Nazarbayev University
Department of Electrical and Electronic Engineering
Astana, Kazakhstan

SLAM: Spatiotemporal Learning with Analog Memristive Circuits for HTM

Olga Krestinskaya

Submitted in part fulfilment of the requirements for the degree of Master of Science in Electrical and Electronic Engineering of the Nazarbayev University,
Astana, Kazakhstan,

Abstract

The on-chip implementation of learning algorithms would accelerate the training of neural networks in crossbar arrays. The circuit level design and implementation of backpropagation algorithm using gradient descent operation for neural network architectures is an open problem. In addition, the learning architecture for Hierarchical Temporal Memory (HTM) has not been proposed yet.

In this work, the HTM learning process is investigated. The analog hardware implementation of backpropagation learning circuit based on memristive crossbar arrays is proposed. The learning stages in HTM are investigated. The learning circuit for HTM Temporal Memory is proposed. The integration of HTM Spatial Pooler with the backpropagation learning stage is illustrated. The study of rule-based HTM Spatial Pooler without learning is shown.

The analog backpropagation learning circuits for various memristive learning architectures, such as Deep Neural Network (DNN), Binary Neural Network (BNN), Multiple Neural Network (MNN), Hierarchical Temporal Memory (HTM) and Long-Short Term Memory (LSTM) are proposed. The implementation of additional circuit and activation functions that can be used in the construction of various biologically inspired learning architectures is shown.

The circuits are simulated in SPICE using TSMC 180nm CMOS process models, and HP memristor models. The proposed learning methods are tested for various visual data processing applications, such as face recognition and handwritten digits recognition.

Declaration

I hereby, declare that this manuscript, entitled SLAM: Spatiotemporal Learning with Analog Memristive Circuits for HTM, is the result of my own work except for quotations and citations which have been duly acknowledged. I also declare that, to the best of my knowledge and belief, it has not been previously or concurrently submitted, in whole or in part, for any other degree or diploma at Nazarbayev University or any other national or international institution.

Name: Olga Krestinskaya

Acknowledgements

It is a genuine pleasure to express my sincere gratitude, appreciation and deep feeling of thanks to all people who gave me a lot of support and help during my research and studies.

First and foremost, I'm tremendously fortunate to have the best research advisor, **Prof. Alex Pappachen James**, without whom this work would not be possible. I sincerely appreciate amount of time, dedication, effort, and energy he spends for his students every day. I'm happy to see the transformation that he brings to the students and the university. I want to express my boundless thanks to him for the motivation, wisdom, support and opportunities he provided to me. He created a perfect working environment and provided the opportunities to participate in scientific conferences, have a research internship and work as Research and Teaching Assistant. I appreciate the valuable knowledge I acquire from Prof. James, his guidance and our endless discussions about interesting research ideas and philosophies. He showed that the dedication, passion and hardworking are very important, and professional development that the person can achieve is almost limitless with the presence of these three qualities. He inspired me to become a better individual in professional and personal sense. Prof. James influenced the development my interest to do research and helped me a lot in my professional progress. He changed my way of thinking, values and beliefs. He showed how important is what we achieve in life and how much we do for the society. He showed an excellent example that it is important to care about desires, comfort and wellbeing of other people. He helped me to develop believe in myself, overcome my fear to fail, learn how to acquire the courage to make difficult and important life decisions. I highly appreciate his emotional support and wise advice. I'm proud to graduate under the supervision of Prof. James.

Secondly, I would like to express my gratitude to **Prof. Khaled Nabil Salama** from KAUST for providing an excellent internship opportunity. With the support of Prof. Salama, I gained a valuable experience of working in a clean room and knowledge about the fabrication process and analog circuit design. Also, I want to thank **Prof. Ulrich Buttner** for the time he spent on teaching us in the cleanroom and skilled that I gained. I want to thank the students and interns from KAUST, **Jose Sales Filho and Hamzah Alahmadi** for helping me with understanding of several aspects in analog circuit design.

I would like to thank our "Biomicrosystems" research group, especially my co-authors and

friends, **Aidana Irmanova, Irina Dolzhikova, Timur Ibrayev** and **Anuar Dolzhikov**, for that sleepless nights that we spent writing papers, the exciting discussion on different research topics, their support and enjoyable time. I would like to thanks my friends, especially **Marina Pen**, who tolerate that I never have enough time to meet and have fun because of the work.

The last but most important, I would like to express by boundless love and appreciation to my mother, **Lidiya Borodulina**, for her care, love, understanding and support. She is the most important person who always supported me in every step of my life journey, gave me everything what she could and sacrificed a lot for my well being. Without her efforts to give me high quality education and happy life, my professional achievements would not be possible. I sincerely appreciate the amount of time and energy she spent at work to give me everything what I have now. All the work that I do is dedicated to her.

List of publications

Published:

1) [Journal] **O. Krestinskaya**, T. Ibrayev and A.P. James, Hierarchical Temporal Memory Features with Memristor Logic Circuits for Pattern Recognition, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2017

2) [Journal] T. Ibrayev, U. Myrzakhan, **O. Krestinskaya**, A. Irmanova, A.P. James, Onchip Face Recognition System Design with Memristive Hierarchical Temporal Memory, ISTA 2017, Special Issue in Special Issue - Journal of Intelligent and Fuzzy Systems, IOS Press -SCI Impact Factor 2017: 1.261

3) [Conference] A.P. James , T. Ibrayev, **O. Krestinskaya**, Design and implication of a rule based weight sparsity module in HTM Spatial Pooler, ICECS, 2017

4) [Conference, abstract] **O Krestinskaya**, T Ibrayev, A P James, Feature extraction without learning in Spatial Pooler design of Hierarchical Temporal Memory, Oxford Circuits and Systems Conference, 2017

5) [Conference] **Olga Krestinskaya**, Khaled Nabil Salama and Alex Pappachen James, Analog Backpropagation Learning Circuits for Memristive Crossbar Neural Networks, The International Symposium on Circuits and Systems (ISCAS), 2018

Under review:

6) [Journal] **Olga Krestinskaya**, Khaled Nabil Salama and Alex Pappachen James, Analog Backpropagation Learning Circuits for Memristive Crossbar based DNN, BNN, MNN, HTM and LSTM, IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)

7) [Journal] **Olga Krestinskaya**, Irina Dolzhikova and Alex Pappachen James, Hierarchical Temporal Memory using Memristor Networks: A Survey, IEEE Transactions on Emerging Topics in Computational Intelligence (TETCI)

8) [Journal] **Olga Krestinskaya** and Alex James, Feature extraction without learning in an analog Spatial Pooler memristive-CMOS circuit design of Hierarchical Temporal Memory, Analog Integrated Circuits and Signal Processing

9) [Journal] Kamilya Smagulova, **Olga Krestinskaya** and Alex James, Memristor-based Long Short Term Memory circuit, Analog Integrated Circuits and Signal Processing

10) [Book Chapter] Alex James, Timur Ibrayev, **Olga Krestinskaya** and Irina Fedorova, Introduction to Memristive HTM Circuits, Prof. Alex James (Ed.), InTech.

11) [Conference] **Olga Krestinskaya** and Alex James, Binary weighted memristive deep neural network, 55th ACM/EDAC/IEEE Design Automation Conference (DAC), 2018

12) [Conference] Kamilya Smagulova, **Olga Krestinskaya** and Alex James, Memristive Crossbar Long Short Term Memory, 55th ACM/EDAC/IEEE Design Automation Conference (DAC), 2018

13) [Conference] Aidana Irmanova, **Olga Krestinskaya** and Alex James, Image based HTM word recognizer, 55th ACM/EDAC/IEEE Design Automation Conference (DAC), 2018

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Statement

Analog hardware implementation of the online learning, training and weight update circuit involving backpropagation methods with gradient descent for biologically inspired learning architectures, such as Hierarchical Temporal Memory (HTM), Neural Networks (NN) and Long Short Term Memory (LSTM), is an open research problem.

1.2 Hypothesis

The analog hardware implementation for the learning process with backpropagation will allow to move the biologically inspired learning architectures, such as HTM, NN and LSTM, to the sensor level. This leads to the possibility to implement full analog learning process that has not been implemented yet. The analog learning circuit will ensure faster, more efficient and more accuracy implementation of biologically inspired learning architectures.

1.3 Aims and Objectives

1.3.1 Aims

The main aims of this work are:

1. To implement the online learning and update process involving backpropagation algorithm with gradient descent on analog hardware using memristive crossbar.
2. To apply the designed algorithm to HTM and perform software and hardware testing of the algorithm and learning circuits.
3. To apply the proposed backpropagation learning circuit to various biologically inspired leaning architectures.

1.3.2 Objectives

The main objectives that should be fulfilled to achieve aim 1 are:

1. To implement and simulate the memristive crossbar arrays.
2. To develop the analog hardware design of the backpropagation leaning circuits.
3. To perform software and hardware test of the proposed approach and compare with the ideal backpropagation learning.

The objectives corresponding to aim 2 are:

1. To implement the improved analog hardware architecture of HTM.
2. To investigate the HTM performance for different HTM configurations.
3. To integrate the proposed backpropagation learning circuit into the HTM architecture.

The objectives corresponding to aim 3 are:

1. To investigate various neural networks and biologicall inspired learning architectures, such as Binary Neural Network, Deep Neural Network, Long Short Term Memory and Multiple Neural Network.

2. To integrate the proposed backpropagation analog learning circuits into these architectures.

1.4 Research gap

The full online analog learning and training circuit involving backpropagation with gradient descent for biologically inspired learning architectures has not been implemented yet. In particular, the hardware implementation of the HTM Spatial Pooler learning stage has not been performed on hardware. The complete learning circuit for biologically inspired learning architectures has not been proposed yet. However, various software-based methods, digital, mixed-signal and several analog parts of HTM have been investigated [2, 3, 4, 5, 6, 7].

1.5 Summary

The main problem that is investigated in this thesis is the implementation of the learning and training circuits on analog hardware for further use in various learning architectures based on memristive crossbar arrays, such as HTM, Neural Networks and LSTM to move learning and training process from the separate units directly to the chip.

Chapter 2

Background

2.1 Biological significance of learning

The learning architectures and algorithms in machine learning filed draw their inspiration from biology and human brain processing. The process of changing and shaping the connections between the neurons in the brain follows the synaptic plasticity rules. The strength of connection between the neurons changes with the frequency of activation. The learning architectures and algorithms emulate the structure and processing of biological learning systems [8, 9]. There are several theories how the connections between the neurons in a human brain are formed and how the learning process happens. One of the most popular theories is based on the Hebb‘s learning, which implies that if the ”neurons fire together, they wire together” [10, 11].

2.2 Biologically inspired learning algorithms and architectures

Machine learning methods can be divided into four main approaches: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning. Each learning methods can be implemented using certain learning algorithms, which include Regression algorithms,

Instance-based algorithms, Decision Tree algorithms, Bayesian algorithms, Association Rule Learning, Dimensionality Reduction algorithms, Ensemble algorithms, Artificial Neural Networks and Neuromorphic algorithms. In this work, we focus on neuromorphic algorithms and learning architectures.

2.2.1 Neuromorphic Learning Algorithms

Based on the biological inspiration, neuromorphic algorithms can be divided into three main categories:

- Cognitive systems. Cognitive systems include Hierarchical Temporal Memory and Long Short Memory networks.
- Systems that emulate synapses. These systems include Spike-Timing-Dependent Plasticity (STDP), Long Term Potentiation (LTP) and Short Term Potentiation (STP) [12].
- Systems that emulate synapses with noise and introduce stochasticity. These systems include Synaptic Sampling Machines (SSM) and Spiking SSM.

2.2.2 Learning Architectures

HTM

Hierarchical temporal memory (HTM) is a neuromorphic machine learning algorithm that emulated the performance of the human brain neocortex [13]. The main characteristics of HTM are sparsity, hierarchy and modularity. HTM consists of two main parts: the HTM Spatial Pooler (SP) and the HTM Temporal Memory (TM). The HTM SP converts the inputs to sparse binary patterns and produces Sparse Distributed Representation (SDR) of the input data. This process also refers to the encoding of the information, which is performed throughout the HTM SP [14, 15]. The HTM SP is useful for visual data processing, involving recognition, categoriza-

tion and classification problems. The HTM TM is the HTM part responsible for the learning process.

Algorithmic implementations of the HTM on software revealed great potential of this machine learning method for large variety of applications. The increase in the edge computing devices in the Internet of things era, drives the need for hardware acceleration for near sensor processing and computing. The computational considerations of the processing speed and possibility for the real-time realization pushed for transition of the originally purely software based algorithm to the hardware implementation. As a result, hardware design of HTM became a highly attractive topic for last few years that already produced promising results.

There are several hardware implementations proposed for the HTM SP, such as conventional HTM SP [16] and modified HTM SP [17]. Both architectures are based om memristive devices localed in the initialization and overlap stages. The hardware implementation of the learning stage for HTM SP has not be proposed yet. According to [18], the backpropagation algorithm can be one of the approach to update weight in the HTM SP.

Neural Networks

Neural Network is the one of the first biologically inspired learning algorithm. There is a variety of the architectures and algorithms for the neural networks. In this work, we focus on three types of neural networks: Deep Neural Networks, Binary Neural Networks and Multiple Neural Networks. Since the processing and simulation of Big Data became possible, Deep Neural Networks found their applications in various IoT fields. Deep neural networks are the networks that consist of many hidden layers and involve various activation functions. Such networks are usually applied for the classification tasks [19]. The hardware implementation of Deep Neural Networks has been investigated but the full implementation has not been proposed yet. Binary Neural Networks are neural networks relying binary weights, binary activation function, binary inputs or combined approaches. Recently, several variations of the software implementation of BNN algorithms have been reported, while the analog hardware implementation of BNN system remain an open problem [20]. Binarized Neural Network can be one of the possible

practical alternative solution to the problem of analog hardware implementation of the neural network. The other approach that will be investigated in this paper is Multiple Neural Network. Multiple approach in the Neural Networks implies that the data from different data sources, such as various sensors, is applied to separate neural networks and the output from each network is fetched into the decision network. This approach allows to simplify the complex computation processes, especially when the number of data sources is large [21]. The analog hardware implementation of Multiple Neural Networks has not been proposed yet.

Long Short Term Memory

LSTM is a cognitive architecture that is based on the sequential learning and temporal memory processing in human brain. LSTM processing relies of state change and time dependency of the processed events. The LSTM algorithm is a modification of recurrent neural network that includes the particular part of previous output that should be forgotten and a certain part of the input that should be stored in a memory [22, 23]. LSTM found its application in the contextual data processing, such as natural language processing and pattern recognition. The hardware implementation of LSTM has not been proposed yet; however recent research works include the implementation of the activation functional for LSTM [24].

2.3 Learning methods, rules and algorithms

Each learning architecture has a particular set of algorithms and learning rules that can be applied to enable the learning. Table 2.1 contains a set of learning architectures and corresponding learning rules and algorithms that can be applied there.

The machine learning methods can also be divided into online learning and offline learning methods. Online learning is faster and includes simple algorithms because complicated learning algorithms cannot be implemented. The offline learning includes complicated algorithms and most of the training processes are performed offline and the training results are stored in the

Table 2.1: Learning rules corresponding to enabling learning architectures.

Enabling Learning Architectures	**Learning rules and algorithms**
LSTM, Reccurent Neural Network	LSTM algorithm
Convolutional Neural Network	Backpropagation algorithm
HTM	Hebb‘s rules
Neuronal Boltzmann Machines	Contrastive Divergence learning rule
Spiking Neural Network	Hebbian learning, backpropagation, Theta Neuron Learning, STDP
Belief Networks	Parameter Estimation (statistical approach)

memory. The amount of storage space for online learning algorithms is significantly smaller, comparing to offline learning algorithms.

2.3.1 Learning rules

The learning rules can be divided into four main groups:

- Hebb‘s learning rules. These rules include original Hebb‘s and Anti-Hebb’s rules. Hebbian learning implies that if the synapses are activated synchronously, the strength of connection between them increases, and vice versa. The anti-Hebbian learning implies that the strength of connection decreases the efficiency of a presynaptic neuron to elicit the response of a postsynaptic neuron [25, 26].

- Error-based learning rules. These rules are derived from Hebb‘s rules, such as simple perceptron. Perceptron learning is a supervised learning with reinforcement. The main principle is to minimize the error adjusting the weights in a series of small updates. The other learning method, which is similar to the perceptron learning is delta rule, which is a foundation of backpropagation algorithm. It is based on error calculation. The other rule is a Manhattan rule, which is different from the delta rule by the binary quantization process. The update process in this rule has a constant update value but different signs. The other rule is Oja learning rule, which is a Hebb‘s rule with normalization [3, 27].

- Rate and gradient based learning rules. These rules are more complex and include backpropagation process. These rules are based on gradient of change of the error. The

backpropagation algorithm with gradient descent is one of the widely used algorithms. Bienenstock-Cooper-Munro (BCM) rule is the other rate-based learning rule, which depends on the rate of pre- and post-synaptic spikes. It is based on the LTD method [28, 29].

- Time-based learning rules. These rules include STDP rules. STDP rule is based on the delay between the presynaptic and postsynaptic firing [30].

2.3.2 Recent works and studies

There have been several attempts to implement learning architectures on hardware using memristive crossbar circuits. These include conventional neural networks with backpropagation algorithm, HTM with Hebb's learning and gradient descent rules and spiking neural networks with STDP.

