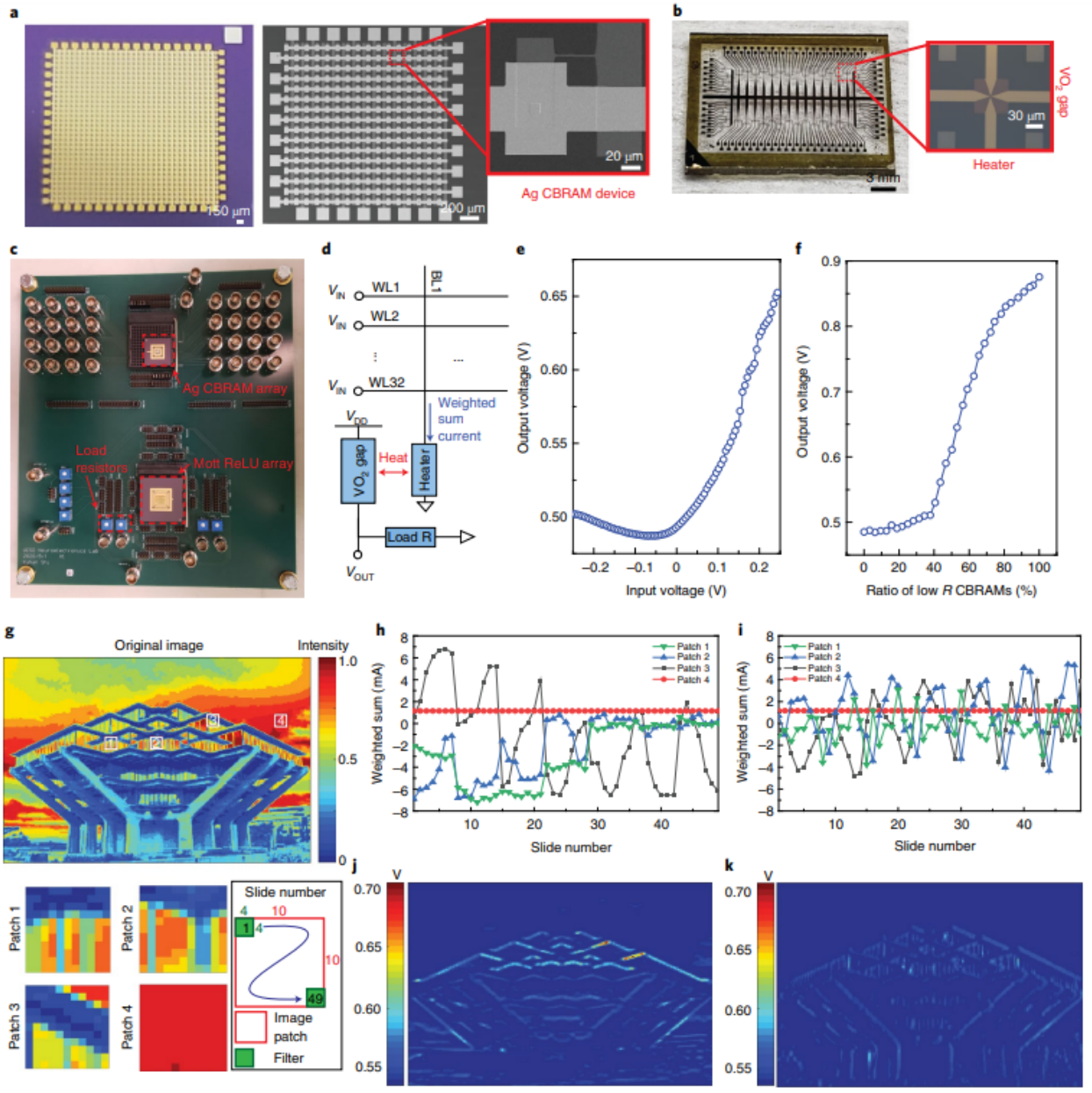

a
150 μm
200 μm
20 μm
Ag CBRAM device
b
3 mm
30 μm
Heater
c
Ag CBRAM array
Load resistors
Mott ReLU array
d
WL1
WL2
WL32
BL1
Weighted sum current
Heat
Heater
Load R
e
Output voltage (V)
Input voltage (V)
f
Output voltage (V)
Ratio of low R CBRAMs (%)
g
Original image
Intensity
Patch 1
Patch 2
Patch 3
Patch 4
Slide number
Image patch
Filter
h
Weighted sum (mA)
Slide number
i
Weighted sum (mA)
Slide number
j
V
k
V

