ABSTRACT

HIGH-PERFORMANCE LEARNING SYSTEMS USING LOW-PRECISION NANOSCALE DEVICES

by
Nandakumar Sasidharan Rajalekshmi

Brain-inspired computation promises a paradigm shift in information processing, both in terms of its parallel processing architecture and the ability to learn to tackle problems deemed unsolvable by traditional algorithmic approaches. The computational capability of the human brain is believed to stem from an interconnected network of 100 billion compute nodes (neurons) that interact with each other through approximately 10^{15} adjustable memory junctions (synapses). The conductance of synapses is modifiable allowing the network to learn and perform various cognitive functions. Artificial neural networks inspired by this architecture have demonstrated even super-human performance in many complex tasks.

Computational systems based on the von Neumann architecture, however, are ill-suited to optimize and operate these large networks, as they have to constantly move data between the physically separated processor and memory units. Crossbar arrays of nanoscale analog memory devices could store large network weight matrices in their respective conductances and could perform matrix operations without moving the weights to a processor. While this 'in-memory computation' provides an efficient and scalable architecture, the trainability of the memory devices is constrained by their limited precision, stochasticity, and non-linearity, and therefore poses a major challenge.

In this dissertation, a mixed-precision architecture is proposed which uses a high-precision digital memory to compensate for the limited precision of the synaptic devices during the training of deep neural networks. In the architecture, the desired weight updates are accumulated in high-precision and transferred to the synaptic devices when the accumulated update exceeds a threshold representing the average device update granularity. Deep neural networks based on experimental nanoscale devices are shown to achieve performance comparable to high-precision software simulations by this approach.

Phase-change memory devices (PCM) on a prototype chip from IBM is used to experimentally demonstrate the proposed architecture. Artificial neural networks whose synapses are realized using PCM devices are trained to classify handwritten images from the MNIST dataset and the mixed-precision approach is successful in achieving training accuracies comparable to floating-point simulations. On-chip inference experiment using the PCM devices shows that the network states are retained reliably for more than 10^6 s. The architecture is estimated to achieve approximately 20× acceleration in training these networks compared to high-precision implementations and has a potential for at least 100× efficiency gain in inference.

Supervised training and inference of third generation spiking neural networks using PCM are also demonstrated using the hardware platform. New array level conductance scaling methods are demonstrated for adaptive mapping of the device conductance to network weights and to compensate for the effect of conductance drift. During the course of the study, $Ge_2Sb_2Te_5$ based PCM and $Cu/SiO_2/W$ based resistive random access memories are characterized for their gradual conductance modulation behavior and statistically accurate models are created. The models are used to pre-validate the experiments and to test the efficacy of different synapse configurations in the training of neural networks.

Collectively, this work demonstrates the feasibility of realizing high-performance learning systems that use low-precision nanoscale memory devices, with accuracies comparable to those obtained from high-precision software training. Such learning systems could have widespread applications including for energy and memory constrained edge computing and internet of things.

HIGH-PERFORMANCE LEARNING SYSTEMS USING LOW-PRECISION NANOSCALE DEVICES

by
Nandakumar Sasidharan Rajalekshmi

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering

Helen and John C. Hartmann Department of Electrical and
Computer Engineering

May 2019

APPROVAL PAGE

HIGH-PERFORMANCE LEARNING SYSTEMS USING LOW-PRECISION NANOSCALE DEVICES

Nandakumar Sasidharan Rajalekshmi

BIOGRAPHICAL SKETCH

Author:	Nandakumar Sasidharan Rajalekshmi
Degree:	Doctor of Philosophy
Date:	May 2019

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2019
- Master of Technology in Microelectronics,
  Indian Institute of Technology Bombay, Mumbai, India, 2016
- Bachelor of Technology in Electrical and Communication Engineering,
  College of Engineering, Thiruvananthapuram, Kerala, India, 2013

Major: Electrical Engineering

Presentations and Publications:

S. Nandakumar, M. Le Gallo, I. Boybat, B. Rajendran, A. Sebastian, and E. Eleftheriou, "A phase-change memory model for neuromorphic computing," *Journal of Applied Physics*, vol. 124, no. 15, p. 152135, October 2018.

S. Nandakumar, S. R. Kulkarni, A. V. Babu, and B. Rajendran, "Building brain-inspired computing systems: examining the role of nanoscale devices," *IEEE Nanotechnology Magazine*, vol. 12, no. 3, pp. 19–35, September 2018.

I. Boybat, M. Le Gallo, S. Nandakumar, T. Moraitis, T. Parnell, T. Tuma, B. Rajendran, Y. Leblebici, A. Sebastian, E. Eleftheriou, "Neuromorphic computing with multi-memristive synapses," *Nature Communications*, vol. 9, no. 1, p. 2514, June 2018.

S. Nandakumar, M. Minvielle, S. Nagar, C. Dubourdieu, and B. Rajendran, "A 250 mV Cu/SiO_2/W memristor with half-integer quantum conductance states," *Nano Letters*, vol. 16, no. 3, pp. 1602–1608, March 2016.

S. Nandakumar, M. Le Gallo, I. Boybat, B. Rajendran, A. Sebastian, and E. Eleftheriou, "Mixed-precision architecture based on computational memory for training deep neural networks," *In Proceedings of the IEEE International Symposium on Circuits and Syastems (ISCAS)*, 2018, pp. 1–5.

I. Boybat, S. Nandakumar, M. Le Gallo, B. Rajendran, Y. Leblebici, A. Sebastian, E. Eleftheriou, "Impact of conductance drift on multi-PCM synaptic architectures," *In Proceedings of the 18th Non-Volatile Memory Technology Symposium (NVMTS)*, 2018, pp. 1-4.

I. Boybat, M. Le Gallo, S. Nandakumar, B. Rajendran, Y. Leblebici, A. Sebastian, E. Eleftheriou, "Multi-PCM synapses for spiking neural networks," *European Phase-Change and Ovonic Symposium*, 2018

S. Nandakumar, I. Boybat, M. Le Gallo, A. Sebastian, B. Rajendran, and E. Eleftheriou, "Supervised learning in spiking neural networks with MLC PCM synapses," *In Proceedings of the 75th Annual Device Research Conference (DRC)*, 2017, vol. 22, no. 2, pp. 1–2.

I. Boybat, M. Le Gallo, S. Nandakumar, T. Moraitis, T. Tuma, B. Rajendran, A. Sebastian, E. Eleftheriou,"An efficient synaptic architecture for artificial neural networks," *In Proceedings of the 17th Non-Volatile Memory Technology Symposium (NVMTS)*, 2017, pp. 1–4.

S. Nandakumar and B. Rajendran, "Synaptic plasticity in a memristive device below 500mV," *ECS Transactions*, vol. 77, no. 2, pp. 31–37, Apr. 2017.

S. Nandakumar and B. Rajendran, "Verilog-A compact model for a novel Cu/SiO_2/W quantum memristor," *In Proceedings of the International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, 2016, no. 973, pp. 169–172.

S. Nandakumar and B. Rajendran, "Physics-based switching model for Cu/SiO_2/W quantum memristor," *In Proceedings of the 74th Annual Device Research Conference (DRC)*, 2016, vol. 16, no. 973, pp. 1–2.

C. Shetty, Sri Nitchith, Rishabh Rawat, S. Nandakumar, Pritesh Shah, Shruti Kulkarni, and Bipin Rajendran,"Live demonstration: Spiking neural circuit based navigation inspired by C. elegans thermotaxis," *In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015, pp. 1905–1905.

For all those who let me be. . .

ACKNOWLEDGMENT

I express my sincere gratitude to my advisor Prof. Bipin Rajendran who has been a constant support and spent countless hours to improve various aspects of my Ph.D. research life. I would like to thank Dr. Abu Sebastian and Dr. Evangelos Eleftheriou for providing the many research opportunities, support, and guidance at IBM Research Zurich. I sincerely thank my dissertation committee members, Prof. Durgamadhab Misra, Prof. Osvaldo Simeone, and Prof. Hieu Pham Trung Nguyen for providing guidance, recommendations, valuable suggestions, and comments during my Ph.D. research.

I would like to express my thanks to Dr. Manuel Le Gallo, Irem Boybat, Christophe Piveteau, Vinay Joshy, Shruti Kulkarni, Anakha V. Babu for various stimulating discussions and contributions to my research.

I would like to thank IBM for the Ph.D. fellowship. I sincerely thank the NJIT ECE department for the constant support and for providing teaching assistantship opportunities.

Last, I would like to express my deepest appreciation to my parents, Sasidharan Nair and Rajalekshmi G. Their continued patience and endless support have been a key element in all my endeavors.

TABLE OF CONTENTS

Chapter **Page**

1 INTRODUCTION 1

1.1 Computational Engine in the Human Brain 3

1.2 Computational Architectures 4

1.2.1 von Neumann architecture 5

1.2.2 Graphics processing units 6

1.2.3 Non-volatile memory arrays: computational memory 7

1.2.4 Potential of analog memory based neural networks 8

1.3 Further in this Dissertation 9

2 NANOSCALE MEMORY DEVICES AND LEARNING 11

2.1 Introduction 11

2.2 Synaptic Devices 12

2.2.1 Synapse 12

2.2.2 Non-volatile analog memories 14

2.3 Ideal Synaptic Characteristics 20

2.3.1 Device level challenges 21

2.4 Training Neural Networks 22

2.4.1 Artificial neural network 22

2.4.2 Spiking neural network 23

2.4.3 Precision requirement 25

2.5 Summary 26

3 PHASE CHANGE MEMORY MODEL 28

3.1 Introduction 28

3.2 Device Characterization and Modeling 30

3.2.1 Accumulative behavior 32

3.2.2 Conductance drift and read noise 34

TABLE OF CONTENTS
(Continued)

Chapter **Page**

3.2.3 Overall model description and validation 36

3.2.4 Generalizability of the PCM model 40

3.3 Spiking Neural Network with Modeled PCM-synapses 43

3.4 Deep Learning with Modeled PCM-synapses 47

3.5 Discussion 51

3.6 Summary 53

4 MIXED-PRECISION TRAINING ARCHITECTURE 54

4.1 Introduction 54

4.2 Computational Memory: Key Challenges 55

4.3 Mixed-Precision Computational Memory Architecture 56

4.4 Evaluation of the Mixed-Precision Architecture 58

4.4.1 The simulation framework 58

4.4.2 Inaccuracies arising from weight-updates 59

4.4.3 Inaccuracies arising from matrix-vector multiplication 65

4.5 Discussion 69

4.6 Summary 71

5 EXPERIMENTAL VALIDATION OF MIXED-PRECISION TRAINING ARCHITECTURE 73

5.1 Introduction 73

5.2 Experiment for Training Handwritten Digit Classification Network . . 73

5.2.1 Training using differential PCM synapses 73

5.2.2 On-chip inference 80

5.2.3 Training using on-chip ohmic multiplication 80

5.2.4 Weight refresh methods 81

5.2.5 Training experiment using non-differential PCM synapse 84

5.3 Training Deeper Neural Networks 86

TABLE OF CONTENTS
(Continued)

Chapter **Page**

5.4 Mixed-Precision Weight Update 88

5.5 Independent Training of Neural Networks 90

5.5.1 Adaptive conductance to weight mapping 90

5.5.2 Crossbar replication for pipeline processing 94

5.6 Performance Assessment 96

5.7 Summary 98

6 TRAINING OF SPIKING NEURAL NETWORK USING PHASE-CHANGE MEMORY ARRAYS 100

6.1 Introduction 100

6.2 PCM Conductance Drift 101

6.3 SNN Training Experiment 103

6.4 Training Accuracy 107

6.5 SNN Inference on PCM Array 109

6.6 Summary 111

7 RESISTIVE RANDOM ACCESS MEMORY FOR SYNAPSE 113

7.1 Introduction 113

7.2 Resistance Switching Mechanism 114

7.3 Fabrication and Characterization of Cu/SiO_2/W RRAM 115

7.3.1 Conductance quantization 116

7.4 Cu/SiO_2/W RRAM as a Synapse 121

7.4.1 Spike-timing dependent plasticity demonstration 123

7.4.2 State dependent phenomenological model 126

7.5 Supervised Learning Emulation 131

7.6 Discussion 134

7.7 Summary 138

8 FUTURE OUTLOOK 139

TABLE OF CONTENTS
(Continued)

Chapter **Page**

APPENDIX PCM-BASED HARDWARE PLATFORM 141

BIBLIOGRAPHY 144

LIST OF TABLES

Table **Page**

3.1 PCM Model Parameters 36

7.1 Cu/SiO$_2$/W Phenomenological Model Parameters 129

LIST OF FIGURES

Figure **Page**

1.1 **a** Drawing of Golgi-stained cortex of a 1.5 month old infant. A network of layers of neurons (dark spots) are visible. The brain inspired structure of artificial neural network is also shown. **b** A crossbar array of analog memory devices performing the weighted summation in a neural network layer. 3

1.2 **a** von Neumann architecture. To perform any operation f on data A, the data A has to be brought into the processing unit **b** Computational memory architecture. The physical aspects of the analog devices states are used to perform certain computations locally on the memory without additional data movement. 7

2.1 **a** Neurons connect and exchange information via junctions between axons and dendrites, called synapses. **b** Synaptic conductance modification observed in a rat hippocampal neuron: an example of STDP observed in biology. 13

2.2 **a** Phase-change memory **b** Resistive random access memory **c** Magnetic tunnel junction **d** Organic memory 15

2.3 **a** Synaptic conductance change observed in the hippocampal neurons in a rat. The change in excitatory postsynaptic current (EPSC) amplitude versus its initial value for causal and anti-causal spike pairs shows its state-dependent and asymmetric conductance change. **b** The average conductance change $\mu_{\Delta G}$ versus average initial conductance μ_G. 20

2.4 **a** Representation of ANN and **b** SNN. We use this framework to discuss the supervised learning algorithms for these networks. 22

3.1 **a** TEM image of a mushroom-type PCM device. Amorphous dome (Amor-GST) inside the crystalline (Cryst-GST) dielectric is visible **b** Pictorial representation of the programming pulses and the resulting relative temperature for RESET, SET, partial SET and read operation in PCM **c** Schematic illustration of the application of PCM devices as synaptic elements in neuromorphic computing. A crossbar array of PCM devices could be used to represent the connection strengths in a neural network layer. 29

3.2 The measured conductance evolution in a single device in accordance with the application of 20 consecutive partial SET pulses. The time instances of programming are indicated. After the application of each pulse, the device conductance is measured 50 times. 31

LIST OF FIGURES
(Continued)

Figure **Page**

3.3 **a** The statistics of cumulative conductance evolution as a function of the number of partial SET pulses. The error bars indicate one standard deviation. **b** The mean $\mu_{\Delta G}$ and **c** standard deviation $\sigma_{\Delta G}$ of conductance change as a function of the average initial conductance μ_G for each programming pulse. The initial conductance distribution for each programming pulse is divided into smaller intervals and μ_G, $\mu_{\Delta G}$ and $\sigma_{\Delta G}$ is determined separately for each interval. Each data point in **b** and **c** corresponds to an average of measurements from at least 100 devices. Also depicted are the fit lines used to obtain the model parameters. **d**, **e** The same data points of $\mu_{\Delta G}$ and $\sigma_{\Delta G}$ are plotted as a function of the pulse number with a constant added for data points corresponding to a single μ_G interval. The dependency of $\mu_{\Delta G}$ and $\sigma_{\Delta G}$ on pulse number is approximated using an exponential function with a decay constant of 2.6. 32

3.4 **a** The average conductance evolution after each programming pulse obtained by the 50 read operation is fitted using Equation 3.3. The estimated drift-coefficient, ν, is shown in the inset **b** The read noise measured from the device array is plotted as a function of the average device conductance. The linear fit used to estimate read noise for the model in a state-dependent manner is also shown. 34

3.5 **a** The distribution of conductance values obtained using the model as a function of the number of partial SET pulses and match with experiments. **b** The mean of the conductance as a function of the pulse number. **c** The standard deviation of the conductance as a function of the pulse number. 38

3.6 The distribution of conductance values after **a** initialization **b** the application of 5 programming pulses and **c** the application of 20 programming pulses. It can be seen that there is a remarkable agreement between the experimental distribution and that predicted by the model. **d** The correlation coefficient between the G and ΔG after the application of each programming pulse calculated based on the model and is compared with the experimental measurement. The conductance evolution of individual devices as measured experimentally **e** and as predicted by the model **f**. 39

3.7 The 10,000 PCM devices are programmed using sequences of 20 current pulses of 90 μA and 50 ns width. The number of read operations performed after each programming event is varied resulting in different time intervals between the programming events. The resulting conductance evolution during all the read operations is illustrated. The proposed model captures the experimentally observed behavior remarkably well. 40

LIST OF FIGURES
(Continued)

Figure **Page**

3.8 **a** Average conductance evolution of 10,000 devices subjected to a sequence of twenty 50 ns, 60 μA pulses followed by fifteen pulses each of 80 μA, 100 μA, and 120 μA amplitude. The error bars show one standard deviation. The corresponding model response fitted to the device is shown overlapped. **b** Mean of the conductance change versus the average initial conductance state. The data points corresponding to different programming amplitudes are color coded. **c** The standard deviation of the conductance change versus the average initial conductance state. 42

3.9 **a** The spiking neural network used for the training simulation. Two PCM devices in the differential configuration used for the synapse is also shown. **b** Raster plot of input spike streams (top) and the desired spike streams from the output neuron (bottom). The observed spikes during each training epoch are plotted over the desired spike trains. The observed spikes are within 0.7 ms of the desired instances after 40 epochs of training. **c** Conductance evolution of a few synapses during the training illustrating the drift and read noise of PCM devices **d** The final weight distribution from the PCM synapses along with the weights from a floating point (FP-64 bit) training is shown for reference. **e** The correlation between the desired and observed spike trains after 40 epochs with the PCM synapse is similar to that using floating point synapses. 44

3.10 **a** The artificial neural network used for handwritten digit classification based on the MNIST dataset. **b** The network weights are implemented using 2 PCMs in differential configuration ($W = \beta(G_p - G_n)$). The conductance evolutions of the device pairs from few synapses are shown for the training duration. **c** The neuron activations and errors have been scaled to determine the weight update probability. The average number of weight updates per image in the two weight layers during the training for two chosen scaling factors are shown. **d** The test accuracies for the two update probabilities. The P(update)=p updated a smaller number of synapses using limited precision and stochastic PCM models and achieved higher test accuracies faster. 48

LIST OF FIGURES
(Continued)

Figure **Page**

4.1 **a** Neural network consisting of multiple layers of neurons with weighted interconnects. During forward propagation, the neuron response x_i is weighted according to the connection strength W_{ji} and summed. Subsequently, a non-linear function f is applied to determine the response of neurons in the next layer, x_j. Similarly, the gradients from the layer, δ_j, is back-propagated through the weight layer to determine the gradients, δ_i, for the preceding layer. **b** Mixed-precision architecture consisting of a computational memory unit and a high-precision digital unit. Computational memory has memristive devices whose conductance values, G_{ji} represent the weights, W_{ji}. The crossbar array can perform the weighted summation during the forward and backward propagation. The resulting xs and δs could be used to determine the weights updates, ΔW in the high-precision unit. The ΔW is accumulated in the variable χ. The number of device programming pulses is determined as $p = \lfloor \chi/\epsilon \rfloor$, where ϵ is the average device update granularity. 58

4.2 The neural network used to evaluate the mixed-precision architecture. The objective is the classification of handwritten digits based on the MNIST data set. There are 784 input neurons, 250 hidden sigmoid neurons, and 10 output sigmoid neurons. The network weights are trained by optimizing a quadratic objective function using gradient descent. All the 60,000 images in the dataset are used in one epoch of training and 10,000 images for testing. 59

4.3 Effect of granularity and stochasticity associated with weight updates. Linear devices with symmetric potentiation and depression granularity are assumed as computational memory elements. The standard deviation of the weight-update randomness, $\sigma(\Delta\widehat{W})$, is taken as a multiple of the weight update granularity, ϵ. The error-bars indicate the standard deviation corresponding to five repetitions of the simulation. It can be seen that even in the extreme cases of highly coarse and random weight-updates, drop in the test accuracy is still within approximately 4% for 2-bit granularity. 60

LIST OF FIGURES
(Continued)

Figure **Page**

4.4 **a** In the linear device simulations, the weights are initialized to a set of states -1, 0, and 1. **b** In non-stochastic two-bit granular updates the devices go through only these discrete states and hence the update granularity also becomes the device resolution. This discrete weight solution gave a test accuracy of approximately 97%. However, in the case of stochastic programming, the devices can achieve intermediate states. **c** The final weight distribution from the 3-bit update granularity simulation is also shown. The higher weight resolution improved the test accuracy by approximately 1%. 61

4.5 Sparsity of device updates in the computational memory array. The device update count per epoch is plotted for different values of ϵ. Only a fraction of the total number of devices is programmed after each image. The number of synapses in each layer times the total training image count is indicated as the reference. As the value of ϵ increases, the number of updates reduces by several orders of magnitude, saving significant programming overhead. 62

4.6 Effect of asymmetric conductance update in mixed-precision training. The test accuracy, when trained with devices of fixed 8-bit potentiation granularity and variable depression granularity, is plotted as a function of the depression granularity (expressed in bits). Weight updates are assumed to be deterministic. The resulting test accuracy shows less than 1% accuracy drop even in the highest asymmetric case in the simulation. 63

4.7 **a** Non-linear device model: the weight update (ΔW) is modeled as exponentially dependent on the current state, W. The exponential function for different amounts of non-linearity is plotted. **b** Corresponding device model pulse response, where 14 potentiation pulses followed by the same number of depression pulses are applied to the device. **c** Non-linear device model as synapse for DNN training. $\beta = 0$ correspond to a linear and symmetric device and higher β values indicate an increasing amount of non-linearity. Approximately 4-bit weight update granularity is assumed for the device model and mixed-precision training. Weight updates were non-stochastic. The result shows that there is no significant degradation in the test accuracy even for $\beta = 5$ that corresponds to a highly non-linear conductance response. 64

LIST OF FIGURES
(Continued)

Figure **Page**

4.8 **a** Effect of read noise. Gaussian distributed additive read noise is added with the computational memory devices whenever they are used for multiplication. The standard deviation (STD) of the read noise is varied as a fraction of the total device conductance range. For the mixed-precision training, a 4-bit weight update granularity and non-stochastic programming are assumed. There is no significant loss in test accuracy even up to a read noise corresponding to 5% of the total weight range. **b** Effect of finite resolution data converter. The weight update granularity is assumed to be 4-bit, without stochasticity and read noise. The curve with triangle indicates simulation results where DACs are used at the crossbar input whereas the output current is read back in floating-point precision. The curve with the inverted triangle indicates results where the crossbar input has floating point precision whereas ADCs are used for reading back the output current. 66

4.9 The quantization function at the computational memory periphery should be chosen such that there is no edge at $x = 0$, as seen in the blue curve, as opposed to the red curve. 67

5.1 **a** Network structure used for the on-chip mixed-precision training experiment for MNIST data classification. Each weight W in the network is realized as the difference in conductance values of two PCM cells, G_p and G_n. **b** Stochastic conductance evolution of G_p and G_n corresponding to 5 synaptic weights associated with the second layer during training. **c** The number of device updates per epoch from the two weight layers in mixed-precision training experiment and high-precision software training. It shows the highly sparse nature of weight update in MPA. 74

5.2 **a** Classification accuracies on the training and test set from the mixed-precision training experiment. The maximum experimental test set classification accuracy, 97.73%, is within 0.57% of that obtained in the floating-point (FP64) software training. The experimental behavior is closely matched by the training simulation using the PCM model. **b** Inference performed using the trained PCM weights on-chip on the training and test dataset as a function of time elapsed after training showing negligible accuracy drop over a period of one month. 75

5.3 The PCM model conductance distribution in the two weight layers ((a) and (b)) at the end of 30 epochs of training matches that from the experiment. It indicates that the model is able to capture the overall device programming dynamics in the training scenario which involves many device programming events with varying amounts of intermittent delays and occasional RESET. 77

LIST OF FIGURES
(Continued)

Figure **Page**

5.4 **a** Training simulation of the experiment using multiple random seeds. **b** Training for longer epochs: even though the programming stochasticity could slow down initial training convergence, the MPA accuracy could be equivalent to floating-point given sufficient training epochs. 78

5.5 **a** The PCM model response whose programming noise is scaled by 0.5 is compared with the actual device response. Error bar shows one standard deviation. **b** Classification performance evolution during the training of the two layer MNIST classifier with the PCM model with reduced noise. 79

5.6 Experimental training and test set classification accuracy, when the network in Figure 3a is trained with 5,000 training images and 10,000 test images from the data set where the on-chip device conductance response is used implement the multiplication in the weighted summation operation in the forward and backward propagation. 81

5.7 **a** A comparative study on the impact of different refresh schemes on training and test accuracies using on-chip training experiment. The training experiment was performed with 5000 training images and 10,000 test images from MNIST. **b** An estimate of the number of programming pulses used in the iterative, multi-pulse, and RESET weight refresh schemes. For iterative programming, an average of four pulses per device programming is assumed. The programming pulses for weight refresh involve both the pulses used for the RESET and successive programming pulses applied (none in RESET case) to re-program the device. 82

5.8 Training and test accuracies from the MPA training experiment for MNIST digit classification using the network in Figure 5.1a, where weight increments and decrements are implemented by programming a single PCM device per synapse. Irrespective of the conductance change asymmetry, the training achieves a maximum training accuracy of 98.77% and a maximum test accuracy 97.47% in 30 epochs, within 1% of that obtained in corresponding high-precision (FP64) training. FP64 training has a training and test accuracy of 99.73% and 98.33%. This experiment demonstrates the ability of the MPA to compensate the conductance update asymmetry often observed in the nano-scale memristive devices. 84

5.9 The training and test accuracies are compared for MPA based training when the outer product for the weight update is implemented using 4-bit versions of the neuron activations and errors in contrast to 64-bit floating-point (FP64) in the two layer MNIST classification problem. 89

LIST OF FIGURES
(Continued)

Figure **Page**

5.10 The multiply-accumulate operation can be viewed as the product and sum of random variables. The analog to digital conversion correspond to a scaling (with a quantization). An additional adaptive scaling factor is necessary to map the device conductance to actual network weights. In a backpropagation based training it is necessary to ensure that the scaling in the forward and backward propagation should be the same. Therefore, $K_{cf} \times K_{vf} = K_{cb} \times K_{vb}$. 91

5.11 **a** The evolution of K_{cf} in the two layers of the neural network during its training to classify the MNIST dataset. **b** The training performance from two identical networks initialized to the same state, when trained with an adaptive and fixed scaling factor. The classification accuracy on the 10,000 test images is plotted. 92

5.12 **a** The replicas of the crossbar arrays could be used to process B batches of input vectors in parallel in any weight layer. The batches of vectors might be from different training data or the different subregions of an image in a CNN. **b** The final fully connected layer of a CNN could be replicated and trained to perform different tasks based on the output of shared feature extraction layers. 95

6.1 Neurons in the SNN integrates (both spatially and temporally) asynchronous spike streams from a large number of input neurons to produce its response. Dedicated connections provided by the non-volatile memory array is an efficient parallel and scalable implementation. 101

6.2 Effect of conductance drift in PCM synapses. **a** A single PCM in the positive half in the differential configuration receives the programming pulse. The effective synapse conductance evolution is shown in linear (left) and logarithmic (right) time scale. **b** A single PCM in the negative half in the differential configuration receives the programming pulse. . 102

6.3 The audio signal is passed through a silicon cochlea chip to generate spike streams. These spike streams are sub-sampled and applied as an input to train the single layer SNN. The desired spike response from the networks representing the images (14×12 pixels) corresponding to the characters in the audio is shown at the output of the network. . . 104

LIST OF FIGURES
(Continued)

Figure **Page**

6.4 **a** The observed versus desired conductance change in the PCM from the SNN training experiment. **b** Accuracy is defined as the number of desired spikes with an observed spike generated from the network within a certain time interval. The lower bound of the shaded lines correspond to 5 ms interval, the middle line 10 ms and the upper bound 25 ms. The corresponding training simulation using the PCM model captures the experimental result very well. **c** The raster plot of the desired and observed spike trains from the trained network. A visualization of the character images whose pixel intensities are generated from the observed spike rates is also shown above the raster plot. 106

6.5 **a** Spike generation accuracy as a function of the number of deices in a multi-PCM synapse. **b** Spike generation accuracy can be improved by a mixed-precision (MPA) training approach. Accuracy when trained with PCM models with artificially scaled programming noise and without conductance drift is shown in inset. 108

6.6 **a** Input spike streams with spike times jittered by random amounts uniformly distributed in [-25 ms 25 ms]. **b** The cross correlation between the jittered spike streams are shifted towards zero compared to the experimental input **c** The training accuracy is improved when trained with input spike streams of reduced correlation. 109

6.7 **a** Inference using trained PCM array. Due to conductance drift, the accuracy drops over time (black line). The effect of drift can be compensated by a time-aware scaling factor (red line). **b** The drifted conductance distribution at the end of 10^5s is compared with the trained conductance distribution. The effect of scaling on the drifted conductance is also shown. **c** The images generated by the SNN at the end of training for the audio input (top). The images generated after 10^5 s (middle). The images generated with drift compensation (bottom). The brightness of each pixel represents the spike rate for the duration of each character. 110

LIST OF FIGURES
(Continued)

Figure **Page**

7.1 **a** Crossbar structure of the fabricated device. **b** FEGSEM (field emission gun scanning electron microscope) image of the device (cross section) showing 10 nm SiO_2 sandwiched between Cu and W. **c** Si 2p XPS spectrum of a 10 nm sputter-deposited SiO_2 film. Our quantitative analysis suggests O/Si ratio of 2. **d** Typical low voltage pinched hysteresis observed in the device in linear scale. **e** The first four and the tenth sweep across the device are shown in log scale. **f** Electrical setup schematic used to characterize the device at the cross-point junction is shown. The connection via device contact pads 1 and 2 or 3 and 4 are used for the device characterization, and those through 1 and 3 or 2 and 4 are used for the electrode characterization. 117

7.2 **a** Voltage across the device and corresponding conductance change of a Cu/SiO_2/W device during current sweep measurement. **b** The corresponding conductance histogram of quantization levels. We can see the first voltage drop and the histogram indicate the presence of a quantized conductance level below $0.5\,G_0$. 118

7.3 **a** Schematic of Cu filament formed inside the SiO_2 dielectric. **b** $E-k$ band representation for $0.5\,G_0$ and $1\,G_0$ states marked on the corresponding $G\,vs.\,I$ plot. **c** Dual current sweep measurement across an already formed filament which shows a conductance transition from $3.5\,G_0$ to $4.5\,G_0$. **d** Plot of the Fermi level splitting ($E_{FL} - E_{FR} \sim e\,V$) across the filament in each quantized conductance state ($(n/2)\,G_0 \pm 0.05\,G_0$) during filament formation in a current sweep. **e** An estimate of Cu filament radius (R_{Cu}) using Equation 7.2. 120

7.4 Bio-inspired programming waveforms used in the experiment to demonstrate STDP in a memristive device. 122

LIST OF FIGURES
(Continued)

Figure **Page**

7.5 **a** Memory switching behavior of the Cu/SiO$_2$/W cross-point device for two voltage sweeps with maximum amplitude of 350 mV and 450 mV. The current corresponding to a conductance of G$_0$($= 2e^2/h$) is marked using a dashed line. The conductance response above the G$_0$ level (blue curve) indicates at least one filamentary path connecting the top and bottom electrode, while the orange curve shows partial conductance switching in the sub-quantization regime. The 3D device structure is shown in the inset. **b** STPD programming and measurement set-up **c** Example STDP programming waveforms applied (top) and the measured device conductance evolution (bottom) where $\Delta t = 5$ ms results in a synaptic potentiation and $\Delta t = -5$ ms results in a synaptic depression. **d** STDP response of the device determined as the average conductance change (normalized) versus the spike time difference based on 400 measurements. $\Delta G_{norm} = (G_f - G_i)/min(G_i, G_f)$ **e** Energy spent per programming event in the device as a function of Δt **f** Evolution of the device conductance during the STDP measurement tracks the overall causal/anti-causal signal correlation. The orange curve is a cumulative sum of the sign of the Δt across the sequence of measurements. 125

7.6 **a** Average ΔG_{norm} defined as $(G_f - G_i)/min(G_i, G_f)$ response from the device measurement when the initial conductance is below 0.05 G_0 with an average of 0.03 G_0 and when **b** it is in the range between 0.05 G_0 and 0.16 G_0 with an average of 0.1 G_0. **c** The response of the phenomenological model, when programmed with a sequence of randomly chosen Δts. **d** The device conductance after the application of the pulse (G_f), calculated from the phenomenological model is well correlated with the experimental values ($R^2 \sim 0.56$), for the same initial conductance values and programming Δts as in the experiment over a dynamic range of two orders of magnitude. 128

7.7 Peak points and the corresponding double-exponential function fit for ΔG_{norm} versus Δt data with low (**a**, **b**) and medium (**c**, **d**) initial conductance are shown separately for positive and negative Δts. . . . 131

7.8 The τ_A and τ_B as a function of G_i is fitted with linear functions for positive Δt in **a** and negative Δt in **b**. The solid markers denote the τ values obtained by the data fitting and open markers are obtained by extending the linear fit. The meeting points of the lines determine the boundary conductance for the STDP operation of the phenomenological model. 132

LIST OF FIGURES
(Continued)

Figure **Page**

7.9 **a** An exemplary spiking neural network with N input neurons connected to a single output neuron. The training task is to discover the synaptic weights such that the output spike response, $S_o(t)$, matches the desired response, $S_d(t)$, for a specified input spike excitation, $S_i(t)$. **b** Training rule: the synapse is potentiated or depressed based on the time difference of desired or observed spikes respectively from the efferent spike using the STDP model. **c** A raster plot of the spike streams from each input neuron in a 1000×1 SNN is shown. **d** The desired and observed spike trains over the training epochs (top) and the final membrane potential (bottom) from the output neuron as a function of time. **e** Device conductance evolution for four exemplary synapses during the training of the 1000×1 neural network. **f** Input (top) and observed output (bottom) neuron spike rate in an SNN trained for sequence prediction. **g** Relative conductance distribution of synaptic device models connected to three output neurons showing the features learned after training. The first set of synapses is connected to a neuron responding to letters N and J. The second set of synapses is connected to an output neuron responding only to letter N. The third output neuron had the desired spike having anti-causal relation with most of the input spikes leading to an effective synaptic depression. 133

.1 **a** Schematic of the hardware platform. **b** PCM array layout along with with a cross-sectional tunneling electron microscopy (TEM) image of the mushroom-type PCM cell. **c** PCM chip specifications. 142

.2 **a** Integrated circuit used in the PCM chip for programming and readout. **b** Iterative programming algorithm. 142

CHAPTER 1

INTRODUCTION

Throughout the story of human evolution, we relied upon artificial media ranging from the surface of rocks to supercomputers to augment our capability to imagine, hypothesize, test, and execute ideas. These media have served to store data and information and has also enabled high-precision computation and data processing. This has enabled us to progressively solve more complex problems and create tools which in turn are capable of generating data at an ever-increasing rate. Data that come in many forms such as visual and audio streams, economic transactions, or medical history of a patient, and which was traditionally processed by the human brain, is today being automated by machine learning techniques running on cloud servers. However, there is a growing need to process such data at the source with high efficiency, as computational resources cannot keep up with the data generation rate and many real-life applications require systems that are constantly on. Therefore, we need more efficient implementations for these data processors and new computer architectures which are alternative to or augment the current von Neumann systems.

The potential of information that could be harnessed from this Big-data cannot be over-stressed. Human population, currently about 7.4 billion, has approximately doubled over the last 50 years, thanks to several technological advancements which have improved life expectancy and living standards. At the same time, today's consumption-driven economy is also creating several stresses in the ecosystem, ranging from over utilizing its natural resources, as well as drastic changes in climate and environmental health. Hence, new tools to process the vast troves of data to generate meaningful information can play a significant role to substantially improve the overall quality of life in a sustainable manner.

What computational model is suitable for large data processing applications in the field? The most promising clue to this question may come from the human brain anatomy itself. Its parallel and distributed computing architecture can process a large amount of information with incredibly high efficiency. Its distributed processing/memory elements learn and adapts and make corrections based on local events. On the other hand, most of today's computers rely on the von Neumann computer architecture has a central processing unit which is physically separated from the memory unit and operates based on pre-written sequential algorithms. In most data processing applications, the amount of data is often much larger than the available on-chip memory and hence they need to constantly access a larger and slower off-chip memory. The limited memory bandwidth chokes the processor causing severe performance degradation. Further, the off-chip memory access is orders of magnitude costlier in energy than on-chip memory access, making the overall system highly energy inefficient for modern-day problems [1]. Also, the sequential algorithms for the new data processing applications have proven to be less efficient than the brain-inspired neural network based solutions. While there are many successful demonstrations of artificial neural networks (ANNs) solving problems often surpassing human capability, they are made possible by the computational power of the graphical processing units (GPUs), the most common parallel architecture today. While they meet the research requirements, their cost and power consumption are still very high to be deployed widely and economically.

In short, ubiquitous deployment of brain-inspired computational units requires energy efficient and scalable architectures. This dissertation discusses some results that could enable the development of computing units for the efficient on-chip implementation of the architecture and learning algorithms for bio-inspired neural networks.

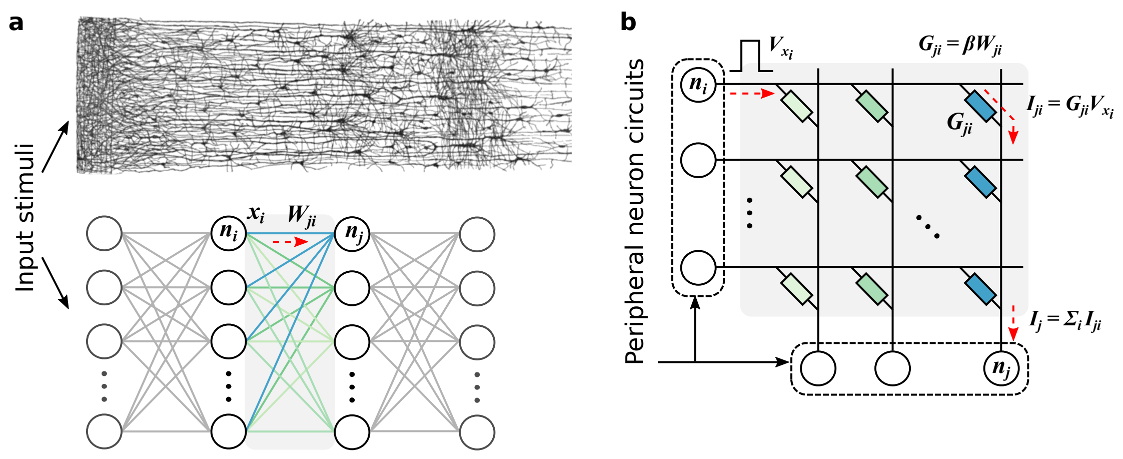


Figure 1.1 **a** Drawing of Golgi-stained cortex of a 1.5 month old infant. A network of layers of neurons (dark spots) are visible. The brain inspired structure of artificial neural network is also shown. **b** A crossbar array of analog memory devices performing the weighted summation in a neural network layer.
Source: [2].

1.1 Computational Engine in the Human Brain

The motivation for the brain-inspired computation is the relative ease with which the human brain perform various cognitive tasks consuming a mere 20 W of power which supercomputers cannot achieve while consuming millions of Watts. Our understanding of the brain's underlying architecture which enables this computational efficiency is being constantly improved with the advancement of brain imaging technologies such as functional magnetic resonance imaging (fMRI). The basic elements of the architecture which inspire artificial neural network models could be described as follows.

The brain is a complex network of unit cells called neurons. The neurons are arranged in layers and are interconnected via junction called synapses. Neurons act as parallel processing units which constantly receives inputs, integrates them, and make simple decisions. Each neuron communicates this information to neighboring neurons in the form of signal spikes via the synaptic junctions of specific conductance. Synapses also act as a local memory by storing an account of the spiking activities by modulating its conductance. It is the plasticity observed in these biological synapses that are thought to be the source of learning and adaptation in the brain.

ANNs have mimicked the connectivity patterns and connection weight adaptation at different levels of abstraction (Figure 1.1a). By appropriately training large neural networks, complex input-output mappings can be learned, enabling highly useful functions to be realized, which are otherwise almost impossible with traditional sequential algorithms. At the same time, we are still very far from achieving energy efficiency, generalizability, or the cognition level of the human brain.

What are we missing? One important aspect seem to be the size of the network itself. Our brain is estimated to have approximately 10^{11} neurons. Each neuron in the human brain is estimated to be connected to approximately 10^4 other neurons. We have observed that the computational power of artificial neural networks (ANNs) are indeed proportional to their sizes [3]. However, ANNs are orders of magnitude below in size compared to their biological counterparts and yet the computational engines we currently use to run them consume at least an order of magnitude higher power. This is one of the compelling reasons to look for nanoscale devices that could create scalable and energy efficient implementations of neural network architectures.

1.2 Computational Architectures

In this section, we look at different computational architectures available today and their suitability to implement neural networks. A deep neural network consists of many layers of neurons, each of which receives a weighted sum of inputs from its previous layer. This operation corresponds to a matrix-vector multiplication which is of computational complexity $\mathcal{O}(N^2)$ for an $N \times N$ matrix and an $N \times 1$ vector. Often, the input vector grows into a matrix when each weight update is based on a batch of training examples or when sub-regions of images are processed separately by a set of shared weights as in a convolution neural network. This increases the computational complexity of matrix processing by another degree. The matrix multiplications have been observed to constitute around 95% of the training time. Hence, the efficacy of any neural network training architecture will be decided by how efficient they are implementing matrix multiplications.

1.2.1 von Neumann architecture

von-Neumann architecture is characterized by a physically separate processing and memory unit (Figure 1.2). For every clock cycle, the operating system brings in instructions into a central processing unit (CPU), decode them, access necessary data, execute the operation, and results are written back to the memory if necessary. In such a sequential machine, the multiplication of two $N \times N$ matrices is $\mathcal{O}(N^3)$ complex and require $3N^2$ (read and write back) access to memory. Even with the fastest algorithm, computational complexity is as high as $\mathcal{O}(N^{2.373})$ [4]. The performance of the processors in this architecture has been improving by a constant increase in transistor density (which however is slowing down as we approach the end of Moore's law scaling), pipe-lining, and superscalar architecture. Modern CPUs have multiple cores each permitting independent operations (MIMD - multiple instructions multiple data). For example, Intel i9-9980XE has 18 cores and AMD Ryzen has up to 32 cores [5, 6]. However, these cores are designed to perform instructions sequentially. Further, these performance improvements have only moved the bottleneck to memory access. Due to the limited on-chip memory and lack of any non-volatile memory, the instructions and data are stored in off-chip memories which are communicated to the processor over a shared bus. As a result, it takes several weeks to train state-of-the-art neural networks on today's von-Neumann machines.

In short, the general purpose design of the CPUs that is primarily tuned to perform accurate computations is not ideal to satisfy the parallel compute requirement of the neural networks involving large matrix multiplication operations.

1.2.2 Graphics processing units

Graphics processing units (GPU) have become an excellent source of computational parallelism in scientific computing today. In contrast to the general computing notion of CPUs, GPUs are application specific. They were originally designed for rendering lighting and projection operations for image pixels by a large number of parallel processing units called streaming processors executing programs called shaders. Later, NVIDIA generalized these computing units to be more programmable for non-graphic operations, gaining the name general purpose graphical processing units (GPGPUs). GPGPUs differ from the CPUs in terms of its core organization and architecture. By eliminating operating systems tasks and using approximate compute units for special functions such as *exp*, *log*, and *sin*, the complexity of GPGPU compute cores are simplified, allowing a large number of them to be integrated on a chip, compared to a few numbers of sophisticated cores available in CPUs. For example, the Tesla V100 GPU is organized as 80 streaming processors each of which contains 64, 32-bit floating point (FP32) and 32 FP64 units [7]. These processing units are designed for single instruction multiple data (SIMD) architecture and do not allow independent operation. While the individual cores run slower than standard CPU cores, the extreme parallelism allows them to achieve performance improvement by around an order of magnitude in certain problems typical in graphical processing. Today's deep learning frameworks leverage this compute parallelism to perform large matrix multiplications which accelerates neural network training. GPUs have enabled the training of large neural networks with large datasets in reasonable amounts of time. Recent GPU architectures such as in Tesla V100 include tensor cores which allow some mixed-precision computing using sub-32 bit numbers to accelerate deep learning. They offer 14 teraFLOPS (floating point operations per second) on single-precision (32-bit) and 112 teraFLOPS using tensor cores.