The implementation of basic Hebb's learning rules has been performed with a single synapse and on a crossbar for both spike and pulse based approaches. The perceptron learning rules has been implemented for both online and offline pattern classification methods. The Manhattan and Oja rules has been implemented on a crossbar. However, the analog learning and training process has not been shown yet. In addition, the attempts to implement the forward propagation for the gradient descent method has already been done. However, the backpropagation and update process has not been implemented in analog hardware yet [2, 4, 5, 6, 7].

2.4 Summary

There is a number of studies concerning the learning rules, algorithms and architectures, inspired form biological concepts, such as human brain processing and neuron learning. Even though, several studies have been performed to implements software based, mixed signal, digital and analog parts of learning circuits, the full analog hardware implementation for the learning and training of biologically inspired architectures have not been proposed yet.

Chapter 3

Backpropagation learning architecture

3.1 Introduction

The developments of Internet Of Things (IoT) applications led to the demand to develop the near edge computation architectures. The near edge computing leads to near sensor computations that support a non-Neumann computing architectures such as neuromorphic chips having integration of memory and processing units. In such architectures, learning remains as a major task that determines its overall effectiveness and use.

Neuromorphic computing draws inspiration from the neuronal networks and neurons in the human brain, where the signal are processed in a complex analog domain [31, 32]. Several neuron models and neural networks has been proposed in the recent years. The weights of synaptic connections between the neurons can be implemented in hardware with the help of memristive crossbar arrays [33], which require the learning algorithms and training units. The learning algorithms, such as gradient descent [29], are used for the training of the weights in the memristive crossbar learning architectures. However, the learning process [34, 35] in crossbars has not been not fully implemented and remain as an open problem. The implementation of analog learning algorithm opens an opportunity to create a fully hardware-based learning architecture. This would transfer the learning/training algorithms from the separate software and FPGA-based units to on-chip analog learning circuits, which can simplify and speed up

the learning and training process.

In this Chapter, the analog hardware implementation of a backpropagation learning circuit for the neural networks based on memristive crossbar array is proposed. The full architecture of the learning process and various activation functions is provided.

3.2 Background

3.2.1 Backpropagation with gradient descent

In this work, we focus on the analog implementation of backpropagation algorithm based on the gradient descent calculation [36] for a three layer artificial neural network [37] and the extension of the algorithm and its application for different learning architectures. The algorithm consists of four steps: forward propagation, backpropagation to the output layer, backpropagation to the hidden layer and weight update process.

In the forward propagation step, the dot product of the input matrix X and the weighted connections between input layer and hidden layer w_{12} is calculated and passed through the sigmoid activation function: $Y_h = \sigma(X \cdot w_{12})$, where Y_h is an output of the hidden layer. The activation function can change depending on the application of the algorithm. Depending on the number of hidden layers the forward propagation through the hidden layer is repeated. The propagation through the output layer is performed in the same manner. The output of the three layer network Y_o is calculated as $Y_o = \sigma(Y_h \cdot w_{23})$, where w_{23} is the matrix representing the weighted connections between the hidden and output layers.

The backpropagation algorithm uses the cost function as the calculation of derivative of error with respect to the change in weight. The calculation of the error is shown in Eq.3.1, where N is a number of neurons in the layer, y_{target} is an ideal output and y_{real} is the obtained output

after the forward propagation.

$$E = \frac{1}{2}\sum_{i=1}^{N}(y_{target} - y_{real})^2 \tag{3.1}$$

The calculation of the derivative of error is different for the hidden layer and for the output layer. The backpropagation through the output layer relies on the difference between the ideal output and the output obtained from the forward propagation through the network. Eq. 3.2 shows the calculation of the derivative of output layer error E_o, where δ denotes the rate of change of the error with respect to the weight w_{23}. The error for the output layer e is calculated as a difference between expected neural network output Y and real output of the network Y_o: $e = Y - Y_o$. The derivative of the sigmoid function is the following: $\frac{\partial Y_o}{\partial w_{23}} = Y_o(1 - Y_o)$.

$$\frac{\partial E_o}{\partial w_{23}} = Y_h \cdot \delta_2 = Y_h \cdot (e \odot \frac{\partial Y_o}{\partial w_{23}}) \tag{3.2}$$

The backpropagation through the hidden layer is based on the derivative of error obtained from the previous layer calculations. The calculation of the derivative of error for the hidden layer is shown in Eq.3.3, where X' is an inverted input matrix and e_h is the error of the hidden layer. The error e_h is calculated propagating back δ_2 as following: $e_h = \delta_2 \cdot w'_{23}$. And derivative of the hidden layer output E_h is the same as in the output layer: $\frac{\partial Y_h}{\partial w_{12}} = Y_h(1 - Y_h)$.

$$\frac{\partial E_h}{\partial w_{12}} = X' \cdot \delta_1 = X' \cdot (e_h \odot \frac{\partial Y_h}{\partial w_{12}}) \tag{3.3}$$

The final stage of the propagation algorithm is weight update. In the update stage, the weight matrices is performed using Eq.3.4 and Eq.3.5, where η is the learning rate responsible for the speed of convergence. The optimized learning rate depends on the type and number of inputs and number of the neurons in the hidden layers.

$$\Delta w_{23} = \frac{\partial Y_o}{\partial w_{23}} \times \eta \tag{3.4}$$

$$\Delta w_{12} = \frac{\partial Y_h}{\partial w_{12}} \times \eta \tag{3.5}$$

The weight matrices are updated considering the calculated change in weight: $w_{23\_new} = w_{23} + \Delta w_{23}$ and $w_{12\_new} = w_{21} + \Delta w_{12}$.

3.3 Analog Backpropagation Learning Circuits

3.3.1 Overall architecture

The proposed hardware implementation of the learning algorithm is illustrated on Fig. 4.23(a). Depending on the application requirements and limitations of the memristive devices, the inputs to the system can be either binary or analog. For example, for the HTM applications the inputs are analog [17], whereas for binary neural network, inputs can be binary [20]. The outputs of the neural network can also be binary or analog, depending on the activation function. For the majority of the applications, binary outputs are required. Memristive crossbar arrays emulate the set of synapses between the neurons in the neural network layers. The synapses can also be binary or analog depending on the applications and memristor limitations. Most of the memristors can store only R_{ON} and R_{OFF} values, which means that the values of the synapses are binary. However, the implementation of the analog weights is also possible using 16-level *Ge2Sb2Te5* (GST) memristors [38].

The example shown in Fig. 4.23(a) demonstrates the basic three layer neural network with the proposed backpropagation architecture. The neural network has three input neurons, two output neurons and five neurons in a hidden layer. The CROSSBAR 1 corresponds to the set of synaptic connections between the input layer and the hidden layer, and the Crossbar 2 represents the synapses connecting the hidden layer to the output layer. The three input signals are shown as V_{in1}, V_{in2} and V_{in3}. The inputs are fetched to the rows of the CROSSBAR 1. Each memristor in a single column of the crossbar corresponds to the connections of all inputs to a particular single output. The crossbar performs dot product multiplication of inputs and

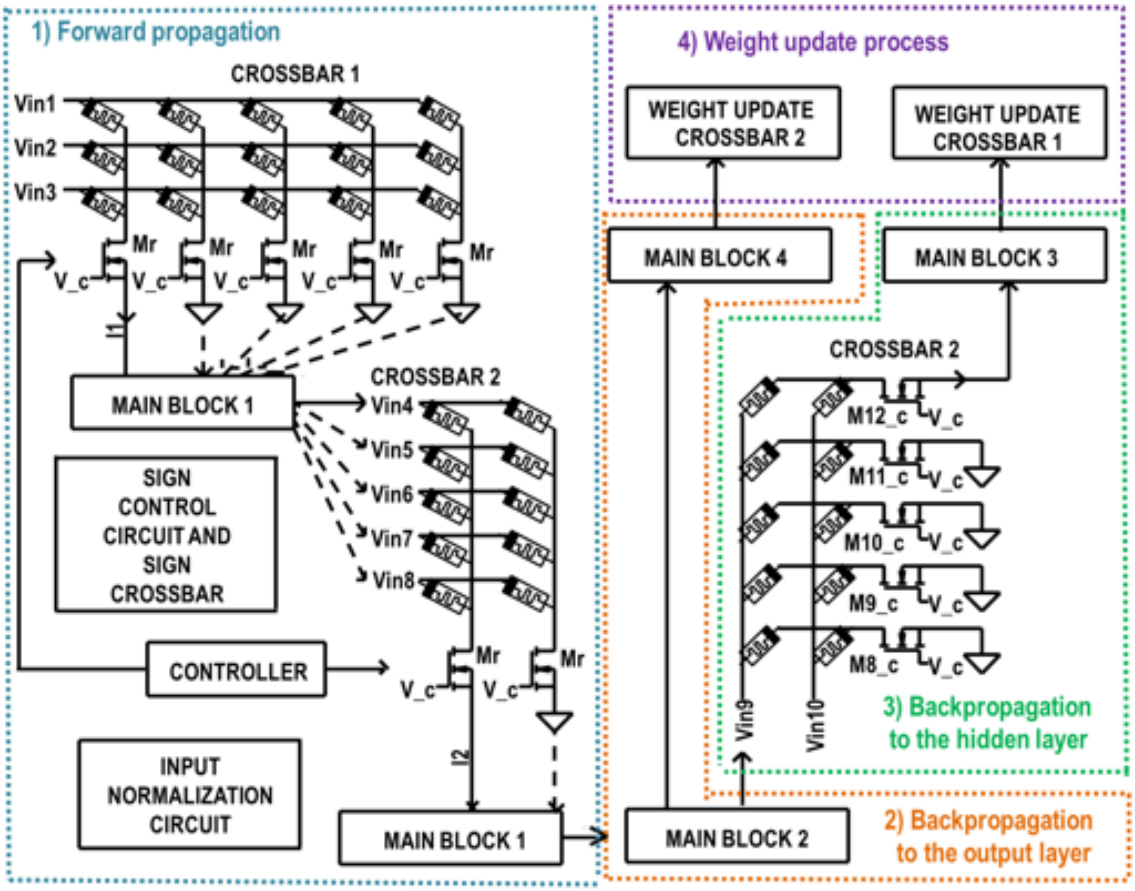


Figure 3.1: Overall architecture of the proposed analog backpropagation learning circuits for memristive crossbar neural networks. In the forward propagation process, MAIN BLOCK 1 (MB1) is involved. The backpropagation through the output layer is performed by MAIN BLOCK 2 (MB2) and MAIN BLOCK 4 (MB4). The backpropagation through the hidden layer is performed by MAIN BLOCK 3 (MB3). The weight update process of the output layer and the hidden layer is performed by MB4 and MB3, respectively.

the weights of a single column. The output of the multiplication for feed-forward propagation is read by the NMOS read transistor M_r connected to a crossbar column. In this work, we investigate the approach, when the output of the crossbar is represented a current flowing through the read transistor. However, the configuration can be change to the voltage output from the crossbar column, if it is required. The outputs from the crossbar columns are read sequentially one at a time to avoid the interference with the currents from the other columns. In addition, the read transistors are used as switches to switch on and off the columns and maintain the order of the reading sequence. The controller is used to control the sequence of the pulses, timing and even the sign of the signals in particular cases. As the negative resistance is impossible to achieve in the memristive device, the negative weights are implemented either input controller, which changes the input sign according to the sign of the synapse weight, or by the sign control circuit and additional crossbar that stores the sign of the sign of the weights.

MAIN BLOCK 1 (MB1) in Fig. 4.23(a) performs the forward propagation and is responsible for the calculation of the activation function. Depending on the application, the activation function can include the sigmoid, derivative of the sigmoid, tangent, derivative of the tangent, approximate sigmoid and approximate tangent functions. To implement the conventional back-propagation algorithm with gradient descent, MB1 refers to the calculation of the sigmoid and the sigmoid derivative functions. The outputs of MB1 are used as the inputs to CROSSBAR 2. The output currents from the second crossbar are fetched to the second MB1 and the final output of the feed-forward propagation are obtained from the second MB1. The final outputs depend on the activation function. For the original backpropagation with gradient descent, the final outputs are binary.

After the forward propagation process, the backpropagation process is implemented. The controller reconnects the second memristive crossbar. The rows of the CROSSBAR 2 are connected to a different set of read transistors and the columns are connected to the set of inputs from the MAIN BLOCK 2 (MB2). If ensures that the propagation is performed in the opposite direction. If the neural network contains more that three layers, all crossbars except the first crossbar corresponding to the synaptic weights between the input and the first hidden layer, are reconnected to perform backpropagation operation. The backpropagation process through

the output layer is implemented using MB2 and MAIN BLOCK 4 (MB4). While the backpropagation through the hidden layer corresponds to the MAIN BLOCK 3 (MB3). Each main block contains the storage unit, where the intermediate parameters and final output values are preserved.

The final stage in the backpropagation process in the weight update stage, where the values of the memristors are updated based on the particular rules. The weight update process is implemented by applying the voltage pulse of a particular duration and amplitude across each memristor. The update pulse depends on the required change on the weights, calculated gradient of error and the memristor type and technology. The update process is applied to all the crossbars in the network. The additional update circuitry based on the switching transistors is used.

3.3.2 Circuit level implementation of the backpropagation algorithm

In this work, the analog circuits for the proposed backpropagation implementation are designed for 180nm CMOS process. The memristor model used in the crossbar simulations is Biolek's modified S-model for HP TiO_2 memristor with the threshold voltage V_{th} of 1V [39]. The control signal that is applied to the gate of the read transistor is $V_s = 1V$ (Fig.4.23(a)). To disconnect the transistor and make the floating node at column of the crossbar, the signal $V_s = 0V$ is applied to the read transistor gate. When the transistor is disconnected, it prevents the current from flowing through it. The crossbar output current from the columns is adjusted to vary within the range from $-70\mu A$ to $70\mu A$ because the following sigmoid circuit implementation is adjusted to perform in this range. However, the current range and the range for the activation function can be adjusted considering the application and the type of memristor that is applied. The control signal V_s and the W/L ratio of the transistor can be controlled to adjust the output current range and the linearity of the read transistor M_r. The transistor M_r should be kept in linear region to obtain an accurate dot product multiplication output. The other method to improve the linearity and keep the transistor in a linear region is to reduce the input voltage level by the external circuit.

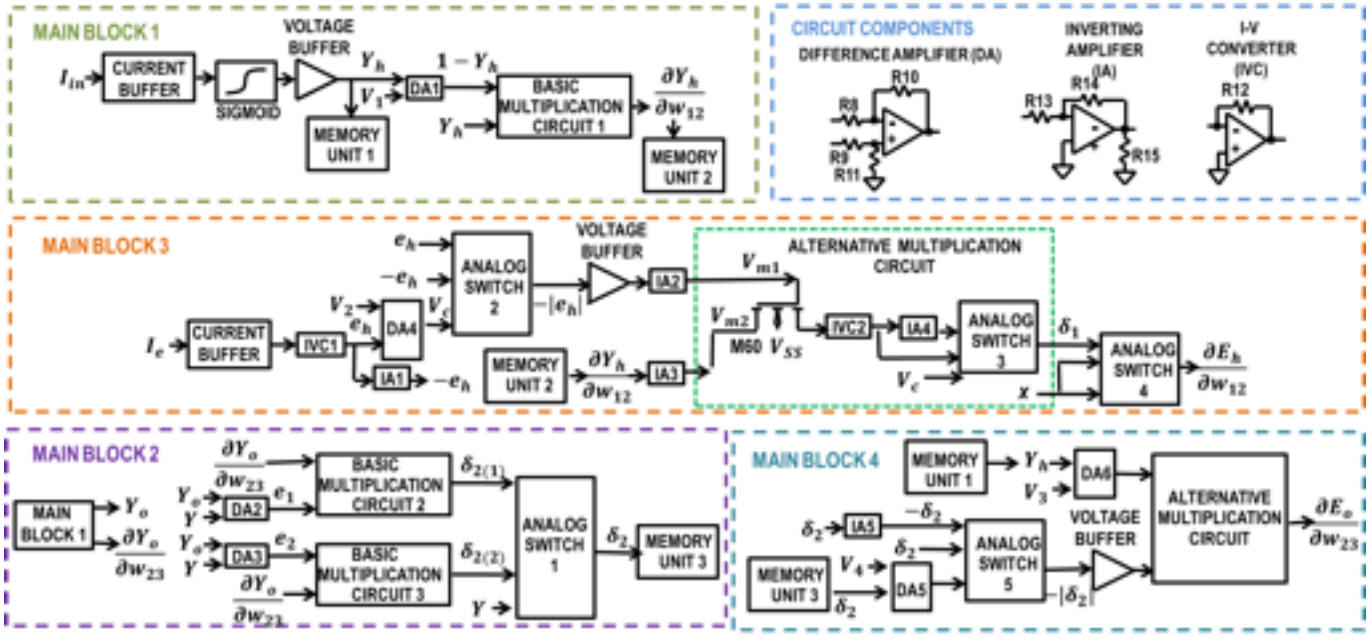


Figure 3.2: The circuit level architecture of the proposed backpropagation implementation. The separate implementation of the MB1, MB2, MB3 and MB4 is illustrated. In addition, the involved circuit components, such as DA, IA and IVC, are shown. The circuit parameters for the proposed components are mentioned in Table 3.1

Figure 4.23(b) illustrates the circuit level details of the main blocks of the proposed backpropagation architecture. MB1 performs the forward propagation for the conventional backpropagation architecture. The parameters of all the elements are shown in Table 3.1. As the design of the circuits is adjusted to TSMC 180nm CMOS technology, the V_{DD} and V_{SS} values in all circuits are $1.8V$ and $-1.8V$. MB1 consists of the current buffer, sigmoid function implementation, voltage buffer, memory units, difference amplifier and basic multiplication circuit. The current buffer is applied to eliminate the current from the crossbar transistor during the reading process and avoid the impact of the MB1 circuit on the output current from a crossbar column. The complete current buffer circuit is illustrated in Fig.4.23(c). The current buffer circuit consists of 14 transistors and is based on the current mirror principle. As the input current I_{in} flowing through the reading transistor to the current buffer can be both positive or negative due to the negative weights of the synapses, the current buffer supports both directions of the current. To ensure that the value of the current is not corrupted by the further load of the MB1 circuit, the virtual ground node is located between $M4$ and $M6$. As the values of all the transistors of the circuit are the same, the voltage between the transistors $M4$ and $M6$ equals

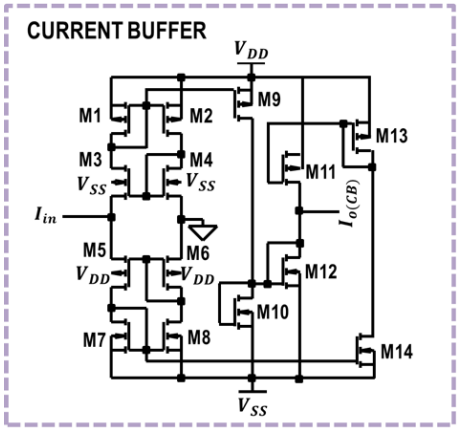


Figure 3.3: The current buffer circuit which is connected to the read transistor M_r in Fig.4.23(a). The circuit is used in MB1 and MB3 to eliminate the loading effect of the activation function to the performance of the crossbar.

to the voltage between $M3$ and $M5$, which ensures that the current from the read transistor circuit flow to the ground. If the input current to the current buffer I_{in} is positive, the current flows through the transistors $M5$ and $M7$ and is mirrored by $M6$ and $M8$. If I_{in} is negative, the transistors $M1$ and $M3$ and mirrored by $M2$ and $M4$. The output current is obtained from the node $I_{o(CB)}$, where the currents from the lower and upper parts of the current mirror are summed up to ensure that the output current is the same as the input current for both positive and negative ranges.