Table 6.1: The performance of the activation device or circuit. Comparison of Mott ReLU, analogue CMOS ReLU[14], and digital ADC with reconfigurable function mapping[15] at single ReLU level. The energy, latency, and leakage power are evaluated from the experimental measurement results shown in **Fig. 6.S2a** and **b**. For the energy estimation, we used a 65 ns pulse for the Mott ReLU case.

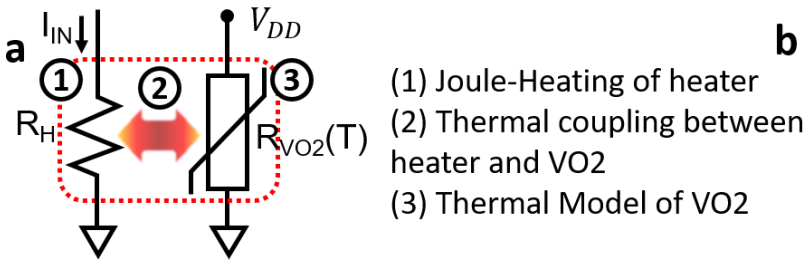
	Mott	**Analogue CMOS[14]**	**Digital ADC[15]**
Energy (Exp./Optimal, pJ)	199.5/0.638[a]	3410	19.4
Latency (Exp./Optimal, ns)	61.4/3.8[a]	91.91	207
Area (μm^2)	0.64	951.06	289[b]
Leakage (μW)	27.0	11060	-

[a] Shows projected optimal energy and latency when the thermal resistance of the heater is increased by ×10 and the parasitic capacitance of a Mott ReLU is $<10^{-11}$F.

[b] This area is only the area per neuron circuit. The digital ADC implementation needs a shared circuit, which occupies 0.086 mm^2 of area.

Table 6.2: Network simulation result. The accuracy results of MLP and LeNet-5 for ideal software (64 bit), 64-bit weights with Mott ReLU (~6 bit) and CBRAM (~5-bit weights) with Mott ReLU (~6 bit). The results show that the Mott ReLU can achieve accuracy comparable to the ideal software ReLU.

	Online Learning		Offline Classification	
	MLP	LeNET-5	MLP	LeNET-5
Software (64 bit)	97.53 %	99.11 %	97.53 %	99.11 %
Mott ReLU (~6 bit)	94.0 %	97.05 %	94.42 %	98.38 %
CBRAM (~5 bit) with Mott ReLU (~6 bit)	84.2 %	94.21 %	89.97 %	98.35 %



b

Parameter	Value
R_H (Ω)	36
R_{th_H} (K/W)	1.22×10^{3}
C_{th_H} (J/K)	3.96×10^{-14}
R_{ox} (K/W)	1.24×10^{5}
C_{ox} (J/K)	1.33×10^{-15}
R_{th_VO2} (K/W)	8.89×10^{5}
C_{th_VO2} (J/K)	3.74×10^{-15}

Figure 6.S1: **a** A schematic illustrates the compact thermal model used for the Mott ReLU device. The model consists of (1) the thermal model for the heater which addresses Joule-heating of the heater, (2) thermal coupling between the heater and VO_2, and (3) thermal model of VO_2. **b** Parameters used for the SPICE model (**Supplementary Note 2**) in our work.