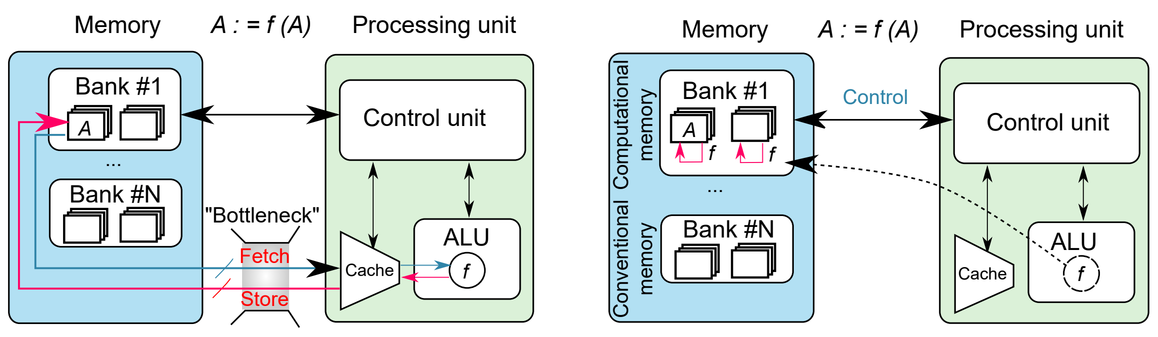


Figure 1.2 **a** von Neumann architecture. To perform any operation f on data A, the data A has to be brought into the processing unit **b** Computational memory architecture. The physical aspects of the analog devices states are used to perform certain computations locally on the memory without additional data movement. *Source:* [8].

However, the computational power of GPUs is achieved at the cost of chip areas as large as 815 mm^2 (larger than recent CPUs) harboring over 20 billion transistors consuming 250 Watts of power. GPUs are also slave devices which are dependent on CPUs for communication. This limits its scalability for larger networks and employment in energy critical applications.

1.2.3 Non-volatile memory arrays: computational memory

Computational memory is an application specific design performing analog computations using the physical aspects of memory devices. This is often considered a non-von Neumann architecture as the operations it is devised for is performed in the memory array, without bringing them into the processor, which saves data movement (Figure 1.2). For example, if a number is represented as the analog conductance of a non-volatile memory device, its multiplication with another number could be computed by applying the second number as a voltage across it and reading the resulting current. There are several recent demonstrations of using this concept such as for performing bulk bit-wise operations [9], computing matrix-vector multiplications [10, 11, 12, 13], and finding temporal correlations [8] more efficiently. For a neural network, if the connection weights are stored as the conductance of memory devices at the crosspoints of a crossbar array (Figure 1.1b), it could perform the weighted summation operation as a matrix-vector multiplication. When the input vector is applied as voltages to the array's word lines with the bit lines held at ground, Ohm's law performs the multiplications, Kirchoff's law performs the summations and the result can be read as currents from the bit lines. An $n \times n$ crossbar array performs n^2 multiplications and $n \times (n-1)$ additions in parallel in analog domain in a fixed amount of time irrespective of the value of n. In effect, it implements $\mathcal{O}(N^2)$ complex matrix operations in $\mathcal{O}(1)$ complexity. Further, the area of a two-terminal non-volatile analog memory device could be as small as $4F^2$ where F is the smallest feature size that can be fabricated in a technology node. Hence, the crossbar analog memory array is the closest architecture we have today that mimic the inherent parallelism and dedicated connectivity of the human brain, which also provides fast computation, high integration density, and scalability.

However, computational memory faces a major challenge in utilizing it to train neural networks. The high-speed computation and areal density of the analog array come at the cost of numerical precision. The floating point numbers typically used to represent the network weights in the digital systems can have an arbitrary precision and dynamic range. On the other hand, the precision with which the numbers can be represented and updated in the analog memory devices are constrained by their physical limitations of conductance programming. What is the right architecture to use such devices for reliably training neural networks? We will analyze this aspect in the succeeding chapters.

1.2.4 Potential of analog memory based neural networks

The computational complexity reduction offered by the analog memory arrays could accelerate matrix multiplications. The simple structure of the array permits the fabrication of larger neural networks. If these devices could be trained effectively without losing accuracy compared to their digital implementations, then it could represent a very efficient parallel architecture for neural networks and economic for ubiquitous deployment compared to GPGPUs.

The scalability of computational arrays may allow us to integrate large neural networks with the CPUs while consuming only a small portion of the resources on-chip. ANNs, when integrated with the standard digital computers, can solve certain classes of problems in a very efficient manner. ANNs could search through large data sets to detect patterns and anomalies without being restrained by the limitations of biological systems such as fatigue, with the digital computer acting as a bookkeeper and a calculator. Neural networks and digital computers are optimized to perform different classes of problems and hence when combined to complement each other, new classes of problems could be solved more efficiently. Architectures combining them could create new applications and perform complex computations without human intervention every time.

1.3 Further in this Dissertation

This dissertation aims to identify some of the challenges posed by nanoscale synaptic devices for efficiently training large neural networks and provide architectural solutions for them. Our approach involves on-chip experiments and simulation studies using reliable statistical models of nanoscale device programming.

In Chapter 2 we review the analog memory device behavior and the challenges they pose for training neural network algorithms. Chapter 3 presents characterization and modeling of the gradual conductance modulation of phase-change memory devices. Chapter 4 presents a mixed-precision architecture and discuss its efficacy in solving various analog memory non-idealities. Chapter 5 presents several experimental validations of the mixed-precision architecture to effectively train large neural networks. The phase-change memory model developed is used to validate the architecture in more complex networks. Chapter 6 presents the training and inference experiment of a spiking neural network using the phase-change memory devices as synapses. Chapter 7 presents the synaptic characteristics of resistive memory devices. Chapter 8 presents future outlook and topics for further research.

CHAPTER 2

NANOSCALE MEMORY DEVICES AND LEARNING

2.1 Introduction

Artificial neural networks (ANNs) have been successful in demonstrating human equivalent performance in several complex cognitive tasks [3]. Weight adaption is the source of learning in such networks. The floating point numbers used to represent the weights are highly precise and reliable. For example, a 32-bit floating point (FP32) number can represent numbers from 10^{-38} to 10^{38} with up to 10 decimal places. However, this precise representation is costly in area and energy. A single bit in these numbers is realized using either an SRAM cell with a minimum of 6 transistors or a DRAM cell which uses one transistor and a capacitor. An FP32 number uses 32 such cells and it needs an additional logic (based on some predefined standards such as IEEE754) to interpret and process the data stored in it. As a result, they inherently separate the processor and memory. Since the analog states of the device in these cells are not directly correlated with the floating point (or fixed point) network-weights stored in them, these digital memory arrays are not suitable for in-memory matrix multiplications. While DRAM is relatively cheaper and allows larger size, it is slower compared to SRAM and the technology mandates integrating them on a separate chip. Such cost-speed trade-offs have led to a memory hierarchy in the current computer architectures, with a limited amount of fast on-chip memory and the majority of the data stored off-chip. Off-chip DRAM access is approximately 200$\times$ costlier compared to accessing registers sitting next to the processor [1]. This constitutes a significant energy cost and bottleneck in the training of large neural networks with millions of parameters and large datasets.

However, recent works demonstrate that the actual precision required by the neural network weights could be much lower, even though a higher precision is observed to be necessary to train them [14]. At the same time, biological neural networks are able to achieve much higher energy efficiency and generalizability while using synapses which are observed to be of limited precision and stochastic [15]. Nanoscale analog memory devices which trade off precision for the area and efficiency present a similar alternative for hardware implementation of neural networks. These devices are often back-end-of-the-line compatible in the standard CMOS processing flow and have a favorable structure/process for 3D stacking. At the same time, the physical nature of these devices presents several challenges to reliably store and alter data in them. This motivates our study of nanoscale devices to implement and train the synapses of large neural networks.

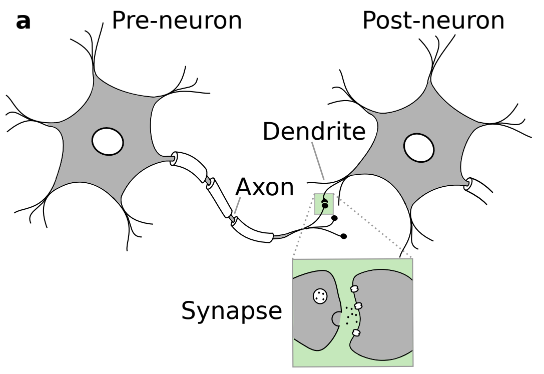
In this chapter, we discuss the general behavior of nanoscale devices and the challenges they present to the implementation of conventional learning algorithms.

2.2 Synaptic Devices

In this section, we compare and contrast the physical mechanisms of conductance change in biological synapses and the analog memory based synaptic devices.

2.2.1 Synapse

Synapses, which are the junctions between neurons in the brain, modulate the signals passed from the axon of one neuron to the dendrites of other neurons (Figure 2.1a). Modulating the synaptic conductance alters the properties of the signals passed through them, and hence they are thought to be the basis of learning, adaption, and memory. There are two types of synapses. Electrical synapses are assumed to be of fixed conductance, lower latency and are assumed to function as fast communication channels in certain networks. On the other hand, chemical synapses are known to



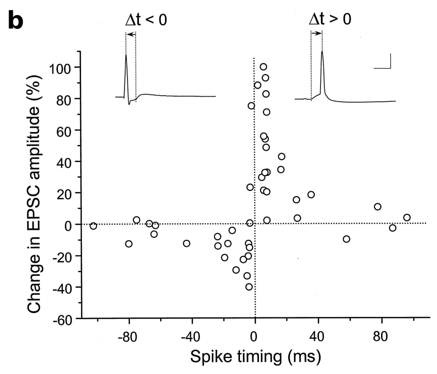


Figure 2.1 **a** Neurons connect and exchange information via junctions between axons and dendrites, called synapses. **b** Synaptic conductance modification observed in a rat hippocampal neuron: an example of STDP observed in biology.
Source: [15].

change their conductance based on the activity on the terminal neurons. Information flow through chemical synapse involves the following processes. Axon terminals of the neurons have sacs called synaptic vesicles filled with chemicals known as neurotransmitters. When an action potential reaches the axon of the neurons, the voltage-gated Ca^{2+} gates are activated causing an inflow of those ions into the cell. Ca^{2+} causes the synaptic vesicles to fuse with the cell membrane and the release of the neurotransmitters. Neurotransmitters diffuse across the synaptic cleft and bind with the receptor proteins on the dendrites of the post-synaptic neurons. Depending on the nature of the Neurotransmitters, the ion channels in the receptor neurons open or closes, leading to a positive or negative ionic current to flow into the cell, causing it to depolarize or hyperpolarize compared to its previous state. Thus, synapse could be either excitatory or inhibitory. Excitatory synapse causes the membrane potential to raise and eventually causing the postsynaptic neuron to fire a spike of approximately 100 mV above the resting potential when sufficient excitation is received.

The conductance of the synapses is determined by the number of neurotransmitters released in response to an action potential into the synaptic cleft or by the number of receptors on the post-synaptic neuron. Coincident spiking activity of the pre- and post-synaptic neurons have been observed to alter the conductivity of the synapses [16, 17] via insertion or internalization of the receptor molecules on the synaptic terminals. An example STDP observed in a rat hippocampal neuron is shown in Figure 7.4b [15], and it plots the conductance change as a function of causal and anti-causal spike time differences. Such spike timing dependent plasticity (STDP) rules based on local neural activities are believed to be integral to the energy and computational efficiency of the brain [18, 19, 20, 21].

2.2.2 Non-volatile analog memories

In order to build systems that mimic the massive parallelism and local learning aspects of the human brain, compact electronic devices that implement the dynamics of neurons and synapses are required. Memristive devices that exhibit conductivity modulation based on past programming history are excellent candidates to realize synaptic memory [22, 23]. There have been numerous synaptic device demonstrations in oxides [24, 25, 26], and chalcogenides [27, 28] which show analog conductivity modulation based on non-volatile rearrangements of atomic configurations within the active volume of the device. Meanwhile, these devices present many challenges in terms of programming stochasticity and asymmetry, granularity, reliability, and the energy required for implementing conductivity modulation, and no single device has so far achieved all the target specifications. For example, Ta/TaO$_x$/TiO$_2$/Ti device has demonstrated femto-Joule level programming energies, but requires programming voltages above 18 V [29]. Similarly, chalcogenide-based phase change memory devices have pico-Joule level programming energies, however, the high programming current limits parallel programmability and require access transistors at every cross-point in an array [28]. Hence, physics-driven device engineering to improve various synaptic device requirements and finding the right trade-offs for the targeted applications are necessary.

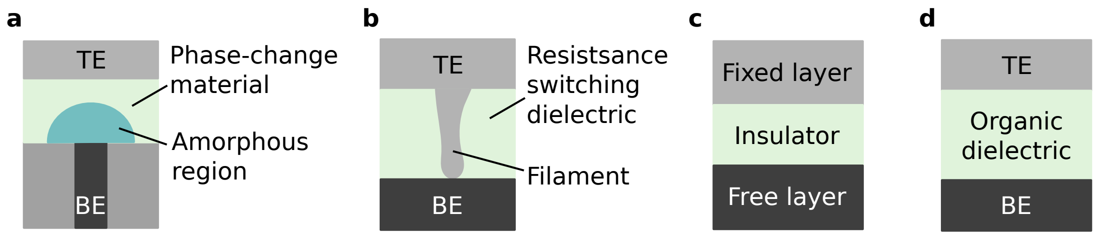


Figure 2.2 **a** Phase-change memory **b** Resistive random access memory **c** Magnetic tunnel junction **d** Organic memory

We now review some of the commonly studied nanoscale non-volatile memory devices that are being explored for neuromorphic computing applications. The following description is taken from the review paper [30].

Phase-change memory Phase change memory is one of the most mature non-volatile memory technologies today and is based on chalcogenide alloys such as GeTe, $Ge_2Sb_2Te_5$, etc. [31, 32]. The reversible electrical resistance switching based on phase transition in such materials was discovered by Ovshinsky in 1968 [33]. If large currents (with density exceeding 10^6 A/cm^2) are passed through poly-crystalline thin films of the material (typically < 100 nm thick) sandwiched between inert metal electrodes (Figure 2.2a), sufficient to raise the temperature above the melting point ($> 600°$C), and if the input excitation is subsequently removed quickly (within few nanoseconds), the molten region can be quenched into an amorphous volume. Since the resistivity of the amorphous phase of the material is much higher compared to the crystalline phase, the device is effectively switched to a high resistance state by this electrical pulse. In the high resistance state, if the applied voltage is such that the electric field across the amorphous volume exceeds a critical field, the device exhibit a negative differential resistance transition accompanied by a rapid increase in the current through the device. With appropriately chosen programming pulses that raise the film temperature above the crystallization temperature (but below the melting point), the amorphous region can be annealed back to its poly-crystalline phase, and the low state resistance of the device can be restored.

PCM devices exhibit excellent endurance ($> 10^{12}$ programming cycles) and retention ($>$ 10 years at 85°C) characteristics [34, 35]. The switching speed of the device lies in the range of few tens to hundreds of nanoseconds. Furthermore, the crystallization of the amorphous volume could be implemented in an incremental manner by using sub-critical or partial crystallization pulses, enabling the device conductance to be gradually increased to higher levels. However, the melt-quench process is less gradual, making it difficult to reduce the conductance levels gradually. As a result, a single PCM cell could be used to mimic gradual potentiation observed in biological synapses. If two PCM devices are used in a differential configuration (i.e., $G_{\text{eff}} = G^{+} - G^{-}$), then both gradual potentiation and depression can be achieved, by incrementally increasing one of the G^{+} or G^{-} devices, with a periodic re-initialization of the conductance of saturated devices [27]. There are many studies showing gradual conductance evolution and STDP behavior in PCM devices [36, 28] and using them for supervised and unsupervised training of conventional [37] and spiking neural networks [38].

Resistive random access memory Resistive random access memory (RRAM) devices exhibit conductance modulation based on electric field driven rearrangement of mobile charged species in a dielectric material sandwiched between two metal electrodes (Figure 2.2b) [39]. The electrochemical process mediating the conduction modulation can be anion-induced or cation-induced. Anion-type RRAMs are characterized by oxygen vacancy low resistance conductance pathways formed by the migration of oxygen ions. This low resistance state can be reversed by applying a field in the opposite direction, causing the recombination of oxygen ions with the vacancies, switching the device back to a high resistance state. Anion-type RRAMs often require an inert electrode which is oxygen ion active or can act as an oxygen ion reservoir during resistance switching. Dielectrics thin films such as TiO_x, HfO_x, SiO_x, TaO_x, AlO_x and WO_x, have demonstrated this kind of oxygen-vacancy mediated resistive switching [40, 41, 42].

Cation-type RRAMs are often characterized by a metallic filament connecting the top and the bottom metal electrodes following a redox reaction; they are also referred to as conductance bridge RAM (CBRAM) devices [43]. These devices require an active top electrode (e.g., Ag, Cu) whose ions are mobile in the dielectric under an applied field. During electrical programming, the metal ions will oxidize, migrate into the dielectric and will get reduced at the other electrode forming a filamentary path. A reversal of applied field will result in the ionic motion in the opposite direction breaking the filament and switching the device back to a high resistance state. CBRAMs have a high on-off ratio and have lower operating voltages, compared to the oxygen vacancy RRAMs.

The low resistance conductance paths formed in the dielectrics are nanoscale filaments, which will result in the observation of quantized conductance states [39, 44, 45]. RRAMs are extensively researched for gradual conductance change and as a synaptic device [46]. The material combination, device geometry, interface effects, doping, annealing, and other fabrication techniques, etc., could be engineered to attain gradual resistance transitions in these devices [47, 48]. For example, W/Al/$Pr_{0.7}Ca_{0.3}MnO_3$ (PCMO)/Pt based RRAM show gradual conductance change due to the oxidation and reduction of AlO_x at the Al/PCMO interface [49] and this dielectric based device has been used for STDP demonstration using bio-mimetic programming waveforms [50]. In a recent work, the filamentary pathway was confined to engineered dislocations in a SiGe epitaxial layer, resulting in gradual conductance changes in the device and improvements in retention, reliability, and endurance [51].

Magnetic random access memory Magnetic RAMs store information in the relative orientation of the magnetization of two ferromagnetic plates separated by a thin insulating material resulting in a magnetic tunnel junction (MTJ) (Figure 2.2c) [52]. One of the plates is of fixed magnetic orientation, while the other is a free layer, whose magnetic orientation can be altered by an external field. The plates could be in parallel or anti-parallel orientation at equilibrium, resulting in a high or low conductance state respectively for the junction. The magnetization of the layer is retained in the absence of an applied voltage, allowing stable binary data storage in the device.

A variant of the MRAM is the spin-transfer torque (STT) RAM, with lower power consumption and more scalability. A spin-polarized current, created by passing it through the fixed magnetic layer, when directed to the free layer results in spin angular momentum exchange due to the interaction between the spins of local magnetization of the layer and that of the free electrons. The free layer magnetic orientation can be switched to a parallel or anti-parallel state depending on the direction of the current [53, 54]. While STT-RAMs predominantly show binary states, there has also been an increased effort in making domain wall based devices to store multiple states [55].

Further, it has also been observed that by either adjusting the programming current amplitude or the pulse-width below the critical conditions for switching, the probability of switching can be tuned [56, 57, 58]. This probabilistic switching behavior could be used to realize gradual conductance change or STDP in a synapse composed of multiple devices configured in a parallel configuration [59, 60].

Ferroelectric random access memory Ferroelectric RAMs use a thin layer of ferroelectric material sandwiched between two metal electrodes. The ferro-electric polarization state of the material is switched between two stable states for conventional solid-state memory applications [61]. Multiple regions of different polarization vectors called ferroelectric domains may be present in a ferroelectric sample [62]. It has been shown recently that the resistance of $BaTiO_3$(2nm)/ $La_{0.67}Sr_{0.33}MnO_3$(30nm) based ferroelectric tunnel junctions can be tuned based on the relative fraction of the ferroelectric domains which points towards one or the other electrode [63]. It is possible to alter the domain population by the application of electrical pulses to the electrodes and hence tune the electrical resistivity. This concept has been used to mimic synaptic plasticity in super-tetragonal $BiFeO_3$ (BFO) tunnel barriers using electrical programming waveforms [64].

Organic Memories Memristors based on the organic compounds are attractive because of the possibility of inexpensive solution-processing based fabrication and chemical tunability of their properties. These devices have an organic thin film that is sandwiched between electrodes (Figure 2.2d). Because of the complex nature of the compounds involved, the physics behind the switching mechanism is often unclear. Structural changes, redox reaction, and field driven polarization have been proposed to explain the switching transitions in these materials [65, 66, 67]. However, except for a recent demonstration [65], these devices generally suffer from low endurance and stability.

In a study based on organic terpyridyl-iron polymer based memristor [66] gradual conductance change, STP and LTP have been demonstrated, taking advantage of the drift of the programmed states. Although these devices require high switching voltages ($\sim$3 V) and long (millisecond) switching times, such explorations demonstrate the feasibility of realizing the complex dynamics of synapses and neurons in potentially inexpensive hardware platforms.

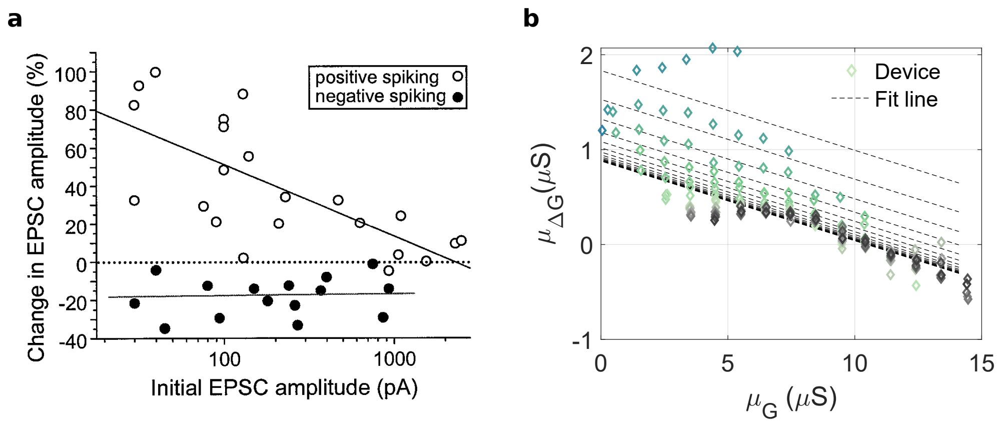


Figure 2.3 **a** Synaptic conductance change observed in the hippocampal neurons in a rat. The change in excitatory postsynaptic current (EPSC) amplitude versus its initial value for causal and anti-causal spike pairs shows its state-dependent and asymmetric conductance change. **b** The average conductance change $\mu_{\Delta G}$ versus average initial conductance μ_G.
Source: [15].

2.3 Ideal Synaptic Characteristics

A common consensus regarding the ideal specifications for the synapses has not yet been reached for high-performance neural networks which find the right trade-off between training accuracy, energy, and generalizability. While most of the artificial neural network implementations use 32-bit floating point numbers, which can be very precise and have high dynamic range, the high-precision requirement originates from the standard implementation of gradient descent based weight update which requires a large number of synapses to be updated by small amounts [68]. Several quantization studies have shown that the actual precision required by the network weights for a task can be much lower [14, 69].

Figure 2.3a shows the state-dependent nature of the synaptic conductance modulation measured in the biological hippocampal neurons in a rat. A continuous decrease in conductance potentiation (positive change) can be observed as the initial conductance of the synapse increases. This indicates a conductance saturation at a level approximately where the amount of potentiation crosses zero. On the other hand, the conductance depressions (negative changes) are smaller and is state-independent. In a different study, the physical aspects of the synaptic conduction have been used to estimate a lower bound for their representation accuracy to approximately 4.6 bits [70, 71]. In Figure 2.3b we show the state-depend nature of conductance update observed in PCM. Its conductance potentiation is stochastic and shows a negative correlation with the present state similar to those observed in the biological example and the average change crosses zero at some point again indicating conductance saturation. In summary, the non-ideal characteristics such as non-linearity, asymmetry, stochasticity, limited dynamic range etc observed in nanoscale devices are also shared by biological synapses to a certain extent. The fact that the brain is able to function with such synapses indicates the existence of solutions to training the nanoscale device networks effectively and efficiently. This motivates our explorations.

2.3.1 Device level challenges

The memory modulations in the analog memory devices are attained by rearranging a few atoms in a nanometric volume, and hence the reliability with which we can achieve a precise conductance change is very low. We term this as the limited granularity of the device. If G is the conductance and ΔG the conductance update, $G/\Delta G$ is $1-20$ typically. Further, tiny disturbances in atomic arrangements in a small volume can lead to relatively higher conductance fluctuations, and hence the updates are stochastic. Due to the limited dynamic range, the device state changes are state-dependent and shows a non-linear update behavior. Also, due to the difference in the physical mechanism involved in the conductance increase (potentiation) and decrease (depression), the updates are unequal for similar programming pulses and it leads to highly asymmetric conductance update behavior.

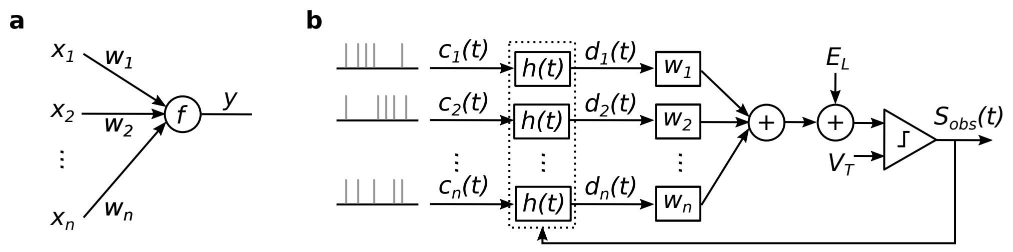


Figure 2.4 **a** Representation of ANN and **b** SNN. We use this framework to discuss the supervised learning algorithms for these networks.

Experiments using the non-ideal characteristics of PCM devices to train neural network classifying handwritten digits based on MNIST dataset have shown limited performance (approximately 82% test accuracy compared to 98% in high-precision software baseline) [72]. Analysis using a linear device model has proposed at least 10-bit precision and less than 2% mismatch between the potentiation and depression to achieve high-precision comparable results [73].

2.4 Training Neural Networks

In this section, we describe the algorithms used to evaluate the training performance of non-volatile memory array based implementations. The supervised training algorithms are reviewed specifically because of their higher performance, which hence serves as a suitable benchmark.

2.4.1 Artificial neural network

ANNs are brain-inspired, and they represent a higher level of abstraction. ANNs have layers of neurons interconnected via synaptic weights. However, in contrast to the spike-domain encoding of data and integrating neurons in biology, the data in ANN are encoded using real numbers and the neurons represent non-linear functions such as sigmoid, tanh, or Rectified Linear Units (ReLU). The basic computing element in an ANN can be shown as in Figure 2.4a. Let $[x_1, x_2, \dots x_n]$ denote the inputs to a weight layer with $[w_1, w_2, \dots w_n]$ as the connection strengths to one of the output neurons, then that neuron's response is given by the relation

$$y = f(\Sigma_i x_i w_i), \tag{2.1}$$

where f is a non-linear activation function of the neuron. These networks are trained using gradient descent. The weight updates are computed to minimize an error function $E = g(y, y_d)$, a function of observed (y) and desired responses (y_d). The gradient of the E w. r. t. a weight w_i is

$$\frac{dE}{dw_i} = x_i . \frac{dg}{dy} . \frac{df}{dz} \tag{2.2}$$

where $z = \Sigma_i x_i w_i$. Then the desired weight update according to gradient descent is,

$$\Delta w_i = -\eta \frac{dE}{dw_i} = \eta . x_i . \delta \tag{2.3}$$

where η is the learning rate and $\delta = -\frac{dg}{dy} . \frac{df}{dz}$ is the error in the net input to the neurons. In a multi-layered neural network, the weight updates for any layer could be similarly computed using the chain rule of differentiation. The training algorithm is typically called backpropagation as the gradient with respect to preceding weight layers could be computed by propagating the error δ in one layer through the weight layers in the backward direction [74].

2.4.2 Spiking neural network

Spiking neural networks (SNNs) is a closer abstraction to biological neural networks than ANNs. They attempt to incorporate the complex dynamics of the biological neurons and spike domain data encoding. The first complete biologically plausible model of the spiking neuron was originally developed by Hodgkin and Huxley, which incorporates the detailed dynamics of the membrane potential and the Na, K and leak ion channels in a set of four coupled differential equations [75]. However, this model is not suitable or necessary for engineering applications, and several simplified models have been proposed. The second order model proposed by Izhikevich [76] and the adaptive exponential integrate and fire model (AEIF) proposed by Brette and Gerstner [77] are sufficiently rich to capture most of the spiking dynamics observed in biological neurons. We use a computationally simpler leaky integrate and fire (LIF) model [78] for our training purposes. The LIF model represents the potential of a neuron as the voltage across a capacitor connected in parallel with a leaky conductance path and is charged by incoming input currents. The membrane potential $V(t)$ evolves according to the differential equation:

$$C\frac{dV(t)}{dt} = -g_L(V(t) - E_L) + I_{syn}(t). \tag{2.4}$$

When $V(t)$ exceeds a threshold V_T, a spike is issued and transmitted to the downstream synapses; the membrane potential is reset to its resting value E_L after the spike. C and g_L model the membrane's capacitance and leak conductance, respectively. Biological neurons enter a refractory period immediately after a spike is issued during which another spike cannot be issued. This can be implemented by holding the membrane potential at $V(t) = E_L$ for a short refractory period t_{ref} after the issue of a spike.

Neurons are interconnected via synapses as before, but only spikes from a neuron induce an input (typically modeled as a current) to a post-synaptic neuron. The post-synaptic neuron integrates currents from all its pre-synaptic inputs causing its membrane potential to rise above a threshold and issue a spike. The synaptic conductance can be modulated to adjust the spike rate or time. Information could be encoded in the precise spike time, or in the spike rate, or in the phase of a spike with respect to a reference, or the ensemble average spike response of a group of neurons.

The spike discontinuity in the neuronal non-linearity makes direct application of gradient descent based weight updates rules impossible in SNNs. Several approximations have been proposed for the derivative of the spikes leading to approximate versions of back-propagation to train SNNs [79, 80]. We discuss Normalized Approximate Descent (NormAD) which will also be used for the supervised training experiment in Chapter 6 [79].

The Figure 2.4b demonstrate the main parameters in NormAD training algorithm. $c_i(t)$ is obtained by convolving the input spike trains with post-synaptic current kernel, $\alpha(t)(= [exp(-t/\tau_1) - exp(-t/\tau_2)]u(t))$ where $u(t)$ is the Heaviside step function and $\tau_1 = 5\,\text{ms}$ and $\tau_2 = 1.25\,\text{ms}$. This double exponential approximate the opening and closing of ion channels in the post-synaptic receptor neuron. $c_i(t)$ is convolved with $h(t)$ to obtain $d_i(t)$, where $h(t) = \frac{1}{C_m} exp(-t/\tau_L)u(t)$ is the impulse response of integration in the LIF neuron. The weighted sum of resulting $d_i(t)$ vector undergo a thresholding to determine the spike response of the neuron. This spike pattern is compared with the desired one and the following weight update rule is applied whenever there is a mismatch between these two spike streams.

$$W \leftarrow W + \eta.e(t).\frac{\hat{d}(t)}{\parallel \hat{d}(t) \parallel} \tag{2.5}$$

where, η is the learning rate, $e(t)$ is the difference between the desired and observed spike trains, and $\hat{d}(t)$ is obtained by convolving $c_i(t)$ with $\hat{h}(t) = \frac{1}{C_m} exp(-t/\tau_{L'})u(t)$ where $\tau_{L'} < \tau_L$. The algorithm is capable of successfully training single layer SNNs to produce spikes at desired time instances. Note that the ΔW in equation (2.5) is again a product of the input to the weight layer (Figure 2.4b) and the error at its output, similar to ANNs.

2.4.3 Precision requirement

From both the ANN and the SNN weight update rules, we see that the desired weight change is proportional to the product of input to the synapse and error at the post-synaptic neuronal input. However, this only gives the direction of updates and the relative proportion of weight changes. The absolute value of the update is determined by the learning rate which is a hyper-parameter set arbitrarily and tuned based on training performance. Typically, the learning rates chosen in successfully trained neural networks are such that the ratio $\Delta W/W$ is $< 10^{-3}$. The choice of learning rate is critical in attaining the highest accuracy possible by the chosen network architecture in the smallest amount of time. The learning rate should be large enough for fast convergence and should be small enough not to overshoot the minima in the error surface which the gradient descent is traversing. Adaptive gradient descent rules such as AdaGrad [81], Adadelta [82], and Adam [83] have been proposed essentially to have a learning rate specific to individual weights based on its history of updates. These rules have enabled networks to achieve higher accuracies at a faster rate for supervised learning tasks. Hence, it is clear that neural network weights have to be updated with high precision in order to achieve high performance.

The computational step that determines the precision for the floating point is the weight update accumulation. In floating point arithmetic, two numbers are added by making their exponents equal by shifting their fractional parts and then they are added together using standard binary addition. When the ratio between two numbers being added is $> 10^3$ the smaller number is shifted right relative to the larger by ≥ 10 bits ($2^{10} = 1024$). In such cases, even 16-bit floating point representation becomes inadequate as it has only 10-bits for its fractional part. In the case of our analog memories whose weight update resolution is less than < 5 bits, the method we adopt to transfer the weight updates to the devices determine our ability to achieve high performing learning machines using such nanoscale devices.

2.5 Summary

In this chapter, we reviewed the physical mechanism of major nonvolatile memory devices and their conductance update behavior necessary for implementing on-chip training. We also reviewed supervised learning algorithms for ANNs and SNNs that will be used to train nonvolatile memory based learning systems. We observed that high-precision requirement of the gradient descent based training is mainly originating from the need to make fine weight updates, which is also the major challenge posed by the nanoscale devices due to their limited conductance update precision.

CHAPTER 3

PHASE CHANGE MEMORY MODEL

3.1 Introduction

We now describe the basic physics of nanoscale phase change memory (PCM) devices that were used in our studies, and a stochastic conduction model that we developed, as reported in [84].

Phase-change memory (PCM) is arguably the most advanced emerging non-volatile memory technology [32]. PCM is based on the property of certain materials such as $Ge_2Sb_2Te_5$ that exhibit a significant difference in resistivity depending on whether they are in the ordered crystalline phase or the disordered amorphous phase. In a PCM device, a tiny volume of such a material is sandwiched between two metal electrodes. A typical device structure is shown in the cross-sectional TEM image in Figure 3.1a. By the application of suitable electrical pulses and subsequent Joule heating, it is possible to reversibly alter the phase-configuration of the material within the device. Pulses that result in an increase in the size of the amorphous region are typically referred to as RESET pulses. In this case, the application of the pulse results in the melting of a critical volume of the material and which is then rapidly quenched to induce glass transition. The pulses that reduce the size of the amorphous region are referred to as SET pulses. Here, the temperature reached within the device is favorable for crystallization (see Figure 3.1b) [85]. Typically, the SET pulses that induce partial crystallization of the material is referred to as partial-SET pulses and all these pulses are collectively referred to as programming pulses.

The electrical resistance/conductance of the device will depend on the resulting phase-configuration. In fact, it is possible to achieve a continuum of resistance values in a single device and this can be exploited for neuromorphic applications. For

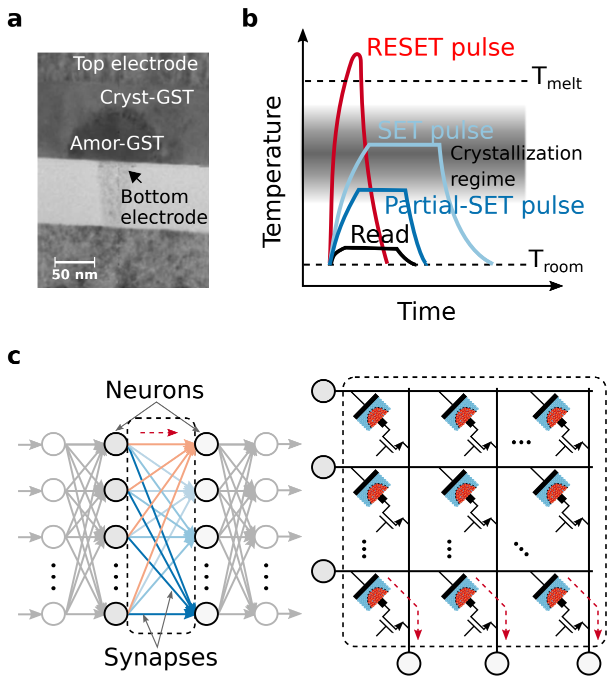


Figure 3.1 **a** TEM image of a mushroom-type PCM device. Amorphous dome (Amor-GST) inside the crystalline (Cryst-GST) dielectric is visible **b** Pictorial representation of the programming pulses and the resulting relative temperature for RESET, SET, partial SET and read operation in PCM **c** Schematic illustration of the application of PCM devices as synaptic elements in neuromorphic computing. A crossbar array of PCM devices could be used to represent the connection strengths in a neural network layer.
Source: [84].

example, as shown in Figure 3.1c, PCM devices organized in a cross-bar configuration can be used to emulate the synaptic elements in an artificial neural network [28, 86]. The synaptic weights are captured by the conductance values of the PCM devices. The inputs from one layer of neurons are weighted by these conductance values (via Ohm's law) and the resulting current along the columns serve as inputs to the next layer of neurons. During the training of a neural network, the initial conductance values are typically chosen randomly, which is then modified (synaptic plasticity) via some appropriate learning rule. The programming pulses can be used to alter the conductance values during the training process. Unlike RESET pulses, which cause in an abrupt transition to lower conductance values, successive application of a partial SET pulse results in a more progressive increase in the conductance value. This cumulative evolution of conductance is highly beneficial for neuromorphic applications. Hence, often in PCM, only the partial SET pulses are used to implement synaptic plasticity rules [86]. To avoid the use of RESET pulses, PCM devices are organized in a differential configuration [27]. A comprehensive understanding of this accumulative behavior across a large number of devices is central to the realization of large-scale neural networks. Besides crystallization, there are other structural dynamics at play in PCM devices. These devices exhibit a temporal evolution of conductance values after the application of each programming pulse. This is attributed to a spontaneous structural relaxation of the material [87] and could also play a key role in neuromorphic computing.

In this chapter, we present a comprehensive model of PCM devices that capture the accumulative behavior, conductance drift and read noise. Extensive experimental characterization of 10,000 PCM devices has been performed to develop this statistical model. Finally, we demonstrate the efficacy of this model by using it to match experimentally observed array level characteristics and to train spiking and non-spiking artificial neural networks.

3.2 Device Characterization and Modeling

For device characterization, we used mushroom-type PCM devices fabricated in the 90 nm technology node [88]. The phase-change material is doped $Ge_2Sb_2Te_5$ (GST). A prototype chip comprising 3 million devices was used in the study [89]. Individual devices are addressed via word lines and bit lines and the devices have access transistors in series. The devices are programmed using current pulses of designated amplitude and width generated in the peripheral circuits. The conductances are read by applying a 0.3 V read pulse and the resulting current is read using an 8-bit ADC. The ADC is calibrated to span a conductance range between $0.1\,\mu$S and $27\,\mu$S.

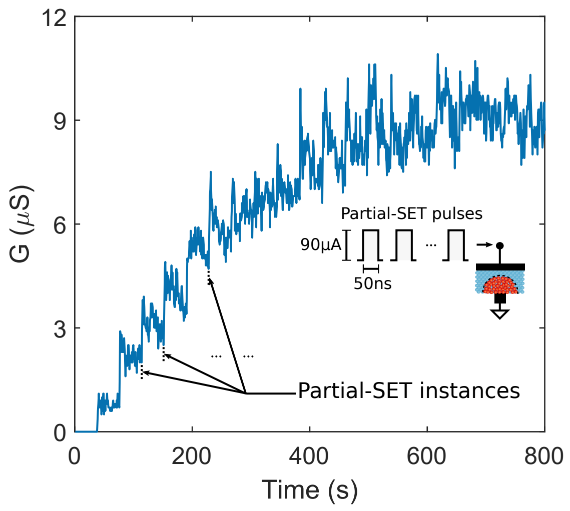


Figure 3.2 The measured conductance evolution in a single device in accordance with the application of 20 consecutive partial SET pulses. The time instances of programming are indicated. After the application of each pulse, the device conductance is measured 50 times.
Source: [84].

First, the device conductances were initialized to a distribution close to 0.1 μS using iterative programming [90]. Subsequently, we applied 20 partial SET pulses of 90 μA amplitude and 50 ns duration. After the application of each pulse, devices are read 50 times. In addition, an immediate conductance measurement is performed approximately 100 ns after the programming pulse. However, subsequent measurements are obtained at time intervals in the order of seconds. As a result, consecutive programming pulses were applied with an average interval of 38.6 s for the 10,000 devices. The resulting conductance evolution, except for the immediate read after 100 ns, for one such representative device is shown in Figure 3.2. In subsequent sections, we will use all the measurements from the 10,000 devices (barring a few whose conductances where outside the ADC limits) to develop the statistical model.

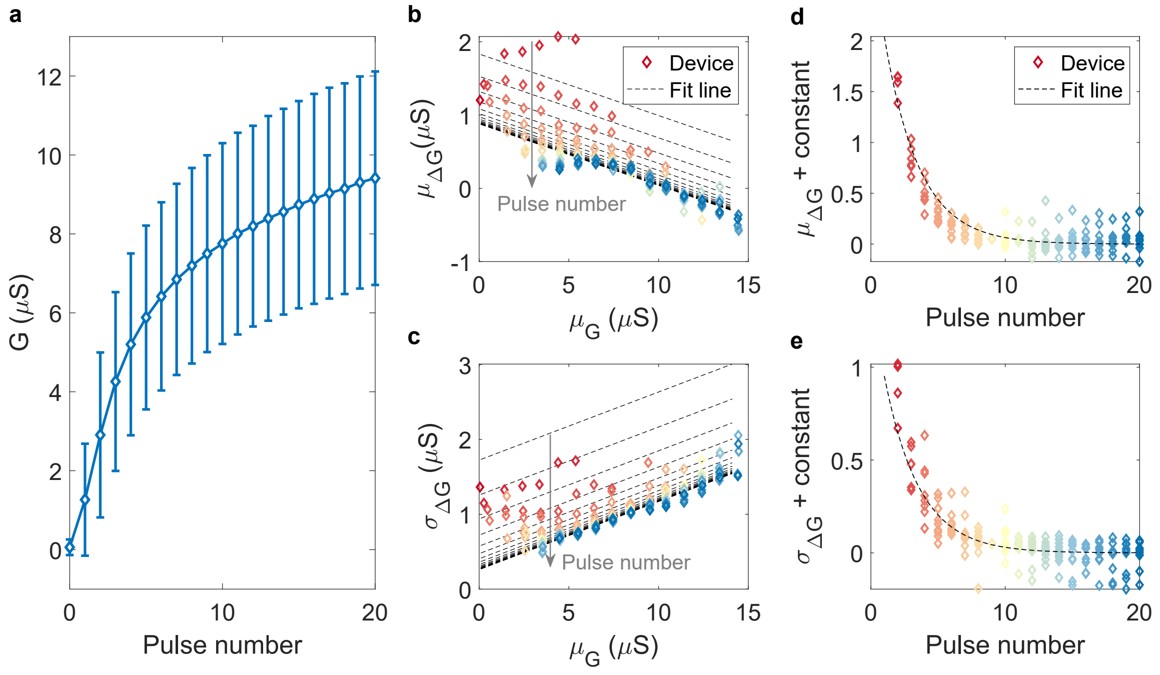


Figure 3.3 **a** The statistics of cumulative conductance evolution as a function of the number of partial SET pulses. The error bars indicate one standard deviation. **b** The mean $\mu_{\Delta G}$ and **c** standard deviation $\sigma_{\Delta G}$ of conductance change as a function of the average initial conductance μ_G for each programming pulse. The initial conductance distribution for each programming pulse is divided into smaller intervals and μ_G, $\mu_{\Delta G}$ and $\sigma_{\Delta G}$ is determined separately for each interval. Each data point in **b** and **c** corresponds to an average of measurements from at least 100 devices. Also depicted are the fit lines used to obtain the model parameters. **d**, **e** The same data points of $\mu_{\Delta G}$ and $\sigma_{\Delta G}$ are plotted as a function of the pulse number with a constant added for data points corresponding to a single μ_G interval. The dependency of $\mu_{\Delta G}$ and $\sigma_{\Delta G}$ on pulse number is approximated using an exponential function with a decay constant of 2.6.
Source: [84].