After the current buffer in MB1, the input current flows to the sigmoid circuit. The implementation and the transistor parameters of the sigmoid circuit are selected based on the sigmoid shown in [1]. The implementation of the sigmoid function is shown in Fig.4.23(d). The sigmoid is designed to read the input current and produce the output in terms of voltage. The current $I_{in(s)}$ is the input current to the sigmoid, and $V_{o(s)}$ is the output voltage of the sigmoid, which is normalized between 0 and V_{DD}. The node for the input current and output voltage in the circuit in Fig.4.23(d) is the same; therefore, the sigmoid is sensitive to any connected load and requires the voltage buffer to connect it to the other circuits.

The voltage buffer is based on the operational amplifier (OpAmp) circuit shown in Fig.4.23(e). The two stage OpAmp in Fig.4.23(e) is used in the most of the circuit components. Based on the application of the OpAmp, the amplifier output is read either from V_{op1} or from V_{op2}. The voltage buffer eliminates the effect of loading effect and shifts the sigmoid signal and normalize

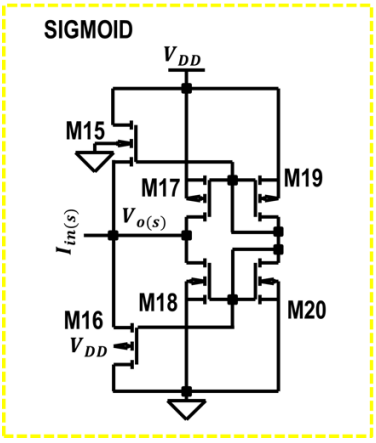


Figure 3.4: Sigmoid activation function used in MB1 inspired from [1].

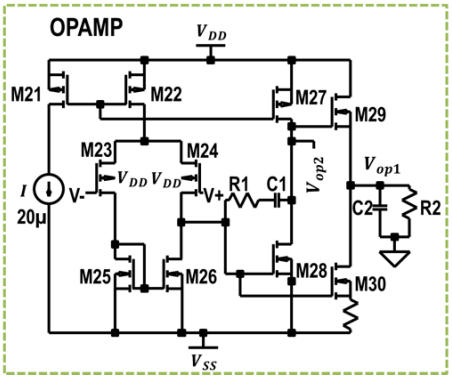


Figure 3.5: Two stage OpAmp design used for all OpAmp based components in the proposed analog memristive learning circuit.

it between $0V$ and $1V$. To implement the voltage buffer, $V-$ of the OpAmp is connected to the V_{op2}, $V+$ is connected to the sigmoid output $V_{o(s)}$ and final Y_h is read from V_{op1}. The output signal Y_h is stored in the MEMORY UNIT 1. The storage unit can be implemented using analog memristive circuits, where the voltage values are stored in the memristive units.

The other part of the MB1 implements the derivative of the sigmoid function $\frac{\partial Y_o}{\partial w_{23}}$. As a derivative of a sigmoid can be achieved by multiplying the sigmoid output Y_h by $1 - Y_h$, the voltage buffer output Y_h is fetched into the difference amplifier (DA1) and connected to $V+$ through R_8. The voltage $V_1 = 1V$ and is connected $V-$ node through the resistor R_9. As the output of the difference amplifier is $V_{op1} = (R_{10}/R_8)(V_1 - Y_h)$ and $R_{10} = R_8$, the DA1 output is $1 - Y_h$, which is read from the node V_{op1}. The multiplication of $1 - Y_h$ and Y_h is performed using Hilbert multiplier circuit refering to BASIC MULTIPLICATION CIRCUIT 1 shown in Fig.4.22(c). The multiplication relies on the currents I_1 and I_2 flowing through

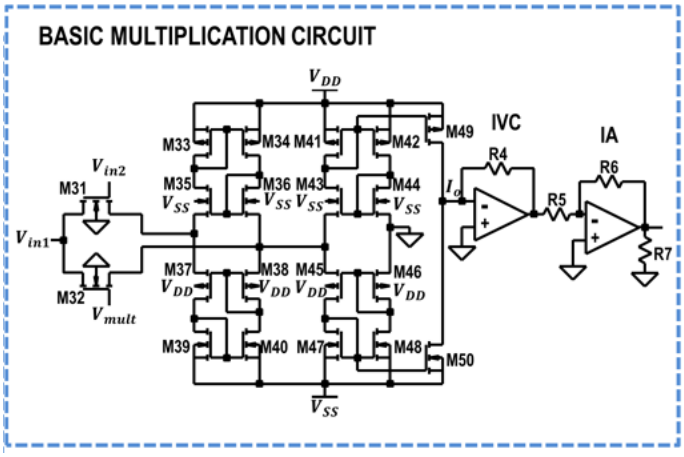


Figure 3.6: Multiplication circuit based on the Hilbert multiplier principle. The circuit is used in MB1 and MB2.

the transistors M_{31} and M_{32}, respectively. The output current I_o equals to the difference ΔI between the currents I_1 and I_2 corresponding to the Eq.3.6, where K_n is proportional to W/L of the transistors and ΔI is derived from the subtraction circuit with the parameters of the transistors M_{33} - M_{50}. The voltage V_{mult} equals to $0V$.

$$\Delta I = I_1 - I_2 = \frac{K_n}{2}(V_{in2} - V_{mult}) \times V_{in1} = \frac{K_n}{2} \times V_{in2} \times V_{in1}, \tag{3.6}$$

In the multiplication circuit, the first part of the circuit with transistors $M33$-$M40$ calculates the difference in currents ΔI from the node between $M36$ and $M38$. The other part of the circuit with $M41$-$M50$ acts as a current buffer and outputs the current $I_o = \Delta I$. The output current I_o is converted to voltage by the current to voltage converted (IVC). IVC is based on the OpAmp (Fig.4.22(c)) with the output read from V_{op1}. The output voltage from the IVC equals to $-I_o R_4$. The IVC is followed by the inverting amplifier (IA), which inverts the negative voltage from IVC and the outputs the voltage $-(R_6/R_5) \times (-I_o R_4)$. As $R_6 = R_5$, the output voltage from the IA equals to $I_o R_4$. The output form the basic multiplication circuit $\frac{\partial Y_h}{\partial w_{12}}$ is stored in the MEMORY UNIT 2. The output current from all the read transistors in CROSSBAR 1 are read in a sequence and the output voltages Y_h and $\frac{\partial Y_h}{\partial w_{12}}$. Then, the same is repeated with CROSSBAR 2 and Y_o and $\frac{\partial Y_o}{\partial w_{23}}$ from the the second MB1 are stored in the memory units.

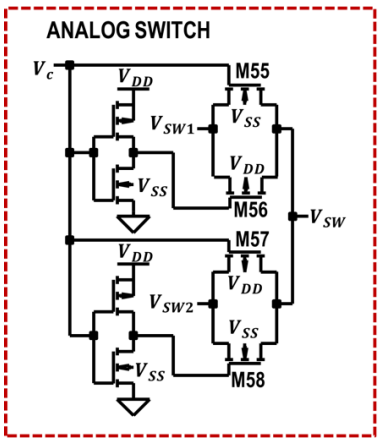


Figure 3.7: Analog switch design used in MB2, MB3 and MB4.

The backpropagation process to the output layer starts from MB2 and is completed by MB4. MB2 output the parameter δ_2 for each neuron that is calculated in a sequential manner connecting MB2 circuit to CROSSBAR 2 transistors one at a time. The output layer error e is calculated by the difference amplifier as $Y - Y_o$. The ideal output Y is binary (either $0V$ or $1V$) and the output of the sigmoid is always positive; therefore, the error δ_2 can be both positive and for negative. To improve the linearity of the multiplier and normalize the multiplier output, which is different for positive and negative inputs in terms of the output amplitude, the error calculation is divided into two parts, where the errors e_1 and e_2 correspond to the positive and negative cases of the output error e. The subtraction $Y - Y_o$ is performed by DA2 and DA3. Next, the error e is multiplied by the sigmoid derivative $\frac{\partial Y_o}{\partial w_{23}}$, which is performed for positive and negative error e. The multiplication $e_1 \times \frac{\partial Y_o}{\partial w_{23}}$ is performed by BASIC MULTIPLICATION CIRCUIT 2 with the output $\delta_{2(1)}$ corresponding to the positive error scenario. Ans the multiplication $e_2 \times \frac{\partial Y_o}{\partial w_{23}}$ is done by BASIC MULTIPLICATION CIRCUIT 3 with the output $\delta_{2(2)}$ refering to the calculations of δ_2 for the negative error case. To select positive and negative output δ_2, $\delta_2(1)$ and $\delta_2(2)$ are fetched to ANALOG SWITCH 1 shown in Fig,4.22(d). The outputs $\delta_2(1)$ and $\delta_2(2)$ are connected to V_{SW2} and V_{SW1}, respectively. The switch is controlled by the ideal output signal Y which is fetched into V_c node. If the error is negative ($Y = 0V$), $V_{SW} = V_{SW2}$, and $V_{SW} = V_{SW1}$ for positive case ($Y = 1V$). The output of analog switch provides the value δ_2, which is stored in MEMORY UNIT 3.

The backpropagation to the hidden layer is performed by MB3. First the δ_2 is propagated back

through the CROSSBAR 2 and the hidden layer error in terms of current I_e is calculated as a dot product of the δ_2 and weights w_{23}. The current I_e flowing through the read transistor from CROSSBAR 2 is fetched into the current buffer to eliminate the loading effect to the output current. The output current from CURRENT BUFFER in MB3 (Fig.4.23(b)) is converted into voltage e_h using IVC 1. The current I_e has the range from $-70\mu A$ to $70\mu A$ and is converted to the range of voltages e_h from $-0.25V$ to $0.25V$. In addition, the e_h value is inverted to obtain $-e_h$ using the IA1 for the further multiplication process. The $\frac{\partial Y_h}{\partial W_{12}}$ is calculated using ALTERNATIVE MULTIPLICATION CIRCUIT, ANALOG SWITCH 2, IA4 and ANALOG SWITCH 3.

As the ALTERNATIVE MULTIPLICATION CIRCUIT consists of IVC2, IA4 and ANALOG SWITCH 3 and is based on a single NMOS transistor $M60$, it is sensitive to the sign of voltages values V_{m1} and V_{m2}. The input to V_{m1} of the transistor M_{60} is kept always positive. And the second input V_{m2} of the multiplication transistor M_{60} is kept always negative to ensure linear multiplication of V_{m1} and V_{m2}. The second input V_{m2} is kept always negative by connecting it to always positive $\frac{\partial Y_h}{\partial W_{12}}$ read from MEMORY UNIT 2 through the IA3. IA3 inverts the voltage $\frac{\partial Y_h}{\partial W_{12}}$ and outputs $-\frac{\partial Y_h}{\partial W_{12}}$. To keep V_{m1} always negative, e_h is converted to always negative value $-|e_h|$ using ANALOG SWITCH 2. This allows the multiplier to perform equal multiplication for positive and negative e_h. ANALOG SWITCH 2 is controlled by V_c which is obtained by shifting the e_h signal by the DA4. This voltage shifts the e_h signal, which is initially in the range from $-0.25V$ to $0.25V$, to V_c with the range from $0.6V$ to $1V$ to control the inverter in analog switch driven by $V_{DD} = 1.8V$ and having the switching threshold of about $0.82V$. The ANALOG SWITCH 2 parameters are the same of ANALOG SWITCH 1. Based on the control voltage V_c, ANALOG SWITCH 2 selects either input e_h or $-e_h$. As V_c equals to the shifted e_h, if e_h is negative analog switch selects e_h, otherwise it selects $-e_h$. To eliminate the effect of ANALOG SWITCH 2 to the multiplication transistor $M60$, the voltage $-|e_h|$ is supplied to the voltage buffer. To convert $-|e_h|$ to always positive, it is fetched to IA2. During the multiplication process, the output current from M_{60}, which is proportional to $V_{m1} \times V_{m2}$, is converted to voltage using IVC2. The output from IVC2 is fetched into the ANALOG SWITCH 3 and the IA4. IV4 inverts the multiplication output to use in the

ANALOG SWITCH 3. ANALOG SWITCH 3 controls the sign of δ_1. The same control signal V_c as in ANALOG SWITCH 2 is used to control ANALOG SWITCH 3. If the sign of e_h was inverted by ANALOG SWITCH 2, the sign of $\delta_1 = e_h \times \frac{\partial Y_h}{\partial W_{12}}$ is inverted back. Based on the control voltage V_c obtained form the DA4, the output of the ANALOG SWITCH 3 δ_1 provides either the inverted output from IA4 (if the initial input was inverted to get the negative value of $-|e_h|$) or not inverted output from IVC2.

The final multiplication by the input signal X' by δ_1 is performed by ANALOG SWITCH 4. The proposed method of multiplication is valid only if the initial inputs to CROSSBAR 1 are binary. The multiplication by $X = 0V$, produces $\frac{\partial E_h}{\partial w_{12}} = 0$, while the multiplication by $X = 1V$ gives $\frac{\partial E_h}{\partial w_{12}} = \delta_1$. If the inputs to the system are analog, the ANALOG SWITCH 4 should be replaced by another multiplication circuit.

The backpropagation through the output layer is finished by MB4, where $\frac{\partial E_o}{\partial w_{23}}$ is calculated. The setup of the circuits is similar to the previous stage, with the output of the alternative multiplication circuit is $\frac{\partial E_h}{\partial w_{12}}$. The multiplication principle is similar to MB3 and performed only for negative V_{m2} and positive V_{m1} in the alternative multiplication circuit. Before fetching into V_{m1}, the signal Y_h from MEMORY UNIT 1 is shifted by $0.25V$ to improve the linearity of the multiplier using the DA6. The always negative input to V_{m2} is achieved using the same principle as in MB3. The signal δ_2 from MEMORY UNIT 3 having a range from $-0.2V$ to $0.2V$ is converted to always negative $-|\delta_2|$ input to V_{m2} for the multiplier performance improvement using ANALOG SWITCH 5, the DA5 and IA5. The DA5 is the same as DA4 in MB3 and produces the control signal for ANALOG SWITCH 5. The IA5 inverts the signal δ_2 and is the same as IA1 in MB3. The ANALOG SWITCH 5 provides always negative signal $-|\delta_2|$, which is fetched to V_{m2} through the voltage buffer similar to the voltage buffer in MB3. This eliminates the effect of the analog switch to the transistor current in the alternative multiplication circuit. The alternative multiplication circuit is the same as in MB3, with the difference in the R_{12} and C_1 in the IVC2 and R_{14} in IA4. The final analog switch in the alternative multiplication circuit is controlled by the signal from DA5; if the δ_2 value was inverted initially, it inverts it again and vice versa. The output of the alternative multiplication circuit is $\frac{\partial E_o}{\partial w_{23}}$.

The outputs $\frac{\partial E_h}{\partial w_{12}}$ from MB4 and $\frac{\partial E_o}{\partial W_{23}}$ from MB3 are used in the weight update process of the CROSSBAR 2 and CROSSBAR 1, respectively. Based on the amplitude and sign of these parameters, the update signal with particular duration and amplitude is applied across each memristor and the resistances of the memristors, corresponding to particular values of the synapses in analog neural network, are updated.

