Dynamic Memristors: from Devices to Applications

by

Wen Ma

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering)
in the University of Michigan
2018

Doctoral committee:

Professor Wei D. Lu, Chair
Associate Professor Emmanouil Kioupakis
Associate Professor Zhengya Zhang
Associate Professor Zhaohui Zhong

Wen Ma

wenma@umich.edu

ORCID iD: 0000-0002-3144-5243

ACKNOWLEDGEMENTS

Firstly, I would like to thank my PhD advisor, Prof. Wei D. Lu, for his great support and guidance in my course of study and numerous research projects. He is always very kind and optimistic, and trying to provoke students to think not only about the technical details but also the big picture of the projects. Secondly, I want to thank my thesis committee members: Prof. Kioupakis, Prof. Zhang, and Prof. Zhong. They have provided very valuable suggestions and comments on my thesis revision. Thirdly, I will not forget the enormous help I got from my previous and current group members: Siddharth Gaba, Patrick Sheridan, Ting Chang, Lin Chen, Jiantao Zhou, Chao Du, Ugo Otuonye, Shinhyun Choi, Yuchao Yang, Bing Chen, Taeho Moon, Sungho Kim, Fuxi Cai, Jihang Lee, Yeonjoo Jeong, Jonghoon Shin, Qiwen Wang, Seunghwan Lee, John Moon, Billy Schell. Fourthly, I want to express my gratitude to the LNF and MC^2 staff, especially Dr. Pilar Herrera-Fierro who helped me in the recipe development of the sulfurization wet chemistry process, and Allen Hunter, who helped with the real-time SEM sessions. In the end, I want to say many thanks to my parents, who currently live in China and always support me through difficult and happy moments in both work and life.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Memristors have been extensively studied as a promising candidate for next generation non-volatile memory technology. More recently, memristors have also become extremely popular in neuromorphic applications because of their striking resemblance to biological synapses. The memristor was firstly proposed conceptually as the fourth basic electric circuit element whose resistance is dependent on the history of electrical stimulation. Physical implementations of memristors are normally based a solid state, nanoscale metal-insulator-metal (MIM) sandwich structure, and the resistance change is achieved by controlling the ion (either cation or anion) redistribution inside the insulating/switching layer. Specifically, a conductive filament can be formed with a high-concentration of metal cations or oxygen vacancies, leading to an increase in device conductance during set, and a decrease in device conductance during reset associated with the annihilation of the filament. Devices based on such resistive switching mechanisms are often termed resistive random-access memory (RRAM) devices, and offer advantages of simple structure, high density, low power, good endurance, etc. for memory and computing applications.

In this dissertation, two kinds of memristor devices will be discussed, using Ag_2S and WO_x as the switching material, respectively. The WO_x device allows incremental modulation of the device conductance, and enables efficient hardware emulation of important synaptic learning functions including paired pulse facilitation, sliding threshold effect, rate dependent plasticity and spike timing dependent plasticity (Chapter 3), showing the resemblance between memristors and biological synapses. Neural networks based on the memristor crossbar array have been used

to successfully perform image reconstruction tasks based on the sparse coding algorithm (Chapter 2). A 32×32 WO_x memristor crossbar array was used for vector-matrix multiplication acceleration, and the device non-ideality effects in the memristor crossbar array on the image reconstruction performance were examined. Additionally, interesting short-term decay dynamics can be observed in both Ag_2S and WO_x based devices. Different from the requirements of non-volatile memory which aims for long term memory storage, the volatile nature of these devices can be used to directly encode and process temporal information. Specifically, the Ag_2S memristor can encode different neural spiking information in the temporal domain into analog switching probability distributions (Chapter 5). These devices are termed "dynamic memristors" and can be applied in novel computing schemes such as reservoir computing systems for efficient temporal information processing including speech recognition (Chapter 4). Both devices show very promising properties for neuromorphic computing – overcoming the von-Neumann bottleneck by incorporating information processing into memory storage. It is believed in the future, very efficient neuromorphic computing chips can be designed and implemented using these memristors that offer potential advantages in terms of area consumption, computing speed and power consumption.

Chapter 1 Introduction

1.1 Background

The von Neumann architecture, also known as the von Neumann model and Princeton architecture, is a computer architecture proposed by John von Neumann in 1945 [1], as shown in Figure 1.1.

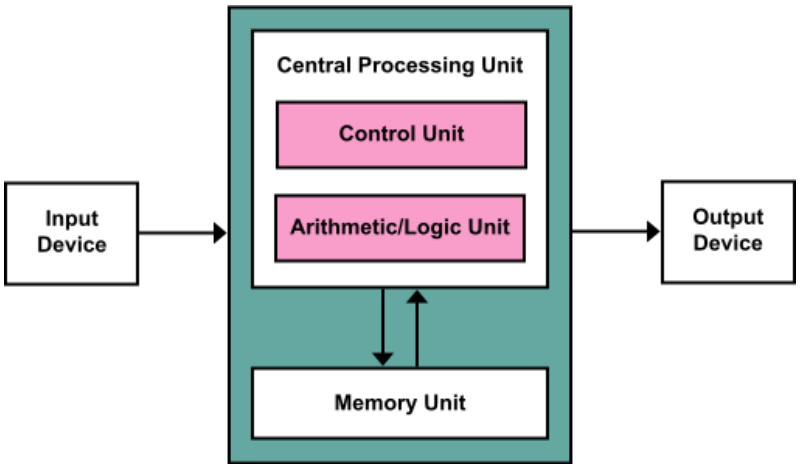


Figure 1.1 Von Neumann architecture scheme

The von Neumann architecture has been the backbone of today's computers and along with the Moore's law led the revolution in electronics we have witnessed in the past several decades. The separation of the processing unit and the memory unit, however, leads to the von Neumann bottleneck [2]. Specifically, the bandwidth of the bus between the central processing unit (CPU) and the memory is much lower than the rate at which the CPU can work, and with

large amounts of data, the CPU is forced to wait for needed data to be transferred to or from memory. As CPU speed and memory size have increased much faster than the bandwidth between them, the bottleneck has become more severe with newer generations of processors.

One method to address the von Neumann bottleneck is using neuromorphic engineering, or neuromorphic computing, a concept originally developed by Carver Mead in the 1980s [3]. The key idea of neuromorphic computing is to mimic the human brain, specifically the neuro-biological architectures in the nervous system, using electronic components such as analog/digital circuits in very-large-scale-integration (VLSI) systems. It is sometimes also called brain-inspired or bio-inspired computing due to the emphasis on implementing models of neural systems and understanding how the morphology of individual neurons, synapses, circuits, and overall architecture leads to desirable computations. Recently, implementation of neuromorphic computing on the hardware level have been attempted using oxide-based memristors [4, 5], Mott insulators [6], and transistors. In particular, memristors with incremental conductance updates and high connectivity can be used to effectively emulate the synaptic connections in neural networks and have started drawing growing attention [4, 15].

1.2 Memristors

Memristors are normally (with gross simplification) as resistors with memory. The memristor concept was first envisioned by circuit theorist Leon Chua in 1971 [8] as a non-linear two-terminal electrical component relating the electrical charge and the magnetic flux (Figure 1.2). Since the flux/charge depends on the history of the applied voltage/current, a memristor's resistance (termed memristance) is not constant but depends on the history of the current/voltage that has been applied on the device.

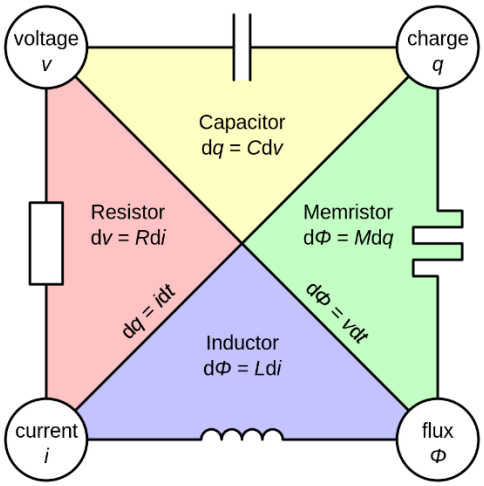


Figure 1.2 Conceptual symmetries of resistor, capacitor, inductor and memristor

In 2008, HP Labs made a solid state resistive switching device based on a metal-TiOx-metal structure [9] and linked the physical device to the concept of memristor. The resistive switching behavior is characterized with a pinched hysteresis loop [10], which was described by Leon Chua as the signature of memristors. One example of the pinched hysteresis loop is shown in Figure 1.3. Such resistive switching (RS) devices have in fact been studied previously [7], and are generally referred to as resistive random-access memory (RRAM), although prior studies focused on physical characterizations and device optimizations without realizing the linkage between the RS effects and the memristor theoretical framework. The switching materials in a memristor can vary from silicon/silicon dioxide, perovskites, transition metal oxides to chalcogenides. Other than RRAM, MRAM (magnetic random-access memory, spintronic), FeRAM (ferroelectric), and PCM (phase change memory) may be argued to fall in the broadly-defined memristor category too as they also show resistive switching and pinched hysteresis

effects [11]. These devices are promising for applications such as non-volatile memories, logic circuits and neuromorphic computing architectures [11].

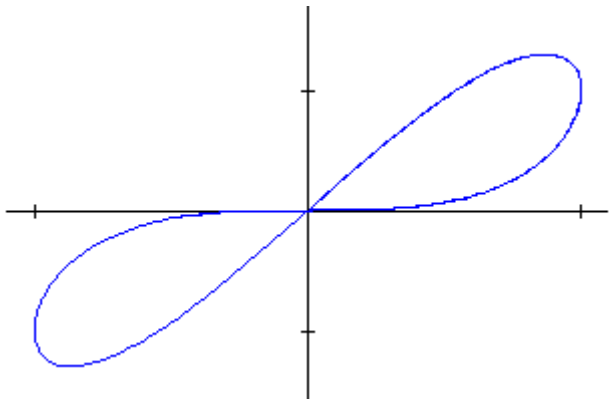


Figure 1.3 Example of a pinched hysteresis curve in current versus voltage plot.

1.3 Memristor Genres

As early as in the 1960s, different materials based on the metal-insulator-metal (MIM) configurations have been reported to show resistive switching effects, which is the fingerprint of memristors. The insulator can be binary or multinary oxides, chalcogenides as well as organic materials. In the majority of devices, the switching mechanism is based on the formation/annihilation of a conductive filament during set/reset that changes the device from a high-resistance state (HRS) to a low-resistance state (LRS) and vice versa. Careful material and device characterizations have verified that the filament formation is based on ionic transport and electrochemical redox reactions inside the switching layer [7].

One class of such nanoionics devices operate through the migration of anions, typically oxygen ions in an oxide-based switching layer. During set, the oxygen ions were driven towards the anode (or equivalently, oxygen vacancies are driven towards the cathode). The redistribution

of anions, e.g. accumulation of oxygen vacancies during set, changes the local stoichiometry of the transition metal oxide and leads to changes of the local conductivity and the overall device conductance (Figure 1.4). This type of device is called valency change memory (VCM). A good example is the WO_x based memristor device, which are used by us to build neuromorphic systems and will be discussed in the first part of the thesis.

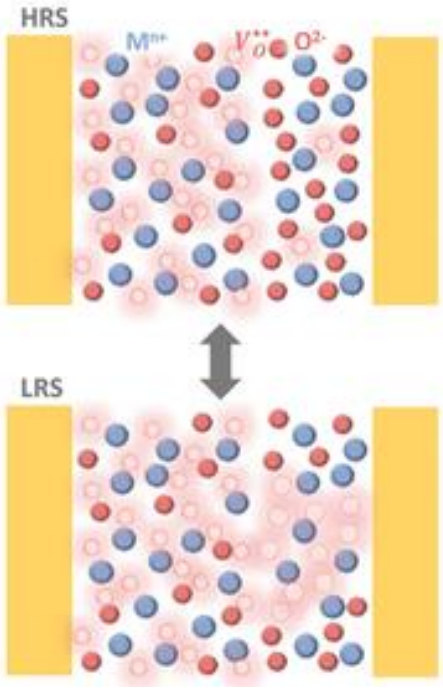


Figure 1.4 Schematic illustration of the resistive switching process in VCM devices. Under a high applied field, oxygen vacancies migrate from a V_O-rich reservoir layer into the near-stoichiometric switching layer that initially has high resistance. As the V_O concentration in the switching layer increases, percolated paths for electron conduction can form at high V_O concentrations, and switch (set) the device from a high-resistance "off" state to a low-resistance "on" state. From [37].

With the WO_x memristive device, incremental device conductance increase/decrease, similar to the potentiation/depression of synapses in neuromorphic systems, can be achieved by

incrementally adding or removing oxygen vacancies in the filament region. Importantly, this incremental conductance modulation can be obtained conveniently by controlling the number (or amplitude) of the applied voltage pulses, as shown in Figure 1.5. Specifically, by applying a train of positive/negative voltage pulses the memristor conductance is gradually strengthened/weakened, leading to a behavior analogous to synaptic potentiation/depression.

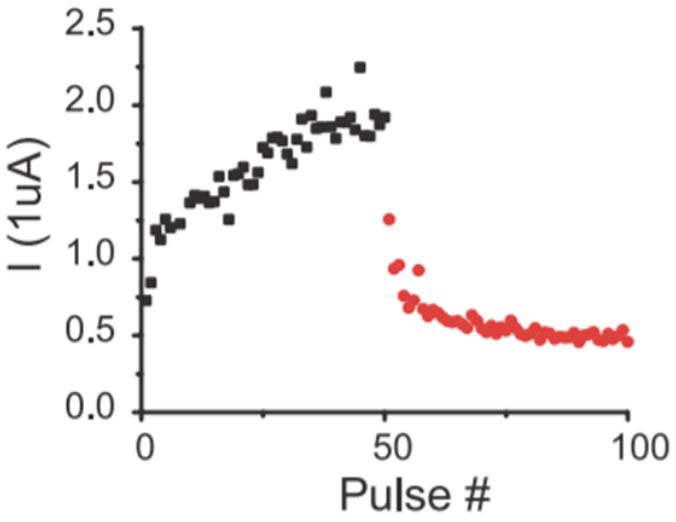


Figure 1.5 Incremental conductance change in a WO_x memristor. A train of 50 1.7 V/80 µs potentiating pulses are followed by a train of 50 -1.7 V/80 µs depressing pulses. The device conductance was measured with a read pulse of 0.5V/3 ms after each potentiation (black) or depression (red) pulse.

Another class of nanoionics-based memristor devices relies on the electrochemical metallization effect [37]. Here an active electrode material (usually Ag or Cu) is used, and the oxidation, transportation and reduction of the metal cations in solid electrolyte film lead to the filament formation/rupture. This type of device is often called conductive bridge RAM

(CBRAM). A schematic showing the filament growth processes in CBRAM is shown in Figure 1.6.

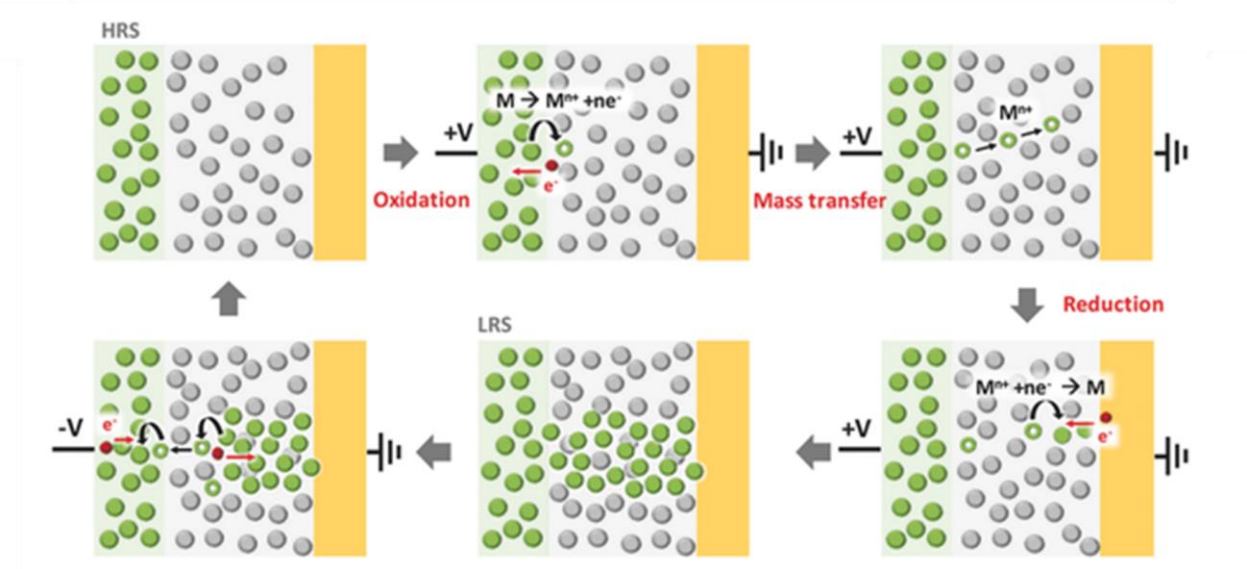


Figure 1.6 Schematic illustration of the resistive switching process in CBRAM devices. From [37]. First, the active anode material (e.g., Ag or Cu) becomes ionized into Ag or Cu cations under an external voltage bias. Second, facilitated by the applied electric field, the Ag or Cu cations migrate toward the inert electrode through the solid electrolyte. Finally, the displaced Ag or Cu cations become reduced to elemental Ag or Cu atoms after capturing electrons, leading to the nucleation and growth of metal clusters within the switching layer through nucleation and deposition processes, eventually resulting in the creation of nanoscale metal filaments that lead to low-resistance state (LRS). The reverse process, where a segment of the filament is broken by physically and chemically removing a portion of the filament material, leads to the reset process that switches the device back to the high-resistance state (HRS).

Ag_2S was among the earliest materials used in CBRAM devices. We focused on the low-programming voltage and dynamic switching characteristics of Ag_2S-based memristors. The device characteristics and applications will be discussed in the second part of this thesis. Figure 1.7 shows similar incremental conductance modulations with a Ag_2S memristor, using voltage pulse trains with low voltage amplitude and short durations.

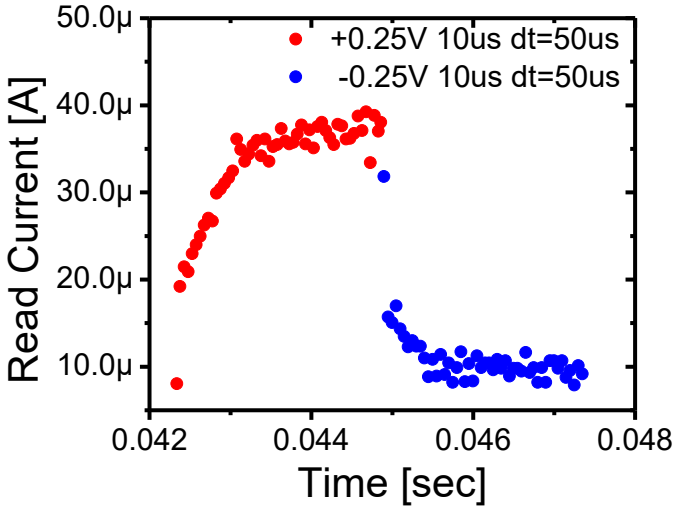


Figure 1.7 Incremental conductance changes in a Ag_2S memristor. The data points are the read currents measured with a read pulse of 0.05 V/10 µs after each potentiation (red) or depression (blue) pulse.

1.4 Memristors as Artificial Synapses

In the biological nervous system, a synapse forms the connection between neurons that modulates electrical or chemical signal transmission [12]. It is widely believed that the synaptic connectivity plays a key role in the formation of memory and information processing. The connection between two neurons can be strengthened or weakened, and the connectivity patterns help lead to memory formation [13]. The process of synaptic strengthening is known as long-term potentiation (LTP) and synaptic weakening as long-term depression (LTD). In general, short term synaptic weight changes are usually controlled by the release of neurotransmitters (glutamate) that can bind to and help increase the activity of the receptors, while long term synaptic plasticity effects can be attributed to the dendritic growth and the increase in the number

of receptors modulated by the influx of calcium into the post-synaptic cell [14]. The different molecular neurobiological processes involved in synaptic plasticity are shown in Figure 1.8.

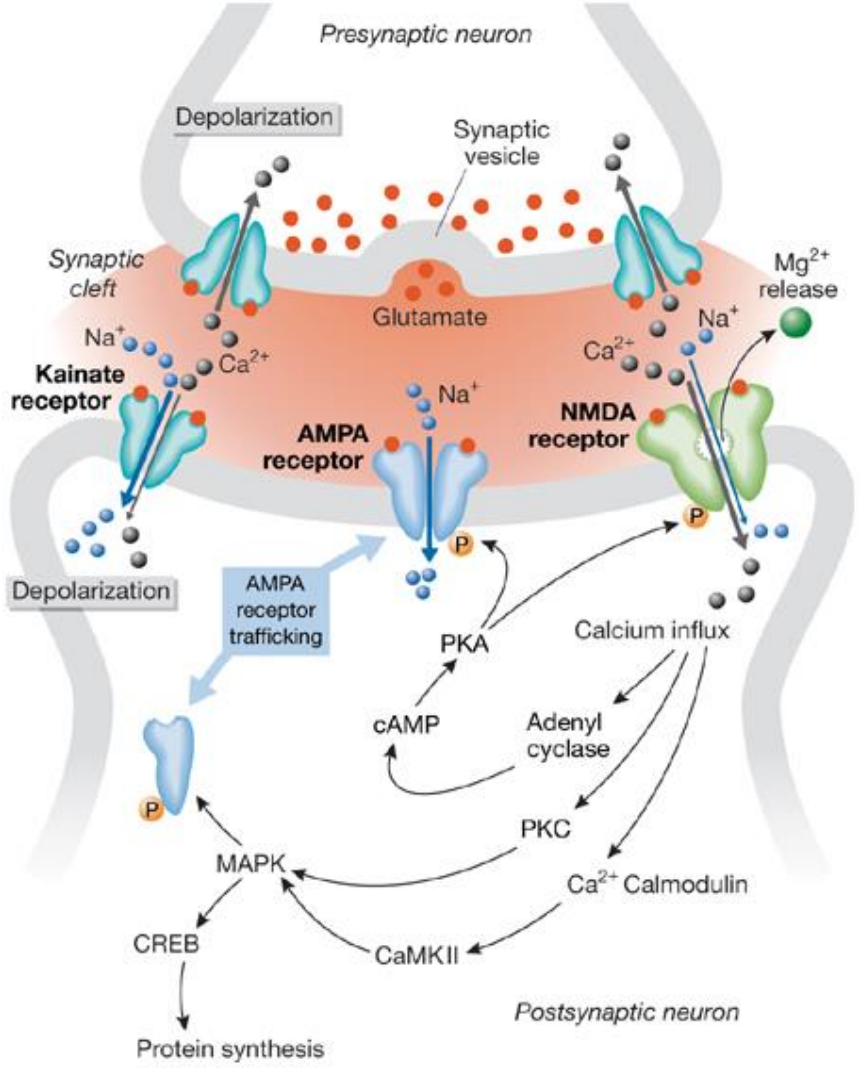


Figure 1.8 Different molecular processes involved in synaptic plasticity.

These complex molecular/ionic processes, driven by spikes from the pre- and post-synaptic neurons, lead to gradual synaptic weight changes and are often summarized empirically as different learning “rules”. In this regard, a memristor bears striking resemblance to a synapse

in the way that the conductance (weight) can be incrementally modified when stimulated by voltage pulses (spikes), driven by internal dynamic ionic processes in the memristor device [4, 15]. Many synaptic learning rules, such as rate dependent plasticity and spike timing dependent plasticity (STDP) have already been demonstrated with memristors [15-17], reinforcing the similarity between the two. Other synaptic behaviors have also been successfully emulated using either single memristor device or a small scale memristor network, including associative memory [18], heterosynaptic plasticity [19], and synaptic metaplasticity [20].

In biological nervous system, synapses enable a very large connectivity between neurons, which is believed to be critical for highly efficient, biologically inspired computing systems. For example, there are around 10^{11} neurons in the human brain and each of the neurons has approximately 7000 synaptic connections to others (on average) [22]. Using nanoscale memristors rather than conventional CMOS approaches to implement the synaptic connections in an artificial neural network can be very area and energy efficient [4, 9], making memristor-based approaches highly desirable for very large scale neuromorphic system implementations.

1.5 Neuromorphic Computing

Neuromorphic computing, or brain-inspired computing, has been steadily advancing over the past couple of decades. Using a conventional computing architecture, the same computing task, which can be achieved with human brain using only around 20 watts, would require supercomputers up to megawatts [36]. Bio-inspired neuromorphic computing offers potential for very low power consumption and high parallelism, and have been an active research topic recently.

BrainScaleS [24] and SpiNNaker [25] are two representative neuromorphic computing projects from the European Human Brain Project, originally funded by the European Commission's Future Emerging Technologies program from 2005 to 2015. These systems are mainly built as hardware accelerators to help test neuroscience models. While BrainScaleS use analog circuits, SpiNNaker's architecture focuses on a large number of digital cores. For BrainScaleS, each 20cm diameter silicon wafer contains around 200,000 neurons and 49 million synapses, and learning can be accelerated tens of thousands times faster than real time. SpiNNaker is built to mimic the brain's biological structure and behavior as well. It uses SDRAM to hold synaptic weights. With more than one million cores and 1000 simulated neurons per core, it can potentially simulate one billion neurons in real time, which is a little over 1% of human brain's estimated 85 billion neurons. Early work on both machines focused on the self-organization in neural networks, also time- and energy-efficiency optimization.

In the US, the most notable neuromorphic computing project is IBM's TrueNorth Chip [23], introduced in 2014 and developed under the DARPA SyNAPSE (Systems of Neuromorphic Adaptive Plastic Scalable Electronics) project. The chip has 4096 cores, each one simulating 256 neurons for a total of just over a million neurons. In addition, each neuron has 256 synapses, rendering a total number of over 268 million synapses. The chip has about the same number of neurons as a rodent, and requires the use of 5.4 billion transistors. The first demonstration was published in 2014, and used binary synaptic weights and was not capable of performing online training. A follow up paper [26] in 2016 demonstrated the mapping of convolutional neural networks, which is a deep learning model for image recognition, to the TrueNorth chip. Other similar works include HRL's neuromorphic chip [27] and Stanford's Neurogrid [28].

More recently, Intel also introduced its first neuromorphic chip, dubbed the Loihi test chip [29], consisting of 128 computing cores. Each core has 1024 neurons giving the chip in total of more than 130,000 neurons and 130 million synapses. The chip transmits data through patterns of spikes between neurons, using the concept of spiking neural networks where information encoding is done using the frequency of spikes or the timing between spikes. Initial testing has shown the chips are capable of applications like path planning and dictionary learning.

Since hardware implementation of neuromorphic systems is largely limited by the ability to implement the synaptic functions with large connectivity, incorporating memristors into neuromorphic systems has become an attractive alternative compared to the CMOS-implementations discussed above. Under the SyNAPSE project, we performed the first experimental demonstration showing the potential of memristors to emulate synaptic functions [4]. Such concept has now been adopted by numerous groups in the US and around the world [21, 38]. In US companies, IBM has attempted to use phase change memory (PCM) as a replacement for transistors based neurons [30], and Hewlett Packard labs continued their research on memristors/RRAM and plans to use them in their neuromorphic computing project as well as “in-memory computing systems [31].

The core of memristor-based neuromorphic computing architecture is a memristor crossbar array that can perform vector matrix multiplication directly in physics through a single parallel read operation [32]. Such vector matrix multiplication accelerates the execution of many machine learning algorithms, and tasks such as pattern classification [33], feature extraction [35] or image reconstruction [34] have been successfully demonstrated recently with memristor crossbar arrays.

1.6 Organization of Dissertation

This chapter discusses the basic concepts of memristors. Nanoionics based memristors, existing neuromorphic chips based on CMOS, and motivation of using memristors in neuromorphic computing systems to overcome the von-Neumann bottleneck have been introduced. Two types of memristors (WO_x and Ag_2S based) that will be studied in this thesis were briefly introduced in terms of their ability to implement incremental conductance change to emulate synapses.

In chapter 2, a 32×32 WO_x memristor crossbar arrays used in one of the machine learning applications - image reconstruction will be discussed. The device non-ideality effects from the memristor crossbar array on the reconstruction performance will be examined as well.

Chapter 3 discusses studies that show different synaptic behaviors and learning functions can be emulated with WO_x memristors with short term dynamics, demonstrating the resemblance between dynamic memristors and biological synapses. The concept and modeling of second-order memristor are also discussed.

Chapter 4 discusses another machine learning application – speech recognition, achieved using a WO_x memristor-based reservoir computing system. The short-term decay dynamics is essential for this task based on temporal information encoding. An autonomous signal generation task will also be discussed, as well analysis to validate the memristor based reservoir.

In Chapter 5, the short-term decay of the Ag_2S memristors is used to encode neural spiking information in the temporal domain. The dynamic memristor can encode the temporal information of spiking pulse trains into analog switching probability distributions.

Chapter 6 introduces two other projects that produced preliminary results so far. The first one is emulating the synaptic competition effect using a planar 1×n Ag_2S memristive network. The second one is training the WO_x memristor crossbar array for grid cell – place cell firing pattern generation, potentially useful for spatial navigation purposes.

In the end, Chapter 7 concludes the findings and summarizes this thesis.

References

[1] von Neumann, J. First draft of a report on the EDVAC. IEEE Ann. Hist. Comput. 15, 27–75 (1993).

[2] Backus, J. Can programming be liberated from the von Neumann style?: a functional style and its algebra of programs. Commun. ACM 21, 613–641 (1978).

[3] Mead, C. Neuromorphic Electronic Systems. Proc. IEEE 78, 1629–1636 (1990).

[4] Jo, S. H. et al. Nanoscale Memristor Device as Synapse in Neuromorphic Systems. Nano Lett. 10, 1297–1301 (2010).

[5] Yu, S., Wu, Y., Jeyasingh, R., Kuzum, D. & Wong, H. S. P. An electronic synapse device based on metal oxide resistive switching memory for neuromorphic computation. *IEEE Trans. Electron Devices* **58,** 2729–2737 (2011).

[6] Pickett, M. D., Medeiros-Ribeiro, G. & Williams, R. S. A scalable neuristor built with Mott memristors. Nat. Mater. 12, 114–117 (2012).

[7] Waser, R. & Aono, M. Nanoionics-based resistive switching memories. *Nat. Mater.* **6,** 833–840 (2007).

[8] Chua, L. O. Memristor—The Missing Circuit Element. IEEE Trans. Circuit Theory 18, 507–519 (1971).

[9] Strukov, D. B., Snider, G. S., Stewart, D. R. & Williams, R. S. The missing memristor found. Nature 459, 1154–1154 (2009).

[10] Chua, L. If it's pinched it's a memristor. Memristors Memristive Syst. 9781461490685, 17–90 (2014).

[11] Mazumder, P., Kang, S. M. & Waser, R. Memristors: Devices, models, and applications [scanning the issue]. Proc. IEEE 100, 1911–1919 (2012).

[12] Foster, M. & Sherrington, C.S. A Textbook of Physiology, volume 3 (7th ed.). London: Macmillan. p. 929 (1897).

[13] Attneave, F., B., M. & Hebb, D. O. The Organization of Behavior; A Neuropsychological Theory. Am. J. Psychol. 63, 633 (1950).

[14] Shouval, H. Z., Bear, M. F. & Cooper, L. N. A unified model of NMDA receptor-dependent bidirectional synaptic plasticity. Proc. Natl. Acad. Sci. U. S. A. 99, 10831–6 (2002).

[15] Chang, T. *et al.* Synaptic behaviors and modeling of a metal oxide memristive device. *Appl. Phys. A Mater. Sci. Process.* **102,** 857–863 (2011).

[16] Du, C., Ma, W., Chang, T., Sheridan, P. & Lu, W. D. Biorealistic Implementation of Synaptic Functions with Oxide Memristors through Internal Ionic Dynamics. Adv. Funct. Mater. 25, 4290–4299 (2015).

[17] Wang, Z. Q. *et al.* Synaptic learning and memory functions achieved using oxygen ion migration/diffusion in an amorphous InGaZnO memristor. *Adv. Funct. Mater.* **22,** 2759–2765 (2012).

[18] Pershin, Y. V. & Di Ventra, M. Experimental demonstration of associative memory with memristive neural networks. *Neural Networks* **23,** 881–886 (2010).

[19] Yang, Y., Chen, B. & Lu, W. D. Memristive Physically Evolving Networks Enabling the Emulation of Heterosynaptic Plasticity. *Adv. Mater.* **27,** 7720–7727 (2015).

[20] Zhu, X., Du, C., Jeong, Y. & Lu, W. D. Emulation of synaptic metaplasticity in memristors. *Nanoscale* **9,** 45–51 (2017).

[21] Yang, J. J., Strukov, D. B. & Stewart, D. R. Memristive devices for computing. Nat. Nanotechnol. 8, 13–24 (2012).

[22] Drachman, D. A. Do we have brain to spare? Neurology 64, 2004–2005 (2005).

[23] Merolla, P. A. et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. Science (80). 345, 668–673 (2014).

[24] Schemmel, J. *et al.* A wafer-scale neuromorphic hardware system for large-scale neural modeling. in *ISCAS 2010 - 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems* 1947–1950 (IEEE, 2010).

[25] S.B. Furber, F. Galluppi, S. Temple, L.A. Plana, The spinnaker project. Proc. IEEE 102, 652–665 (2014).

[26] Esser, S. K. et al. Convolutional Networks for Fast, Energy-Efficient Neuromorphic Computing. Proc. Natl. Acad. Sci. U. S. A. 113, 11441–11446 (2016).

[27] Cruz-Albrecht, J. M., Derosier, T. & Srinivasa, N. A scalable neural chip with synaptic electronics using CMOS integrated memristors. Nanotechnology 24, 384011 (2013).

[28] Benjamin, B. V. et al. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. Proc. IEEE 102, 699–716 (2014).

[29] Mayberry, M. Intel's New Self-Learning Chip Promises to Accelerate Artificial Intelligence. (2017). Available at: https://newsroom.intel.com/. (Accessed: 8th November 2017).

[30] Tuma, T., Pantazi, A., Le Gallo, M., Sebastian, A. & Eleftheriou, E. Stochastic phase-change neurons. Nat. Nanotechnol. 11, 693–699 (2016).

[31] Courtland, R. Can HPE's 'The Machine' Deliver? - IEEE Spectrum. Available at: https://spectrum.ieee.org/computing/hardware/can-hpes-the-machine-deliver. (Accessed: 8th November 2017).

[32] Hu, M. et al. Dot-product engine for neuromorphic computing. in Proceedings of the 53rd Annual Design Automation Conference on - DAC '16 1–6 (ACM Press, 2016).

[33] Prezioso, M. *et al.* Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* **521,** 61–64 (2015).

[34] Sheridan, P. M., Du, C. & Lu, W. D. Feature Extraction Using Memristor Networks. *IEEE Trans. Neural Networks Learn. Syst.* **27,** 2327–2336 (2016).

[35] Sheridan, P. M. *et al.* Sparse coding with memristor networks. *Nat. Nanotechnol.* **12,** 784–789 (2017).