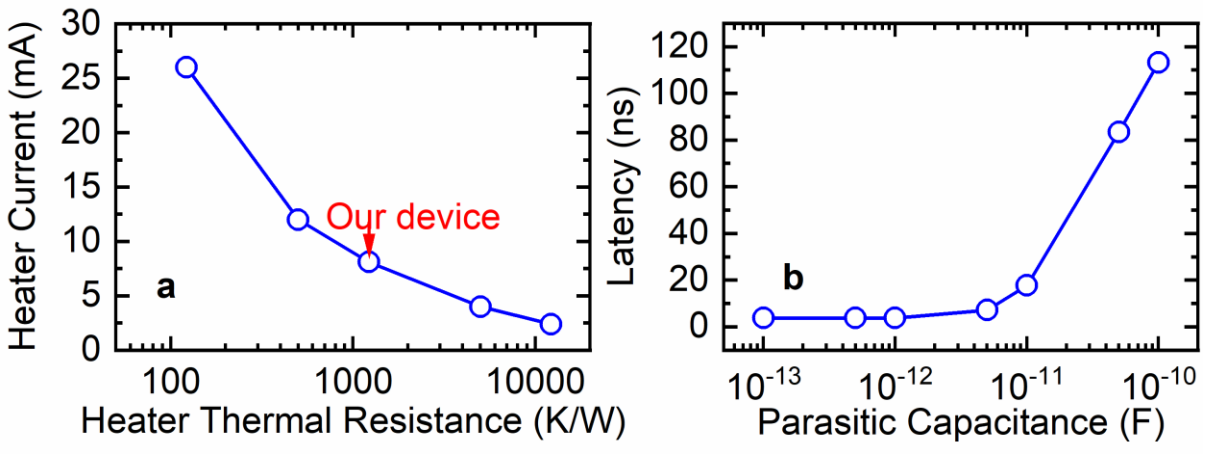


Figure 6.S2: a Heater current required to set the resistance of the VO_2 gap to 1 kΩ with various thermal resistance of the nanowire heater. As the thermal resistance of the heater is increased by ×10, the required heater current to achieve the same resistance is reduced by ×3.4. b Latency of the Mott ReLU device with various parasitic capacitance. ~3.8 ns of latency can be achieved when the parasitic capacitance is minimized $<10^{-11}$ F.

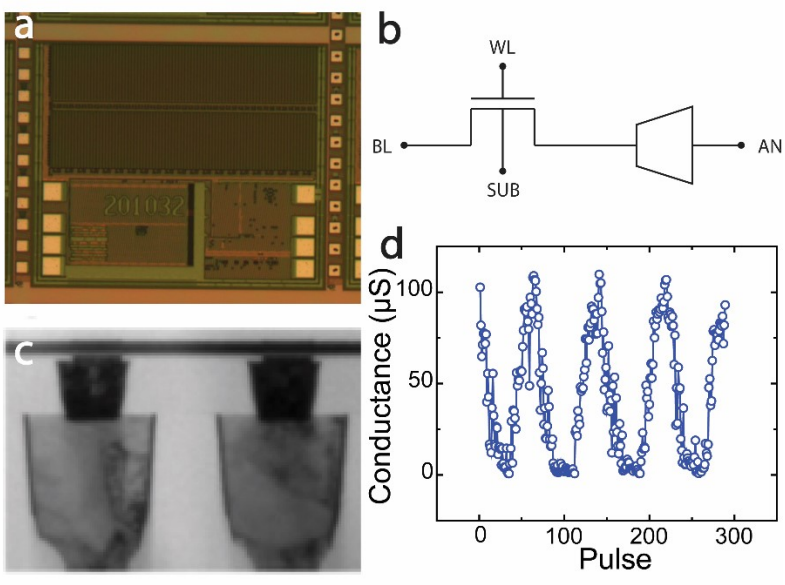


Figure 6.S3: **a** A picture, **b** schematic (1T1R architecture), **c** TEM image, and **d** gradual switching behaviour of a CBRAM cell. CBRAM cells can provide gradual weight tuning for both programming and erasing over many cycles.