Table 3.1 represents the parameters for all circuit elements including into the main backpropagation architecture from Section **??**.

3.4 Simulation results

The circuit simulations were performed in SPICE, and the verification of the backpropagation algorithm is done in Matlab. The results for the simulation are shown in Fig.3.8 and Fig. 3.9. Fig.3.8 illustrates the outputs for the main functions in each block comparing them to the ideal outputs. Fig.3.8 (a) shows the output Y_h change with the input current variation from the crossbar read transistor M_r. The ideal sigmoid on the graph corresponds to a common sigmoid function represented by Eq.3.7, where $f(x)$ is the sigmoid output and x is an input variable.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.7}$$

The generalized sigmoid from Fig.3.8 (a) represents the approximation of the obtained analog output refering to Eq.3.8, where $f(x)$ is the sigmoid output and x is an input variable.

$$f(x) = 0.991 - \frac{1.0058}{(1 + e^{2.8612(x+0.00196)})^{0.7644}} \tag{3.8}$$

The real output shows the sigmoid obtained from the circuit simulation in SPICE. The implementation of the backpropagation algorithm was verified in Matlab of both ideal and generalized sigmoid activation functions, and the achieved performance of these two approaches is similar. Therefore, the ideal sigmoid and be replaces with the generalized sigmoid function.

Table 3.1: Circuit parameters for the proposed backpropagation learning analog architecture.

Component	Transistors (W/L ratio)	Component	Resistors	Capacitors
CROSSBAR 1 and CROSSBAR 2	$M_r = 0.18\mu/0.36\mu$	-	-	
CURRENT BUFFER (MB1 and MB3)	$M1 = M2 = M3 = M4 = M5 = M6 = M7 = M8 = M9 = M10 = M11 = M12 = M13 = M14 = 2\mu/5\mu$	-	-	
SIGMOID	$M15 = 0.486\mu/0.18\mu$, $M16 = 0.954\mu/0.18\mu$, $M17 = M19 = 2.72\mu/0.18\mu$ and $M18 = M20 = 0.234\mu/0.18\mu$	-	-	
OPAMP	$M21 = 3\mu/0.18\mu$, $M22 = 30\mu/0.18\mu$, $M23 = M24 = 36\mu/0.18\mu$, $M25 = M26 = 6\mu/0.18\mu$, $M27 = 45\mu/0.18\mu$, $M28 = M29 = M30 = 18\mu/0.18\mu$	VOLTAGE BUFFER, MB1:	$R_1 = 10k\Omega$, $R_2 = 3k\Omega$ and $R_3 = 300\Omega$;	$C_1 = C_2 = 1pF$
		VOLTAGE BUFFER, MB3 :	$R_1 = 10k\Omega$, $R_2 = 1k\Omega$, R_3 is removed	$C_1 = 10pF$,$C_2 = 1pF$
		DA1, BASIC MULTIPLICATION CIRCUIT 2 (IVC,IA, MB2), BASIC MULTIPLICATION CIRCUIT 3 (IA, IVC), IA4:	$R_1 = 10k\Omega$, $R_2 = 1k\Omega$, R_3 is removed	$C_1 = C_2 = 1pF$
		BASIC MULTIPLICATION CIRCUIT 1 (IVC):	$R_1 = 10k\Omega$, $R_2 = 1k\Omega$, R_3 is removed	$C_1 = 1pF$ $C_2 = 10pF$
		BASIC MULTIPLICATION CIRCUIT 1 (IA):	$R_1 = 10k\Omega$, $R_2 = 1k\Omega$, R_3 is removed	$C_1 = C_2 = 1pF$
		DA2,DA3:	$R_1 = 10k\Omega$, R_2 and R_3 are removed	$C_1 = 1pF$, C_2 is removed
		IVC1, DA4, IA1, IVC2 (MB3 and MB4), IA5:	$R_1 = 10k\Omega$, R_2 and R_3 are removed	$C_1 = 1p$ $C_2 = 10pF$
		IA3, IVC2 (MB4):	$R_1 = 10k\Omega$, R_2 and R_3 are removed	$C_1 = C_2 = 10pF$
		IA2:	$R_1 = R_2 = 10k\Omega$ and R_3 is removed	$C_1 = C_2 = 1p$
		IA4, DA5, DA6:	$R_1 = 10k\Omega$, R_2 and R_3 is removed	$C_1 = C_2 = 1p$
		VOLTAGE BUFFER, MB4:	$R_1 = 10k\Omega$, $R_2 = 1k\Omega$ and R_3 is removed	$C_1 = C_2 = 10p$
BASIC MULTIPLICATION CIRCUIT	$M_{31} = M_{32} = 0.36\mu/0.18\mu$, $M_{33} = M_{34} = M_{35} = M_{36} = M_{37} = M_{38} = M_{39} = M_{40} = M_{41} = M_{42} = M_{43} = M_{44} = M_{45} = M_{46} = M_{47} = M_{48} = M_{49} = M_{50} = 10\mu/2\mu$.	BASIC MULTIPLICATION CIRCUIT 1	$R_4 = 5k\Omega$, $R_5 = R_6 = 1k\Omega$, $R_7 = 100\Omega$	-
		BASIC MULTIPLICATION CIRCUIT 2:	$R_4 = 4.5k\Omega$, $R_5 = R_6 = 1k\Omega$, $R_7 = 100\Omega$	-
		BASIC MULTIPLICATION CIRCUIT 3:	$R_4 = 6.5k\Omega$, $R_5 = R_6 = 1k\Omega$, $R_7 = 100\Omega$	-
ANALOG SWITCH 1,2,3,4,5	$M_{51} = M_{53} = M_{56} = M_{57} = 0.72\mu/0.18\mu$ and $M_{52} = M_{54} = M_{55} = M_{58} = 0.36\mu/0.18\mu$	-	-	-
DIFFERENCE AMPLIFIER (DA)	-	DA1:	$V1 = 1V$,$R_8 = R_{10} = 10k\Omega$, $R_9 = 1k\Omega$, $R_{11} = 2.5k\Omega$.	-
		DA2, DA3:	$R_8 = R_{10} = 30k\Omega$, $R_9 = R_{11} = 1k\Omega$	-
		DA4, DA5:	$V_2 = V_4 = -0.85V$, $R_8 = R_{10} = R_9 = R_{11} = 1k\Omega$	-
		DA6:	$V_3 = 0V$, $R_8 = R_{10} = R_9 = R_{11} = 1k\Omega$	-
INVERTING AMPLIFIER (IA)	-	IA1, IA3, IA4 (MB3), IA5:	$R_{13} = R_{14} = R_{15} = 1k\Omega$	-
		IA2	$R_{14} = R_{15} = 5k\Omega$, $R_{13} = 1k\Omega$	-
		IA4	$R_{13} = 1k\Omega$, $R_{12} = 1.1k\Omega$, $R_{15} = 100$	-
I-V CONVERTER (IVC)	-	IVC1	$R_{12} = 3k\Omega$	-
		IVC2 (MB3)	$R_{12} = 5k\Omega$	-
		IVC2 (MB4)	$R_{12} = 6k\Omega$	-

Fig.3.8 (b) illustrates the output voltage $\frac{\partial Y_h}{\partial w_{12}}$ representing the sigmoid derivative from MB1. Fig. 3.8 (c) and (d) represents the obtained δ_2 for the ideal output $Y = 1$ and $Y = 0$, respectively. In this simulations, we check the possibility of the binary output and the performance of the basic blocks. Fig. 3.8 (e) represents the ideal and real outputs of the multiplier δ_1 in MB3 for $\frac{\partial Y_h}{\partial w_{12}}$ variation from $-0.25V$ to $0.25V$. Fig. 3.8 (f) and (g) represent the multiplication output $\frac{\partial E_o}{\partial w_{23}} = X' \cdot \delta_1$ for $X = 0V$ and $X = 1V$, respectively. Fig. 3.8 (h) represents the output $\frac{\partial E_h}{\partial w_{12}} = Y_h \cdot \delta_2$ of the MB3 for the variation of Y_h from $-0.25V$ to $0.25V$ and the variation of δ_2 from $0V$ to $1V$.

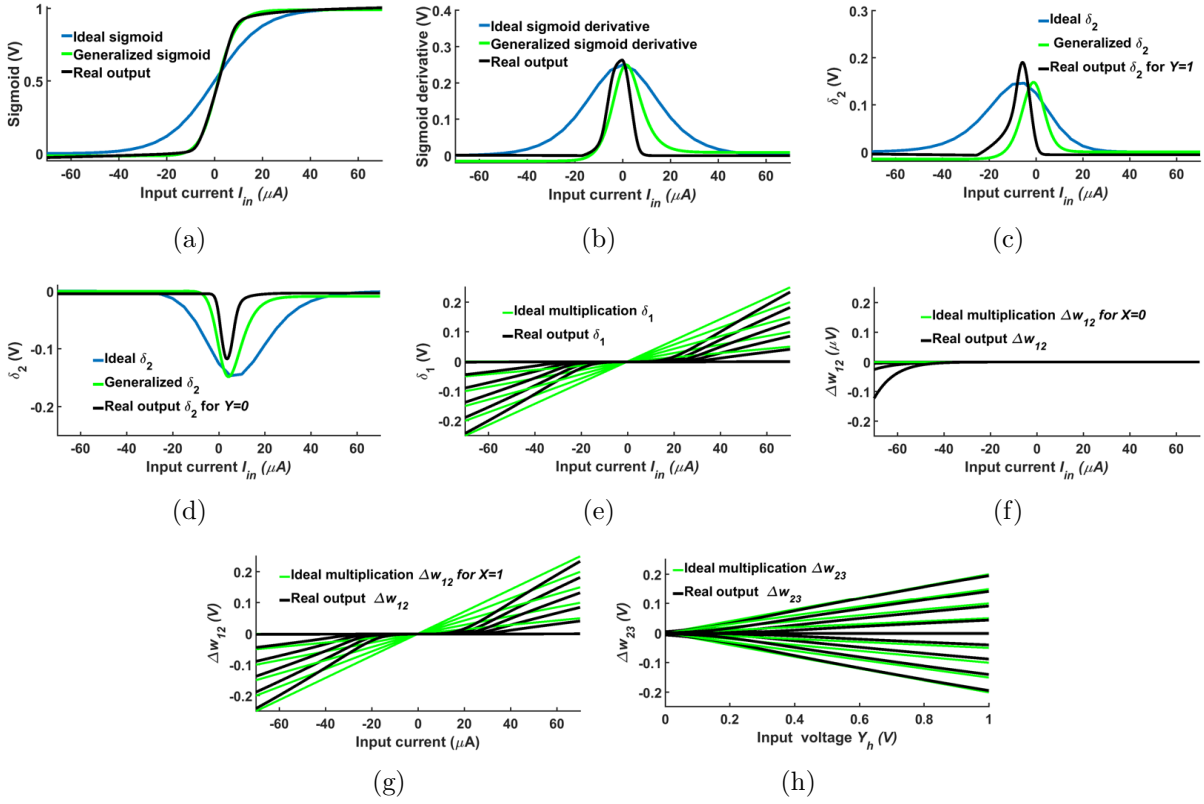


Figure 3.8: Output voltages from the circuit simulation: (a) sigmoid Y_h from MB1, (b) sigmoid derivative $\frac{\partial Y_h}{\partial w_{12}}$, (c) δ_2 for $Y = 1$ from MB2, (d) δ_2 for $Y = 0$ from MB2, (e) δ_1 from MB3, (f) $\Delta w_{12} = \frac{\partial E_h}{\partial w_{12}}$ for $X = 0V$, (g) $\Delta w_{12} = \frac{\partial E_h}{\partial w_{12}}$ for $X = 1V$ and (h) $\Delta w_{23} = \frac{\partial E_o}{\partial w_{23}}$ from MB4.

Fig. 3.9 illustrates the timing diagrams for the circuits. Fig. 3.9 (a) shows the forward propagation results from input layer to hidden layer and the current read from one of the transistors in the CROSSBAR 1, and Fig. 3.9 (b) illustrates the corresponding sigmoid and sigmoid derivative output. Fig. 3.9 (c) and (d) illustrate the same parameters as (a) and (b) but for the propagation from the hidden layer to the output layer and the CROSSBAR 2. Fig.

3.9 (e) illustrates the ideal output Y and the error e calculated as the difference between ideal output Y and real output Y_h from MB2. Fig. 3.9 (f) shows the corresponding value of δ_2. Fig. 3.9 (g) shows the error e_h in terms of output current I_e from the crossbar obtained from the backpropagation to the hidden layer, and Fig. 3.9 (h) shows the corresponding δ_1 value from ANALOG SWITCH 3. Fig. 3.9 (i) illustrates the input signal X and corresponding change in weight $\frac{\partial E_h}{\partial w_{12}}$ obtained from ANALOG SWITCH 4 in MB3. Fig. 3.9 (j) shows the required change in weight $\frac{\partial E_o}{\partial w_{23}}$ from MB4.

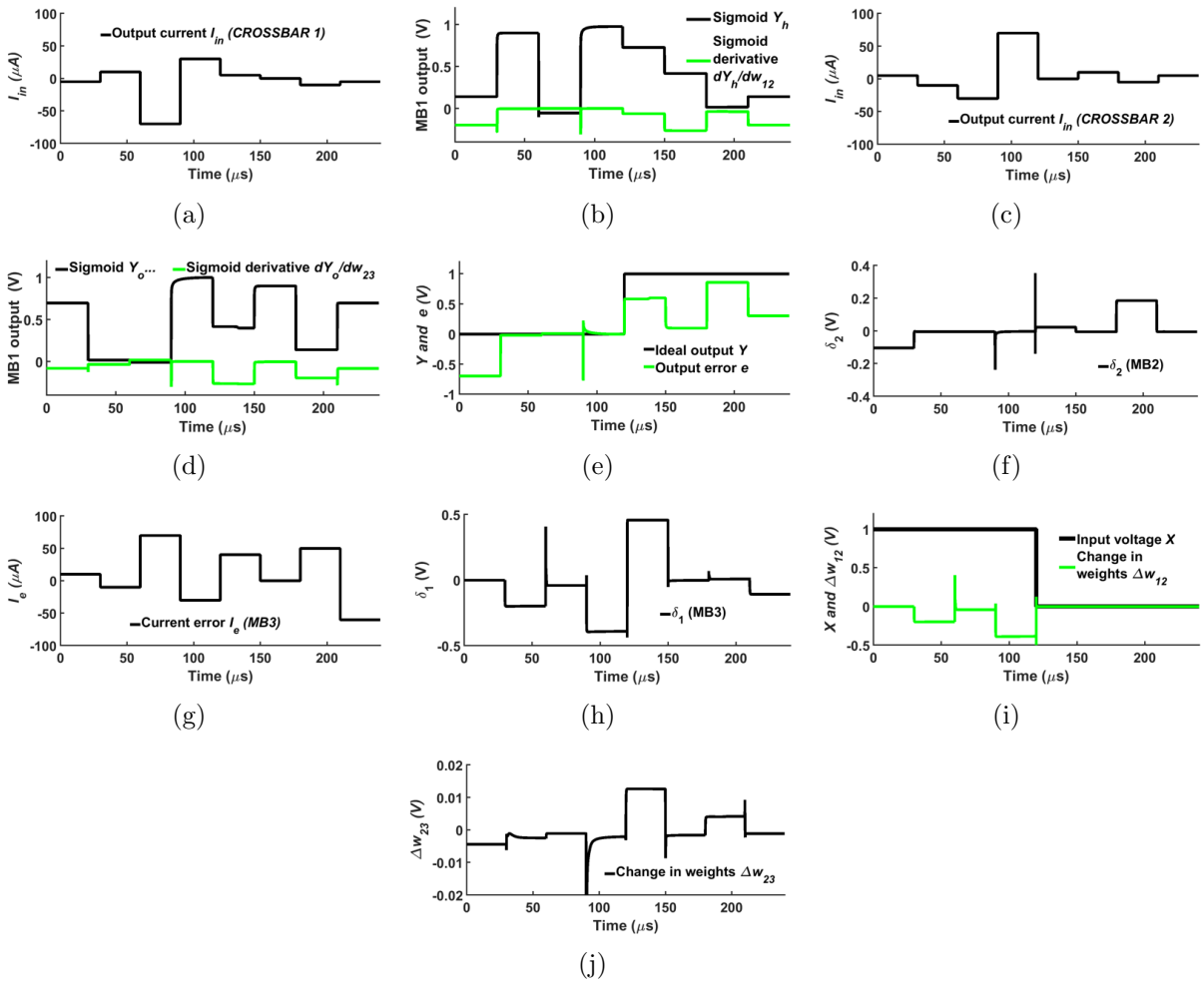


Figure 3.9: Timing diagram for the proposed circuits: (a) output current $I1$ form the CROSSBAR 1, (b) corresponding sigmoid and sigmoid derivative functions, (c) output current $I2$ from the CROSSBAR 2, (d) corresponding sigmoid and sigmoid derivative outputs, (e)ideal output Y and output error e, (f) corresponding δ_2, (g) hidden layer error e_h in terms of current I_e, (h) corresponding δ_1, (i) input X and required change in weight $\Delta w_{12} = \frac{\partial E_h}{\partial w_{12}}$ for the CROSSBAR 1 and (j) required change in weight $\Delta w_{23} = \frac{\partial E_o}{\partial w_{23}}$ for the CROSSBAR 2.