3.2.1 Accumulative behavior

First, we characterized the accumulative behavior arising from the successive application of partial SET pulses. To decouple the accumulative behavior from conductance drift, the 50th read measurement was used. The distribution of the conductance values as a function of the pulse number is shown in Figure 3.3a. It can be seen that the average conductance change is high at low conductance values and it gradually reduces as the conductance values increase. It can also be seen that there is significant randomness associated with the conductance values. This is mostly attributed to the inherent randomness associated with the crystallization process [91, 92]. In fact, the inter- and intra- device variability in the array has been observed to be of comparable magnitude [93, 94, 95].

To obtain a quantitative description of this behavior, we studied how the conductance change arising from the application of a single SET pulse depends on the conductance state of the device prior to the application of the pulse as well as the device's programming history. The devices were split into different groups based on their conductance values. Each group corresponds to a conductance interval of 1 μS. For each group, the mean ($\mu_{\Delta G}$) and standard deviation ($\sigma_{\Delta G}$) of the conductance change due to the application of a single programming pulse is plotted against the mean conductance (μ_G) of each group (see Figure 3.3b and 3.3c). The data points are generated only for those groups with 100 or more devices. This is repeated for the conductance values measured after the application of each programming pulse. In Figure 3.3b and 3.3c, each color corresponds to a single programming pulse with the red color indicating the first pulse and the blue color the 20^{th} pulse. We observe that there is a negative correlation between the $\mu_{\Delta G}$ and μ_G that suggests a linear decrease in the conductance change as the device conductance increases. In addition, in a particular conductance range, the conductance change observed seem to decrease with increasing number of applied pulses. This behavior can be captured using a linear fit of a negative slope to map the relation between $\mu_{\Delta G}$ and μ_G for any particular pulse number. Further, the dependency on the pulse number is encoded in the y intercept of this linear fit. It can be seen that for any given conductance value, the extent of conductance change induced by a single partial SET pulse reduces significantly with increasing number of applied pulses. This could be captured using an exponential empirical relation (Figure 3.3d and e).

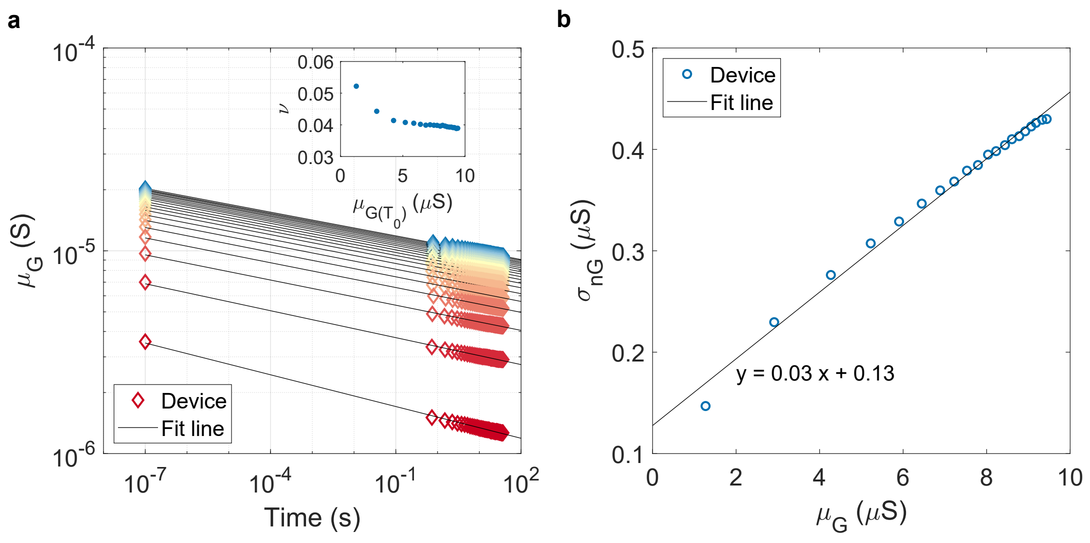


Figure 3.4 **a** The average conductance evolution after each programming pulse obtained by the 50 read operation is fitted using Equation 3.3. The estimated drift-coefficient, ν, is shown in the inset **b** The read noise measured from the device array is plotted as a function of the average device conductance. The linear fit used to estimate read noise for the model in a state-dependent manner is also shown.
Source: [84].

It can be seen that the behavior of $\sigma_{\Delta G}$ is also very similar to that of $\mu_{\Delta G}$ except that there is a positive correlation with the μ_G in this case. Therefore, the mean and standard deviation of the ΔG is modeled respectively using lines of negative and positive slopes and with an intercept which is an exponential function of the pulse number (p) as in the following equations (also in Figure 3.3b and c):

$$\mu_{\Delta G} = m_1 G + (c_1 + A_1 e^{-p/\alpha}) \tag{3.1}$$

$$\sigma_{\Delta G} = m_2 G + (c_2 + A_2 e^{-p/\alpha}) \tag{3.2}$$

where the fit parameters m_1, m_2, c_1, c_2, A_1, A_2, and α are -0.084, 0.091, 0.880, 0.260, 1.40, 2.15, and 2.6, respectively.

3.2.2 Conductance drift and read noise

In this section, we model the conductance drift in the devices arising from structural relaxation. For this, we use the 50 read measurements obtained after the application of each SET pulse. The mean conductance evolution after each programming event as a function of time is plotted in Figure 3.4 a. The response is fitted using the model [35, 96],

$$G(t) = G(T_0)\left(\frac{t}{T_0}\right)^{-\nu} \tag{3.3}$$

According to Equation 3.3, if the device conductance, $G(T_0)$ is known at time T_0 after programming, the conductance at any time t can be estimated with the knowledge of the drift coefficient, ν. The estimated ν from the fit lines have a mean value of 0.04 (Figure 3.4a inset). Note that the logarithmic dependence on time suggests that after programming, the conductance drift slows down with time. We observe that the partial SET pulses result in a state that drifts, with a drift coefficient that decreases with increasing conductance $\mu_G(T_0)$. The application of a partial SET pulse re-initiates structural relaxation and conductance drift. Hence, we speculate that each partial SET pulse creates a new unstable glass state because of the atomic rearrangement that occurs upon its application, which then structurally relaxes to an energetically more favorable amorphous state [97, 87].

In addition to the conductance drift, there are also significant fluctuations in the conductance values (read noise) mostly arising from the $1/f$ noise exhibited by amorphous phase-change materials [98]. To model this, we estimated the noise from the last ten reads from the fifty read measurements. The objective was to decouple the read noise from the conductance drift. The standard deviation of the zero-mean read noise is plotted as a function of the mean conductance (see Figure 3.4b). It can be seen that the read noise increases with the device conductance. The read noise standard deviation, σ_{nG}, for the device conductance range is fitted using the linear relation,

$$\sigma_{nG} = m_3 G + c_3 \tag{3.4}$$

where $m_3 = 0.03$ and $c_3 = 0.13$.

3.2.3 Overall model description and validation

In this section, we combine the various elements of the model describing the accumulative behavior, conductance drift, and read noise to generate a complete statistical model and validate it based on the experimental data. The objective is to capture the evolution of conductance values for a large collection of devices after a certain time T_0 after programming with an arbitrary number of partial SET pulses. More specifically, we would like to determine the device conductance $G(t)$ at any time t, which has been initialized to approx. $0.1\,\mu$S, and is subjected to a sequence of $90\,\mu$A, 50 ns programming pulses with arbitrary time intervals between them.

Table 3.1 PCM Model Parameters

Symbol	Value	Symbol	Value	Symbol	Value
m_1	-0.084	c_1	0.880	A_1	1.40
m_2	0.091	c_2	0.260	A_2	2.15
α	2.6	T_0	38.6 s	ν	0.04
m_3	0.03	c_3	0.13		

To simulate this, three quantities are recorded per device: (a) $G_i(T_0)$, the conductance after T_0 time after the i^{th} programming pulse for $i = 0, 1, 2\ldots$, (b) P_{mem}, a quantity that captures the programming history, and (c) t_p, the time of the last programming event. t_p is initialized to zero. Based on the chosen initial conductance value $G_0(T_0)$, P_{mem} is initialized to $P_{mem,0} = e^{-p_0/\alpha}$, where p_0 is the effective number of pulses applied to reach the initial conductance $G_0(T_0)$. p_0 is zero for initialization around $0.1\,\mu S$ and p_0 for higher values of conductance is determined from the average conductance evolution curve shown in Figure 3.3a. The effective number of pulses versus conductance can be approximated empirically as,

$$p_0 = 0.027\,\mu_G^3 - 0.15\,\mu_G^2 + 0.81\,\mu_G \tag{3.5}$$

for conductance ranging from $0.1\,\mu$S to around $8\,\mu$S. After initialization, for the N^{th} programming event, P_{mem} is first updated as $P_{mem,N} = P_{mem,N-1}e^{-1/\alpha}$ for $N = 1, 2, \ldots$. Then $G(t)$, which has seen N programming pulses can be determined as follows:

$$\mu_{\Delta G_N} = m_1 G_{N-1}(T_0) + (c_1 + A_1 P_{mem}) \tag{3.6}$$

$$\sigma_{\Delta G_N} = m_2 G_{N-1}(T_0) + (c_2 + A_2 P_{mem}) \tag{3.7}$$

$$\Delta G_N = \mu_{\Delta G_N} + \sigma_{\Delta G_N}\chi \tag{3.8}$$

$$G_N(T_0) = G_{N-1}(T_0) + \Delta G_N \tag{3.9}$$

$$G(t) = G_N(T_0)\left(\frac{t - t_p}{T_0}\right)^{-\nu} + n_G \tag{3.10}$$

Here, χ represents a Gaussian random number of mean zero and variance 1. Another Gaussian random variable with mean zero, n_G, captures the conductance fluctuations arising from PCM noise, whose standard deviation is calculated based on the instantaneous conductance state as dictated by the linear fit in Equation 3.4 (also in Figure 3.4b). All the model parameters are listed in Table 3.1. Please note that the conductance values predicted by the model are in micro-Siemens.

First, the model is used to validate the same experimental data that was used to generate the model parameters. In particular, the model is used to generate the distribution of conductance values as a function of the number of programming pulses. As shown in Figure 3.5, the mean and variance match remarkably well with experimental data. It can also be seen that the distributions themselves are remarkably similar. Figure 3.6a-c show the conductance distribution from the 50^{th} read, after initialization, after the application of five programming pulses, and after

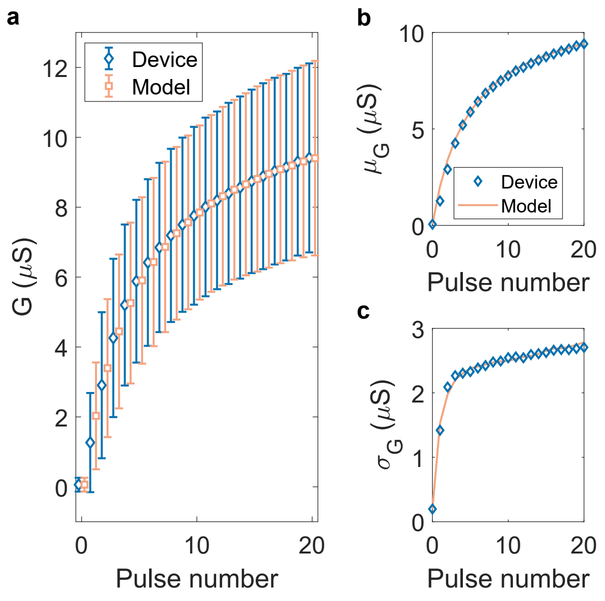


Figure 3.5 **a** The distribution of conductance values obtained using the model as a function of the number of partial SET pulses and match with experiments. **b** The mean of the conductance as a function of the pulse number. **c** The standard deviation of the conductance as a function of the pulse number.
Source: [84].

the application of 20 programming pulses, respectively. The model also matches the correlation coefficient observed between G and ΔG for the pulses applied. From Figure 3.6e, f it can be seen that the statistical model also captures the individual device behavior remarkably well.

Additional measurements were performed where the devices are programmed with 20 programming pulses, however, with varying time intervals between the application of each pulse. The time interval was determined based on the number of reads performed and in the current experiment, each read process took approximately 1 s for the 10,000 devices. Figure 3.7 shows the programming events in time (top) and the resulting evolution of the mean conductance of the 10,000 devices (bottom). The spikes in the programming event plot correspond to the application of partial SET pulse and the device conductances are read at all other time instances. As

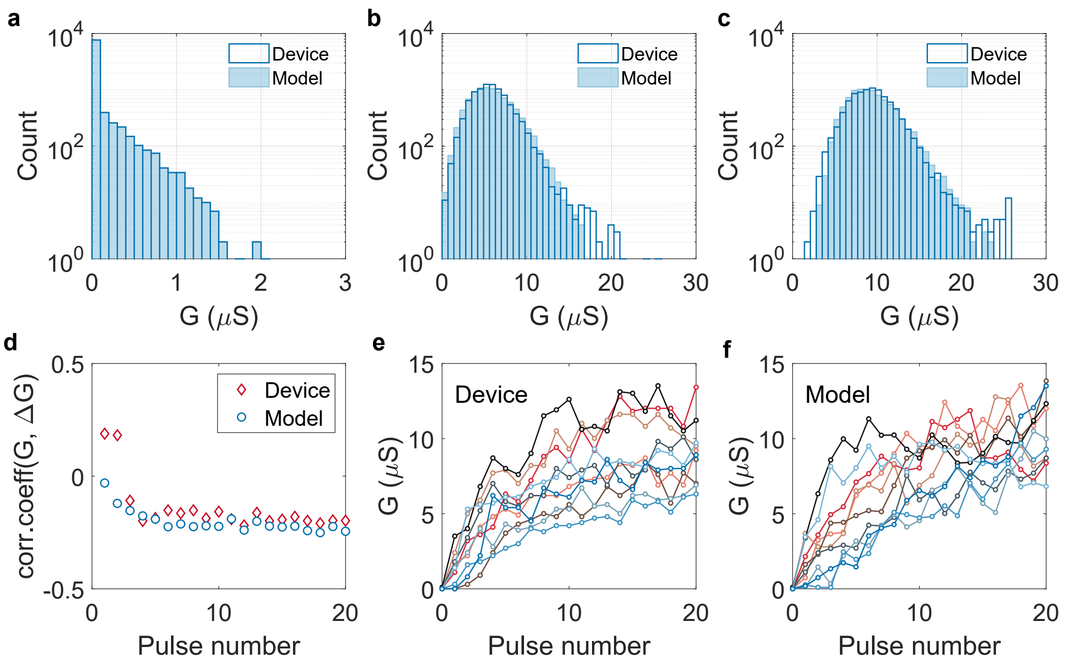


Figure 3.6 The distribution of conductance values after **a** initialization **b** the application of 5 programming pulses and **c** the application of 20 programming pulses. It can be seen that there is a remarkable agreement between the experimental distribution and that predicted by the model. **d** The correlation coefficient between the G and ΔG after the application of each programming pulse calculated based on the model and is compared with the experimental measurement. The conductance evolution of individual devices as measured experimentally **e** and as predicted by the model **f**.
Source: [84].

discussed earlier, it can be seen that with the application of each programming pulse, the drift process is re-initiated. Another interesting observation is that the net change in conductance seems to be independent of structural relaxation. There is some evidence that structural relaxation slows down crystal growth rate [85]. But at least in these devices and these time scales, this does not seem to be significant. The final conductance values at the end of programming seem to converge to similar conductance levels independent of the rate of programming. Hence, our proposed model is able to capture this behavior remarkably well with the additional incorporation of Equation 3.3.

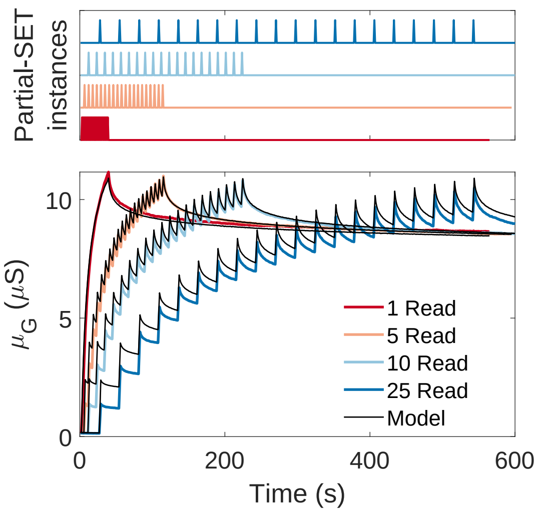


Figure 3.7 The 10,000 PCM devices are programmed using sequences of 20 current pulses of 90 μA and 50 ns width. The number of read operations performed after each programming event is varied resulting in different time intervals between the programming events. The resulting conductance evolution during all the read operations is illustrated. The proposed model captures the experimentally observed behavior remarkably well.
Source: [84].

3.2.4 Generalizability of the PCM model

The model we presented here is based on doped $Ge_2Sb_2Te_5$ in a mushroom structure fabricated in 90 nm technology. This model is largely data-driven and is not based on specific material properties. The key aspects of the model are the negative correlation of the $\mu_{\Delta G}$ and positive correlation of the $\sigma_{\Delta G}$ with the average current state μ_G (Figure 3.3 b, c). An intuitive explanation for these observations is as follows. Phase change memory devices have a chalcogenide sandwiched between a top electrode and a bottom electrode, with one of the contacts making a narrow contact (in both mushroom and pore PCM structures) with the dielectric acting as a point of heating. The temperature distribution within a PCM cell is decided by many factors such as thermal and electrical resistance and the specific heat capacity of the cell materials. Typically, the temperature distribution along the vertical symmetry axis is a skewed parabola with the maximum temperature located slightly above the heating contact in properly designed devices [99, 100]. Further, the crystal growth velocity in PCM increases monotonically until it reaches a peak crystallization temperature [85]. Though this peak temperature may depend on the material, in partial SET pulse driven gradual conductance change operation, the programming pulses are chosen to operate below this temperature. Therefore, when a device initialized with a RESET operation is subjected to partial SET pulses, the point of maximum crystal growth will be around the point of maximum temperature. This results in large conductance change for the first few pulses. For further programming pulses of the same amplitude, assuming that the temperature distribution remains more or less unchanged and that the crystalline-amorphous boundary has moved to lower temperature regions due to earlier crystal growth, every subsequent programming will result in smaller conductance increase. This conductance saturation is captured in the model by the point where $\mu_{\Delta G}$ versus μ_G crosses zero. This zero-crossing behavior also makes the model bounded in its conductance range, even when simulated with a large sequence of pulses. The increase in the programming noise at higher conductance states may be attributed to the higher variability in the number of trap-states within the reduced volume of the amorphous region as sub-threshold conduction in these devices are trap-mediated [101].

The point of conductance saturation could be further extended if it is programmed with larger current pulse amplitudes. The resulting temperature profile will have a higher peak, and it will cause a positive conductance update at the current saturation point driving crystal growth creating a new saturation point at a higher conductance. To test this argument, we applied 20 pulses of 50 ns width and 60 μA amplitude to 10,000 PCM devices. This is followed by fifteen pulses each of 80 μA, 100 μA, and 120 μA. We observed that each of the higher pulse amplitude extended the conductance range. The conductance evolution and the state-dependent nature

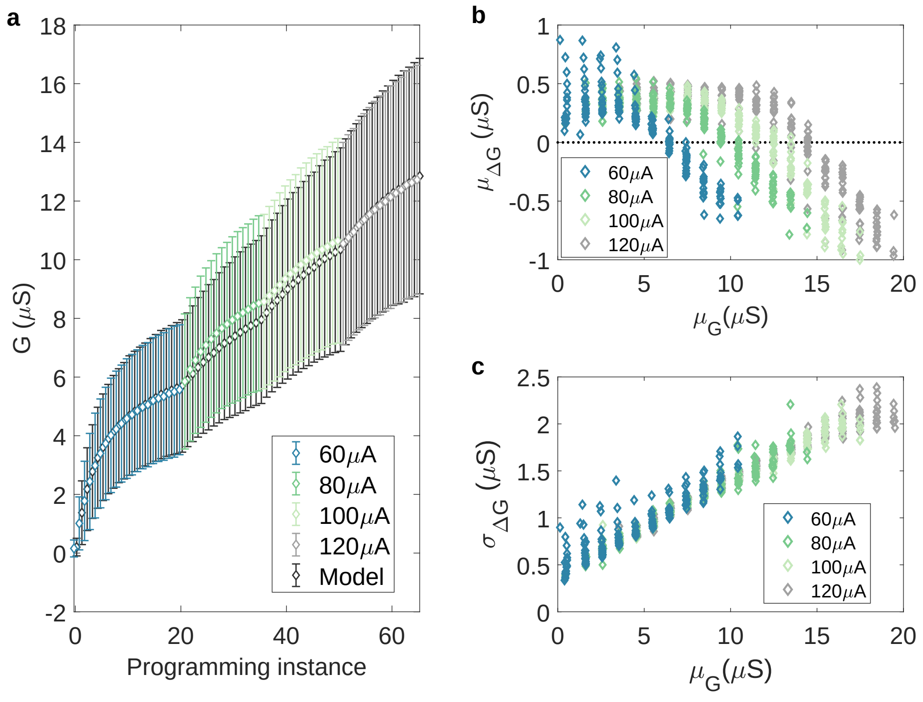


Figure 3.8 **a** Average conductance evolution of 10,000 devices subjected to a sequence of twenty 50 ns, 60 μA pulses followed by fifteen pulses each of 80 μA, 100 μA, and 120 μA amplitude. The error bars show one standard deviation. The corresponding model response fitted to the device is shown overlapped. **b** Mean of the conductance change versus the average initial conductance state. The data points corresponding to different programming amplitudes are color coded. **c** The standard deviation of the conductance change versus the average initial conductance state.

of the conductance change ΔG is shown in Figure 3.8. The $\mu_{\Delta G}$ versus μ_G plot shows a similar trend for different pulse amplitudes and the zero crossing point is increasing with an increasing programming current. To capture this behavior in the model we modified the Equation 3.1 so that its y-intercept is a linear function of the pulse amplitude in addition to its exponential dependence on pulse number. On the other hand, the $\sigma_{\Delta G}$ did not show a dependence on the pulse amplitude. The modified model response is shown in Figure 3.8 a, which shows good agreement with the measured device behavior.

While $Ge_2Sb_2Te_5$ is the most commonly used material in PCM devices, other chalcogenide alloys have been explored for improved properties such as faster crystallization, reduced drift and lower programming currents [102, 103]. However, the qualitative description of the temperature distribution and the modeling approach we presented here is expected to remain valid in different phase change material systems provided the conductance modulation is driven by Joule heating and have similar crystal growth dynamics. For example, comparable partial SET conductance accumulation behavior has been reported in GeTe based PCMs [104]. Hence, the model we presented could be tuned to capture the gradual crystallization behavior if sufficient statistics on device characteristic are available. Phase change memory devices have also been demonstrated to be scalable via ab-initio simulations [105, 106] and experiments [107]. The temperature profile within the scaled devices remain more or less the same under constant voltage scaling [108], and hence similar state-dependent conductance modulation behavior under partial SET programming pulses could be expected in them as well. However, the scaling of the electrode contact area reduces the amorphous volume involved in the conductance modulation, which could result in reduced granularity and higher stochasticity. For a given trap density, the changes in this smaller amorphous region could lead to higher programming noise. The ability of a PCM cell to provide gradual conductance state will also depend on the cell design. For example, in a mushroom cell, if the peak temperature point is too far away from the heater electrode, the amorphous region will not cover the heater unless very high powers are applied, effectively making multi-level operation almost impossible [99]. Therefore, the model could possibly be tuned to adapt to different phase change materials and technology nodes with sufficient data, provided the devices are not binary and have state-dependent gradual conductance change.

3.3 Spiking Neural Network with Modeled PCM-synapses

The developed model could be used to simulate the training behavior of neural networks and other possible learning systems which require adaptive weights. To

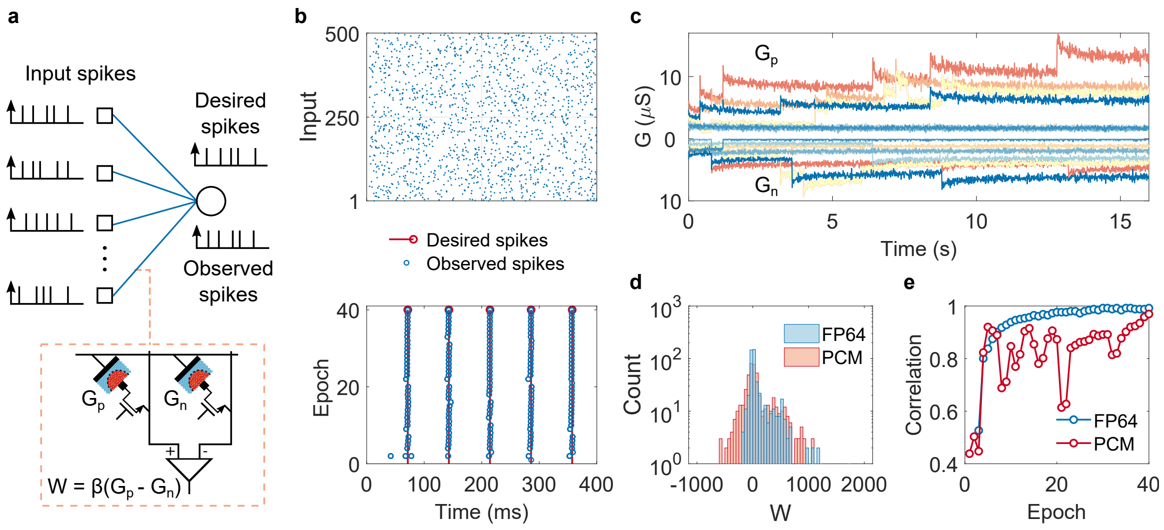


Figure 3.9 **a** The spiking neural network used for the training simulation. Two PCM devices in the differential configuration used for the synapse is also shown. **b** Raster plot of input spike streams (top) and the desired spike streams from the output neuron (bottom). The observed spikes during each training epoch are plotted over the desired spike trains. The observed spikes are within 0.7 ms of the desired instances after 40 epochs of training. **c** Conductance evolution of a few synapses during the training illustrating the drift and read noise of PCM devices **d** The final weight distribution from the PCM synapses along with the weights from a floating point (FP-64 bit) training is shown for reference. **e** The correlation between the desired and observed spike trains after 40 epochs with the PCM synapse is similar to that using floating point synapses.
Source: [84].

illustrate this, we train a spiking neural network (SNN) and a non-spiking artificial neural network (ANN) with PCM based synapses whose conductance modulations are emulated by the model and discuss the effect of device behavior such as limited granularity, stochasticity in the training. The PCM devices are assumed to be arranged in a crossbar array representing the connection strength between adjacent layers of neurons as in Figure 3.1c. The crossbar arrangement enables them to perform the weighted summation necessary for the dataflow through the network in constant time irrespective of the layer size.

SNNs are third generation neural networks that attempt to mimic biological neural network behavior. Biological neurons integrate its input over time in the analog domain, while communicating with other neurons via spikes, enabling highly energy efficient signal encoding and processing. In SNNs, this neuronal behavior is typically emulated using a leaky-integrate and fire (LIF) model. The LIF neuron could be represented as a leaky capacitor that integrates incoming currents, with the integration reset when the voltage across the capacitor exceeds a threshold, and a spike is sent to the downstream neurons. This continuous time behavior makes the training and inference of SNNs in conventional digital hardware extremely inefficient. Non-volatile memory array based synapse networks with dedicated neuronal circuits at the periphery could potentially provide a more efficient non-von Neumann architecture for SNN implementation and training.

An example of SNN is shown in Figure 3.9a. It has one LIF output neuron receiving 500 spikes streams via input synapses. The network is expected to generate a desired spike pattern from the inputs which is generated here from a Poisson random process for illustrative purposes. The network spike input and the desired output spike response is illustrated in Figure 3.9b. In response to each spike input, synapses generates currents modeled by the expression $I_{syn} = W \times (e^{-t/\tau_1} - e^{-t/\tau_2})$ as a function of time, t, where W is the synaptic strength. The task of the supervised learning algorithm is to adjust the weights such that the observed spikes match a desired pattern. One weight adjustment rule for SNNs that has been demonstrated recently is the NormAD algorithm [79] which provides the network weight updates ΔW as

$$\Delta W = r \int_0^T e(t) \frac{\hat{d}(t)}{||\hat{d}(t)||} dt \tag{3.11}$$

where r is the learning rate, T is the duration of the training pattern, $e(t)$ is the difference between the desired and observed output spike trains. $\hat{d}(t)$ is obtained by convolving the synaptic current (I_{syn}) with an approximate impulse response of the LIF circuit.

The network synapses are realized using the PCM model. Because of the abrupt RESET behavior of the device, each synapse is realized using two PCM devices G_p and G_n in differential configuration such that $W = \beta(G_p - G_n)$, where β is a scaling factor [27]. Hence, synaptic potentiation is achieved by applying a partial SET pulse to G_p and depression is achieved by applying a partial SET pulse to G_n. This unidirectional programming often causes the device pairs to saturate preventing any further weight update. Therefore, an occasional weight refresh is performed based on the following criteria. If G_p or $G_n > G_x$, and $|G_p - G_n| < 0.25\,G_x$, where G_x is a threshold, both the devices are RESET and the conductance difference is programmed to the device which had the higher conductance. For the hardware implementation, the RESET pulse shape could be determined from the PCM programming curve [90, 85, 12]. Here, the stochastic RESET behavior is simulated using an abrupt conductance transition to a distribution of mean $1\,\mu$S and standard deviation $0.5\,\mu$S.

For the training, the device conductances are initialized to a distribution of mean $2\,\mu$S and a standard deviation $0.5\,\mu$S. During each training epoch, the 400 ms long spike sequences are presented to the network and the weight updates are computed. The scaling factor β is chosen to match the PCM based weight distribution to that obtained from an equivalent network trained with floating-point synaptic weights (Figure 3.9 d). While this scaling enables PCM based synapses to represent the desired weight range, the achievable weight updates are limited by the device granularity. Further, we assume the states of the individual devices are unknown to determine the optimum programming pulse. Hence, the estimated weight updates are converted to programming pulses by assuming an average conductance change of $0.75\,\mu$S for each partial SET pulse (as the $7.5\,\mu$S conductance range used in our study is typically reached within 10 pulses). The resulting conductance evolutions during training for a few synapses are shown in Figure 3.9 c. As we see here, the PCM synapses drift and have read noise while computing the weight updates. The conductance programming is without any read-verify operation and is stochastic, which will simplify the system implementation and accelerate the training process. The training performance is evaluated based on a correlation between the desired and observed spike trains [79] and is plotted in Figure 3.9 e. The corresponding numbers from training which used floating point synapses are shown for reference. In spite of the stochastic nature of PCM weight updates and conductance drift after programming, the SNN incorporating these devices exhibit training performance that is at par with the baseline software network.

3.4 Deep Learning with Modeled PCM-synapses

Now we discuss the training of an artificial neural network (ANN) whose synapses are realized using the PCM differential configuration. The network is designed for the benchmark handwritten digit recognition task, based on 28×28 gray-scale images from the MNIST dataset. The dataset has 60,000 training images and 10,000 test images. In this exercise, we attempt to modify the standard backpropagation training algorithm to account for the limited device granularity and its effect is analyzed.

The network used for the task is shown in Figure 3.10 a, which has two fully connected weight layers. The input layer neurons are linear, the hidden layer and the output layer neurons perform a logistic function (sigmoid) on their inputs. The training is performed using the back-propagation algorithm, an adaptation of gradient descent for multi-layer ANNs. The first stage of training, known as forward propagation involves presenting the image pixels at the input layer and determining the output layer response. Out of the ten output neurons, the one corresponding to the input image is expected to have the highest neuron activation. The actual

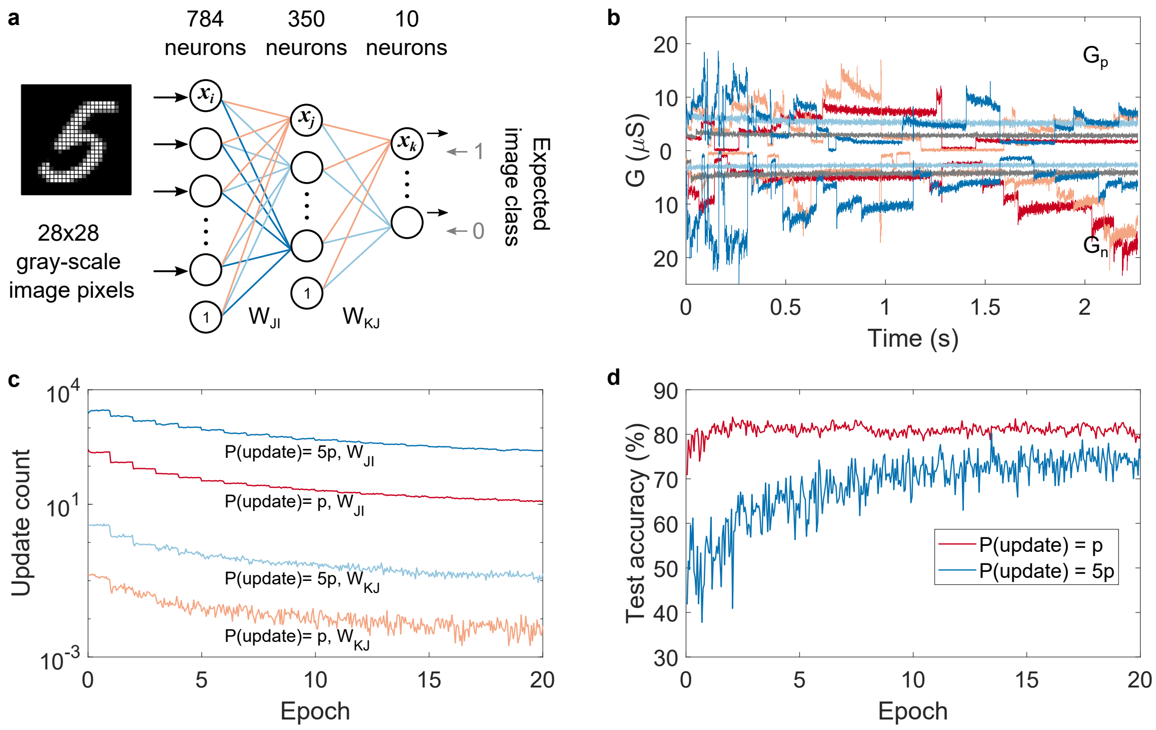


Figure 3.10 **a** The artificial neural network used for handwritten digit classification based on the MNIST dataset. **b** The network weights are implemented using 2 PCMs in differential configuration ($W = \beta(G_p - G_n)$). The conductance evolutions of the device pairs from few synapses are shown for the training duration. **c** The neuron activations and errors have been scaled to determine the weight update probability. The average number of weight updates per image in the two weight layers during the training for two chosen scaling factors are shown. **d** The test accuracies for the two update probabilities. The P(update)=p updated a smaller number of synapses using limited precision and stochastic PCM models and achieved higher test accuracies faster.
Source: [84].

response is compared with the original class label and an aggregate network error is determined. The algorithm tries to minimize the error by adjusting the weights in the network. For this, the gradient of the error function with respect to each weight in the network is determined using the back-propagation algorithm. This involves sending the error computed in the last layer to the previous layers successively through the corresponding synaptic weights. If x_i is the neuron activation of the pre-neuron from forwarding propagation and δ_j the error computed at the input of post-layer neuron of any weight during back-propagation, then desired weight-update for the synapse between these neurons can be computed as

$$\Delta W_{ij} = \eta . x_i . \delta_j \tag{3.12}$$

where η is a suitably chosen learning rate. The weighted summation or the matrix-vector multiplication necessary for the forward and backward propagation can be realized using the same crossbar array of devices, by feeding the vector as voltages respectively along the word line or bit line and reading the matrix-multiplication results as currents along the corresponding bit line or word line respectively.

During each epoch of the training, the network is presented with 60,000 training images and weight updates are computed using Equation 3.12 after each image. Software training of the ANN with high precision floating point weights gives around 98 % classification accuracy on the test set. Typically, $\Delta W/W < 10^{-3}$ for most of the ΔWs during this training. On the other hand, the PCM devices used for the synaptic implementation has a state-dependent and stochastic conductance update with very limited precision. As a result, when such non-volatile memory arrays are used for neural network training, transferring the desired weight updates becomes a major challenge. There are different proposals in the literature to solve this issue of low precision synapses by either using an additional memory for gradient accumulation [14, 69], or using more complex synapse structures [95, 109]. However, the additional overheads in these approaches constrain the maximum computational efficiency achievable in crossbar array based training architectures for neural networks. Here, we analyze the effect of the PCM response in training, where the weights are realized using two PCMs in differential configuration and the weight updates are implemented using single-shot programming pulses applied to the device model. Hence, the main ambiguity is in converting the desired weight updates into programming pulses. Due to the large disparity in the desired granularity of ΔW and the observed ΔG from the device, linear mapping between the two will lead the network to rarely experience any weight update. To solve this issue we used a scaled version of the x and δ to represent the probability to apply a programming pulse to the connected device. By adjusting the scaling factor, we could control the number of devices getting programmed in the network during each update. For illustration, we conducted two training simulations, where the update probabilities are p and $5p$, where p is a suitably chosen scaling factor. During training, we assumed, $1\,\mu$s computation delay per crossbar array resulting in a total training time of 2.26 s for 20 epochs. The conductance drift and read noise were re-evaluated after every training image during the simulations and a weight refresh was performed after every 1,000 images (Figure 3.10 b). Due to the probabilistic nature of the weight updates, out of the 278,260 weights in the ANN, only a small fraction of the total weights received updates after every image. The average number of devices updated after each image is shown in Figure 3.10 c for the two update probabilities. The corresponding classification accuracy of the network on the test set, which is not used for training is in Figure 3.10 d. The training experiment that received lesser programming updates achieved higher accuracy and converged faster. In contrast to the high-precision software-based training which has the flexibility to choose arbitrary learning rates, the weight updates in a low-precision device (such as the PCM synapse) is limited to the programming granularity of the device within its conductance range. The probabilistic sparse update scheme we use here could be viewed as an alternative approach to implement back-propagation, where instead of controlling the update of individual devices, the distance traveled on the error surface is chosen by controlling the number of devices being updated at any time. However, the limited device precision, non-linearity, and stochasticity seem to limit the maximum test accuracy achievable in this network to approximately 83 %, which is comparable to experimentally observed training result in a similar network using PCM devices [72].

While the training performance obtained in the simulation may seem subpar to the high-precision training, it is worth noting that biological synapses are stochastic and have state-dependent conductance update similar to the nano-scale non-volatile memory devices [15]. Some studies also suggest that they have a limited precision ($\sim$ 4.6 bit) [70]. Training algorithms, designed assuming floating point precision for the network weights, are not optimized for the limited precision weights. Considering the possible computational advantages of non-volatile memory based neural network implementations, adaptations or innovations of algorithms are necessary for accounting the underlying architecture and hardware limitations. In such studies, the model we presented which takes into account the device dynamics and variabilities will be highly useful.

3.5 Discussion

Now, we analyze the possible computational advantages of PCM based implementation of neural networks compared to the existing von Neumann architectures. We will also discuss how our modeling approach is applicable to other phase change material systems and how it might be affected by device scaling.

In a crossbar based matrix-vector multiplying unit, the PCM device acts as both local memory and an analog multiplier. The array structure enables them to perform standard $\mathcal{O}(N^2)$ complex matrix-vector multiplications in $\mathcal{O}(1)$ complexity with reduced data movement irrespective of the matrix size. The PCM devices are estimated to have 2 to 3-bit digital precision [110] and higher if some stochasticity could be tolerated. The area of a PCM cell with an access transistor is $\sim 25\,F^2$ (where F corresponds to the minimum lithographic pitch in a technology node), which could be reduced to $\sim 6\,F^2$ with a suitable diode based access device [111].

On the other hand, one bit SRAM area is $\geq 120\,F^2$ and the area of a 16-bit multiply-accumulate (MAC) required for neural network architectures is at least three orders of magnitude higher [111, 112]. This results in trade-offs between the number of parallel computing units and on-chip memory for hardware implementations of neural networks using conventional CMOS technology. Since the available silicon real estate per die is limited, energy-hungry off-chip memory access becomes essential for storing the network parameters [1]. On the other hand, due to its in-memory computation capability, crossbar arrays are estimated to outperform modern GPUs by four orders of magnitude [73]. This is particularly advantageous for SNNs, as it needs continuous time simulation which calls for more analog and inherently parallel architectures [30]. The computational efficiency of the crossbar array could be maintained to a large extent in the ANN training if its weight-update stage could also be performed directly on the array devices, based on the coincidence of stochastic pulses that represent the neuronal activations and back-propagated errors. However, this necessitates $\sim$ 10-bit update precision for the device to achieve state-of-the-art training accuracies [73]. Most of the non-volatile memory devices today are binary, while a few devices including PCM offers a few extra bits of precision.

However, recent results suggest that lack of precision in the non-volatile memory could be compensated by an accompanying higher-precision unit [14, 69, 109]. The granularity of the synapse could also be improved by using multiple devices per synapse [95]. In order to study such approaches for larger and more complex neural network problems, compact models that reliably capture the device statistics are required. The model presented in this paper serves this purpose. Furthermore, the insights developed from such training explorations could also be used to determine the specifications for future devices.

3.6 Summary

In this chapter, we characterized and analyzed the stochastic gradual conductance evolution behavior of phase change memory. The model we developed here captures the statistical device behavior accurately. The incorporation of temporal behaviors such as drift and read noise enables accurate system level simulations. We used the model to simulate the training behavior and observed the limitations imposed by the non-ideal behaviors typically observed in such nano-scale devices. Particularly, the inability to deliver finer updates due to the limited granularity resulted in achieving much lower accuracies compared to those in high-precision when trained with the back-propagation algorithm.

CHAPTER 4

MIXED-PRECISION TRAINING ARCHITECTURE

4.1 Introduction

We now propose a mixed-precision architecture to train computational memory based neural network implementations and analyze the effect of limited precision, stochasticity, asymmetry, non-linearity etc of the cross-point devices as presented in [69, 113].

Deep neural networks (DNN) share attributes of biological neural networks in its multi-layered structure and weighted interconnections called synapses. Neurons are parallel processing units performing non-linear transformations on the weighted inputs they receive from the preceding layers. Today, these network weights are trained using real-world examples to perform a wide variety of tasks such as image recognition and synthesis, speech recognition, and big-data analysis. The performance of these networks is typically observed to be proportional to the size of the network and the training data [3]. During the training phase, appropriate synaptic weights of the network are determined such that the network is able to perform the assigned learning task with sufficient accuracy. Typically, this is achieved by a supervised training algorithm known as backpropagation. First, the training examples are presented to the input layer which is then forward propagated through the connection weight matrices and neuronal activation functions. The final layer neuronal responses are compared with the desired outputs to compute the resulting error. The objective of the training process is to reduce this error by minimizing a cost function and is typically achieved via the gradient descent algorithm. In a multi-layered neural network, the gradients of the cost function with respect to all the weights are determined by back-propagating the errors from the final layer. The weights are updated based on the gradient information. This forward-backward data propagations and weight updates are repeated several times over the entire training data set. This brute force approach to training neural networks is computationally intensive and time-consuming, even when executed in general purpose graphical processing units (GPGPUs). Also, the high power consumption of this training approach makes its application prohibitive in several emerging domains such as the internet of things and edge computing. Much of the inefficiency arises from the constant shuttling of data between the physically separated memory and processing units in conventional von Neumann computing systems, in order to execute the large matrix operations involved in DNN training.

4.2 Computational Memory: Key Challenges

Non-volatile resistive memory devices have several attributes making them suitable candidates for building computational memory elements. These devices operate based on a variety of physical mechanisms such as field driven atomic rearrangement (metal-oxide based resistive memory [114] (ReRAM) and conductive bridge memory (CBRAM) [47]), spintronic effects (spin transfer torque based magnetic memory (STT-MRAM) [115] and phase transition (phase-change memory (PCM) [31]). Irrespective of the underlying physical mechanism, all these devices store information in their resistance or conductance states which are programmed by the application of suitable electrical pulses. However, there are many challenges associated with programming the desired conductance change in these devices. First, there are limitations on the minimum conductance change that can be reliably induced. For example in STT-MRAM, it is very difficult to achieve more than two conductance levels due to the underlying physical mechanism. Similarly in filamentary resistive memory devices such as CBRAM, the positive feedback mechanism involved in the filament growth process makes it difficult to control the process and to achieve intermediate states [45]. This inability to achieve a sufficiently small conductance change also limits the storage resolution. Another major challenge arises from stochasticity associated with device programming. In these nanoscale devices, slight changes in atomic configurations can lead to significantly different conductance values. Asymmetry in the conductance change, i.e., the average increment (potentiation) and decrement (depression) in conductance that can be reliably realized in a device is also an important challenge. Some devices also show significant state dependence where the conductance update depends on the current state of the device. For instance, this makes potentiation progressively harder with increasing conductance values. We refer to this as non-linear conductance response. In addition to weight update challenges, random volatile conductance fluctuations in the constituent elements of the computational memory and the finite resolution of the data converters used to interface them with the processing units could significantly impact the accuracy of the computations performed. In this chapter, we describe mixed-precision architecture based on computational memory and describe how it can address the aforementioned challenges. We use a neural network meant for classifying handwritten digits to benchmark the system performance.