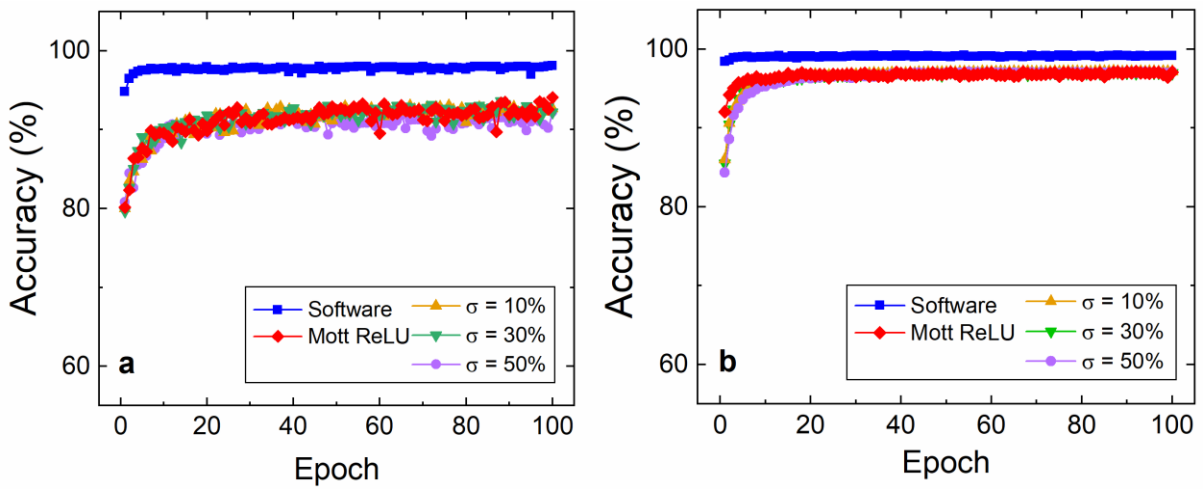


Figure 6.S4: The accuracy of **a** MLP and **b** LeNet-5 for the whole MNIST set for each epoch (60k images) with device-to-device (or inter-device) variations. The resistance is shifted by ΔR ($= N(0, \sigma) \times R_{min}$) which is determined for each device at the beginning of the simulation and fixed during the rest of the training. When the resistance of the device is variated for MLP, the accuracy does not degrade up to 50% of variations. Similarly, LeNet-5 with variations on the device resistance shows no accuracy degradation up to 50% of variations.