[36] Ananthanarayanan, R., Esser, S. K., Simon, H. D. & Modha, D. S. The cat is out of the bag. in Proceedings of the Conference on High Performance Computing Networking, Storage and

Analysis - SC ’09 1 (ACM Press, 2009).

[37] Lee, J. & Lu, W. D. On-Demand Reconfiguration of Nanomaterials: When Electronics Meets Ionics. Adv. Mater. 1702770 (2017).

[38] Zidan, M. A., Chen, A., Indiveri, G. & Lu, W. D. Memristive computing devices and applications. J. Electroceramics 1–17 (2017).

Chapter 2 Image Reconstruction using WO_x Memristor Crossbar Array

2.1 Introduction

Memristor- or RRAM-based neuromorphic systems have attracted growing interest for energy efficient computing applications [1, 2]. Previously, rate-based and spike timing based synaptic learning has been demonstrated [7-13], and network-level fabrication/simulation has been explored [14-16] for functions such as pattern recognition [15]. Memristor crossbar arrays, as shown in Figure 2.1, offer the ability to perform vector-matrix multiplication operations directly in physics, and the ability to store offline learned basis functions (dictionary elements) or perform online learning in the network. These properties have made memristor crossbars promising candidates for implementing sparse coding and other machine learning algorithms to efficiently perform feature extraction and analysis, particularly for complex inputs such as images.

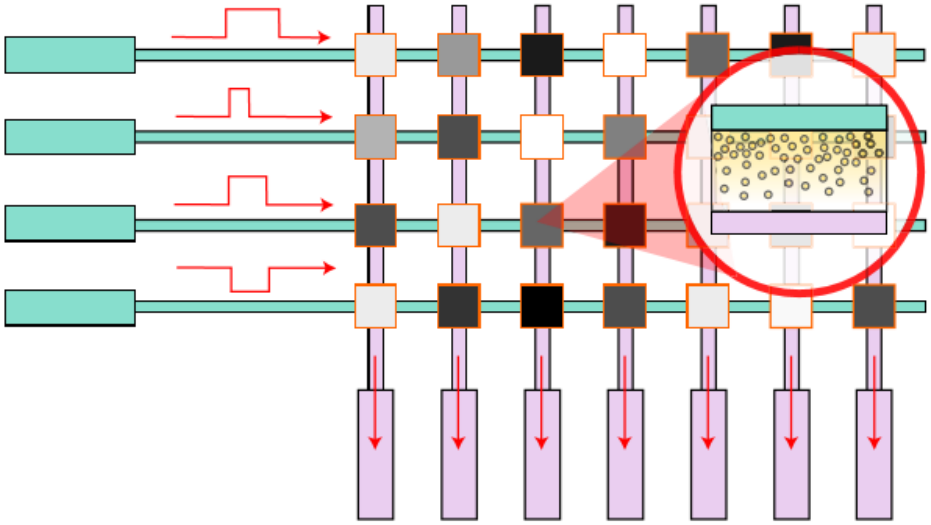


Figure 2.1 Schematic of a memristor crossbar-based computing system, showing the input neurons (green), the memristor crossbar array, and the leaky integrating output neurons (pink). A memristor is formed at each crosspoint, and can be programmed to different conductance states (represented in greyscale) by controlling the internal ion redistribution (inset).

One key factor enabling the learning/training process in neural networks is the ability to update synaptic weights in an analog manner through incremental conductance changes, as we originally demonstrated in 2010 [2]. An equivalence of analog memristors is using the several digital memristor devices in parallel with some stochastic nature [20], but the circuit overhead will be larger.

2.2 WO_x Memristors: Device Structure and Fabrication

The WO_x memristor device has a Pd top electrode, a WO_x switching layer and a W bottom electrode, shown in Figure 2.2. To fabricate the device, a 60 nm thick of W global layer was sputtered onto the SiO_2 substrate. Then the bottom electrode was patterned using the E-beam lithography and formed by dry etching using Ni as the hard mask. After removing Ni completely using wet etching in HCl solution, a rapid thermal oxidation is performed to partially oxidize the

W to form the switching layer. Then, top electrode was patterned using the E-beam lithography again and Pd was deposited by E-beam evaporation. After lift-off, the Pd top electrode was used as the etching mask to etch away the WO_x covering the bottom electrode.

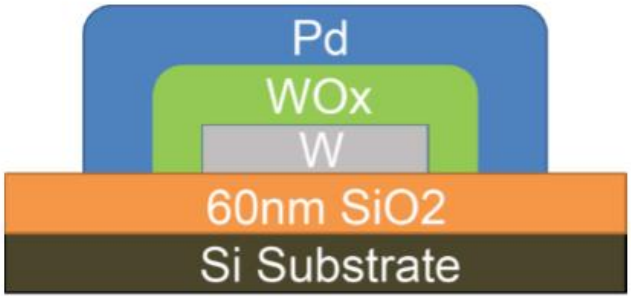


Figure 2.2 Schematic of the WO_x memristor structure, showing the partially oxidized W bottom electrode, the WO_x switching layer and the Pd top electrode.

It is believed that longer oxidation time will lead to more oxygen vacancies, such that as the tungsten bottom electrode was being oxidized, the top part of the WO_x film will have a higher oxygen vacancy concentration than the bottom portion, due to the longer elevated process experienced by the top portion of the film which was grown first [14]. The schematic of the oxygen vacancy distribution is shown in Figure 2.3.

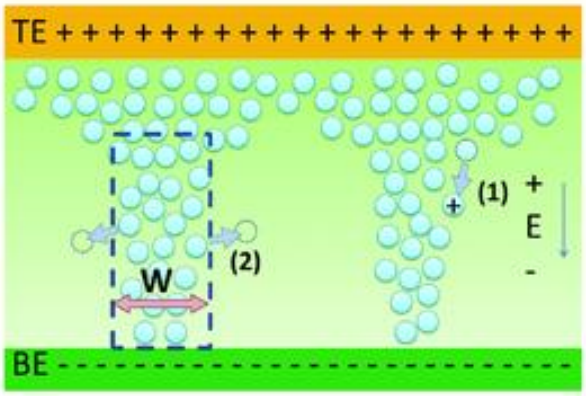


Figure 2.3 Schematic illustration of the internal V_O dynamics, showing (1) electric field driven V_O drift and (2) spontaneous diffusion.

Since oxygen vacancy (V_O) acts as dopants in the WO_x film and will increase the film's conductivity [5, 7], the top V_O rich portion will behave like an oxygen reservoir. So that when a positive voltage bias is applied onto the top electrode, the positively-charged oxygen vacancies drift towards the bottom electrode and will make the WO_x/W interface more ohmic, and the device conductance increases. On the other hand, a negative voltage will cause the device conductance to decrease. By controlling the amount of V_Os that have been redistributed, incremental conductance changes can be obtained, as shown in Figure 2.4.

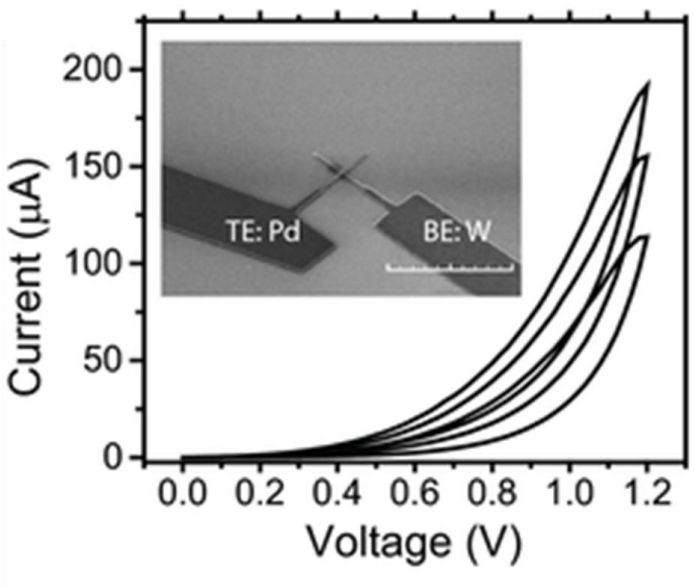


Figure 2.4 DC characteristics of the WO_x device showing pinched hysteresis and incremental conductance increase. For example, repeated positive voltages gradually increase the device conductance, as shown here. Inset: Scanning electron microscope image of a typical device. Scale bar: 20 μm.

2.3 Memristor Crossbar Hardware System

The memristor crossbar array, based on a Pd/WO_x/W device structure, was fabricated using similar methods described in the previous section. An SEM image of the finished crossbar array is shown in Figure 2.5.

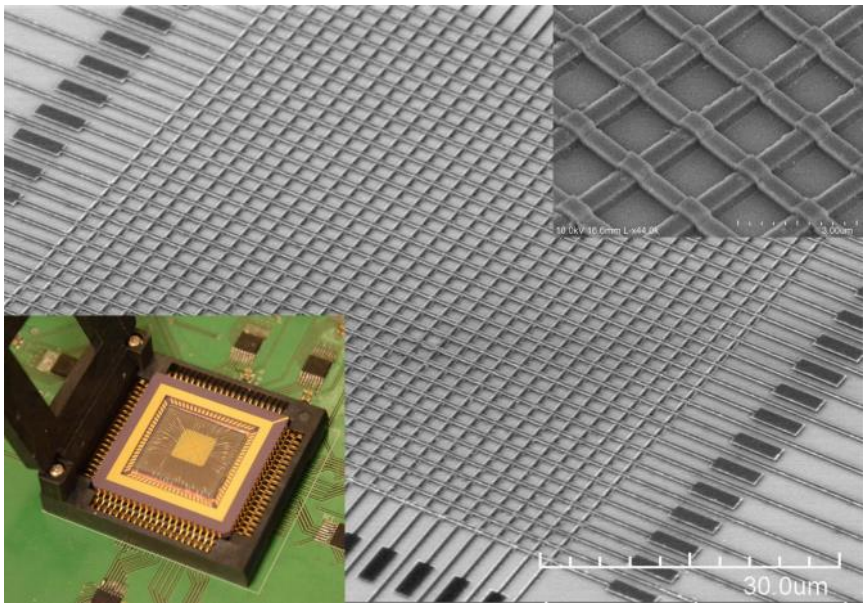


Figure 2.5 SEM image of the WO_x memristor array. Top right inset: zoom in view of the crossbar. Bottom left inset: crossbar array wire-bonded to a chip carrier.

To fabricate the WO_x crossbar shown in Figure 2.5, 60 nm of W was deposited with sputtering on 100 nm thermally grown oxide on a Si carrier wafer. The 500nm width bottom electrodes (BEs) were patterned by e-beam lithography and reactive ion etching (RIE) using Ni as a hard mask. The Ni hard mask was then removed by HCl etching. 250 nm of SiO_2 was then deposited using plasma enhanced chemical vapor deposition (PECVD) followed by RIE to form a spacer along the sidewalls of the BEs. The WO_x switching layer was then formed on the exposed W BEs using rapid thermal annealing (RTP) at 425°C for 60 seconds. The top electrodes (TEs) are Pd (90 nm)/Au (50 nm) patterned by e-beam lithography, e-beam evaporation and lift-off processes. A finished crossbar array was wire-bonded to an 84-pin chip carrier and measured using a custom-built measurement board, shown in Figure 2.6.

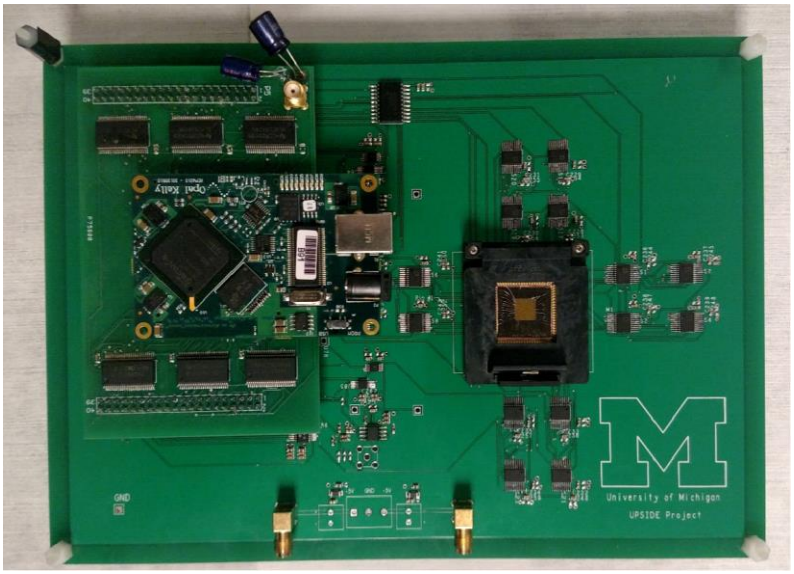


Figure 2.6 Measurement board image.

In the array, the WO_x memristor device at each cross point can be programmed into different conductance states, offering analog programming capabilities. As a test, a grayscale Mona-Lisa image was stored in the memristor array and read out using the memristor board without read-verify and reprogramming. The read out from the stored image, obtained from a read current from each crosspoint in the array, is shown in Figure 2.7.

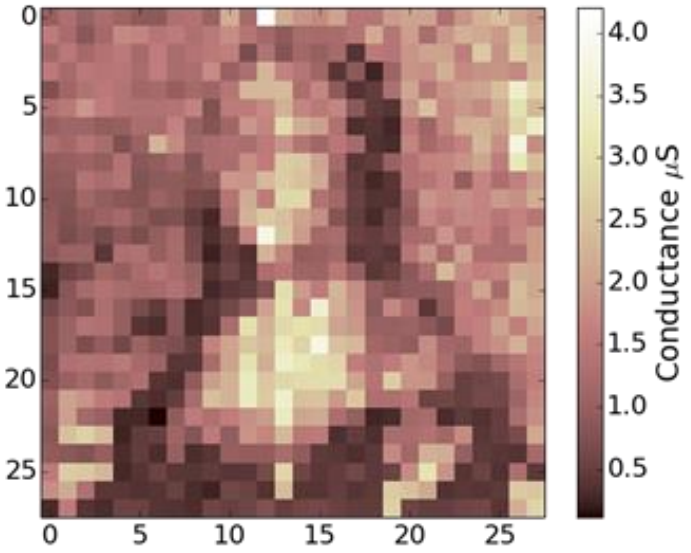


Figure 2.7 Image of Mona Lisa stored and read out from the WO_x memristor array

2.4 Sparse Coding Algorithm

Sparse representation reduces the complexity of the input signals and enables more efficient processing and storage, as well as improved feature extraction and pattern recognition functions [18, 19]. Given a signal *x*, which may be a vector (for example, representing the pixel values in an image patch), and a dictionary of features *D*, the goal of sparse coding is to represent *x* as a linear combination of features from *D* using a sparse set of coefficients *a*, while minimizing the number of features used. By doing so, the original input data can be compressed into a few features with relative weights, thus making it more efficient to transfer and store data. More importantly, sparse coding can identify the hidden features in the input, so that better analysis can be performed in the (compressed) feature space. This algorithm can thus be particularly attractive for analyzing complex inputs such as images and videos. A schematic of

sparse coding is shown in Figure 2.8, where an input (for example, the image of a clock) is formed by a few features selected from a large dictionary [17, 19].

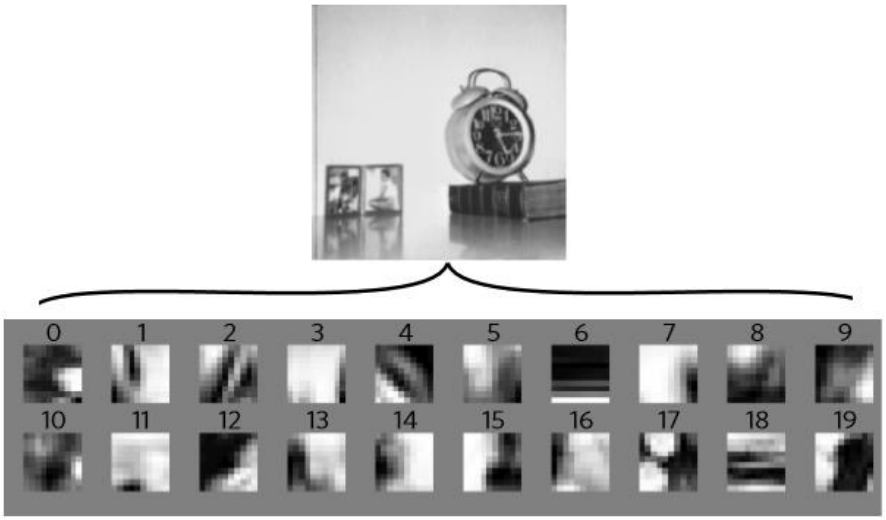


Figure 2.8 Schematic of the sparse coding concept. An input (for example, the image of a clock) can be decomposed into and represented with a minimal number of dictionary elements.

Image analysis task was performed experimentally using the memristor board based on a sparse coding algorithm called the Locally Competitive Algorithm (LCA) [3, 4], whose goal is to minimize an energy function defined in Equation 2.1 using a neural network of leaky-integrate-and-fire neurons and adaptive synaptic weights.

$$\min_{a}(|x - Da^T|_2 + \lambda|a|_0) \quad (2.1)$$

Here x is an m×1 input column vector, representing the pixel intensity of an input image patch. D is an m×n dictionary matrix, where each m×1 column corresponds to one feature (dictionary element). For image analysis, the dictionary elements are analogous to the receptive fields (RFs) found in the visual cortex. a is a 1×n row vector representing output neurons'

activities. The reconstruction of input x can be obtained by the activities of the neurons and their respective features (receptive fields), as $\hat{x} = Da^T$. Note that in LCA the sparsity is enforced through lateral inhibition among neurons so that an optimal, sparse representation of the original input can be obtained. λ balances the requirement of reconstruction error (first term) and sparsity (second term) in the energy function.

The memristor crossbar network is very efficient for computing dot products required by LCA and other algorithms through a single parallel read operation, as shown in Figure 2.9. The vector-matrix multiplication can be computed directly in physics using Ohm's law. The input image was implemented with read pulses with fixed amplitudes but different pulse widths. The dictionary elements were programmed into each column in the array as memristor conductance values, and the charges accumulated at each output neuron is the dot product of the input image vector with the corresponding dictionary element vector.

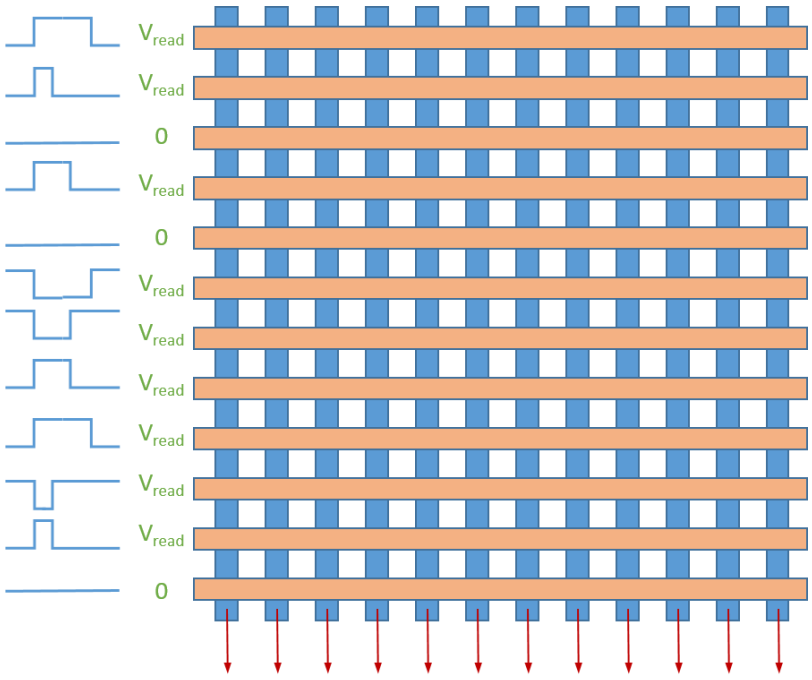


Figure 2.9 Vector-matrix multiplication function obtained through a single parallel read operation in the memristor array.

During our experimental image reconstruction using a memristor crossbar array of 16×32, the 32 dictionary elements were obtained through offline learning using winner-take-all (WTA) and Oja's rule [4]. During WTA learning, the network finds the dictionary element that mostly resembles the input by identifying the element (the winner) that produces the largest dot-product with the input. After identifying the winning neuron, its synaptic weights, forming the receptive field ϕ_w associated with the neuron, is then updated by following Oja's rule (Equation 2.2)

$$y = x^T \phi_w \tag{2.2}$$

$$\Delta \phi_w = \beta (x - y\phi_w) y \tag{2.3}$$

Where *x* is the input and *y* is the output from the winning neuron, representing the maximum dot product of the input vector and the receptive fields in the network (Equation 2.3).

To train the network, 4×4 patches are randomly sampled from a training set of nine natural images (128×128 pixels) as the training input, shown in Figure 2.10.

Figure 2.10 Training sets used to obtain the receptive fields in the dictionary.

In our implementation, the dictionary is obtained through offline training using a realistic memristor model in software, and then programmed into the memristor array using a single-shot programming scheme. However, the actual stored dictionary (Figure 2.11 (b)) in the array differs from the ideal dictionary obtained from simulations (Figure 2.11 (a)) due to the presence of device variations.

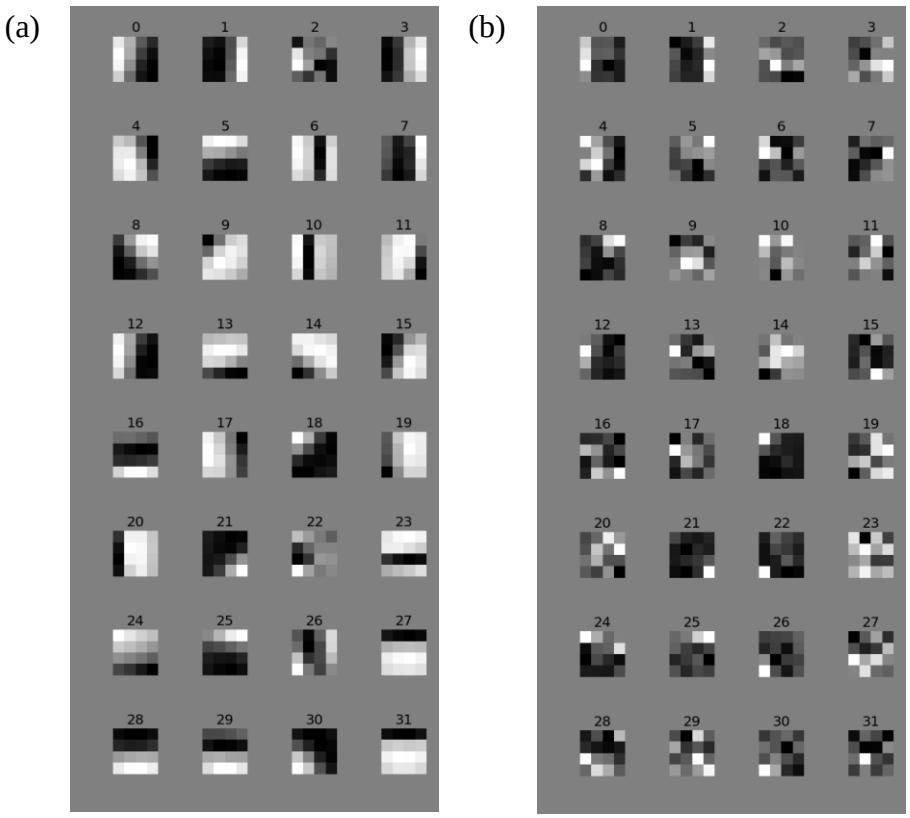


Figure 2.11 32 4×4 dictionary elements. (a) Ideally trained. (b) Experimentally stored.

Experimentally, we mapped the LCA algorithm in the memristor crossbar hardware system and successfully demonstrated natural image reconstruction [21]. The input image (120×120), shown in Figure 2.12 (a), was divided into 4×4 patches and processed using a 16×32 memristor array. The 4×4 input patches are mapped into 1×16 input vectors and fed into the network, which ends up having 32 dictionary elements after offline training (Figure 2.11) that offers a 2× over-completeness (e.g. having 2× the number of dictionary elements compared with the input size). Figure 2.12(b) and Figure 2.12(c) show examples of the original 4×4 patch and the reconstructed 4×4 patch using the LCA system implemented in the 16×32 memristor array. After analyzing each patch, the outputs are stitched together to form the 120×120 image, as shown in Figure 2.12 (d).

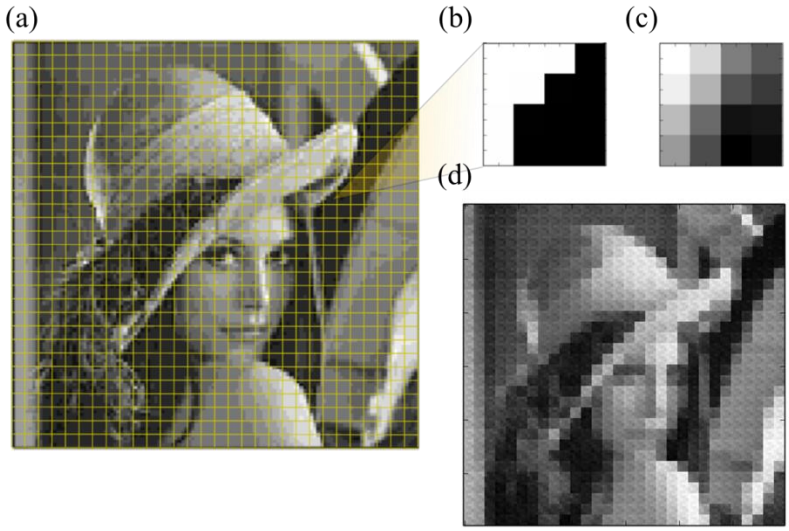


Figure 2.12 Natural image reconstruction using the memristor array. (a) 120×120 original Lena image. (b) A 4×4 patch in (a). (c) Example of an experimentally reconstructed patch. (d) Experimentally reconstructed image after stitching the patches together.

2.5 Device Variation Model

Ideally one would like to perform online learning experimentally using the same memristor crossbar system. However, the slow speed of the board combined with the large training set prevented us from experimentally implementing online learning in the memristor crossbar at this stage. To test the feasibility of online learning we instead performed a detailed, realistic simulation using a device model that incorporates device variations during the incremental weight updates. The largest difference between the online learning case and the offline learning case is that in the offline case the receptive fields (dictionary elements) were obtained using an ideal device model and variations were only considered when the receptive fields were stored in the memristor array; while in the online learning case device variations were carefully considered during the learning stage that required large numbers of incremental weight updates.

To capture the device-to-device variations, we measured the incremental conductance changes from 128 devices in the array using write/erase pulse trains, shown in Figure 2.13 (a). The experimental data were fitted using a compact WO_x memristor model [5] based on Equation 2.4 and 2.5, and the device variations were modeled by adding Gaussian distributions to parameters η_1 and η_2 in the dynamic equation of the internal state variable *w*, and to the initial conductance

$$I = w(\gamma \sinh(\delta V)) + (1 - w)(\alpha(1 - e^{-\beta V})) \tag{2.4}$$

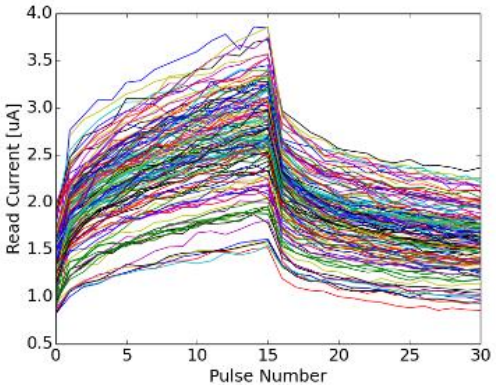
$$\frac{dw}{dt} = \eta_1 \sinh(\eta_2 V) F(w, V) \tag{2.5}$$

Where Equation 2.4 is the current-voltage equation dependent on the internal state variable *w*, and Equation 2.5 describes the update rate of the state variable. $F(w,V)$ is a window function:

$$F(w, V) = \begin{cases} 1 - w, if\ V > 0 \\ w, if\ V < 0 \end{cases} \tag{2.6}$$

Figure 2.13 (b) shows the simulated incremental conductance change curves using this simplified WO_x memristor model, which consistently matches with the experimental ones (Figure 2.13 (a)).

(a)



(b)

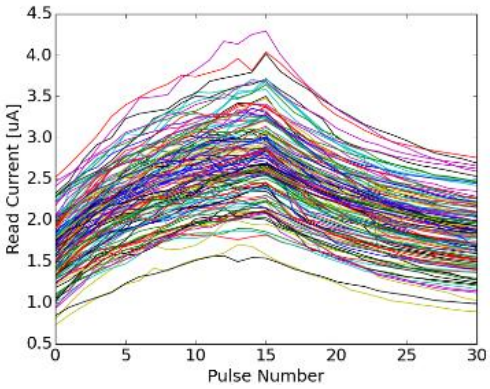


Figure 2.13 Device variabilities observed during write/erase pulse weight updates. (a) Experimental. (b) Simulation.

Figure 2.14 (b) shows the simulated stored dictionary elements after adding device variations to the ideal offline learned dictionary in Figure 2.11 (a). Qualitatively similar behaviors can be observed compared with the experimentally stored dictionary elements shown in Figure 2.14 (a).

(a) (b)

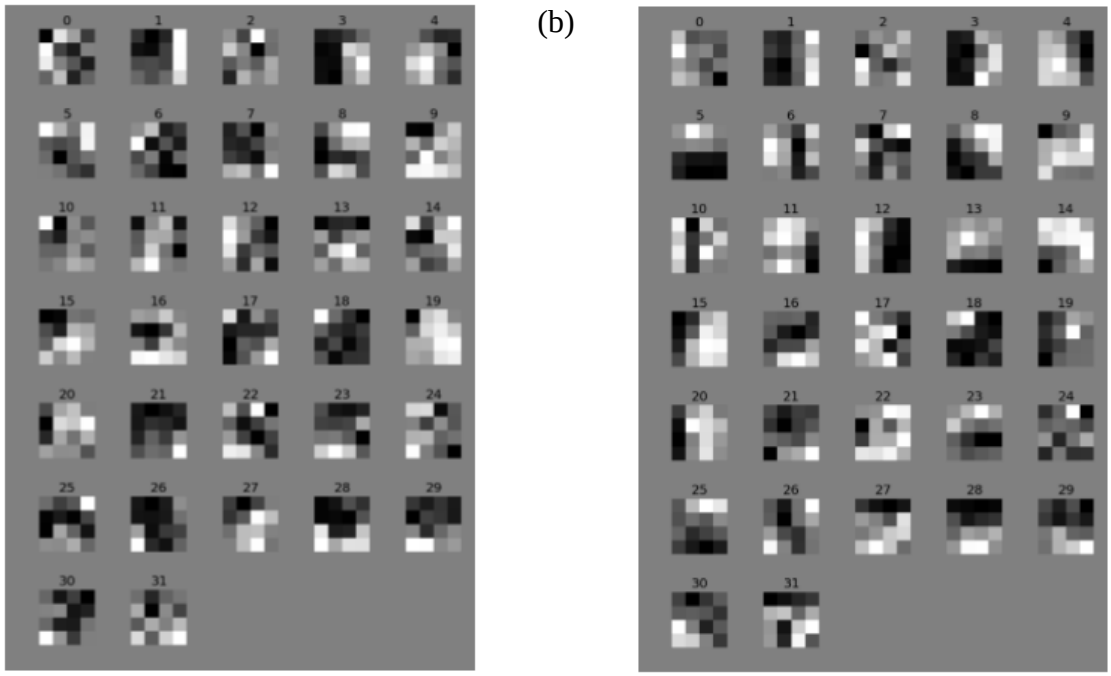
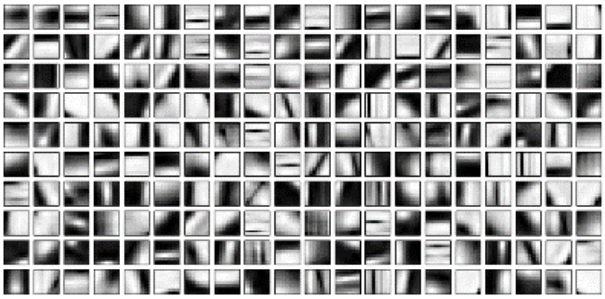


Figure 2.14 Dictionary elements with device variations. (a) Experimentally stored in the memristor crossbar. (b) Simulated stored weights after considering device variations.

2.6 Effects of Device Variations

The device variabilities effects were first carefully considered during offline dictionary storage. At the same time, we expanded the memristor array size to 100×200 in the simulation work to study image patches as large as 10×10. As for offline dictionary storage, the device variabilities only affect the final weight storage step when the desired weight change is programmed into the corresponding device through a single shot. It is surprising to see that with device variations the image reconstruction result is actually better than the case using the ideal dictionary, as shown in Figure 2.15. This effect can be explained by the fact that noise in the stored dictionary provides some fine, high spatial frequency features that are missing in the ideal dictionary, due to relatively small dictionary size.

(a) (b)



(c) (d)

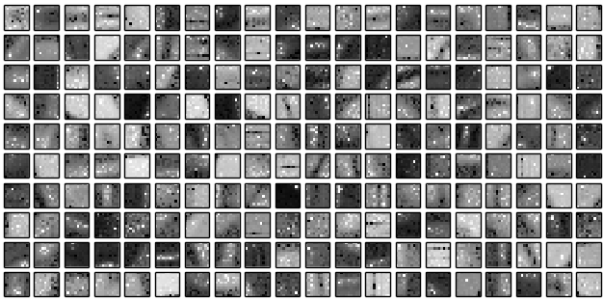
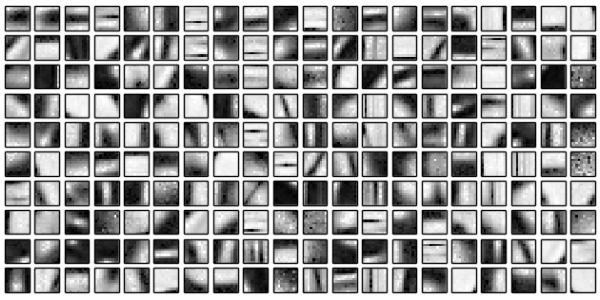


Figure 2.15 Device variabilities effects on image reconstruction using 10×10 patches. (a) Reconstructed Lena image using the ideal dictionary. Reconstructed error, namely MSE = 8.678e-3. (b) Ideal dictionary. (c) Reconstructed Lena image using dictionary with moderate noise. MSE = 5.939e-3. (d) Offline stored dictionary with noise.

To reinforce this statement, we increased the variations in the offline stored dictionary (Figure 2.16), and found that the MSE of the reconstructed Lena image first decreased all the way when the variation in parameter η_2 was increased from 1% to 15%, but then increased to a higher value at 20% variation in η_2.