Table 3.2 represents the calculation of the on-chip area and maximum power dissipation for separate components for the analog leaning circuit implementation and additional components

and activation functions. In addition, the example of area and power dissipation for a small crossbar is shown.

Table 3.2: Power consumption and on-chip area calculation for the separate circuit components.

Circuit component	**Power consumption**	**On-chip area**
Crossbar (4 input neurons and 10 output neurons)	$5\mu W$	$1.36\mu m^2$
Sigmoid	$11.4\mu W$	$184.00\mu m^2$
Current buffer	$149.0\mu W$	$280.00\mu m^2$
OpAmp (maximum)	$39.8mW$	$2801.76\mu m^2$
Analog switch	$162.3\mu W$	$1.55\mu m^2$

Table 3.3 shows the on-chip area and maximum power dissipation for separate components for the main blocks of the proposed analog backpropagation learning circuit implementation.

Table 3.3: Area and power calculations for the main blocks of the proposed design and total area and power for three layer network.

Configuration	Area (μm^2)	Maximum Power(mW)
MB1 (hidden layer)	4885.86	3.70
MB2 + MB1 (output layer)	8264.88	10.64
MB3	15238.69	61.78
MB4	9734.33	39.53
Total	**38123.76**	**115.65**

3.5 Discussion

The proposed analog hardware implementation of the backpropagation algorithm can be used to implement the online training of different learning architectures. The analog memristive learning architecture allows to remove the additional software based or digital offline training and learning process. This can increase the processing speed and reduce the processing time.

The limitations of the proposed architecture includes the scalability of the memristive crossbar arrays and limitations of the current memristive devices. The problems of the parasitics, leakage current and sneak paths in the memristive crossbar have to be investigated and solved. The other drawback is a limitation of the memristive devices in terms of the number of resistance

levels that can be achieved for particular memristive devices. The abilities of the different memristors should be investigated. The limitations of the memristive devices, electromagnetic effects, frequency effects and their effect on the accuracy and the performance of the proposed learning architecture have to be studied.

The CMOS part of the learning architecture can also be improved further. The scalability of the architecture should be studied and on-chip area and power dissipation of the learning architecture with additional components, such as mixed-signal controller and weight update circuit should be investigated and measured. As the OpAmp based components in the proposed architecture produce the highest power dissipation and possess the largest on-chip area, the design of the OpAmp can be improved to reduce the area and total power consumption or replace it with the low power amplifier.

The addition, the memory and storage units, weight update and control circuits should be built. The testing of the complete system combining all the components has to be performed and the limitations, such as loading effects and parasitics, have to be identified. The effect of the additional components to the overall system performance and processing speed have to be determined.

3.6 Summary

We proposed the analog CMOS-memristive learning circuit for implementation of various learning architectures. We demonstrate a complete circuit level implementation of learning circuits for three layer neural network using memristive crossbar. The presented design of crossbar does not take into account memristive nonlinearities, sneak path problems, leakage currents and variability of real devices, and will be a topic to investigate further. The area and power of the proposed circuit design need to be further optimized a fully parallel implementation for real-time applications.

Chapter 4

Learning in HTM

4.1 Introduction

In this chapter the HTM learning process in investigated. Various aspects of HTM learning are considered. The architecture of modified HTM Spatial Pooler and HTM Temporal Memory incorporating the learning process is proposed. The architecture of how the proposed backpropagation analog memristive learning circuits can be used in conventional and modified HTM architectures is shown. The rule-based HTM approach without learning is investigated and hardware implementation of this approach is designed.

4.2 Background

4.2.1 HTM overview

HTM is a machine learning algorithm that is inspired by the biological functionality of the neocortex [13]. This algorithm attempts to reproduce the following characteristics of the neocortex: processing and representation of the information in the sparsely distributed manner, encoding of the sensory data in real time [14], consideration of the temporal changes of the input data and ability to make predictions based on the history of previous events [15].

The implementation of HTM requires consideration of two modules: Spatial Pooler (SP) and Temporal Memory (TM). The studies on functionality of HTM revealed that SP alone is able to learn and classify the data sets that are related to the numbers, pixels and letters [40]. This was validated on the practical applications related to the biometric recognition, object categorization [41], face recognition [42, 16, 17], speech recognition [16], gender classification [43], handwritten digits recognition [44, 40, 45] and natural language processing [46]. The purpose of SP in HTM is to train the system to recognize certain patterns of the input data such that for different inputs that possess similar characteristics particular attributes in the output are activated [13]. In other words, SP generates the sparse distributed representation (SDR) from the given input data. While SP considers the common patterns in the spatial domain, TM looks for the temporal patterns. It examines the temporal changes of the input data and makes predictions based on previous experience of having certain patterns being followed by particular types of other patterns.

HTM was inspired by the neocortex functions and describes the overall theory outlined in the book *On Intelligence* [13]. The HTM theory was developed based on several functional and structural biologically relevant observations of the neocortex. These observations included the structure and functional hierarchy of information processing within the neocortex, the generality of the neocortex algorithms, the ability to process features in a sparse, distributed manner, the layered process of extracting and deciphering complex information, the real-time encoding of sensory information, the dynamic streaming and sequencing of memories for data processing, and the ability to process data online. These principles are incorporated into HTM algorithm design, which consists of two distinct modules: SP [47] and TM [48].

The design and architecture of HTM are based on the modeling of pyramidal neurons in the cortex and are very different from those of the neuron models used in artificial neural networks. The model permits active dendrites for the recognition of independent patterns from a large population of cells. The ability to integrate synapses into the model enables the prediction of the sequence transition of cell activities. Furthermore, the HTM architecture relies on the principles of Hebbian learning, the network connectivity of sensory cortices, homeostatic excitability control, and the structural plasticity, which depends on the activity.

The HTM algorithm is organized into regions comprising columns of cells, as shown in Fig. 4.1, with each column of cells forming a single computational unit in which SP operates. The mechanism of emulating the sparse activation of neurons is referred to as inhibition, which is the ability of columns in SP to inhibit neighboring columns within an inhibition radius (size of the local neighborhood) from becoming active. The junctions between cells are called synapses, and the synapses on a columns dendritic segment enable connecting to the bits in the input space. The receptive field is the available input space in which the columns can connect, and the permanence value is a measure of growth between columns and one of the cells in the receptive field. If the value of a synapse's permanence is above a permanence threshold, it is considered to be fully connected.

Since HTM is a cortical algorithm, it provides a new model and direction for the hardware implementation of neocortical functions [49]. The robustness and feasibility of HTM have been tested using image processing, object categorization and recognition applications [50]. These algorithmic implementations are promising. [51] and [52] depict early digital designs for the VLSI architecture and FPGA implementation of HTM, respectively. The analog design implementations include a memristor-based design [42] and scalable memristor crossbar architectures [16] for the SP and a mixed-signal design of spin neurons and a crossbar-based implementation [53] for both the SP and TM.

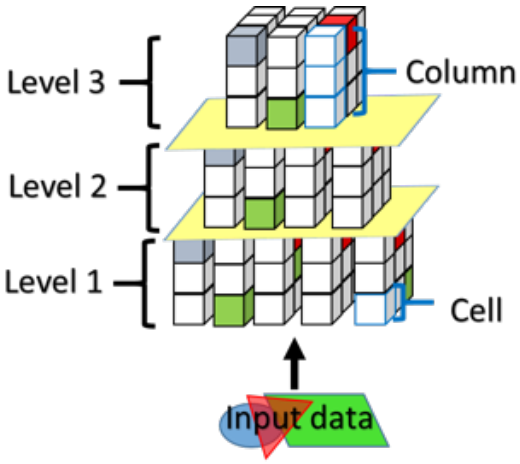


Figure 4.1: Representation of the 3-level hierarchy in the HTM

The SP on its own is capable of learning and classifying different data sets, such as numbers, letters and pixels [40]. Thus, the implementation of the SP alone in applications such as image

processing, pattern recognition and speech recognition yields good performances. The SP is realized via implementation in four phases: initialization, overlap computation, inhibition, and learning [18]. The steps for implementing the SP are as follows:

1. The SP accepts input data bits from sensory data or from other regions of the HTM.
2. The HTM regions are initialized by selecting a fixed number of columns that can receive inputs. Each column is connected to the input using a dendrite segment having a set of potential synapses. The potential synapse and its corresponding permanence value are initialized randomly around the permanence threshold. Some of these potential synapses with a permanence value greater than the permanence threshold will already be connected.
3. The number of synapses on each column connected to active (ON) input bits is calculated; these connected synapses are referred to as active synapses.
4. These active synapses are multiplied by a boosting factor. The boosting factor represents the frequency of activeness of the column relative to its neighborhood.
5. Within the inhibition radius, the columns with the highest activations become active, while the others are disabled. Since the inhibition radius depends on the spread of input bits, the column activations are sparsely distributed.
6. The SP region follows a Hebbian-style learning rule to update the permanence values of all the potential synapses; thus, the synapses are changed from being connected to being unconnected and vice-versa.
7. Step 3 is repeated for subsequent inputs.

The activation rule in step 5 is implemented using the k-winners-take-all computation within a local neighborhood. Ideally, the parameter k can be adjusted to regulate the desired number of winning columns [18]. However, in [42] and [16], the inhibition phase is implemented by the WTA circuit; as a result, the value of the desired activity level is limited to 1. A more formal mathematical description of SP is provided in the *supplementary material*.

The second part of HTM is the TM. Whereas the SP is responsible for the sparse representation of the input data, the TM considers the changes that occur over time, learns the patterns and attempts to make predictions based on the history of input information. The TM aims to learn the connections between cells within the same layer, while the SP aims to learn the feed-forward connections between input bits and columns. The temporal memory algorithm is also known as the sequence memory. The main roles of the TM are the following: (1) to learn sequences of active columns from the SP over time; (2) to predict which patterns come next based on the temporal context of each input.

The TM collects the input from the SP, with the feed-forward inputs of the TM originating from the active columns. The steps for implementing the TM are as follows:

1. Activate the cells in each of the active columns that are in the predictive state. If none of the active columns are in the predictive state, then activate all of the cells in the column. These active cells now represent input related to prior input.

2. Find the total number of synapses connected to active cells for all the dendrite segments of every cell in a given layer. If this number exceeds a threshold, the corresponding dendrite segment is assigned to be active. The cells corresponding to this dendrite segment are set to the predictive state unless already set because of the feed-forward input. The collection of cells in the predictive state represents the predicted pattern for the layer.

3. The activation of a dendrite segment is required to update the permanence values of the synapses in the given segment. The permanence values of the potential synapse are increased for active cells and decreased for inactive cells. These modifications are temporarily marked, and the synapses on already trained segments are made active, leading to prediction.

4. The feed-forward inputs can change the cell state from inactive to active (if this occurs, the temporary marks are removed) and affect each potential synapse of the cell. In other words, the permanence of synapses is only updated in the case of the correct prediction of feed-forward activation of a cell.

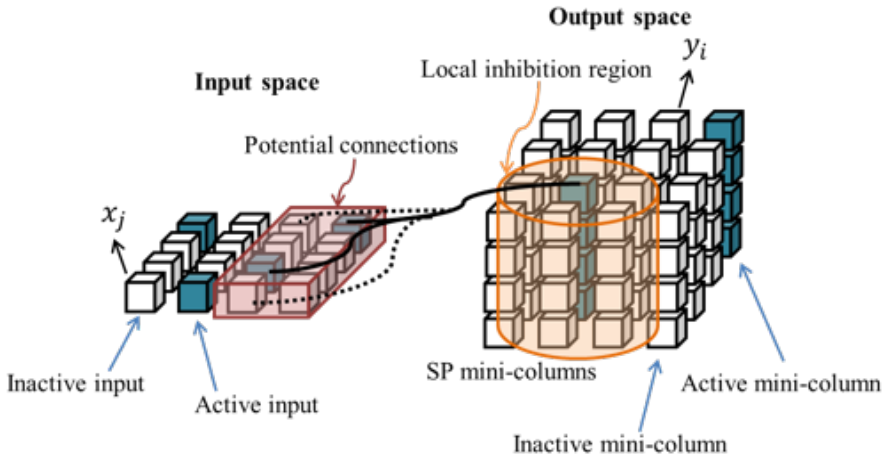


Figure 4.2: The HTM SP converts the inputs to sparse distributed output patterns. The synaptic connections between SP mini-column to the neurons of the input space or input mini-columns are formed. The local inhibition region corresponds to the selection of the HTM SP mini-columns that receive the largest number of the active inputs within an inhibition radius γ. The synaptic permanences are modified using Hebbian-like rules by strengthening the active inputs and weakening the inactive inputs.

5. On the other hand, to change the cell state from the predictive state to the inactive state, we need to undo any permanence changes marked as temporary for each potential synapse. When one cell incorrectly predicts the feed-forward activation of another cell, the permanence values of the previously active synapse are decreased.

The predictive state in the TM is purely an internal state of the cell. The active cells resulting from the feed-forward input propagate their activity to avoid the chain of predictions leading to further predictions.

4.2.2 Mathematical framework

Spatial Pooler

Fig.4.2 illustrates an arrangement of the input space and the output space of the HTM SP, where x_j refers to the j-th input neuron in the input space. The outputs are topologically arranged into mini-columns, where y_i is the ith output SP mini-column in the output space. The mini-column y_i is connected to the part of the input space, which is called potential

connections. The synapse refering to the i-th SP mini-column is located in a hypercube of the input space with the center in x_i^c and edge length γ. If the synaptic permanence of the synapse is greater than the threshold value, the synapse is connected. Eq.4.1 represents the potential input connections $PI(i)$ between to the input space and i-th mini-column [18, 17].

$$PI(i) = \{j | \imath(x_j; x_i^c, \gamma) \text{ and } (z_{ij} < \rho)\} \tag{4.1}$$

where, $\imath(x_j; x_i^c, \gamma) = 1, \forall x_j \in (x_i^c, \gamma)$, and $z_{ij} \sim U(0,1)$, z is selected randomly from the uniform distribution U having range $[0,1]$. The parameter ρ denotes the fraction of inputs that are potential connections within the hypercube of the input space.

The set of connected synapses are represented as a binary matrix $\mathbf{B}$ in Eq.4.2. The synapses are considered to be connected, if their value is above the threshold value θ_c.

$$\mathbf{B}_{ij} = \begin{cases} 1 & \text{if} \quad \mathbf{S_{ij}} \geq \theta_c \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

The threshold parameter θ_c refers to the percentage of connected synapses. For instance, $\theta_c = 0.4$ indicates that 40% of the potential synapse are connected. The synaptic permanence from the j-th input to the i-th SP mini-column is represented by the matrix $\mathbf{S_{ij}} \in [0,1]$ shown in Eq.4.3.

$$\mathbf{S}_{ij} = \begin{cases} U(0,1) & \text{if} \quad j \in PI(i) \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

Next, the local inhibition part of the HTM SP is executed through the local inhibition mechanism that implies the SP mini-columns in a particular neighborhood inhibit each other. This neighborhood $\mathbf{N}_i$ of the i-th SP mini-column is defined by Eq.4.4, where, $\|y_i - y_j\|$ determines the Euclidean distance between the mini-columns i and j and the parameter ϕ is used to control

the inhibition radius.

$$\mathbf{N}_i = \{j | \, \|y_i - y_j\| < \phi, i \neq j\} \tag{4.4}$$

The parameter ϕ is useful when the inputs are also topologically arranged to the mini-columns to ensure that all the inputs are affected by the inhibition process. If the receptive field size increases, the parameter ϕ also increases. To determine ϕ, the average number of connected input spans of all the SP mini-columns are multiplied by the number of mini-columns per inputs. If the dimensions of SP inputs and mini-columns are same, ϕ is set equal to γ.

The measure that determines the activation of SP mini-columns for a given input pattern $\mathbf{Z}$ is called input overlap. The input overlap shown in Eq.4.5 is calculated as a feed-forward input to each mini-column.

$$o_i = \beta_i \sum_j \mathbf{B}_{ij} \mathbf{Z}_j \tag{4.5}$$

The parameter β_i represents the boosting factor that controls the excitability of each SP mini-column. The appropriate value of the boosting factor is either selected initially or modified during training and learning phase of the HTM SP.

The activation of the SP mini-column depends on two main conditions: the input overlap is greater than a stimulus threshold θ_s and is among the top s percentile ($prctile$) of its neighborhood. Eq.4.6 refers to the active column selection, where α_i is the activity of the SP mini-columns, and $\mathbf{NO}(i) = \{o_j | j \in \mathbf{N}(i)\}$ with s being the target activation density. This activation rule refers to the implementation the k-winners-take-all computation within a local neighborhood.

$$\alpha_i = 1, \text{if} \quad (o_i \geq \text{prctile}(\mathbf{NO}(i), 1 - s)) \text{ and } (o_i \geq \theta_s) \tag{4.6}$$

In the original algorithm, the parameter k, can be adjusted to regulate the desired number of

winning columns [18]. However, to simplify the algorithm for the hardware implementation, the value of the desired activity level is limited to 1 because the the inhibition phase is implemented by the Winner-Takes-All (WTA) circuits [42, 16].

In the learning phase of the HTM SP, feed-forward connections are learned using Hebbian rule and the boosting factor is updated considering the last T inputs in time. The reinforcement of input connections is performed by the increasing of the permanence value ρ by ρ^{+}. Whereas, the permanence of inactive connections is decrease by ρ^{-}. It should be noticed that the permanence values are kept withing the boundaries of from 0 to 1.