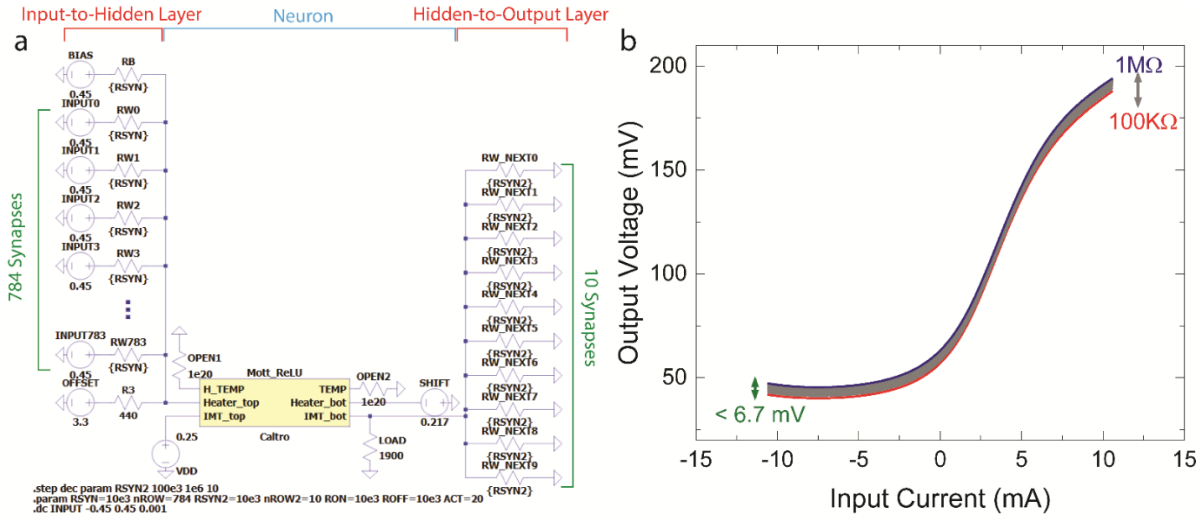


Figure 6.S5: **a** A schematic of the circuit used for SPICE simulations of a Mott ReLU device in the MLP shown in **Fig. 6.4a**. A Mott ReLU device gets a weighted sum current from 785 synaptic devices on the input-to-hidden layer and drives 10 synaptic devices on the hidden-to-output layer. **b** Output voltage of Mott ReLU connected to 10 synaptic devices when the input current is ranged from –10 mA to 10 mA (with 5 mA of offset applied by an additional row with a 3.3 V input and 440Ω resistor). The input current is generated by a column with 785 synaptic devices. When the resistance values of the synaptic devices are changed from 100kΩ to 1MΩ, the output voltage change is ~6.7 mV or less.

a Lateral Filter

1	1	1	1
0	0	0	0
0	0	0	0
-1	-1	-1	-1

b Vertical Filter

1	0	0	-1
1	0	0	-1
1	0	0	-1
1	0	0	-1

c

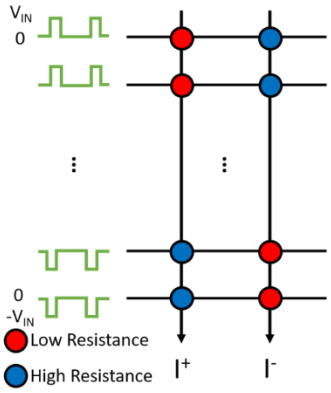


Figure 6.S6: **a** Lateral and **b** Vertical edge detection filters used for convolutional filtering operation are shown in **Fig. 6.6g** and **h** in the main text. **c** The filters are mapped to the CBRAM crossbar array using a differential pair scheme[30-32]. The devices at low resistance state are programmed to ~300 Ω while the devices at high resistance state are programmed to ~1 GΩ.

Table 6.6.S1: System-level Benchmarking Results. The system-level benchmarking results computed by NeuroSim with the Mott ReLU devices, analogue CMOS ReLU circuits, and digital CMOS peripherals for offline classification of MLP and LeNet-5. The synaptic arrays are implemented with 130 nm technology node. The activation functions are implemented based on the energy, area, and leakage power of activation function circuits in **Table 6.1**. For a read operation, we used a 65 ns pulse for Mot ReLU. Mott ReLU achieves significant performance gains over analogue CMOS ReLU and digital ADC peripherals.

Offline Classification						
Network	MLP			LeNet-5		
Activation	Mott	CMOS[14] (Analogue)	CMOS[15] (Digital)	Mott	CMOS[14] (Analogue)	CMOS[15] (Digital)
Energy (μJ)	249.7	1856	165.07	8347.1	139280	1784.5
Latency (ms)	1.3	20.17	41.55	420	4028	8301
Peripheral Area (μm^2)	107.33	12757	94956	113.57	24491	103025
Peripheral Leakage (mW)	3.75	55.33	-	7.37	143.83	-

9. References

1 Wong, H. S. P., Lee, H.-Y., Yu, S., Chen, Y.-S., Wu, Y., Chen, P.-S., Lee, B., Chen, F. T. & Tsai, M.-J. Metal–Oxide RRAM. Proceedings of the IEEE 100, 1951-1970, doi:10.1109/jproc.2012.2190369 (2012).

2 Zidan, M. A., Strachan, J. P. & Lu, W. D. The future of electronics based on memristive systems. Nature Electronics 1, 22-29, doi:10.1038/s41928-017-0006-8 (2018).

3 Kang, D.-H., Kim, J.-H., Oh, S., Park, H.-Y., Dugasani, S. R., Kang, B.-S., Choi, C., Choi, R., Lee, S., Park, S. H., Heo, K. & Park, J.-H. A Neuromorphic Device Implemented on a Salmon-DNA Electrolyte and its Application to Artificial Neural Networks. Advanced Science 6, 1901265, doi:10.1002/advs.201901265 (2019).