(a) (b)



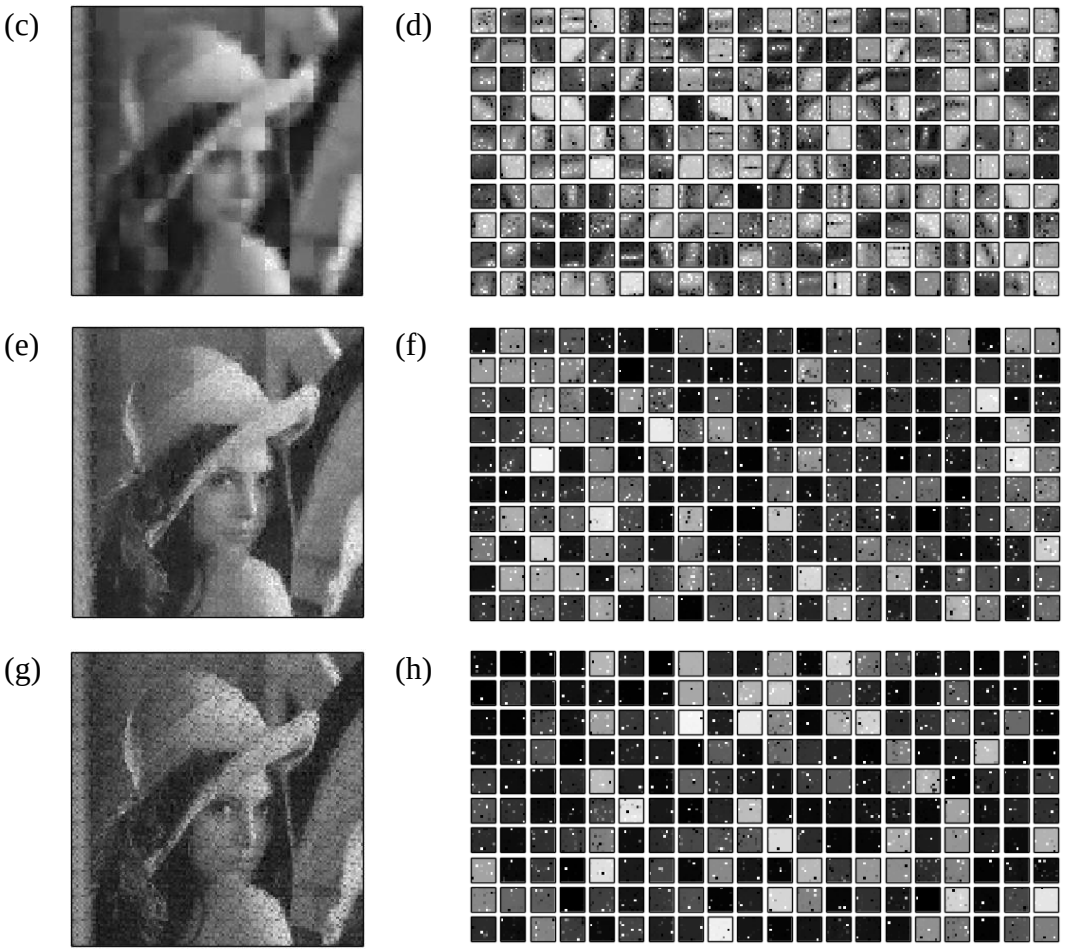


Figure 2.16 Image reconstruction results with different levels of variabilities in the offline stored dictionary. (a) Reconstructed Lena image by adding 1% (a), 5% (c), 15% (e), and 20% (g) variation to parameter η_2. (b) Offline stored dictionary with 1% (b), 5% (d), 15% (f), and 20% (h) variation in η_2.

Figure 2.17 shows quantitatively the effects of the parameter variations during offline weight storage. Expectedly, variations in the prefactor η_1 have a much weaker effect than variations in the exponent η_2. In both cases, the MSE can be made lower than the ideal dictionary case by adding moderate device variations. At very large device variations reflected in η_2, the MSE degrades quickly.

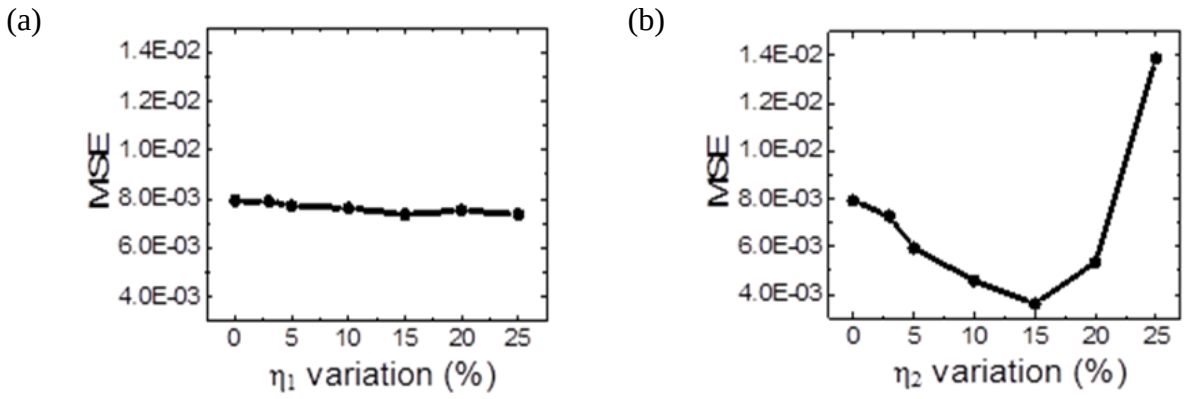
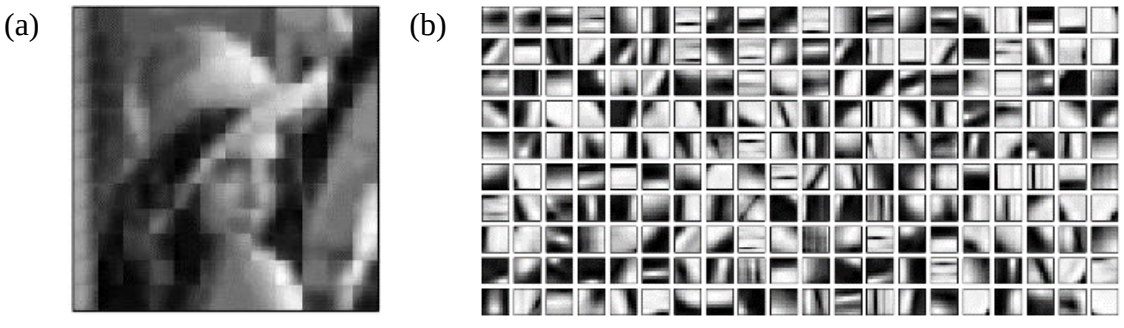


Figure 2.17 MSE of the reconstructed image as a function of variations in (a) η_1 and (b) η_2 using offline stored dictionary.

The device variabilities effects were also carefully considered during online learning using the memristor model and Oja's update rule. Figure 2.18 shows the simulated online learned dictionary with different levels of device variations. It is interesting to see that the image reconstruction result is not sensitive to the noise during online dictionary learning, which is expected as the online learning algorithm is self-adaptive and can adjust to device variabilities during the training iteration loops.



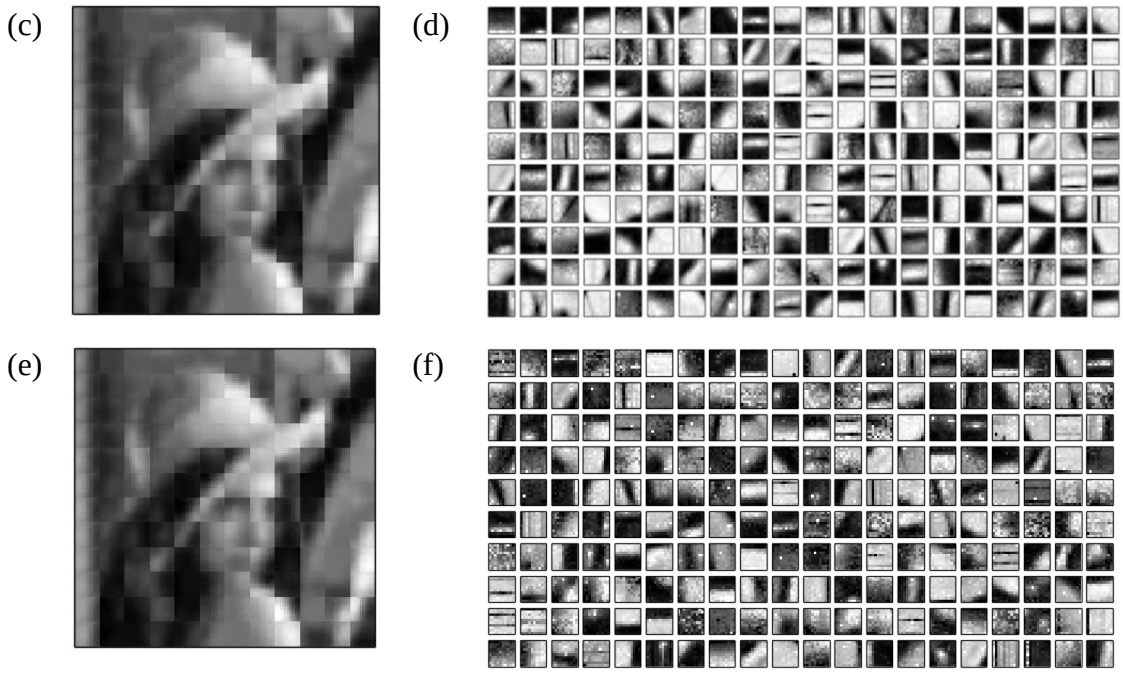


Figure 2.18 Image reconstruction results with different levels of variabilities in the online learned dictionary. (a) Reconstructed Lena image by adding 1% (a), 5% (c), 10% (e) variation to parameter η_2. (b) Online learned dictionary with 1% (b), 5% (d), 10% (f) variation in η_2.

Figure 2.19 shows quantitatively the effects of the parameter variations during online weight update. Neither the variations in the prefactor η_1 nor the variations in the exponent η_2 have a noticeable effect in the reconstruction result. It is demonstrated that online learning can more effectively handle device variations and is suitable for emerging devices such as memristor based systems where large device variations are expected.

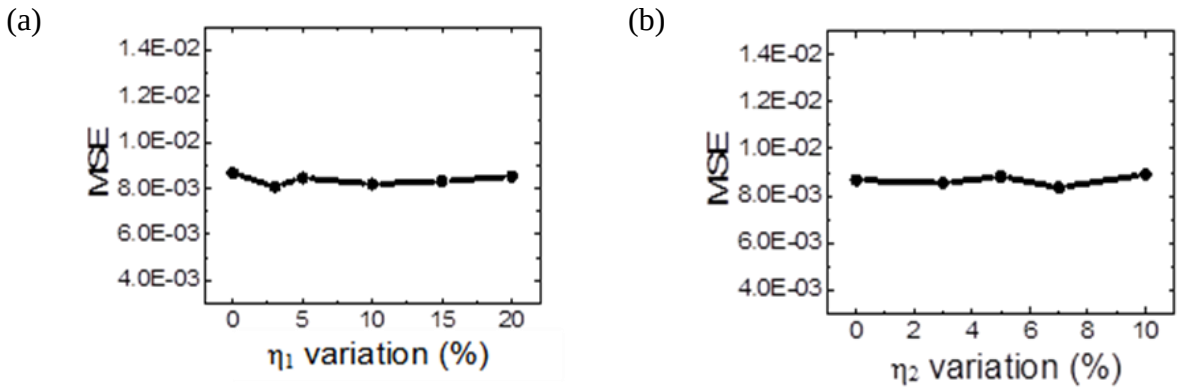


Figure 2.19 MSE of the reconstructed image as a function of variations in (a) η_1 and (b) η_2 using online learned dictionary.

2.7 Effects of Device Failures

Besides device variations, there can also be defects in the memristor array. In particular, device stuck at the low resistance state (SA1) is a common failure mode during memristor programming (programmed too hard that the device got shorted), while devices stuck at the high resistance state (SA0) can occur during device fabrication [6].

Figure 2.20 shows that successful image reconstruction can still be obtained at a high SA0 rate of 10%. In fact, the MSE was found slightly improved when SA0 rate ranges between 10% and 15%, although the overall effect of SA0 failure is small.

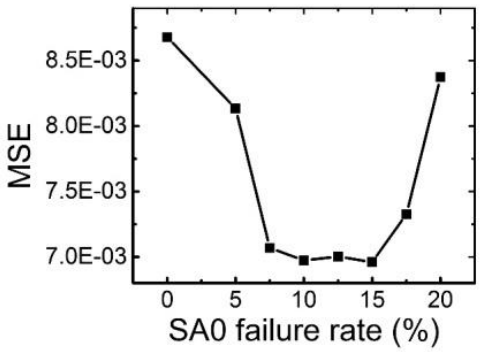
(a)  (b) 

Figure 2.20 Effects of the SA0 device failure during offline weight storage. (a) Typical Lena image reconstruction result with 10% SA0 device failure in stored weights. (b) MSE of the reconstructed image as a function of SA0 device failure rate for offline stored weight cases.

SA1 device failure, however, will significantly deteriorate the reconstruction result and the MSE can increase by an order of magnitude at large SA1 failure rate (Figure 2.21). This effect can be understood by the fact that the SA1 devices will always produce the highest output neuron activity and prevent LCA from operating correctly.

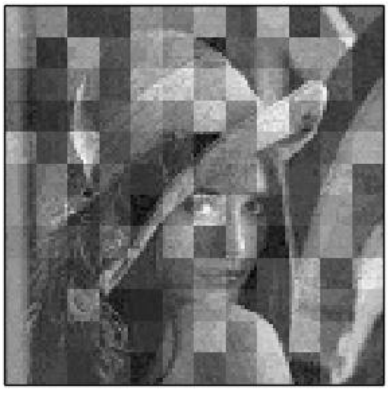
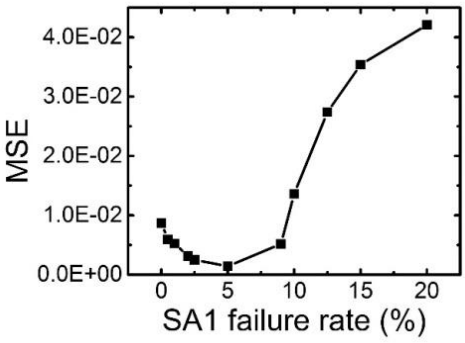
(a)  (b) 

Figure 2.21 Effects of the SA1 device failure during offline weight storage. (a) Typical Lena image reconstruction result with 10% SA1 device failure in stored weights. (b) MSE of the reconstructed image as a function of SA1 device failure rate for offline stored weight cases.

The effects of SA1 will be even worse during online learning where SA1 devices prevent successful feature development. Considering the fact that SA1 failure is a common failure mode during memristor programming, methods to mitigate its effects need to be carefully developed to allow for practical operation of memristor based neuromorphic systems.

To address the devastating SA1 problems, we developed an approach that can detect the SA1 failed columns and remove these columns during online learning. Using a 100×200 dictionary as an example, we take advantage of the high density offered by the memristor crossbar system and add 100 spare columns to the array. All devices are assumed to have a certain SA1 failure rate. Online training is first performed using the first 200 columns and how many times every column gets trained are being monitored. Since the SA1-failed columns will get trained more times due to the high conductance cells, once a column gets trained for more than 10% of the total training times, this column is identified as SA1-failed and thus substituted by a column randomly chosen from the spare columns. The flow chart is the above described scheme is shown in Figure 2.22.

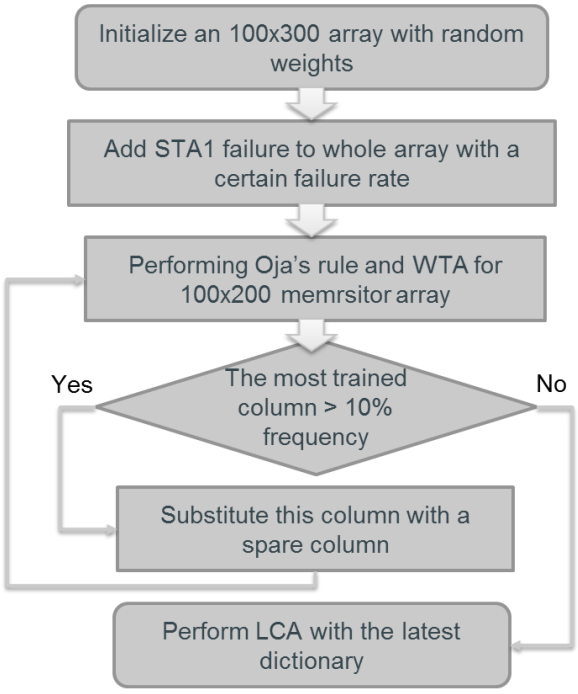


Figure 2.22 Flow chart showing how the failed columns with SA1 devices can be detected and replaced to allow better network performance.

From Figure 2.23, we see that the MSE gradually decreases as the SA1 columns are being replaced with the spare columns using the proposed method.

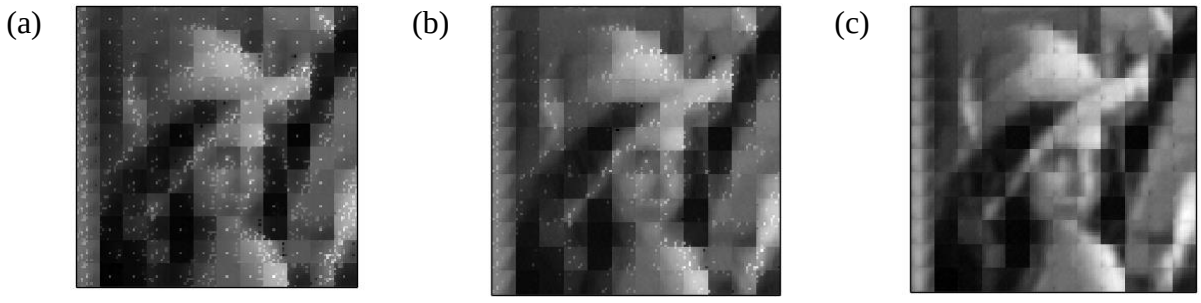


Figure 2.23 Image reconstruction result improves when (a) 0 (b) 20 (c) 45 spare columns are used to replace columns with SA1 devices, resulting in MSE of (a) 2.193e-2 (b) 1.428e-2 (c) 1.058e-2.

Additionally, the receptive fields obtained look closer to those in the ideal case when over-trained SA1 failed receptive fields (ones with lots of dark pixels) are replaced with normal receptive fields, shown in Figure 2.24.

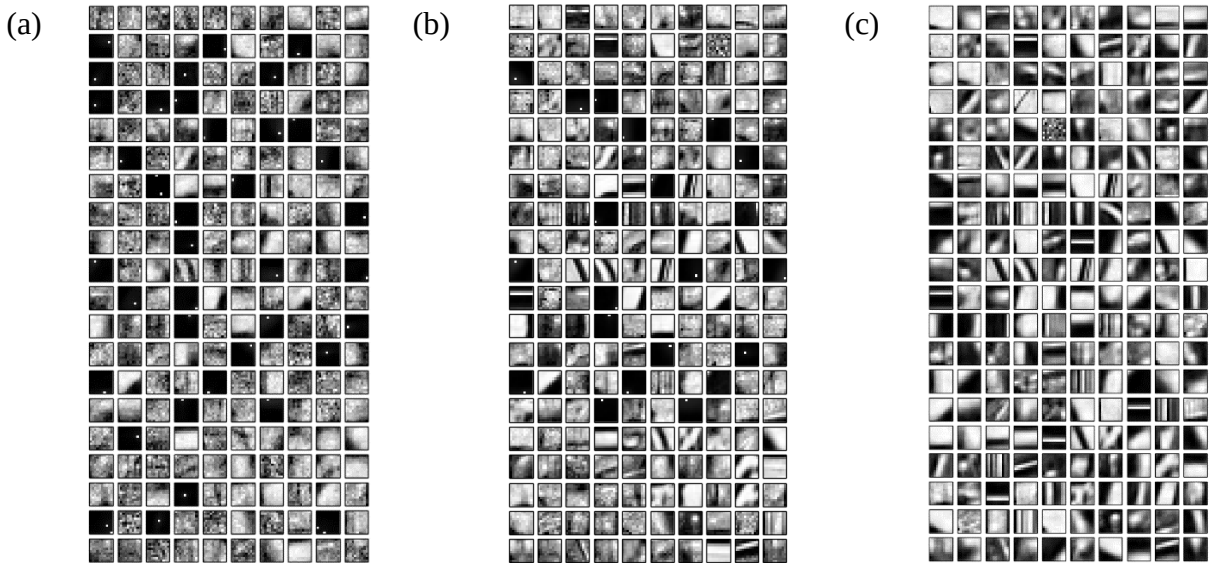


Figure 2.24 Receptive fields obtained when (a) 0 (b) 20 and (c) 45 spare columns were used to replace the faulty columns.

The number of spare columns required to eliminate all SA1-failed RFs are shown in Figure 2.25 as a function of device SA1 rate. Note that the same failure rate is expected as well in the spare columns.

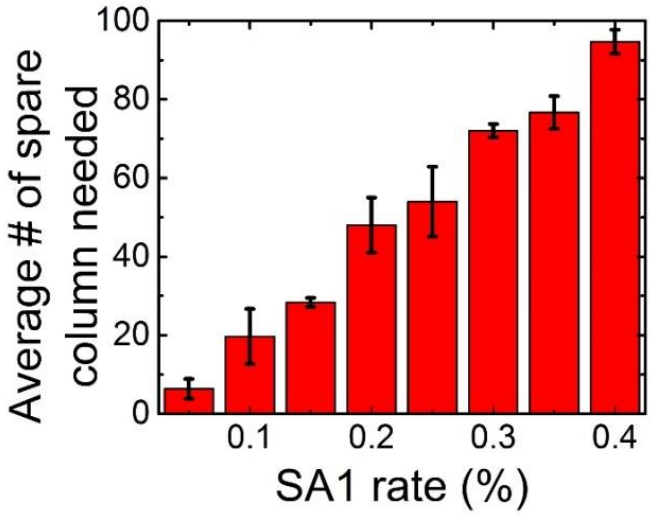


Figure 2.25 Average number of spare columns needed to eliminate all the faulty columns with SA1 devices, as a function of SA1 failure rate.

Overall, the MSE of the reconstruction decreases as more SA1 failed columns are replaced, although oscillations are also found when the new column also contains SA1 devices (Figure 2.26).

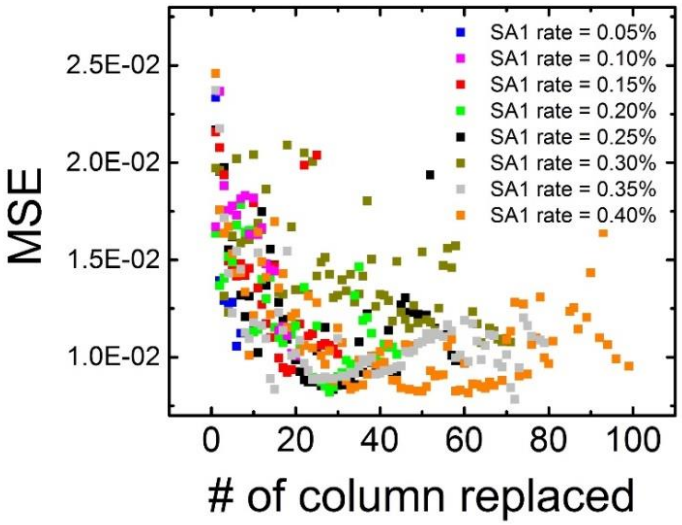


Figure 2.26 MSE as a function of the number of spare columns used to replace the faulty columns, for different SA1 rates.

2.8 Video Processing

Many real-world applications such as video processing are well suited for sparse coding. We carried out analysis on how the memristor-based hardware will perform in video processing tasks. Figure 2.27 shows an example of a 256×192 grayscale image, which was down-sampled from an original 640×480 image. The image was then processed by the 16×32 memristor crossbar using 4×4 patches. During the process, 3072 (64×48) 4×4 patches were processed using the LCA algorithm and each patch took 300 iterations to allow the network to stabilize. Due to the limited data transfer rate between the memristor array and the digital circuitry in the existing test board, the current memristor board will not be able to perform this video analysis in real time. However, in an integrated memristor/CMOS system with a minimum possible read pulse width of 10 ns, our analysis shows that it will take 0.034 second to process such an image, meeting the requirement of real-time streaming video analysis at a rate of 24 frames/s (< 0.042 second process time per frame).

Figure 2.27 256×192 video frame reconstructed using 4×4 patches using the 16×32 memristor crossbar. Left: original. Right: reconstructed.

To process standard 480p (640×480) videos with at least 24 frames/second frame rate without down-sampling, larger patches (e.g. 10×10) will be needed. A memristor array size of 100×200 will be able to process the 480p videos in real time using 10×10 patches. Figure 2.28 shows simulation results of image reconstruction using the larger memristor array.

Figure 2.28 640×480 video frame reconstructed using 10×10 patches with a 100×200 memristor crossbar. Left: original. Right: reconstructed.

2.9 Conclusion

A sparse-coding hardware system based on a memristor crossbar architecture has been successfully demonstrated. This approach, based on pattern matching and neuron lateral inhibition, is an important milestone in the development of large-scale, low-power neuromorphic computing systems. The use of a crossbar architecture allows matrix operations, including

vector-matrix multiplication and matrix transpose operations, to be performed directly and efficiently in the analog domain.

Image reconstruction was experimentally demonstrated using the memristor system. However, memristors suffer from large device variabilities due to fabrication non-idealities as well as stochastic filament growth. Offline stored dictionary with moderate device-to-device variabilities has been shown to be able to improve the reconstructions result, as the noise coming from the device variations can help capture the high spatial frequency information missing in the ideal dictionary. Online dictionary learning was shown to be feasible even in the presence of realistic device variations, and it can efficiently tolerate device variations even for simple algorithms using WTA. Effects of different parameters on image reconstruction for online and offline learned dictionaries have been compared.

In addition, device failures such as stuck-at-0 (SA0) and stuck-at-1 (SA1) can be experienced during programming. While the effect of SA0 is small, the SA1 can be detrimental to the memristor network performance during experimental image reconstruction. A detect-and-substitute solution with certain redundancy to mitigate the SA1 failure has proposed and demonstrated. High speed video processing application was also simulated based on the memristor crossbar array hardware and shown to be feasible with a moderate-sized memristor crossbar.

References

[1] Yang, J. J., Strukov, D. B. & Stewart, D. R. Memristive devices for computing. Nat. Nanotechnol. 8, 13–24 (2012).

[2] Jo, S. H. et al. Nanoscale Memristor Device as Synapse in Neuromorphic Systems. Nano Lett. 10, 1297–1301 (2010).

[3] Rozell, C. J., Johnson, D. H., Baraniuk, R. G. & Olshausen, B. A. Sparse Coding via Thresholding and Local Competition in Neural Circuits. Neural Comput. 20, 2526–2563 (2008).

[4] Sheridan, P. M., Du, C. & Lu, W. D. Feature Extraction Using Memristor Networks. IEEE Trans. Neural Networks Learn. Syst. 27, 2327–2336 (2016).

[5] Chang, T. et al. Synaptic behaviors and modeling of a metal oxide memristive device. Appl. Phys. A 102, 857–863 (2011).

[6] Sheridan, P. & Lu, W. D. Defect considerations for robust sparse coding using memristor arrays. in Proceedings of the 2015 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH´15) 137–138 (IEEE, 2015).

[7] Chang, T., Jo, S.-H. & Lu, W. Short-Term Memory to Long-Term Memory Transition in a Nanoscale Memristor. ACS Nano 5, 7669–7676 (2011).

[8] Hasegawa, T. et al. Learning Abilities Achieved by a Single Solid-State Atomic Switch. Adv. Mater. 22, 1831–1834 (2010).

[9] Seo, K. et al. Analog memory and spike-timing-dependent plasticity characteristics of a nanoscale titanium oxide bilayer resistive switching device. Nanotechnology 22, 254023 (2011).

[10] Ohno, T. et al. Short-term plasticity and long-term potentiation mimicked in single inorganic synapses. Nat. Mater. 10, 591–595 (2011).

[11] Nayak, A. et al. Controlling the Synaptic Plasticity of a Cu_2S Gap-Type Atomic Switch. Adv. Funct. Mater. 22, 3606–3613 (2012).

[12] Kim, S. et al. Experimental Demonstration of a Second-Order Memristor and Its Ability to Biorealistically Implement Synaptic Plasticity. Nano Lett. 15, 2203–2211 (2015).

[13] Du, C., Ma, W., Chang, T., Sheridan, P. & Lu, W. D. Biorealistic Implementation of Synaptic Functions with Oxide Memristors through Internal Ionic Dynamics. Adv. Funct. Mater. 25, 4290–4299 (2015).

[14] Querlioz, D., Bichler, O. & Gamrat, C. Simulation of a memristor-based spiking neural network immune to device variations. in The 2011 International Joint Conference on Neural Networks 1775–1781 (IEEE, 2011).

[15] Sheridan, P., Ma, W. & Lu, W. Pattern recognition with memristor networks. in 2014 IEEE International Symposium on Circuits and Systems (ISCAS) 1078–1081 (IEEE, 2014).

[16] Strukov, D. B. & Likharev, K. K. CMOL FPGA: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices. Nanotechnology 16, 888–900 (2005).

[17] Olshausen, B. A. & Field, D. J. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature 381, 607–609 (1996).

[18] Wright, J. et al. Sparse Representation for Computer Vision and Pattern Recognition. Proc. IEEE 98, 1031–1044 (2010).

[19] Olshausen, B. A. & Field, D. J. Sparse coding with an overcomplete basis set: A strategy employed by V1? Vision Res. 37, 3311–3325 (1997).

[20] Garbin, D. et al. HfO2-Based OxRAM Devices as Synapses for Convolutional Neural Networks. IEEE Trans. Electron Devices 62, 2494–2501 (2015).

[21] Sheridan, P. M. et al. Sparse coding with memristor networks. Nat. Nanotechnol. 12, 784–789 (2017).

Chapter 3 Synaptic Function Learning with Dynamic Memristors

3.1 Introduction

Memristors have attracted broad interest as a promising candidate for future memory and computing applications. Particularly, it is believed that memristors can effectively implement synaptic functions and enable efficient neuromorphic systems [1-4]. Most previous studies, however, focus on implementing different synaptic learning rules, notably spike-timing-dependent plasticity (STDP) [15], by carefully engineering external programming parameters [5-10] instead of focusing on emulating the internal cause that leads to the apparent learning rules. The lack of the necessary internal dynamics implies those implementations are only phenomenological and non-biorealistic, and important dynamic effects at both the cell level and system level may be missed.

WO_x memristive devices show a short-term memory effect where the conductance, after being increased with a programming pulse, will decay over time [28]. The decay can be fitted using a stretched exponential form, which coincidentally resembles the human natural forgetting curve [28]. Through repeated stimulation, short term to long term memory transition has also been demonstrated with WO_x memristors [28]. Below we show that by taking advantage of both the long-term and short-term time scales of internal oxygen vacancy dynamics in WO_x based memristor, diverse synaptic functions at different time scales can be implemented naturally. Specifically, such internal ionic dynamics enables the memristor to exhibit important rate- and

timing-dependent behaviors including pair-pulse facilitation (PPF) [14], sliding threshold effect, and STDP [15] using simple non-overlapping spike signals.

3.2 Internal Ionic Dynamics

The time scales of the internal ionic dynamics can be captured by monitoring the WO_x memristor conductance decay after stimulation. In one such test, 10 positive write pulses (1.2 V 1 ms) at 5 ms intervals were applied to drive the memristor conductance to a higher value. Then small read pulses (0.4 V 1 ms) were applied immediately afterwards to track the memristor conductance change after the stimulation. The decay, possibly due to the diffusion of oxygen vacancies, occurs at two very different time scales, shown in Figure 3.1.

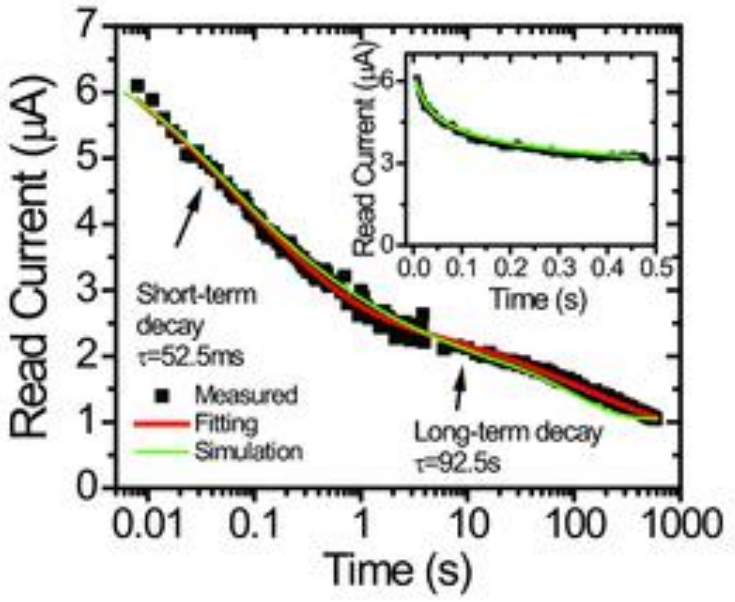


Figure 3.1 WO_x memristor conductance decay. The device was stimulated with 10 write pulses (1.2 V, 1 ms) and the conductance decay was monitored after the stimulation. The experimental data (black squares) can be fitted by the sum of two stretched exponential functions with distinct relaxation time constants (red line). Green line shows simulation results from the memristor model that also captures the memristor behavior at different time scales. Inset: Experimental

(black squares) and simulation results from the memristor model (green line) plotted in linear time scale.

As can be seen in Figure 3.1, there is a very fast short-term decay with time constant ~52.5ms in the beginning and a long-term slow decay with time constant ~92.5s afterwards. To be specific, the decay observed experimentally can be accurately fitted with the following equation:

$$I = A_1 \times I_{short} + A_2 \times I_{long} = A_1 I_{os} \exp\left[-\left(\frac{t}{\tau_s}\right)^{\beta_s}\right] + A_2 I_{ol} \exp\left[-\left(\frac{t}{\tau_l}\right)^{\beta_l}\right] \quad (3.1)$$

Stretched exponential functions, which usually describe relaxation in disordered systems, are used to capture both short-term and long-term decays.

3.3 Modeling the Second Order Memristor

Following the memristor theoretical framework, the device can be mathematically modeled as a second-order memristor with a set of equations including two state variables - one that directly determines the device conductance (weight) and the other that affects the dynamics of the first (conductance) state variable. Below we show that the natural decay of the state variable w_m provides an internal timing mechanism similar to that exhibited by Ca^{2+} concentration in biological synapses and can be used to *natively* implement a broad range of timing- and rate-dependent synaptic plasticity rules [20-22].