4.3 Mixed-Precision Computational Memory Architecture

A schematic illustration of the mixed-precision computational memory architecture for training DNNs is shown in Figure 4.1. It has a computational memory unit comprising several memristive crossbar arrays coupled to a high-precision digital computing unit. If the weights W_{ji} in any layer in a DNN (Figure 4.1**a**) are mapped to the device conductances G_{ji} in the computational memory with an optional scaling factor, then the desired weighted summation operation during the data propagation stages of DNN training can be implemented as follows. For the forward propagation, the neuron activations, x_i, are converted to voltages, V_{x_i} and applied to the word lines. Currents will flow through individual devices based on its conductance and the total current through any bit-line, $I_j = \Sigma_i G_{ji} V_{x_i}$ will correspond to $\Sigma_i W_{ji} x_i$, which becomes the input for the next layer neurons. Similarly, for the backward propagation through the same layer, the voltages, V_{δ_j} corresponding to the gradient δ_j is applied to the bit lines of the same crossbar array and the weighted sum obtained along the word-lines, $\Sigma_j W_{ji} \delta_j$, can be used to determine the gradients for the neuron in the preceding layer.

The desired weight updates are determined according to the back-propagation algorithm as $\Delta W_{ji} = \eta \delta_j x_i$, where η is the learning rate. Since the devices representing the network weights often have very limited programming granularity, these weight updates cannot be directly transferred to the devices reliably. Hence we accumulate these updates in a high-precision variable χ in the high precision digital unit. In the proposed architecture, the weight updates in the devices are implemented by applying single-shot programming pulses without using iterative read-verify schemes. Let ϵ denote the average conductance change that can be reliably programmed into the devices in the computation memory unit using the given pulse. Then the number of programming pulses, p to be applied can be determined by rounding towards zero, i.e., $\lfloor \chi/\epsilon \rfloor$. The stored value of the variable χ is decremented by the factor $p\epsilon$, even though the actual change in the device conductance may not precisely match this value, as we do not read the device state after programming. Depending on the sign of p we increase or decrease the device conductance. Thus, we are effectively transferring the accumulated weight updates to the device only when it becomes comparable to its programming granularity. The computational memory device programming is greatly simplified in our scheme since the actual conductance state of the devices are not used to update the device or to confirm whether the requested weight updates are accurately attained as equivalent conductance changes in the devices. In spite of this, we show that this scheme works remarkably well and that the performance is comparable to those of floating-point implementations.

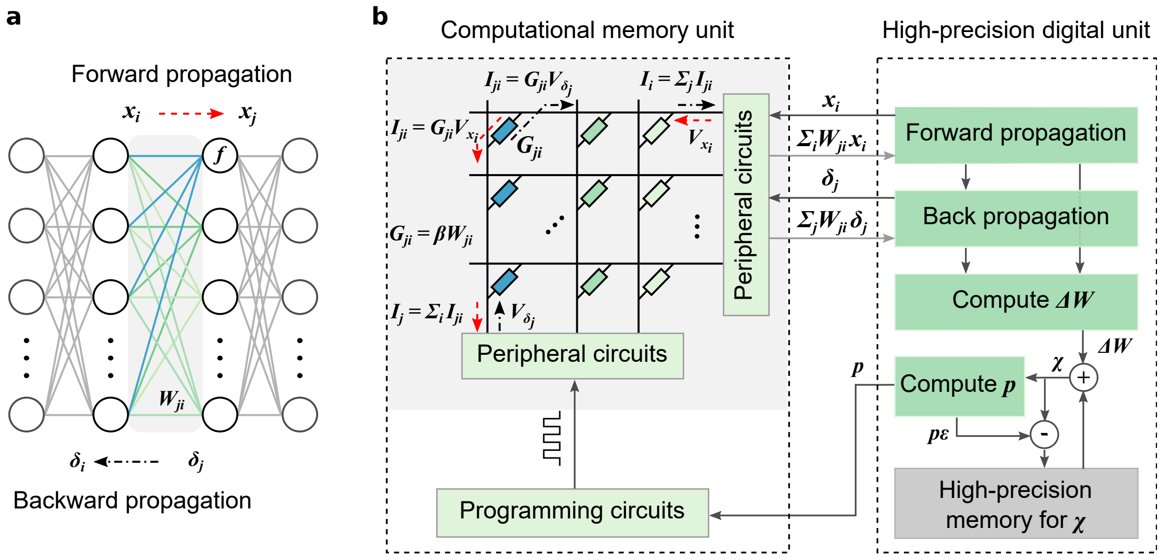


Figure 4.1 **a** Neural network consisting of multiple layers of neurons with weighted interconnects. During forward propagation, the neuron response x_i is weighted according to the connection strength W_{ji} and summed. Subsequently, a non-linear function f is applied to determine the response of neurons in the next layer, x_j. Similarly, the gradients from the layer, δ_j, is back-propagated through the weight layer to determine the gradients, δ_i, for the preceding layer. **b** Mixed-precision architecture consisting of a computational memory unit and a high-precision digital unit. Computational memory has memristive devices whose conductance values, G_{ji} represent the weights, W_{ji}. The crossbar array can perform the weighted summation during the forward and backward propagation. The resulting xs and δs could be used to determine the weights updates, ΔW in the high-precision unit. The ΔW is accumulated in the variable χ. The number of device programming pulses is determined as $p = \lfloor \chi/\epsilon \rfloor$, where ϵ is the average device update granularity.

4.4 Evaluation of the Mixed-Precision Architecture

4.4.1 The simulation framework

The performance of the mixed-precision architecture is analyzed based on its classification accuracy on the MNIST handwritten digit dataset using a neural network as shown schematically in Figure 4.2. The number of neurons in the input, the hidden and the output layer is 784, 250, and 10, respectively. The hidden and output

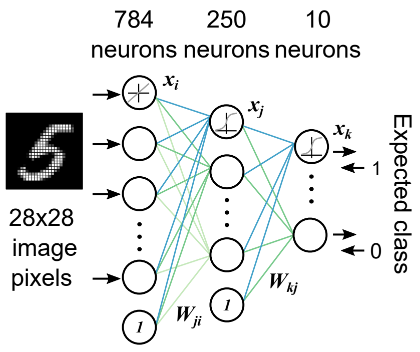


Figure 4.2 The neural network used to evaluate the mixed-precision architecture. The objective is the classification of handwritten digits based on the MNIST data set. There are 784 input neurons, 250 hidden sigmoid neurons, and 10 output sigmoid neurons. The network weights are trained by optimizing a quadratic objective function using gradient descent. All the 60,000 images in the dataset are used in one epoch of training and 10,000 images for testing.

neurons are sigmoid. The network is trained using the entire training set of 60,000 images for ten epochs, and test accuracy is reported based on 10,000 test images. The pixel values of the 28×28 gray-scale images are normalized between 0 and 1 before they are supplied as input to the network. No other preprocessing is performed on the images. We used the quadratic objective function for the back-propagation-based training and used a fixed learning rate. The network gave 98 % floating point (64-bit) test accuracy when trained using stochastic gradient descent. This classification result is used as *reference* to evaluate the performance of our mixed-precision approach. The final weight distribution from this high-precision training was approximately in the range [-1 1].

4.4.2 Inaccuracies arising from weight-updates

In this section, we will evaluate how the proposed architecture copes with the issues associated with weight updates. We assume a hypothetical linear device with a conductance range of [-1 1] similar to the floating-point trained weight distribution. The device is assumed to have n-bit update granularity such that it covers its conductance range in $2^n - 2$ steps and hence, it will have $2^n - 1$ possible levels

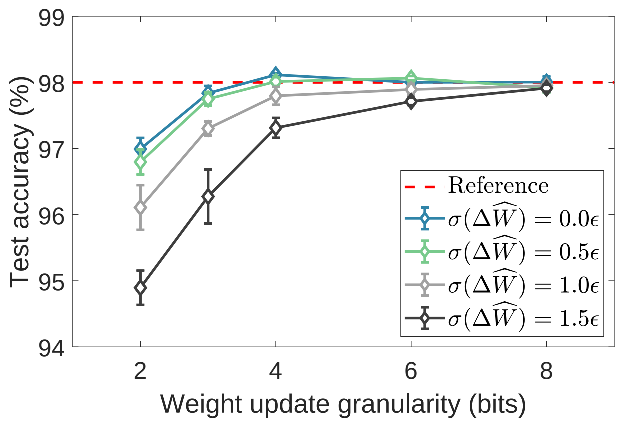


Figure 4.3 Effect of granularity and stochasticity associated with weight updates. Linear devices with symmetric potentiation and depression granularity are assumed as computational memory elements. The standard deviation of the weight-update randomness, $\sigma(\Delta\widehat{W})$, is taken as a multiple of the weight update granularity, ϵ. The error-bars indicate the standard deviation corresponding to five repetitions of the simulation. It can be seen that even in the extreme cases of highly coarse and random weight-updates, drop in the test accuracy is still within approximately 4% for 2-bit granularity.
Source: [69].

in the absence of conductance change stochasticity. An odd number of levels was chosen to include zero. Therefore, in our mixed-precision training approach, we will update the device when the weight-update accumulation exceeds the conductance change step size, $\epsilon = 2/(2^n - 2)$. In subsequent discussions, we will use both ϵ and n interchangeably to indicate the granularity associated with the weight-updates.

The conductance updates in the non-volatile devices are often stochastic. Even though it is desirable to induce a change in conductance corresponding to an integer multiple of ϵ, the observed change is often quite different from the desired one. Therefore, the actual weight update from the device, denoted by $\Delta\widehat{W}$, is modeled as a Gaussian random variable whose mean is ϵ and whose standard deviation (σ) is a fractional multiple of ϵ. This device model is used as the computational memory elements representing the neural network synapses during its training using the mixed-precision scheme. The devices are initialized to {-1, 0, 1} states with a

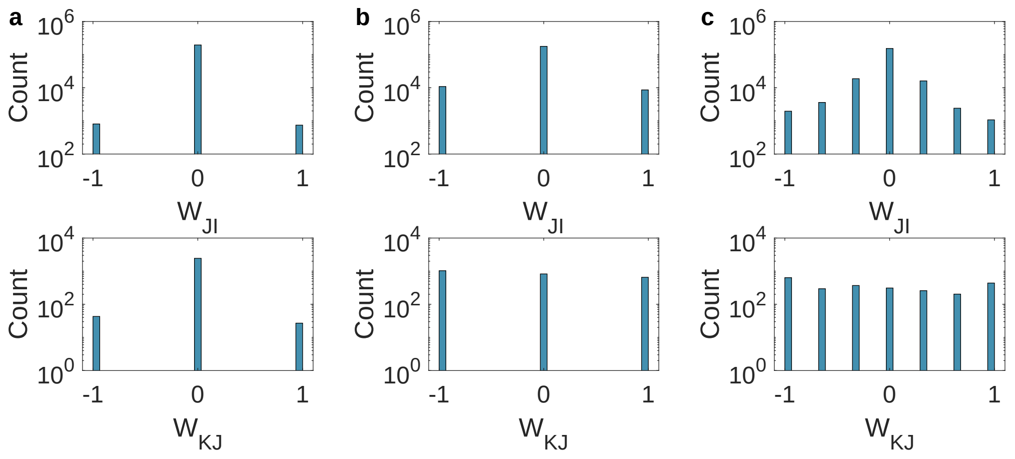


Figure 4.4 **a** In the linear device simulations, the weights are initialized to a set of states -1, 0, and 1. **b** In non-stochastic two-bit granular updates the devices go through only these discrete states and hence the update granularity also becomes the device resolution. This discrete weight solution gave a test accuracy of approximately 97%. However, in the case of stochastic programming, the devices can achieve intermediate states. **c** The final weight distribution from the 3-bit update granularity simulation is also shown. The higher weight resolution improved the test accuracy by approximately 1%.
Source: [113].

discrete distribution whose variance is normalized by the number of neurons in the pre- and post-synaptic layers. Device read noise and analog-digital converters are ignored at this stage. The simulated classification accuracies with limited granularity and with different amounts of stochasticity in the updates are shown in Figure 4.3. In the case where the weight updates are non-stochastic the test accuracy drop is only 1% for 2-bit, and with 3-bit granularity, the accuracy is very close to that obtained in the floating-point simulation (*reference*). As the stochasticity increases, the performance degrades with diminishing number of bits. However, it is remarkable that even though the standard deviation of the weight update is equal to or greater than the mean weight update granularity itself, drop in the test accuracy is still within approximately 4% for 2-bit granularity. The test accuracy becomes closer to the reference floating-point accuracy as the device granularity is further reduced.

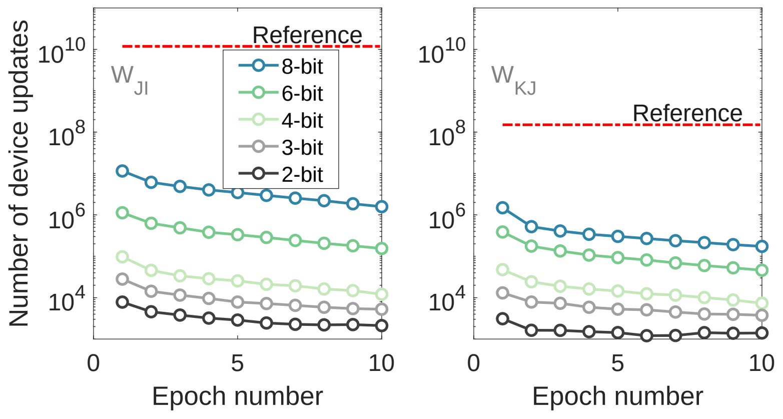


Figure 4.5 Sparsity of device updates in the computational memory array. The device update count per epoch is plotted for different values of ϵ. Only a fraction of the total number of devices is programmed after each image. The number of synapses in each layer times the total training image count is indicated as the reference. As the value of ϵ increases, the number of updates reduces by several orders of magnitude, saving significant programming overhead.
Source: [69].

The distributions of the initial and trained weights in the two layers of the neural network for the 2-bit and 3-bit update granularity are shown in Figure 4.4. The distributions are shown for non-stochastic device programming and hence the final weights are also discrete and the number of levels corresponds to the update granularity. We observe that increasing the number of levels improved the classification performance until an update granularity of 4-bit beyond which the test accuracies remained approximately constant. The stochasticity associated with conductance updates helps to create a non-discrete weight distribution. On the other hand, we found this to have no significant advantage and we typically observe a decrease in the classification performance with increasing stochasticity (Figure 4.3).

From the previous discussion, it can be seen that increasing the resolution of conductance change beyond a certain value does not necessarily improve the network performance. Moreover, in the mixed-precision scheme, there is a significant reduction in the device programming cost with the use of larger ϵs. In Figure 4.5, we show the

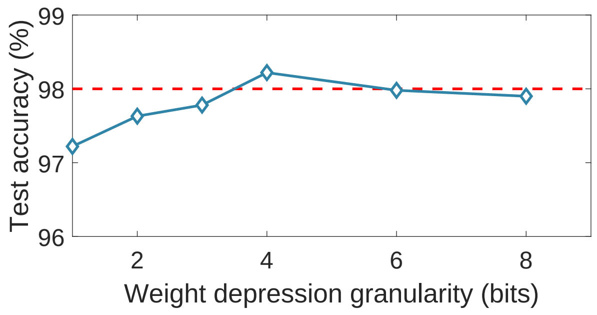


Figure 4.6 Effect of asymmetric conductance update in mixed-precision training. The test accuracy, when trained with devices of fixed 8-bit potentiation granularity and variable depression granularity, is plotted as a function of the depression granularity (expressed in bits). Weight updates are assumed to be deterministic. The resulting test accuracy shows less than 1% accuracy drop even in the highest asymmetric case in the simulation.
Source: [69].

number of device updates during each epoch of training. The maximum number of device updates, calculated as the product of the synapse count and the training image count, assuming all the weights are updated after each image presentation, is indicated as the reference. However, in the mixed-precision approach, we accumulate the updates in high precision. As a result, the smaller updates are combined and delivered together to the device. Hence, as the device update granularity (ϵ) increases, the devices need to be programmed less often, resulting in eventual energy savings. Programming resistive memory devices incur significant time and power penalty and hence it is desirable to reduce the number of such programming instances, without compromising the network performance. For the chosen network architecture and classification problem, 4-bit update granularity seems to offer the best case scenario of highest test accuracy with reduced programming expense.

Next, we study the influence of asymmetric conductance update response. We assume a device with fixed but unequal potentiation and depression granularity. The mixed-precision method can cope with this behavior by using different thresholds, ϵ_P for conductance increment and ϵ_D for conductance decrement. For example, in

Figure 4.6 we assume an 8-bit potentiation granularity and the depression granularity is varied. The one-bit depression in the figure corresponds to a situation where the update granularity, ϵ_D, equals the entire weight range in contrast to the previous definition. The weight updates are assumed to be deterministic. The resulting test accuracies show less than 1% drop even for the maximum asymmetric case tested, demonstrating the efficacy of the proposed scheme to tolerate device update asymmetry effectively.

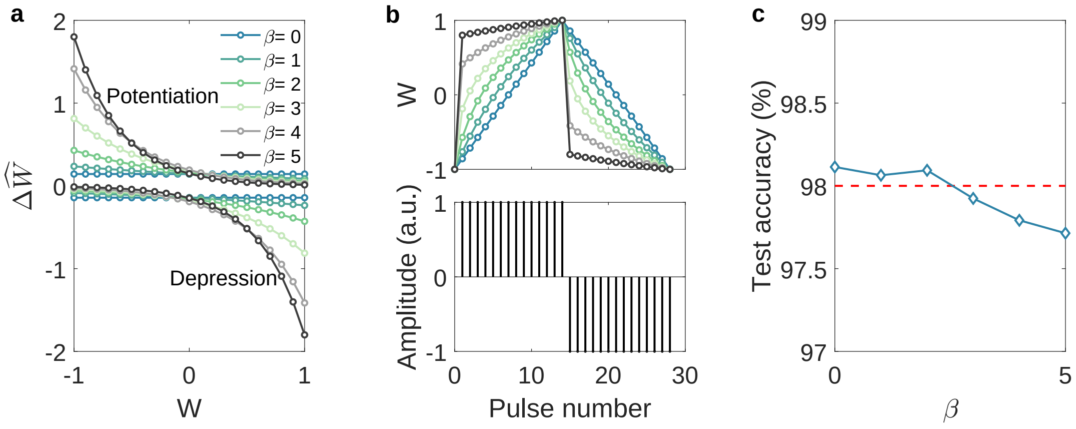


Figure 4.7 **a** Non-linear device model: the weight update (ΔW) is modeled as exponentially dependent on the current state, W. The exponential function for different amounts of non-linearity is plotted. **b** Corresponding device model pulse response, where 14 potentiation pulses followed by the same number of depression pulses are applied to the device. **c** Non-linear device model as synapse for DNN training. $\beta = 0$ correspond to a linear and symmetric device and higher β values indicate an increasing amount of non-linearity. Approximately 4-bit weight update granularity is assumed for the device model and mixed-precision training. Weight updates were non-stochastic. The result shows that there is no significant degradation in the test accuracy even for $\beta = 5$ that corresponds to a highly non-linear conductance response.
Source: [113].

Subsequently, we investigate the influence of the non-linear conductance response. To analyze this we simulated the training problem using a device model whose non-linearity could be tuned. We chose an exponential function to model the state dependency of ΔW as suggested by Querlioz et al. [116]. The model essentially captures the behavior where a resistive memory device closer to the boundary conductance will exhibit a smaller update compared to those away from it when updated towards the boundary.

$$\Delta\widehat{W}_P = \alpha e^{-\beta \frac{W - W_{min}}{W_{max} - W_{min}}} \tag{4.1}$$

$$\Delta\widehat{W}_D = \alpha e^{-\beta \frac{W_{max} - W}{W_{max} - W_{min}}} \tag{4.2}$$

Here, $\Delta\widehat{W}_P$ and $\Delta\widehat{W}_D$ model the potentiation and depression, respectively for a device at a conductance of W. W_{min} and W_{max} represent the limits of the device conductance. We used the parameter β to tune the amount of non-linearity and α to adjust the update granularity. To make a reasonable comparison in training performance using models of different amount of non-linearity, we assume that two criteria have to be satisfied: the device models must have the same on-off ratio and they must take the same number of programming steps to span the whole conductance range, irrespective of the non-linearity. The $\Delta\widehat{W}$ versus W, and W versus pulse number responses satisfying these conditions for different values of β are shown in Figure 4.7(a) and (b). Here, $\beta = 0$ correspond to a linear device. The number of pulses for full range potentiation or depression is assumed to be corresponding to that of a 4-bit update granularity. The same update granularity is assumed to determine the ϵ for the mixed-precision scheme for varying amount of non-linearity. The resulting test accuracies are plotted as a function of β in Figure 4.7(c). It can be seen that there is no significant degradation in the test accuracy even for $\beta = 5$, which is very close to the behavior of a binary device.

4.4.3 Inaccuracies arising from matrix-vector multiplication

In this section, we analyze the influence of conductance fluctuations and finite resolution of data converters. Resistive memory devices typically exhibit fluctuations

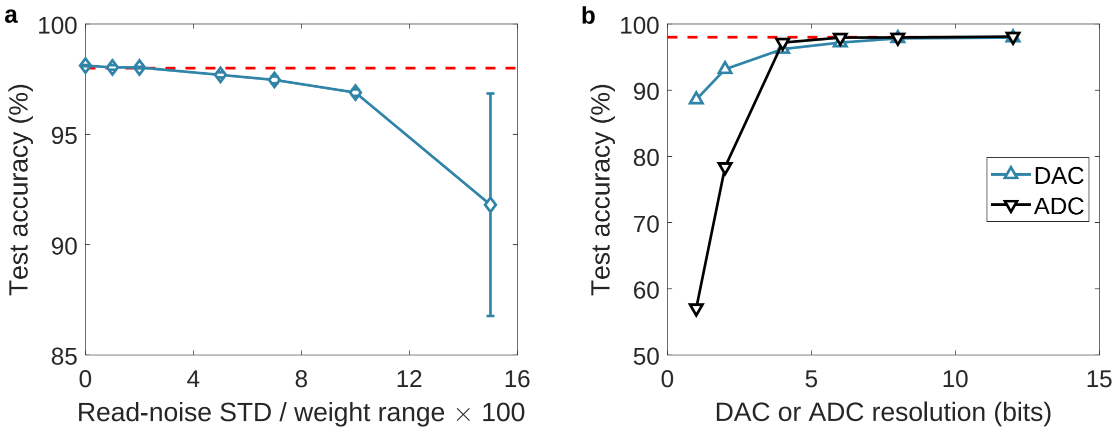


Figure 4.8 **a** Effect of read noise. Gaussian distributed additive read noise is added with the computational memory devices whenever they are used for multiplication. The standard deviation (STD) of the read noise is varied as a fraction of the total device conductance range. For the mixed-precision training, a 4-bit weight update granularity and non-stochastic programming are assumed. There is no significant loss in test accuracy even up to a read noise corresponding to 5% of the total weight range. **b** Effect of finite resolution data converter. The weight update granularity is assumed to be 4-bit, without stochasticity and read noise. The curve with triangle indicates simulation results where DACs are used at the crossbar input whereas the output current is read back in floating-point precision. The curve with the inverted triangle indicates results where the crossbar input has floating point precision whereas ADCs are used for reading back the output current.
Source: [69].

in conductance arising from trapping/detrapping processes [117]. The effect of this read noise in the DNN training using the proposed scheme is tested by adding a zero mean white Gaussian noise to a linear device model. The noise is added to the weights whenever it is used in the matrix multiplication in the forward and the backward propagation. It is also incorporated during the testing phase (only forward propagation). The standard deviation of the noise is varied as a fraction of the total weight range. The resulting test accuracies are shown in Figure 4.8a. It can be seen that the methodology is quite robust to up to 5% read noise.

An additional source of error in the matrix-vector multiplication is due to quantization from the DACs and ADCs. During forward propagation, the neuron activations evaluated in the digital domain are converted to analog voltages using

DACs before they are applied to the word lines of the crossbar array. The weighted sums obtained as currents in the bit lines are read back using ADCs. Similarly, the back-propagated errors are converted to analog voltages when applied to the crossbar array matrices. The ranges for the digital to analog converters are fixed for *sigmoid* and *tanh* neuron activations, whereas for the ReLu neurons this could be a challenge as their range dependents on the data and weight distribution. Here, we chose sigmoid neurons for our network, which fixed the DAC range in the forward propagation. Furthermore, we normalized the back-propagated errors to fix the range for its interface converters to analog voltages. The normalization factor is multiplied with the learning rate during the weight update calculation. However, the input for the analog to digital converters are matrix-vector multiplication results and their distribution is dependent on the number of neurons and the weight distribution in the layer. In this work, the range for ADC was determined by observing the distribution of corresponding variables representing the weighted sums. To study the effect of DACs and ADCs separately, the bit precision of one of them is varied, whereas the other variables are represented in floating-point precision. Figure 4.8b shows that 8-bit resolution is sufficient to avoid a noticeable degradation in test accuracy.

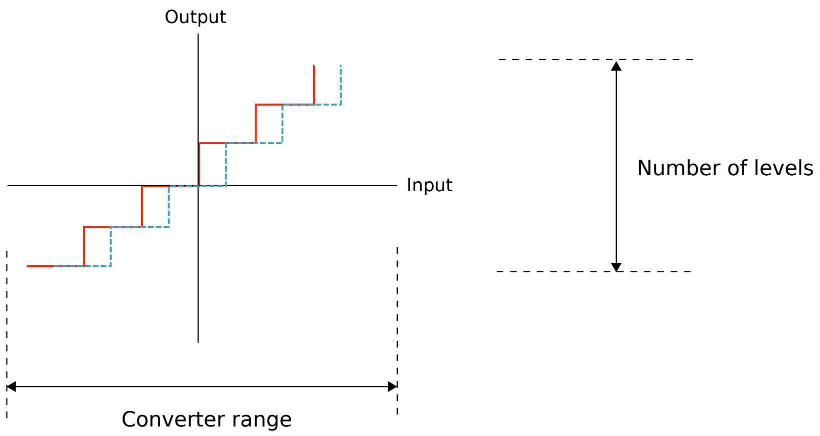


Figure 4.9 The quantization function at the computational memory periphery should be chosen such that there is no edge at $x = 0$, as seen in the blue curve, as opposed to the red curve.

Quantization function The weight updates during the training are determined as the product of the neuron activations and back-propagated errors ($\Delta W = \eta.x.\delta$). While the magnitude of the ΔW is determined by the learning rate η, whether the weights are potentiated or depressed or not updated is determined by the sign of the x and δ. It is essential to maintain the direction of weight update for the faithful implementation of the gradient descent based backpropagation algorithm. Therefore, the minimum amount of information to be maintained after quantization of neuron activations and error function is if they are positive, negative, or zero, requiring a minimum of 3 levels. Hence, quantization functions should be designed not to lose this information as the signal is propagated through the multiple weight layers implemented using the computational memory arrays. For example, the quantization function must not exhibit an edge at the point $x = 0$ (i.e. as seen with the blue curve in Figure 4.9 as opposed to the red curve), as this leads to severe performance degradation, due to near-zero activations and gradients being mapped to a value which is much larger in magnitude. Considering that in most cases one wants the range to be symmetric $q_{min} = -q_{max}$ this motivates our choice of quantization function:

$$q(x) = clip\left(\frac{q_{max} - q_{min}}{2^n} \cdot round\left(\frac{2^n}{q_{max} - q_{min}}(x - q_{min})\right) + q_{min}\right) \tag{4.3}$$

where $clip(x)$ clips the value of x to $[q_{min}, q_{max}]$ and $round(x)$ rounds x to the nearest integer value.

Since the magnitudes of the backpropagated gradients change over training examples, they are normalized before feeding into the crossbar array. The resulting output vector from the array is scaled back using the same normalization factor. The full matrix-vector multiplication $W \cdot x$ with a matrix W and an input vector x is therefore simulated in the following manner:

$$q_{ADC}\left(W \cdot q_{DAC}\left(\frac{x}{||x||}\right)\right) ||x|| \tag{4.4}$$

where $q_{DAC}()$ and $q_{ADC}()$ are the quantization functions corresponding to the DAC, and the ADC, respectively. Since the data input to the DAC is in the range $[-1, 1]$ or sometimes $[0, 1]$ if the preceding neuron activations are Tanh or Sigmoid, its quantization ranges for the different layers can be easily determined. Meanwhile, the ADC inputs are summations of currents from the crossbar array and its range is influenced by the array size, array input distribution, and the array device conductance distribution. While it may be possible to determine the ADC quantization ranges from the knowledge of these factors, for the current study we used distributions obtained from the corresponding reference training performed in high-precision. Scaling of the conductance to the actual synaptic weight is also obtained from the high-precision trained weight distribution. However, the requirement of reference can be avoided by adaptive scaling factors at the crossbar periphery in future implementations (Chapter 5).

4.5 Discussion

The non-volatile memory crossbar array based computational memory unit is ideally suited to perform matrix-vector multiplications. By utilizing the computational memory to perform those operations when training DNNs, the forward and the backward propagation of data can be significantly accelerated. Also, the processor-memory bottleneck is reduced as the synaptic weights are not transferred during the propagations. However, the necessity to frequently update the memory devices poses an additional challenge compared to applications of computational memory where the matrix does not change [12, 13]. In back-propagation based training algorithm it is desirable to update the weight matrix after the presentation of each training instances. Using devices like PCM, which can attain a continuum of conductance states, it is possible to iteratively program the devices to the desired conductance states accurately [90]. On the other hand, this involves repeated read/write cycles and incur significant time/power penalty. The necessity to program a large number of devices very often could overshadow the performance gain that we obtain from the in-place matrix-vector multiplication in the crossbar array.

On the other end of the spectrum lies the non-von Neumann coprocessor approach proposed recently [72, 73]. As before, the synaptic weights are stored in resistive memory devices organized in crossbar arrays and the matrix-vector multiplications during forward and backward propagations are realized in place using these arrays. However, they suggest a fully parallel conductance update by overlapping pulses from the pre- and post-synaptic neuron layers. By realizing the neurons and associated circuits in place, this offers the possibility of a fully parallel non-von Neumann system. By accelerating all the three components of training DNNs, namely forward propagation, backward propagation, and weight update, this approach could be the fastest and most energy-efficient compared to alternate approaches. However, the non-idealities associated with programming the memory devices will pose significant challenges in realizing state-of-the-art classification accuracies. An ideal device is expected to have a 10-bit symmetric weight update granularity [73]. Experimental demonstrations using more realistic phase change memory devices have shown a limited test accuracy of less than 83% [72].

Our mixed-precision approach is designed to take into account the limited device update granularity seen in experimental devices today. The proposed architecture is significantly tolerant of conductance programming asymmetry and update non-linearity. In contrast to the above-discussed methods, we deliver the conductance updates only when weight updates accumulated in high precision become comparable to the device update size. As a result, the number of device programming instances are reduced by several orders of magnitude as the update size increases. As a result, the advantage of matrix-vector multiplication acceleration in the data propagation stages is preserved without significant device programming overhead. We follow a blind single pulse programming approach without read-back to deliver an ϵ amount of update. The value of ϵ is chosen based on the device dynamics. The simulations show that the resulting sparse weight updates training are able to achieve classification accuracies comparable to those from the floating-point simulations in a similar number of training epochs. Further, the high precision accumulation and less frequent weight-updates combined with the inherent error tolerance of neural network training enable the architecture to cope with the high device programming stochasticity.

We believe that the weight update and accumulation overhead associated with this mixed-precision architecture is significantly less compared to the training acceleration we obtain. The training acceleration is achieved by computing the multiply-accumulate operation of approximately $\mathcal{O}(N^2)$ complexity in fixed time for each $N \times N$ neural network layer. The device updates are sparse and the weight update accumulation in high precision is equivalent to the weight update scheme in standard stochastic gradient decent except that the memory is initialized to zero here. The additional thresholding/flooring and subtraction operations are computationally simple and do not incur additional memory read/write operations as they can be performed concurrently with the weight-update accumulation. Still, it is desirable to further accelerate the weight update stage of DNN training as the weight-update determination is an $\mathcal{O}(N^2)$ operation.

4.6 Summary

The limited granularity of the conductance change is the major limiting factor in achieving high training performance from non-volatile memory arrays. We proposed a mixed-precision scheme to compensate this by accumulating the weight updates in high-precision and programming it to the device only when the accumulated amount is comparable to the device update granularity. This scheme allows the use of the crossbar memory array to accelerate the forward and backward data propagation stages of the back-propagation algorithm. The architecture is shown to be tolerant to the device non-idealities such as limited granularity, stochasticity, asymmetry, and non-linearity.

CHAPTER 5

EXPERIMENTAL VALIDATION OF MIXED-PRECISION TRAINING ARCHITECTURE

5.1 Introduction

In this chapter, we present experimental validations of the mixed-precision architecture (MPA) introduced in Chapter 4. The neural network synapses are realized using the PCM array fabricated in 90 nm technology. The PCM model developed in Chapter 3 is used to pre-validate the experiment. We demonstrate floating point precision comparable training and test accuracies and inference over a month's time from the experiment. The synaptic device we used is stochastic, non-linear, asymmetric, and limited precision conductance update behavior in response to the programming pulses. Successful experimental demonstration using such non-ideal devices validate the efficacy of the training architecture.

5.2 Experiment for Training Handwritten Digit Classification Network

5.2.1 Training using differential PCM synapses

We first experimentally demonstrate the efficacy of the mixed-precision architecture by training a two-layer neural network to classify handwritten digits from the MNIST dataset (Figure 5.1a). The network was trained using the 60,000 training images for 30 epochs with a batch size of 1, i.e., the weight updates were computed after every training example. The order of the images was randomized for each training epoch. The gray-scale images were normalized to lie between 0 and 1 and no other pre-processing was performed on the training set. We used a fixed learning rate of 0.4.

Each weight of the network, W, is realized using two PCM devices in a differential configuration ($W \propto (G_p - G_n)$). The 198,760 weights in the network

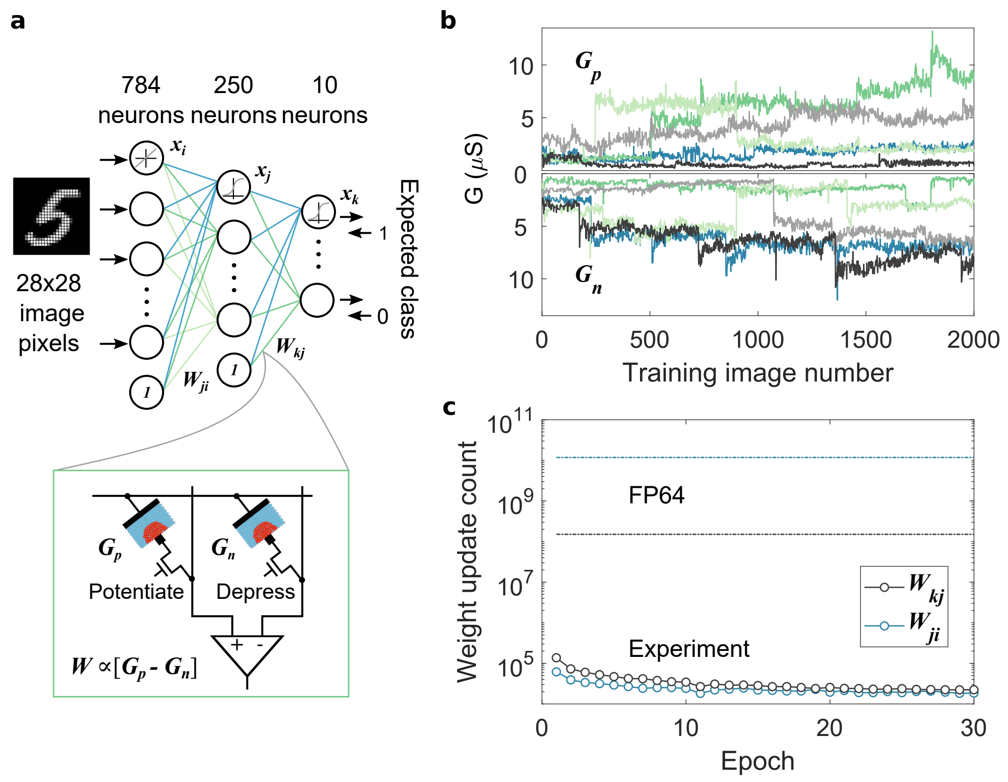


Figure 5.1 **a** Network structure used for the on-chip mixed-precision training experiment for MNIST data classification. Each weight W in the network is realized as the difference in conductance values of two PCM cells, G_p and G_n. **b** Stochastic conductance evolution of G_p and G_n corresponding to 5 synaptic weights associated with the second layer during training. **c** The number of device updates per epoch from the two weight layers in mixed-precision training experiment and high-precision software training. It shows the highly sparse nature of weight update in MPA.

are mapped to 397,520 PCM devices in the hardware platform. A few empirical rules have been proposed for weight initialization to achieve high training performance. A normalized initialization [118] suggests a uniform distribution, given as

$$W \sim U\Big[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\Big] \tag{5.1}$$

where n_j and n_{j+1} are the number of neurons in the pre- and post weight layers. This initialization ensures that the weighted sum falls in a range where the neuron activation function (sigmoid in this case) has a non-zero gradient. However, the

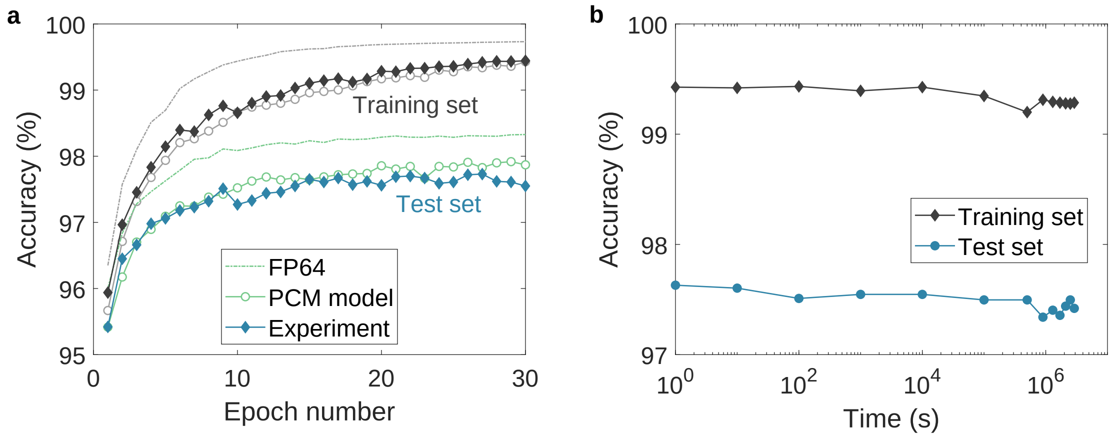


Figure 5.2 **a** Classification accuracies on the training and test set from the mixed-precision training experiment. The maximum experimental test set classification accuracy, 97.73%, is within 0.57% of that obtained in the floating-point (FP64) software training. The experimental behavior is closely matched by the training simulation using the PCM model. **b** Inference performed using the trained PCM weights on-chip on the training and test dataset as a function of time elapsed after training showing negligible accuracy drop over a period of one month.

stochastic nature of the conductance programming makes it challenging to achieve narrow distributions in different weight layers using nanoscale devices. In PCM, we use an iterative programming scheme to achieve a specific target conductance. A sequence of programming pulses are applied and the error with respect to the target conductance after each pulse is used to determine the pulse amplitude of the next programming pulse. We used 500 ns pulses and current pulses of amplitudes in the right half of the PCM programming curve. The maximum number of such programming attempts per device was limited to 20. The standard deviation and distance from the target value were observed to be dependent on the target value and conductance drift. In our experiment, the iteratively initialized conductance distribution had a mean of 1.6 μS and a standard deviation of 0.83 μS. The device conductances were mapped to the neural network weights using a linear mapping between [-8 μS 8 μS] in the conductance domain and [-1 1] in the weight domain.

Since the platform allowed only serial access to the devices, all the conductance values were read and the matrix-vector multiplications for the forward and backward data-propagation were performed in software. The resulting weights updates were accumulated in the variable χ as in the MPA. When the magnitude of χ exceeds ϵ ($= 0.096$ corresponding to an average conductance change of $0.77\,\mu$S per chosen programming pulse), a 50 ns, $90\,\mu$A pulse is applied to G_p to increase the weight if $\chi > 0$ or to G_n to decrease the weight if $\chi < 0$; $|\chi|$ is then reduced by ϵ. These device updates were performed using blind single-shot pulses without a read-verify operation and the device states were not used to determine the number or shape of programming pulses. Since the continuous partial SET programming could cause some of the devices to saturate during training, a weight refresh operation is performed every 100 training images to detect and reprogram the saturated synapses. For the weight refresh operation, the conductance pairs were read and if one of the device conductance is above $8\,\mu$S and if their difference is less than $6\,\mu$S, both the devices were RESET using 500 ns, $360\,\mu$A pulses and their difference was converted to a number of SET pulses based on an average observed conductance change per pulse. During this weight refresh, the maximum number of pulses was limited to 3 and the pulses were applied to G_p or G_n depending on the sign of their initial conductance difference. The updated conductances of the devices were read and used for the subsequent data-propagation.

The PCM devices show temporal dynamics such as conductance drift and read noise. As a result, each matrix multiplication in every layer will see a slightly different weight matrix even without any weight update. However, the cost of re-reading the entire conductance array for every matrix-vector multiplication in our experiment was prohibitively large due to the serial interface. Therefore, we read all devices once after the initialization and along with every succeeding read of programmed devices, we read all the devices in the second layer and a set of 785 pairs of devices from the

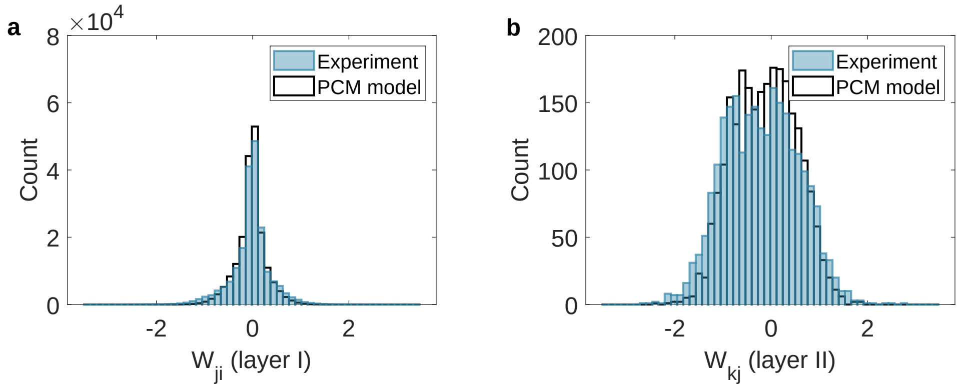


Figure 5.3 The PCM model conductance distribution in the two weight layers ((a) and (b)) at the end of 30 epochs of training matches that from the experiment. It indicates that the model is able to capture the overall device programming dynamics in the training scenario which involves many device programming events with varying amounts of intermittent delays and occasional RESET.

first layer in a round robin fashion. As a result, the device states in the second layer is updated on an average every 2 s, while those in the first layer is updated every 500 s.

The resulting evolution of conductance pairs, G_p and G_n for five random synapses from the second layer experienced by the neural network during the training is shown in Figure 5.1b. It illustrates the stochastic conductance update, drift between multiple training images, and the read noise experienced by the neural network during training. Due to the accumulate-and-program nature of the mixed-precision training, only a few devices were updated after each image. In Figure 5.1c, the number of weight updates per epoch in each layer during training is shown. Compared to a high-precision training where all the weights are updated after each image, there are many orders of magnitude reduction in the number of updates in the mixed-precision scheme, thereby reducing the device programming overhead.

The classification performance of the network was evaluated after every epoch on the entire training set and on a 10,000 test set which was not used for the training (Figure 5.2a). The network achieved a maximum test set accuracy of 97.73% which is only 0.57% lower than the equivalent classification accuracy of 98.30% achieved in the high-precision training.