4 Ge, R., Wu, X., Kim, M., Shi, J., Sonde, S., Tao, L., Zhang, Y., Lee, J. C. & Akinwande, D. Atomristor: Nonvolatile Resistance Switching in Atomic Sheets of Transition Metal Dichalcogenides. Nano Lett 18, 434-441, doi:10.1021/acs.nanolett.7b04342 (2018).

5 van de Burgt, Y., Lubberman, E., Fuller, E. J., Keene, S. T., Faria, G. C., Agarwal, S., Marinella, M. J., Alec Talin, A. & Salleo, A. A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing. Nature Materials 16, 414, doi:10.1038/nmat4856 (2017).

6 Zhao, X., Liu, S., Niu, J., Liao, L., Liu, Q., Xiao, X., Lv, H., Long, S., Banerjee, W., Li, W., Si, S. & Liu, M. Confining Cation Injection to Enhance CBRAM Performance by Nanopore Graphene Layer. Small 13, 1603948, doi:10.1002/smll.201603948 (2017).

7 Chakrabarti, B., Lastras-Montano, M. A., Adam, G., Prezioso, M., Hoskins, B., Payvand, M., Madhavan, A., Ghofrani, A., Theogarajan, L., Cheng, K. T. & Strukov, D. B. A multiply-add engine with monolithically integrated 3D memristor crossbar/CMOS hybrid circuit. Sci Rep 7, 42429, doi:10.1038/srep42429 (2017).

8 Kim, S., Choi, B., Yoon, J., Lee, Y., Kim, H. D., Kang, M. H. & Choi, S. J. Binarized Neural Network with Silicon Nanosheet Synaptic Transistors for Supervised Pattern Classification. Sci Rep 9, 11705, doi:10.1038/s41598-019-48048-w (2019).

9 Oh, S., Huang, Z., Shi, Y. & Kuzum, D. The Impact of Resistance Drift of Phase Change Memory (PCM) Synaptic Devices on Artificial Neural Network Performance. IEEE Electron Device Letters 40, 1325-1328, doi:10.1109/LED.2019.2925832 (2019).

10 He, K., Zhang, X., Ren, S. & Sun, J. in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 770-778 (2016).

11 Kataeva, I., Ohtsuka, S., Nili, H., Kim, H., Isobe, Y., Yako, K. & Strukov, D. in 2019 IEEE International Symposium on Circuits and Systems (ISCAS). 1-5.

12 Gao, B., Bi, Y., Chen, H.-Y., Liu, R., Huang, P., Chen, B., Liu, L., Liu, X., Yu, S., Wong, H. S. P. & Kang, J. Ultra-Low-Energy Three-Dimensional Oxide-Based

Electronic Synapses for Implementation of Robust High-Accuracy Neuromorphic Computation Systems. ACS Nano 8, 6998-7004, doi:10.1021/nn501824r (2014).

13 Yang, T.-J. & Sze, V. in 2019 IEEE International Electron Devices Meeting (IEDM). 22.21.21-22.21.24.

14 Krestinskaya, O., Salama, K. N. & James, A. P. Learning in Memristive Neural Network Architectures Using Analog Backpropagation Circuits. IEEE Transactions on Circuits and Systems I: Regular Papers 66, 719-732, doi:10.1109/tcsi.2018.2866510 (2019).

15 Giordano, M., Cristiano, G., Ishibashi, K., Ambrogio, S., Tsai, H., Burr, G. W. & Narayanan, P. Analog-to-Digital Conversion With Reconfigurable Function Mapping for Neural Networks Activation Function Acceleration. IEEE Journal on Emerging and Selected Topics in Circuits and Systems 9, 367-376, doi:10.1109/JETCAS.2019.2911537 (2019).

16 Ambrogio, S., Narayanan, P., Tsai, H., Shelby, R. M., Boybat, I., di Nolfo, C., Sidler, S., Giordano, M., Bodini, M., Farinha, N. C. P., Killeen, B., Cheng, C., Jaoudi, Y. & Burr, G. W. Equivalent-accuracy accelerated neural-network training using analogue memory. Nature 558, 60-67, doi:10.1038/s41586-018-0180-5 (2018).