The I-V response of the memristor device can be attributed to the sum of a Schottky current term $(1 - w_c) \times \alpha \times [1 - exp(-\beta V)]$, which originates from the W/WO_x interface, and a tunneling current term $w_c \times \gamma \times \sinh(kV)$, which comes from the Pd/WO_x interface. The

ionic dynamics can be captured in the second order memristor model using the two state variables discussed above:

$$I = (1 - w_c) \times \alpha \times [1 - \exp(-\beta V)] + w_c \times \gamma \times \sinh(kV) \tag{3.2}$$

$$\frac{dw_m}{dt} = \lambda_m \mathrm{W}(w_m, \mathrm{V}) \sinh(\rho_m |V|) - \frac{w_m - w_{m0}}{\tau_m^*(w_m)} \tag{3.3}$$

$$\frac{dw_c}{dt} = \lambda_c \mathrm{W}(w_c, \mathrm{V}) \exp(\varepsilon w_m) \sinh(\rho_c \mathrm{V}) - \frac{w_c - w_{c0}}{\tau_c^*(w_m)} \tag{3.4}$$

$$\frac{1}{\tau_m^*(w_m)} = \frac{w_m}{\tau_s} \tag{3.5}$$

$$\frac{1}{\tau_c^*(w_m)} = \frac{1}{\tau_l} + \frac{\sigma \cdot w_m}{\tau_s} \tag{3.6}$$

$$W(w, V) = \begin{cases} 1 - \exp\left(-\dfrac{w_{max} - w}{0.0001}\right), & if\ V \geq 0 \\ 1 - \exp(-\dfrac{w - w_{min}}{0.0001}), & if\ V < 0 \end{cases} \tag{3.7}$$

w_c is the state variable governing memristor conductance, representing the effective area of the conducting region [19, 23]. w_m is the temporary high mobility of the oxygen vacancies when they are driven out of equilibrium after stimulation, likely due to local lattice distortion and strain [25]. w_m will affect how w_c responds to stimulation through the $\exp(\varepsilon w_m)$ term in Equation 3.4. w_m and w_c both have decays. The effective decay time constant τ_m^* was chosen in the form of Equation 3.5 to better capture the stretched exponential decay instead of the simple exponential decay. The decay of w_c can also be affected by the oxygen vacancy mobility so τ_c^* is a function of w_m too. When w_m is large, $\tau_c \sim \tau_s / \sigma w_m$ so the decay follows a fast short term

fashion, while when w_m is small, $\tau_c \sim \tau_l$ and the conductance decay follows a very long decay constant. The choice of the "window function" $W(w,V)$ can capture the non-linear state dependent programming response.

The memristor model was implemented in LTspice and capture and quantitatively explain all experimental behaviors, including the two-stage decay (Figure 3.1) and the synaptic behaviors which will be discussed in the next section, using a single set of material dependent parameters.

Parameter	Value	Parameter	Value
α	1.5e-6	β	4
γ	3.2e-6	κ	5
λ_m	1e-6	λ_c	1e-6
ρ_m	15.5±2.5	ρ_c	14±2
w_{c0}	0-0.3	W_{m0}	0.001
τ_s	0.0025	τ_l	298
σ	0.25	ϵ	15

Table 3.1 Memristor parameters used in simulation

In the simulation, all parameters were fixed except for ρ_m and ρ_c which need to be adjusted slightly to account for small device variations in the length of oxygen vacancy poor region. w_{c0} is the initial value before the simulation whose value varies and listed in Table 3.1, while w_{m0} is always fixed to be 0.001. Exact values of ρ_m, ρ_c and w_{c0} used in the following

simulations are listed in Table 3.1. As there are two internal variables w_c and w_m, representing device conductance and oxygen vacancy mobility respectively, the memristor can be considered as a "second-order" memristor [13, 26] where w_c is modulated by w_m. The effect of w_m is mostly short term with a decay constant of tens of milliseconds, but it can affect w_c which has long term effects on the memristor conductance.

The internal ionic dynamics of the second order memristors can be used to emulate numerous synaptic behaviors, both at short term such as paired pulse facilitation (PPF) [14] and at long term such as spike timing dependent plasticity (STDP) [15]. Different synaptic learning functions can be captured experimentally as well as using the second order memristor model. All simulations were performed in LTspice. Here by using simple spike signals, instead of engineered arbitrary overlapping pulses, both state variables w_m and wc can be excited and lead to different electric responses that replicate synaptic functions which can be used for neuromorphic computing applications.

3.4 Synaptic Functions Emulation

3.4.1 Paired Pulse Facilitation

Paired pulse facilitation is an important short-term phenomenon in neuroscience [14] when two excitatory presynaptic pulses are applied successively, the second spike will induce a larger excitatory postsynaptic current (EPSC) than the first one. In addition, the amplitude of the EPSC caused by the second pulse decreases when the interval between two pulses increases, as shown in Figure 3.2. The mechanism is believed to be the exponential decay of the residual Ca^{2+} caused the by the first spike enhances the overall Ca^{2+} level, which results in higher EPSC by the second spike.

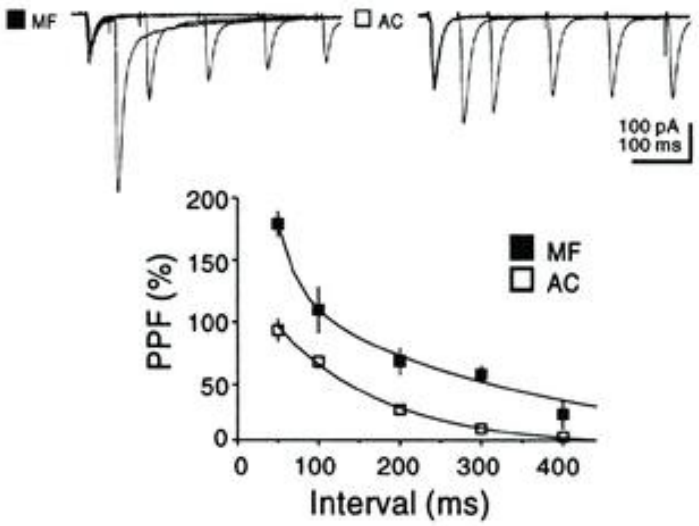


Figure 3.2 Results of PPF from mossy fiber (MF) and assoc/com (AC) synapses. From [18]. Top: EPSP obtained from paired pulses with different intervals. The EPSP from the second pulse is enhanced, and the enhancement is weaker with longer intervals between the pulses. Bottom: PPF ratio as a function of pulse interval. The ratio was defined as $(p_2 - p_1)/p_1$, where p_1 and p_2 are the amplitude of the EPSCs evoked by the first and second pulse, respectively.

Similar effects have been obtained in the WO_x memristor device (Figure 3.3). Two identical, non-overlapping pulses were applied (1.4 V, 1 ms). As can be seen in Figure 3.3, the current induced by the second voltage pulse is indeed larger than that by the first one. Similar to the residual Ca^{2+} effect in biological synapses, the ion mobility w_m is enhanced by the first pulse increases and the effect then decays. If the second pulse is applied before w_m decays to its resting value, an enhanced current will be observed, leading to the PPF effect. The shorter the interval, the less w_m will decay, which leads to a relatively larger current.

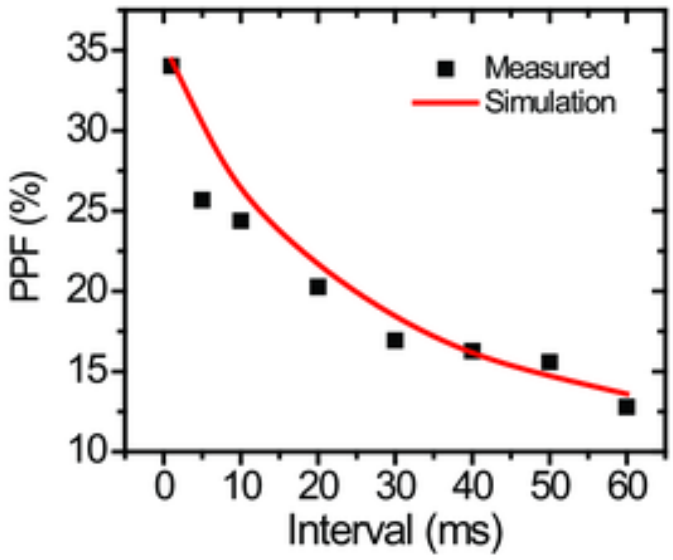


Figure 3.3 Paired-pulse facilitation effect in WO_x memristors. PPF ratio as a function of pulse interval. Squares: Experimental data. Line: Simulation results from the memristor model using experimental parameters.

3.4.2 Frequency Dependent Plasticity

A well-known phenomenon in neuroscience studies is the frequency dependent plasticity, a direct extension of PPF, where more than two excitatory presynaptic spikes are applied to the synapse. By changing the frequency of the stimuli, which is inversely related to the pulse interval, different amplitudes of synaptic weight change can be obtained. Experimentally, we applied ten continuous write pulses (1.25 V, 1 ms) with different intervals, i.e. frequencies, and monitored the current during the pulse. A clear trend of potentiation in the current from the last pulse with respect to the stimulation frequency has been observed, shown in Figure 3.4.

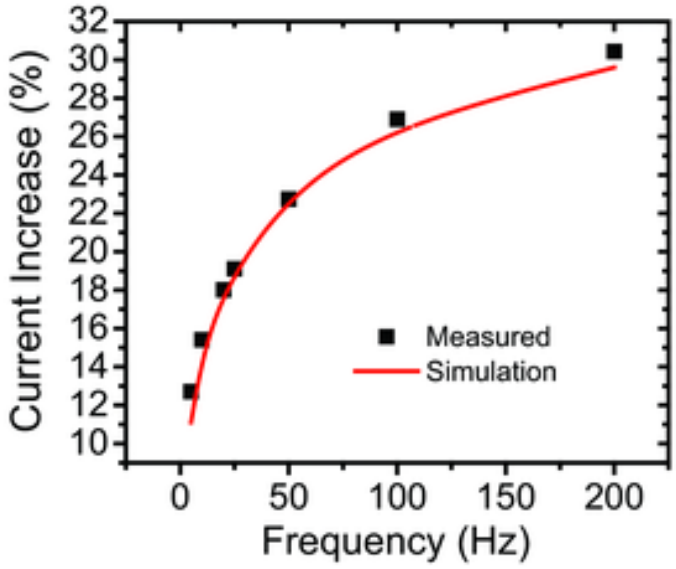


Figure 3.4 Change in memristor current after the application of pulse trains consisting of ten write pulses (1.25 V, 1 ms) with different frequencies. Higher stimulation frequency leads to larger conductance enhancement. Squares: Experimental data. Line: Simulation results from the memristor model using experimental parameters.

3.4.3 Experience Dependent Plasticity

Another important synaptic behavior is called the experience dependent plasticity [16] where the synapse can show either potentiation or depression depending on what stimulation they have been subjected to in the past (Figure 3.5). It has been found generally that high frequency stimulation leads to synaptic potentiation while low frequency stimulation leads to depression, and there is a threshold frequency at which the synaptic weight does not change over stimulation [17]. The threshold frequency is dependent on the experience of the synapse (and hence is not a constant). This effect has thus been termed the sliding threshold effect [17]. A good illustration is that after some high synaptic activity, the threshold will increase (sliding to the right) so that synaptic depression is favored. While after some low activity, the threshold will decrease (sliding to the left) and synaptic potentiation becomes easier. The sliding threshold

effect likely plays a significant role in learning as it prevents the connections to grow too fast at strong stimulation and enables learning even at weak stimulation conditions.

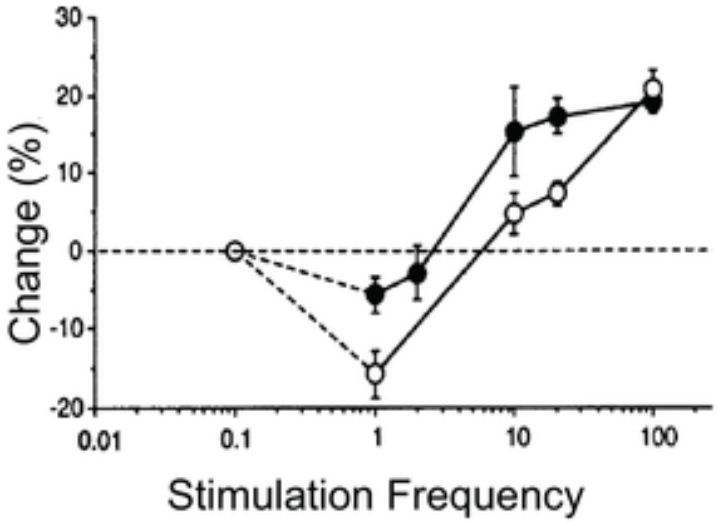


Figure 3.5 Relative change in synaptic weight as a function of stimulation frequency for two different cases. Low stimulation frequency results in depression and high stimulation frequency results in facilitation, and the threshold moves to lower frequency under the light-deprived condition (filled symbols) compared to the normal condition (open symbols). Data were obtained in rat visual cortex and reproduced with permission from [17].

Experimentally, we tested this effect by applying a series of pulse trains to the WO_x memristor. Each pulse train consists of five identical programming pulses (1 V, 1 ms) with different frequencies. As can be seen in Figure 3.6, firstly, pulse train of 200 Hz (high activity) was applied and an increase in current was observed. Subsequently, a 10 Hz pulse train caused the memristor current to drop. Then following a 1 Hz pulse train (low activity), the same 10 Hz pulse train created an increase in the memristor current instead. These results clearly show that the effect of the stimulation on memristor depends on previous activity, and the threshold for potentiation/depression is a sliding one.

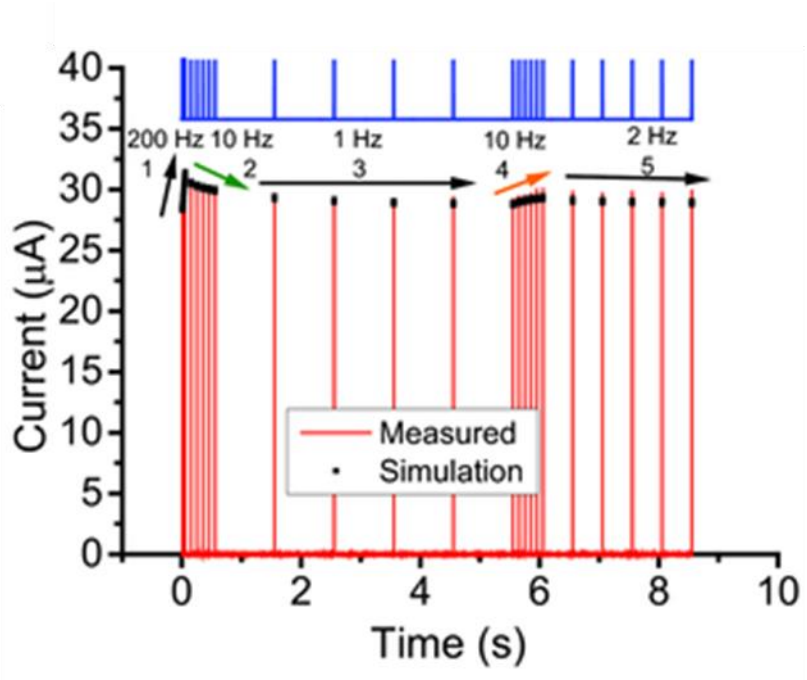


Figure 3.6 Memristor response to consecutive programming pulse trains (1 V, 1 ms, blue lines) at different frequencies. The 10 Hz pulse train caused current decrease in step 2 following strong stimulation in step 1, but current increase in step 4 following weak stimulation in step 3. Black squares: Simulation results from the memristor model using experimental parameters.

The experimentally observed experience-dependent behaviors can be accurately reproduced in simulation based on the second order memristor model. Specifically, the 200 Hz pulse train drove w_m to a high value which shortens the effective decay time constant $\tau_m^*(w_m)$ and $\tau_c^*(w_m)$, and thus results in faster decay in w_c and w_m. Consequently, a 10 Hz pulse train was not sufficient to overcome the fast w_c decay and conductance drop was observed. On the other hand, after the 1 Hz pulse train, w_m relaxed to a low value so the decay of w_c and w_m is slow. As a result, the same following 10 Hz pulse train was enough to bring the conductance up. Therefore, the same device can either exhibit conductance increase or decrease under the same stimulation depending on previous activity.

Based on this understanding, we carried out another experiment to unambiguously verify the sliding threshold effect. Three levels of activities were applied to the device first. They are ten pulses at either 20, 50 or 100 Hz. Then, five write pulses (1.2 V 1 ms) with different stimulation frequencies were applied to record the net current change. As shown in Figure 3.7, a low stimulation frequency usually leads to conductance decrease while a high stimulation frequency leads to an increase. More importantly, the threshold frequency at which the net current change is zero shifts to higher values when the previous stimulation gets stronger.

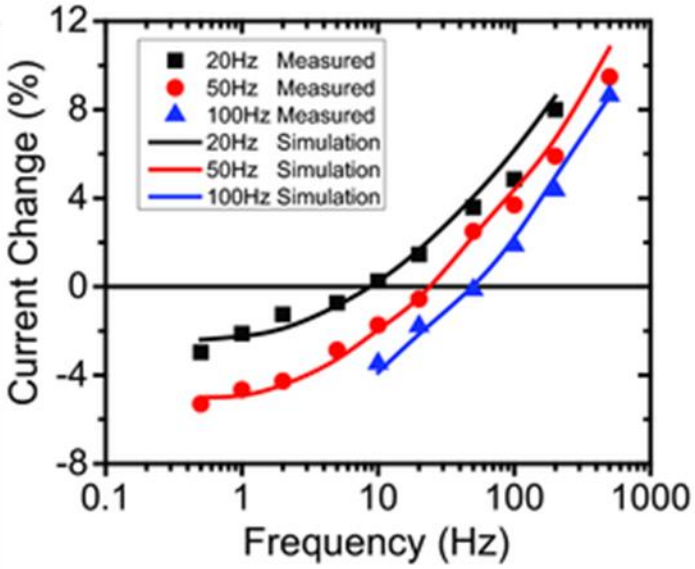


Figure 3.7 Memristor current change as a function of the stimulation frequency after the memristor has been experienced to different levels of activities. Pulse trains consisting of five pulses (1.2 V, 1 ms) with different repetition frequencies were used to program the memristor. Black squares, red circles, and blue triangles represent experimental data. The solid lines are simulation results from the memristor model using experimental parameters.

3.4.4 Short Term Dynamics Leading to Long Term Change

As suggested in Equation 3.4, w_m is a short-term state variable but it can affect the long-term state variable w_c and lead to long term plasticity change. In the PPF effect, only short-term

change was monitored. However, long-term change can also be achieved by varying the pulse intervals during PPF measurements. To verify the long-term effect of the memristor, we read the device state using a read pulse 5 s after applying a pair of stimulation pulses with different intervals. The conductance changes monitored at 5 s is considered a long-term effect since all short-term effects should have completely vanished after 5 s.

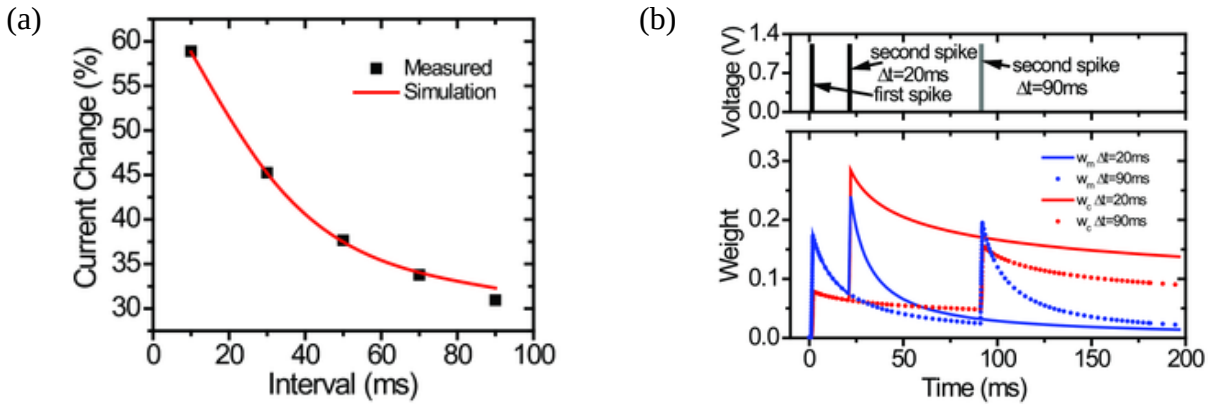


Figure 3.8 Effect of short-term behavior on long-term weight change. a) Changes in the memristor conductance measured 5 s after the application of a pair of programming pulses (1.4 V, 1 ms) versus the interval between the pulses. The differences in activity at short-term lead to measurable differences in long-term memristor weight. Black squares: Experimental data. Solid line: Simulation results from the memristor model using experimental parameters. b) Simulation results illustrating how the short-term behavior affected long-term weight change. The difference in long-term weight is caused by the different values of residue w_m at the moment when the second pulse is applied. State variable w_c and state variable w_m under two conditions (interval between two pulses Δt = 20, 90 ms) are shown.

Figure 3.8 (a) plots the percentage change in read current (read current measured after 5 s minus the initial read current divided by initial read current) as a function of the stimulation pulse pair interval. It can be seen that a larger conductance increase can be observed if the programming pulses were applied with shorter intervals far back in the history, verifying that the short-term dynamics, represented by state variable w_c and dependent on the pulse intervals, can

indeed affect the long-term device state. This result can again be explained with the aid of the second order memristor model, as shown in Figure 3.8 (b). The exponential decay of w_m can accumulate over time if the two stimulation pulses are close enough, say within 20 ms. Then the elevated w_m will cause a large increase in the long-term state variable w_c. On the other hand, if the interval between two stimulation pulses is long enough that w_m already decays to a low value then there is almost no accumulation in w_m. As a result, the second pulse will only cause a small increase in w_c. In this sense, w_m can be thought of as playing the role of the postsynaptic calcium concentration, whose short-term dynamics provide an intrinsic timing mechanism [12] for synaptic weight change, while w_c plays the role of the state variable that determines the weight.

3.4.5 Spike Timing Dependent Plasticity

Following the unified model [11] based on calcium dependent synaptic weight changes, different synaptic learning rules can be implemented natively. We have demonstrated PPF and rate-dependent plasticity based on the native V_O dynamics similarly in WO_x memristors. Below we show that similar to the neuroscience model predictions based on calcium dynamics, oxygen vacancy dynamics in WO_x based memristors can also lead to timing based synaptic learning rules such as STDP, which is widely considered a key synaptic plasticity rule in biological and artificial neural networks.

In STDP, the relative timing of the pre-synaptic spike and post-synaptic spike determines the potentiation or depression in the synaptic weight and also how much [15]. Specifically, if the presynaptic spike arrives before the post one (pre-post pair), potentiation will be induced, while a reversed sequence (post-pre pair) will cause depression. Moreover, a larger interval between the pre-synaptic spike and post-synaptic spike will lead to smaller weight modification. In

neurobiology, the relative timing information between the pulses is natively embedded in the natural decay of calcium levels [11, 12]. Similarly, in our devices, STDP is achieved naturally through the oxygen vacancy dynamics reflected in the state variable w_m.

As illustrated in Figure 3.9, the pre-synaptic spike is represented by a negative erase pulse (-1.1 V, 1 ms) and the post-synaptic spike is represented by a positive write pulse (1.1 V, 1 ms), both applied at the top electrode (post-synaptic side). Before each test, the device was stimulated with a pulse train consisting of 10 positive pulses (1.2 V, 1 ms, 200 Hz) to bring the conductance up to a middle level. Within each test, 30 pairs of either the positive-negative pulse pair, representing post-pre spike sequence, or the negative-positive pulse pair, representing pre-post spike sequence, was applied at a 5 Hz repetition frequency. Device conductance was always measured 0.2 s after the last pulse pair and compared to an initial value before applying any pre-post or post-pre pairs.

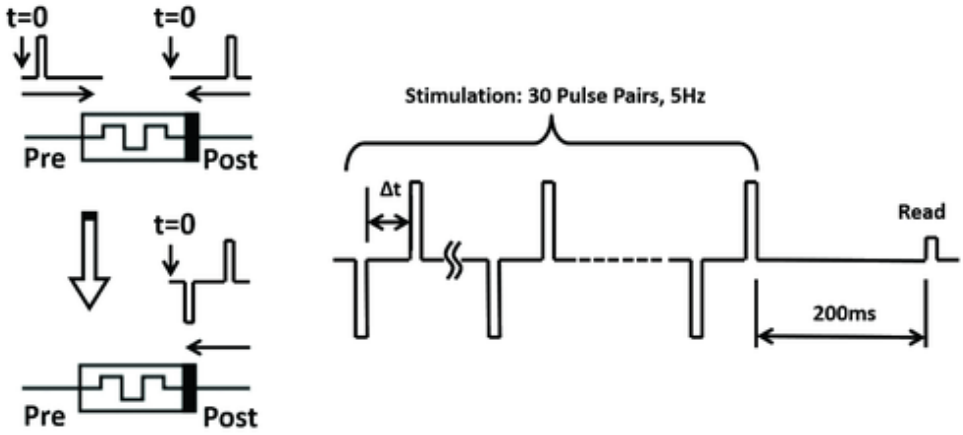


Figure 3.9 STDP experiment setup. Left: Experimental setup, a pre-post pair consisting of identical spikes is equivalent to a negative/positive pulse pair applied on the postsynaptic side; right: the pre-post programming protocol including 30 pulse pairs (-1.1 V, 1 ms/1.1 V, 1 ms) applied at 5 Hz for stimulation, followed by read 200 ms after stimulation. Post-pre pairs are applied similarly.

As shown in Figure 3.10 (a), for pre-post conditions ($\Delta t > 0$), the memristor conductance increases while for post-pre conditions ($\Delta t < 0$), conductance decreases. We can see that even though the net flux applied to the device is zero, i.e. we have symmetric positive and negative pulses, the net effect is found to be non-zero and more strongly dependent on the second pulse. This effect can be explained with the aid of the second order memristor model, as illustrated in Figure 3.10 (b). The long-term state variable w_c shows a net decrease after a positive-negative pulse pair because the erase effect from the second pulse is stronger than the first one due to the accumulation of w_m residue. Thus, a larger Δt will lead to a longer decay of w_m before the second pulse comes, and a smaller residue w_m. So the net weight change reflected in w_c is smaller.

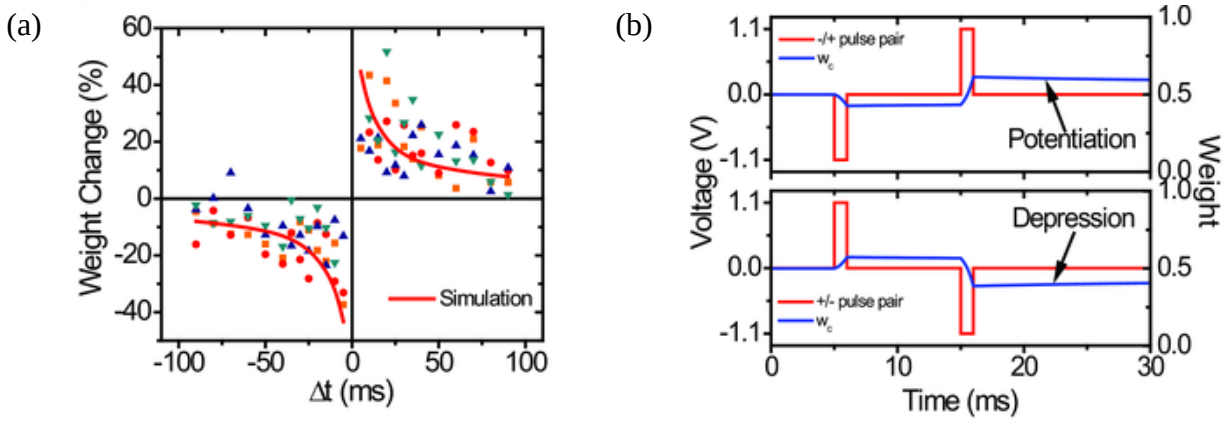


Figure 3.10 STDP in WO_x memristor. a) Memristor weight change as a function of the relative timing between the pre- and postsynaptic pulses, $\Delta t = t_{post} - t_{pre}$. Symbols: Experimental results obtained from four different tests. Solid lines: Simulation results from the memristor model using experimental parameters. b) Simulation results illustrating how relative timing of the pulses affects memristor weight. Only w_c is shown here for clarity. The second pulse in the pair has a larger effect on w_c due to residue enhanced w_m from the first pulse, and can cause either potentiation or depression depending on the relative timing between the pre- and postsynaptic pulses.

There are two key factors accounting for STDP: 1) the second pulse in a pulse pair determines the sign of the long-term weight change and 2) the effectiveness of the pulse pair is larger with shorter interval inside the pulse pair. These effects be directly implemented by the residue w_m effect that led to the observed experimental behavior. Furthermore, the effects of the role played by w_m were clearly revealed in the two-state-variable memristor model, as shown in Figure 3.11.

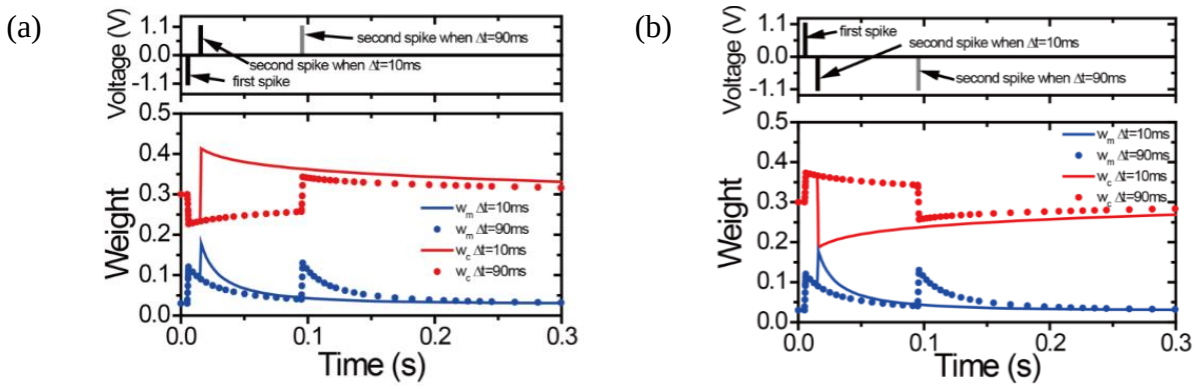


Figure 3.11 Simulation results illustrating how STDP is obtained with simple, non-overlapping pulses. Results of different intervals (Δt = 10 ms and 90 ms) highlighting how w_m and w_c evolves are shown for both a) pre-post (potentiation) condition and b) post-pre (depression) condition. The effect of the 2nd pulse is enhanced due to the residue enhanced w_m (enhanced from the first pulse). As a result, a shorter interval leads to a larger w_m enhancement at the moment of the 2nd pulse and a larger potentiation or depression effect, depending on whether Δt is positive or negative.

3.5 Conclusion

We showed that several important, rate- and timing-based synaptic behaviors at different time scales can be implemented using simple, biorealistic pulses in memristors naturally, by employing the internal ionic dynamics of the device. Following the theoretical framework based

on Ca^{2+} driven synaptic plasticity, different synaptic behaviors including paired-pulse facilitation, frequency-dependent facilitation, sliding threshold effect, and timing-based plasticity (STDP) can be implemented and quantitatively explained using a second-order memristor model. Specifically, we note that to emulate the synaptic behaviors in a biorealistic manner, two critical features need to be present: first, the device should exhibit analog resistance switching behavior, i.e., incremental conductance change during SET and RESET operations; second, there should exist an internal physical process that offers some short-term decay dynamics to enable activity-dependent conductance change. Looking toward the future, along with other recent studies [13], these dynamic memristors will enable the implementation of adaptive neuromorphic systems in a biorealistic fashion, and higher order memristive systems for efficient memory, computing, and other applications.

Appendix

LTspice code

```
********** LTspice code for metal oxide memristors**********
*Parameters:
*alpha is prefactor for Schottky barrier
*beta is exponent for Schottky barrier
*gamma is prefactor for tunneling
*delta is exponent for tunneling
  ************************************************************************
.SUBCKT memristor 1 2 params:
+ alpha=1.5e-6 beta=4 gamma=3.2e-6 delta=5 wmax=1 wmin=0
*State variable:
```

```
.param lambda=1e-6 rhoc=14.5 rhom=17 taul=298 taus=0.0025 epsilon=15 sigma=0.25
.param cc={1}
.param cm={1}
Cpvar1 c 0 {cc}
Cpvar2 m 0 {cm}
*rate equation considering the diffusion effect
Gc 0 c value={trunc1(V(1,2),cc*V(c))*(lambda*exp(epsilon*cc*V(c))*sinh(rhoc*V(1,2)))
(cc*V(c)-0.001)*(1/taul+sigma*cm*V(m)/taus)}
Gm 0 m value={trunc2(V(1,2),cm*V(m))*(lambda*sinh(rhom*abs(V(1,2))))-(cm*V(m)
0.001)*(cm*V(m)/taus)}
.ic V(c) = 0.001
.ic V(m) = 0.001
**********************************************************
*auxiliary functions to limit the range of w
.func sign2(var) {(sgn(var)+1)/2}
.func trunc1(var1,var2) {sign2(var1)*sign2(wmax-var2)*(1-exp(-(wmax-var2)/0.0001))+sign2(
var1)*sign2(var2-wmin)*(1-exp(-(var2-wmin)/0.0001))}
.func trunc2(var1,var2) {sign2(var1)*sign2(wmax-var2)*(1-exp(-(wmax-var2)/0.0001))+sign2(
var1)*sign2(var2-wmin)*(1-exp(-(var2-wmin)/0.0001))}
**************************************************************************
*Output:
Gw 1 2 value={(1-cc*V(c))*alpha*(1-exp(-
beta*V(1,2)))+(cc*V(c))*gamma*sinh(delta*V(1,2))}
.ENDS memristor
```

References

[1] Yang, J. J., Strukov, D. B. & Stewart, D. R. Memristive devices for computing. Nat. Nanotechnol. 8, 13–24 (2012).

[2] Chang, T., Yang, Y. & Lu, W. Building Neuromorphic Circuits with Memristive Devices. IEEE Circuits Syst. Mag. 13, 56–73 (2013).

[3] Kuzum, D., Yu, S. & Philip Wong, H.-S. Synaptic electronics: materials, devices and applications. Nanotechnology 24, 382001 (2013).

[4] Borghetti, J. et al. 'Memristive' switches enable 'stateful' logic operations via material implication. Nature 464, 873–876 (2010).

[5] Jo, S. H. et al. Nanoscale Memristor Device as Synapse in Neuromorphic Systems. Nano Lett. 10, 1297–1301 (2010).

[6] Kuzum, D., Jeyasingh, R. G. D., Lee, B. & Wong, H.-S. P. Nanoelectronic Programmable Synapses Based on Phase Change Materials for Brain-Inspired Computing. Nano Lett. 12, 2179–2186 (2012).

[7] Alibart, F. et al. A memristive nanoparticle/organic hybrid synapstor for neuroinspired computing. Adv. Funct. Mater. 22, 609–616 (2012).

[8] Krzysteczko, P., Münchenberger, J., Schäfers, M., Reiss, G. & Thomas, A. The Memristive Magnetic Tunnel Junction as a Nanoscopic Synapse-Neuron System. Adv. Mater. 24, 762–766 (2012).

[9] Wang, Z. Q. et al. Synaptic Learning and Memory Functions Achieved Using Oxygen Ion Migration/Diffusion in an Amorphous InGaZnO Memristor. Adv. Funct. Mater. 22, 2759–2765 (2012).

[10] Zamarreño-Ramos, C. et al. On Spike-Timing-Dependent-Plasticity, Memristive Devices, and Building a Self-Learning Visual Cortex. Front. Neurosci. 5, 26 (2011).

[11] Shouval, H. Z., Bear, M. F. & Cooper, L. N. A unified model of NMDA receptor-dependent bidirectional synaptic plasticity. Proc. Natl. Acad. Sci. U. S. A. 99, 10831–6 (2002).