The update process of the boosting factor depends on the time-average activity and the recent activity of the SP mini-columns [18]. The parameter $\bar{\alpha_i}(t)$ refers to the time-average activation level in time t considering T previous inputs and current activity $\alpha_i(t)$ of the i-th mini-column shown in Eq.4.7.

$$\bar{\alpha_i}(t) = \frac{(T-1) \times \bar{\alpha_i}(t-1) + \alpha_i(t)}{T} \tag{4.7}$$

The calculation of the recent activity $< \bar{\alpha_i}(t) >$ of the mini-columns in a particular neighborhood in time t is shown in Eq.4.8.

$$< \bar{\alpha_i}(t) >= \frac{1}{|\mathbf{N}(i)|} \sum_{j \in \mathbf{N}(i)} \bar{\alpha_i}(t) \tag{4.8}$$

The update process of the boosting factor is shown in Eq.4.9, where the parameter η controls the adaptation effect in the HTM SP.

$$\beta_i(t) = e^{-\eta(\bar{\alpha_i}(t) - <\bar{\alpha_i}(t)>)} \tag{4.9}$$

Temporal Memory

The predictive state of the cells in the HTM TM is shown in Eq.4.10, where A^t is the activation matrix of size $N \times S$ refering to N columns and S neurons per column with elements a_{ij}^t of A^t denoting the activation state of the i-th cell and j-th column at the time point t. The permanence of the d-th segment of ith cell in the j-th column is denoted as D_{ij}^d. The parameter $\widetilde{D}_{ij}^d$ corresponds to connected synapses, and θ refers to the segment activation threshold value [48].

$$\pi_{ij}^t = \begin{cases} 1 & \text{if } \exists_d \left\| \widetilde{D}_{ij}^d \cdot A^t \right\|_1 > \theta \\ 0 & \text{Otherwise} \end{cases} \tag{4.10}$$

The calculation of the activation state is shown in Eq.4.11, where, W^t refers to the top $s1$ percentage of column that has largest number of synaptic inputs. Typically $s1$ is set to 1% to 2% [48]. The cell in the winning column is activated it was in the previous time step.

$$a_{ij}^t = \begin{cases} 1 & \text{if } j \in W^t \text{ and } \pi_{ij}^{t-1} = 1 \\ 1 & \text{if } j \in W^t \text{ and } \sum_i \pi_{ij}^{t-1} = 0 \\ 0 & \text{Otherwise} \end{cases} \tag{4.11}$$

The dendrite segment activations and the learning of lateral connections is preformed using a Hebbian-like rule. Eq.4.12 represents the reinforcement of the depolarized cell in a segment that subsequently become active, where $\dot{D}_{ij}^d$ is a binary matrix representing only positive elements of D_{ij}^d, and the small values ρ^- and ρ^+ are for negative reinforcement of inactive synapse and positive reinforcement of active synapse in dendrite segments, respectively.

$$\Delta D_{ij}^d = \rho^+ \dot{D}_{ij}^d \cdot A^{t-1} - \rho^- \dot{D}_{ij}^d \cdot (1 - A^{t-1}) \tag{4.12}$$

Eq. 4.13 refers to the long term that is introduced by including a small decay to the active

segments that did not become active, where $a_{ij}^{t} = 0$ and $\left\| \widetilde{D}_{ij}^{d} \cdot A^{t-1} \right\|_{1} > \theta$, with $\widetilde{\rho}^{-} << \rho^{-}$ [48, 17].

$$\Delta D_{ij}^{d} = \widetilde{\rho}^{-} \dot{D}_{ij}^{d} \tag{4.13}$$

The research study [48] states that the Hebbian-like learning rule can be replaces by the gradient descent method on the cost function, such as prediction error, which could lead to the better accuracy results.

4.2.3 HTM algorithm

The Algorithm 2 summaries the original HTM structure and involved main processing steps. Lines 1- 33 represents the HTM SP implementation, while lines 34-47 refer to the HTM MT processing. Lines 1-20 refer to the initialization stage and local inhibition calculations. Line 21 refers to the overlap calculations. The lines 24-25 represent the inhibition stage of HTM SP. Lines 26-29 includes the learning stage of the HTM SP, where reinforcement of active connections and suppression of inactive connections is performed. Lines 30-33 correspond to the update of boost factor. The HTM TM processing starts at line 34, where the number of columns and number of neurons per column is initialized the the threshold θ is defined. Lines 35-38 refer to the determination of predictive state of the cells. The calculation of the activation function state is performed in lines 39-44. Finally, the reinforcement of depolarization cell corresponds to line 45, and the long-term depression is shown in lines 46-47.

Algorithm 1 HTM algorithm

▷ **HTM SP**

Define the size of input neighborhood with potential connections, x_i^c, γ, ρ, η, θ_c, size of the local inhibition region, θ_s
Determine ϕ by multiplying the average number of connected input spans of all the SP mini-columns by the number of mini-columns per inputs.
if size (inputs)=size(mini-columns) **then**
    $\phi = \gamma$
$z_{ij} \sim U(0,1)$
if $\forall x_j \in (x_i^c, \gamma)$ **then**
    $\imath(x_j; x_i^c, \gamma) = 1$
for $\imath(x_j; x_i^c, \gamma)$ and $(z_{ij} < \rho)$ **do**
    $PI(i) = j$
if $j \in PI(i)$ **then**
    $\mathbf{S}_{ij} = U(0,1)$
else
    $\mathbf{S}_{ij} = 0$
if $\mathbf{S_{ij}} \geq \theta_c$ **then**
    $\mathbf{B}_{ij} = 1$
else
    $\mathbf{B}_{ij} = 0$
for $|y_i - y_j| < \phi, i \neq j$ **do**
    $\mathbf{N}_i = j$
$o_i = \beta_i \sum_j \mathbf{B}_{ij}\mathbf{Z}_j$
for $j \in \mathbf{N}(i)$ **do**
    $\mathbf{NO}(i) = o_j$
if $(o_i \geq \text{prctile}(\mathbf{NO}(i), 1-s))$ and $(o_i \geq \theta_s)$ **then**
    $\alpha_i = 1$
else
    $\alpha_i = 0$
if input connections are active **then**
    $\rho = \rho + \rho^+$
else if input connections are inactive **then**
    $\rho = \rho + \rho^-$
for time period t **do**
    $\bar{\alpha_i}(t) = \frac{(T-1)\times\bar{\alpha_i}(t-1)+\alpha_i(t)}{T}$
$< \bar{\alpha_i}(t) >= \frac{1}{|\mathbf{N}(i)|} \sum_{j \in \mathbf{N}(i)} \bar{\alpha_i}(t)$
$\beta_i(t) = e^{-\eta(\bar{\alpha_i}(t) - <\bar{\alpha_i}(t)>)}$

▷ **HTM TM**

Define N, S, θ
if $\exists_d \left\| \widetilde{D}_{ij}^d \cdot A^t \right\|_1 > \theta$ **then**
    $\pi_{ij}^t = 1$
else
    $\pi_{ij}^t = 0$
if $j \in W^t$ and $\pi_{ij}^{t-1} = 1$ **then**
    $a_{ij}^t = 1$
else if $j \in W^t$ and $\sum_i \pi_{ij}^{t-1} = 0$ **then**
    $a_{ij}^t = 1$
else
    $a_{ij}^t = 0$
$\Delta D_{ij}^d = \rho^+ \dot{D}_{ij}^d \cdot A^{t-1} - \rho^- \dot{D}_{ij}^d \cdot (1 - A^{t-1})$
if $a_{ij}^t = 0$ and $\left\| \widetilde{D}_{ij}^d \cdot A^{t-1} \right\|_1 > \theta$, with $\widetilde{\rho}^- << \rho^-$ **then**
    $\Delta D_{ij}^d = \widetilde{\rho}^- \dot{D}_{ij}^d$

4.3 HTM with memristor logic circuits and its application for visual data classification problem

4.3.1 Overview of Proposed Architecture

Spatial Pooler

The primary difference from the aforementioned HTM algorithm is that we are changing the criteria for the selection of winning columns that occurs in the inhibition phase. Instead of considering the k-th largest overlap value, we propose to calculate the threshold and establish the selection of the winning columns based on the average value of the *overlaps* in the modified SP.

The implementation of the proposed SP design is discussed in Section 4.3.2 and requires consideration of the availability of resources. Two types of resources being regarded in the hardware design of interest are processing speed and on-chip area. The user preferences concerning the above-mentioned resources dictate whether the operations will be performed in parallel or serially. The former implementation requires concurrent replication of the circuit for each of the inhibition blocks such that the processing of an image is performed at once. This benefits the design in terms of fast processing. However, a high on-chip area demand is suffered. Alternatively, serial implementation would require a single module for the processing of one inhibition block after another, which would reduce the required on-chip area. Thus, trade-offs between high-speed processing and a low-area network should be made.

Temporal Memory

In the current work on TM, we use a single training image known as a ***class map***. The class map considers the training images within the same class and combines all of their features. This allows the matching of the patterns in each of the trained classes to be executed through the comparison of the testing image with a single image. As a result, the memory requirements

as well as the time for the processing are reduced.

Such a design of TM is based on two properties: focusing and reflection. The fact that the circuit of TM is established after the realization of the SP shifts the *focus* of the learning procedure onto the important and unimportant features of the original training images. The *reflection* property is related to the changes in the features over time. This is realized by considering the importance of the input bits and introducing corresponding changes to the weights of the TM cells.

The weights of the TM cells can be incremented or decremented by the weight update value $\pm\Delta$. The positive weight update procedure, i.e., increasing the weight by $+\Delta$, occurs when the input bit represents an important bit, in other words, when the input bit from the feature extracted image is 1. A 0 value of the input bit of the feature extracted image represents an unimportant feature and will contribute to the negative update of the weight ($-\Delta$). Thus, in contrast to the process of the formation of the feature extracted images, which used binary weights with values of either 1 or 0, the learning process utilizes the multi-valued weights of the TM cells.

The formation of the class map for one of the image classes is demonstrated on Fig. 4.3. The inputs to the TM are the multiple feature extracted binary images, while the output is a single analog image of the same size that combines all the important and unimportant features. The training sequence that is applied in the TM eliminates the memory cells that initially had the same weights.

4.3.2 Circuit Design of Modified HTM Architecture

System overview

The overall design of the proposed face recognition system is shown in Fig. 4.4. The system consists of an input data controller with data storage, the HTM SP, an output data controller, HTM TM and a memristive pattern matcher. The images from the capturing device, such as

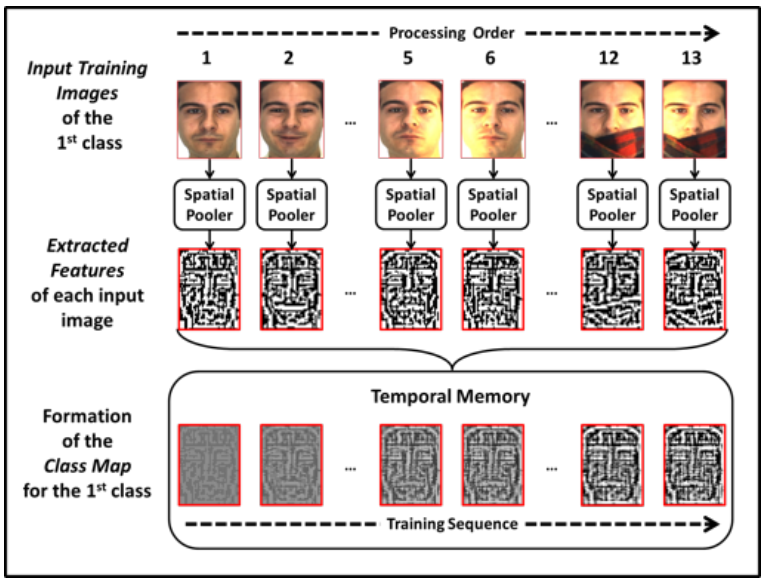


Figure 4.3: The underlying principle of single-class-map formation using TM and feature extracted images obtained from the SP

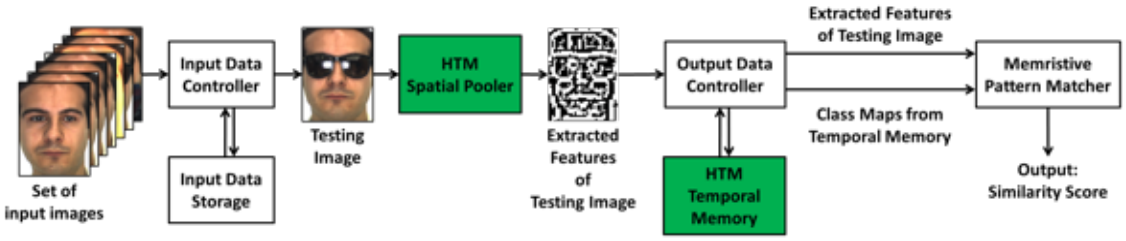


Figure 4.4: High-level block diagram of the proposed pattern recognizer based on Modified HTM. The pattern recognizer consists of an input data controller for captured image storage and preprocessing, an HTM SP for feature extraction, HTM TM for training of the recognizer, an output data controller to control switching between train and testing modes and a memristive patter matcher used for image classification.

a camera, are sent to the input data controller, which selects the required number of images and stores them in the input data storage. Depending on the capacity of the system, the size of the input images and the number of available on-chip resources, the input data controller selects a parallel or serial processing type and rearranges the images into blocks, followed by pre-processing and filtering. Then, the testing image or parts of the image are processed by the proposed modified HTM SP, which performs feature extraction and produces the binary output image containing only the most relevant facial features. Next, the HTM SP output image is sent to the output data controller, which routes the images to HTM TM or the memristive pattern matcher according to the training or testing mode, respectively. Note that the test images in the testing mode and the training images in the training mode are separate sets of

images.

During the training mode, the output data controller saves the image with the corresponding class number into the TM. When a new training image of the same class arrives, the training template of this class, called the *class map*, is updated according to the proposed HTM TM algorithm. In the testing mode, the output data controller directs the image into the memristive pattern matcher and retrieves the class maps of each class from TM. The memristive pattern matcher compares the testing image with all class maps and determines the similarity score for each comparison based on XOR logic and averaging operations. The class of the image corresponds to the match with the minimum difference between the testing image and the training class map (or the maximum similarity score) and is determined using a Winner-Takes-All (WTA) circuit.

Spatial Pooler

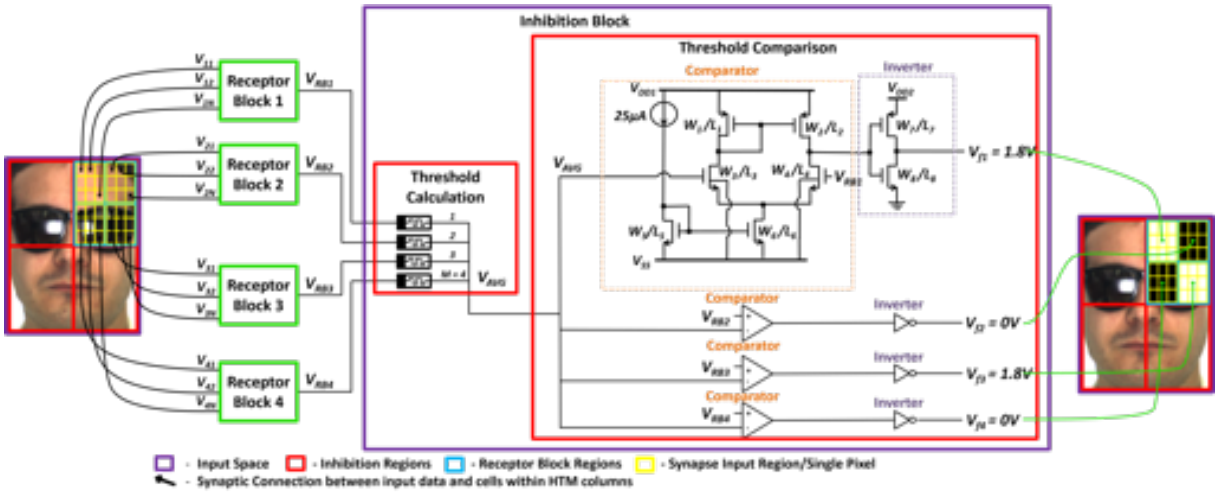


Figure 4.5: HTM SP configuration. The processed image is divided into receptor blocks consisting of N image pixels. M receptor blocks form a single inhibition block. An inhibition block consists of two main parts: a thresholding calculation block and a threshold comparison block. The inhibition block produces a binary output.

Figure 4.5 illustrates the circuit diagram of a single processing block of the proposed SP of the modified HTM. The process of extracting features from an input image is achieved by initially reading input bits (in yellow) from the input space (in purple) by processing the entire image in a block-by-block manner. Each of such blocks on the input space, which are called

inhibition regions (in red), circumscribes the input bits constituting different receptor block regions (blue boxes in the top-right inhibition region). A single processing block, illustrated in Fig. 4.5, processes a single inhibition region and determines which bits within it are important and which are not important.