17 Stefanovich, G., Pergament, A. & Stefanovich, D. Electrical switching and Mott transition in VO2. Journal of Physics: Condensed Matter 12, 8837 (2000).

18 Qazilbash, M. M., Brehm, M., Chae, B. G., Ho, P. C., Andreev, G. O., Kim, B. J., Yun, S. J., Balatsky, A. V., Maple, M. B., Keilmann, F., Kim, H. T. & Basov, D. N. Mott transition in VO2 revealed by infrared spectroscopy and nano-imaging. Science 318, 1750-1753, doi:10.1126/science.1150124 (2007).

19 Del Valle, J., Salev, P., Tesler, F., Vargas, N. M., Kalcheim, Y., Wang, P., Trastoy, J., Lee, M. H., Kassabian, G., Ramirez, J. G., Rozenberg, M. J. & Schuller, I. K. Subthreshold firing in Mott nanodevices. Nature 569, 388-392, doi:10.1038/s41586-019-1159-6 (2019).

20 Madan, H., Jerry, M., Pogrebnyakov, A., Mayer, T. & Datta, S. Quantitative Mapping of Phase Coexistence in Mott-Peierls Insulator during Electronic and Thermally Driven Phase Transition. ACS Nano 9, 2009-2017, doi:10.1021/nn507048d (2015).

21 LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278-2324 (1998).

22 Radu, I. P., Govoreanu, B., Mertens, S., Shi, X., Cantoro, M., Schaekers, M., Jurczak, M., De Gendt, S., Stesmans, A., Kittl, J. A., Heyns, M. & Martens, K. Switching mechanism in two-terminal vanadium dioxide devices. Nanotechnology 26, 165202, doi:10.1088/0957-4484/26/16/165202 (2015).

23 Del Valle, J., Salev, P., Kalcheim, Y. & Schuller, I. K. A caloritronics-based Mott neuristor. Sci Rep 10, 4292, doi:10.1038/s41598-020-61176-y (2020).

24 Shi, Y., Nguyen, L., Oh, S., Liu, X., Koushan, F., Jameson, J. R. & Kuzum, D. Neuroinspired unsupervised learning and pruning with subquantum CBRAM arrays. Nature Communications 9, 5312, doi:10.1038/s41467-018-07682-0 (2018).

25 Chen, P.-Y., Peng, X. & Yu, S. NeuroSim: A Circuit-Level Macro Model for Benchmarking Neuro-Inspired Architectures in Online Learning. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 37, 3067-3080, doi:10.1109/tcad.2018.2789723 (2018).

26 Shrivakshan, G. & Chandrasekar, C. A comparison of various edge detection techniques used in image processing. International Journal of Computer Science Issues (IJCSI) 9, 269 (2012).

27 Amer, S., Hasan, M. S., Adnan, M. M. & Rose, G. S. SPICE Modeling of Insulator Metal Transition: Model of the Critical Temperature. IEEE Journal of the Electron Devices Society 7, 18-25, doi:10.1109/JEDS.2018.2875627 (2019).

28 Zhao, W. & Cao, Y. Predictive technology model for nano-CMOS design exploration. ACM Journal on Emerging Technologies in Computing Systems (JETC) 3, 1 (2007).

29 Zhao, W. & Cao, Y. New generation of predictive technology model for sub-45 nm early design exploration. IEEE Transactions on Electron Devices 53, 2816-2823 (2006).

30 Bayat, F. M., Prezioso, M., Chakrabarti, B., Nili, H., Kataeva, I. & Strukov, D. Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits. Nat Commun 9, 2331, doi:10.1038/s41467-018-04482-4 (2018).

31 Li, X., Tang, J., Zhang, Q., Gao, B., Yang, J. J., Song, S., Wu, W., Zhang, W., Yao, P., Deng, N., Deng, L., Xie, Y., Qian, H. & Wu, H. Power-efficient neural network with artificial dendrites. Nature Nanotechnology 15, 776-782, doi:10.1038/s41565-020-0722-5 (2020).