[12] Graupner, M. & Brunel, N. Calcium-based plasticity model explains sensitivity of synaptic changes to spike pattern, rate, and dendritic location. Proc. Natl. Acad. Sci. 109, 3991–3996 (2012).

[13] Kim, S. et al. Experimental Demonstration of a Second-Order Memristor and Its Ability to Biorealistically Implement Synaptic Plasticity. Nano Lett. 15, 2203–2211 (2015).

[14] Zucker, R. S. & Regehr, W. G. Short-Term Synaptic Plasticity. Annu. Rev. Physiol. 64, 355–405 (2002).

[15] Bi, G. Q. & Poo, M. M. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. J. Neurosci. 18, 10464–72 (1998).

[16] Bienenstock, E. L., Cooper, L. N. & Munro, P. W. Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. J. Neurosci. 2, 32–48 (1982).

[17] Kirkwood, A., Rioult, M. G. & Bear, M. F. Experience-dependent modification of synaptic plasticity in visual cortex. Nature 381, 526–528 (1996).

[18] Salin, P. A., Scanziani, M., Malenka, R. C. & Nicoll, R. A. Distinct short-term plasticity at two excitatory synapses in the hippocampus. Proc. Natl. Acad. Sci. U. S. A. 93, 13304–9 (1996).

[19] Chang, T. et al. Synaptic behaviors and modeling of a metal oxide memristive device. Appl. Phys. A 102, 857–863 (2011).

[20] Yang, S. N., Tang, Y. G. & Zucker, R. S. Selective induction of LTP and LTD by postsynaptic [Ca2+]i elevation. J. Neurophysiol. 81, 781–7 (1999).

[21] Shouval, H. Z., Bear, M. F. & Cooper, L. N. A unified model of NMDA receptor-dependent bidirectional synaptic plasticity. Proc. Natl. Acad. Sci. U. S. A. 99, 10831–6 (2002).

[22] Graupner, M. & Brunel, N. Calcium-based plasticity model explains sensitivity of synaptic changes to spike pattern, rate, and dendritic location. Proc. Natl. Acad. Sci. 109, 3991–3996 (2012).

[23] Strachan, J. P. et al. State Dynamics and Modeling of Tantalum Oxide Memristors. IEEE Trans. Electron Devices 60, 2194–2202 (2013).

[24] Strukov, D. B. & Williams, R. S. Exponential ionic drift: fast switching and low volatility of thin-film memristors. Appl. Phys. A 94, 515–519 (2009).

[25] Nian, Y. B., Strozier, J., Wu, N. J., Chen, X. & Ignatiev, A. Evidence for an Oxygen Diffusion Model for the Electric Pulse Induced Resistance Change Effect in Transition-Metal Oxides. Phys. Rev. Lett. 98, 146403 (2007).

[26] Pershin, Y. V. & Di Ventra, M. Neuromorphic, Digital, and Quantum Computation With Memory Circuit Elements. Proc. IEEE 100, 2071–2080 (2012).

[27] Wang, Z. et al. Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing. Nat. Mater. 16, 101–108 (2016).

[28] Chang, T., Jo, S.-H. & Lu, W. Short-Term Memory to Long-Term Memory Transition in a Nanoscale Memristor. ACS Nano 5, 7669–7676 (2011).

Chapter 4 Memristor-Based Reservoir Computing

4.1 Introduction

Unlike feedforward networks which are good for static patterns such as images, recurrent neural networks are extremely valuable for spatio-temporal pattern processing as the loops in the network offers features in the temporal domain to efficiently compute temporal data. As a result, recurrent neural networks are often used for temporal tasks such as video and speech processing. A typical example is the long short-term memory (LSTM) [4], a special kind of recurrent and deep neural network, which has become the state-of-the-art technology for speech recognition. However, traditional recurrent neural networks are very difficult to train [5], as there are often many feedback loops inside the network and the training algorithm can be very complex.

Reservoir computing is an emerging neural network concept that offers the capability of efficient temporal processing but is much easier to train [6]. It exhibits state-of-the-art performance for processing empirical data, such as financial forecasting [7] and chaotic time series prediction [8]. Some speech recognition tasks can be successfully performed as well [9, 10]. Like many neural network concepts, reservoir computing is inspired by the concept that the brain processes information generating patterns of transient neuron activity (e.g. a changing reservoir state) excited by input sensory signals.

Traditional reservoir computing implementations have three distinct components: an input layer, the reservoir and an output layer, as shown in Figure 4.1. The input layer feeds the

input signals to the reservoir through fixed connections. The reservoir has many interconnected nonlinear nodes with internal feedback loops. After applying input signals, the network exhibits transient responses, which are read out at the output layer through a linear weighted sum of the internal nodes. However different from conventional recurrent networks, the connections within the reservoir remain fixed and do not require training. The only trainable part is the output weights connecting the reservoir nodes with the output, significantly reducing the training cost.

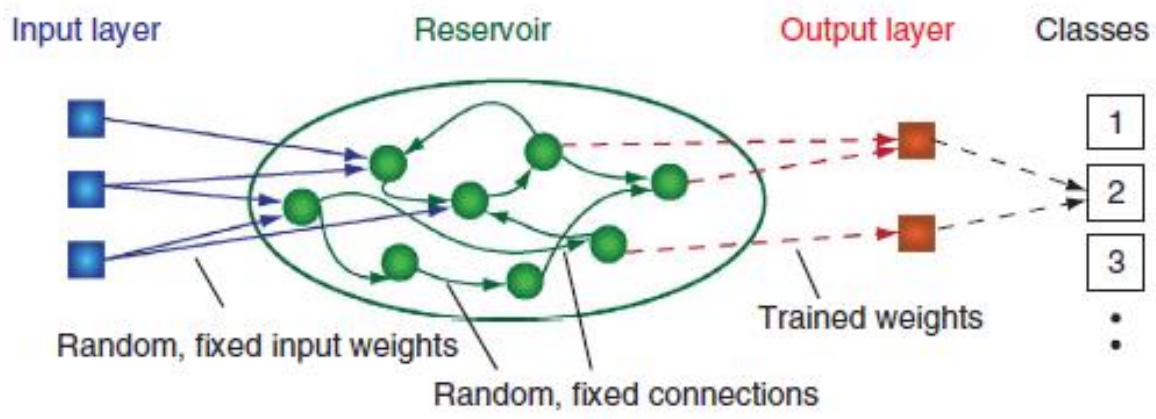


Figure 4.1 Classic reservoir computing scheme. The input is coupled into the reservoir through a randomly connected input layer to the N nodes in the reservoir. The connections between the reservoir nodes are random and fixed. The reservoir's transient dynamical response is read out by an output layer, which produces linearly weighted sums of the reservoir nodes $x_i(t)$, i.e. the output $y(t) = \sum_{i=1}^{N} w_i \cdot x_i(t)$. The coefficients w_i are trainable to best suit the target classes. From [1].

A good reservoir should nonlinearly transform the input signal into a high dimensional feature space (reservoir dynamics states) in which the signal is represented. This mapping allows the different inputs to be linearly separable so classification or other analysis can be achieved in the readout layer. The nonlinear mapping is normally achieved by using a large number of reservoir nodes with carefully designed connecting weights. Usually it requires several hundreds

or thousands of nonlinear nodes to obtain good performance. A second criterion is the reservoir should exhibit a fading memory or short-term memory: the reservoir state should be influenced by inputs from the recent past but not the far past. This property is essential for temporal sequence processing, such as speech.

4.2 Reservoir Built with a Nonlinear Node with Delayed Feedback

Another way to implement reservoir computing is by replacing the reservoir structure of multiple connected nodes with a dynamical system having a nonlinear node subjected to delayed feedback [1], as shown in Figure 4.2. A key feature of the continuous-time delay systems is that the dimension of the state space becomes infinite, as their state at time t depends on the output of the nonlinear node during the continuous time interval $[t-\tau, t]$, τ being the delay time. The dynamics of the delay system thus satisfy both the high dimensionality and the short-term memory requirement of the reservoir.

To be specific, with one delay interval of length τ, we define N equidistant points separated by τ/N. These N equidistant points are the "virtual nodes" in the reservoir. The values of the delayed variable at each of the N points are the states of the virtual nodes. These states characterize the transient response of the reservoir to a certain input at a time, and can thus emulate a network serving as reservoir.

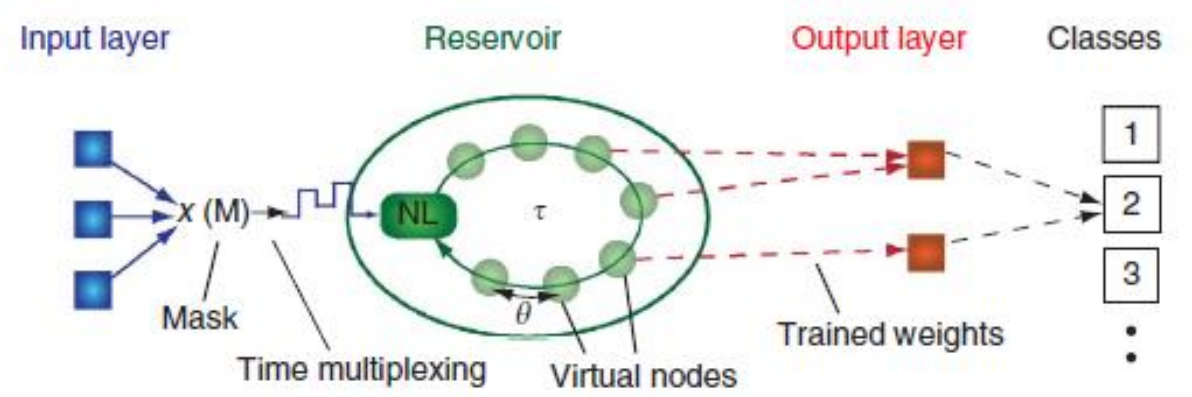


Figure 4.2 Scheme of reservoir computing using a nonlinear node with delayed feedback. A reservoir is obtained by dividing the delay loop into N intervals using time multiplexing. The input states are sampled and held for a duration τ, which is the delay in the feedback loop. The output nodes are linear weighted sums of the tapped states in the delay line $\sum_{i=1}^{N} w_i \cdot x(t - \frac{\tau}{N}(N - i))$. From [1].

Reservoir computing utilizing the delay system as the reservoir has been demonstrated using photonic systems [2]. A long optical fiber implements the delayed feedback loop, and an integrated telecom Mach-Zehnder modulator provides an electro-optic nonlinear modulation transfer function. The experimentally implemented optical based reservoir computing using a single nonlinear optoelectronic device subjected to delayed feedback has been used to successfully classify spoken digits [2].

4.3 Memristor as a Reservoir

As discussed in Chapter 3, the WO_x memristor device exhibits native short-term memory (i.e. volatile) behavior. This behavior is clearly illustrated in an experiment in which pulse stream composed of write pulses having the same amplitude (1.4 V, 500 µs) but at different timeframes are applied to the device, and the response of the memristor, which is represented by the read current through a small read pulse (0.6 V, 500 µs) following each write pulse, is

monitored. The results are shown in Figure 4.3. Two properties, similar to results obtained in dynamic synapses, can be observed: 1) if multiple pulses are applied with short intervals, the response will gradually increase (as indicated by the red arrow in the figure), showing an accumulation effect, 2) if there is a long enough period without any stimulation, then the device state will decay towards the original resting state, as indicated by the green arrow in the figure.

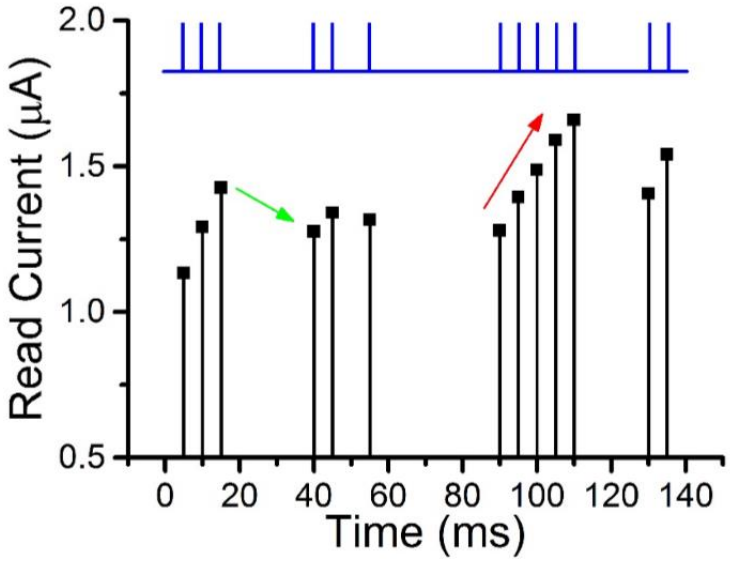


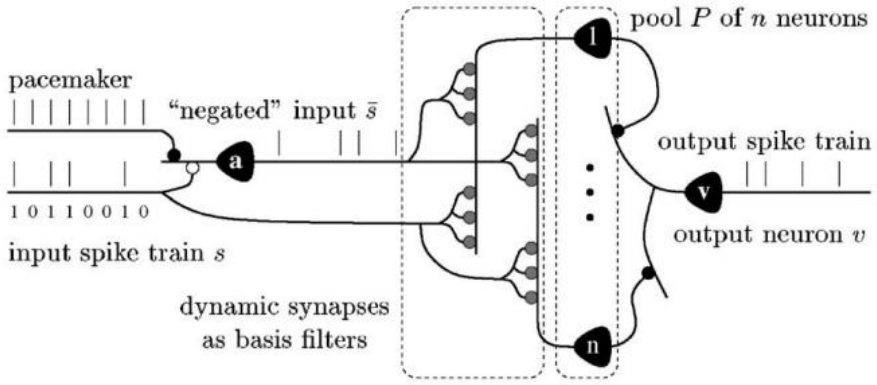
Figure 4.3 Response of a typical WO_x memristor to a pulse stream with different time intervals between pulses.

This temporal response is attributed to the internal ionic processes of the WO_x memristor, including the drift under electric field during the spike and the spontaneous diffusion of oxygen vacancies after the spike, as discussed in Chapter 3. The memristor's short-term memory effect can be described by a time constant τ, and for devices used in this study is ~ 50 ms. As a result, when programming the device, the device state depends not only on the programming pulse itself, but also depends on whether other programming pulses have been applied in the

immediate past within a period of ~50 ms. Prior programming pulses applied within this range will affect the device state, with pulses applied closer to present time having a stronger effect, while events happened much earlier will not affect the present device state since the device would have decayed to the initial state already.

Such temporal response can make the WO_x memristor suitable for reservoir computing based on the delayed system approach. In a theoretical study [18], a dynamic reservoir using basis filters as delayed lines was proposed using dynamic synapses (Figure 4.4). The basis filters should be carefully designed to provide a proper set of delayed lines. The output is computed directly from the input and the delayed versions of it. A proposed biological plausible implementation for the set of basis filters is by using numerous biologically realistic dynamic synapses with a uniform delay. The dynamics or the temporal response of the proposed synapses are in fact very similar to what we found in the WO_x memristors. As a result, we expect by extracting the memristor response in the temporal domain with a fixed interval/delay, a proper reservoir can be successfully built.

(a)



(b)

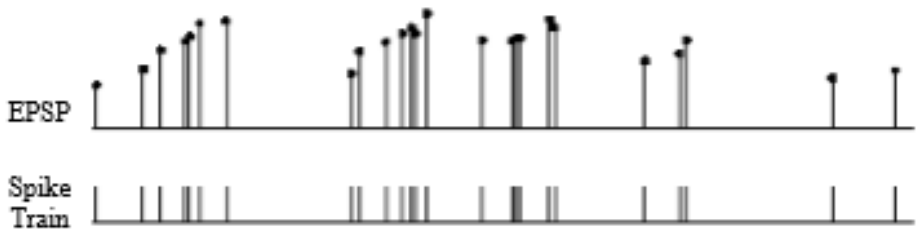


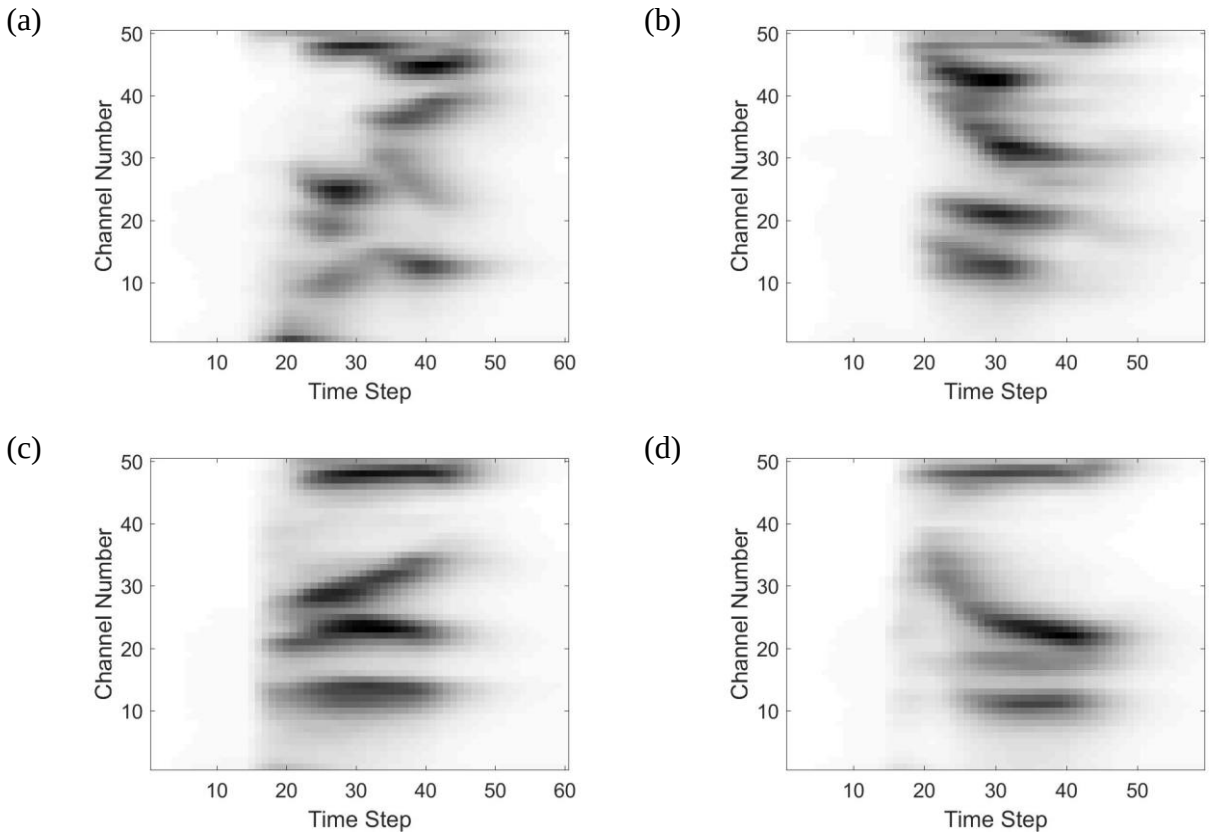
Figure 4.4 Using dynamic synapses as delay lines to build a reservoir. (a) A network of spiking neurons with dynamic synapses as basis filters to provide delayed lines. (b) Dynamic synapses' response to a spike train. The EPSPs generated by synapse are different depending on the timing of spikes. From [18].

4.4 Spoken Digit Recognition Task

As discussed earlier, our dynamic memristive devices show both short term memory and non-linear I-V relationship. So it is intuitive to use memristors for reservoir computing. Here we choose WO_x devices as our nonlinear nodes to build a reservoir for a standard benchmark speech recognition task – isolated spoken digit recognition.

The input dataset used for the test comes from the widely-used TI-46 speech corpus [11], which was originally designed and collected at Texas Instruments, Inc. (TI) in 1980. The 46-word vocabulary consists of two sub-vocabularies: (1) the TI 20-word vocabulary (consisting of the digits zero through nine plus the words "enter," "erase," "go," "help," "no," "rubout," "repeat," "stop," "start," and "yes" as well as (2) the TI 26-word "alphabet set" (consisting of the letters "a" through "z"). The corpus contains read utterances from 16 speakers (eight males and eight females) each speaking 26 utterances of the 46-word vocabulary: 16 tokens designated as training and ten as test. Here we are only using part of the TI 20-word vocabulary, digits from 0 to 9. The recorded sounds were all sampled at 12.5 kHz.

The input for the reservoir is a set of 50-dimensional state vectors (frequency channels) with up to 100 time steps. Each of the inputs is a spoken digit, preprocessed using the Lyon's passive ear model [12], which is a biologically plausible model based on human cochlear channels. The inputs representing digit 0-9 after being processed by the cochlear ear model are shown in Figure 4.5. Each data point on the graph stands for the firing probability of each neuron corresponding to each frequency channel (50 channels in total on the y axis) at each time point (x-axis).



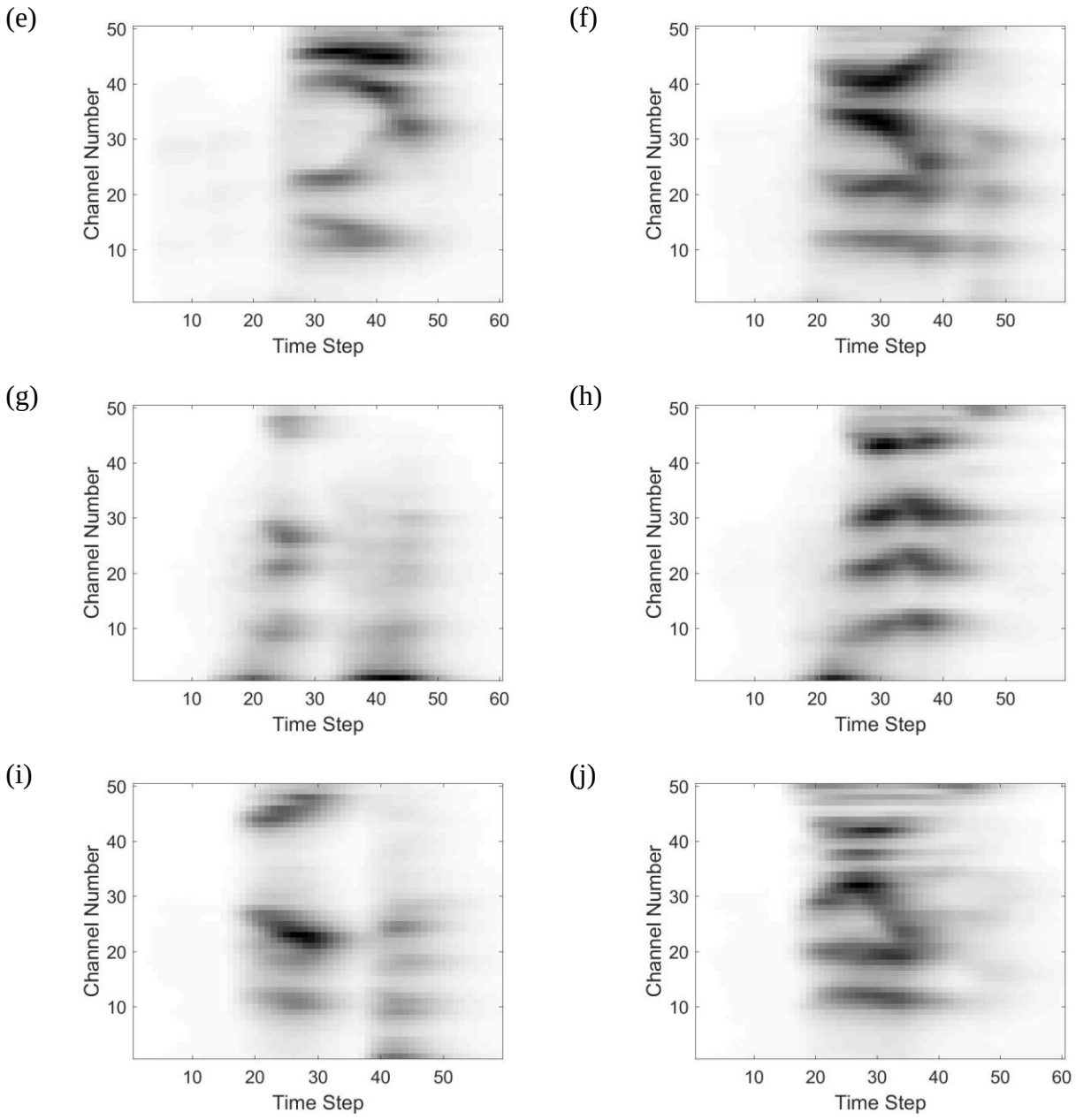
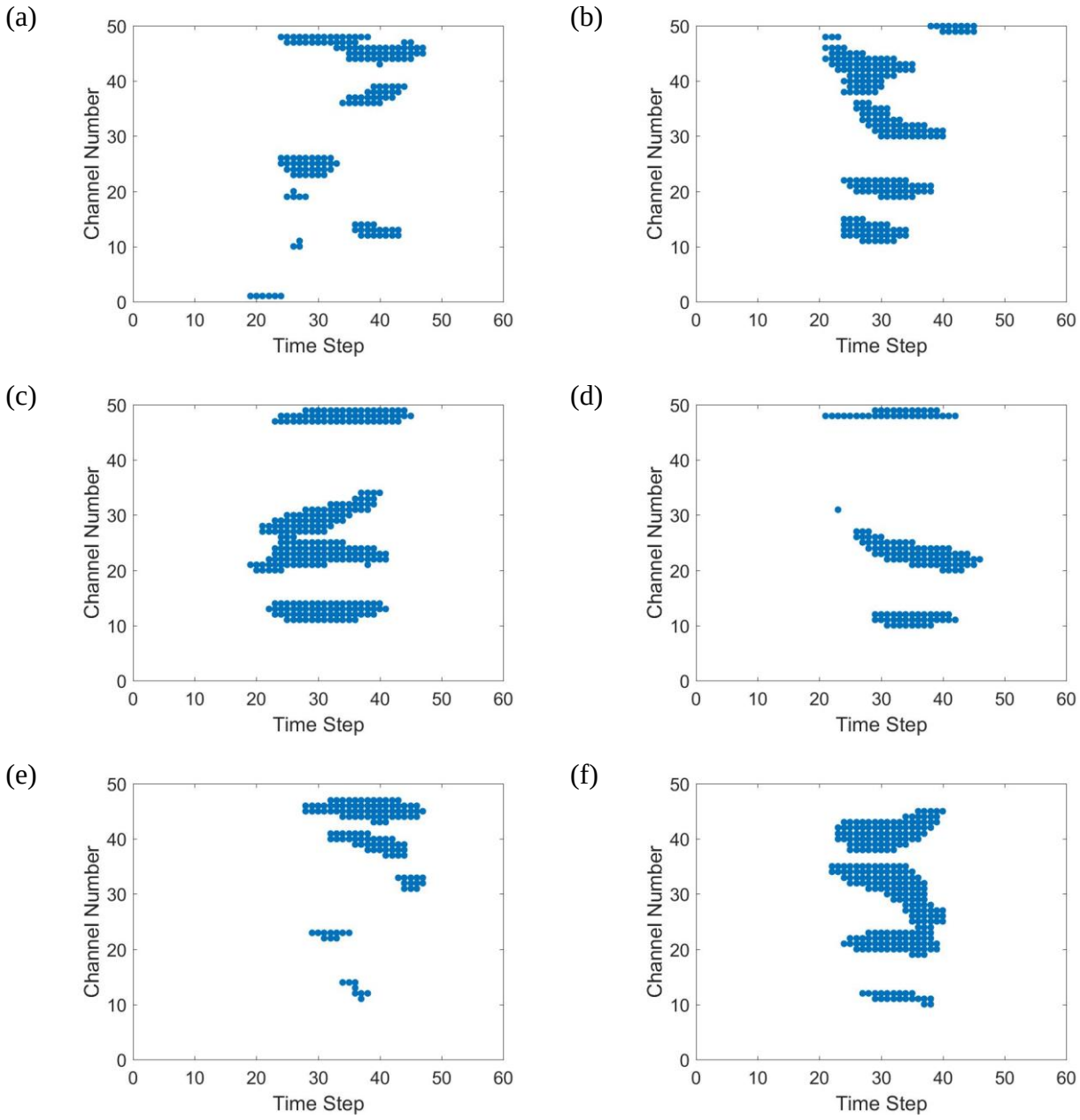


Figure 4.5 Cochleagrams of the ten input digits 0-9 after processing the 1-dimensional sound waveform signal using the Lyon passive ear model. These are taken from the first utterance of the first female speaker. Different speakers and different utterance number should give the cochleagrams some variations. Digit (a) 0. (b) 1. (c) 2. (d) 3. (e) 4. (f) 5. (g) 6. (h) 7. (i) 8. (j) 9.

To implement the reservoir computing concept using our memristors, the input graph (also termed cochleagram) are digitized into discrete input pulse trains. When the firing

probability is higher than 0.5, a spike is placed at this spot. When the firing probability is lower than 0.5, there will be no spike. With this approach, the analog cochleagrams are transformed into digital spike streams as the inputs to the reservoir (Figure 4.6).



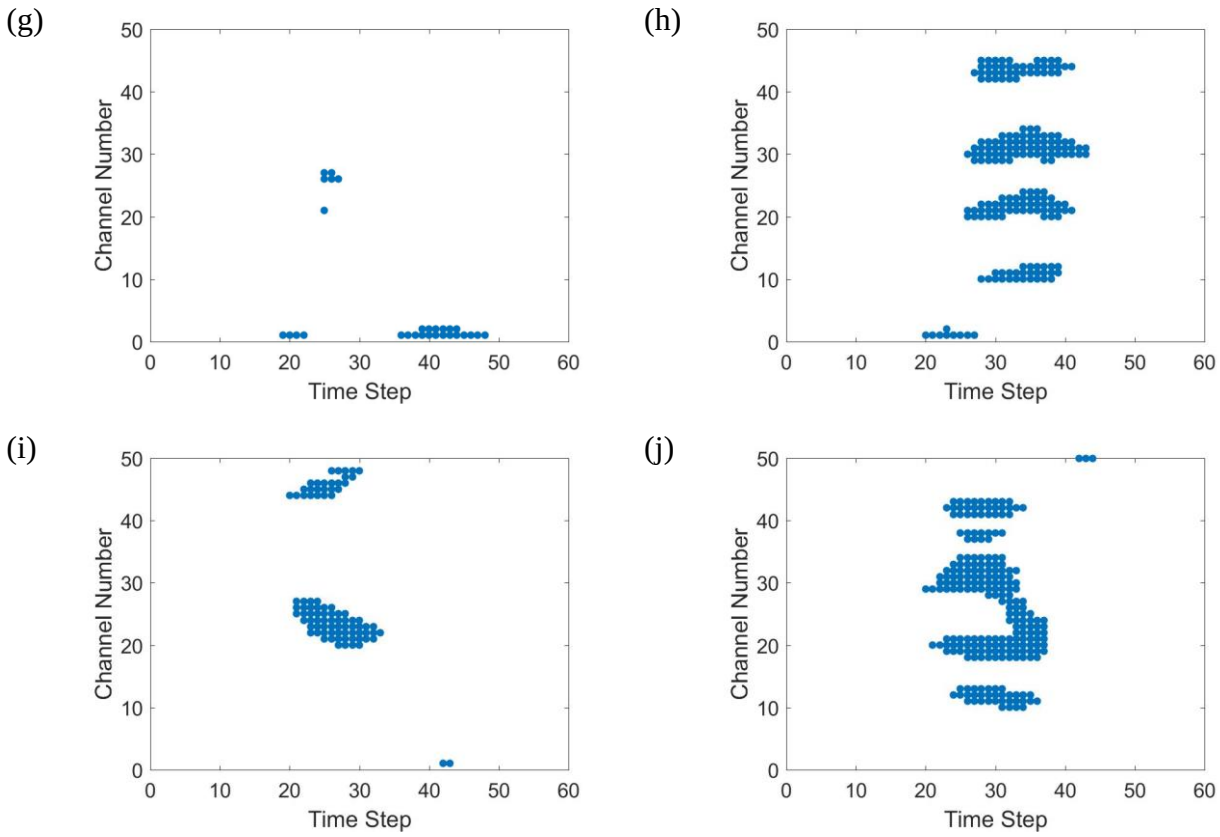


Figure 4.6 Digitized spike train graphs of the ten input digits 0-9 after setting a threshold for the previous analog cochleagrams. These are taken from the first utterance of the first female speaker. Digit (a) 0. (b) 1. (c) 2. (d) 3. (e) 4. (f) 5. (g) 6. (h) 7. (i) 8. (j) 9.

50 WO_x memristor devices are used for this isolated spoken digit classification task, with each device processing the input spike train at one of the 50 channels. The 50 devices were randomly chosen from a 32×32 crossbar array. Figure 4.7 (a) shows the schematic of using the memristor devices in parallel to process the input spike trains.

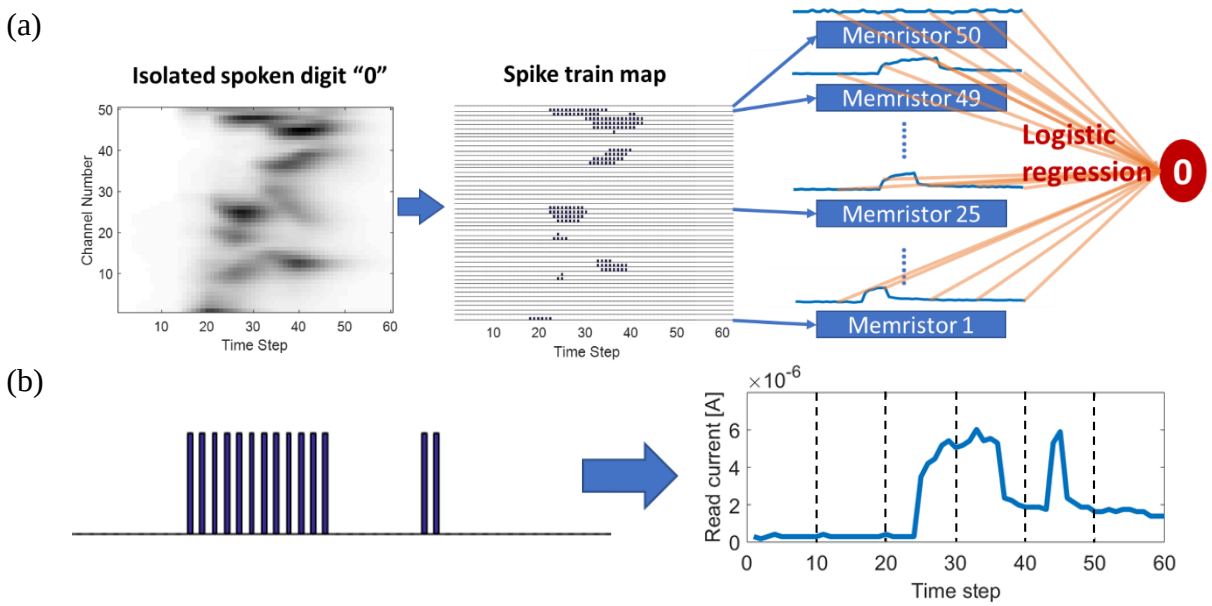


Figure 4.7 Schematic of using multiple memristor devices in parallel to process the digitized input spike trains. (a) Each memristor will process the spike train from one frequency channel. The readout layer was trained offline using logistic regression to classify the input from the memristor-based reservoir states taken from the memristor's temporal response. Here the temporal response is divided into 5 intervals making a total of 250 virtual nodes in the reservoir. (b) Nonlinear temporal response of a memristor to the example spike train taken from female speaker 1, first utterance, frequency channel 48. Left: input spike train from the channel. Right: memristor's current response when subject to the spike train on the left. The current response in the temporal domain can be divided into different number of intervals to implement the delay system.