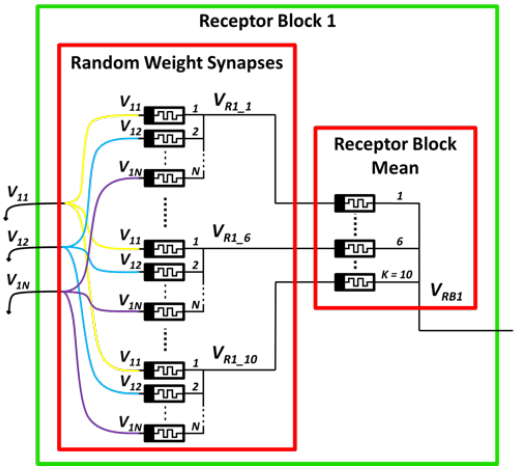


Figure 4.6: Structure of a single receptor block, illustrating parallel synaptic connections represented by $N \times K$ memristors and a block calculating mean of those input synapses, which represents the output of the single receptor block

Figure 4.6 illustrates the structure of a single **receptor block**, which performs an operation similar to that of a single HTM column in HTM theory (hence, both of the terms will be used interchangeably). An illustrated exemplar receptor block has in total $N \times K$ **random weight synapses**, each of which is implemented by a memristor device, and the permanence value of the block corresponds to the memristance value of the device. A synaptic connection is counted as either connected or disconnected based on whether the memristance value is high (R_H) or low (R_L), respectively. If there are M receptor blocks within a single processing block, as illustrated in Fig. 4.5, then within each of these receptor blocks, the group of N synaptic connections having the same index number k such that $k \in [1, K]$ acquire the same set of random weights. According to the proposed design, the parameter K is chosen to have a fixed value $K = 10$, representing 10 repetitions of the same input data.

Hence, the repetition operation is achieved when the bits of a single receptor block region are fetched by a corresponding **receptor block**. Specifically, the voltage signals representing input bits are applied to the memristors of a particular receptor block j such that $j \in [1, M]$ to

produce the output signal V_{RBj}, which represents the averaged weighted sum of its input bits. As illustrated in Fig. 4.6, the average within each receptor block (designated as the **receptor block mean**) is determined by memristor devices having the same high memristance value, R_H. According to HTM theory, the output signal V_{RBj} can then be said to represent the overlap value of the column j, which indicates the importance of the bits connected to it in comparison with the bits connected to other columns within the inhibition region also having M columns.

Next, as illustrated by the **threshold calculation block** in Fig. 4.5, based on the output voltage value of each receptor block, the average value V_{AVG} is calculated for the entire inhibition region. As illustrated by the **threshold comparison block** in Fig. 4.5, this average value V_{AVG} is used as a reference in determining which bits are important and which are not important within a single inhibition block. If the output overlap value V_{RBj} of a receptor block j is higher than the average value V_{AVG}, then the bits connected to that receptor block are counted as important. This, in turn, results in the output voltage signal having the value of $V_{fj} = 1.8\ V$. Selecting a 180 nm TSMC CMOS technology, the voltage 1.8 V corresponds to logic "1", while $V = 0\ V$ refers to logic "0". In contrast, when the overlap value V_{RBj} is less than the average value, the bits connected to that receptor block are counted as unimportant, and the output voltage signal will be $V_{fj} = 0V$.

By processing all the inhibition regions of the input image in the same way, a binary feature extracted image is produced. A feature extracted image will have white bits representing important features and black bits representing the remainder of an image or unimportant features. White bits will be placed at locations of the original image that had important bits. This means that, regardless of their original value, all the bits connected to the receptor block having $V_{RBj} \geq V_{AVG}$ will be represented by the white bits or by the output voltage signal $V_{fj} = 1.8V$. On the other hand, black bits will be placed at locations where the original image had unimportant bits. This means that the remaining bits will be represented by black bits or by the output voltage signal $V_{fj} = 0V$.

The proposed design was verified using the memristor model proposed by [54] for each synaptic connection. The model parameters were set to emulate the memristor device proposed by [55].

The threshold comparison block was simulated using the BSIM model for a TSMC CMOS technology size of 180 nm. In particular, the comparator consists of PMOS transistors having width-to-length ratios of $W_1/L_1 = W_2/L_2 = 3.24\mu/0.18\mu$ as well as NMOS transistors having width-to-length ratios of $W_3/L_3 = W_4/L_4 = 2.0\mu/0.18\mu$ and $W_5/L_5 = W_6/L_6 = 1.08\mu/0.18\mu$. The inverter, in turn, consists of a PMOS transistor with $W_7/L_7 = 0.72\mu/0.18\mu$ and an NMOS transistor with $W_8/L_8 = 0.36\mu/0.18\mu$. The supply voltages $V_{DD1} = V_{DD2} = 1.8\ V$.

Selection of Parameters and Design Trade-Offs

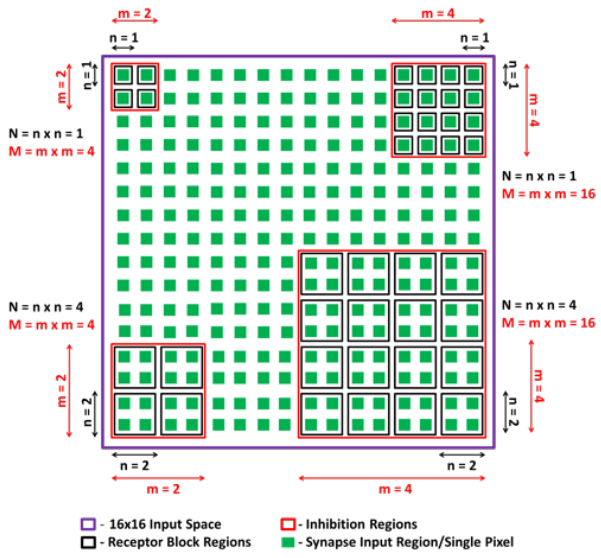


Figure 4.7: An exemplary image consisting of $16 \times 16 = 256$ bits along with 4 different possibilities of how the image can be processed.

There are certain circuit parameters that need to be selected according to design trade-offs, as will be explained in this subsection. Specifically, this includes the selection of N and M circuit parameters of a single processing block, which is illustrated in Fig. 4.5, as well as the total number of such processing blocks, indicated by the parameter P, required to create a whole SP.

Figure 4.7 shows an exemplary image consisting of $16 \times 16 = 256$ bits along with 4 different possibilities of how it can be processed. Depending on the selected pair of parameters (N, M), the image can be processed by dividing its bits (green boxes) into receptor block regions (black boxes) and inhibition regions (red boxes) of various sizes. Although a single receptor block

region with dimensions of $n \times n$ bits reads in total $N = n \times n$ bits, a single inhibition region with dimensions of $m \times m$ incorporates signals from, in total, $M = m \times m$ receptor blocks. Because a single processing block processes the input bits of a single inhibition region, the circuit illustrated in Fig. 4.5 is capable of processing a total of $N \times M$ bits.

The whole SP then consists of P number of such processing blocks. Because each processing block is independent in terms of computations, the entire SP is capable of processing $P \times M \times N$ bits simultaneously. This, in turn, allows the whole image to be processed in a parallel manner by dividing its bits into inhibition regions, each of which is processed by a separate processing block. If the input to the system is preprocessed to have fixed dimensions of $X \times Y$ bits, the require parameter P can then be calculated as $P = (X \times Y)/(N \times M)$.

The selection of appropriate N, M, and P parameters is crucial to providing the most optimal system characteristics. In particular, the performance of the proposed SP was evaluated in terms of its on-chip area and dissipated power as well as its qualitative effect on the system's processing time and its quantitative effect on the system's recognition accuracy.

Consider the exemplary image having in total $16 \times 16 = 256$ bits (Fig. 4.7) processed by the SP constructed by processing blocks having $(N, M) = (1, 4)$ parameters (top-left corner) and $(N, M) = (4, 16)$ parameters (bottom-right corner). Assuming that all input images are preprocessed to have exactly the same dimensions, in the case when $(N, M) = (1, 4)$, the SP should consist of $P = 64$ processing blocks, and in the case when $(N, M) = (4, 16)$, the SP should consist of $P = 4$ processing blocks. Because the number of components required for the SP is higher in the first case, the on-chip area and the dissipated power are expected to be higher for $P = 64$ in comparison with $P = 4$. The time required to process an input set is, however, the same, as all the bits are fetched in parallel. Finally, it will be shown later that because the inhibition process with $P = 64$ becomes more localized, the accuracy results are higher for $P = 64$ compared to $P = 4$. Hence, this work incorporates the analysis to determine circuit parameters to achieve optimal system characteristics based on the face database that was selected for verification.

Temporal Memory

The hardware implementation of HTM TM consists of a discrete analog memristive memory array and circuit for updating data in the array based on the proposed HTM TM concept. The memory array consists of memristive memory cells, where each cell corresponds to a single image pixel. For an image of size $A \times B$, a memory array consisting of $A \times B$ memory cells is required.

The operating principle of a single memory cell is based on the ability of a memristor to memorize its state and change the resistance according to the applied voltage. Fig. 4.8 shows the proposed memristive memory cell consisting of three branches. Such a cell requires five input signals, V_{w1}, V_{w2}, V_{w3}, V_r, and V_c, and produces one output signal: V_o. The input signals V_w, V_r, and V_c correspond to the process of writing (storing) the value to the cell, reading stored values and clearing the cell, respectively. The memristive memory cell allows one to store $L = v^k$ different values, which implies the use of distinct resistor values ($R_1 \neq R_2 \neq R_3$). The parameter L refers to the number of possible distinct values that can be stored in the cell, v is the number of voltage levels that can be applied to change the memristance of M_1, M_2 and M_3, and k is the number of memory cell branches. For the selected memristor model introduced in [54], three voltage levels v can be applied to change the memristance: $1.2V$, $2V$ and $3V$. Fig. 4.8 illustrates the memory cell with 3 branches, which means that the number of possible voltage values L that can be stored in the cell is 27. Depending on the application and required memory capacity, the number of cell branches can be increased, which would lead to an increase in L.

To store (write) the value to the memory array, a set of voltages V_{w1}, V_{w2}, V_{w3} must be applied to each cell branch. Each combination of write voltages ($1.5V$, $2V$ and $3V$) corresponds to a particular voltage value that is stored. During the write cycle, the resistance values of the memristors are changed and preserved until the read cycle. In the write cycle, V_r and V_c are grounded. During the read cycle, the input voltage $V_r = 0.05V$ should be applied, while all V_w and V_c are set to 0. When the read voltage V_r is applied, the cell generates a particular V_o value corresponding to the stored value. To rewrite the data stored in the memory cell, the

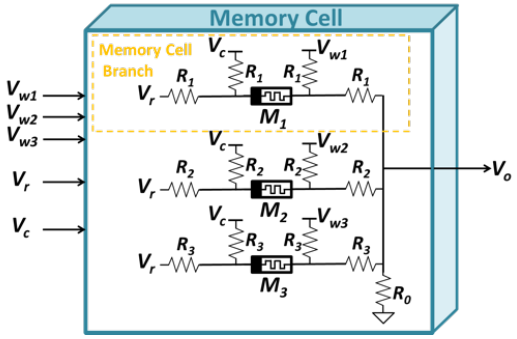


Figure 4.8: Proposed memristive memory cell consisting of three branches. V_{w1}, V_{w2}, and V_{w3} are input write voltages; V_r corresponds to the input read voltage; and V_c refers to the input clear voltage. V_o is the output produced during the read cycle. M_1, M_2, and M_3 are memristance values that change according to the applied voltage. R_1, R_2, and R_3 are the resistors values, where $R_1 \neq R_2 \neq R_3$.

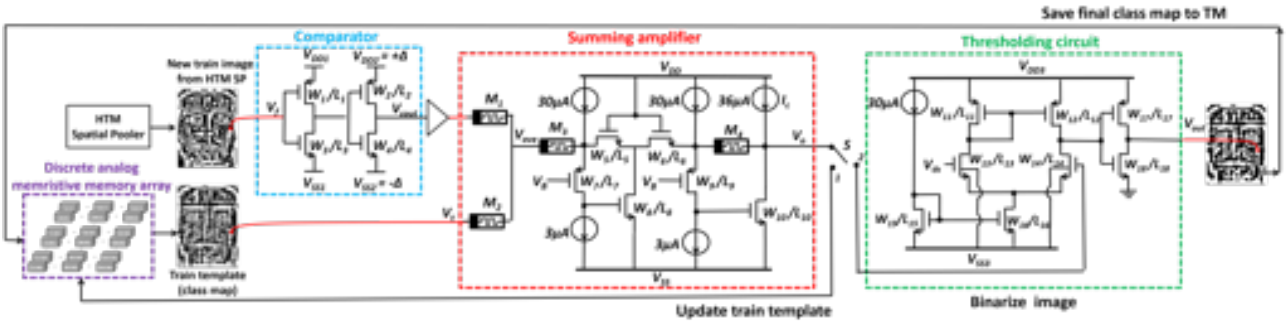


Figure 4.9: HTM TM configuration and memory update circuit for a single image class, consisting of a memory array, comparator, summing amplifier and thresholding circuit. The memristive memory array is used to store the temporary grayscale training class map. When the new training image arrives, the training template is updated using the comparator and the summing amplifier. When the training phase for the class is finished, the training class map is binarized using a thresholding circuit and stored in the same memory array.

clear operation is used. To clear the cell, the signal of $3V$ should be applied to V_c, while V_r and V_w-s are grounded.

The proposed memristive memory cells form a discrete analog memristive memory array to store the training image of a particular class, where each array cell represents a single image pixel. For c number of classes, c discrete analog memristive memory arrays are required. During the training mode, the memory arrays are updated. Figure 4.9 illustrates the circuit diagram of the proposed HTM TM and update operation. The HTM TM update circuit consists of comparator, summing amplifier and thresholding circuit. The first training binary image of each class is saved in a corresponding memory array. During the training mode, each memory

array is updated when a new training image of a corresponding class arrives. During the update process, the training image template stored in the memory array becomes grayscale due to the $\pm\Delta$ operation. When the training of a certain class is finished, the final training template is binarized and stored in a corresponding memory array again.

In the training mode, when a new training image from the HTM SP is directed to the HTM TM by the output data controller, each pixel of this image is processed by the comparator. The comparator circuit determines whether a training image pixel is black (with a voltage of $0V$) or white (with a voltage of $1.8V$). If the comparator input V_f is 0, the comparator output V_{cout} is $-\Delta$. If the input is 1, the output becomes $+\Delta$. The comparator consists of two CMOS inverters with PMOS and NMOS transistors. In the first inverter, $V_{DD1} = 1.8V$ and $V_{SS1} = -0.5V$. In the second inverter, $V_{DD2} = +\Delta$ and $V_{SS1} = -\Delta$ to ensure that the comparator output V_{cout} is $\pm\Delta$. For this work, the Δ value is selected as 0.05. For both inverters, the PMOS transistor ratio is $W_1/L_1 = W_2/L_2 = 0.72\mu/0.18\mu$, and the NMOS transistor ratio is $W_3/L_3 = W_4/L_4 = 0.36\mu/0.18\mu$. The second inverter of the comparator circuit is an underdrive inverter to ensure that the $\pm\Delta$ operation can be carried out.

The comparator output is applied to the summing amplifier. The CMOS-memristive summing amplifier consists of an averaging phase, corresponding to the memristors M_1 and M_2, and an amplification phase. The pixel value of the training class map V_t stored in TM and the comparator output V_{cout} are averaged by the memristive averaging circuit with $M_1 = M_2$ to obtain the value of V_{ave}, where $V_{ave} = (V_{cout} + V_t)/2$. The memristance values M_1 and M_2 are selected to be approximately 30 $k\Omega$ to eliminate the effect of the summing amplifier on the comparator output V_{cout}. The memristors M_1 and M_2 are preprogrammed to be 30 $k\Omega$ before the training phase. Then, the amplifier doubles V_{ave} to produce an output signal $V_o = V_{cout} + V_t$, which implies updating of the saved training value V_t by $\pm\Delta$. The modified amplifier configuration proposed in [56] is used in the amplification part. To double the amplifier input V_{ave}, the memristance value ratio $M_4/M_3 = 285k/100k$ is selected. The transistor ratios are $W_5/L_5 = W_6/L_6 = W_8/L_8 = W_9/L_9 = W_{10}/L_{10} = 2\mu/0.18\mu$ and $W_7/L_7 = 2.02\mu/0.18\mu$. The voltage values are $V_{DD} = 1.8V$, $V_{SS} = -1.8V$ and $V_B = -0.4V$. The value of V_{SS} is adjusted to compensate for the effect of the offset in the summing amplifier. In addition, the

current source I_c, which provides a current of 36 μA, can be adjusted to ensure a precise $\pm\Delta$ operation and to select the desired level of the output voltage V_o. The output value V_o is within the range from 0 to 1 V.