32 Yao, P., Wu, H., Gao, B., Tang, J., Zhang, Q., Zhang, W., Yang, J. J. & Qian, H. Fully hardware-implemented memristor convolutional neural network. Nature 577, 641-646, doi:10.1038/s41586-020-1942-4 (2020).

10. Acknowledgements

This work was supported by Office of Naval Research (N000142012405 and N00014162531), Samsung Electronics, the National Science Foundation (ECCS-1752241, ECCS-2024776 and ECCS-1734940), the National Institutes of Health (R21 EY029466, R21 EB026180 and DP2 EB030992) and Qualcomm Fellowship. The experimental aspects of this work were supported as part of the Quantum Materials for Energy Efficient Neuromorphic Computing (Q-MEEN-C) Energy Frontier Research Center (EFRC), funded by the US Department of Energy, Office of Science, Basic Energy Sciences under award #DE-SC0019273. The fabrication of the devices was performed at the San Diego Nanotechnology Infrastructure (SDNI) of the University of California San Diego, supported by the National Science Foundation (ECCS-1542148).

Chapter 6, in full, is a reprint of the material as it appears in Nature Nanotechnology, 2021, Oh, Sangheon; Shi, Yuhan; Valle, Javier Del; Salev, Pavel; Lu, Yichen; Huang, Zhisheng; Kalcheim; Yoav; Schuller; Ivan K; Kuzum, Duygu, Energy-efficient Mott activation neuron for full-hardware implementation of neural networks, 2021. The dissertation author was the primary author of this paper.

CONCLUSION

In conclusion, this dissertation presents energy-efficient hardware implementations of neuromorphic computing using PCM, subquantum CBRAM, Ag-based CBRAM, and CuO_x-based RRAM. Our findings demonstrate that these devices can be used to improve the performance and energy efficiency of deep learning applications. This dissertation also presents the Mott activation neuron integrated with eNVM synaptic arrays. Activation function, which usually implemented with large circuitry blocks or a general processor unit, is implemented with much smaller size and higher energy efficiency than other conventional approaches. Our approach opens new avenues in implementing deeper and more complex network architecture with higher area and energy efficiency using eNVM based synaptic arrays and Mott-ReLU activation devices.

There are several directions in which this work can be extended in the future. Some potential areas of further exploration include: (1) Investigating the potential of using other types of eNVM devices, especially RRAMs based on bulk[1] or interfacial switching[2] rather than filamentary switching, in the hardware implementation of neural networks. Deterministic switching from bulk switching mechanism instead of stochastic switching from filamentary switching will eliminate the need of repetitive write-and-verify loops[1], which is a significant overhead on RRAM programming. (2) Developing new algorithms and techniques for training neural networks on non-volatile memory hardware, with the aim of improving their energy efficiency and performance. There is a recent work on this direction that uses activity-difference-based training algorithm instead of conventional backpropagation training algorithm[3]. With a training algorithm optimized for eNVM characteristics, the performance and energy efficiency

can be improved by several orders of magnitude[3]. Overall, the future work in this area has the potential to significantly advance the field of neural network hardware implementation, and to enable the development of more efficient and effective artificial intelligence systems.

References

1 Li, Y., Fuller, E. J., Sugar, J. D., Yoo, S., Ashby, D. S., Bennett, C. H., Horton, R. D., Bartsch, M. S., Marinella, M. J., Lu, W. D. & Talin, A. A. Filament-Free Bulk Resistive Memory Enables Deterministic Analogue Switching. Adv Mater 32, e2003984, doi:10.1002/adma.202003984 (2020).

2 Li, X., Wu, H., Bin, G., Wu, W., Wu, D., Deng, N., Cai, J. & Qian, H. Electrode-induced digital-to-analog resistive switching in TaOx-based RRAM devices. Nanotechnology 27, 305201, doi:10.1088/0957-4484/27/30/305201 (2016).

3 Yi, S.-i., Kendall, J. D., Williams, R. S. & Kumar, S. Activity-difference training of deep neural networks using memristor crossbars. Nature Electronics, doi:10.1038/s41928-022-00869-w (2022).