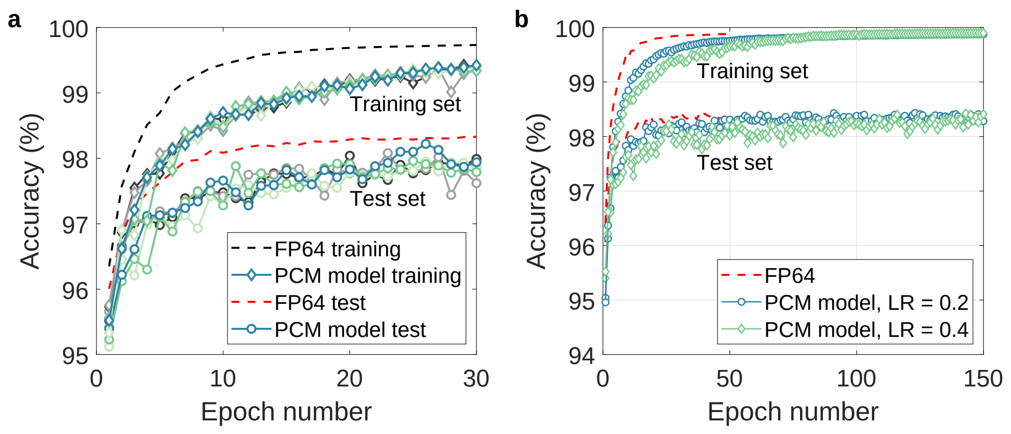


Figure 5.4 **a** Training simulation of the experiment using multiple random seeds. **b** Training for longer epochs: even though the programming stochasticity could slow down initial training convergence, the MPA accuracy could be equivalent to floating-point given sufficient training epochs.

The high precision comparable training performance from the mixed-precision deep learning architecture, where the computational memory comprises noisy non-linear devices with highly stochastic behavior demonstrates the existence of a solution to these complex deep learning problems in device-generated weight space. And even more remarkably, it highlights the ability of the mixed-precision architecture to successfully find the solution. We used the device model we developed, to pre-validate the training experiment in simulation and the resulting training and test accuracies are also plotted in Figure 5.2a. The model was able to predict the experimental classification accuracies on both training and test set within 0.3 %, making it a potential tool to evaluate the trainability of PCM-based computational memory for more complex deep learning applications. The model was also able to predict the distribution of synaptic weights across the two layers remarkably well (Figure 5.3).

The trajectories of training and test accuracies in Figure 5.2a still lags behind those observed in the floating-point training. This difference in performance could be attributed to the device programming noise. During training, the weight updates are computed using the stochastic gradient descent and accumulated in high-precision.

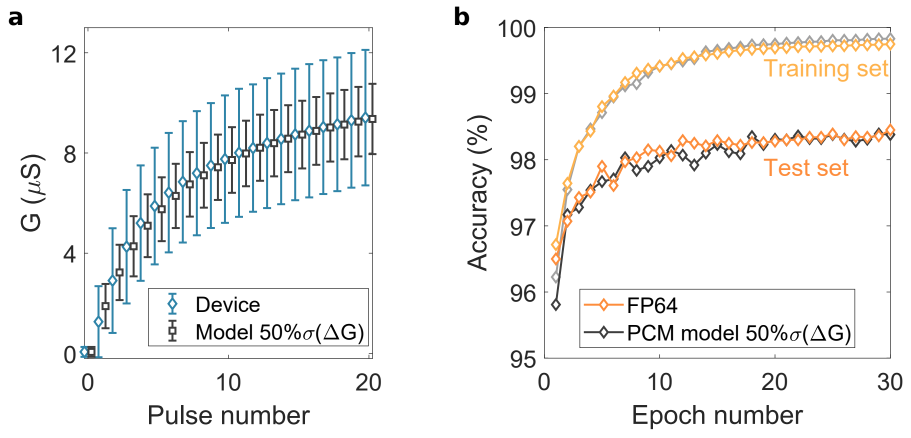


Figure 5.5 **a** The PCM model response whose programming noise is scaled by 0.5 is compared with the actual device response. Error bar shows one standard deviation. **b** Classification performance evolution during the training of the two layer MNIST classifier with the PCM model with reduced noise.

However, the device programming used to transfer these accumulated updates is highly stochastic. This may cause the network to take longer to reach the ideal solution (Figure 5.4). In Figure 5.4, we also show, via simulations using the PCM model, that the training performance could be improved by slightly lowering the learning rate. A lower learning rate implies a longer accumulation of computed weight updates before it reaches the threshold and is transferred to the device. This reduction in programming frequency leads to a reduced number of deviations from the desired conductance and thus, faster convergence. While lowering the learning rate might seem to improve the convergence and reduce the programming energy, eventually, it could cause the training to be slower.

Another direction that we pursued was to engineer the device for reduced programming stochasticity. To test this we scaled the standard deviation of the programming noise ($\sigma(\Delta G)$) of our PCM model and observed that a 50 % reduction in stochasticity was necessary to achieve floating point equivalent training convergence. The model response we used to test the training and the resulting training performance is shown in Figure 5.5.

5.2.2 On-chip inference

After the training, the network weights in the PCM array are read repeatedly over time and the classification performance (inference) was evaluated on training and test datasets (Figure 5.2b). The network showed less than 0.3 % drop in classification accuracies for over a month's time indicating the feasibility of using trained PCM based computational memory as an inference engine.

5.2.3 Training using on-chip ohmic multiplication

Since the PCM experimental platform allows only serial access to the individual devices, the experimental demonstrations were primarily performed such that multiplications and additions as part of the matrix computations were performed in the software outside the chip with conductance values read from the array using a constant read-voltage. However, this approach neglects the device non-linearity in response to the different read voltage amplitude. The actual conductance mechanism in the chalcogenide phase change material is an active area of research [101, 119]. In this section, we describe an experiment which is used to validate the mixed precision training architecture even in the presence of PCM non-linear I-V response.

For a real cross-bar array based on matrix-vector multiplication, the neuron activations and the backpropagated errors are converted to voltages and applied at the word/bit lines. Our experimental platform allows a read voltage from 0.12 V to 0.3 V. Hence, we normalized the activation and error vectors and used a linear mapping to convert their magnitude in the range of [0 1] to [0.0 V 0.3 V]. The voltage values below the minimum read voltage of 0.12 V are neglected. The effective mapping is shown in Figure 5.6a.

In this experiment, all neuron activations and error vectors are converted to read voltages and are used to read the conductance values for each training instance. Due to the sequential nature of the memory access in the current experimental

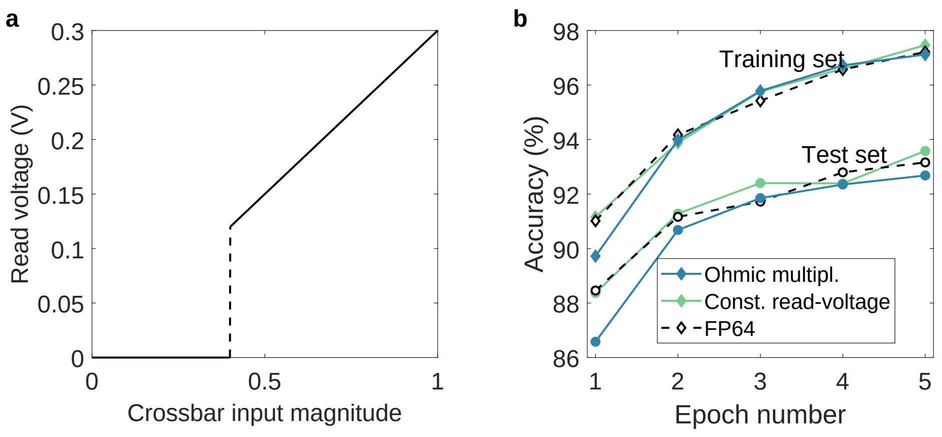


Figure 5.6 Experimental training and test set classification accuracy, when the network in Figure 3a is trained with 5,000 training images and 10,000 test images from the data set where the on-chip device conductance response is used implement the multiplication in the weighted summation operation in the forward and backward propagation.

platform, this conductance read operation is very time-consuming. Hence, the training experiment which implements Ohmic multiplication was performed using a subset of the original MNIST database. We used the first 5,000 images for the training and the entire 10,000 test images for determining the test accuracies. The network was trained for 5 epochs. The resulting classification performance on the training and test set is compared with a similar experiment using the same training set, however, was performed without the on-chip ohmic multiplication (Figure 5.6b). This is equivalent to using a constant read voltage for the reference experiment. We observed that the PCM non-linear response did not affect the training efficacy of the mixed-precision approach.

5.2.4 Weight refresh methods

When the neural network weights are implemented as the difference of two PCM devices ($W = \beta(G_p - G_n)$), the weight increment is achieved by a partial-SET pulse to the G_p device and weight decrement is achieved by a partial-SET pulse to the G_n device. This leads to monotonic conductance increase for all the devices in the

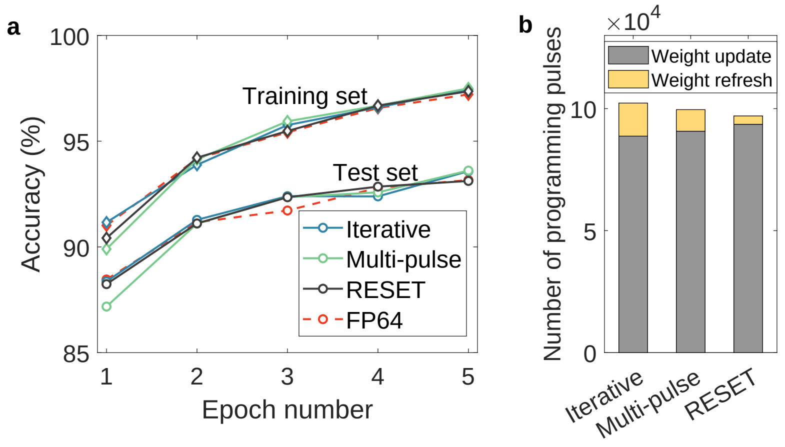


Figure 5.7 **a** A comparative study on the impact of different refresh schemes on training and test accuracies using on-chip training experiment. The training experiment was performed with 5000 training images and 10,000 test images from MNIST. **b** An estimate of the number of programming pulses used in the iterative, multi-pulse, and RESET weight refresh schemes. For iterative programming, an average of four pulses per device programming is assumed. The programming pulses for weight refresh involve both the pulses used for the RESET and successive programming pulses applied (none in RESET case) to re-program the device.

network, and the devices receiving a sufficiently large number of programming pulses will eventually saturate. This necessitates periodic detection of saturated devices and their reprogramming. In our training experiment, the weight refresh is implemented as follows. After every 100 training examples, all the device conductance pairs are read. The conductance pair is checked if one of them is above 8 μS and if the difference between them is less than 6 μS. When this condition is met, a weight refresh operation is performed. An ideal weight refresh operation will RESET both G_n and G_p and their original conductance difference $G_p - G_n$ is reprogrammed to one of these devices based on the sign of the difference. However, PCM programming is a stochastic process. While the RESET could be done in using a single pulse of sufficient amplitude, the conductance reprogramming to a reasonable accuracy could take many write-read-verify cycles and hence could be expensive. Here, we experimentally verify the impact of simpler conductance reprogramming methods.

We repeated the MNIST based training experiment with actual PCM devices in the array, with three different reprogramming methods. Here, a subset of the original dataset was used to train the network. We trained the network using 5,000 images for 5 epochs and tested using 10,000 test images. In the first case, the conductance reprogramming used the iterative method to accurately reprogram the difference [90]. In the second case, the conductance difference was divided by the average conductance change to determine the number of programming pulses to approximately reprogram the difference. Here, we blindly applied a few programming pulses without a read-verify operation. In the third case, we completely avoided the reprogramming, and the weight refresh operation involved only the conductance RESET. The resulting experimental training and test accuracies are plotted in Figure 5.7. The corresponding high-precision training results are also shown for reference. The classification performance from the three schemes almost overlap. Further, in this experiment, where training was performed using a subset of actual training data, the experimental classification performance was overlapping the high-precision results.

This result indicates that the weight refresh operation could be as simple as resetting a few devices in the array. This RESET based weight refresh scheme brings the devices out of saturation and allows further updates. We further observed that the number of programming pulses necessary for the weight update increased incrementally as we go from iterative refresh to multi-pulse refresh to RESET based refresh. However, if we consider the complexity and delay associated with the reprogramming in the weight refresh scheme, a RESET based weight refresh could be a more suitable solution, as the reprogramming of the weights, in this case, will automatically happen during the succeeding training epochs.

5.2.5 Training experiment using non-differential PCM synapse

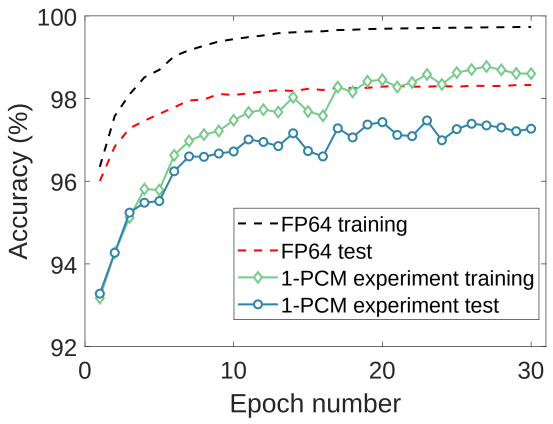


Figure 5.8 Training and test accuracies from the MPA training experiment for MNIST digit classification using the network in Figure 5.1a, where weight increments and decrements are implemented by programming a single PCM device per synapse. Irrespective of the conductance change asymmetry, the training achieves a maximum training accuracy of 98.77% and a maximum test accuracy 97.47% in 30 epochs, within 1% of that obtained in corresponding high-precision (FP64) training. FP64 training has a training and test accuracy of 99.73% and 98.33%. This experiment demonstrates the ability of the MPA to compensate the conductance update asymmetry often observed in the nano-scale memristive devices.

Next, we experimentally evaluated the MPA's efficacy to compensate large device asymmetries while training neural networks. Abrupt RESET behavior of the PCM makes its conductance potentiation and depression behavior highly asymmetric and hence, we use the differential configuration with an occasional weight refresh. A non-differential single PCM configuration for the synapse in the mixed-precision training architecture will enable us to implement both weight increment and decrement by programming the same device in either direction and hence avoid the conductance saturation and associated weight refresh overhead. We demonstrated the feasibility of this approach in Chapter 4 using linear device models. The main challenges in realizing it experimentally using PCM devices is the conductance drift (specific to PCM) and the difficulty in achieving narrow initial conductance distributions.

We conducted the same training experiment as before for the MNIST digit classification, except that now each synapse was realized using a single PCM with a reference level to realize bipolar weights. The experiment requires 198,760 PCM devices. Potentiation is implemented by $90\,\mu$A, 50 ns partial SET pulses and depression is implemented using $400\,\mu$A, 50 ns RESET pulses. In the mixed-precision architecture, this asymmetric conductance update behavior is compensated by using different ϵs for the two polarities of updates. We used ϵ_P corresponding to $0.77\,\mu$S for weight increment and ϵ_D corresponding to $8\,\mu$S for weight decrement.

To represent the positive and negative weight ranges using the positive conductance a reference level (G_{ref}) is necessary. Then the network weights $W \propto (G - G_{ref})$. Ideally, the reference levels could be fabricated using any resistive device which is one time programmed to the necessary conductance level. In the case of PCM devices, due to its conductance drift, the reference levels must also be implemented using the PCM technology which follows the same average drift behavior of the devices that represent the synapses. In this experiment, we used the average conductance of all the PCM devices read from the array to represent the G_{ref}. While this average based in G_{ref} is impractical in real implementation due to the excessive overhead if this experiment succeeds it demonstrate the MPA tolerance to device asymmetry. There are research results indicating the significant reduction in PCM drift via structural and material engineering, and this could lead to a simpler implementation of the reference level in the future. Also, we could explore the possibility to design neural networks with positive weights alone.

Now, for the current experiment, the PCM devices including those used for the reference levels (which is same as the synaptic PCMs in this experiment) need to be initialized to the middle of its conductance range. The PCMs were iteratively initialized to a distribution with mean conductance of $4.5\,\mu$S and standard deviation of $1.25\,\mu$S. While a narrower distribution was desirable [118], it was difficult to achieve in the chosen initialization range. However, this was compensated by mapping the conductance values to a narrower weight range at the beginning of the training which is progressively relaxed over the next few epochs. The conductance range [0.1, 8] μS is mapped to [-0.7, 0.7] during the epoch 1, which is incremented in uniform steps/epoch to [-1, 1] by epoch 3 and is held constant thereafter. Also, the mean of the initial conductance distribution was chosen slightly higher than the middle of the conductance range to compensate for the eventual conductance drift.

The classification performance of the experiment on training and test set for the 30 epochs of training is shown in Figure 5.8. Maximum test accuracy of 97.27% which is approximately within 1 % of that obtained in high-precision training was achieved within 30 training epochs. These results conclusively show the efficacy of the mixed-precision training approach and provide a pathway to overcome the stringent requirements on the device update precision (10 bit) and symmetry (within 2%) thought to be necessary to achieve high performance from memristor based learning systems [72, 73, 120, 109].

5.3 Training Deeper Neural Networks

We further evaluated the efficacy of the mixed-precision architecture to successfully train deeper neural networks to perform more complex tasks in collaboration with Christophe Piveteau and Vinay Joshi at IBM who developed a mixed-precision training simulator based on the tensorflow deep learning framework. Christophe tested the architecture on a convolutional neural network (CNN) for classifying CIFAR-10 dataset images and a long-short-term-memory (LSTM) network for character level language modeling (using Penn Treebank dataset). Vinay demonstrated the training of a generative-adversarial network (GAN) for image synthesis based on the MNIST dataset using the MPA architecture. A brief description of the methods, main results, and observations are provided here.

The simulator was implemented as an extension to the Tensorflow deep learning framework. The network synapses were realized using the PCM model we developed in Chapter 3. Custom tensorflow operations were implemented that take into account the various attributes of the PCM devices such as conductance range, read noise, and conductance drift as well as the characteristics of the data converters.

The CNN network we trained has approximately 1.5 million trainable parameters. The convolution layer weights are mapped to the computational memory crossbar array by unrolling the filter weights and stacking them to form a 2D matrix [121]. The network is trained using 50,000 CIFAR-10 images with a batch size of 100 for 400 epochs. We observed that the test accuracy of the CNN on CIFAR-10 (86.70%) trained via MPA eventually exceeds that obtained from equivalent high-precision training using 32-bit floating point precision (FP32) (86.10%). This was achieved while having a significantly lower training accuracy and is suggestive of some highly beneficial regularization effects arising from the use of stochastic memristive devices to represent synaptic weights.

LSTMs are a class of recurrent neural networks used mainly for language modeling and temporal sequence learning. LSTM cells are a natural fit for crossbar arrays, as they basically consist of fully-connected layers. The network is trained using a popular benchmark dataset called Penn Treebank. The network is trained to predict the next character in a sequence from the 50 character alphabet from the dataset. The characters are represented using one-hot coding. The network with 3.3 million trainable parameters has two LSTM modules stacked with a final fully connected layer. The mixed-precision training performance of the LSTM network was evaluated using a perplexity metric defined as 2 raised to the power of average cross-entropy. The average cross-entropy represents the effective number of bits per character. While the test perplexity from MPA and FP32 training are comparable, the difference between training and test perplexities is significantly smaller in MPA suggesting regularization effects similar to those observed in the CNNs.

GANs are generative neural networks trained using a recently proposed adversarial training method [122]. The performance of the generator to replicate the training data set (MNIST) distribution is evaluated using a Frechet distance (FD) metric. MPA-based training achieved FD close to FP32-based training. The training of GANs is particularly sensitive to the mini-batch size and the choice of optimizers, even when training in FP32. We observed that the solution converges to an optimal value only in the case of stochastic gradient descent (SGD) with a momentum optimizer and a mini-batch size of 100; compared to the cases where a momentum optimizer was avoided or batch size of one was used. Compared to alternate in-memory computing approaches where both the propagations and weight updates are performed in the analog domain [109], a significant advantage of the proposed mixed-precision approach is its ability to seamlessly incorporate these more sophisticated optimizers as well as the use of batch sizes larger than one during training.

5.4 Mixed-Precision Weight Update

In the mixed-precision architecture, the matrix products corresponding to the forward and backward propagation are computed using the parallel computing architecture provided by the computational memory arrays, which could provide more than two orders of computational efficiency improvement compared to the conventional digital implementation. However, the weight update stage in the mixed-precision architecture is implemented in the digital domain and is also crucial in determining the overall acceleration and computational efficiency. In this section, we discuss a mixed-precision implementation of the weight update stage, which could be optimized to attain further improvements in efficiency.

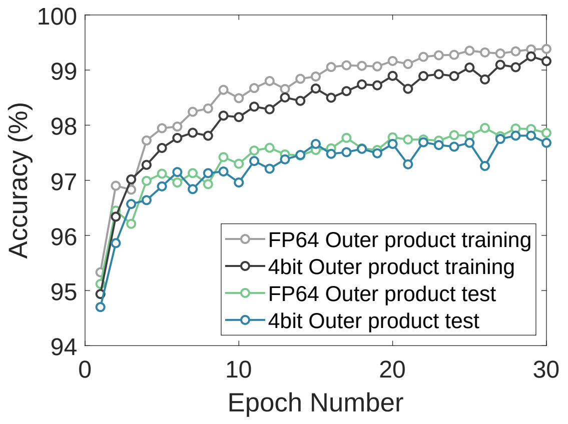


Figure 5.9 The training and test accuracies are compared for MPA based training when the outer product for the weight update is implemented using 4-bit versions of the neuron activations and errors in contrast to 64-bit floating-point (FP64) in the two layer MNIST classification problem.

Weight update computation involves determining the gradient direction and their relative magnitude with respect to other weights and then scaling it to an appropriate magnitude (using the learning rate) and accumulating over the existing weights. We observed that the precision required for these stages are different. The first stage could be implemented using a few bits of precision (e.g. ~ 4 bit), while the subsequent scaling could be approximated by bit shift operations. The direction of gradient and its relative magnitude is estimated by the outer-product of the neuron activations and the back-propagated errors. If the errors and activations are approximated using low bit precision, the resulting gradient matrix obtained as the outer product of these vectors will be sparse. The subsequent gradient accumulation requiring the highest precision ($>= 16$ bit) hence becomes sparse. Often, due to the large size of the weight matrices, they need to be stored in the off-chip memory. The sparse gradient accumulation in our mixed-precision approach will lead to a significant reduction in this energy-hungry off-chip memory access, reducing the overall complexity of the weight-update stage. Also, the low-precision representation of the neuron activation and errors will reduce the complexity of the multiplier (area and computation time) and the on-chip memory storage. Figure 5.9 show the negligible performance drop observed when the weight update is implemented using low-precision representation of the neuron activations and error vectors in MPA for the training problems such as MNIST digit classification.

5.5 Independent Training of Neural Networks

In the training simulations and experiments so far, we have used a fixed scaling factor for mapping the device conductance to the network weights. The scaling factor was determined based on the corresponding high-precision training implementation. However, for the computational memory based architecture to function as an independent deep learning accelerator, it is essential to determine the mapping between device conductance and the desired network weights automatically. Here, we discuss how this can be achieved by an adaptive scaling at the array periphery. In this scheme, the scaling factor becomes also a trainable parameter learned via backpropagation. Further, computational memory arrays should be able to represent different network architectures to be an independent training engine. Therefore, we also illustrate in this section how crossbar replicas can be used to improve the efficiency of pipeline processing in neural networks such as in CNNs.

5.5.1 Adaptive conductance to weight mapping

The in-memory computations in the crossbar array are in the analog domain. The backpropagation based training algorithm necessitates storing intermediate results in the digital domain. As a result, an analog-digital conversion is necessary between the computational memory and digital processing units and the resulting scaling need to be taken into account while mapping conductance to weights. Figure 5.10a shows the basic operation that happens when a crossbar array is used to perform a multiply-accumulate operation where the connection strengths are realized using

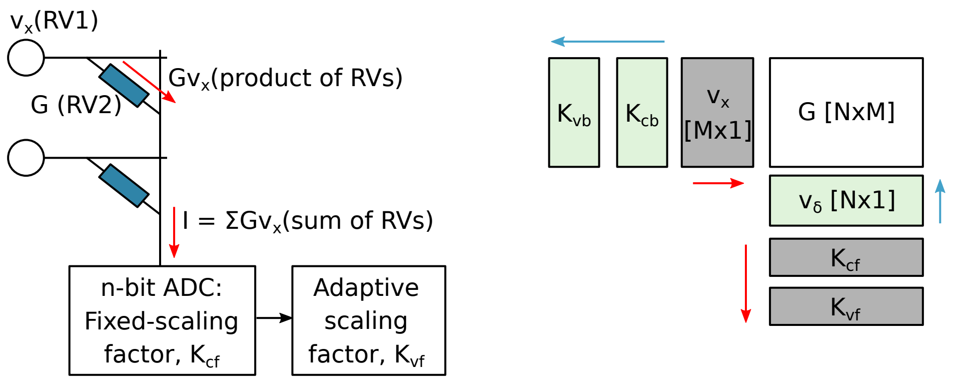


Figure 5.10 The multiply-accumulate operation can be viewed as the product and sum of random variables. The analog to digital conversion correspond to a scaling (with a quantization). An additional adaptive scaling factor is necessary to map the device conductance to actual network weights. In a backpropagation based training it is necessary to ensure that the scaling in the forward and backward propagation should be the same. Therefore, $K_{cf} \times K_{vf} = K_{cb} \times K_{vb}$.

nanoscale memory devices. In contrast to floating-point numbers, they have a limited dynamic range. Based on the physical nature of the device and chosen programming conditions, it is feasible to evaluate the statistical nature of the conductance distributions beforehand. Hence, assume that each conductance represents a random number G of known statistics. Note that the actual conductance distribution will also depend on the training problem. The approximate statistics of the input to the crossbar array depends on the training dataset and the activation function of the input neurons. Therefore, let the input which is applied as a voltage to the array be represented as a second random variable v_x.

Then the variance of the product of these two random variables is

$$var(GV) = var(G)var(v_x) + var(G)(E(v_x))^2 + var(v_x)(E(G))^2 \tag{5.2}$$

and the variance of the sum of the products is

$$var\left(\sum Gv_x\right) = \sum var(Gv_x). \tag{5.3}$$

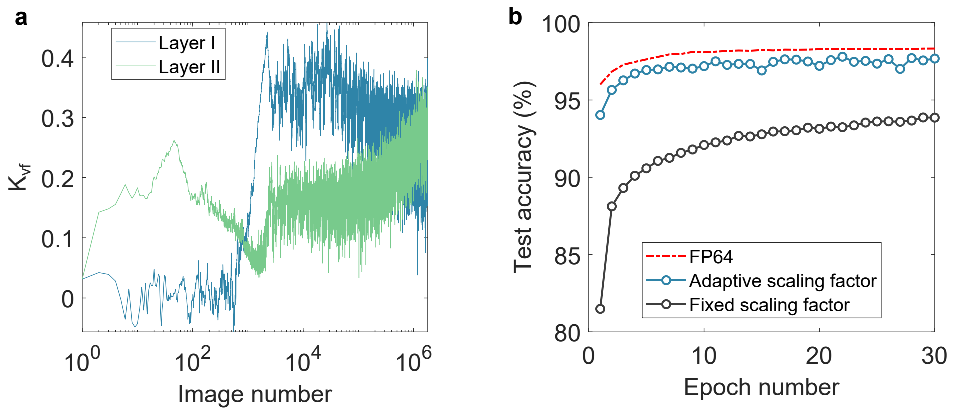


Figure 5.11 **a** The evolution of K_{cf} in the two layers of the neural network during its training to classify the MNIST dataset. **b** The training performance from two identical networks initialized to the same state, when trained with an adaptive and fixed scaling factor. The classification accuracy on the 10,000 test images is plotted.

Equation 5.3 gives the variance of the current which represents the multiply-accumulate result from one of the bit-lines. The standard deviation of the output current is proportional to the square root of the number of word lines. This information could be used to tune the input range of the ADC at the periphery. The design of ADC will be simpler if the output current has a zero mean. This means that the goal of the synaptic array design should be to achieve symmetric conductance distribution around zero. This could be achieved if both the positive and negative half of a differential synapse configuration is implemented in the same array and the required subtraction is performed in the analog domain. Now, for an n-bit signed representation, the ADC output will be in the range $[-2^{n-1}, 2^{n-1}-1]$. This translation of the input current to output digital numbers in ADC represents a fixed scaling factor, say K_{cf} in the forward propagation path. An adaptive scaling factor K_{vf} is used next to scale the ADC output as appropriate input for the neurons. K_{vf} should be initialized such that the neuron input from array lies in the range of non-zero gradient for the neuron activation function. K_{vf} is adjusted using gradient descent during backpropagation based training.

The conductance-to-weight scaling factor should remain the same during the forward and backward propagation for any weight layer. However, the optimum scaling factor of the ADCs in the forward and backward path will differ as they respectively are functions of the number of word lines (number of input neurons) and bit lines (number of output neurons) which are not necessarily equal all the time. Hence the corresponding scaling factors, K_{cb} and K_{vb}, for the backward propagation need to be adjusted. For example, if m and n represent the number of input and output neurons for the weight layer then we have,

$$K_{cf} \propto \frac{1}{\sqrt{m}} \tag{5.4}$$

$$K_{cb} \propto \frac{1}{\sqrt{n}} \tag{5.5}$$

$$K_{vb} = K_{vf}\frac{\sqrt{n}}{\sqrt{m}} \tag{5.6}$$

The proposed scheme for the initialization and adaptation for the scaling factor for the crossbar peripheral circuits is tested using a training simulation for classifying the MNIST handwritten images using the two-layer neural network and the PCM model-based computational memory. An 8-bit quantization is assumed. The PCM conductances were initialized to a distribution of mean 1.6 μS and standard deviation 0.83 μS. The conductance updates were supposed to be programmed via 90 μA, 50 ns pulses. From the cumulative conductance evolution characteristics for this pulse stream, the standard deviation of the conductance after a few pulses remain approximately at 2.5 μS. The training images normalized between zero and one is assumed to represent the input voltages to the crossbar for the first layer and hence its distribution is determined from the training dataset. For the second layer input, a mean of 0.5 and a standard deviation of 0.4 is assumed for the sigmoid neuron output. For the backpropagated normalized error a mean of 0 and a standard deviation of 0.4 is assumed. These numbers were used to determine the input ranges for the ADCs in the two layers in the forward and backward direction. The ADC output range, fixed by its bit precision, was scaled to represent the input for synaptic neurons. From Equation 2.3 shows the weight update is a function of the error, δ computed at the input of neurons. δ is proportional to the gradient of the neuron activation function. Hence, weight should be initialized to map the net input to the neurons where the neuron activation function have a non-zero gradient so that the weights receive non-zero updates and take part in the learning. For the sigmoid neurons, this non-zero gradient lies approximately in the range [-7 7]. We initialized K_{vf} to map the ADC outputs to a neuron input range of [-4 4] for a slightly higher gradient, at the beginning of training. Subsequently, K_{vf} for the two layers are trained using backpropagated gradients with a learning rate of 0.01. The device conductance was trained using a learning rate of 3.2 which was determined by scanning a range of learning rates during training. The evolution of the K_{vf} in the two weight layers of the neural network is shown in Figure 5.11a. Figure 5.11b shows maximum test accuracy of 97.8% which is comparable to what we obtained by carefully choosing the conductance to weight mapping in Section 5.2.

5.5.2 Crossbar replication for pipeline processing

Dedicated computational memory arrays for all the layers in the network allows multiple layers to be processed in parallel. However, for the pipeline to be efficient, the delay in each of the layers should be comparable. For example, in CNN, the number of sub-regions of the images that are being convolved is significantly different in each convolution layer. The known implementation of convolution layers in the computational memory require each sub-regions of the input images to be applied as a vector to the crossbar array representing the filter coefficients [121]. This leads to a significant mismatch in delay between layers, especially when large matrix-matrix multiplications are being implemented as repeated vector-matrix multiplications in

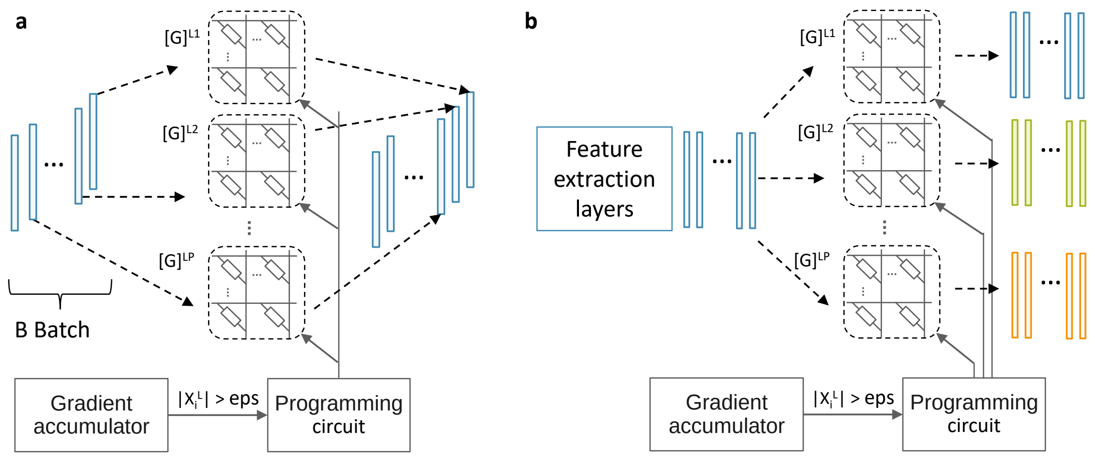


Figure 5.12 **a** The replicas of the crossbar arrays could be used to process B batches of input vectors in parallel in any weight layer. The batches of vectors might be from different training data or the different subregions of an image in a CNN. **b** The final fully connected layer of a CNN could be replicated and trained to perform different tasks based on the output of shared feature extraction layers.

a crossbar array. However, this could be solved to a certain extent by replicating the relatively inexpensive computational arrays. In a mixed-precision architecture, the weight update accumulator could be shared between the replicas (Figure 5.12a). Ideally, these replicas are expected to represent identical conductance matrices, hence initialized and trained using the same sequence of programming pulses. Meanwhile, due to device programming stochasticity, each array will be slightly different. The effect of this stochasticity needs further analysis. The error resilient nature of the neural network could be expected to tolerate the statistical difference between the replicas.

Another possible way to make use of computational memory replication is to share the resources of a neural network to perform multiple tasks. In the notion of transfer learning, the convolution layers trained for one task could be used for another task by retraining just the final fully connected layers. Here, the replicated fully connected layers could be trained separately such that the features extracted using the convolution layers could be used to make different decisions simultaneously (Figure 5.12b).

5.6 Performance Assessment

In this section, we present a first-order assessment of the computational advantages of the MPA compared to the traditional high-precision training engines. Further, we discuss methods for improvements.

In a high precision implementation of DNN training, if z denotes the fraction of computational resources that could be implemented using computational memory and $1-z$ denotes the fraction of conventional computational resources required for weight update (calculation of desired weight change and it's accumulation), then the overall computational advantage is given by $1/\left(\frac{z}{A}+\frac{1-z}{B}\right)$, where A represents the efficiency improvement that can be achieved due to the computational memory implementation of the data propagation stages and B represents the efficiency improvement that can be achieved for the remaining stages in the digital domain. The assumption being made is that the additional overhead arising from MPA is negligible. The main overhead is the need to program the devices. However, it can be seen that the number of devices programmed per training example is highly sparse depending on the device granularity [69]. For example, in the MNIST based training experiment, on average, less than two devices per image were updated. First, we investigate the efficiency gain, A, due to the use of computational memory for the forward and backward propagation stages. The main advantage is that in a conventional computing unit, the weights need to be moved into a processing unit and processed whereas, in the computational memory approach, the PCM devices can store the weights as well as perform the matrix-vector multiplications using them without the associated data movement. Moreover, the multipliers associated with the conventional approach will have a significantly large area footprint. For example, the area of a 16-bit floating point MAC is on the order of $10^6\,F^2$. The area consumed by a single PCM device is around $25\,F^2$. On a first order estimation, more than a hundred of such 1024×1024 crossbar arrays of $25\,F^2$ PCM cells could be fit in $2\,\%$ of an $815\,\mathrm{mm}^2$ chip area in

14nm technology node, where we assumed 20 % array efficiency with the remaining 80 % composed of the control circuitry. Hence, we could assume that $A >> 100$ if we consider both the area and energy efficiency. As a result, the bottleneck arises from the weight update stage and its implementation decides the overall computational efficiency of the mixed-precision architecture. In a high-precision implementation, the weight update stage will constitute approximately 40% or smaller of the overall computational requirements. Hence, the minimum computational advantage arising from the mixed-precision approach is approximately 2.5. However, we demonstrated the feasibility of a mixed-precision weight update stage to successfully train neural networks. If the backpropagated errors are normalized the neuron activations and the errors could be represented in low-bit precision and hence during the outer product of them will lead to highly sparse weight update matrices. This leads to reduced memory requirements to store intermediate results and much lower off-chip memory access to update the large weight matrices or weight update accumulators. Furthermore, the re-normalization and learning rate multiplications could be implemented as relatively inexpensive bit-shift operations. For example, in the MNIST training simulation, a 3-bit representation during the weight update computation was sufficient to reduce the weight update count to less than 10% with negligible loss in test performance. This leads to a $10\times$ acceleration of the weight update stage ($B = 10$) and hence a potential overall 20 to $25\times$ computational advantage for MPA compared to the high-precision approach.

The computational memory based MPA architecture could be more efficient compared to an entirely digital mixed-precision implementation using application specific integrated circuits (ASICs). The relative advantage could come from the area efficiency of the PCM array and the in-memory computation. An ASIC implementation where the required matrix products are computed using an array of low-precision MAC units will occupy a significantly larger chip area compared to a computational memory array, and still will have to access the high-precision memory for the weights and convert them to the low-precision version before performing the actual computation. A high-precision memory accumulating the gradients will be an essential part of both the implementations here as they are based on backpropagation algorithm. Additional resource utilization and pipeline improvements are possible in computational memory implementation if we can afford one batch delay in weight update. Here, we suggest using the dense computational arrays with local weight storage to perform the forward and backward data-propagations for one batch while the remaining chip area could be utilized for weight update computations for the previous batch of training data. This one-batch delay in the weight update is observed to have a negligible impact on the training performance on the MNIST benchmark problem. This pipelining is much less efficient in a fully ASIC implementation as the same resources need to be shared between the data-propagation and weight update stages.

We have discussed the computational efficiency of the MPA architecture as a training accelerator. The trainability of the computational memory also makes them an attractive candidate for systems that learn throughout its life while acting in an inference mode predominantly. It is projected that such inference modes can have at least two orders of magnitude better efficiency compared to its high-precision counterparts [73]. Hence, the effective system efficiency gain in such cases will be determined by the fractional lifetime it spends in the training and inference phase.

5.7 Summary

The efficacy of the mixed-precision deep learning architecture to achieve floating-point comparable performance using low-precision nanoscale devices is experimentally demonstrated using PCM devices in a million device array on the benchmark problem of handwritten digit classification based on MNIST dataset. We demonstrated on-chip inference using PCM weights for a period of a month with a negligible drop in the trained performance over time. We also experimentally demonstrated the feasibility of using a highly asymmetric device for synapse using a non-differential single device configuration for training which achieved performance within 1 % of that from floating-point training. Further, the generalizability of the architecture to more complex problems such as image recognition, language modeling, and image synthesis based on CNN, LSTM, and GAN networks is analyzed using a statistically accurate stochastic switching model of PCM, demonstrating network performance which is comparable with floating-point baseline. This also points to the flexibility of the architecture to adapt to different training optimizers. We show that the hardware efficiency of nanoscale computational memories can be maintained while preserving the accuracy of floating-point software simulations if the gradient information is accumulated with sufficient precision in the digital domain and transferred to the device when it becomes comparable to the update precision. The architecture significantly relaxes the device requirements, particularly those related to linearity, variability, and update precision to realize high-performance learning machines. The sparse device programming in the accumulate and transfer scheme also improves the overall reliability and endurance of the nanoscale devices. We further show evidence for an inherent regularization originating from the non-linear and stochastic behavior of such nanoscale devices in this mixed-precision architecture. We also provide pathways to optimize the weight update stage in the digital domain through a low precision representation of the neuron activations and back-propagated errors with dynamic scaling and high-precision accumulation, which together reduces the computation cost, area, and memory access. The architecture is flexible enough for a wide class of deep learning training methods and algorithms beyond the simple stochastic gradient descent and could potentially have a $20\times$ computational advantage compared to the corresponding high-precision implementation.

CHAPTER 6

TRAINING OF SPIKING NEURAL NETWORK USING PHASE-CHANGE MEMORY ARRAYS

6.1 Introduction

ANNs and SNNs represent different levels of abstractions of the biological neural network which are being explored to implement intelligent computing engines. Even though SNNs mimic the biological networks more closely, ANNs remain the choice of machine learning community due to the difficulty in implementing SNNs efficiently in von-Neumann machines and the direct non-applicability of the gradient descent training algorithms to SNNs. In contrast to ANNs where information is represented in real numbers of high-precision, information in SNNs is coded as a stream of spikes [123, 124]. Each neuron in an SNN integrates these asynchronous spike streams from a large number of input neurons temporally to determine its spike response. The requirement to emulate the behavior of a large number of neurons with complex temporal integration behavior and with an asynchronous inter-neuron communication calls for a fully parallel nano-scale device hardware implementation. A computational memory based implementation where non-volatile memory devices provide dedicated connectivity between layers of neurons (Figure 6.1) analogous to the biological network could lead to an efficient implementation of SNNs as well.

This work experimentally demonstrates on-chip learning of a single layer SNN, which is trained to convert the spikes generated from an audio signal to corresponding images of the characters pronounced. The network synapses are implemented using eight PCM devices in a differential configuration. We evaluate the suitability of the stochastic PCM to represent synapses in an SNN trained to generate spikes at precise times rather than an average spike rate. Effect of multi-PCM synapse configuration and mixed-precision training in improving the training accuracy is

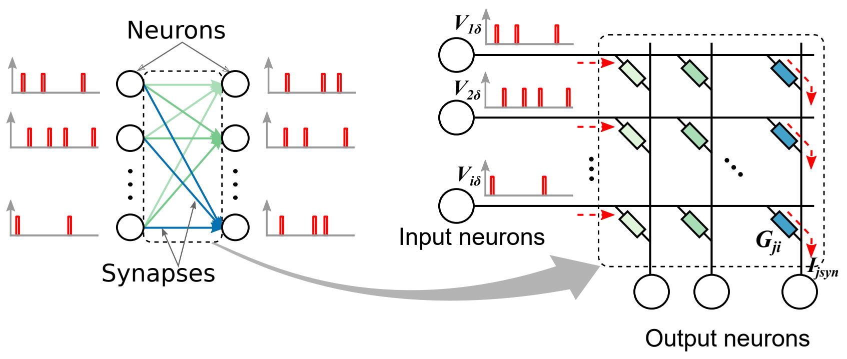


Figure 6.1 Neurons in the SNN integrates (both spatially and temporally) asynchronous spike streams from a large number of input neurons to produce its response. Dedicated connections provided by the non-volatile memory array is an efficient parallel and scalable implementation.

analyzed using faithful stochastic model capturing the device programming behavior. Further, the effect of drift is analyzed in the SNN inference and a compensation method is proposed.

6.2 PCM Conductance Drift

In this section, the behavior of conductance drift is analyzed in different synapse configurations, devised to compensate other non-ideal characteristics of PCM. For example, a multi-PCM configuration has been proposed to improve the granularity of the conductance update in synapse [95]. Here, the synapse conductance is the sum of multiple PCM devices and during each update, only one of the devices is updated chosen by a global arbitration clock. The asymmetric conductance update due to the abrupt RESET transition in the PCM devices are compensated by a differential configuration, where each synapse is realized as the difference of two PCM conductance values, i.e., $W = \beta(G_p - G_n)$, where β is a scalar multiplier [27]. This allows the weight increments and decrements to be implemented by partial-SET pulses applied respectively to G_p or G_n devices.