Here the collection of memristors form the reservoir, and the reservoir state is represented by the combination of the memristor conductance values. Because of the short-term dynamics of the memristors, the temporal response of the reservoir to the input spike train can be transient and analog, as shown in Figure 4.7 (b). Here response from a single device is plotted. The spike train, e.g. obtained from female speaker 1, first utterance, frequency channel 48 (left panel), is fed to a memristor, and the right panel shows the temporal current response of this memristor to the spike train, showing rich nonlinear dynamics.

To implement the delay system, we performed time-multiplexing on the nonlinear temporal response from the memristor node, by dividing the whole sequence into n intervals.

Specifically, we use the current data points from time step 60/n, 60×2/n, 60×3/n, 60×4/n … 60×(n-1)/n, 60 as the neuron activities in the reservoir. One example of n = 5 is shown in the schematic in Figure 4.7 (a) and one example of n = 6 is shown in the right plot in Figure 4.7 (b). As there are 50 frequency channels, and n time step points are used in each channel, there are n×50 = 50n virtual nodes in total in the reservoir. Logistic regression is used for training the final 50n×10 readout network, with the 10 output nodes stand for 10 different digits. During compute, the test data (not in the training data) are fed into the reservoir and prediction is made from the readout function based on the reservoir state for each test case. The recognition rate is calculated in general as well as for each individual digit, as shown in Figure 4.8 for a case when n = 5. Good agreement was obtained between experiment and simulation.

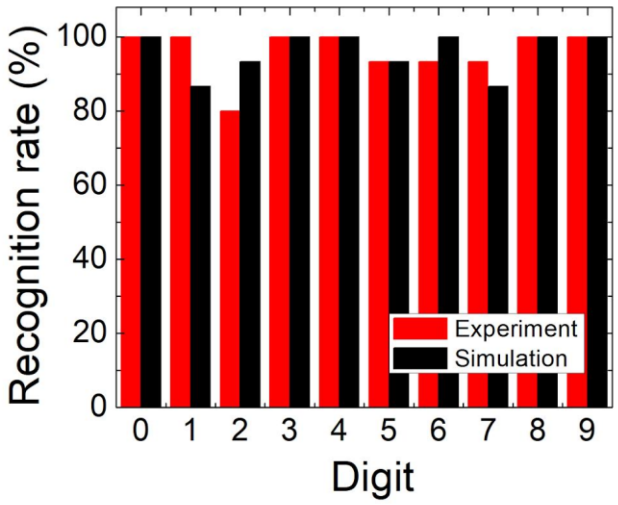


Figure 4.8 Isolated spoken digit recognition rate for individual digits. 250 virtual nodes were used, and the overall recognition rate is 98% in simulation and 97% in experiment.

We have also tested the dependence of the network performance on the number of virtual nodes (determined by n). We have tested cases with n = 1, 2, 3, 4, 5, 6, 10, 15, 20, 30. The recognition rate increases when n increases, i.e. more virtual nodes are used that results in a

larger reservoir (Figure 4.9). A clear improvement over the control system without the memristor reservoir is also observed. The improvement is also more pronounced at smaller network sizes. Further optimization of the device parameters (e.g. using devices having different decay time constants that match with the input temporal features) and the algorithm may lead to even further improvements in the system performance.

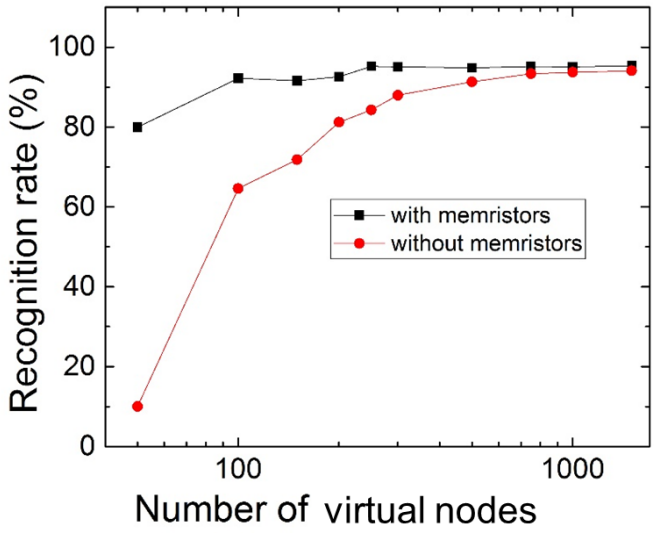


Figure 4.9 Simulated isolated spoken digit recognition rate with different number of virtual nodes (reservoir size) by dividing the temporal response into different number of intervals, based on time multiplexing on a 50 channel, 60 time step cochleagram.

4.5 Autonomous Signal Generation Task

Another benchmark test for reservoir computing is the autonomous time series generation. Reservoir computing can be used for short-term prediction tasks that focus on generating a few future time steps. However, for long term forecasting, traditional reservoir computing architecture needs to be modified by feeding the output back into the reservoir as the next input (Figure 4.10) [3]. A typical task is long term prediction of chaotic series.

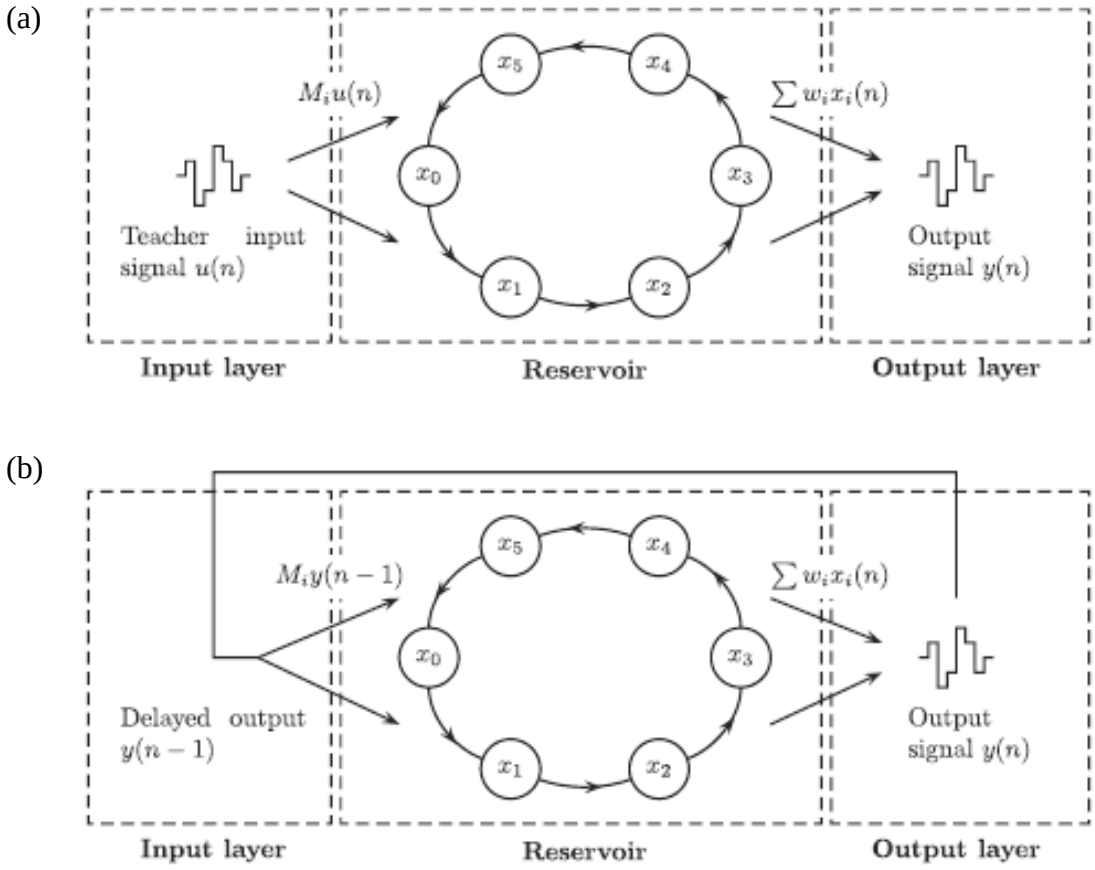


Figure 4.10 Schematic of the training process and autonomous signal generation of reservoir computing. (a) During training, the readout weights w_i are optimized for output to be as close as possible to u(n+1) when the input is u(n). (b) During autonomous run, the reservoir is driven by its own output. w_i are kept constant and the system is evaluated in terms of how long it can generate the desired output. From [3].

Reservoir computing systems with the output feedback essentially mean the input signal being its own output delayed by one time step, y(n-1). During training, the reservoir is driven by a teacher signal u(n), and the system is trained to predict the next value of the teacher time series u(n+1). Then, the reservoir input is switched from the teacher sequence to its own output signal

y(n-1) and the system is left running autonomously. If the system functions properly, the reservoir output y(n) will be the same as the rest of the teacher sequence not used for training.

Feeding the output back into the reservoir allows the network to autonomously generate time series without external input. A standard benchmark task is to generate the Mackey-Glass equation time series $\frac{dx}{dt} = \beta \frac{x(t-\tau)}{1+(x(t-\tau))^n} - \gamma x(t)$, which is a non-linear time delayed differential equation that can display a wide range of periodic and chaotic behaviors depending on the values of the parameters. The equation was originally studied by Mackey and Glass [19] to illustrate the appearance of complex dynamics in physiological control systems by way of bifurcations in the dynamics. Many oscillatory and chaotic physiological disorder or disease patterns can be characterized by changes in the parameters of the system, such as β, τ, and γ.

We studied the capability of memristor-based reservoir for long term prediction through simulation using the realistic WO_x model discussed in section 2.5 with a simple exponential decay term $\frac{dw}{dt} = \eta_1 \sinh(\eta_2 V) - \frac{w}{\tau}$ on the dynamic equation. Here, 50 independent memristor devices are used as nonlinear nodes, and time-multiplexing is used to implement the virtual internal nodes in the reservoir. Specifically, the virtual nodes are implemented as delayed response from the same memristor device when subject to the Mackey-Glass time series signal. Figure 4.11 shows the first 1000 time steps of a Mackey-Glass equation time series that shows slight chaotic behavior and will be used for the training and autonomous run (prediction).

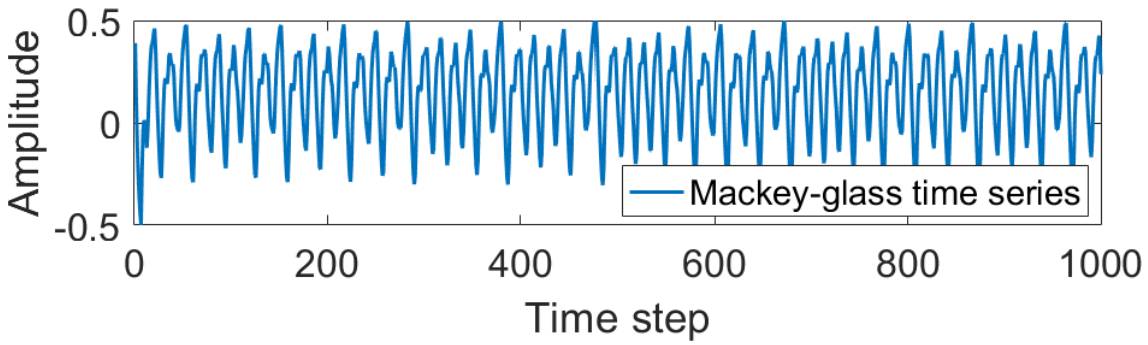


Figure 4.11 The first 1000 time steps of a Mackey-Glass equation time series. Parameters: β = 0.2, γ = 0.1, τ = 18, n = 10.

Since the memristor's response to applied voltage is nonlinear, the devices also naturally perform a nonlinear transformation of the input voltage to the output current. Initially, 2500 virtual nodes are used in the reservoir for the chaotic signal generation. In other words, 50 previous points on each of the 50 memristor current response curves (multiple colored curves in Figure 4.12 (a)) are used to train one output point on the prediction curve (blue curve in Figure 4.12 (a)). The different memristor responses were obtained by shifting the Mackey-Glass time series input voltage signal by different amount of voltages values. Afterwards, the system is left free running by connecting the predicted output as the next input to the system. Excellent results were obtained in the memristor-based system, as shown in Figure 4.12 (a) and (c). As a comparison, a control system without using the memristors could not perform successful prediction at all (Figure 4.12 (b) and (d)).

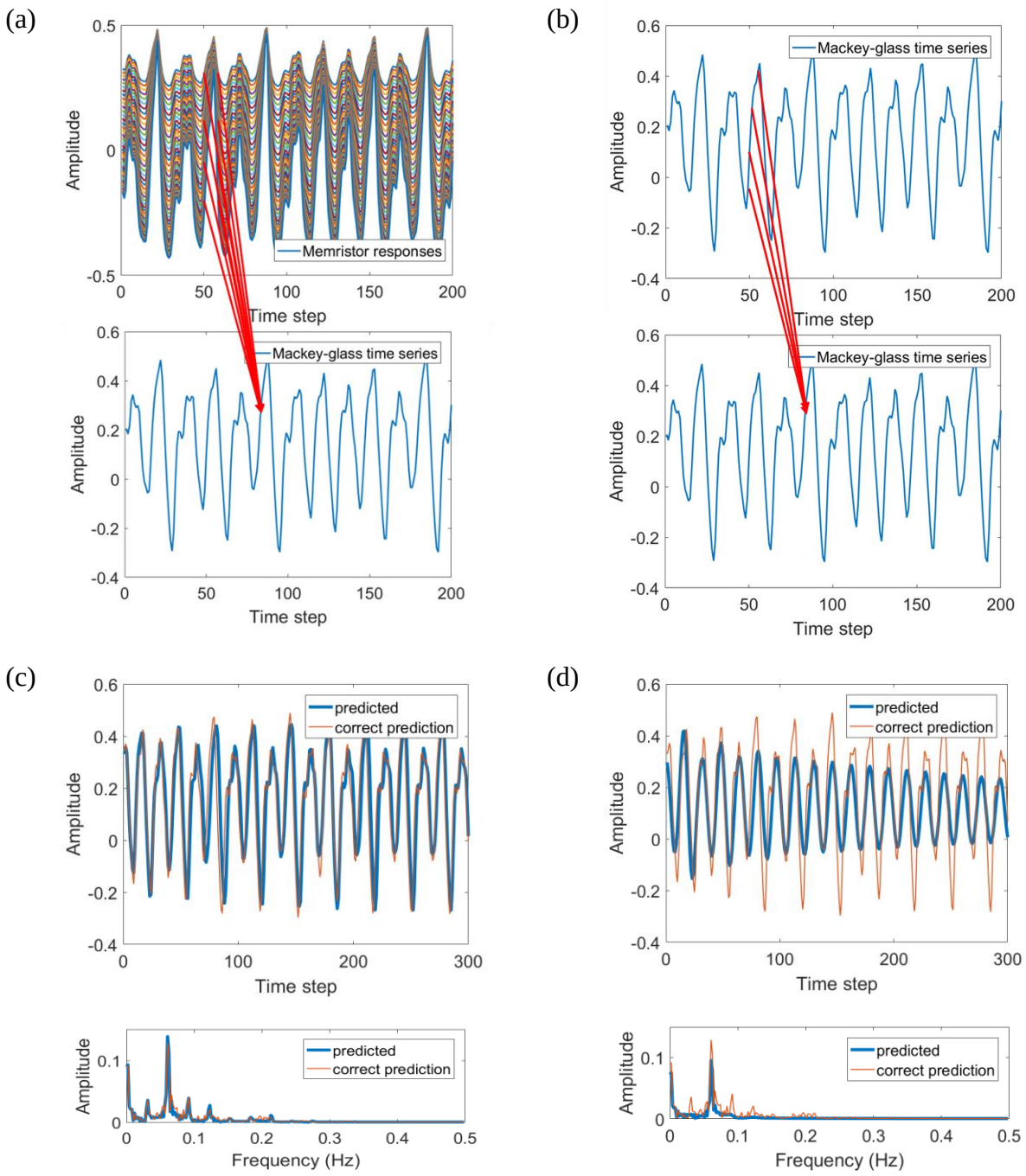


Figure 4.12 Training scheme and prediction result using the memristor or without using the memristor. The results were obtained through simulation. (a) With the memristor, training was performed using the n to n+m-1 points on the memristor response curves to train the readout function to get the output of the n+m point on the Mackey-Glass curve, and then using the n+1 to n+m points on the memristor response curves to get the output of the n+m+1 point on the Mackey-Glass curve, and so on. (b) On a control system without the memristor, the training was performed using the n to n+m-1 points on the Mackey-Glass curve to train the readout function to get the output of the n+m point on the Mackey-Glass curve, and then using the n+1 to n+m points on the Mackey-Glass curve to get the output of the n+m+1 point on the Mackey-Glass

curve, and so on. Training method in both cases is linear regression. (c) With the memristor-based reservoir, the network is left running freely with its own output as the next input. Excellent prediction results were obtained without noticeable discrepancies during the first 300 predicted time steps. Lower panel shows the Fourier transform of the predicted signal and the correct signal. Sampling frequency is 1 Hz. (d) Without the memristor prediction is unsuccessful. The Fourier transform shows that the predicted signal can capture the major frequency in the original time series but cannot capture the chaotic behavior.

We also changed the number of virtual nodes in the reservoir to explore the quantitative performance of the Mackey-Glass time series prediction task. Figure 4.13 plots the error from the Fourier transform of the first 300 predicted time steps by using different numbers of points on the memristor response curve (memristor approach) or the Mackey-glass curve (non-memristor approach) for training and prediction. The memristor approach started with a lower square error (MSE) than the non-memristor approach at very few reservoir virtual nodes, and it began to improve quickly at reservoir size = 50 (devices) × 20 (previous points) = 1000 and produced satisfactory outputs roughly at reservoir size = 50 (devices) × 50 (previous points) = 2500. However, with the non-memristor approach, the MSE dropped slowly and stayed at a high value. The network based on the non-memristor approach can only capture the major frequency component and is not suitable for chaotic series prediction. One typical example of a complete failure based on the non-memristor approach to make long term predictions is shown in Figure 4.12 (d).

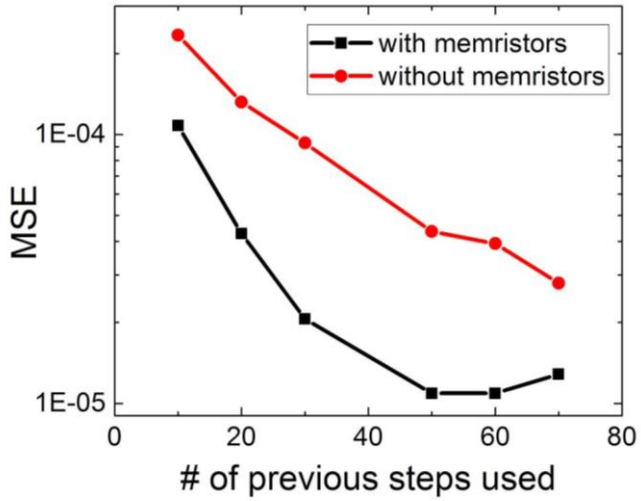


Figure 4.13 Mean square error of the Fourier transforms of the first 300 time steps of prediction running by using different numbers of points on the memristor response curve (memristor approach) or non-memristor approach. 50 memristor devices were used. For the memristor approach, the number of virtual nodes = number of memristor devices × number of previous steps used. For the non-memristor approach, the number of virtual nodes = number of previous steps used.

A systematic study on the effect of number of memristor devices used on the Mackey-Glass time series generation is also performed (Figure 4.14). In general, the MSE of the prediction based on the memristor approach gradually decreases with increasing number of memristor devices used. It shows that the non-linear I-V relationship transformation and the short-term decay dynamics naturally offered by the memristor allow the system to meet the two reservoir criteria: 1) non-linear mapping of the input onto a high dimensional space inside the reservoir. 2) short term (e.g. fading) memory property. By incorporating time-multiplexing and the delayed system concept, a high dimensionality can be created, and successful chaotic time-series prediction can be achieved.

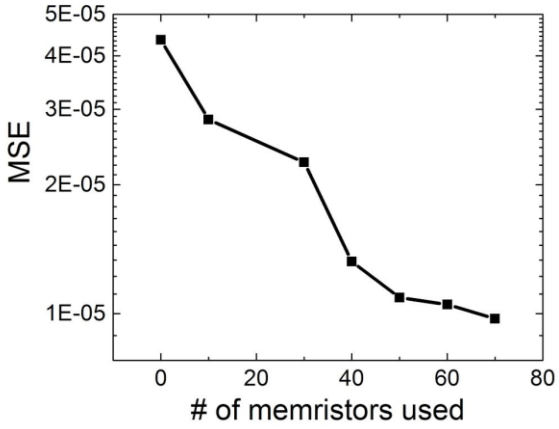


Figure 4.14 Mean square error of the Fourier transform of the first 300 time steps of prediction running by using different numbers of memristor devices. 50 previous time steps were used.

4.6 Conclusion

In conclusion, memristor-based reservoir computing approach has been demonstrated to be efficient and valid using the concept of delay systems. It can offer great advantage in hardware implementation since only a few (sometimes just one) devices are needed. Two specific tasks are used for the evaluation of the reservoir, isolated spoken digit recognition and autonomous time-series generation. Other applications of reservoir computing involve phoneme recognition [13], water inflow forecasting [14] or financial forecasting [7]. Besides recent approaches using optical components [2, 15, 16, 17], this is the first demonstration of using nanoscale memristors to implement reservoir computing.

References

[1] Appeltant, L. et al. Information processing using a single dynamical node as complex system. Nat. Commun. 2, 468 (2011).

[2] Larger, L. et al. Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing. Opt. Express 20, 3241 (2012).

[3] Antonik, P., Haelterman, M. & Massar, S. Brain-Inspired Photonic Signal Processor for Generating Periodic Patterns and Emulating Chaotic Systems. Phys. Rev. Appl. 7, 54014 (2017).

[4] Hochreiter, S. & Schmidhuber, J. Long Short-Term Memory. Neural Comput. 9, 1735–1780 (1997).

[5] Bengio, Y., Simard, P. & Frasconi, P. Learning Long-Term Dependencies with Gradient Descent is Difficult. IEEE Trans. Neural Networks 5, 157–166 (1994).

[6] Lukoševičius, M. & Jaeger, H. Reservoir computing approaches to recurrent neural network training. Comput. Sci. Rev. 3, 127–149 (2009).

[7] Ilies, I., Jaeger, H. & Kosuchinas, O. Stepping forward through echoes of the past: forecasting with echo state networks. *Short report on the winning entry to the NN3 financial forecasting competition, available online at http://www. neural-forecasting-competition. com/downloads/NN3/methods/27-NN3 Herbert Jaeger report. pdf* 53 (2007).

[8] Jaeger, H. Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. Science (80-.). 304, 78–80 (2004).

[9] Verstraeten, D., Schrauwen, B., Stroobandt, D. & Van Campenhout, J. Isolated word recognition with the Liquid State Machine: A case study. Inf. Process. Lett. 95, 521–528 (2005).

[10] Verstraeten, D., Schrauwen, B. & Stroobandt, D. Reservoir-based techniques for speech recognition. in Neural Networks 1050–1053 (IEEE, 2006). doi:10.1109/IJCNN.2006.246804

[11] Liberman, M. "TI46 speech corpus." *Linguistic Data Consor tium, Available: http://www. ldc. upenn. eduiCatalog/CatalogEntry. jsp* (1993).

[12] Lyon, R. A computational model of filtering, detection, and compression in the cochlea. in ICASSP ’82. IEEE International Conference on Acoustics, Speech, and Signal Processing 7, 1282–1285 (Institute of Electrical and Electronics Engineers, 1982).

[13] Triefenbach, F., Jalalvand, A., Schrauwen, B. & Martens, J.-P. Phoneme Recognition with Large Hierarchical Reservoirs. Advances in Neural Information Processing Systems 23 23, 1–9 (2010).

[14] Sacchi, R., Ozturk, M. C., Principe, J. C., Carneiro, A. A. F. M. & Silva, I. N. da. Water Inflow Forecasting using the Echo State Network: a Brazilian Case Study. in 2007 International Joint Conference on Neural Networks 2403–2408 (IEEE, 2007).

[15] Duport, F., Schneider, B., Smerieri, A., Haelterman, M. & Massar, S. All-optical reservoir computing. Opt. Express 20, 1958–1964 (2012).

[16] Vandoorne, K. et al. Toward optical signal processing using Photonic Reservoir Computing. Opt. Express 16, 11182 (2008).

[17] Paquot, Y. et al. Optoelectronic Reservoir Computing. Sci. Rep. 2, 287 (2012).

[18] Natschläger, T. & Maass, W. Spiking neurons and the induction of finite state machines. Theor. Comput. Sci. 287, 251–265 (2002).

[19] Mackey, M. C. & Glass, L. Oscillation and chaos in physiological control systems. Science. 197, 287–9 (1977).

Chapter 5 Temporal Information Encoding with Ag_2S Memristors

5.1 Introduction

As mentioned in Chapter 1, solid state memristive devices [1-3] are popular candidates for efficient neuromorphic computing [4-6]. The internal dynamics of memristive devices [7-10] can further allow them to faithfully emulate a broad range of synapse behaviors. Beyond processing conventional computing tasks, the dynamic memristors can potentially directly process biological spiking data and may even lead to coupled bio/electronics for information storage, analysis and feedback in a closed loop.

To this end, we studied Ag_2S atomic switches [8] which can be programed with low voltage pulses (~80 mV), close to actual neuron action potential amplitudes. Very similar to the WO_x memristors, Ag_2S based memristor devices also show short term decay dynamics and incremental conductance change, and have rich dynamic behaviors in the temporal domain when subject to certain stimulation as a function of time, i.e. neuronal spike trains. Indeed, short term facilitation to long term potentiation transition [8] has also been demonstrated with Ag_2S atomic switch devices. With such low programming voltages and native dynamic processes, these devices can potentially enable direct coupling with biological systems.

In this chapter, the device structure and fabrication method of the Ag_2S memristor will be discussed first, and then short-term and long-term decay dynamics will be analyzed, and a detailed statistical analysis will be carried out by systematically subjecting the Ag_2S memristive

devices to pulse trains that emulate different spiking patterns. The temporal information of these spiking trains was successfully encoded in the analog switching probability distribution, and can be accurately captured by a statistical model. Such dynamic memristive devices can potentially be used for temporal information encoding.

5.2 Ag_2S Memristor: Device Structure and Fabrication

The Ag_2S resistive switching memory devices were fabricated on a 100 nm SiO_2 substrate with an active device area of 200 nm by 200 nm defined by the crossbar pattern, shown in Figure 5.1 (a). The 20nm thick palladium bottom electrode along with a 5 nm thick NiCr adhesion layer was patterned using E-beam lithography and deposited by E-beam evaporation. A 30 nm thick Ag film (200 nm by 200 nm patch patterned by E-beam lithography) was then deposited by E-beam evaporation and lift-off processes. To fully sulfurize the Ag film patch, anodic polarization [11, 12] was performed using the Ag/Pd electrodes on the sample as the working electrode, and an immersed Pt rod as the cathode. The two-electrode electrochemical cell was immersed in a 0.025 M Na_2S solution whose pH value was adjusted to 12.5 by adding NaOH powder. The current density used was 0.3 mA/cm^2. After sulfurization, a granular surface was observed under the SEM, shown in Figure 5.1 (b), which is consistent with the observations in the literature [12] using similar solution-based fabrication methods. The resulting Ag_2S layer thickness was increased to ~70 nm after sulfurization. Finally, the top Ag electrode (120 nm thick) was formed by e-beam lithography, e-beam evaporation and lift-off.

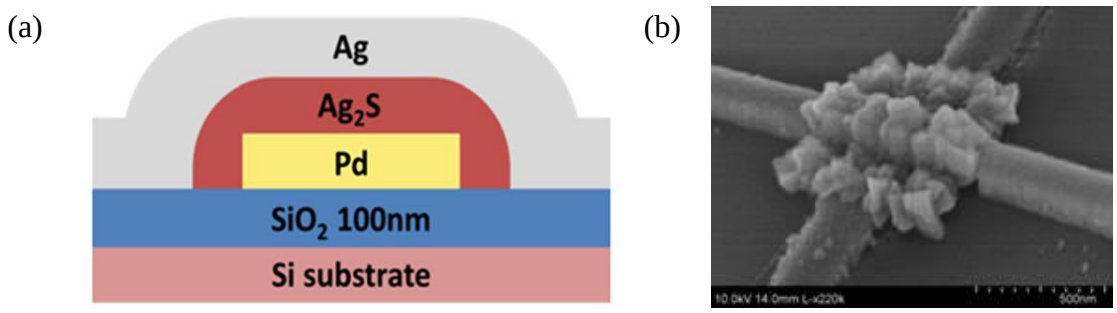


Figure 5.1 Ag_2S memristor device structure. (a) Device schematic showing the Ag/Ag_2S/Pd sandwich structure. (b) SEM image of a device. The Ag_2S film shows a granular structure.

The as-fabricated devices are initially conductive and show forming-free resistive switching behavior, possibly due to Ag ion diffusion into the Ag_2S switching layer during the e-beam evaporation process. Figure 5.2 shows a typical DC switching curve showing a standard digital-type (0 or 1) resistive switching process. 18 devices were tested and similar switching characteristics were observed with a relatively tight distribution of threshold voltage.

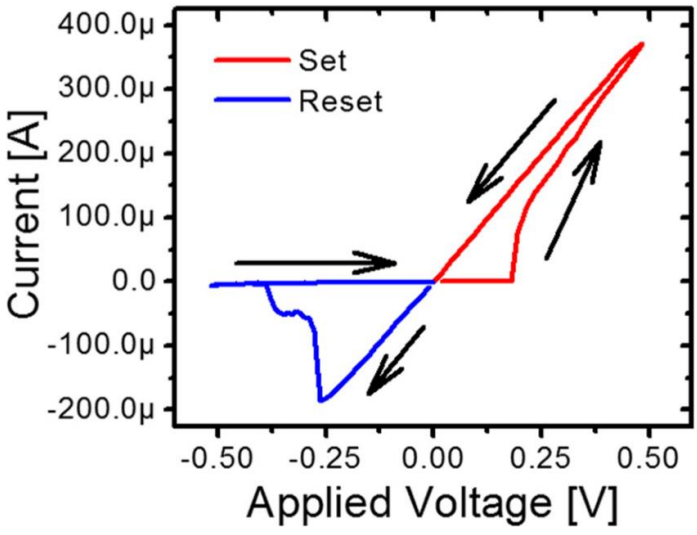


Figure 5.2 Typical DC switching curve of the Ag/Ag_2S/Pd device. The bias is applied on the Ag top electrode.

5.3 Short-term and Long-term Dynamics

Previous studies on Ag_2S based resistive switching devices have shown both short-term dynamics and long-term retention effects [8, 13]. During programming, Ag ions that are oxidized from the Ag top electrode (TE) will migrate towards the bottom electrode (BE) under the electric field, and reduce into Ag atoms to form a conducting filament between the BE and TE. However, if the metal filament size is very small, the diffusion of the Ag atoms can lead to gradual decrease of the conductance and eventual breakage of the filament, as shown in Figure 5.3.

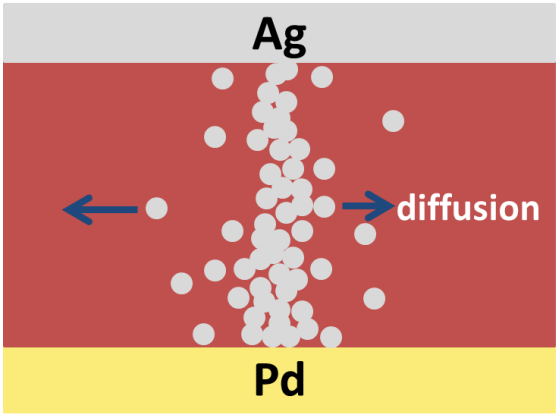


Figure 5.3 Schematic illustration showing an Ag metal filament formed between the TE and BE, and the spontaneous lateral diffusion of Ag that causes the gradual loss of conductance.

These short-term dynamics were also observed in our Ag_2S based devices where a single programming set pulse with high amplitude (0.45 V or higher) was used to program the device, and the device conductance was periodically checked with low-voltage read pulses. To ensure the best consistency, all measurements were carried out on the same single device, and similar effects have been observed consistently in all devices studied. Figure 5.4 shows typical decay

curves for the device after being programmed with a single 100 us long set pulse at three different pulse amplitudes (0.45 V, 0.60 V and 0.65 V). A clear decay can be observed in the device conductance after programming. The device relaxes back to the original high-resistance state within a few seconds, while slower relaxation was observed for stronger programming conditions.

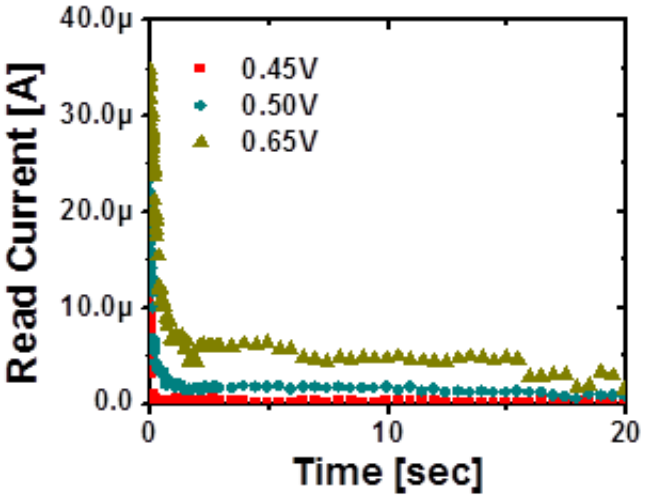


Figure 5.4 Different decay behaviors after stimulation with a single 100 µs long set pulse with different amplitudes of 0.45 V, 0.60 V, and 0.65 V. The read currents were measured by short 0.05 V read pulses.

This relaxation process can be explained by the spontaneous diffusion of the Ag atoms forming the filament. The spontaneous diffusion process in a disordered system can be modeled using a stretched exponential function:

$$I(\mathrm{t}) = I_0 \exp\left[-\left(\frac{t}{\tau}\right)^r\right] \tag{5.1}$$

Equation 5.1 was used to fit the decay curves in Figure 5.4 and good agreements were obtained, as shown in Figure 5.5. The decay time constant τ obtained from fitting was indeed found to increase with programming strength.