Conductance drift is another aspect of the PCM that could potentially affect the training performance and the networks' ability to retain the trained state. The

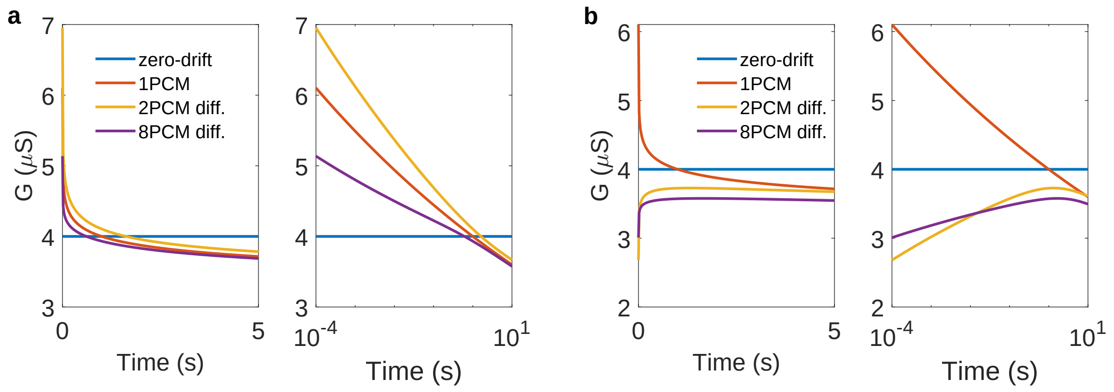


Figure 6.2 Effect of conductance drift in PCM synapses. **a** A single PCM in the positive half in the differential configuration receives the programming pulse. The effective synapse conductance evolution is shown in linear (left) and logarithmic (right) time scale. **b** A single PCM in the negative half in the differential configuration receives the programming pulse.

rate of change of conductance from drift can be approximated from Equation (3.3) for $\nu << 1$ as,

$$\frac{dG(t)}{dt} = \frac{-\nu}{t}\frac{G(t_0)}{t_0^{-\nu}}. \tag{6.1}$$

Equation (6.1) shows that the rate of change of conductance is proportional to the drift coefficient and conductance, and inversely proportional to the time elapsed since programming. It indicates that while at smaller time scales, the conductance of individual devices will determine the rate of conductance decay, but as time grows, the rate of conductance decrease drops exponentially.

The impact of this drift behavior in three exemplary synapse configurations - with a single PCM, 2-PCM in a differential configuration, and an 8-PCM in a differential configuration - is analyzed in Figure 6.2. A drift-free conductance level representing $4\,\mu$S is shown for reference. The PCM in the single PCM synapse is also assumed to have been programmed to a state with a conductance of $4\,\mu$S at time t_0. The same conductance level is represented in 2 PCM differential synapse using $G_p = 6\,\mu$S and $G_n = 2\,\mu$S. In the 8 PCM synapse configuration, $G_p = 2\,\mu$S for the four positive devices and $G_n = 1\,\mu$S for the negative ones. Such uniform distribution of the conductances in a multi-PCM synapse is feasible in reality by using a clock scheme to distribute the programming pulses among the devices [95]. In Figure 6.2a we assume that these conductance levels were achieved via a programming pulse applied to one of the PCMs (in the positive half in the differential configurations) at time zero. Other PCMs are assumed to have received programming pulses a few seconds earlier with random offsets. The effective conductance evolution plot shows that the 2-PCM differential configuration which had the highest conductance in an individual PCM shows the highest rate of decay, while the multi-PCM wherein which each individual PCM had smaller conductance levels had smaller slopes. Figure 6.2b shows a similar situation when a PCM in the negative half in the differentially configured synapse received the last programming pulse. Again, the ability of a multi-PCM to represent the same network weight using smaller conductance values of individual devices enable it to have smaller overall conductance drift. However, in this discussion, the assumption that the $4\,\mu$S PCM to represent a $4\,\mu$S synapse conductance is valid only if the synapses need only positive weights. To represent a bipolar weight distribution, single PCM will require an additional reference level at approximately at the middle of the device conductance range, consequently the $4\,\mu$S synapse conductance will be represented by a much higher conductance. Hence, a non-differential single PCM will have the worst performance in conductance drift. Based on our analysis here, the multi-PCM synapse with pulse amplitude modulated programming gives better granularity and lower drift and hence used for implementing the synapses in the supervised training experiment in the next section.

6.3 SNN Training Experiment

The training problem we use here is inspired by our ability to conjure objects in our imagination upon hearing its name. The underlying neural network must be

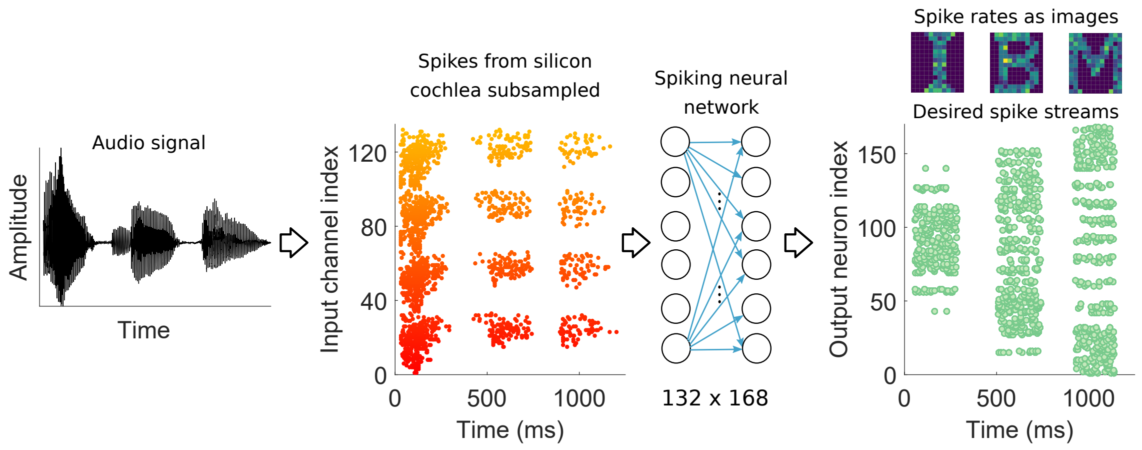


Figure 6.3 The audio signal is passed through a silicon cochlea chip to generate spike streams. These spike streams are sub-sampled and applied as an input to train the single layer SNN. The desired spike response from the networks representing the images (14×12 pixels) corresponding to the characters in the audio is shown at the output of the network.

associating the audio signal with an image representing the information contained in the audio. Therefore, we formulated it as a supervised training of an SNN whose input is the audio signal and the desired response are the images it is expected to imagine. In order to perform the training experiment in the spiking domain, we used the assistance of Dr. Shih Chii Liu, to convert the audio signal to a set of spike streams using a Silicon cochlea developed at *Institute of Neuroinformatics, Zurich*. The audio signal used says 'IBM'. Cochlear circuit has 64 band-pass filters with frequency bands logarithmically distributed from 8 Hz to 20 kHz. The cochlea circuit generated spikes representing the left and right channels. Further, due to the synchronous delta modulation scheme used to create the spikes, there were on-spikes and off-spikes. Combining all the filter responses corresponding to non-zero spiking, for left and right channels and the on and off spikes, there were 132 input spike streams. The silicon cochlea generated spikes with a time resolution of 1 μs. The spikes were further sub-sampled to a spike time resolution of 0.1 ms and to have an average spike rate of 10 Hz. A raster plot of the spikes thus generated is plotted in

Figure 6.3. The audio spikes are fed as input to the single layer neural network. The desired response of the network is generated as Poisson spike streams whose spike rates are scaled versions of the pixel intensities of a 14×12 image representing the characters 'I', 'B', and 'M'. The spike trains representing the images have an average duration of 230 ms and is mapped to the corresponding time window in the audio signal.

The 22,176 synapses in the network is realized using 177,408 PCM devices in differential configuration ($W = \beta(G_p - G_n)$). The differential configuration allows both weight potentiation and depression to be implemented using identical SET pulses resulting in similar conductance changes and it compensates the conductance update asymmetry due to the abrupt RESET observed in PCM. Each half of the differential configuration, G_p and G_n, is the sum of 4 PCM device conductances. The multi-PCM configuration increases the update resolution for a given dynamic range required for the synapse. It also reduces the effective drift observed by the network. For each synaptic update desired by the training algorithm, the device to be programmed is chosen cyclically, so that on average all devices receive an approximately equal number of pulses [95]. To further extend the conductance range offered by the individual devices and to improve the programming resolution, we used pulse amplitude modulated programming.

Each input spike stream is convolved with a current kernel $I_{ker} = (e^{-t/\tau_1} - e^{-t/\tau_2})$ and weighted with the synaptic strength W to generate a continuous time current input from each synapse. The net currents from all the input synapses are integrated by the LIF (leaky-integrate and fire) neurons to generate the output spikes. The observed output spikes are compared with the desired ones and the weight updates ΔW are generated using a supervised SNN training algorithm called NormAD [79].

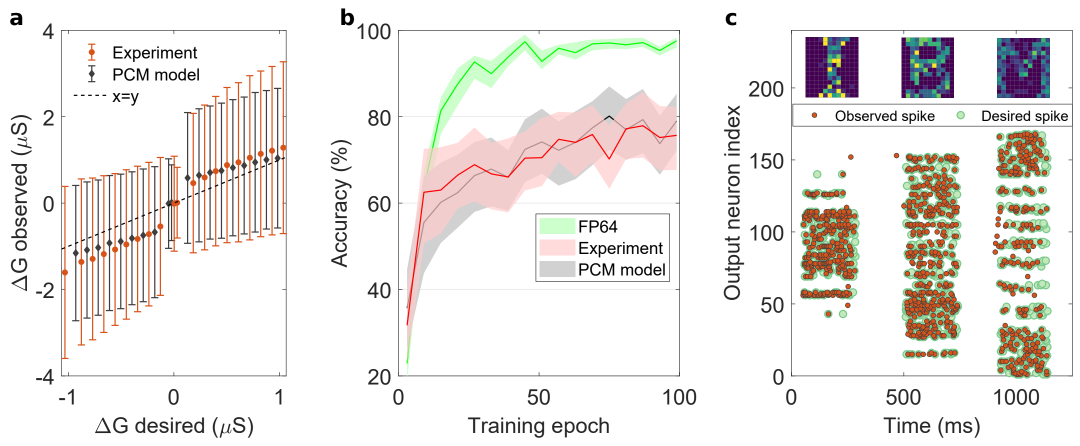


Figure 6.4 **a** The observed versus desired conductance change in the PCM from the SNN training experiment. **b** Accuracy is defined as the number of desired spikes with an observed spike generated from the network within a certain time interval. The lower bound of the shaded lines correspond to 5 ms interval, the middle line 10 ms and the upper bound 25 ms. The corresponding training simulation using the PCM model captures the experimental result very well. **c** The raster plot of the desired and observed spike trains from the trained network. A visualization of the character images whose pixel intensities are generated from the observed spike rates is also shown above the raster plot.

$$\Delta W = r \int_0^T e(t) \frac{\hat{d}(t)}{\|\hat{d}(t)\|} dt \tag{6.2}$$

where r is the learning rate, T is the pattern duration, $e(t)$ is the difference between desired and observed spike trains. $\hat{d}(t)$ is a convolution between input spike stream, I_{ker}, and an approximate impulse response of the LIF neuron. Here, the weight updates computed at each generated or desired spike instance is accumulated over the training pattern duration (one epoch) and is applied to the network. The weight updates are linearly scaled to a current pulse amplitude to program the PCM device. The maximum amplitude of the programming pulse was limited to 130 μA.

The stochastic conductance updates observed in the experiment for the desired updates are shown in Figure 6.4a. A raster plot of the spikes observed from the trained network as a function of time along with the desired spikes are shown in

Figure 6.4c. The character images shown are created using the average spike rate for the duration of each character and it indicates that the network was successfully trained to capture the images. The ability of the network to generate the spikes at the desired time instants are further evaluated using an accuracy (Figure 6.4b) defined as the percentage of the total 987 desired spikes which have an observed spike within a certain time interval. In the line plot of accuracy with shaded bounds, the lower bound, middle line, and the upper bound respectively correspond to spike time tolerance intervals of 5 ms, 10 ms and 25 ms. The average spike rate for each of the character duration was 20 Hz corresponding to an inter-arrival time of 50 ms. Hence, in this problem 25 ms can be viewed as an upper bound on the timing error to make a decision based on spike-time. The training experiment using stochastic phase-change memory devices generates more than 80% of the spikes within this error bound. The training experiment was pre-validated using the PCM programming model (Chapter 3) and it captures the experimental response accurately. The training accuracy obtained from the corresponding 64-bit floating point (FP64) training simulation is shown for reference.

6.4 Training Accuracy

Methods to improve the training performance of the SNN is evaluated using the PCM programming model. Limited update granularity of a PCM device could be improved by using multiple devices in parallel and updating one of them for each synapse weight update. Figure 6.5a shows increase in training accuracy with an increase in the total number of devices per synapse (used in differential configuration). However, increasing the number of devices beyond sixteen in this problem did not yield higher accuracy. Figure 6.5b shows that a similar accuracy can be achieved using a two PCM differential synapse if we follow a recently proposed mixed-precision training architecture [69]. Here, the weight updates are computed every time there is an

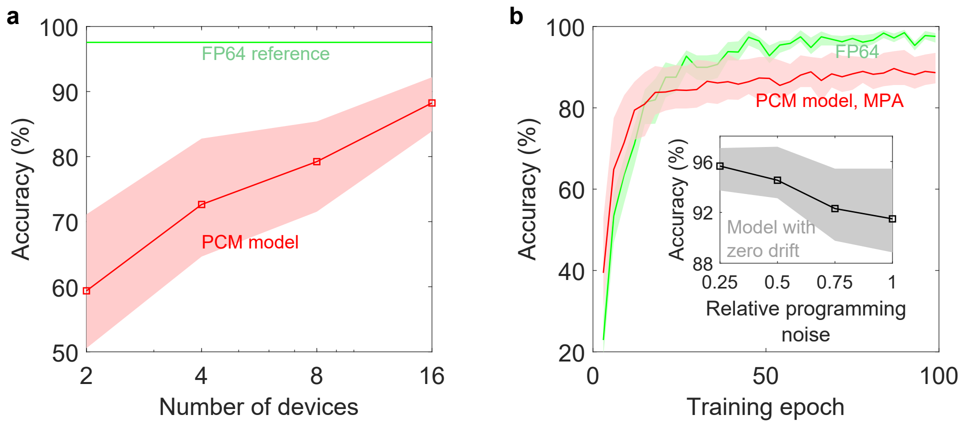


Figure 6.5 **a** Spike generation accuracy as a function of the number of deices in a multi-PCM synapse. **b** Spike generation accuracy can be improved by a mixed-precision (MPA) training approach. Accuracy when trained with PCM models with artificially scaled programming noise and without conductance drift is shown in inset.

observed or a desired spike and accumulated. When the accumulation exceeds an average device update granularity, a programming pulse (90 μA, 50 ns is assumed for the PCM model-based training simulation) is applied and the amount corresponding to the device granularity is subtracted from the accumulation. Figure 6.5b inset shows that further training improvement can be realized by device engineering to reduce the conductance drift and programming noise.

Further, the ability of a neural network to classify its inputs depends on the correlation between the inputs. In Figure 6.6 we show, using the PCM model simulation, that the accuracy gap between those from the experiment and floating point training simulation can be reduced by decreasing the correlation between the input spike streams. We added a random temporal jitter uniformly distributed in [-25 ms 25 ms] to each input spike which causes the cross-correlation between the spike streams to decrease. The correlation coefficient was determined after smoothing the spike streams using a Gaussian kernel ($e^{-t^2/2\sigma^2}$) of $\sigma = 5$ ms. This slight decrease in the input correlation which led to improvement in training accuracy suggests that encoding schemes and network structures that inherently separates input features

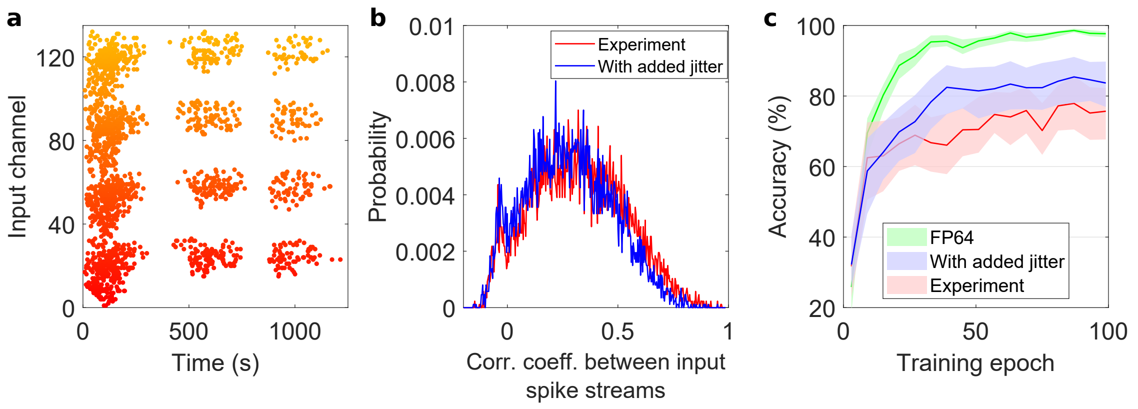


Figure 6.6 **a** Input spike streams with spike times jittered by random amounts uniformly distributed in [-25 ms 25 ms]. **b** The cross correlation between the jittered spike streams are shifted towards zero compared to the experimental input **c** The training accuracy is improved when trained with input spike streams of reduced correlation.

could potentially improve training performance using low precision devices such as PCM.

6.5 SNN Inference on PCM Array

The ability of PCM array based SNN to retain the trained state is evaluated by reading the conductances at logarithmic intervals of time and observing the network response (Figure 6.7a). Both the spike-time accuracy (Figure 6.7a) and the spike rate (Figure 6.7c) drops due to conductance drift over time. In Figure 6.7b, the black line shows the drifted conductance distribution compared to the distribution observed at the end of training. The effect of this conductance drop in terms of the network response is visible in Figure 6.7c. Here, the first row of images shows the letters 'IBM' that was created by the network using the conductance state immediately after the training and the second row shows the corresponding response after 10^5 s, which is barely recognizable compared to the initial response. The conductance decay causes the net current flowing into the neurons to decrease gradually causing the neuron spike rate to drop and eventually to stop spiking.

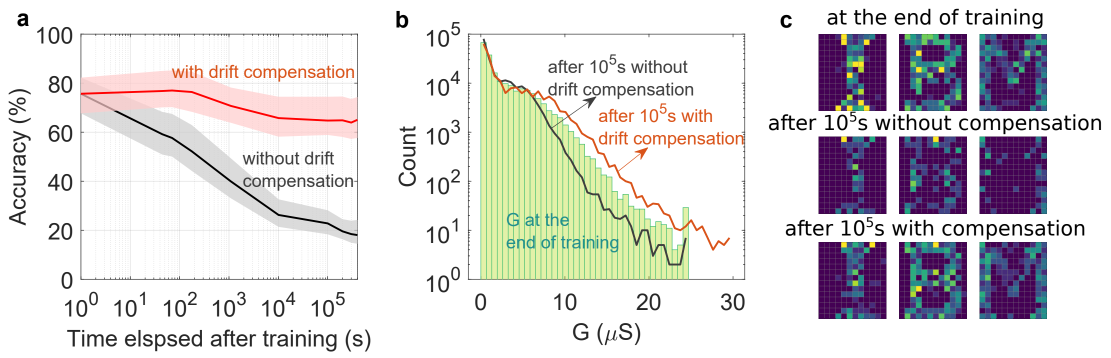


Figure 6.7 **a** Inference using trained PCM array. Due to conductance drift, the accuracy drops over time (black line). The effect of drift can be compensated by a time-aware scaling factor (red line). **b** The drifted conductance distribution at the end of 10^5s is compared with the trained conductance distribution. The effect of scaling on the drifted conductance is also shown. **c** The images generated by the SNN at the end of training for the audio input (top). The images generated after 10^5 s (middle). The images generated with drift compensation (bottom). The brightness of each pixel represents the spike rate for the duration of each character.

The conductance drift in PCM follows an empirical relation (Equation (3.3)) which captures the average device behavior. If all the conductance in the network drop by the same factor the effect of drift could be easily compensated by scaling all the crossbar output by this factor. However, different devices are programmed at different time instances during training and the conductance drift is state dependent. It has been shown that each programming event re-initializes the conductance drift (Figure 3.7) [84]. To accommodate the programming time difference between devices, Equation (3.3) may be modified as follows:

$$G(t) = G(t_0)\left(\frac{t - t_p}{t_0}\right)^{-\nu} \tag{6.3}$$

where t is a common time frame for all the devices with its origin at the beginning of training. t_p is the time any particular device receives a programming pulse and t_0 represent a constant time interval from t_p. In typical neural network training algorithms, the t_p for each device will differ and hence will start to drift from different time origins. Also, depending on whether the synapse received a potentiation or depression pulse, the effective synapse conductance could increase or decrease for a time duration which is comparable to the differences of t_ps for the devices in that differential PCM synapse. Therefore, if the solution determined by the training for the task is sensitive to the conductance drift in a time scale comparable to the total training time itself, then the solution could not be recovered by an array level scaling. On the other hand, if the solution is less sensitive to small-scale drift, then the long-term conductance drift ($\frac{t_p}{t} \rightarrow 0$) could possibly be compensated by a scaling factor proportional to t_e^{ν}, where the t_e is the time elapsed since training. Figure 6.7a shows that by scaling the conductance values at any time t_e after the training by a factor $t_e^{0.035}$ the training accuracy drop from conductance drift could be compensated.

However, note that a full recovery of the trained state was not possible by an array level scaling. The resulting scaled conductance distribution (red line) at 10^5 is shown in Figure 6.7b. The state-dependent nature of the drift coefficient causes the compensation factor to over-compensate at higher conductance ranges. The inference performance of SNN using PCM synapses could be improved by reducing conductance drift and recently demonstrated projected-PCM cell architecture with an order of magnitude lower drift coefficient is a promising step in this direction [125].

6.6 Summary

We presented an experimental demonstration of the supervised training and inference of spiking neural network using close to 180,000 phase-change memory devices. The experiment demonstrates the feasibility of using stochastic and non-linear conductance response of the device to achieve reasonable training performance. Using a statistically accurate model which captures the device non-idealities and the experiment training performance, we tested different synapse configurations for performance improvement. The limited device granularity could be compensated by multi-PCM synapse configurations. The training performance for limited granularity configurations could also be improved by the mixed-precision training approach. Further, the drift induced performance drop could be compensated by a global scaling factor, which is a function of the time elapsed since training, at the periphery of the array.

CHAPTER 7

RESISTIVE RANDOM ACCESS MEMORY FOR SYNAPSE

7.1 Introduction

We now analyze the feasibility of using resistive random access memories (RRAM) as synapses via demonstrating spike time dependent plasticity in Cu/SiO_2/W based devices and phenomenological model based SNN training simulations. Basic characterization and conductance quantization of the device are also presented from [45].

While PCM is one of the most mature non-volatile resistive memory technology and shows gradual conductance change desirable for a synaptic device, there are constraints that arise from the physical mechanism of the phase change that other class of non-volatile memories might be able to solve. The resistance changes in PCM is via heat-assisted melting and crystal growth, which require special device structure to produce local hot spots and high programming currents. Further, PCM requires a series transistor in the array to supply the large current requirements of individual devices and limits the maximum array density. RRAMs are memory devices with a metal-insulator-metal (MIM) structure whose resistance states are programmed via field driven atomic rearrangement. Their relatively simpler structure, operation, and low power programming could possibly enable a higher density and computational memory arrays with lower power. The suitability of RRAMs as a trainable synaptic element in computational memories is explored here.

In this chapter, we discuss the behavior of a Cu/SiO_2/W based RRAM fabricated at the nanofabrication facility at IIT Bombay, India. The constraint to be compatible with the current complementary metal-oxide-semiconductor (CMOS) technology is a major deciding factor in the material choices for logic and memory devices for the future. A SiO_2-based device with Cu top electrode is a suitable candidate for the low-temperature back-end-of-line (BEOL) process integration. Also, Cu is known to be highly mobile in SiO_2. The high mobility of ionic species in the dielectric is favorable for low power operation, though may have deleterious effects on the data retention capabilities. Our experiments and analysis are designed to understand the fundamental mechanisms of conductance switching in these devices, scaling limits, and directions for the material trade-offs. We demonstrate and discuss the conductance quantization in the device, demonstrates and analyzes the sub-quantization spike-timing-dependent plasticity (STDP), and develop a phenomenological model capturing the state-dependent conductance transition nature to analyze the potential of the device as a synaptic element.

7.2 Resistance Switching Mechanism

RRAMs have a dielectric of a few nanometers to a few tens of nanometer in thickness layered between a top electrode and a bottom electrode. When a programming pulse is applied across the device, the resulting high electric field forces an atomic rearrangement within the dielectric causing a resistance change. This atomic rearrangement is considered to be an electrochemical process which can be anion induced or cation induced depending on the nature of the metal electrodes and the dielectric. Anion-type RRAMs are characterized by a metal oxide dielectric and inert electrodes which are oxygen-ion active or can act as an oxygen ion reservoir during resistance switching. In these devices, the resistance transition is by migration of oxygen ions (or vacancies) which forms low-resistance conductance pathways between the electrodes and are hence referred to as Oxide RAMs (OxRAM). On the other hand in cation-type RRAMs, metallic ions from one of the active electrodes such as Ag, Cu migrates across the dielectric and get reduced at the opposite electrode forming conducting pathways. Due to the metallic nature of these bridges, they are also called conductance bridge RAM (CBRAM) [43]. Due to the relatively higher mobility of the metallic ions in dielectrics such as SiO_x and GeS_x, CBRAMs tend to have smaller operating voltages compared to OxRAMs.

The RRAMs as fabricated is in a high-resistance state (HRS) and its resistance is determined by the dielectric, its thickness, and the device area. RRAMs typically undergo a forming process which is a reversible, soft dielectric break-down induced by an applied voltage making the first resistance transition to a low resistance state (LRS). The resistance transition from the HRS to LRS is called SET programming. During a SET, atoms of sufficiently low activation energy get oxidized, migrates along the applied field and get reduced at one of the electrodes forming conduction pathways. Applying a voltage of opposite polarity causes ionic migration in the reverse direction causing the conductance paths to break returning the device back to an HRS. This reverse transition is called RESET.

7.3 Fabrication and Characterization of Cu/SiO_2/W RRAM

Our devices consist of a 10 nm SiO_2 dielectric layer sandwiched between an active Cu (100 nm) top electrode and an inert W/TiN/Ti (50 nm/10 nm/10 nm) bottom electrode, over a p-type Si(100) wafer with a SiO_2 electrical isolation layer. The W/TiN/Ti stack used as a bottom electrode was sputter-deposited and patterned with a photo-lithography and lift-off process. The 10 nm thick SiO_2 film and the Cu top electrode were then sputter-deposited over a patterned double layer photoresist (LOR3B and S1813) in a single vacuum and room temperature process. The $100\times100\,\mu m^2$ cross-point devices (Figure 7.1a) were finally obtained by a common lift-off process. The stoichiometry of the SiO_2 layer was confirmed by X-ray photoelectron spectroscopy on the Si 2p core level (contribution of Si $2p_{3/2}$ at 103.6 eV, Figure 7.1c) [126, 127, 128]. The devices were subjected to 400°C, 5 min annealing in Ar environment after top electrode patterning.

The switching response of the device was characterized by applying a voltage sweep to the Cu electrode with the W kept at ground potential using B1500 semiconductor parameter analyzer. The maximum current that flows through the device is limited to 100 μA by a current compliance circuit in the characterization tool. The device exhibits bipolar resistance switching with a pinched hysteresis loop characteristic of memristive switching, as in Figure 7.1d. The initial SET transition happened at approximately 350 mV and RESET at 125 mV. The SET voltage dropped below 250 mV after a few cycles of SET and RESET (Figure 7.1e). The bipolar switching behavior indicates that the conductance transitions in these devices are due to the field-driven motion of charged ions through the dielectric. The ionic species involved in the resistance transition was confirmed to be Cu via control experiment which was unable to exhibit this low voltage resistance switching in the absence of Cu as one of the electrodes. The actual conductance evolution trajectory will depend on factors such as magnitude and duration of programming pulses, the thickness of the dielectric layer, distribution of migrating ionic species and imperfections in the dielectric.

7.3.1 Conductance quantization

The possibility of field localization in the RRAM dielectrics, either due to its non-uniform thickness or a local defect, suggest the formation of conduction pathways constrained to a narrow region. If the lateral dimension of the pathways is comparable to Fermi wavelength (λ_F) and its length is less than the mean free path for collision, electrons will undergo ballistic transport. According to Landauer theory for ballistic electron transport, this leads to a limited number of conduction modes available for the electrons and cause quantization of measured device conductance [129, 130].

We observed occasional conductance steps during the resistance switching and suspected conductance quantization pertaining to the nano-filaments. We found

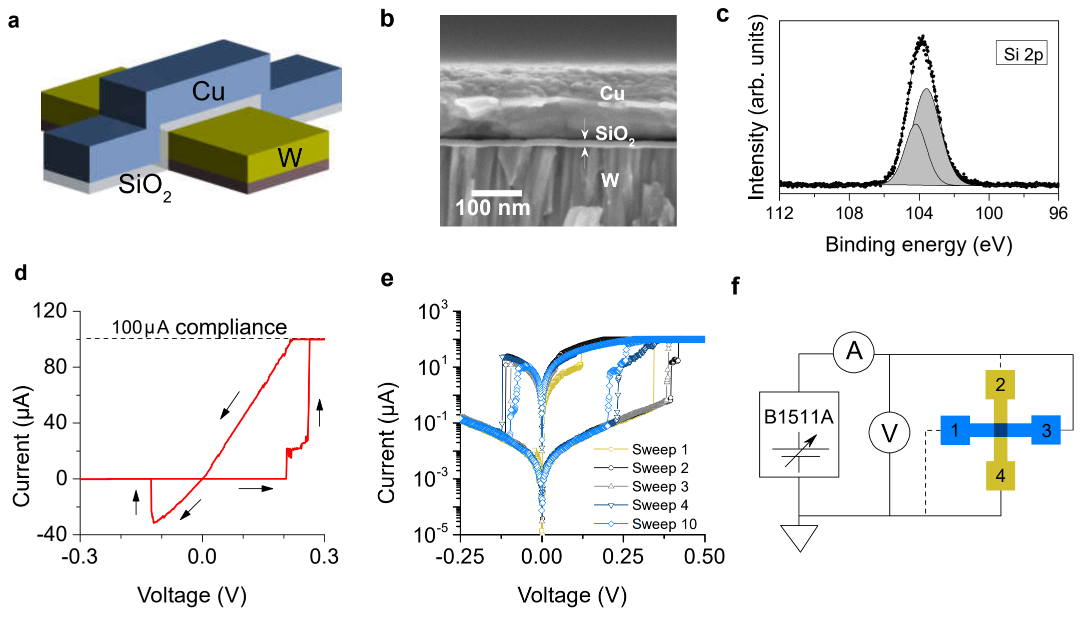


Figure 7.1 **a** Crossbar structure of the fabricated device. **b** FEGSEM (field emission gun scanning electron microscope) image of the device (cross section) showing 10 nm SiO_2 sandwiched between Cu and W. **c** Si 2p XPS spectrum of a 10 nm sputter-deposited SiO_2 film. Our quantitative analysis suggests O/Si ratio of 2. **d** Typical low voltage pinched hysteresis observed in the device in linear scale. **e** The first four and the tenth sweep across the device are shown in log scale. **f** Electrical setup schematic used to characterize the device at the cross-point junction is shown. The connection via device contact pads 1 and 2 or 3 and 4 are used for the device characterization, and those through 1 and 3 or 2 and 4 are used for the electrode characterization.
Source: [45].

that a current sweep measurement was better suited to observe the conductance quantization and controlled filament growth in RRAM devices. Voltage sweep measurement shows a fast conductance transition assisted by a positive feedback mechanism. A growing filament further amplifies the electric field in the filament gap accelerating the filament growth rate until the current compliance is hit. This places limited control over the filament growth process. On the other hand, in a current sweep mode, there is enough voltage across the device while it is in the HRS mode to drive the filament growth, however, the moment a filament is formed to establish

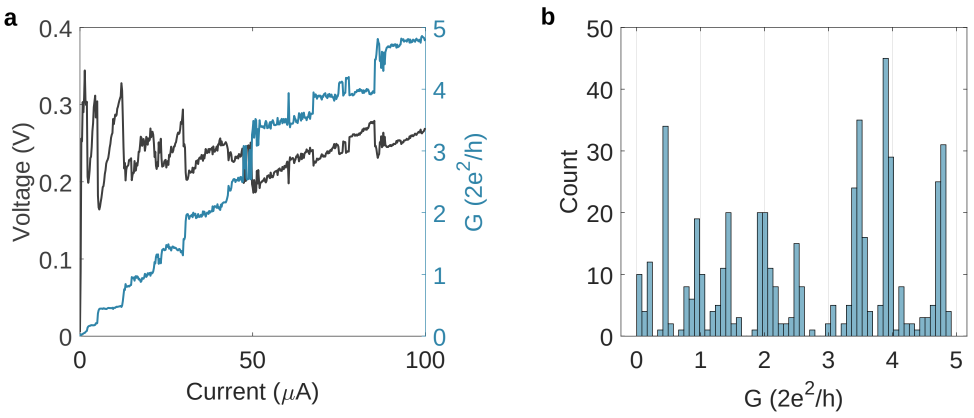


Figure 7.2 **a** Voltage across the device and corresponding conductance change of a Cu/SiO$_2$/W device during current sweep measurement. **b** The corresponding conductance histogram of quantization levels. We can see the first voltage drop and the histogram indicate the presence of a quantized conductance level below $0.5\,G_0$. *Source:* [45].

the lowest conduction mode the voltage across the device drops immediately due to a sudden increase in the conductance. This slows down any further filament growth. The subsequent increase in current causes the voltage to rise again, however, from a lower voltage and until the filament growth adds a new conduction mode causing a sudden drop in voltage. This negative feedback leads to controlled filament growth and permits observation of quantized states in the device.

We used a $200\,\text{nA}/10\,\text{ms}$ current sweep from 0 to $100\,\mu\text{A}$ and monitored the voltage across the devices. We observed conductance quantization levels at integer and half-integer multiples of $2e^2/h$ (typically called conductance quantum, G$_0$) (Figure 7.2) [45] repeatedly although every conductance level was not present. The quantization of conductance observed here proves the filamentary nature of the conductance path formed between the electrodes in the Cu/SiO$_2$ based CBRAM [131, 132, 133, 134, 135, 136]. Though there could be multiple filaments we assume a single filament for the simplicity of the following discussion.

In the ballistic mode of electron transport in the metallic filament, the number of energy sub-bands that electron can occupy becomes limited and discrete [130, 137, 138]. In theory, each sub-band contributes a conductance of $2e^2/h$ to the channel [139, 140]. Generally, the conductance quantization is reported using low amplitude AC signals and the number of sub-bands available for conduction from the both the contacts will appear to be same. However, in our case the voltage across the device causes a Fermi level split. In Figure 7.3b, E_{FL} and E_{FR} respectively represent the Fermi levels of the left and right contacts and energy level split $\Delta E = eV$, where e is the electron charge and V is the applied voltage. Mathematically, the density of state n_{1D} for one sub-band obey the relation, $n_{1D}v_g = (2/h)\theta(E)$ where v_g is the electron velocity and $\theta(E)$ is the heaviside step function. When there are multiple sub-bands each at energy level E_i, the current through the channel at zero temperature limit can be written as,

$$\begin{aligned} I &= I_L - I_R \\ &= e\int_{-\infty}^{\infty} dE n_{1D} v_g [f(E-E_{FL}) - f(E-E_{FR})] \\ &= \frac{2e}{h}\int_{E_{FR}}^{E_{FL}} \sum_i \theta(E-E_i)dE \\ &= \frac{2e}{h}\left(\sum_{E_i<E_{FR}} \int_{E_{FR}}^{E_{FL}} \theta(E-E_i)dE + \sum_{E_{FR}<E_i<E_{FL}} \int_{E_i}^{E_{FL}} \theta(E-E_i)dE\right) \end{aligned} \tag{7.1}$$

where I_L and I_R are the current injected into the channel by the left and right contacts. If we assume that the contacts are reflection free and the channel is scatter-less and one of the energy levels is halfway between the top and bottom electrodes, Equation 7.1 can be reduced to

$$\begin{aligned} I &= \frac{2e}{h} N\, eV + \frac{2e}{h}\frac{eV}{2} \\ &= G_0\Big(N + \frac{1}{2}\Big)V \end{aligned}$$

where N is the number of sub-bands below both the contact Fermi levels. Hence, the observation of half-integer conductance can be attributed to the Fermi level split

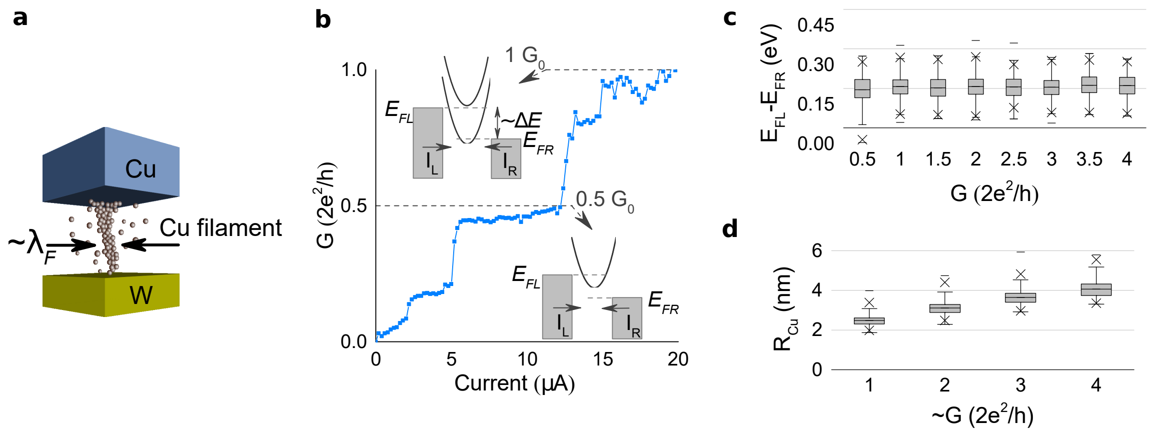


Figure 7.3 **a** Schematic of Cu filament formed inside the SiO_2 dielectric. **b** $E-k$ band representation for $0.5\,G_0$ and $1\,G_0$ states marked on the corresponding $G\,vs.\,I$ plot. **c** Dual current sweep measurement across an already formed filament which shows a conductance transition from $3.5\,G_0$ to $4.5\,G_0$. **d** Plot of the Fermi level splitting $(E_{FL}-E_{FR}\sim e\,V)$ across the filament in each quantized conductance state $((n/2)\,G_0 \pm 0.05\,G_0)$ during filament formation in a current sweep. **e** An estimate of Cu filament radius (R_{Cu}) using Equation 7.2.
Source: [45].

causing an unequal number of conduction modes available for transport from the two contacts. Such characteristics have been observed previously where a finite voltage across the filament leads to an observation of quantization at half integer multiples of G_0 [141, 142, 143, 144, 145, 146].

The number of available conduction modes in the nano-filament is a function of the lateral filament radius. Hence increasing conductance in the current sweep measurement is an indication of a growing filament. We observe that the voltage across the filament appears to be bounded in Figure 7.2. If we neglect the voltage-drop across any contact resistance, this voltage corresponds to the Fermi-level split. The maximum of this separation could correspond to the sub-band spacing of the discrete energy levels in the 1D channel (sub-band diagram for $1G_0$ in Figure 7.3b). The distribution of the voltage measured across the device for the range of conductances $G = (n/2)\,G_0 \pm 0.05\,G_0$ is shown in Figure 7.3c, which have an approximately constant mean value. While theoretically, we expect an increasing sub-band spacing for a $1D$

channel and hence an upward trend in the voltage across it, the relatively constant distribution of voltage suggests that the increase in sub-band spacing is compensated by an increasing filament radius. This approximate sub-band spacing can be used to estimate the filament radius from the solution of the Schrodinger's equation for cylindrical filament structure using the equation,

$$R_{Cu} = \left[\frac{\hbar^2}{2m^*\Delta E}\right]^{1/2}\sqrt{(x^{\ell}_{n+1})^2 - (x^{\ell}_{n})^2} \tag{7.2}$$

where x^{ℓ}_{n} is the n^{th} zero of Bessel function to the ℓ^{th} order and $m^*(= 1.01\, m_e)$ is the electron effective mass [147] in Cu. The estimated filament radius is shown in Figure 7.3d which lie approximately in $2.5 - 4$ nm range for the first few quantized levels in Cu.

7.4 Cu/SiO$_2$/W RRAM as a Synapse

Mimicking human brain's architecture and computational primitives to build intelligent information processing systems is the key goal of neuromorphic engineering research activities worldwide. While there have been several demonstrations of neuromorphic computational platforms using standard CMOS technology [148, 149, 150, 151, 152, 153, 154, 155, 156, 157], and demonstrations of nanoscale devices to mimic neuronal and synaptic dynamics [91, 158, 50, 159, 160, 28, 161, 162, 163, 45], none of these have achieved the target energy efficiency specifications necessary to build systems that can learn in real-time and in-the-field [164].

While the low operating voltages of the CBRAM is advantageous, a more gradual conductance change is desirable for synaptic implementation. Doping the dielectric with the active electrode material (e.g. Cu) could reduce the resistivity contrast between the dielectric and filament material and could improve the probability of a non-filamentary conductance transition. Doping the dielectric by annealing has been demonstrated to achieve gradual conductance change in

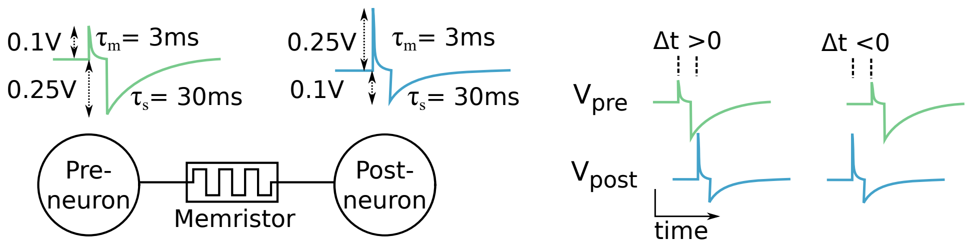


Figure 7.4 Bio-inspired programming waveforms used in the experiment to demonstrate STDP in a memristive device.

Cu/SiO$_x$/W based device [26]. Further, the ionic migration velocity in the dielectric is exponentially dependent on the electric field and hence the actual conductance evolution characteristic is also dependent on the time-scale and shape of the programming waveform [165, 166, 167]. Also, the feasibility of using CBRAM devices for STDP has been shown by model simulations [168].

In this work, we experimentally demonstrate STDP like memory switching behavior in a CMOS compatible memristive device using low-voltage bio-mimetic programming waveforms. We specifically engineered the ionic doping in our device by thermal annealing to achieve analog or incremental conductance changes ideal for synaptic memory, as opposed to a high on-off ratio between the on and off states required for binary data storage. For gradual conductance change, the programming waveform is designed such that it minimizes the time spent by the device under high voltages which reduces the probability of abrupt switching while providing sufficient electric field to initiate ionic movement [165, 167]. The programming waveforms shown in Figure 7.4 are designed to meet these requirements. Also, we develop a compact model capturing its key synaptic behavior and analyze its learning capability as a synaptic device in exemplary spiking neural networks.

7.4.1 Spike-timing dependent plasticity demonstration

In this section, we describe how the STDP characterization of the device was performed. We used Agilent B1500 semiconductor parameter analyzer with a B1530 unit. B1530 is an arbitrary waveform generator and fast measurement unit (WGFMU) capable of applying 100 ns pulses and accurate current measurement with up to 5 ns sampling rate. The set-up is connected to a probe station where the device is probed and characterized. The spike programming waveforms were designed using Matlab and were converted to voltage signals using the WGFMU. A slow dual voltage sweep with a ramp rate of approximately 2 V/s was used to characterize the device for its discrete switching behavior. For the STDP characterization of the device, 1 s long patterns where used at a time, and in that duration each 250 ms was used for a waveform corresponding to one spike pair. The waveforms were created from data-points at a resolution of 0.5 ms. Read pulses of 5 ms duration and 50 mV amplitude were inserted at the ends and in between the programming waveforms to determine the state of the device. Programming waveforms corresponding to randomly chosen 2000 Δts ($\in [-80\,\text{ms}, +80\,\text{ms}]$) were applied to the Cu terminal of the device. The resulting current as a function of time is read using the measurement unit. The average current during the read pulse is divided with the read voltage to get the device conductance changes due to each STDP event. However, after approximately 1000 programming events the device became relatively unresponsive and was stuck to a narrow range of conductance. Also, the conductance response outside the window of [−40ms, +40ms] was not well correlated and are discarded from the study.

The action potentials corresponding to the spikes from the pre- and post-synaptic neurons, V_{pre} and V_{post} are approximated (Figure 7.4) using two decaying exponentials as below:

$$
\begin{aligned}
V_{pre}(t) &= A_1 e^{-t/\tau_m} u(t) - A_2 e^{-(t-3\tau_m)/\tau_s} u(t-3\tau_m) \\
V_{post}(t) &= A_2 e^{-t/\tau_m} u(t) - A_1 e^{-(t-3\tau_m)/\tau_s} u(t-3\tau_m)
\end{aligned}
\tag{7.3}
$$

where $A_1 = 0.1\,\mathrm{V}$, $A_2 = 0.25\,\mathrm{V}$, $\tau_m = 3\,\mathrm{ms}$, $\tau_s = 30\,\mathrm{ms}$ and $u(t)$ is the Heaviside step function. The waveforms in Equation (7.3) are designed to capture the depolarization-repolarization-hyperpolarization cycles in the biological action potential waveforms.