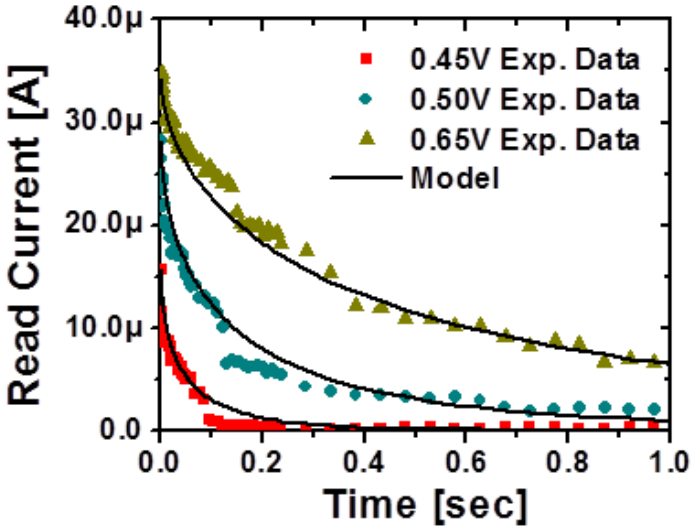


Figure 5.5 Fitting the decay curves in Figure 5.4 with the stretched exponential equation $I = I_0 \exp(-(t/\tau)^r)$. Parameters: $\tau = 0.0423$ s (0.45 V), 0.1344 s (0.50 V), and 0.4268 s (0.65 V), $r = 0.6$.

Figure 5.6 (a) and Figure 5.6 (b) show the decay curves of the device when subjected to even stronger programming conditions such as longer pulses. The decay still follows a stretched exponential decay at the short time scale, but discrete conductance drops were observed at longer time scales.

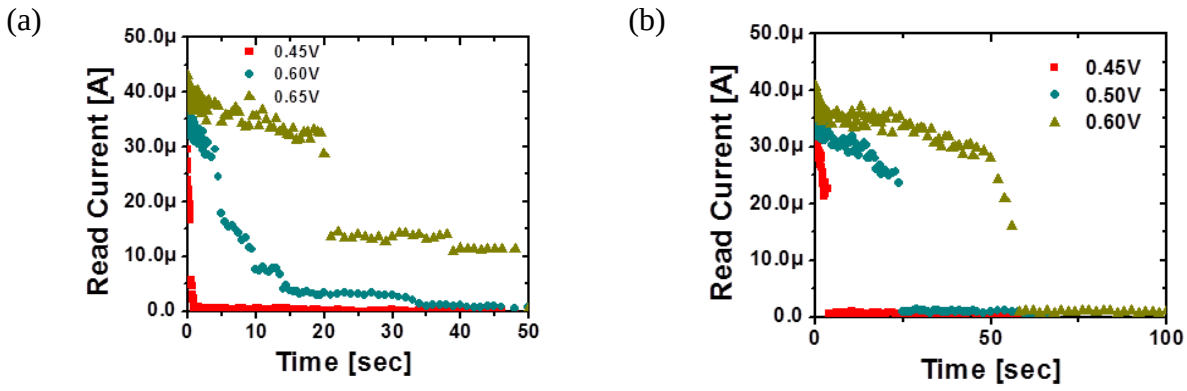


Figure 5.6 Conductance decay after stimulation with longer set pulses. 1 ms long in (a) and 2 ms long in (b) with different amplitudes. A gradual initial decay followed by discrete conductance drop(s) can be observed.

The continuous stretched exponential decay was caused by the gradual loss of the filament due to spontaneous diffusion in the beginning, while the sudden drops were due to the eventual abrupt breakage of the filaments when enough Ag atoms have gone missing and a gap in the filament is created, shown in Figure 5.7.

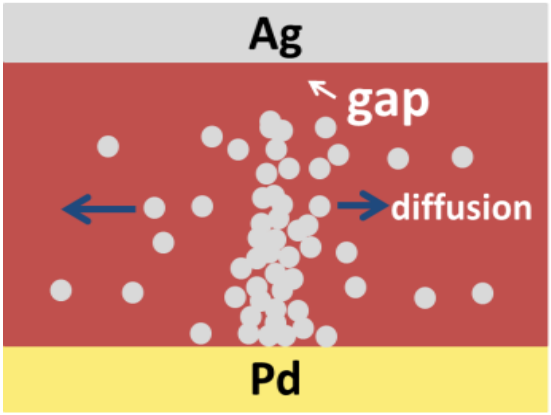


Figure 5.7 Schematic showing the eventual breakage of the filament due to gap formation that leads to abrupt conductance changes.

5.4 Encoding Temporal Information into Switching Probability Distribution

Previous studies on filament based no-volatile RRAM devices have shown that the switching probability is cumulative [14]. If the programming pulse amplitude is below the switching threshold, then the probability of switching is a linear sum of the switching probability of each pulse, regardless of the interval between pulses, shown in Figure 5.8. As an extreme example, a single long pulse has the same effect of a series of short pulses [14].

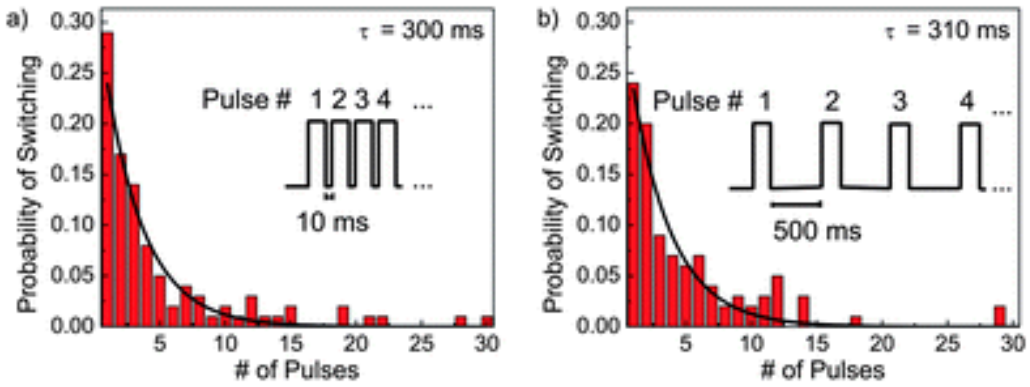


Figure 5.8 Switching probabilities when subjected to a series of short pulses. From [14]. The average switching time can be calculated by measuring the number of pulses, regardless of the gap between the pulses. The pulse width was kept constant at 100 ms while the gap was changed from 10 ms (a) to 500 ms (b). The voltage amplitude was fixed at 2.5 V for all pulses.

The situation can be very different if the device shows short term decay, exhibiting volatile behavior, so that the increase in switching probability is gradually lost over time, which could be used to encode temporal information of the input pulses/spikes in the switching characteristics of the memristive devices.

To test this concept, we applied a pulse train, each pulse with 1 ms duration, to a single Ag_2S device and studied the statistical distribution of when the device is switched. Experimental protocol is shown in Figure 5.9. The amplitude of the pulses was intentionally set to be below the

threshold so that switching normally does not occur at the first pulse but rather happens after the accumulation of the switching probability through several pulses. Here, 'switching' is defines as when the programming current exceeds a threshold value of 0.03 mA. After each test, the pulse number n at which the device is switched is recorded. Then device is reset, and tests are repeated. The statistical distribution of when switching occurs, the number of n, is analyzed for different programming conditions with different pulse amplitudes (0.12 V, 0.15 V, and 0.18 V) and different pulse intervals (dt = 1 ms and 2 ms).

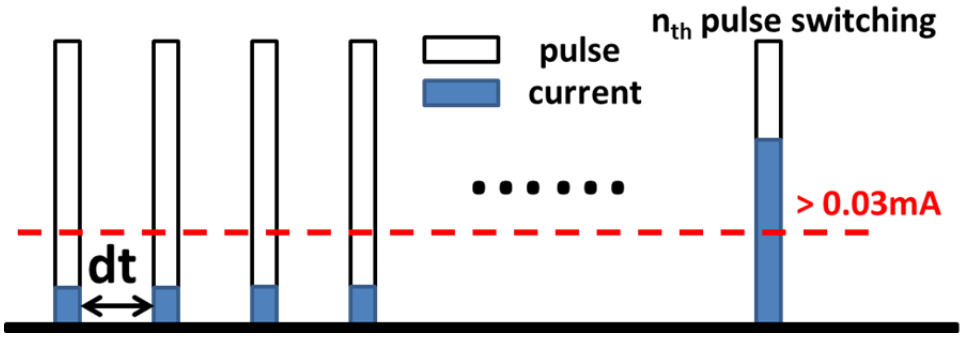


Figure 5.9 Pulse train experiment setup. The device current was monitored during the programming pulses. The device is considered switched ON if the measured current is above a threshold value (e.g., 0.03 mA).

Figure 5.10 shows the histogram (probability density function, pdf) of n through the measurements described above. Several interesting effects have been observed. First of all, different from non-volatile filament based devices that show a first order Poisson distribution where the maximum probability occurs near n = 0, here for pulses with low enough amplitudes (0.12 V and 0.15 V cases, Figure 5.10 (a), (b), (d) and (e)) a different distribution is observed with the maximum switching probability clearly shifted away. In addition, as the pulse interval is increased from 1 ms to 2 ms, the maximum switching probability peak also shifts to higher n

values, meaning the device is more likely to switch at a later event, along with a drop in the peak value. For higher pulse amplitudes (comparing the 0.12 V case with the 0.15 V case), the maximum switching probability shifts towards smaller n, accompanied by an increase in the peak value. At high enough pulse amplitude (like 0.18 V), the device behaves almost like the non-volatile case (Figure 5.8) with the maximum probability always appearing near n = 0 and little changes in the distribution were found when the interval changes from 1 ms to 2 ms.

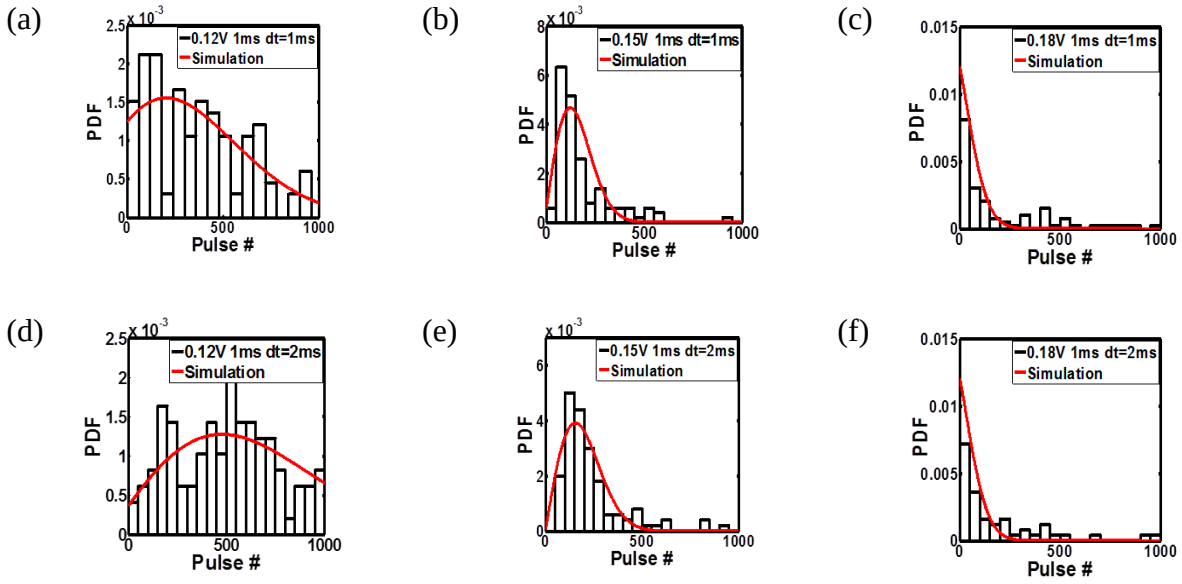


Figure 5.10 Switching probability density distribution of the device when subjected to pulse trains with different patterns. (a)-(c) Pulses having width of 1 ms and interval of 1 ms, with amplitudes of 0.12 V (a), 0.15 V (b), and 0.18 V (c), respectively. (d)-(f) Pulses having width of 1 ms and interval of 2 ms, with amplitudes of 0.12 V (d), 0.15 V (e), 0.18 V (f), respectively. Red lines correspond to the modeling results. The same set of parameters: $\lambda_1 = 3600$, $\lambda_2 = 8$, $\alpha = 0.0003$, $\beta = 40 \pm 2$, $\tau_0 = 0.000001$, and $r = 0.6$ were used to model all cases.

To quantitatively analyze these results, a statistical model was developed using the following equations:

$$\mathrm{T(n)} = \prod_{i=1}^{n-1}[1 - P(i)]P(n) \tag{5.2}$$

$$\mathrm{P(n)} = P(n-1)\exp\left[-\left(\frac{dt}{\tau(n)}\right)^{r}\right] + \delta P \tag{5.3}$$

$$\tau\mathrm{(n)} = \tau_0 + \lambda_1\{exp[\lambda_2 P(n-1)] - 1\} \tag{5.4}$$

$$\delta\mathrm{P} = \alpha\sinh(\beta\mathrm{V})\, p_w[1 - P(n-1)] \tag{5.5}$$

Here, T(n) is the probability density function (pdf) representing the probability that switching occurs at the n_{th} pulse, and is used to explain the experimental data in Figure 5.10. P(n) stands for the independent switching probability by a single pulse, regardless of whether a switching event has occurred before. The relationship between T(n) and P(n) can be found in Equation 5.2, meaning the probability of first switching at the nth pulse is the multiplication of probability of non-switching at the first n-1 pulses and switching at the nth pulse. P(n) increases by δP each time a subthreshold programming pulse is added to the existing pulse train, reflected in the second term in Equation 5.3. While the volatile nature also causes the existing switching probability to decay from the value obtained from the last pulse, shown as the first term in Equation 5.3.

The decay time constant τ(n) was hypothesized to be exponentially dependent on P(n) to capture the experimentally observed improvements in decay time at stronger programming conditions. The hypothesized exponential relationship was based on the extracted τ dependence on programming conditions where stronger programming was found to lead to an exponential like increase in τ based on the data shown in Figure 5.5.

The switching probability boost by a single pulse δP is dependent on the pulse amplitude V and the pulse width pw, and can be modulated by a window function 1-P(n-1) [15]. The window function reflects the decrease in the effectiveness of the boost with increasing P(n). Based on these equations, the probability that the device is switched at the nth pulse, namely T(n), can be calculated from Equation 5.2. All results for different input patterns were calculated using the same set of parameters: $\lambda_1 = 3600$, $\lambda_2 = 8$, $\alpha = 0.0003$, $\beta = 40 \pm 2$, $\tau_0 = 0.000001$, $r = 0.6$, and pw = 1ms.

The calculated PDFs agree reasonably well with the experimental results, shown in Figure 5.10. Specifically, the model correctly predicts the deviation from the Poisson distribution and the shift of the maximum switching probability peak to higher n numbers with longer intervals or lower programming amplitudes. Mathematically, this phenomenon can be explained by the competing effects of the $\prod_{i=1}^{n-1}[1 - P(i)]$ term and the P(n) term in Equation 5.2. The decay reduces P(n) but increases $\prod_{i=1}^{n-1}[1 - P(i)]$, which is a multiplication of a number of probabilities. So the decay effect is amplified through multiplication that the deviation from the Poisson distribution can be clearly seen for volatile programming cases (0.12 V and 0.15 V), where quite a few pulses are required to switch the device. For stronger programming conditions such as the 0.18 V case, P reaches a large value at the first pulse and only a few pulses are typically required to switch the device. As a result, the decay effect is less pronounced due to the less amplification in the $\prod_{i=1}^{n-1}[1 - P(i)]$ term, which leads to a more similar distribution to the Poisson form than the volatile case.

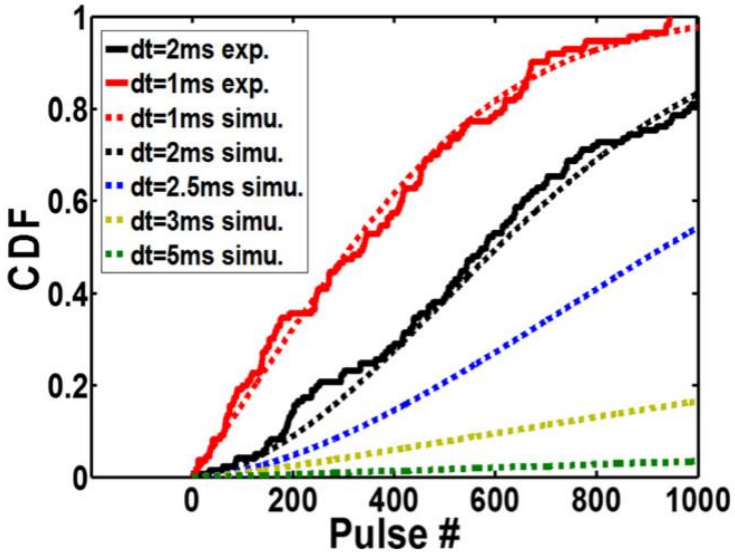


Figure 5.11 Experimental and simulated CDFs for different input conditions. Pulse amplitude = 0.12 V, width = 1 ms, with different pulse intervals. The CDF R(n) is calculated as T(1) + T(2) + … + T(n).

Now with the statistical model developed above, we can now predict how the switching probability evolves with different input conditions (pulse amplitude, width and interval). Figure 5.11 plots the calculated cumulative distribution function (cdf, the possibility that switching occurs after n pulses) for different input conditions. All pulses used are 0.12V and interval dt was changed. As can be seen in Figure 5.11, a high frequency signal (dt = 1 ms) results in a close-to-100% cumulative switching probability within 1000 pulses, while a low frequency input (dt = 2.5 ms) results in only 50% switching probability within the same number of applied pulses. Similar trends were observed in other pulse intervals and amplitudes as well.

5.5 Conclusion

In this project we studied the short-term dynamics in Ag_2S memristive devices and developed an approach to process temporal information. Different characteristics in response to

different input pulse train patterns shown in Figure 5.10 are direct results of the short-term dynamics of the Ag_2S memristive devices. It would be interesting to compare these results to the temporal features in biological systems. In biology, the action potentials are usually 1-2 ms long and firing rates ranges between 5 Hz and 200 Hz. In both our experimental and simulation work on the Ag_2S device, when the interval was set to be 0.2 s which corresponds to a very slow firing rate of 5 Hz, almost no switching events were observed after 1000 sub-threshold programming pulses (a very low switching probability). The Ag_2S device only has a reasonable chance to be programmed if a high frequency event (burst of spikes) occurred. As a result, an array of Ag_2S memristive devices can potentially be used for direct detection of a high frequency burst of biological spikes (action potentials). With the high integration density and large connectivity, such memristive devices with short term decay dynamins can offer unique opportunities for both conventional computing as well as brain/neural computer interface applications.

References

[1] Waser, R. & Aono, M. Nanoionics-based resistive switching memories. Nat. Mater. 6, 833–840 (2007).

[2] Sawa, A. Resistive switching in transition metal oxides. Mater. Today 11, 28–36 (2008).

[3] Waser, R., Dittmann, R., Staikov, G. & Szot, K. Redox-Based Resistive Switching Memories - Nanoionic Mechanisms, Prospects, and Challenges. Adv. Mater. 21, 2632–2663 (2009).

[4] Yu, S., Wu, Y., Jeyasingh, R., Kuzum, D. & Wong, H.-S. P. An Electronic Synapse Device Based on Metal Oxide Resistive Switching Memory for Neuromorphic Computation. IEEE Trans. Electron Devices 58, 2729–2737 (2011).

[5] Jo, S. H. et al. Nanoscale Memristor Device as Synapse in Neuromorphic Systems. Nano Lett. 10, 1297–1301 (2010).

[6] Kim, K.-H. et al. A Functional Hybrid Memristor Crossbar-Array/CMOS System for Data Storage and Neuromorphic Applications. Nano Lett. 12, 389–395 (2012).

[7] Chang, T., Jo, S.-H. & Lu, W. Short-Term Memory to Long-Term Memory Transition in a Nanoscale Memristor. ACS Nano 5, 7669–7676 (2011).

[8] Ohno, T. et al. Short-term plasticity and long-term potentiation mimicked in single inorganic synapses. Nat. Mater. 10, 591–595 (2011).

[9] Kim, S. et al. Experimental Demonstration of a Second-Order Memristor and Its Ability to Biorealistically Implement Synaptic Plasticity. Nano Lett. 15, 2203–2211 (2015).

[10] Du, C., Ma, W., Chang, T., Sheridan, P. & Lu, W. D. Biorealistic Implementation of Synaptic Functions with Oxide Memristors through Internal Ionic Dynamics. Adv. Funct. Mater. 25, 4290–4299 (2015).

[11] Lee, J. I., Howard, S. M., Kellar, J. J., Han, K. N. & Cross, W. Electrochemical interaction between silver and sulfur in sodium sulfide solutions. Metall. Mater. Trans. B 32, 895–901 (2001).

[12] Yin, S. Solution Processed Silver Sulfide Thin Films for Filament Memory Applications. Thesis Berkeley (2010).

[13] Ohno, T. et al. Sensory and short-term memory formations observed in a Ag_2S gap-type atomic switch. Appl. Phys. Lett. 99, 203108 (2011).

[14] Gaba, S., Sheridan, P., Zhou, J., Choi, S. & Lu, W. Stochastic memristive devices for computing and neuromorphic applications. Nanoscale 5, 5872 (2013).

[15] Chang, T. et al. Synaptic behaviors and modeling of a metal oxide memristive device. Appl. Phys. A 102, 857–863 (2011).

Chapter 6 Future Work

6.1 Introduction

In this Chapter, I will introduce two interesting projects that have only produced preliminary results so far. They are closely related to the field of neuromorphic computing as the first one is aimed at emulating a very fundamental synaptic behavior – synaptic competition effect, and the second one is aimed at emulating the spatial navigational system using the place cells and grid cells that generate firing patterns for spatial recognition and memory.

6.2 Synaptic Competition and Clustering

Synaptic plasticity is a very important neurochemical foundation for learning and memory. It is the ability of synapses to strengthen or weaken in response to activity. Such as in Hebbian learning [5], which explains a basic mechanism for synaptic plasticity during the brain learning process – "neurons that fire together wire together". A very early work [2] on using self-assembling connections of conducting particles on a liquid surface to emulate Hebbian learning is shown in Figure 6.1.

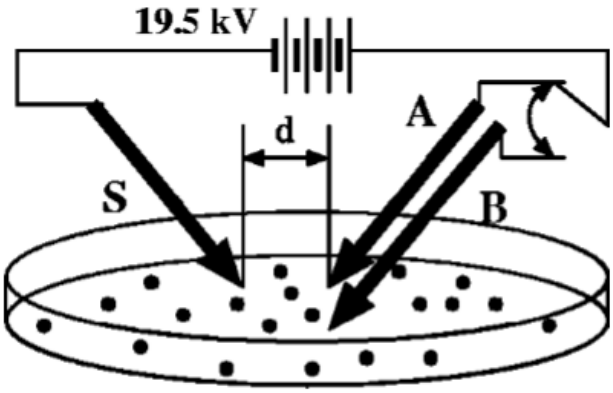


Figure 6.1 Hebbian learning experimental setup. Spherical steel particles are distributed randomly in a cylindrical cell filled with castor oil. The voltage of 19.5 kV is applied to S. A and B are connected alternately. From [2].

Steel particles (1 mm diameter) on castor oil (3 mm thick) was used as the neural network. Training was conducted by applying voltage pulses on electrodes A and B alternatively to stimulate the synapses between A and S and between B and S. The applied voltages as a function of time is shown in Figure 6.2.

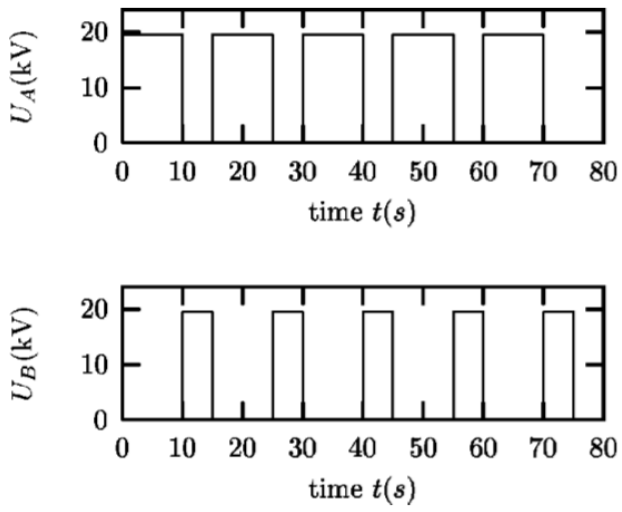


Figure 6.2 Applied voltage as a function of time in the Hebbian learning experiment. The voltage of 19.5 kV is applied to connection A-S for 10 s. Subsequently, electrode A is disconnected and

the voltage is applied to electrode for 5 s. The voltage is switched from A to B and vice versa five times.

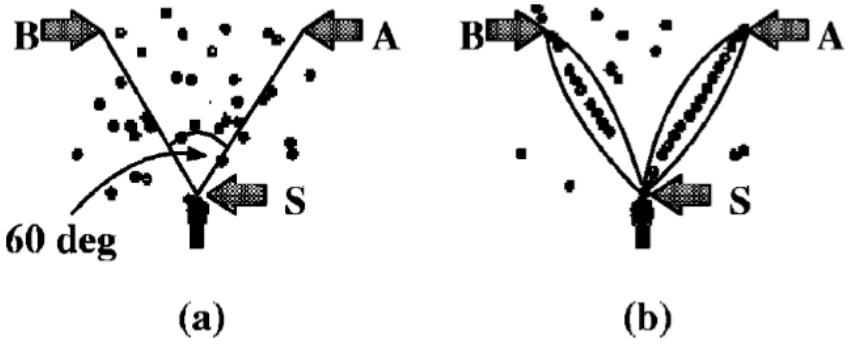


Figure 6.3 Particle distributions. From [2]. (a) Initially the particles are distributed randomly. The positions of the electrodes S, A, and B are indicated by the arrows. Axis A-S and axis B-S form a 60° angle. (b) The distribution of the particles after 60s, and the area used to determine the number of particles for each connection. Although the last 55-60 s were used to connect B-S, there are still more particles on the axis A-S.

As can be seen from Figure 6.3, even though the last stimulus was applied between electrodes B and S, the connection between A and S is stronger than that between B and S, as the connection strength (synaptic plasticity) represents a memory for the history of the system, where longer voltage pulses were applied onto the A electrode (Figure 6.3). The competition between synapse A-S and B-S was achieved in this Hebbian learning experiment setup. As synapse A-S received stronger stimulation, it finally won the competition.

In biology, there are several underlying mechanisms that work together to achieve synaptic plasticity, including changes in the quantity of neurotransmitters released into the synaptic cleft, which usually leads to short term synaptic plasticity, and changes in how effectively neuron cells respond to these neurotransmitters through the alteration of the number or conductance of neurotransmitter receptors, which leads to long term synaptic plasticity. Yet

another way to achieve long lasting changes in synaptic strength is through protein synthesis, which is to enlarge or shrink dendritic spines in terms of physical size.

The availability of new protein synthesis can lead to cooperative and competitive interactions between synapses. A potential finite resource for synapses to compete for is called calmodulin [1], as shown in Figure 6.4, which can cooperate with calcium to trigger either protein synthesis or increase in AMPA receptor sensitivity, both leading to long term potentiation (LTP).

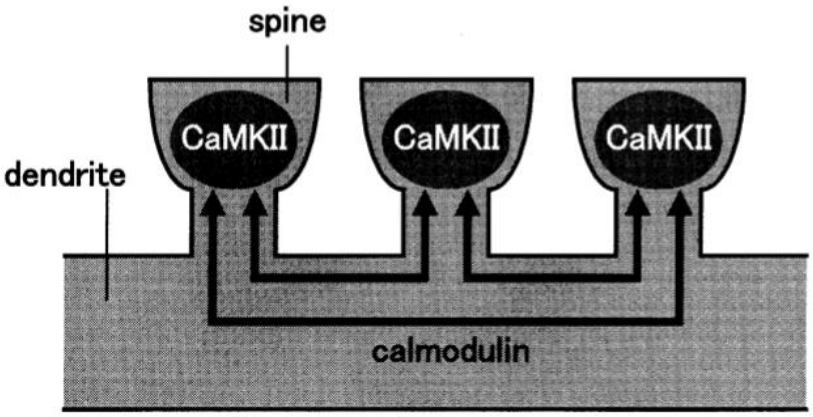


Figure 6.4 Schematic of the finite resource model. Neighboring synapses formed on the same dendrite are considered and diffusional transportation of free calmodulin among the spines via the dendrite is assumed. From [1].

Synapse clustering, which is the physical remodeling of synapses into functional groups, can increase the computational power of neurons by allowing information to be combined in a greater-than-additive manner [3]. As shown in Figure 6.5, a dendritic region receives a pattern of activity. Some spines (spine a) receive the strongest LTP inputs, while others (b, c, d, e) receive weaker inputs. Signaling mechanism including protein synthesis will be first triggered around

spine a. Spines in the surrounding regions (b, c, and d) compete for the limited resources - calmodulin. Spine e is too far away so will not enter the competition. Winning spines (a, b, c) are selectively strengthened while the losing spine d does not express long lasting plasticity. In the long term, new spine grows around the winning spines (a, b, c), while the previously existing spines (such as d) are removed. In the end, a structural cluster is formed reflecting the activity bump that forms in the dendrite. Synapse clustering is also a type of synaptic competition effects that involves many synapses on the same dendrite.

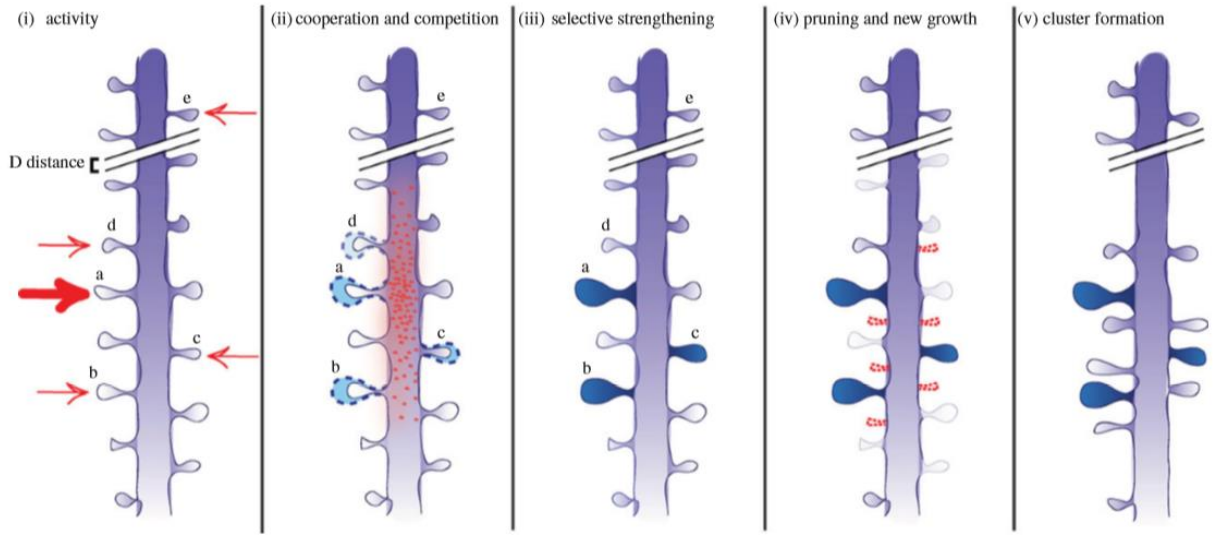


Figure 6.5 Competition and cooperation in structural remodeling. From [3].

Therefore, it will be interesting to emulate this competition effect between synapses using our solid-state nanoscale memristive devices as well. The competition phenomenon is important in the neuromorphic computing field to serve as a natural means to realize weight normalization [4] in many machine learning algorithms during weight updates. For example, in Oja's rule the

weights are normalized by default based on the update rule, but other learning rules such as Hebbian learning will make the weights change in one direction (always increasing), which can lead to severe problems in the learning process.

6.2.1 Device Structure

We designed a planar 1×n device structure to demonstrate the synaptic competition effect. As shown in Figure 6.6, three synapses were formulated between four inert electrodes, representing three pre-synaptic neurons and one post-synaptic neuron. The inert electrodes were patterned by E-beam lithography, and are composed of 5 nm of NiCr and 30 nm of Pd. The films were deposited by E-beam evaporation. The substrate is 100 nm SiO_2 on Si.

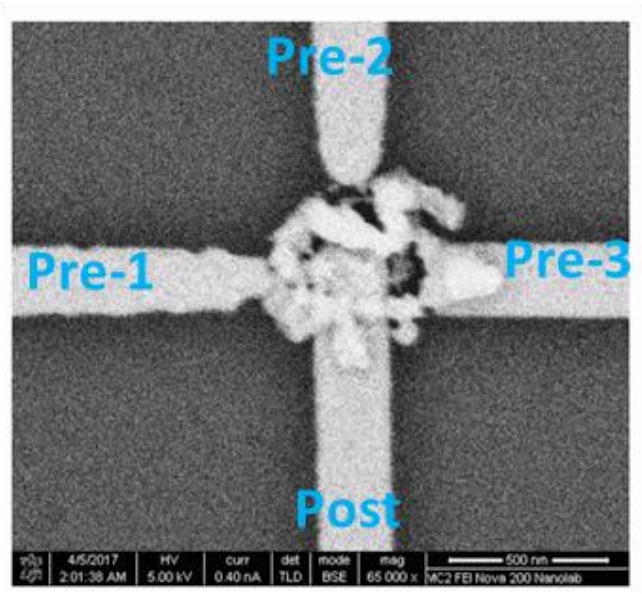


Figure 6.6 Planar 1×3 device structure designed for synaptic competition.

To provide the limited resource for synapses to compete for, a small patch of silver sulfide was deposited onto the post-synaptic dendrite region, i.e. one of the inert electrodes, using deposited silver and solution-based sulfurization method similar to what is described in Chapter 5. The silver patch before the sulfurization was also patterned by E-beam lithography, deposited by E-beam evaporation, and is only 40 nm thick. In order to achieve the best competing result, the shape and size of the silver patch needs to be further optimized.

The goal is to attract the silver sulfide (silver) into one of three gaps where the strongest electric field is applied to, while the other two gaps where there is no signal applied will be depleted of silver and these two connections will be at the high resistance states (non-conducting). The winning connection will be at the low resistance state (conducting). There would never be a homosynaptic reset voltage needed to annihilate the existing conducting filament but a heterosynaptic set voltage can automatically lead to the break of the existing filament and the formation of a new one where the stimulation is delivered at.

This hypothesis agrees well with the neuroscience phenomenon that the synapse that receives the strongest LTP input will be strengthened while the others either receiving weaker inputs or no input at all will decay and eventually be eliminated. Therefore, a rough conservation of total synaptic weight through balanced synaptic depression and potentiation can be achieved to maintain the system's stability [4], which has potential applications for neural network computing based tasks where weight renormalization is required.

6.2.2 Electrical Measurement

In order to monitor the conductance change at the other two synaptic junctions while stimulation is being delivered to one of the three connections, a very low voltage (-0.05 V) is applied to the non-stimulated pre-synaptic neuron electrodes just to read out the conductance state. It has been observed that a very large voltage bias (~10 V) is needed to form the initial filament. After one of the synapse has been formed, then usually a low voltage bias (<5 V) is sufficient to switch the other synapses. For the subsequent cycles, the switching voltages will be even lower, at < 1 V.