The amplitudes of V_{pre} and V_{post} are chosen such that they are below the minimum set and reset voltages of the device when there is little overlap between the waveforms. However, when the spikes are closer, the magnitude of the instantaneous voltage across the device increases, resulting in non-volatile conductivity modulation. Even though biological action potentials last only a few milliseconds, experimental evidence suggests synapses are able to capture the correlations between spike events that are even $50 - 100$ milliseconds apart. In our experiment, the time constants of the programming waveforms are chosen to have non-zero overlap between the pre- and post-synaptic neuron spike waves for the desired duration for this STDP window. While the signal propagation direction of action potentials is along the axons away from the cell body or soma, some experiments also suggest a back-propagation of action potentials along the dendrites which could play a key role in the synaptic plasticity [169, 170]. Inspired by this, our programming scheme assumes that the V_{post} signal is sent in the backward direction from the post-synaptic neuron when it spikes. If spikes from both the pre- and post-synaptic neurons occur close in time, there will be significant overlap in the voltage waveforms generated across the device. Depending on the direction and magnitude of the field across the device, the conductance of the device could increase (potentiation) or decrease (depression).

For the experimental demonstration, we assume that the Cu electrode is connected to the post-synaptic neuron and W to the pre-synaptic neuron. However,

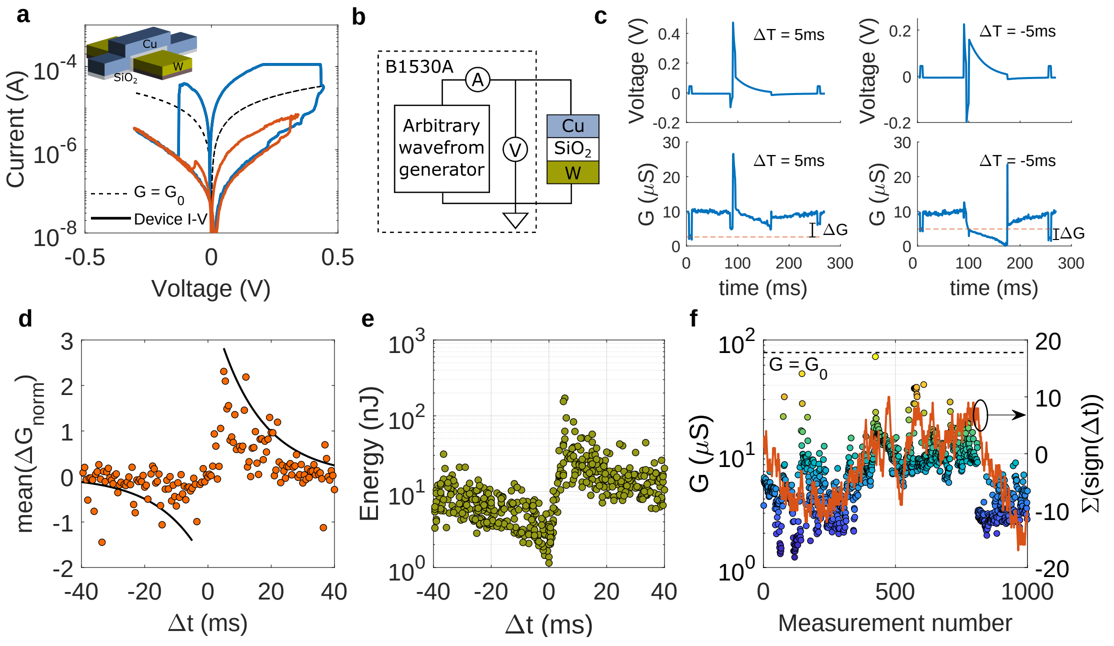


Figure 7.5 **a** Memory switching behavior of the Cu/SiO$_2$/W cross-point device for two voltage sweeps with maximum amplitude of 350 mV and 450 mV. The current corresponding to a conductance of $G_0 (= 2e^2/h)$ is marked using a dashed line. The conductance response above the G_0 level (blue curve) indicates at least one filamentary path connecting the top and bottom electrode, while the orange curve shows partial conductance switching in the sub-quantization regime. The 3D device structure is shown in the inset. **b** STPD programming and measurement set-up **c** Example STDP programming waveforms applied (top) and the measured device conductance evolution (bottom) where $\Delta t = 5$ ms results in a synaptic potentiation and $\Delta t = -5$ ms results in a synaptic depression. **d** STDP response of the device determined as the average conductance change (normalized) versus the spike time difference based on 400 measurements. $\Delta G_{norm} = (G_f - G_i)/min(G_i, G_f)$ **e** Energy spent per programming event in the device as a function of Δt **f** Evolution of the device conductance during the STDP measurement tracks the overall causal/anti-causal signal correlation. The orange curve is a cumulative sum of the sign of the Δt across the sequence of measurements.

instead of applying the V_{pre} and V_{post} to the respective terminals we compute the difference waveform $V_{post} - V_{pre}$ as a function of time and is applied to the Cu terminal with the W electrode at ground (Figure 7.5b). Such waveforms were created using the WGFMU for different spike time differences. Example waveforms applied to the device for a positive and negative time difference of 5 ms and the corresponding device conductance evolutions are plotted in Figure 7.5c. Each programming waveform is appended with an initial and final non-disruptive read pulse of 50 mV amplitude to measure the device conductance. As indicated in the figure, the $\Delta t = 5$ ms signal leads to potentiation and $\Delta t = -5$ ms leads to depression in device conductance. The average conductance change for the spike time differences in an interval of [-40 ms, $+40$ ms] is plotted in Figure 7.5d based on 400 randomly chosen Δts. The average ΔG_{norm} $(= (G_f - G_i)/min(G_i, G_f))$ versus Δt, where G_i is the initial and G_f is the final conductance for a spike time difference of Δt, is similar to the STDP response from a biological synapse shown in Figure 2.1b. The distribution of energy consumed during these programming events (Figure 7.5e) indicates that potentiation cycles consume higher energy compared to depression. On average, the device consumes 10 nJ per programming event. In Figure 7.5f we show the conductance evolution of the device during the STDP measurement. We observed that the device always stayed below the quantized conductance level of $G_0 (= 2e^2/h)$ with its minimum conductance at $0.016\, G_0$. The filamentary paths often formed in the conductance-bridge resistive memory devices act as nanoscale electron channels and result in conductance levels which are integer multiples of G_0. The sub-quantized levels in our device are indicative of non-filamentary atomic rearrangement based conductance modulation. Further, the superimposed orange curve, which is a cumulative representation of the sign of the applied Δts during each spike-time-difference based programming event follows the same trend as the device conductance evolution. Thus, the device plasticity has successfully captured the overall causal-anti-causal spike pair relations.

7.4.2 State dependent phenomenological model

To study how such device characteristics will be effective in emulating synapses in neural network implementations, we developed a phenomenological model capturing the observed plasticity behavior. Previously, we developed a simple analytical model for explaining the device operation based on growth/decay dynamics of the filament, using differential equations incorporating Mott-Gurney ionic transport [171]. However, to study the feasibility of the device for implementing learning algorithms via numerical simulations, a computationally simpler model that predicts the next state of the device, G_f, given the current state, G_i, and spike pair time difference, Δt, is more suitable.

A careful analysis of the device response reveals the state dependency of the conductance change as a function of the timing difference of the applied waveforms. We study the device response for three regimes, demarcated in units of quantum conductance G_0: (a) when $G_i < 0.05\,G_0$, (b) when $0.05\,G_0 < G_i < 0.16\,G_0$, (c) when $G_i > 0.16\,G_0$. From the average ΔG_{norm} vs Δt plotted for different ranges of initial conductance, we observe that when the initial conductance is in the intermediate range, conductance change in the direction of potentiation and depression is in the same range for the same spike time difference, while potentiation is pronounced and depression is weak in the low initial conductance regime (Figure 7.6a,b)). Albeit noisy, a similar trend was also visible in high initial conductance regime, where device shows more tendency for conductance depression than potentiation. This behavior indicates a reduction in the incremental conductance change and a conductance saturation as the device reaches closer to its upper and lower conductance limits.

Our model essentially captures these state dependent conductivity modulation characteristics by modelling the normalized change in conductivity ΔG_{norm} as:
When $\Delta t > 0$,

$$\Delta G_{norm} = A \exp\left(\frac{-\Delta t}{\alpha_{ap} + g\beta_{ap}}\right) - A \exp\left(\frac{-\Delta t}{\alpha_{bp} + g\beta_{bp}}\right) \tag{7.4}$$

and when $\Delta t \leq 0$,

$$\Delta G_{norm} = -A \exp\left(\frac{\Delta t}{\alpha_{an} + g\beta_{an}}\right) + A \exp\left(\frac{\Delta t}{\alpha_{bn} + g\beta_{bn}}\right) \tag{7.5}$$

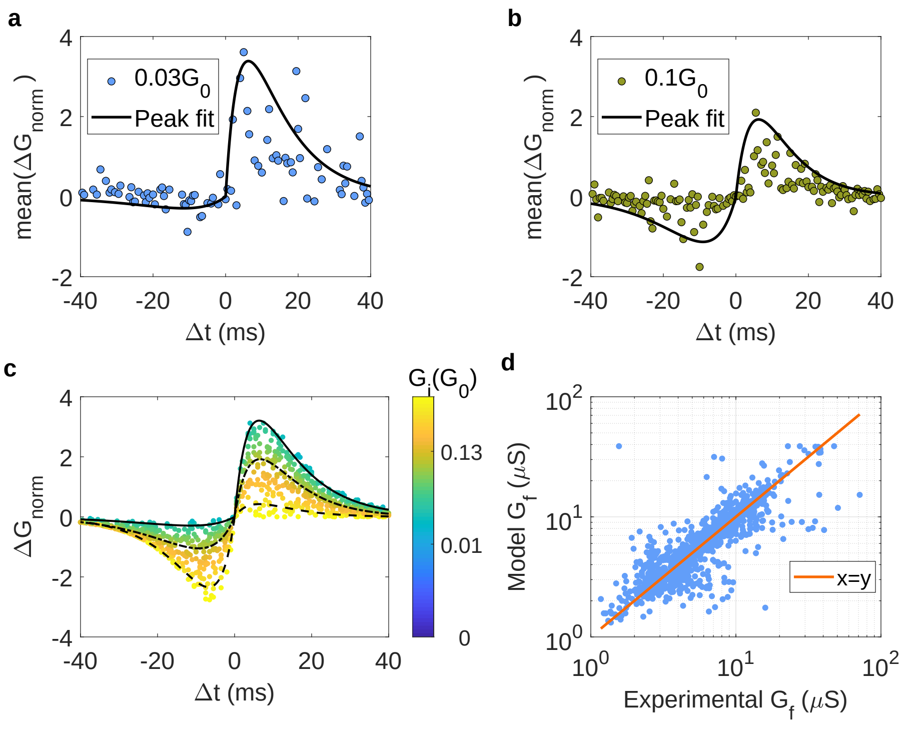


Figure 7.6 **a** Average ΔG_{norm} defined as $(G_f - G_i)/min(G_i, G_f)$ response from the device measurement when the initial conductance is below $0.05\,G_0$ with an average of $0.03\,G_0$ and when **b** it is in the range between $0.05\,G_0$ and $0.16\,G_0$ with an average of $0.1\,G_0$. **c** The response of the phenomenological model, when programmed with a sequence of randomly chosen Δts. **d** The device conductance after the application of the pulse (G_f), calculated from the phenomenological model is well correlated with the experimental values ($R^2 \sim 0.56$), for the same initial conductance values and programming Δts as in the experiment over a dynamic range of two orders of magnitude.

$A = 9$, $g = \log_{10}(G_i/G_0)$, and other parameters are listed in Table 7.1. Further, the model is limited to operate within a conductance range of $G_{min} = 0.016\,G_0$ and $G_{max} = 0.5\,G_0$. The Equations 7.4, 7.5 are formed as the difference of two decaying exponentials with different time constants. The time constants are functions of the device conductance and converges to the same value at the boundary points such that the conductance change gradually becomes zero. Such state-dependent behavior is akin to the saturating conductance responses observed in many of the gradual conductance and STDP demonstrations in the memristive devices [28, 172, 173, 26, 25, 174, 175]. Biological STDP is also known to exhibit such a state-dependent behavior. For example, experimental measurements in rat hippocampal neurons indicated that significant long-term potentiation occurred only in synapses of low initial strength, although such an obvious dependence was not observed in long-term depression [15]. This state-dependent conductance modulation could lead to an asymmetry between potentiation and depression, especially when the current state is closer to either of the conductance boundaries.

Table 7.1 Cu/SiO$_2$/W Phenomenological Model Parameters

Symbol	Value	Symbol	Value
α_{ap}	5.2 ms	β_{ap}	−3.8 ms
α_{bp}	6.9 ms	β_{bp}	1.9 ms
α_{an}	9.1 ms	β_{an}	−1.9 ms
α_{bn}	2.3 ms	β_{bn}	−5.7 ms

The STDP response from the phenomenological model is shown in Figure 7.6c, where the device conductance is initialized to 1 μS and is programmed with a sequence of random Δts. To compare the model and device response, the model is initialized with the exact device conductance measured before each Δt from the experiment and the correlation between the model and the device conductance responses after each Δt based STDP programming is plotted in Figure 7.6d. The R^2 estimation between these experimental and model conductance values measured over two orders of magnitude is 0.56.

Data fitting Here, we describe how the parameters in the phenomenological model (equation (7.4, 7.5) and Table 7.1) are obtained from the device STDP response. We first obtained peak-fits of the experimentally measured average ΔG_{norm} data points for medium and low initial conductance range separately for the positive and negative Δt range (Figure 7.7). The phenomenological model for the STDP behavior of the device was obtained by fitting a function, $f = A(exp(-\Delta t/\tau_A) - exp(-\Delta t/\tau_B))$, to the peak-fit points of the ΔG_{norm} vs Δt in a state-dependent manner. Figure 7.7a shows the peak-points in a moving Δt window of 5 ms within the range of $[0, 40\text{ms}]$ for ΔG_{norm} data points which are in a low initial conductance range ($G_i \in [0.016\, G_0, 0.05\, G_0]$ with a mean of $0.03\, G_0$ where $G_0 = 2e^2/h$). The corresponding function f fit line obtained using gradient descent is also shown over the peak-fit points. Figure 7.7b shows the peak-fit points and the corresponding model fit line for the negative Δt range. Figure 7.7c,d show the similar model fitting results for the medium initial conductance range ($G_i \in [0.05\, G_0, 0.16\, G_0]$) which has a mean of $0.1\, G_0$. The exponential fit lines give the time constants τ_A and τ_B as a function of the initial conductances for positive and negative Δts.

The phenomenological model determine the next state for each spike-pair based programming using the relation $\Delta G_{norm} = A(exp(-\Delta t/\tau_A) - exp(-\Delta t/\tau_B))$. We observed that the τ_A and τ_B are functions of the conductance of the device before programming and the sign of the time difference between the programming spike pairs (Δt). We approximated the time constant versus initial conductance relation using linear functions of $log(G_i)$ as $\tau = \alpha + \beta \times log(G_i/G_0)$. Fig 7.8a shows the linear fits for $\Delta t > 0$ and Fig 7.8b shows the linear fits for $\Delta t < 0$. At the points where the two lines meet, we have $\tau_A = \tau_B$ and $\Delta G_{norm} = 0$. We limit the range of the phenomenological model within the boundary points at which G_{norm} becomes zero. Slopes and intercepts of the linear fits are used to determine the parameters αs and βs.

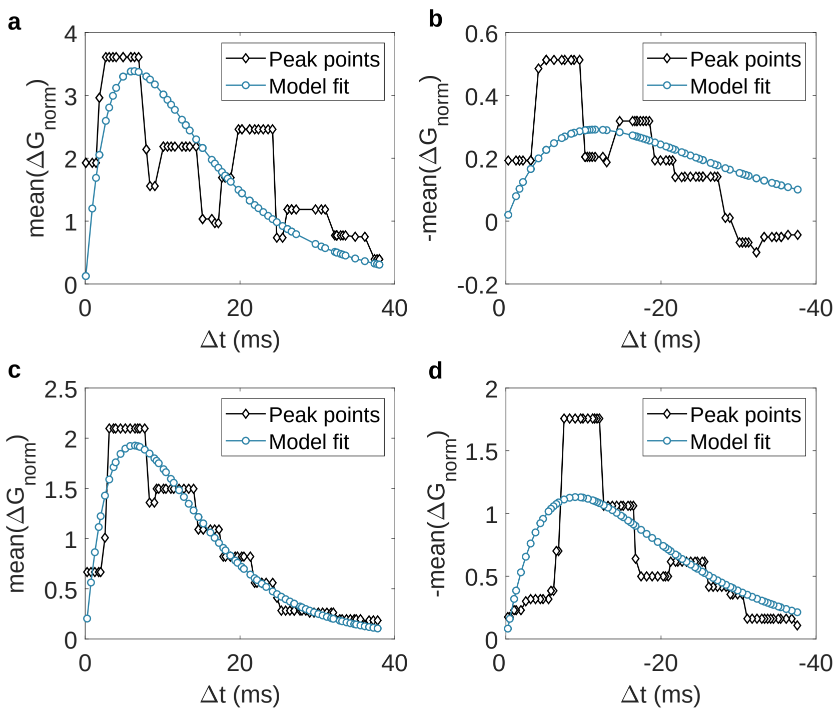


Figure 7.7 Peak points and the corresponding double-exponential function fit for ΔG_{norm} versus Δt data with low (**a**, **b**) and medium (**c**, **d**) initial conductance are shown separately for positive and negative Δts.

7.5 Supervised Learning Emulation

Next, we discuss how this device could be used in an exemplary spiking neural network for implementing event-triggered learning. An STDP derived supervised learning algorithm, similar to the ReSuMe [176], is used to train a network with 1000 inputs neurons and one output neuron (Figure 7.9a). The task is to determine the weights of the 1000 synapses of the output neuron such that it creates spikes at the desired instants as dictated by a teacher neuron when they are excited by spike streams generated by a Poisson process (Figure 7.9c,d). The phenomenological model for the device plasticity was employed to emulate the synaptic behavior during the training of the spiking neural network. At the beginning of training, synapses are initialized as a distribution around the geometric mean of the maximum and minimum conductance of the model ($G_{ref} = \sqrt{G_{max}G_{min}}$). This G_{ref} is considered as a reference level

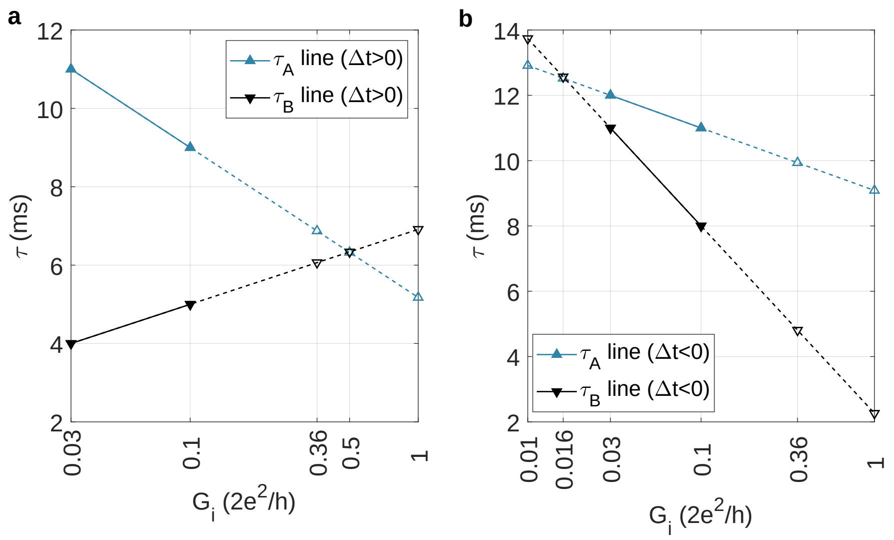


Figure 7.8 The τ_A and τ_B as a function of G_i is fitted with linear functions for positive Δt in **a** and negative Δt in **b**. The solid markers denote the τ values obtained by the data fitting and open markers are obtained by extending the linear fit. The meeting points of the lines determine the boundary conductance for the STDP operation of the phenomenological model.

around which the synaptic weights are allowed to vary during the training such that individual synapses could be either excitatory or inhibitory. Implementation of such a reference level in hardware may require an additional memory device along with each synaptic device (Supplementary Note 3). The training rule for the synapses is shown in Figure 7.9a,b. When the teacher neuron spikes, the synapses are potentiated based on the time elapsed since the most recent input spike. Similarly, the synapse will be depressed when there is an observed spike from the output neuron, based on the time difference with the last input spike. When the output neuron spike coincides with a teacher neuron spike (i.e., the desired response is obtained from the network), there will not be any synaptic modulation. The amount of potentiation or depression for each time difference is determined in a state-dependent manner using the device STDP model. This process is continued for repeated presentations of the input and desired spike patterns to the network until the observed spike stream is similar to the desired pattern. In our simulations shown in Figure 7.9c,d, the network generates all the spikes at the desired times within ±10 ms in 9 epochs. The weight evolution generated by our model for a few synapses during the training is shown in Figure 7.9e.

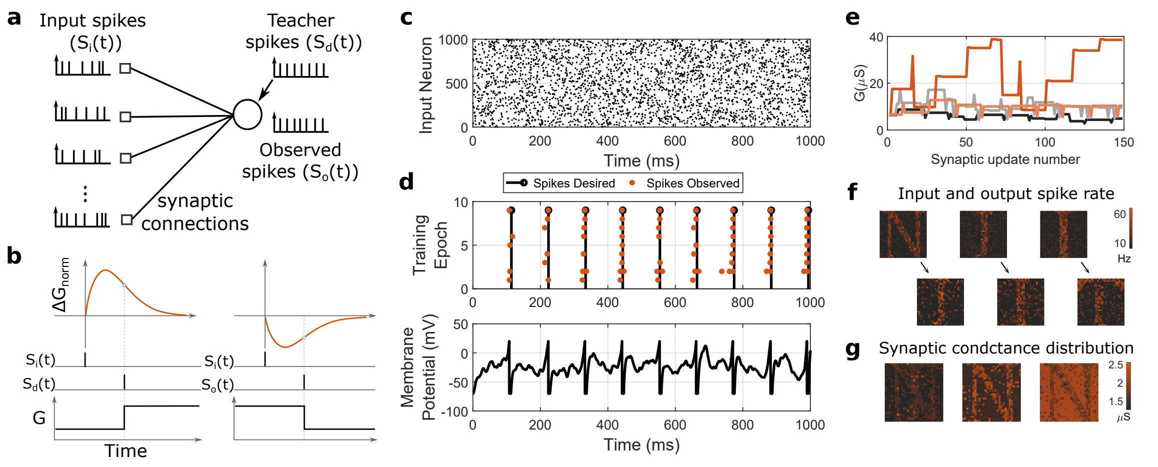


Figure 7.9 **a** An exemplary spiking neural network with N input neurons connected to a single output neuron. The training task is to discover the synaptic weights such that the output spike response, $S_o(t)$, matches the desired response, $S_d(t)$, for a specified input spike excitation, $S_i(t)$. **b** Training rule: the synapse is potentiated or depressed based on the time difference of desired or observed spikes respectively from the efferent spike using the STDP model. **c** A raster plot of the spike streams from each input neuron in a 1000×1 SNN is shown. **d** The desired and observed spike trains over the training epochs (top) and the final membrane potential (bottom) from the output neuron as a function of time. **e** Device conductance evolution for four exemplary synapses during the training of the 1000×1 neural network. **f** Input (top) and observed output (bottom) neuron spike rate in an SNN trained for sequence prediction. **g** Relative conductance distribution of synaptic device models connected to three output neurons showing the features learned after training. The first set of synapses is connected to a neuron responding to letters N and J. The second set of synapses is connected to an output neuron responding only to letter N. The third output neuron had the desired spike having anti-causal relation with most of the input spikes leading to an effective synaptic depression.

Such algorithms are at the heart of supervised learning platforms in spike domain. Once input and output data are translated to spike domain, it can be used to efficiently train networks for different tasks, provided the synaptic realization has sufficient analog programmability. The $N \times 1$ network could be extended for more complex problems. For example, we realize a 900×900 spiking neural network whose synapses are represented by our phenomenological model (Supplementary Note 4).

The network acts as a sequence predictor for English letters N$\rightarrow$J$\rightarrow$I$\rightarrow$T such that when the input layer is presented with one of the letters, the network creates the image of the next letter in the sequence at the output layer. The pixel intensities of 30×30 grayscale images are used as rate constants for a Poisson process to generate the corresponding input and desired output spike streams. The network synapses are trained using the same supervised STDP rule as before. The input and the resulting output spike rates from the network after training can be mapped to the input and predicted output image as shown in Figure 7.9f. The conductance distributions of the synapses connected to three output neurons (akin to a receptive field) are shown in Figure 7.9g. The relative distribution of the conductance illustrates the features the network have learned, enabling it to efficiently map the input spike streams to the corresponding output spike streams, in an event-triggered manner. Note that the first set of synapses are connected to an output neuron responding to both letters N and J, illustrating the complex information representation capacity of synapses obeying the plasticity rules. The phenomenological model that we used in these SNN simulations could be easily tuned for larger conductance range or asymmetric potentiation-depression behavior to capture similar STDP trends observed in other devices. This will enable potential evaluation of synaptic device specifications in new neural network architectures and their learning capacities.

7.6 Discussion

There have been numerous demonstrations of CBRAM devices, including some based on the Cu/SiO_2/W material system, as non-volatile memory devices programmed to two distinct resistance states [177, 132, 178, 47, 179, 180, 181]. These binary devices are characterized by high on-off ratios ($\sim 10^3$) due to the large resistivity difference between the dielectric and the bridging metallic filament. Though the stochastic switching nature of the CBRAM devices has been utilized to perform neuromorphic computations [182, 183, 184], it is desirable to have incremental conductance change in the synaptic device to implement event-triggered learning in neuromorphic circuits. While there have been attempts to obtain multi-level or gradual conductance change operation by modulating the filament thickness in CBRAM, [185, 186] the high power consumption due to the metallic filaments bridging the electrodes makes them unsuitable for neuromorphic applications [180]. In our device, the dielectric is effectively doped with Cu atoms using annealing and the device demonstrates more gradual conductance transitions in a regime below quantized ballistic transport.

We have implemented spike-timing dependent plasticity behavior in our device in its sub-quantized high resistance states using bio-mimetic programming waveforms with peak programming voltages below 500 mV. Additionally, the bipolar nature of the device is suitable for simple pulse overlap programming and is hence ideal for implementing local learning rules in a compact manner. In contrast, a PCM which require two different temperature distributions for the crystal growth and amorphize necessitates more complicated programming waveforms and circuits to implement STDP behavior [28]. STDP behavior has also been demonstrated before in silicon based [187] and oxide based resistive memory devices [24, 50, 172, 188]. However, these devices are characterized by either high operating voltages (typically $> 1\,\mathrm{V}$) or high operating current (0.1 mA to 10 mA) or low on-off ratio (< 10). The plasticity demonstration in the low voltage $Cu/SiO_2/W$ device hence opens up the potential for realizing denser and more energy-efficient synaptic networks, with further device optimization.

A critical advantage of the $Cu/SiO_2/W$ device is that it is CMOS compatible, and can be integrated during the back-end metallization steps, making it possible to even achieve high-density 3D arrays. Furthermore, unlike conventional memory arrays, memristor-based synapse arrays are read in parallel (to perform the weighted summation operation in neural networks) and also programmed in parallel using overlapping programming pulses from the periphery (to implement local spike-triggered learning). Hence, the sneak-path issue that plagues conventional cross-point memory operation is not so severe for neuromorphic applications, although it is essential to perform the device programming without altering the neighboring devices. As a result, it may become necessary to integrate selector devices at the cross-point along with the memristor devices for the proper operation of large synaptic arrays. While a three-terminal CMOS transistor could be used as a selector, there are also efforts to develop two terminal bipolar selector devices with high on-off ratio and tunable threshold voltages so as to maintain high integration density for synaptic applications [189, 190]. The programming waveforms can be modified in a straightforward manner based on the combined characteristic of the CBRAM and selector device [191].

For STDP realization in memristive devices, several programming methods have been proposed. One of the initial approaches has been to convert the sign and magnitude of the spike-time difference into the polarity and width of a rectangular programming pulse [187]. However, this necessitates a global circuit to generate the programming pulse with tunable width. Different forms of pulse-overlap based programming schemes have also been proposed such as amplitude modulated rectangular pulse sequences [168] and continuous analog waveforms [192, 50] drawing different levels of biological inspiration. Also, programming waveforms could be reshaped to realize different forms of STDP behavior. In this work, we use an analog programming waveform which has been tuned such that sufficient electric field is set up across the device to initiate ionic drift while keeping the duration and amplitude of the voltage sufficiently low to avoid rapid switching transitions to the extreme conductance levels.

For the plasticity demonstration in our device, we have reduced the dielectric resistance by ionic doping, and operated the device in the sub-quantized regime, effectively reducing the dynamic range available in the device. The dynamic range of devices which have demonstrated gradual conductance range is also often similarly limited and calls for joint co-optimization of the algorithmic learning parameters and the conductance modulation characteristics of the device. The measurement of excitatory post-synaptic currents in rat hippocampal neurons suggests that the effective dynamic range of biological synapses could be as large as two orders of magnitude [15]. A recent study on the synaptic spine heads found that there could be two or more synaptic connections between neurons and estimated that each synaptic junction could store 4.6 bits of information [70]. Neural network training problems requiring higher resolution than that is available from a single device may also benefit from materials innovations or architectural and algorithmic improvements [51, 95, 69].

Local learning rules such as STDP is key to the decentralized and parallel processing capabilities of the biological neural networks. Realization of biological plasticity mechanisms in nanoscale memristive devices when combined with their high integration density and scaling potential enables power efficient implementations of large-scale learning networks. While this proof-of-concept demonstration establishes the basic feasibility of our device for learning, there are challenges in terms of device reliability, and variability that warrants further research and optimization. For example, the Cu/SiO_2/W based devices we fabricated was responsive to approximately 1000 STDP measurements. The higher mobility of Cu in SiO_2, though useful for low voltage operation, might be the reason behind the rapid deterioration in endurance and retention performance [45]. Further, the stochastic nature of atomic rearrangement upon programming results in stochasticity in the conductance modulations as well. Also, the device-to-device and cycle-to-cycle variation of the resistive memory devices could affect the array-level performance, though it is expected that some of these reliability issues could be mitigated by improved industrial fabrication processes. Furthermore, the targeted neuromorphic applications are more error tolerant as decisions are based on overall synaptic distributions rather than the absolute conductance levels of any particular device, making resistive memory based synapses more attractive for cognitive hardware.

7.7 Summary

We studied the conductance modulation behavior of Cu/SiO_2/W RRAM. These devices can be programmed using a few hundreds of millivolts and typically demonstrate a fast binary switching based on the presence or absence of a nano-filament. The observation of conductance quantization at room temperature confirms the presence of filamentary paths. We demonstrated that by suitable programming waveforms and annealing to dope the dielectric with metallic ions, STDP-like gradual conductance based learning rules can be implemented using such devices. Based on the state-dependent nature of conductance change in such devices, we developed a phenomenological model and used it to analyze the feasibility of using such devices for the training of spiking neural networks. Despite the limited on-off ratio, granularity, and non-linearity, the device was able to reach performance levels comparable with high-precision training in certain spike pattern generation task in an SNN. However, the array level and individual device variability, endurance, and retention of these devices need to be further evaluated.

CHAPTER 8

FUTURE OUTLOOK

In the era of Big-data, the ability to process the data at the source in an energy efficient manner opens up enormous possibilities and applications. Nanoscale memory devices can realize compact high-density neural networks which combine storage and processing and is a suitable tool for such applications. However, programming the analog memory to achieve high accuracy in neural network benchmark tasks have been a challenge. Through the mixed-precision architecture and several experiments and simulations, we have shown that high-precision comparable classification accuracies can be achieved using nanoscale devices. Our methods have demonstrated the existence of solutions to the complex neural network problems in weight space that can be realized using nanoscale devices, and training methods to achieve it. This work hence opens up the possibility to create energy efficient and scalable learning networks on chip.

Mixed-precision architecture based neural network training demonstration using nanoscale PCM devices is at the intersection of different abstraction levels of designing futuristic computing engines. Based on its current limitations and applications, we can propose several directions for further research.

- Algorithmic development: It may very well be that low-precision realizations of neural networks can approach the accuracy of software baseline. However, training such networks in hardware using nanoscale devices require a high-precision gradient accumulator to efficiently search the low-precision solution space. The mixed-precision training we presented demonstrates that updating a large number of weights by a very small amount could be in some sense equivalent to updating a small number of devices by a larger amount. Further inspirations for algorithmic improvement could also possibly be drawn from the biological neural networks, which use spike triggered and asynchronous updates for training its limited precision synapses.

- Device engineering: The developments in the mixed-precision domain indicates the promise of emerging nanoscale memory devices in neuro-inspired computing architectures. However, the scalability of the devices without losing the granularity, improvement in stochasticity, and reduction of conductance drift need to be demonstrated. Also, it is essential to improve the reliability of analog memory devices.

- Architectural improvement: While our work demonstrates proof-of-concept feasibility of the system, there are many challenges in developing independent learning accelerators using the mixed-precision architecture. The backpropagation based training mandates interfacing the analog computation in the memory array with digital circuits. The area efficiency and speed of computation of the crossbar array are determined by the encoding efficiency of the resulting peripheral circuits. For instance, the area of a 1T1R PCM cell is approximately $25\,F^2$. However, it is a hard problem to compress the peripheral encoding/decoding or neuronal circuit per word line or bit line to fit in a comparable area. This requires some intelligent resource sharing, encoding schemes, and hardware designs. Alternatively, there have been a few nanoscale device based implementations for the neuronal functions in SNNs [91, 193]. The may be possible to integrate such devices into computational memory arrays to realize more scalable architecture.

- Generalization: The applicability of the mixed-precision architecture to a wider class of problems need to be further accessed. There are challenges in efficiently porting different network architectures and algorithms to computational memory. While the limited dynamic range, precision, and stochasticity, might seem disadvantage at first, they may be able to provide some inherent regularization, avoiding some expensive computations in the traditional architectures. Also, we need to evaluate the feasibility of remapping the same computational memory core to different network structures.

- Application domain: Beyond as a training accelerator, the nanoscale training demonstrations promises economic solutions for the ubiquitous deployment of life long learning machines. Neural networks realized with nanoscale memory devices may be deployed in IoT sensors, which require evaluating the device performance in harsh environmental conditions. Computational memory may be used to realize compact neuro-processing cores in CPUs. In addition to the hardware design challenges, it also brings questions on how a modern operating system could make use of new computing resources. We could also significantly improve the neural network processing power on mobile platforms, allowing more privacy-aware cloud computing services.

APPENDIX

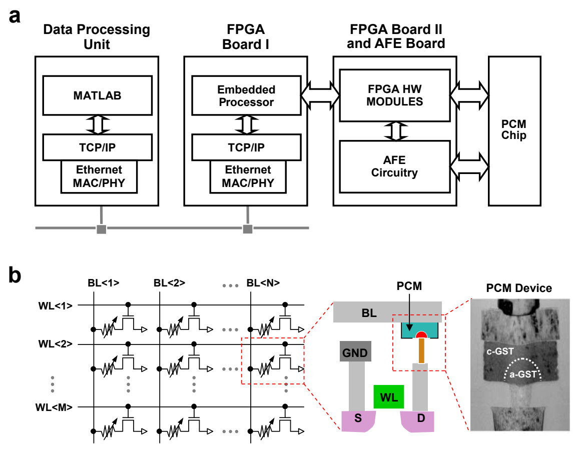
PCM-BASED HARDWARE PLATFORM

The experimental hardware platform is built around a prototype phase-change memory (PCM) chip that contains 3 million PCM devices [89]. The PCM cells are based on doped-$Ge_2Sb_2Te_5$ (d-GST) and are integrated into the chip in 90 nm CMOS baseline technology. In addition to the PCM cells, the chip integrates the circuitry for cell addressing, on-chip ADC for cell readout, and voltage- or current-mode cell programming. The experimental platform comprises the following main units:

- a high-performance analog-front-end (AFE) board that contains a number of digital-to-analog converters (DACs) along with discrete electronics, such as power supplies, voltage and current reference sources,
- a FPGA board that implements the data acquisition and the digital logic to interface with the PCM device under test and with all the electronics of the AFE board, and
- a second FPGA board with an embedded processor and Ethernet connection that implements the overall system control and data management as well as the interface with the data processing unit.

The embedded microcode allows the execution of the multi-cell programming and readout experiments which implement the matrix-vector multiplications and weight updates on the PCM chip. The hardware modules implement the interface with the external DACs used to provide various voltages to the chip for programming and readout, as well as the interface with the memory device under test, i.e., the addressing interface, the programming-mode or read-mode interfaces, etc.

There are two sub-arrays within the chip - The first sub-array contains 2 million cells, where each cell occupies an area equal to 25 F^2 (F is the technology feature size, F = 90 nm) and is accessed by a 240 nm-wide transistor. In the second sub-array containing 1 million cells, two 240 nm-wide access transistors are used in



c

PCM Chip Specifications	
Chip size	2.75mm x 2.2mm
Baseline technology	90nm CMOS
Access device	NFET, W/L=240nm/120nm
Array size	2Mcells@25F^2 + 1Mcells@50F^2
PCM device	Doped $Ge_2Sb_2Te_5$, mushroom
ADC resolution	8-bits
Supply voltage	3.3V

Figure .1 **a** Schematic of the hardware platform. **b** PCM array layout along with with a cross-sectional tunneling electron microscopy (TEM) image of the mushroom-type PCM cell. **c** PCM chip specifications.

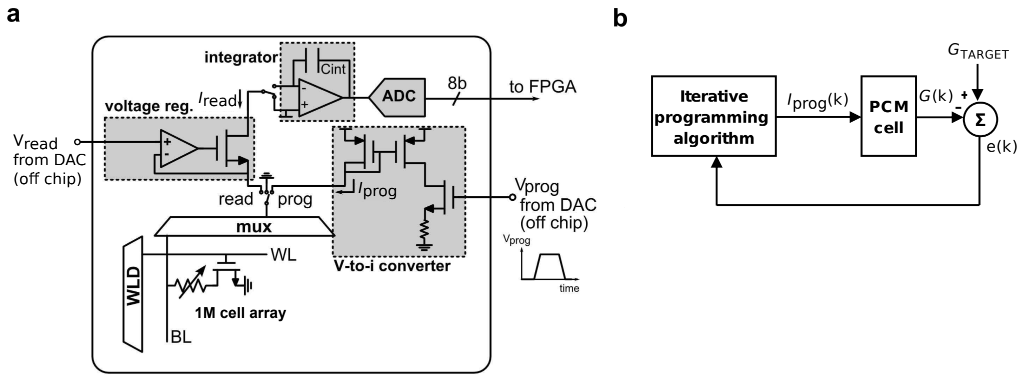


Figure .2 **a** Integrated circuit used in the PCM chip for programming and readout. **b** Iterative programming algorithm.

parallel per PCM element, and hence the cell size is twice as large (50 F^2). All the experiments performed in this work were done on the second sub-array. This sub-array is organized as a matrix of 512 word lines (WL) and 2048 bit lines (BL). The PCM cells were integrated into the chip in 90 nm CMOS technology using the key-hole process described in [88]. The bottom electrode has a radius of $\sim$ 20 nm and a length of $\sim$ 65 nm. The phase change material is $\sim$ 100 nm thick and extends to the top electrode, whose radius is $\sim$ 100 nm. The selection of one PCM cell is done by serially addressing a WL and a BL. The addresses are decoded and they then drive the WL driver and the BL multiplexer. The single selected cell can be programmed by forcing a current through the BL with a voltage-controlled current source. For reading a PCM cell, the selected BL is biased to a constant voltage (typically 100-300 mV) by a voltage regulator via a voltage V_{read} generated off-chip. The sensed current, I_{read}, is integrated by a capacitor, and the resulting voltage is then digitized by the on-chip 8-bit cyclic ADC. The total time of one read is $1\,\mu$s. The readout characteristic is calibrated via the use of on-chip reference polysilicon resistors. For programming a PCM cell, a voltage V_{prog} generated off-chip is converted on-chip into a programming current, I_{prog}. This current is then mirrored into the selected BL for the desired duration of the programming pulse. Each programming pulse is a box-type rectangular pulse with duration of 10 to 400 ns and amplitude varying between 0 and $500\,\mu$A. The access-device gate voltage (WL voltage) is kept high at 2.75 V during programming. Iterative programming, which is used for device initialization in our experiments, is achieved by applying a sequence of programming pulses [194]. After each programming pulse, a verify step is performed and the value of the cell conductance programmed in the preceding iteration is read at a voltage of 0.2 V. The programming current applied to the PCM cell in the subsequent iteration is adapted according to the sign of the value of the error between the target level and read value of the cell conductance. The programming sequence ends when the error between the target conductance and the programmed conductance of the cell is smaller than a desired margin or when the maximum number of iterations (20) has been reached. The total time of one program-and-verify step is approximately $2.5\,\mu$s.

BIBLIOGRAPHY

[1] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, December 2017.

[2] S. Cajal, *Comparative study of the sensory areas of the human cortex.* Worcester, Massachusetts: Clark University, 1899.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[4] F. Le Gall, "Powers of tensors and fast matrix multiplication," in *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC).* ACM Press, 2014, pp. 296–303.

[5] Intel Corporation. (2018) 6th gen Intel® core$^{\text{TM}}$ X-series processor family datasheet, vol. 1. [Online]. Available: https://www.intel.com/content/www/us/en/products/processors/core/6th-gen-x-series-datasheet-vol-1.html (visited on 21 January 2019)

[6] AMD Corporation. (2018) Ryzen$^{\text{TM}}$ 2nd gen threadripper$^{\text{TM}}$ 2990WX processor. [Online]. Available: https://www.amd.com/en/products/cpu/amd-ryzen-threadripper-2990wx (visited on 21 January 2019)

[7] NVIDIA Corporation. (2017) Nvidia tesla V100 gpu architecture the world's most advanced data center gpu. [Online]. Available: https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf (visited on 21 January 2019)

[8] A. Sebastian, T. Tuma, N. Papandreou, M. Le Gallo, L. Kull, T. Parnell, and E. Eleftheriou, "Temporal correlation detection using computational phase-change memory," *Nature Communications*, vol. 8, no. 1, p. 1115, 2017.

[9] V. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M. A. Kozuch, O. Mutlu, P. B. Gibbons, and T. C. Mowry, "Buddy-ram: Improving the performance and efficiency of bulk bitwise operations using DRAM," *arXiv preprint arXiv:1611.09988*, 2016.

[10] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: programming 1T1M crossbar to accelerate matrix-vector multiplication," in *Proceedings of the 53rd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2016, pp. 1–6.

[11] P. M. Sheridan, F. Cai, C. Du, W. Ma, Z. Zhang, and W. D. Lu, "Sparse coding with memristor networks," *Nature Nanotechnology*, vol. 12, no. 8, p. 784, 2017.

[12] M. Le Gallo, A. Sebastian, G. Cherubini, H. Giefers, and E. Eleftheriou, "Compressed sensing recovery using computational memory," in *Proceedings of the IEEE International Electron Devices Meeting*, 2017, pp. 28.3.1 – 28.3.4.

[13] M. Le Gallo, A. Sebastian, R. Mathis, M. Manica, H. Giefers, T. Tuma, C. Bekas, A. Curioni, and E. Eleftheriou, "Mixed-precision in-memory computing," *arXiv preprint arXiv:1701.04279*, 2017.

[14] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 3123–3131.

[15] G. Q. Bi and M. M. Poo, "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type," *Journal of neuroscience*, vol. 18, pp. 10 464–10 472, December 1998.

[16] P. J. Sjöström, G. G. Turrigiano, and S. B. Nelson, "Rate, timing, and cooperativity jointly determine cortical synaptic plasticity," *Neuron*, vol. 32, no. 6, pp. 1149–1164, 2001.

[17] A. Citri and R. C. Malenka, "Synaptic plasticity: Multiple forms, functions, and mechanisms," *Neuropsychopharmacology*, vol. 33, no. 1, pp. 18–41, August 2007.

[18] F. Zenke, E. J. Agnes, and W. Gerstner, "Diverse synaptic plasticity mechanisms orchestrated to form and retrieve memories in spiking neural networks," *Nature Communications*, vol. 6, no. 6922, April 2015.

[19] L. F. Abbott and W. G. Regehr, "Synaptic computation," *Nature*, vol. 431, pp. 796 – 803, 2004.

[20] B. Nessler, M. Pfeiffer, L. Buesing, and W. Maass, "Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity," *PLOS Computational Biology*, vol. 9, pp. 1–30, April 2013.

[21] T. Masquelier, R. Guyonneau, and S. J. Thorpe, "Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains," *PLOS ONE*, vol. 3, pp. 1–9, January 2008.

[22] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 459, no. 7250, pp. 1154–1154, June 2009.

[23] P. Mazumder, S. M. Kang, and R. Waser, "Memristors: devices, models, and applications," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1911–1919, June 2012.

[24] S. Park, H. Kim, M. Choo, J. Noh, A. Sheri, S. Jung, K. Seo, J. Park, S. Kim, W. Lee, J. Shin, D. Lee, G. Choi, J. Woo, E. Cha, J. Jang, C. Park, M. Jeon, B. Lee, B. H. Lee, and H. Hwang, "RRAM-based synapse for neuromorphic system with pattern recognition function," in *Proceedings of the IEEE International Electron Devices Meeting*, December 2012, pp. 10.2.1–10.2.4.

[25] S. Park, A. Sheri, J. Kim, J. Noh, J. Jang, M. Jeon, B. Lee, B. R. Lee, B. H. Lee, and H. Hwang, "Neuromorphic speech systems using advanced ReRAM-based synapse," in *Proceedings of the IEEE International Electron Devices Meeting*, December 2013, pp. 25.6.1–25.6.4.

[26] W. Chen, R. Fang, M. B. Balaban, W. Yu, Y. Gonzalez-Velo, H. J. Barnaby, and M. N. Kozicki, "A CMOS-compatible electronic synapse device based on Cu/SiO_2/W programmable metallization cells," *Nanotechnology*, vol. 27, no. 25, p. 255202, June 2016.

[27] M. Suri, O. Bichler, D. Querlioz, O. Cueto, L. Perniola, V. Sousa, D. Vuillaume, C. Gamrat, and B. DeSalvo, "Phase change memory as synapse for ultra-dense neuromorphic systems: Application to complex visual pattern extraction," in *Proceedings of the International Electron Devices Meeting*, December 2011, pp. 4.4.1–4.4.4.

[28] B. L. Jackson, B. Rajendran, G. S. Corrado, M. Breitwisch, G. W. Burr, R. Cheek, K. Gopalakrishnan, S. Raoux, C. T. Rettner, A. Padilla *et al.*, "Nanoscale electronic synapses using phase change devices," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 9, no. 2, p. 12, 2013.

[29] I.-T. Wang, Y.-C. Lin, Y.-F. Wang, C.-W. Hsu, and T.-H. Hou, "3D synaptic architecture with ultralow sub-10 fJ energy per spike for neuromorphic computation," in *Proceedings of the IEEE International Electron Devices Meeting*, December 2014, pp. 28.5.1–28.5.4.

[30] S. Nandakumar, S. R. Kulkarni, A. V. Babu, and B. Rajendran, "Building brain-inspired computing systems: Examining the role of nanoscale devices," *IEEE Nanotechnology Magazine*, vol. 12, no. 3, pp. 19–35, September 2018.

[31] H. S. P. Wong, S. Raoux, S. Kim, J. Liang, J. P. Reifenberg, B. Rajendran, M. Asheghi, and K. E. Goodson, "Phase change memory," *Proceedings of the IEEE*, vol. 98, no. 12, pp. 2201–2227, December 2010.

[32] G. W. Burr, M. J. Brightsky, A. Sebastian, H. Y. Cheng, J. Y. Wu, S. Kim, N. E. Sosa, N. Papandreou, H. L. Lung, H. Pozidis, E. Eleftheriou, and C. H. Lam, "Recent progress in phase-change memory technology," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 2, pp. 146–162, June 2016.

[33] S. R. Ovshinsky, "Reversible electrical switching phenomena in disordered structures," *Physical Review Letters*, vol. 21, no. 20, pp. 1450–1453, 1968.

[34] S. Lai and T. Lowrey, "OUM - a 180 nm nonvolatile memory cell element technology for stand alone and embedded applications," in *Proceedings of the International Electron Devices Meeting*, 2001, pp. 36.5.1–36.5.4.

[35] A. Pirovano, A. Lacaita, F. Pellizzer, S. Kostylev, A. Benvenuti, and R. Bez, "Low-field amorphous state resistance and threshold voltage drift in chalcogenide materials," *IEEE Transactions on Electron Devices*, vol. 51, no. 5, pp. 714–719, May 2004.

[36] D. Kuzum, R. Jeyasingh, B. Lee, and H.-S. P. Wong, "Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing," *Nano Letters*, vol. 12, no. 5, pp. 2179–2186, 2011.

[37] G. Burr, R. Shelby, C. di Nolfo, J. Jang, R. Shenoy, P. Narayanan, K. Virwani, E. Giacometti, B. Kurdi, and H. Hwang, "Experimental demonstration and tolerancing of a large-scale neural network (165,000 synapses), using phase-change memory as the synaptic weight element," in *Proceedings of the IEEE International Electron Devices Meeting*, December 2014, pp. 29.5.1–29.5.4.

[38] S. Nandakumar, I. Boybat, M. L. Gallo, A. Sebastian, B. Rajendran, and E. Eleftheriou, "Supervised learning in spiking neural networks with MLC PCM synapses," in *Proceedings of the 75th Device Research Conference*, 2017.

[39] R. Waser and M. Aono, "Nanoionics-based resistive switching memories," *Nature materials*, vol. 6, no. 11, pp. 833–40, November 2007.

[40] M. D. Pickett, D. B. Strukov, J. L. Borghetti, J. J. Yang, G. S. Snider, D. R. Stewart, and R. S. Williams, "Switching dynamics in titanium dioxide memristive devices," *Journal of Applied Physics*, vol. 106, no. 7, p. 074508, 2009.

[41] L. Goux, P. Czarnecki, Y. Y. Chen, L. Pantisano, X. P. Wang, R. Degraeve, B. Govoreanu, M. Jurczak, D. J. Wouters, and L. Altimime, "Evidences of oxygen-mediated resistive-switching mechanism in TiN\ HfO_2\Pt cells," *Applied Physics Letters*, vol. 97, no. 24, pp. 2–5, 2010.

[42] W. T. C. Chien, Y. R. Chen, Y. C. Chen, A. T. Chuang, F. M. Lee, Y. Y. Lin, E. K. Lai, Y. H. Shih, K. Y. Hsieh, and C.-Y. Lu, "A forming-free wox resistive memory using a novel self-aligned field enhancement feature with excellent reliability and scalability," in *Proceedings of the IEEE International Electron Devices Meeting*, 2010, pp. 19.2.1–19.2.4.

[43] M. Kund, G. Beitel, C.-U. Pinnow, T. Rohr, J. Schumann, R. Symanczyk, K.-d. Ufert, and G. Muller, "Conductive bridging RAM (CBRAM): an emerging non-volatile memory technology scalable to sub 20nm," in *Proceedings of the IEEE International Electron Devices Meeting*, 2005, pp. 754–757.

[44] X. Zhu, W. Su, Y. Liu, B. Hu, L. Pan, W. Lu, J. Zhang, and R. W. Li, "Observation of conductance quantization in oxide-based resistive switching memory," *Advanced Materials*, vol. 24, no. 29, pp. 3941–3946, 2012.

[45] S. Nandakumar, M. Minvielle, S. Nagar, C. Dubourdieu, and B. Rajendran, “A 250 mV Cu/SiO2/W memristor with half-integer quantum conductance states,” *Nano letters*, vol. 16, no. 3, pp. 1602–1608, 2016.

[46] S. Nandakumar and B. Rajendran, “Synaptic plasticity in a memristive device below 500 mv,” in *Proceedings of the 231st Meeting of the Electrochemical Society*, 2017.

[47] I. Valov, R. Waser, J. R. Jameson, and M. N. Kozicki, “Electrochemical metallization memories–fundamentals, applications, prospects,” *Nanotechnology*, vol. 22, no. 25, p. 254003, 2011.

[48] J. J. Yang, D. B. Strukov, and D. R. Stewart, “Memristive devices for computing,” *Nature Nanotechnology*, vol. 8, no. 1, pp. 13–24, 2013.

[49] S. Park, J. Noh, M.-L. Choo, A. M. Sheri, M. Chang, Y.-B. Kim, C. J. Kim, M. Jeon, B.-G. Lee, B. H. Lee, and H. Hwang, “Nanoscale RRAM-based synaptic electronics: toward a neuromorphic computing device,” *Nanotechnology*, vol. 24, no. 38, p. 384009, September 2013.

[50] N. Panwar, D. Kumar, N. Upadhyay, P. Arya, U. Ganguly, and B. Rajendran, “Memristive synaptic plasticity in $Pr_{0.7}Ca_{0.3}MnO_3$ RRAM by bio-mimetic programming,” in *Proceedings of the 72nd Annual Device Research Conference*, June 2014, pp. 135–136.

[51] S. Choi, S. H. Tan, Z. Li, Y. Kim, C. Choi, P.-Y. Chen, H. Yeon, S. Yu, and J. Kim, “SiGe epitaxial memory for neuromorphic computing with reproducible high performance based on engineered dislocations,” *Nature Materials*, vol. 17, pp. 1–6, January 2018.

[52] N. Locatelli, V. Cros, and J. Grollier, “Spin-torque building blocks,” *Nature Materials*, vol. 13, no. 1, pp. 11–20, 2014.

[53] D. C. Ralph and M. D. Stiles, “Spin transfer torques,” *Journal of Magnetism and Magnetic Materials*, vol. 320, no. 7, pp. 1190–1216, 2008.

[54] T. Kawahara, K. Ito, R. Takemura, and H. Ohno, “Spin-transfer torque RAM technology: Review and prospect,” *Microelectronics Reliability*, vol. 52, no. 4, pp. 613–627, 2012.

[55] A. Sengupta and K. Roy, “Encoding neural and synaptic functionalities in electron spin: A pathway to efficient neuromorphic computing,” *Applied Physics Reviews*, vol. 4, no. 4, p. 041105, 2017.

[56] H. Zhao, B. Glass, P. K. Amiri, A. Lyle, Y. Zhang, Y. J. Chen, G. Rowlands, P. Upadhyaya, Z. Zeng, J. A. Katine, J. Langer, K. Galatsis, H. Jiang, K. L. Wang, I. N. Krivorotov, and J. P. Wang, “Sub-200ps spin transfer torque switching in in-plane magnetic tunnel junctions with interface perpendicular anisotropy,” *Journal of Physics D: Applied Physics*, vol. 45, no. 2, 2012.

[57] A. F. Vincent, J. Larroque, N. Locatelli, N. B. Romdhane, O. Bichler, C. Gamrat, W. S. Zhao, J.-O. Klein, S. Galdin-Retailleau, and D. Querlioz, "Spin-transfer torque magnetic memory as a stochastic memristive synapse for neuromorphic systems," *IEEE transactions on biomedical circuits and systems*, vol. 9, no. 2, pp. 166–174, 2015.

[58] U. Roy, T. Pramanik, L. F. Register, and S. K. Banerjee, "Write error rate of spin-transfer-torque random access memory including micromagnetic effects using rare event enhancement," *IEEE Transactions on Magnetics*, vol. 52, no. 10, pp. 1–6, 2016.

[59] J. Bill and R. Legenstein, "A compound memristive synapse model for statistical learning through STDP in spiking neural networks," *Frontiers in Neuroscience*, vol. 8, pp. 1–18, December 2014.

[60] A. Singha, B. Muralidharan, and B. Rajendran, "Analog memristive time dependent learning using discrete nanoscale RRAM devices," in *Proceedings of the International Joint Conference on Neural Networks*, July 2014, pp. 2248–2255.

[61] V. Garcia and M. Bibes, "Ferroelectric tunnel junctions for information storage and processing," *Nature Communications*, vol. 5, pp. 1–12, 2014.

[62] J. Guyonnet, *Ferroelectric Domain Walls*, ser. Springer Theses. Cham: Springer International Publishing, 2014.

[63] A. Chanthbouala, V. Garcia, R. O. Cherifi, K. Bouzehouane, S. Fusil, X. Moya, S. Xavier, H. Yamada, C. Deranlot, N. D. Mathur, M. Bibes, A. Barthélémy, and J. Grollier, "A ferroelectric memristor," *Nature Materials*, vol. 11, no. 10, pp. 860–864, October 2012.

[64] S. Boyn, J. Grollier, G. Lecerf, B. Xu, N. Locatelli, S. Fusil, S. Girod, C. Carrétéro, K. Garcia, S. Xavier, J. Tomas, L. Bellaiche, M. Bibes, A. Barthélémy, S. Saïghi, and V. Garcia, "Learning through ferroelectric domain dynamics in solid-state synapses," *Nature Communications*, vol. 8, pp. 1–7, 2017.

[65] S. Goswami, A. J. Matula, S. P. Rath, S. Hedström, S. Saha, M. Annamalai, D. Sengupta, A. Patra, S. Ghosh, H. Jani, S. Sarkar, M. R. Motapothula, C. A. Nijhuis, J. Martin, S. Goswami, V. S. Batista, and T. Venkatesan, "Robust resistive memory devices using solution-processable metal-coordinated azo aromatics," *Nature Materials*, vol. 16, no. 12, pp. 1216–1224, October 2017.

[66] X. Yang, C. Wang, J. Shang, C. Zhang, H. Tan, X. Yi, L. Pan, W. Zhang, F. Fan, Y. Liu, Y. Chen, G. Liu, and R.-W. Li, "An organic terpyridyl-iron polymer based memristor for synaptic plasticity and learning behavior simulation," *RSC Advances*, vol. 6, no. 30, pp. 25 179–25 184, 2016.

[67] T. Berzina, A. Smerieri, M. Bernab, A. Pucci, G. Ruggeri, V. Erokhin, and M. P. Fontana, "Optimization of an organic memristor as an adaptive memory element," *Journal of Applied Physics*, vol. 105, no. 12, 2009.

[68] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed precision training," *Iclr2018*, pp. 1–10, October 2017.

[69] S. Nandakumar, M. Le Gallo, I. Boybat, B. Rajendran, A. Sebastian, and E. Eleftheriou, "Mixed-precision architecture based on computational memory for training deep neural networks," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 2018, pp. 1–5.

[70] T. M. Bartol, C. Bromer, J. P. Kinney, M. A. Chirillo, J. N. Bourne, K. M. Harris, and T. J. Sejnowski, "Hippocampal spine head sizes are highly precise," *bioRxiv*, p. 016329, 2015.

[71] T. J. Sejnowski, *Micro-, Meso- and Macro-Dynamics of the Brain*, ser. Research and Perspectives in Neurosciences, G. Buzsáki and Y. Christen, Eds. Cham: Springer International Publishing, 2016. [Online]. Available: http://link.springer.com/10.1007/978-3-319-28802-4

[72] G. W. Burr, R. M. Shelby, S. Sidler, C. Di Nolfo, J. Jang, I. Boybat, R. S. Shenoy, P. Narayanan, K. Virwani, E. U. Giacometti *et al.*, "Experimental demonstration and tolerancing of a large-scale neural network (165,000 synapses) using phase-change memory as the synaptic weight element," *IEEE Transactions on Electron Devices*, vol. 62, no. 11, pp. 3498–3507, 2015.

[73] T. Gokmen and Y. Vlasov, "Acceleration of deep neural network training with resistive cross-point devices: design considerations," *Frontiers in Neuroscience*, vol. 10, p. 333, 2016.

[74] P. Werbos, *Beyond regression: new tools for prediction and analysis in the behavioral sciences.* Ph.D. thesis, Harvard University, Cambridge, MA, 1974.

[75] A. L. Hodgkin and A. F. Huxley, "Currents carried by sodium and potassium ions through the membrane of the giant axon of Loligo," *The Journal of Physiology*, vol. 116, no. 4, pp. 449–472, April 1952.

[76] E. M. Izhikevich and G. M. Edelman, "Large-scale model of mammalian thalamocortical systems," *Proceedings of the National Academy of Sciences*, vol. 105, no. 9, pp. 3593–3598, 2008.

[77] R. Brette and W. Gerstner, "Adaptive exponential integrate-and-fire model as an effective description of neuronal activity," *Journal of Neurophysiology*, vol. 94, no. 5, pp. 3637–3642, 2005, pMID: 16014787.

[78] L. F. Abbott, "Lapicque's introduction of the integrate-and-fire model neuron (1907)," *Brain Research Bulletin*, vol. 50, pp. 303–304, 1999.

[79] N. Anwani and B. Rajendran, "Normad-normalized approximate descent based supervised learning rule for spiking neurons," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, 2015, pp. 1–8.

[80] S. Woźniak, A. Pantazi, and E. Eleftheriou, “Deep networks incorporating spiking neural dynamics,” *arXiv preprint arXiv:1812.07040*, pp. 1–9, December 2018.

[81] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[82] M. D. Zeiler, “ADADELTA: an adaptive learning rate method,” *CoRR*, vol. abs/1212.5701, 2012.

[83] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *CoRR*, vol. abs/1412.6980, 2014.

[84] S. Nandakumar, M. Le Gallo, I. Boybat, B. Rajendran, A. Sebastian, and E. Eleftheriou, “A phase-change memory model for neuromorphic computing,” *Journal of Applied Physics*, vol. 124, no. 15, p. 152135, 2018.

[85] A. Sebastian, M. Le Gallo, and D. Krebs, “Crystal growth within a phase change memory cell,” *Nature Communications*, vol. 5, p. 4314, 2014.

[86] G. W. Burr *et al.*, “Neuromorphic computing using non-volatile memory,” *Advances in Physics: X*, vol. 2, no. 1, pp. 89–124, 2017.

[87] M. Le Gallo, D. Krebs, F. Zipoli, M. Salinga, and A. Sebastian, “Collective structural relaxation in phase-change memory devices,” *Advanced Electronics Materials*, 2018.

[88] M. Breitwisch, T. Nirschl, C. Chen, Y. Zhu, M. Lee, M. Lamorey, G. Burr, E. Joseph, A. Schrott, J. Philipp *et al.*, “Novel lithography-independent pore phase change memory,” in *Proceedings of the IEEE Symposium on VLSI Technology*, 2007, pp. 100–101.

[89] G. F. Close *et al.*, “Device, circuit and system-level analysis of noise in multi-bit phase-change memory,” in *Proceedings of the IEEE International Electron Devices Meeting*, 2010, pp. 29.5.1–29.5.4.

[90] N. Papandreou, H. Pozidis, A. Pantazi, A. Sebastian, M. Breitwisch, C. Lam, and E. Eleftheriou, “Programming algorithms for multilevel phase-change memory,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 2011, pp. 329–332.

[91] T. Tuma, A. Pantazi, M. Le Gallo, A. Sebastian, and E. Eleftheriou, “Stochastic phase-change neurons,” *Nature Nanotechnology*, vol. 11, pp. 693–699, 2016.

[92] M. Le Gallo, T. Tuma, F. Zipoli, A. Sebastian, and E. Eleftheriou, “Inherent stochasticity in phase-change memory devices,” in *Proceedings of the 46th IEEE European Solid-State Device Research Conference*, 2016, pp. 373–376.

[93] I. Boybat, M. Le Gallo, T. Moraitis, Y. Leblebici, A. Sebastian, and E. Eleftheriou, "Stochastic weight updates in phase-change memory-based synapses and their influence on artificial neural networks," in *Proceedings of the 13th IEEE Conference on Ph. D. Research in Microelectronics and Electronics (PRIME)*, 2017, pp. 13–16.

[94] N. Gong, T. Ide, S. Kim, I. Boybat, A. Sebastian, V. Narayanan, and T. Ando, "Signal and noise extraction from analog memory elements for neuromorphic computing," *Nature Communications*, vol. 9, p. 2102, 2018.

[95] I. Boybat, M. Le Gallo, S. Nandakumar, T. Moraitis, T. Parnell, T. Tuma, B. Rajendran, Y. Leblebici, A. Sebastian, and E. Eleftheriou, "Neuromorphic computing with multi-memristive synapses," *Nature Communications*, vol. 9, no. 1, p. 2514, 2018.

[96] D. Ielmini, D. Sharma, S. Lavizzari, and A. L. Lacaita, "Reliability impact of chalcogenide-structure relaxation in phase-change memory (PCM) cells-Part I: experimental study," *IEEE Transactions on Electron Devices*, vol. 56, no. 5, pp. 1070–1077, 2009.

[97] M. Boniardi and D. Ielmini, "Physical origin of the resistance drift exponent in amorphous phase change materials," *Applied Physics Letters*, vol. 98, no. 24, p. 243506, 2011.

[98] M. Nardone, V. Kozub, I. Karpov, and V. Karpov, "Possible mechanisms for 1/f noise in chalcogenide glasses: A theoretical description," *Physical Review B*, vol. 79, no. 16, p. 165206, 2009.

[99] D.-H. Kim, F. Merget, M. Först, and H. Kurz, "Three-dimensional simulation model of switching dynamics in phase change random access memory cells," *Journal of Applied Physics*, vol. 101, no. 6, p. 064512, mar 2007.

[100] C. Ma, J. He, J. Lu, J. Zhu, and Z. Hu, "Modeling of the temperature profiles and thermoelectric effects in phase change memory cells," *Applied Sciences*, vol. 8, no. 8, p. 1238, July 2018.

[101] D. Ielmini and Y. Zhang, "Analytical model for subthreshold conduction and threshold switching in chalcogenide-based memory devices," *Journal of Applied Physics*, vol. 102, no. 5, p. 054517, 2007.

[102] S. Raoux, F. Xiong, M. Wuttig, and E. Pop, "Phase change materials and phase change memory," *MRS Bulletin*, vol. 39, no. 08, pp. 703–710, 2014.

[103] F. Rao, K. Ding, Y. Zhou, Y. Zheng, M. Xia, S. Lv, Z. Song, S. Feng, I. Ronneberger, R. Mazzarello, W. Zhang, and E. Ma, "Reducing the stochasticity of crystal nucleation to enable subnanosecond memory writing," *Science*, vol. 358, no. 6369, pp. 1423–1427, December 2017.

[104] M. Suri, O. Bichler, D. Querlioz, B. Traor?, O. Cueto, L. Perniola, V. Sousa, D. Vuillaume, C. Gamrat, and B. DeSalvo, “Physical aspects of low power synapses based on phase change memory devices,” *Journal of Applied Physics*, vol. 112, no. 5, p. 054904, September 2012.

[105] J. Liu, X. Xu, L. Brush, and M. P. Anantram, “A multi-scale analysis of the crystallization of amorphous germanium telluride using ab initio simulations and classical crystallization theory,” *Journal of Applied Physics*, vol. 115, no. 2, 2014.

[106] J. Liu, “Microscopic origin of electron transport properties and ultrascalability of amorphous phase change material germanium telluride,” *IEEE Transactions on Electron Devices*, vol. 64, no. 5, pp. 2207–2215, May 2017.

[107] F. Xiong, E. Yalon, A. Behnam, C. Neumann, K. Grosse, S. Deshmukh, and E. Pop, “Towards ultimate scaling limits of phase-change memory,” in *Proceedings of the IEEE International Electron Devices Meeting*, December 2016, pp. 4.1.1–4.1.4.

[108] S. Kim and H.-S. Wong, “Analysis of temperature in phase change memory scaling,” *IEEE Electron Device Letters*, vol. 28, no. 8, pp. 697–699, August 2007.

[109] S. Ambrogio, P. Narayanan, H. Tsai, R. M. Shelby, I. Boybat, C. di Nolfo, S. Sidler, M. Giordano, M. Bodini, N. C. P. Farinha, B. Killeen, C. Cheng, Y. Jaoudi, and G. W. Burr, “Equivalent-accuracy accelerated neural-network training using analogue memory,” *Nature*, vol. 558, no. 7708, pp. 60–67, 2018.

[110] A. Athmanathan, M. Stanisavljevic, N. Papandreou, H. Pozidis, and E. Eleftheriou, “Multilevel-cell phase-change memory: A viable technology,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 1, pp. 87–100, March 2016.

[111] G. W. Burr, M. J. Breitwisch, M. Franceschini, D. Garetto, K. Gopalakrishnan, B. Jackson, B. Kurdi, C. Lam, L. a. Lastras, A. Padilla, B. Rajendran, S. Raoux, and R. S. Shenoy, “Phase change memory technology,” *Journal of Vacuum Science & Technology B, Nanotechnology and Microelectronics: Materials, Processing, Measurement, and Phenomena*, vol. 28, no. 2, pp. 223–262, March 2010.

[112] T. T. Hoang, M. Sjalander, and P. Larsson-Edefors, “A high-speed, energy-efficient two-cycle multiply-accumulate (MAC) architecture and its application to a double-throughput MAC unit,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 12, pp. 3073–3081, December 2010.

[113] S. Nandakumar, M. Le Gallo, I. Boybat, B. Rajendran, A. Sebastian, and E. Eleftheriou, “Mixed-precision training of deep neural networks using computational memory,” *arXiv preprint arXiv:1712.01192*, December 2017.

[114] H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, and M.-J. Tsai, "Metal–oxide RRAM," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, 2012.

[115] A. D. Kent and D. C. Worledge, "A new spin on magnetic memories," *Nature Nanotechnology*, vol. 10, no. 3, pp. 187–191, 2015.

[116] D. Querlioz, O. Bichler, and C. Gamrat, "Simulation of a memristor-based spiking neural network immune to device variations," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, 2011, pp. 1775–1781.

[117] D. Fugazza, D. Ielmini, S. Lavizzari, and A. Lacaita, "Distributed-Poole-Frenkel modeling of anomalous resistance scaling and fluctuations in phase-change memory (PCM) devices," in *Proceedings of the IEEE International Electron Devices Meeting*, 2009, pp. 1–4.

[118] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. PMLR, May 2010, pp. 249–256.

[119] M. Nardone, M. Simon, I. V. Karpov, and V. G. Karpov, "Electrical conduction in chalcogenide glasses of phase change memory," *Journal of Applied Physics*, vol. 112, no. 7, p. 071101, 2012.

[120] S. Kim, T. Gokmen, H. Lee, and W. E. Haensch, "Analog CMOS-based resistive processing unit for deep neural network training," in *Proceedings of the 60th IEEE International Midwest Symposium on Circuits and Systems*, August 2017, pp. 422–425.

[121] T. Gokmen, M. Onen, and W. Haensch, "Training deep convolutional neural networks with resistive cross-point devices," *Frontiers in Neuroscience*, vol. 11, pp. 1–22, October 2017.

[122] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, pp. 2672–2680.

[123] B. B. Averbeck, P. E. Latham, and A. Pouget, "Neural correlations, population coding and computation." *Nature reviews. Neuroscience*, vol. 7, no. 5, pp. 358–366, 2006.

[124] J.-M. Fellous, "Discovering spike patterns in neuronal responses," *Journal of Neuroscience*, vol. 24, no. 12, pp. 2989–3001, 2004.

[125] W. W. Koelmans, A. Sebastian, V. P. Jonnalagadda, D. Krebs, L. Dellmann, and E. Eleftheriou, “Projected phase-change memory devices,” *Nature communications*, vol. 6, p. 8181, 2015.

[126] J. M. Howard, V. Craciun, C. Essary, and R. K. Singh, “Interfacial layer formation during high-temperature annealing of ZrO_2 thin films on Si,” *Applied Physics Letters*, vol. 81, no. 18, p. 3431, 2002.

[127] S. J. Wang, P. C. Lim, a. C. H. Huan, C. L. Liu, J. W. Chai, S. Y. Chow, J. S. Pan, Q. Li, and C. K. Ong, “Reaction of SiO_2 with hafnium oxide in low oxygen pressure,” *Applied Physics Letters*, vol. 82, no. 13, p. 2047, 2003.

[128] E.-C. Cho, S. Park, X. Hao, D. Song, G. Conibeer, S.-C. Park, and M. a. Green, “Silicon quantum dot/crystalline silicon solar cells,” *Nanotechnology*, vol. 19, no. 24, p. 245201, June 2008.

[129] R. Landauer, “Spatial variation of currents and fields due to localized scatterers in metallic conduction,” *IBM Journal of Research and Development*, vol. 1, no. 3, pp. 223–231, July 1957.

[130] P. F. Bagwell and T. P. Orlando, “Landauer’s conductance formula and its generalization to finite voltages,” *Physical Review B*, vol. 40, no. 3, pp. 1456–1464, July 1989.

[131] S. Tappertzhofen, S. Menzel, I. Valov, and R. Waser, “Redox processes in silicon dioxide thin films using copper microelectrodes,” *Applied Physics Letters*, vol. 99, no. 20, p. 203103, 2011.

[132] C. Schindler, S. C. P. Thermadam, R. Waser, and M. N. Kozicki, “Bipolar and unipolar resistive switching in Cu-doped SiO_2,” *IEEE Transactions on Electron Devices*, vol. 54, no. 10, pp. 2762–2768, October 2007.

[133] S. P. Thermadam, S. Bhagat, T. Alford, Y. Sakaguchi, M. Kozicki, and M. Mitkova, “Influence of Cu diffusion conditions on the switching of Cu–SiO_2-based resistive memory devices,” *Thin Solid Films*, vol. 518, no. 12, pp. 3293–3298, April 2010.

[134] Y. Bernard, V. Renard, P. Gonon, and V. Jousseaume, “Back-end-of-line compatible conductive bridging RAM based on Cu and SiO_2,” *Microelectronic Engineering*, vol. 88, no. 5, pp. 814–816, May 2011.

[135] C.-Y. Liu, Y.-Y. Tsai, S.-K. Liu, and Y.-H. Huang, “Retention failure mechanism for low-resistance-states of Cu-doped SiO_2 resistive memory,” *Ferroelectrics*, vol. 459, no. 1, pp. 99–104, January 2014.

[136] C.-Y. Liu, Y.-Y. Tsai, W.-T. Fang, and H.-Y. Wang, “Resistive switching characteristics of a SiO_x layer with cf4 plasma treatment,” *Journal of Nanomaterials*, vol. 2014, pp. 1–5, 2014.

[137] L. Olesen, E. Laegsgaard, I. Stensgaard, F. Besenbacher, J. Schiotz, P. Stoltze, K. W. Jacobsen, and J. K. Norskov, “Quantized conductance in an atom-sized point contact,” *Physical Review Letters*, vol. 72, no. 14, pp. 2251–2254, April 1994.

[138] N. Agraït, A. L. Yeyati, and J. M. van Ruitenbeek, “Quantum properties of atomic-sized conductors,” *Physics Reports*, vol. 377, no. 2-3, pp. 81–279, 2003.

[139] B. J. Van Wees, H. Van Houten, C. W. J. Beenakker, J. G. Williamson, L. P. Kouwenhoven, D. Van Der Marel, and C. T. Foxon, “Quantized conductance of point contacts in a two-dimensional electron gas,” *Physical Review Letters*, vol. 60, no. 9, pp. 848–850, February 1988.

[140] D. A. Wharam, T. J. Thornton, R. Newbury, M. Pepper, H. Ahmed, J. E. F. Frost, D. G. Hasko, D. C. Peacock, D. a. Ritchie, and G. a. C. Jones, “One-dimensional transport and the quantisation of the ballistic resistance,” *Journal of Physics C: Solid State Physics*, vol. 21, no. 8, pp. L209–L214, March 1988.

[141] L. I. Glazman and A. V. Khaetskii, “Nonlinear quantum conductance of a lateral microconstraint in a heterostructure,” *EPL (Europhysics Letters)*, vol. 9, no. 3, p. 263, 1989.

[142] N. K. Patel, J. T. Nicholls, L. Martn-Moreno, M. Pepper, J. E. F. Frost, D. a. Ritchie, and G. a. C. Jones, “Evolution of half plateaus as a function of electric field in a ballistic quasi-one-dimensional constriction,” *Physical Review B*, vol. 44, no. 24, pp. 13 549–13 555, December 1991.

[143] L. Martin-Moreno, J. T. Nicholls, N. K. Patel, and M. Pepper, “Non-linear conductance of a saddle-point constriction,” *Journal of Physics: Condensed Matter*, vol. 4, no. 5, pp. 1323–1333, February 1992.

[144] H. Xu, “Theory of nonlinear ballistic transport in quasi-one-dimensional constrictions,” *Physical Review B*, vol. 47, no. 23, pp. 15 630–15 637, June 1993.

[145] E. Miranda, S. Kano, C. Dou, K. Kakushima, J. Sune, and H. Iwai, “Nonlinear conductance quantization effects in CeOx/SiO_2-based resistive switching devices,” *Applied Physics Letters*, vol. 101, no. 1, p. 012910, 2012.

[146] A. Mehonic, A. Vrajitoarea, S. Cueff, S. Hudziak, H. Howe, C. Labbé, R. Rizk, M. Pepper, and A. J. Kenyon, “Quantum conductance in silicon oxide resistive memory devices,” *Scientific Reports*, vol. 3, p. 2708, September 2013.

[147] S. O. Kasap, *Principles of Electronic Materials and Devices.* Boston, MA: McGraw-Hill, 2002.

[148] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser,

R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[149] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch, C. di Nolfo, P. Datta, A. Amir, B. Taba, M. D. Flickner, and D. S. Modha, “Convolutional networks for fast, energy-efficient neuromorphic computing,” *Proceedings of the National Academy of Sciences*, 2016.

[150] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, “The SpiNNaker project,” *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, May 2014.

[151] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J. M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, “Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations,” *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, May 2014.

[152] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner, “A wafer-scale neuromorphic hardware system for large-scale neural modeling,” *Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS”10)*, pp. 1947–1950, 2010.

[153] J. Gehlhaar, “Neuromorphic processing: A new frontier in scaling computer architecture,” in *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2014, pp. 317–318.

[154] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri, “A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses,” *Frontiers in Neuroscience*, vol. 9, p. 141, 2015.

[155] Y. H. Chen, T. Krishna, J. S. Emer, and V. Sze, “Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks,” *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, January 2017.

[156] S. Bang, J. Wang, Z. Li, C. Gao, Y. Kim, Q. Dong, Y.-P. Chen, L. Fick, X. Sun, R. Dreslinski, T. Mudge, H. S. Kim, D. Blaauw, and D. Sylvester, “A 288uW programmable deep-learning processor with 270KB on-chip weight storage using non-uniform memory hierarchy for mobile intelligence,” in *Proceedings of the IEEE International Solid-State Circuits Conference*, February 2017.

[157] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, “Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable Convolutional Neural Network processor in 28nm FDSOI,” in *2017 IEEE*

International Solid-State Circuits Conference (ISSCC), February 2017, pp. 246–247.

[158] M. Jerry, W. y. Tsai, B. Xie, X. Li, V. Narayanan, A. Raychowdhury, and S. Datta, "Phase transition oxide neuron for spiking neural networks," in *Proceedings of the 74th Annual Device Research Conference*, June 2016, pp. 1–2.

[159] S. Mandal, A. El-Amin, K. Alexander, B. Rajendran, and R. Jha, "Novel synaptic memory device for neuromorphic computing," *Nature Scientific Reports*, vol. 4, no. 5333, 2014.

[160] B. Rajendran and F. Alibart, "Neuromorphic computing based on emerging memory technologies," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. PP, no. 99, pp. 1–14, 2016.

[161] D. Querlioz, O. Bichler, A. F. Vincent, and C. Gamrat, "Bioinspired programming of memory devices for implementing an inference engine," *Proceedings of the IEEE*, vol. 103, no. 8, pp. 1398–1416, 2015.

[162] D. Kuzum, R. Jeyasingh, S. Yu, and H.-S. Wong, "Low-energy robust neuromorphic computation using synaptic devices," *Electron Devices, IEEE Transactions on*, vol. 59, no. 12, pp. 3489–3494, December 2012.

[163] M. Prezioso, F. Merrikh-Bayat, B. Hoskins, G. Adam, K. Likharev, and D. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, pp. 61–64, 2015.

[164] B. Rajendran, Y. Liu, J. Seo, K. Gopalakrishnan, L. Chang, D. Friedman, and M. Ritter, "Specifications of nanoscale devices and circuits for neuromorphic computational systems," *IEEE Transactions on Electron Devices*, pp. 246–253, January 2013.

[165] N. F. Mott and R. W. Gurney, *Electronic processes in ionic crystals.* Oxford, London: Oxford University Press, 1948.

[166] C. Schindler, G. Staikov, and R. Waser, "Electrode kinetics of Cu-SiO_2-based resistive switching cells: Overcoming the voltage-time dilemma of electrochemical metallization memories," *Applied Physics Letters*, vol. 94, no. 7, pp. 92–95, 2009.

[167] S. Menzel, U. Böttger, M. Wimmer, and M. Salinga, "Physics of the switching kinetics in resistive memories," *Advanced Functional Materials*, vol. 25, no. 40, pp. 6306–6325, October 2015.

[168] S. Yu and H.-S. Philip Wong, "Modeling the switching dynamics of programmable-metallization-cell (PMC) memory and its application as synapse device for a neuromorphic computation system," in *Proceedings of the IEEE International Electron Devices Meeting*, no. 6, December 2010, pp. 22.1.1–22.1.4.

[169] G. Stuart, N. Spruston, B. Sakmann, and M. Häusser, “Action potential initiation and backpropagation in neurons of the mammalian CNS,” *Trends in Neurosciences*, vol. 20, no. 3, pp. 125–131, March 1997.

[170] B. F. Grewe, A. Bonnan, and A. Frick, “Back-propagation of physiological action potential output in dendrites of slender-tufted L5A pyramidal neurons,” *Frontiers in Cellular Neuroscience*, vol. 4, pp. 1–11, May 2010.

[171] S. Nandakumar and B. Rajendran, “Physics-based switching model for Cu/SiO_2/W quantum memristor,” in *Proceedings of the 74th Device Research Conference*, 2016.

[172] M. Prezioso, F. Merrikh-Bayat, B. Hoskins, K. Likharev, and D. Strukov, “Self-adaptive spike-time-dependent plasticity of metal-oxide memristors,” *Nature Scientific Reports 6, art. 21331*, 2016.

[173] C. Wang, W. He, Y. Tong, and R. Zhao, “Investigation and manipulation of different analog behaviors of memristor as electronic synapse for neuromorphic applications,” *Scientific Reports*, vol. 6, no. 22970, March 2016.

[174] M. Prezioso, Y. Zhong, D. Gavrilov, F. Merrikh-Bayat, B. Hoskins, G. Adam, K. Likharev, and D. Strukov, “Spiking neuromorphic networks with metal-oxide memristors,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 2016-July, May 2016, pp. 177–180.

[175] G. Pedretti, V. Milo, S. Ambrogio, R. Carboni, S. Bianchi, A. Calderoni, N. Ramaswamy, A. S. Spinelli, and D. Ielmini, “Stochastic learning in neuromorphic hardware via spike timing dependent plasticity with RRAM synapses,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, pp. 1–1, 2017.

[176] F. Ponulak and A. Kasiński, “Supervised learning in spiking neural networks with ReSuMe: Sequence learning, classification, and spike shifting,” *Neural Comput.*, vol. 22, no. 2, pp. 467–510, February 2010.

[177] M. Kozicki, M. Balakrishnan, C. Gopalan, C. Ratnakumar, and M. Mitkova, “Programmable metallization cell memory based on Ag-Ge-S and Cu-Ge-S solid electrolytes,” in *Proceedings of the IEEE Symposium Non-Volatile Memory Technology*, 2005, pp. 83–89.

[178] U. Russo, D. Kamalanathan, D. Ielmini, A. L. Lacaita, and M. N. Kozicki, “Study of Multilevel Programming in Programmable Metallization Cell (PMC) Memory,” *IEEE Transactions on Electron Devices*, vol. 56, no. 5, pp. 1040–1047, May 2009.

[179] J. R. Jameson, P. Blanchard, C. Cheng, J. Dinh, A. Gallo, V. Gopalakrishnan, C. Gopalan, B. Guichet, S. Hsu, D. Kamalanathan, D. Kim, F. Koushan, M. Kwan, K. Law, D. Lewis, Y. Ma, V. McCaffrey, S. Park,

S. Puthenthermadam, E. Runnion, J. Sanchez, J. Shields, K. Tsai, A. Tysdal, D. Wang, R. Williams, M. N. Kozicki, J. Wang, V. Gopinath, S. Hollmer, and M. Van Buskirk, "Conductive-bridge memory (CBRAM) with excellent high-temperature retention," *Technical Digest - International Electron Devices Meeting, IEDM*, pp. 738–741, 2013.

[180] J. R. Jameson, P. Blanchard, J. Dinh, N. Gonzales, V. Gopalakrishnan, B. Guichet, S. Hollmer, S. Hsu, G. Intrater, D. Kamalanathan, D. Kim, F. Koushan, M. Kwan, D. Lewis, B. Pedersen, M. Ramsbey, E. Runnion, J. Shields, K. Tsai, A. Tysdal, D. Wang, and V. Gopinath, "(invited) conductive bridging RAM (CBRAM): Then, now, and tomorrow," *ECS Transactions*, vol. 75, no. 5, pp. 41–54, 2016.

[181] N. Gonzales, J. Dinh, D. Lewis, N. Gilbert, B. Pedersen, D. Kamalanathan, J. R. Jameson, and S. Hollmer, "An ultra low-power non-volatile memory design enabled by subquantum conductive-bridge RAM," in *Proceedings of the IEEE 8th International Memory Workshop*, May 2016, pp. 1–4.

[182] S. Gaba, P. Sheridan, J. Zhou, S. Choi, and W. Lu, "Stochastic memristive devices for computing and neuromorphic applications," *Nanoscale*, vol. 5, pp. 5872–5878, 2013.

[183] G. Palma, M. Suri, D. Querlioz, E. Vianello, and B. De Salvo, "Stochastic neuron design using conductive bridge RAM," in *Proceedings of the 2013 IEEE/ACM International Symposium on Nanoscale Architectures*, ser. NANOARCH '13, 2013, pp. 95–100.

[184] M. Suri, D. Querlioz, O. Bichler, G. Palma, E. Vianello, D. Vuillaume, C. Gamrat, and B. DeSalvo, "Bio-inspired stochastic computing using binary CBRAM synapses," *IEEE Transactions on Electron Devices*, vol. 60, no. 7, pp. 2402–2409, July 2013.

[185] D. Mahalanabis, M. Sivaraj, W. Chen, S. Shah, H. J. Barnaby, M. N. Kozicki, J. B. Christen, and S. Vrudhula, "Demonstration of spike timing dependent plasticity in CBRAM devices with silicon neurons," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 2016-July, May 2016, pp. 2314–2317.

[186] S. Lim, C. Sung, H. Kim, T. Kim, J. Song, J. J. Kim, and H. Hwang, "Improved synapse device with MLC and conductance linearity using quantized conduction for neuromorphic systems," *IEEE Electron Device Letters*, vol. 39, no. 2, pp. 312–315, 2018.

[187] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Letters*, vol. 10, no. 4, pp. 1297–1301, apr 2010.

[188] E. Covi, S. Brivio, A. Serb, T. Prodromakis, M. Fanciulli, and S. Spiga, "Analog memristive synapse in spiking networks implementing unsupervised learning," *Frontiers in Neuroscience*, vol. 10, no. Oct, pp. 1–13, 2016.

[189] R. Mandapati, A. S. Borkar, V. S. Srinivasan, P. Bafna, P. Karkare, S. Lodha, and U. Ganguly, "The impact of n-p-n selector-based bipolar RRAM cross-point on array performance," *IEEE Transactions on Electron Devices*, vol. 60, no. 10, pp. 3385–3392, 2013.

[190] R. S. Shenoy, G. W. Burr, K. Virwani, B. Jackson, A. Padilla, P. Narayanan, C. T. Rettner, R. M. Shelby, D. S. Bethune, K. V. Raman, M. Brightsky, E. Joseph, P. M. Rice, T. Topuria, A. J. Kellock, B. Kurdi, and K. Gopalakrishnan, "MIEC (mixed-ionic-electronic-conduction)-based access devices for non-volatile crossbar memory arrays," *Semiconductor Science and Technology*, vol. 29, no. 10, 2014.

[191] N. Panwar, B. Rajendran, and U. Ganguly, "Arbitrary spike time dependent plasticity (STDP) in memristor by analog waveform engineering," *IEEE Electron Device Letters*, vol. 38, no. 6, pp. 740–743, June 2017.

[192] C. Zamarreño-Ramos, L. A. Camuñas-Mesa, J. A. Pérez-Carrasco, T. Masquelier, T. Serrano-Gotarredona, and B. Linares-Barranco, "On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex," *Frontiers in Neuroscience*, vol. 5, no. MAR, pp. 1–22, 2011.

[193] A. Parihar, M. Jerry, S. Datta, and A. Raychowdhury, "Stochastic IMT (insulator-metal-transition) neurons: An interplay of thermal and threshold noise at bifurcation," *Frontiers in Neuroscience*, vol. 12, p. 210, 2018.

[194] N. Papandreou, H. Pozidis, T. Mittelholzer, G. Close, M. Breitwisch, C. Lam, and E. Eleftheriou, "Drift-tolerant multilevel phase-change memory," in *Proceedings of the IEEE International Memory Workshop*, 2011, pp. 1–4.