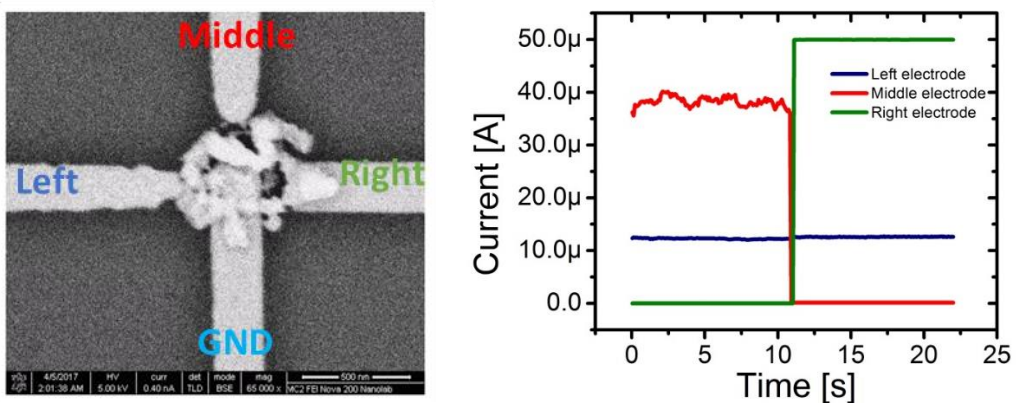


Figure 6.7 Forming the right synapse junction (green) leads to the break of the middle synapse junction (red).

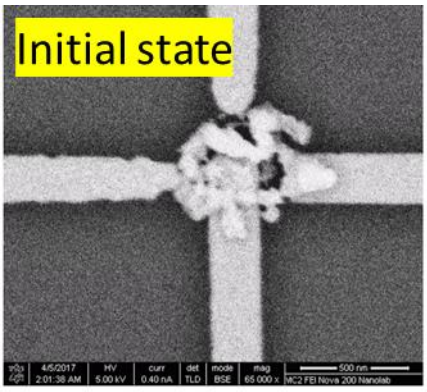
Interestingly, we observed that by setting the right synapse (green curve), at the moment the current abruptly increases in this synapse, the already conducting middle synapse (red curve) was disrupted and its current abruptly decreased to near zero (Figure 6.7). While the relatively far away left synapse (blue curve) was not affected. It is reasonable to believe the growth on the

right synapse leads to the annihilation of the middle synapse by taking up all the silver resources from this electrical measurement, as expected from the synaptic competition model.

6.2.3 Observation in Real Time SEM

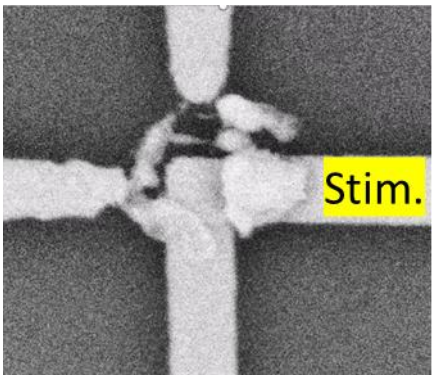
As the switching induced by silver can be transient, it is necessary to observe the movement of the silver sulfide in real time during programming. The measurement was done at the MC^2 center at the University of Michigan using the Nova SEM. The initial state has the silver sulfide distributed uniformly in the gap area leading to three non-conducting synapses in the beginning (Figure 6.8 (a)). Then the voltage bias was applied to the left electrode. From the morphology, it can be seen that the left synapse is formed by attracting the silver/silver sulfide to the left cleft (Figure 6.8 (b)). In the next step, the voltage bias was applied to the right electrode, and the right synapse is formed by attracting all the silver/silver sulfide to the right cleft, leaving the left one erased (Figure 6.8 (c)). Afterwards, the voltage bias was applied to the middle electrode, and the middle synapse is formed by attracting all the silver/silver sulfide to the middle cleft, leaving the left one and right one erased (Figure 6.8 (d)). Finally, the voltage bias was applied to the left electrode again, and the left synapse is formed again, leaving the middle one and right one erased (Figure 6.8 (e)). This process can be repeated for several cycles.

(a)

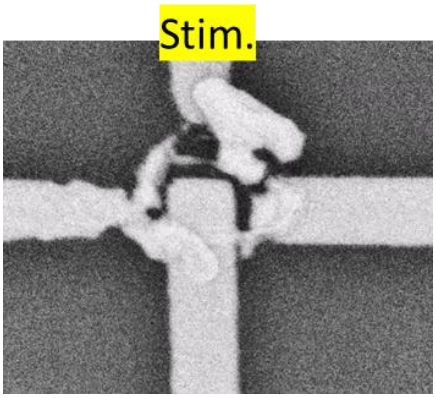


(b)

(c)



(d)



(e)

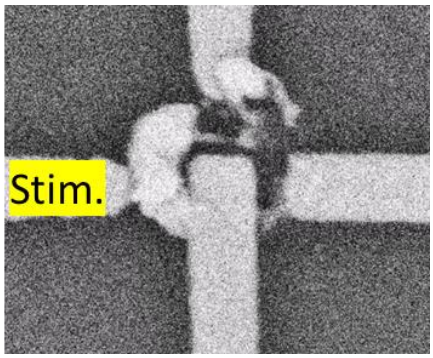


Figure 6.8 Real time SEM video snapshots showing the synaptic competition. (a) Initial state. (b) Voltage bias applied to the left electrode, left synapse won. (c) Voltage bias applied to the right electrode, right synapse won. (d) Voltage bias applied to the middle electrode, middle synapse won. (e) Voltage bias applied to the left electrode, left synapse won again.

Future work will focus on a larger network such as $1\times n$ or $m\times n$ memristor array network, and probably perform quantitate data analysis showing the conservation of the total synaptic weight after the different stimulation sequences, to demonstrate the feasibility of the network real applications utilizing a synaptic learning rule that requires weight renormalization process.

6.3 Grid Cell & Place Cell Firing Pattern Generation

Grid cells, found in the dorsal medial entorhinal cortex (dMEC) area near the hippocampus, have generated broad interests in studies on animal spatial navigation since its

discovery in 2005 [6]. Before that, the firing of the place cells, which is critical for direction and distance memory encoding [7], is only believed to be dependent on sensory inputs such as boundary vector cells, which are abundant in the subiculum [8]. Since the entorhinal cortex is the source of primary input to the hippocampus, it is later thought that the grid cells play a crucial role in the development of place cells [9, 10].

Unlike place cells that fire when the animal reaches a certain place in the local environment [7], grid cells get activated at locations correspond to the vertices of a hexagonal lattice that infinitely spreads across the environment [6]. The hexagonal firing patterns of grid cells can vary in phases and frequencies depending on the location of those in the brain. Several models have proposed that a combination of grid cell firing of different spatial scales can lead to single-peaked place fields [9, 10].

The most widely believed inference model of grid cell formation implied that the 60^{o} angular separations of the oscillations are naturally selected to generate hexagonal grid cell firing patterns through self-organization [13]. Stripe cells are the kind of path integration cells that fire periodically when an animal moves in the cell's favored direction. From geometry, we know that the most frequent coactivations will occur for angle differences between stripe cell orientations that are 60^{o} apart, which explains why hexagonal patterns are favored in biology. It is believed that grid cells form their 2d periodic firing through learning based on input from stripe cells.

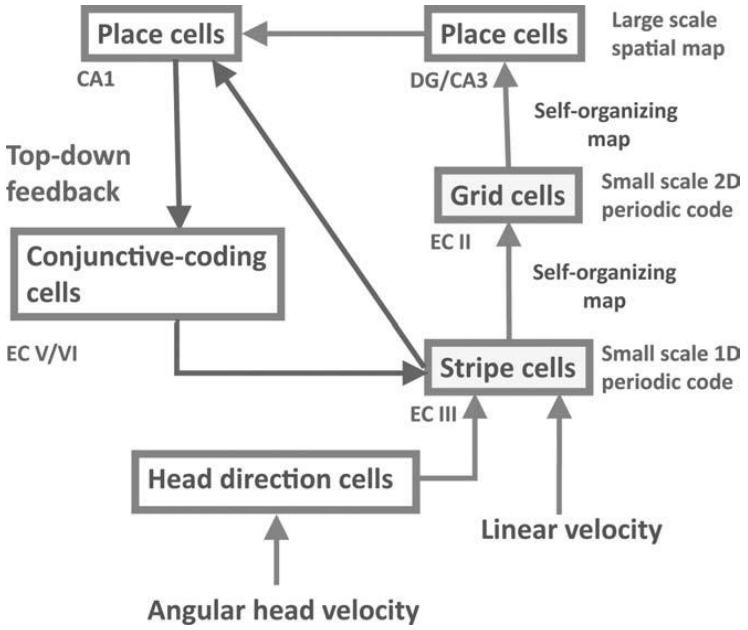


Figure 6.9 A system diagram of interactions within the entorhinal-hippocampal navigational system. From [8]. The model clarifies how stripe cells may give rise to grid cells via a self-organizing map. Earlier work has demonstrated how grid cells may give rise to place cells via a self-organizing map. Together these results suggest that a hierarchy of self-organizing maps may learn key properties of the brain navigational circuits that are based on path integration signals.

Taken together, three layers of self-organizing maps (Figure 6.9) are needed for generating desired firing patterns of these path integration cells. The memristor crossbar neural network is a very suitable candidate for self-organizing learning, or adaptive learning through gradually changing the resistance at each cross point in the network. Thus, it would be intuitive to propose a three-layer memristor-based crossbar network for path integration cell firing pattern generation.

6.3.1 Place Cell Firing Model

Grid cells fire when the animal is located at any of the vertices of a grid of equilateral triangles covering the whole environment, and different cells have different firing frequencies

and phase offsets. While place cell, whose firing is very localized, is active only over a single portion of the space. The connectivity from grid cells to place cells can function as a competitive neural network, the learning in which is critical for self-organizing mapping.

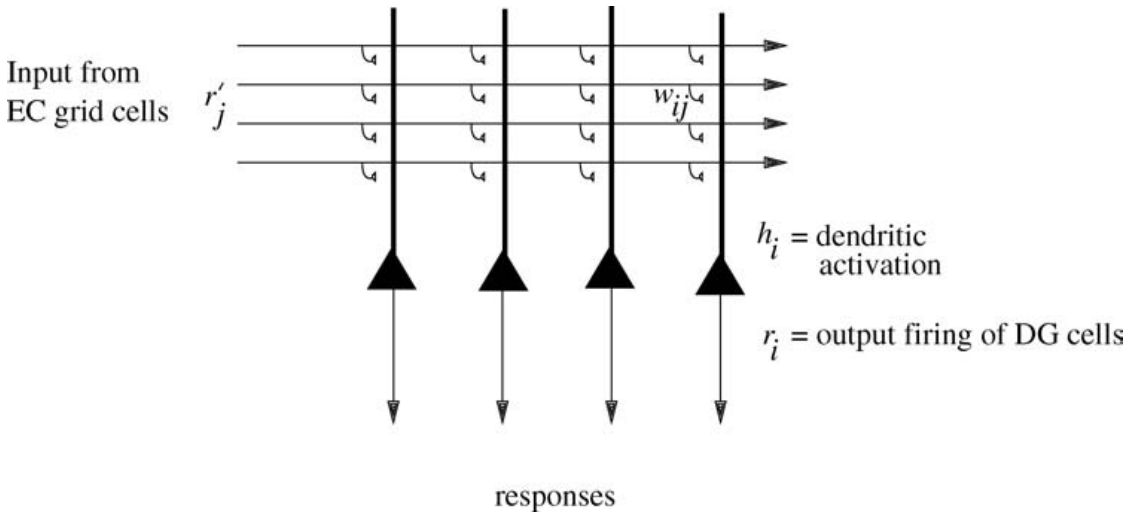


Figure 6.10 2-layer competitive neural network architecture. From [10]. There is an input layer of dMEC grid cells with feedforward modifiable synaptic connections onto an output layer of DG place cells. At each location, activity from the grid cells is propagated through the feedforward connections to activate a winning set of place cells. After training, place cells become associated with patterns of activity in the grid cell layer, which represent particular locations in the environment.

In a proposed fully connected 2-layer neural network, shown in Figure 6.10 [10], the synaptic weights are modifiable. The grid cells have different frequencies realistic in neurophysiological studies and different phase offsets. Space is continuous. During training, the object is moved through a sequence of consecutive locations through the environment. During moving, grid cells fire according to their pre-defined activity profiles. The activity is propagated through the feedforward synaptic connections to activate the place cells. Initial weights are random. A fixed sparseness is used to achieve lateral inhibition between the output neurons. The synaptic weights are strengthened according to the associative Hebbian learning rule and weight

renormalization is performed at each location to prevent the same few neurons from winning all the time.

As the competitive network self-organizes, the weight vector of each output neuron moves towards the center of a set of patterns in the input space. With no training, the place cells are generally spatially repeating, with only a few single-peaked ones. After learning with the Hebbian rule, there is less spatial repetition and some place cells have developed narrow single-peaked place fields. Figure 6.11 shows the 2d place cell firing rate profiles from the simulation after training.

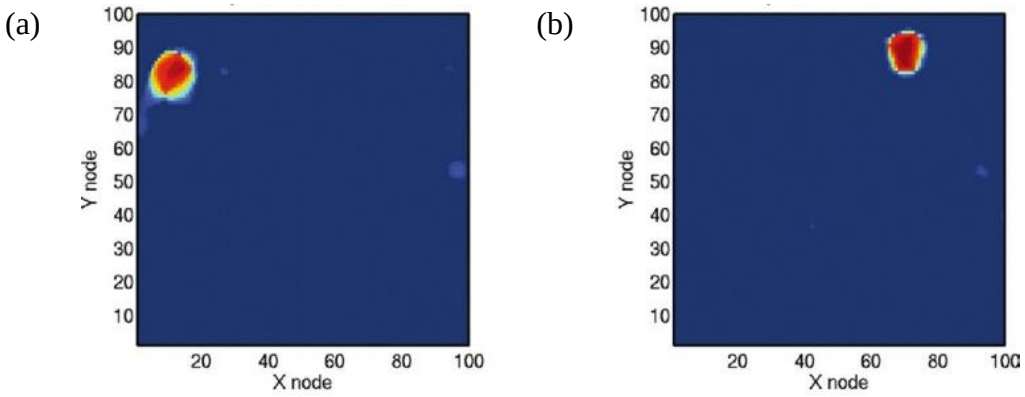


Figure 6.11 Firing rate profiles of two place cells after training with the Hebb rule. From [10].

6.3.2 Grid Cell Firing Model

As previously stated, because of the most frequent appearance of coactivations, cells with orientations that are 60º apart are emphasized in the competing self-organizing map, and hexagonal positional representations characteristics of grid cells are created [13], as show in Figure 6.12. In the simulation, all stripe cells initially projected nonspecifically with random

weights to the grid cells. The stripe cells are defined with 1d periodic firing patterns following sinusoidal functions.

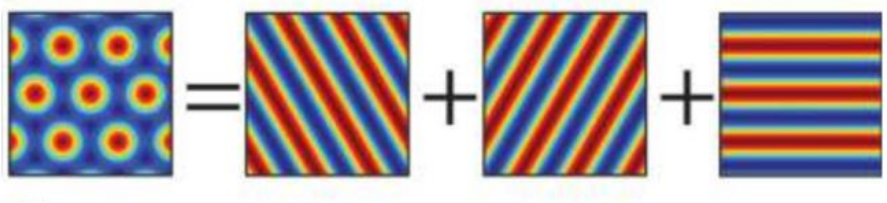


Figure 6.12 Grid cell firing pattern is constructed from a sum of three sinusoidal functions with 60 and 120 degrees angular difference from three stripe cells. From [9].

The simulation was run for 100 trials with each trial of the model rat running along a trajectory constructed from experimentally recorded 10 min run. All stripe cells have the same spatial scale (spatial periodicity in the sinusoidal firing pattern) but 4 different spatial phases and 9 different angles, which lead to 36 stripe cells in total. The angular separation between stripe cells' preferred directions is 20º. In each phase, angles go from -80º to 80º with step 20 from left to right. The adaptive synaptic weights in the 36×5 (36 stripe cells and 5 grid cells) neural network evolve according to the instar learning rule in conjunction with winner-take-all. From Figure 6.13, hexagonal firing pattern began to emerge after the 3rd trial (100 trials in total) and in the end, 3 preferred directions were formed in the grid cell firing patterns. If the number of grid cells in the model were large enough (now there are only 5), all combinations of phases and orientations can be seen in grid cell firing patterns by naturally potentiating 3 synapses that are connected to stripe cells with preferred directions differed by 60º.

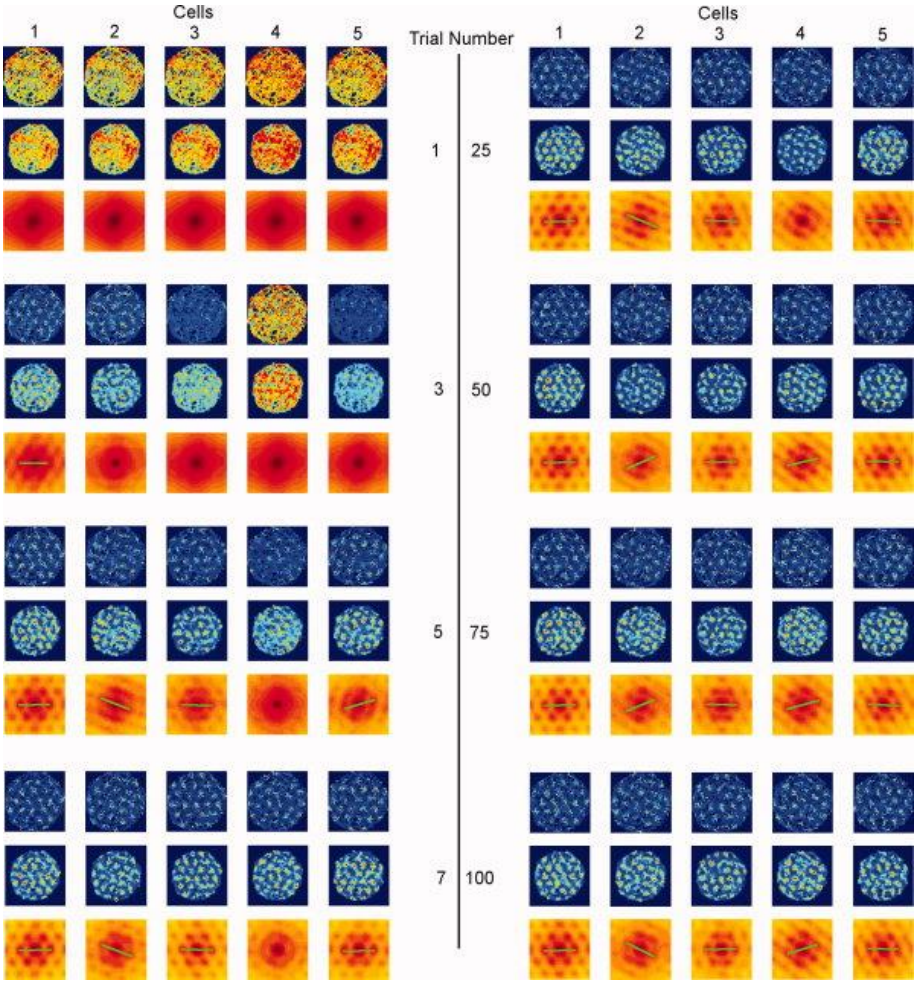


Figure 6.13 Temporal dynamics of grid cell firing pattern development. Firing fields and autocorrelation maps of all five cells after Trials 1, 3, 5, 7, 25, 50, 75, and 100 are shown with each column in the figure corresponding to a different cell. In each trial, the top row shows the normalized average firing rate map of the cell. The center row shows the normalized rate map smoothed using a Gaussian convolution filter. The bottom row shows the autocorrelation maps of the smoothed rate map. The angular separation between stripe cells' preferred directions was fixed at 20°. From [13].

6.3.3 Mimicking Grid-cell-to-place-cell Firing using Memristor Crossbar Array

We aim to emulate the grid-cell to place-cell firing in a memristor network. First, we use a 32×32 WO_x memristor network to project the one-dimensional grid cell firing pattern (as inputs) onto place cells (as outputs of the network). Specifically, the object moves along a straight line from location 1 to location 32. The input grid cells fire at 8 different frequencies and

4 different phases. The activity patterns can be described by the following equation using a cosine function:

$$g(x) = 0.5 * (\cos(2\pi\alpha x + \theta) + 1) \quad (6.1)$$

Where α = 3, 4, 5 … 10 controls the frequency and θ = 0, π/2, π, 3π/2 controls the initial phase. Initial weights were random and ranged between 0 and 0.1. Winner-take-all was used in the training stage and the neuron with the highest membrane potential, integrated from all past activity, will fire and then its membrane potential will be set to a low value afterwards. WO_x memristor model was used in this simulation without considering the decay effect. The goal is to have each of the 32 place cells to fire when the object is at one and only one of the 32 locations, so that one place cell corresponds to one location.

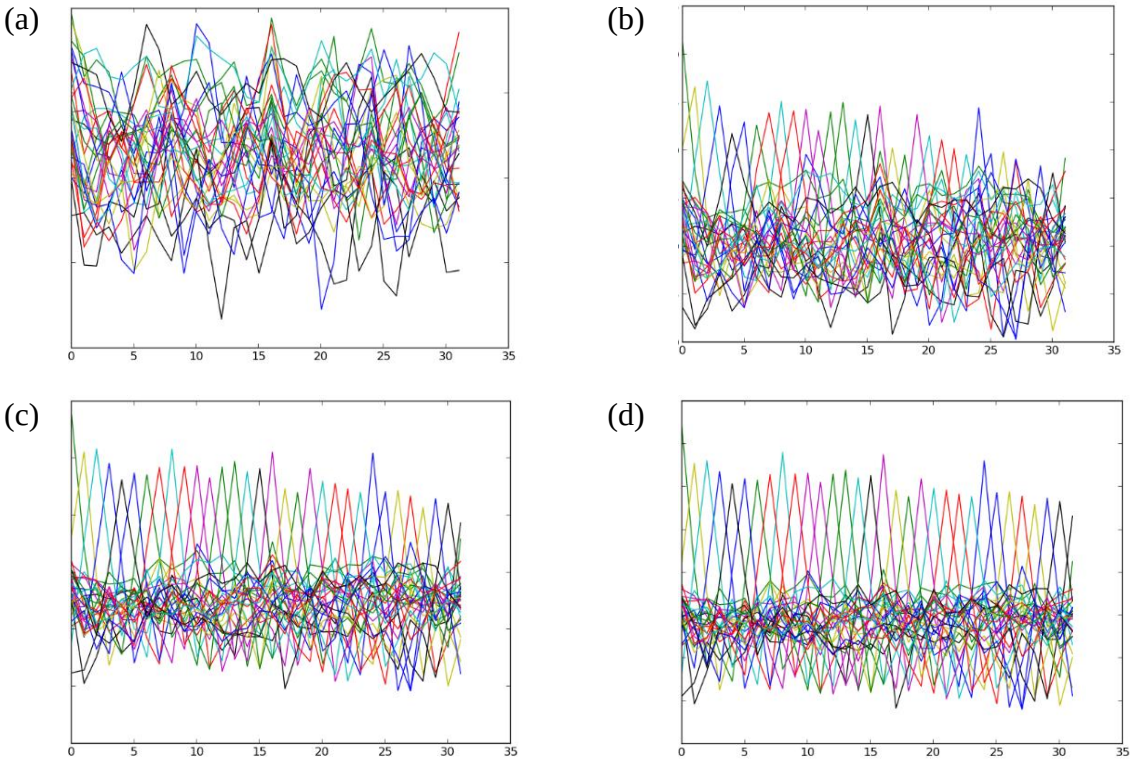
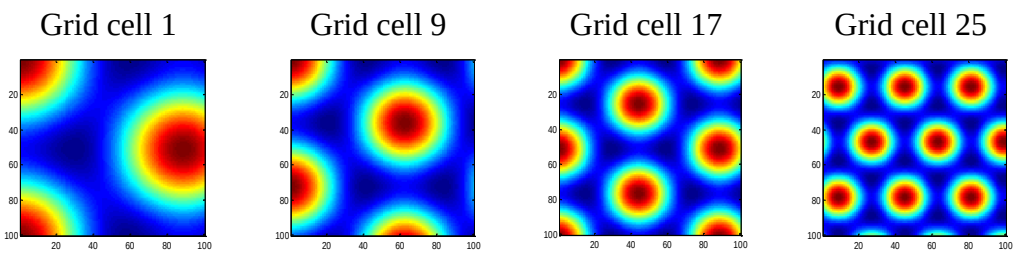


Figure 6.14 Firing patterns of the 32 place cells (a) before training, and after (b) 10 (c) 20 (d) 30 training cycles.

Without training, the firing patterns of the 32 place cells are random without prominent peaks. After 10 training cycles, some single peaks began to appear. After 20 training cycles, the peaks in the place fields are stabilized and each location has a clear corresponding place cell that only fires when the object reaches this location (Figure 6.14).

We further expanded the simulation to a two-dimensional environment with the 32×32 memristor network. In the two-dimensional case, the object is no longer moving through a straight line but wanders on random trajectories in a square box. Weights are updated using Oja's rule to ensure weight renormalization. 4 frequencies and 8 phases were used, making a total of 32 grid cells. Some examples of the grid cell firing patterns (8 out of 32) are shown in Figure 6.15. To make the problem easier, the learning was made supervised by having five food sources at five different locations so that the place cell can only fire and the weights can only be updated when the object is near one of these five spots (Figure 6.16). Mathematically, this is achieved by changing the usual output activity y in Oja's rule into y multiplied by a firing rate function, which is shown in Figure 6.16. However, we do not specify which of the 32 place cells will fire but let the cell with the highest activity update its corresponding weights (winner-take-all).



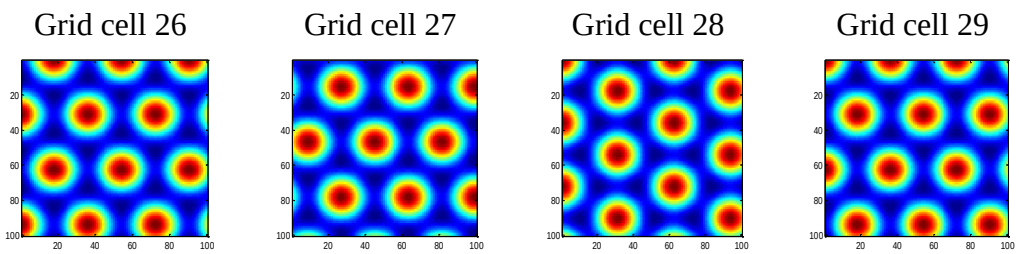


Figure 6.15 Examples of grid cell firing patterns used as inputs.

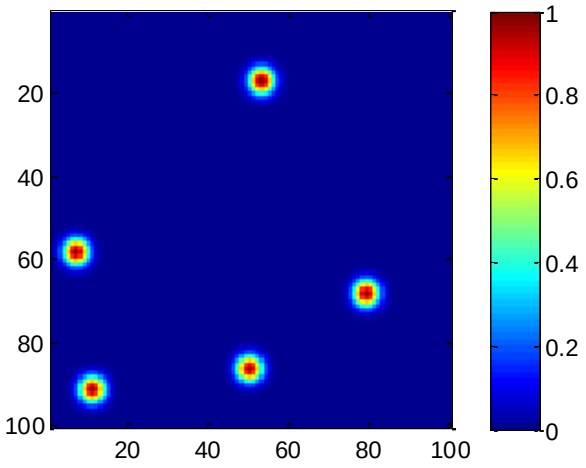


Figure 6.16 Additional firing rate function added to the place cells output activity during learning. The red spots indicate positions of the five food sources.

After training, most of the 32 place cells do not have firing fields. Only five neurons, neuron number 2, 4, 9, 21, and 25 have formed single peaked firing fields within the 2d square box, shown in Figure 6.17, corresponding to the locations of the 5 food sources.

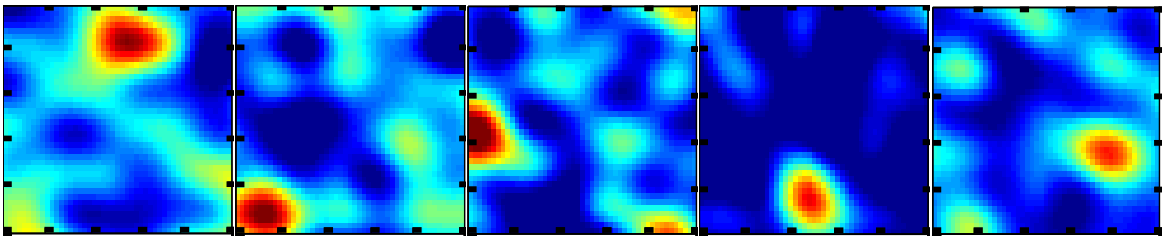


Figure 6.17 Place cell firing results after training. Five cells developed single peaked place fields that correspond to the location of the food sources.

The next step is to first remove the food sources as environment cues to train the network in an unsupervised way, so that the place fields can emerge on their own. Furthermore, another layer of the memristor network can be used to generate the grid cell firing patterns using the stripe cells as inputs. Altogether, two layers of memristor crossbar arrays will be used, one for stripe-cell-to-grid-cell firing pattern generation, and the other for grid-cell-to-place-cell firing pattern generation, in correspondence to the two-layer self-organizing maps proposed in Figure 6.9. Similar neuromorphic systems have been built in VLSI, focusing on the spatial navigational cells [14, 15], so it would be interesting to build such a system with memristive devices/arrays and analyze its performance and benchmark with other approaches.

References

[1] Okamoto, H. & Ichikawa, K. A model for molecular mechanisms of synaptic competition for a finite resource. BioSystems 55, 65–71 (2000).

[2] Sperl, M., Chang, A., Weber, N. & Hübler, A. Hebbian learning in the agglomeration of conducting particles. Phys. Rev. E 59, 3165–3168 (1999).

[3] Ramiro-Cortés, Y., Hobbiss, A. F. & Israely, I. Synaptic competition in structural plasticity and cognitive function. Philos. Trans. R. Soc. Lond. B. Biol. Sci. 369, 20130157 (2014).

[4] Royer, S. & Paré, D. Conservation of total synaptic weight through balanced synaptic depression and potentiation. Nature 422, 518–522 (2003).

[5] Attneave, F., B., M. & Hebb, D. O. The Organization of Behavior; A Neuropsychological Theory. Am. J. Psychol. 63, 633 (1950).

[6] Hafting, T., Fyhn, M., Molden, S., Moser, M.-B. & Moser, E. I. Microstructure of a spatial map in the entorhinal cortex. Nature 436, 801–806 (2005).

[7] O'Keefe, J. & Dostrovsky, J. The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat. Brain Res. 34, 171–175 (1971).

[8] Hartley, T., Burgess, N., Lever, C., Cacucci, F. & O'Keefe, J. Modeling place fields in terms of the cortical inputs to the hippocampus. Hippocampus 10, 369–379 (2000).

[9] Solstad, T., Moser, E. I. & Einevoll, G. T. From grid cells to place cells: A mathematical model. Hippocampus 16, 1026–1031 (2006).

[10] Rolls, E. T., Stringer, S. M. & Elliot, T. Entorhinal cortex grid cells can map to hippocampal place cells by competitive learning. Netw. Comput. Neural Syst. 17, 447–465 (2006).

[11] Fuhs, M. C. & Touretzky, D. S. A Spin Glass Model of Path Integration in Rat Medial Entorhinal Cortex. J. Neurosci. 26, 4266–4276 (2006).

[12] Burgess, N., Barry, C. & O'Keefe, J. An oscillatory interference model of grid cell firing. Hippocampus 17, 801–812 (2007).

[13] Mhatre, H., Gorchetchnikov, A. & Grossberg, S. Grid cell hexagonal patterns formed by fast self-organized learning within entorhinal cortex. Hippocampus 22, 320–334 (2012).

[14] Massoud, T. M. & Horiuchi, T. K. A Neuromorphic VLSI Head Direction Cell System. IEEE Trans. Circuits Syst. I Regul. Pap. 58, 150–163 (2011).

[15] Aggarwal, A. Neuromorphic VLSI realization of the hippocampal formation. Neural Networks 77, 29–40 (2016).

Chapter 7 Summary

This dissertation started with discussing the basic concepts of memristors and the motivation of using memristors in neuromorphic computing systems. Two kinds of nanoscale solid state memristors based on two different types of resistive switching mechanisms were introduced, based on WO_x and Ag_2S as the switching material, respectively. Both types of memristors have been fabricated and characterized, and show interesting incremental conductance change and short-term decay dynamics that is absent in traditional non-volatile RRAM devices. The concept and modeling of second-order memristor were discussed. The short-term and long-term dynamics of memristors offer great advantage of combining computing power and memory storage within one single device. In Chapter 3, different synaptic behaviors and learning functions (frequency-dependent and spike-timing-dependent synaptic plasticity) were emulated by WO_x memristors with short-term dynamics, showing the resemblance between dynamic memristors and biological synapses. These demonstrations suggest memristors can faithfully emulate synaptic dynamics and enable more complicated neuromorphic computing structure.

The first application of the WO_x memristors is image reconstruction and dictionary learning (Chapter 2). The analog programming capability of dynamic memristors is very important for learning/training in artificial neural networks. A 32×32 WO_x memristor crossbar array was used for vector-matrix multiplication acceleration in one of the machine learning

applications - image analysis. The device non-ideality effects in the memristor crossbar array on the reconstruction performance were examined as well. Interestingly, it was found that moderate device to device variations in the array can be beneficial to the reconstruction result as the variations capture the high frequency spatial information that is missing in the ideal dictionary. While stuck-at-1 device failures can be detrimental to the reconstruction, stuck-at-0 failures seem to have little impact. A solution to overcome the stuck-at-1 device failures was proposed and tested valid through simulation.

WO_x memristor show dynamic temporal responses when subjected to different configurations of spiking pulse trains. This property enabled the WO_x memristors for another machine learning application – speech recognition (Chapter 4), achieved using memristor-based reservoir computing. The recognition accuracy can be improved in the future by optimizing the reservoir size, and more benchmarking work can be done on this subject to compare to the state-of-the-art. Another task of autonomous signal generation was also performed to evaluate the effectiveness of the reservoir. Similarly, the short-term decay of the Ag_2S memristors can be used to encode neural spiking information in the temporal domain with analog switching probability distribution (Chapter 5).

The last chapter introduced two other interesting projects that only have preliminary results so far: 1) Emulating the synaptic competition effect using a planar $1\times n$ Ag_2S memristive network; and 2) Training the WO_x memristor crossbar array for grid cell – place cell firing pattern generation, potentially useful for spatial navigation purposes. These projects deserve further exploration both in theoretical analysis/simulation and experiment.

In all, dynamic memristors based on either Ag_2S or WO_x as the switching material showed very promising properties for neuromorphic computing purposes – overcoming the von-

Neumann bottleneck by incorporating information processing into memory storage. It is believed in the future, very efficient neuromorphic computing chips can be designed and implemented using these memristors that offer potential advantages in terms of area consumption, computing speed and power consumption, compared with existing approaches based on the von Neumann architecture.