After the amplification stage, a new training pixel value is either used to update the memory array or is sent to the binarization circuit. If the training image is not the last image for this particular class, the switch S is kept in position 1, and the memory array is updated. During the update process, a memory cell in the array corresponding to this particular pixel is cleared, and the obtained new training pixel is stored (written) to this cell. The voltage level V_o is stored in the memory array using a specific writing circuit and is read from the memory array using the reading circuit. The voltage V_t that is stored in the memory array is within the range from 0 to 1 V. If the training image is the last image for this particular class, the switch is moved to position 2, and the obtained pixel value is binarized using the thresholding circuit. The thresholding circuit generates output $V_{out} = 1.8V$ if $V_o > V_{th}$ and $V_{out} = 0V$ otherwise, where V_{th} is the threshold voltage, selected as $0.8V$. The parameters of the thresholding circuit are $V_{DD3} = 1.8V$, $V_{SS3} = 0V$, $W_{11}/L_{11} = 3.24\mu/0.18\mu$, $W_{12}/L_{12} = 3\mu/0.18\mu$, $W_{13}/L_{13} = W_{14}/L_{14} = 2\mu/0.18\mu$, $W_{15}/L_{15} = W_{16}/L_{16} = 1.08\mu/0.18\mu$, $W_{17}/L_{17} = 0.72\mu/0.18\mu$, and $W_{18}/L_{18} = 0.36\mu/0.18\mu$. The final training pixel output value V_{out} is stored in the same TM array, and the training stage for this particular class is finished. The final class map preserved in the memory array is further used in the recognition (testing) stage. The values that are read from the memory array during the recognition stage are normalized to logic "high" (1.8 V) and logic "low" (0 V), which allows the direct pattern matching operation to be carried out.

Memristive pattern matcher

During the testing mode of operation of the proposed system, the memristive pattern matcher is used for comparison of the testing image with all class maps. The matcher circuit is based on XNOR threshold logic gates [57] and the averaging operation to produce the output, which demonstrates the weighted value of how many bits from the testing image are similar to the bits of the class map. Fig. 4.10 demonstrates the circuit diagram of the 2-bit XNOR pattern

matcher. As demonstrated in the diagram, the circuit-level implementation of the XNOR gate is based on the XOR gate, which in turn utilizes the memristor-CMOS implementation of the NOR gate.

The memristive NOR gate is constructed by two memristors with memristance values of M_1 and a third memristor having a memristance of M_{NOR} being placed in parallel, followed by the CMOS inverter. The memristance values M_1 and M_{NOR} are selected as high. The same configuration that is used for the NOR gate can be utilized for the NAND gate by controlling the voltage V_c. For a supply voltage level $V_{DD} = 1\ V$, the NOR gate implementation requires a voltage $V_c = 1$, while for the NAND gate, it should be $V_c = 0$. However, for the 180 nm TSMC CMOS process and a supply voltage $V_{DD} = 1.8\ V$, the voltage level V_c and memristor M_{XOR} should be accurately adjusted because of the high sensitivity of these logic gates to the changes in these two parameters. The voltage V_c is set to 0.61 V, and M_{XOR} is preprogrammed such that $M_{XOR} = 1.2\ M\Omega$.

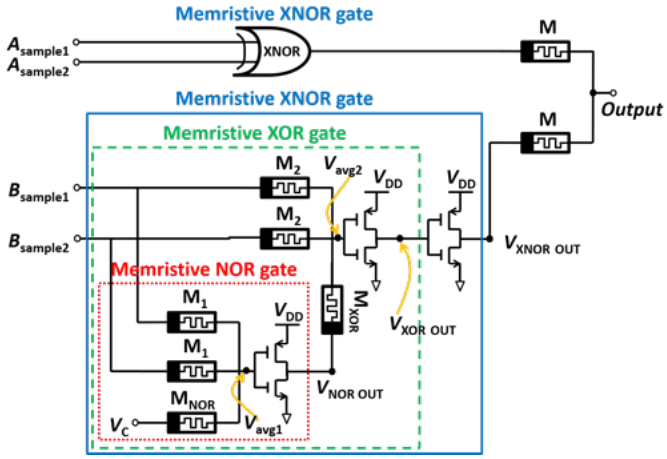


Figure 4.10: A 2-bit XNOR pattern matcher, created through the combination of memristive XNOR and memristive NOR configurations of memristive memory threshold logic.

The obtained structure of the NOR gate can be used to construct the XOR gate by adding three additional memristors and an inverter. The memristance value M_{XOR} depends on the output voltage of the NOR gate $V_{NOR\,OUT}$. For the high voltage value of $V_{NOR\,OUT}$, which occurs when the two inputs to the gate are low (say, $B_{sample1}$=$B_{sample2} = 0$), M_{XOR} should be taken as a low value. For all the other combinations of two gate inputs, which produce an output voltage of the NOR gate of $V_{NOR\,OUT} = 0$, the value of the memristance M_{XOR} is set as high.

Finally, an additional inverter is used to obtain the XNOR logic gate. All the three inverters are identical, with the voltage supply of $V_{DD} = 1.8V$. The parameters for the transistors of the XNOR pattern matcher circuit are as follows: $W_1/L_1 = W_3/L_3 = W_5/L_5 = 0.72\mu/0.18\mu$ and $W_2/L_2 = W_4/L_4 = W_6/L_6 = 0.36\mu/0.18\mu$ for PMOS and NMOS transistors, respectively. The memristors M_1,M_2 and M_{NOR} are set to $R_{off} = 2.5M\Omega$.

The outputs from the memristive XNOR gate are averaged using the set of memristors with the same high memristance value M. The final *Output* from the memristive pattern matcher circuit can be treated as a weighted similarity score. This is because the obtained value represents the number of testing bits that are similar to the bits in the class map divided by the number of bits. The averaging operation can be replaced by the summing operation, which would show the total number of similar bits; however, this would require significantly larger on-chip area and increased power dissipation. The class number can be found by fetching all pattern matcher outputs and sending them to the WTA circuit and then by obtaining the maximum XNOR output. The maximum XNOR output corresponds to the minimum difference between the images.

4.3.3 Verification Results of the New Revised Architecture

Architecture Performance Results

HTM SP simulation results. The simulations are performed using 0.18 CMOS TSMC technology and the 50 $nm \times 50$ nm titanium dioxide TiO_2 memristor models in SPICE and VerilogAMS [54]. The SPICE codes required to simulate large circuits are generated using MATLAB scripts. Fig. 4.11 presents the simulation results of the feature processing circuit for the inhibition block with M=4 and N=4, i.e., the inhibition block containing 4 receptor blocks with 4 inputs per block. Figures 4.11(a) and 4.11(b) present the pulse waveforms that were generated as the inputs to the receptor blocks in the sample SP configuration, where receptor block (RB) 1 and RB 2 have the same input signals and RB 3 and RB 4 have the same input signals. The same set of voltages was selected to show that the outputs of the receptor blocks

for the same inputs are slightly different, as shown in Fig. 4.11(d) and Fig. 4.11(e). This discrepancy is caused by the random weights of the memristive synapses. An example of the voltage distribution inside a receptor block is shown in Fig. 4.11(c). The inputs of a particular receptor block are averaged 10 times using the memristors, yielding 10 different outputs V_{RM_N}. These 10 outputs are sent to the receptor block mean to produce the final receptor block output V_{RBM}, which corresponds to 10 iterations to ensure a sparse random distribution of the input pattern. Even with the same set of input voltages, the separate bunches of random weight synapses (Fig. 4.7) can produce slightly different outputs after the mean operation due to randomization of the memristive synapses.

The average of the four receptor block output values is set as the threshold value, which is denoted as V_{AVG}. Because the difference between the outputs of the receptor blocks is not large and the inputs are selected symmetrically, the threshold V_{AVG} shown in Fig. 4.11(d) and Fig. 4.11(e) is the same for all four clock cycles. The comparator performs the comparison between the obtained threshold value and the receptor block outputs. The output from the comparator is inverted. As soon as the threshold value is applied to the positive input of the comparator, the inhibition block output is high (1.8 V) if the receptor block output V_{RBM} is higher than V_{AVG}, where $M = 1, 2, 3, 4$ in this example, and vice-versa. The output voltages of the inhibition block corresponding to the four receptor blocks are shown in Fig. 4.11(f).

HTM TM simulation results. Figure 4.12 illustrates the simulation results for the HTM TM circuit. The results of four clock cycles are presented. In the first three clock cycles, the train image is not assumed to be the last, and the switch S is set to position 1. The output of the summing amplifier V_o is stored in the memory array. The output from the previous clock cycle becomes the summing amplifier input V_t of the subsequent clock cycle. In the last clock cycle, we assume that the image is the last training image for the particular class, and the switch S is set to position 2. The amplifier output V_o is sent to the thresholding circuit, which produces the output V_{out} for the final training image pixel for this particular class.

Figure 4.12(a) shows the input to the comparator V_f from HTM SP, and Fig. 4.12(b) illustrates

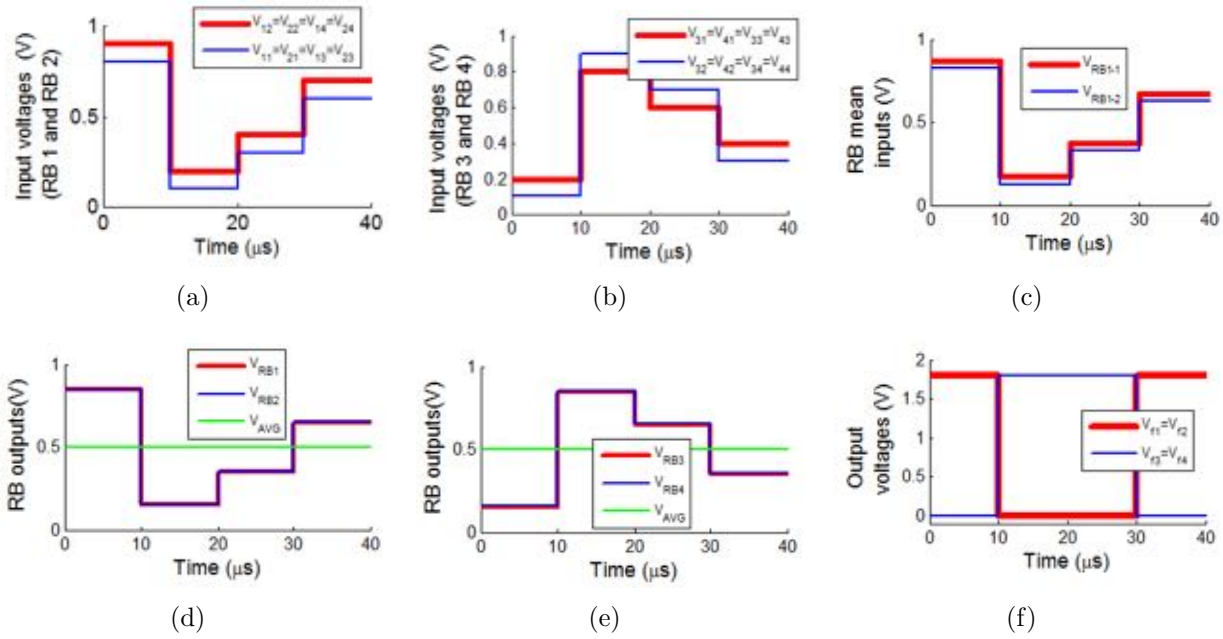


Figure 4.11: Timing diagram for HTM SP operation: (a) input voltages of receptor blocks RB 1 and RB 2, (b) input voltages of receptor blocks RB 3 and RB 4, (c) example of the outputs of the bunches of random weight synapses inside the receptor block, (d) outputs of RB 1 and RB 2 and threshold calculation V_{AVG}, (e) outputs of RB 3 and RB 4 and (f) HTM SP output pattern.

the input to the summing amplifier V_t from the training class map (discrete analog memristive memory array). Fig. 4.12(c) presents the comparator output V_{cout}, and Fig. 4.12(d) shows the output of the averaging stage of the amplifier V_{ave}. Fig. 4.12(e) presents the outputs of the summing amplifier V_o. Fig. 4.12(f) corresponds to the thresholding circuit output V_{out}. In the first clock cycle (from 0 μs to 10 μs), the input V_f from the new training image is 0 V, and the corresponding comparator output V_{cout} is approximately $-\Delta = -0.05$. The previously stored training class map value V_t is 0.8 V. The average value V_{ave} of V_t and V_{cout} is 0.35 V. The summing amplifier output V_o for this clock cycle is 0.75 V, which means that the initial value stored in the memory array $V_t = 0.8$ V was reduced by 50 mV. The output V_o is stored in the TM array and used as the input V_t of the second clock cycle. Even if the logic "high" output from HTM SP corresponds to 1.8 V, the logic "high" output produced by the summing amplifier and stored in the TM array is normalized between 0 V for logic "low" and 1 V for logic "high" to ensure the correct addition of the $\pm\Delta$ value. The thresholding circuit is turned off for the first clock cycle.

For the second clock cycle (from 10 μs to 20 μs), the pixel value of the new training image V_f

is 1.8 V, corresponding to the comparator output $V_{cout} = +\Delta = +0.05$. The averaging circuit output $V_{ave} = 0.37\ V$, and the amplifier output $V_o = 0.794\ V$, which are stored in the TM array. In the last clock cycle, the switch is set to position 2, which activates the threshold circuit. In the last clock cycle, the summing amplifier output is 0.789 V, and the thresholded output $V_{out} = 1.8\ V$. This output is stored in the TM array and used in the recognition stage.

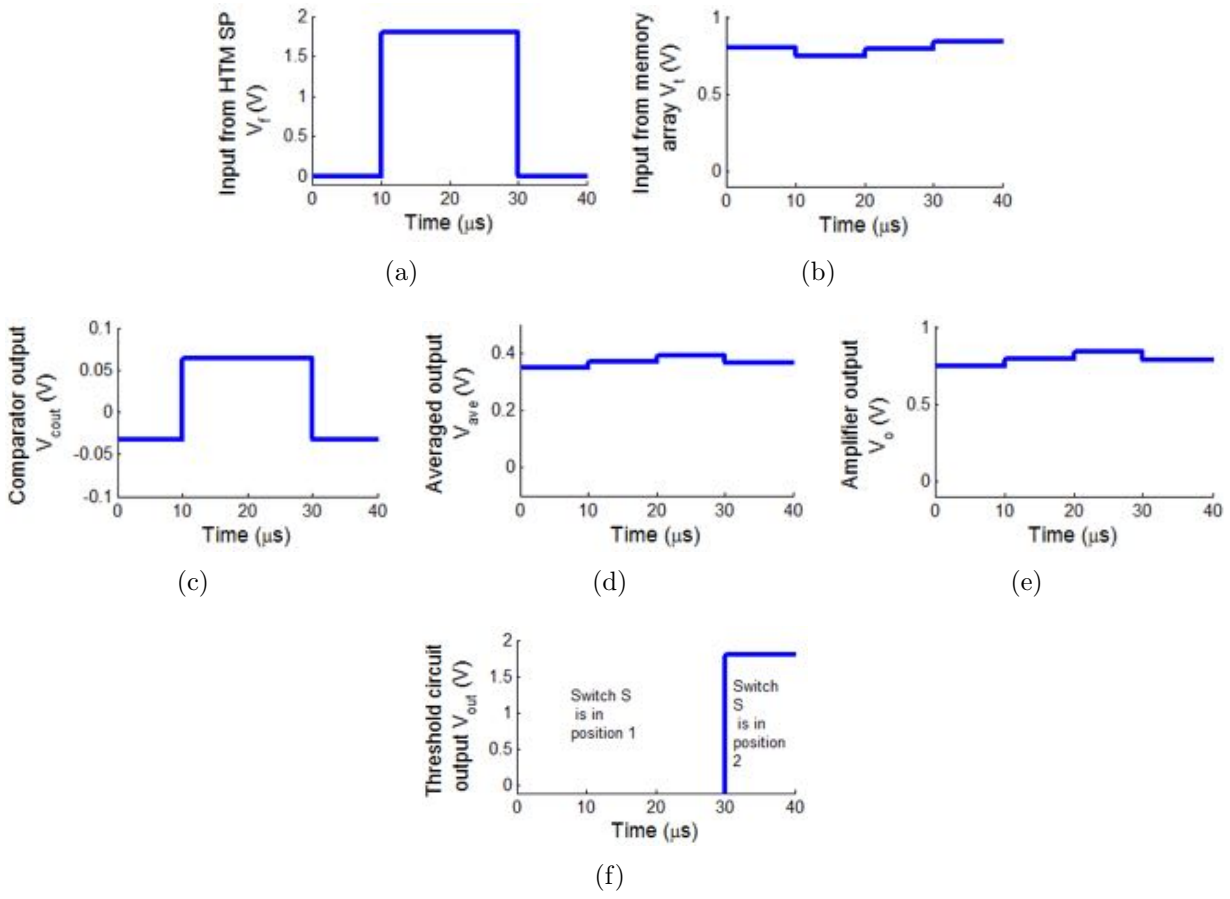


Figure 4.12: Simulation results of the HTM TM update circuit, which consists of four clock cycles. In the first three clock cycles ($0 - 30\ \mu s$), the switch S is in position 1, whereas in the last clock cycle ($30 - 40\ \mu s$) the switch S is in position 2. The input signal V_f refers to the new training image pixel. V_t is the value of the class map currently stored in the memory array. V_{cout} is the comparator output, V_{ave} is the output of the averaging stage of the amplifier, V_o corresponds to the amplifier output, and V_{out} is the output of the thresholding circuit corresponding to the binarized final training class map.

Pattern matcher simulation results. Figure 4.13 shows the simulation results for the pattern matcher (Fig. 4.10). Fig. 4.13(a) and Fig. 4.13(b) illustrate two inputs to the XNOR gates. The four clock cycles represent four different combinations of logic "high" (1.8 V) and logic "low" (0 V) values. Fig. 4.13(c) presents the average value V_{avg1} for the memristive

NOR gate. Fig. 4.13(d) presents the output of the memristive XNOR gate V_{NOROUT}. In the ideal NOR gate output, the first three clock cycles must be equal to 0. However, because the circuit was adjusted to make it compatible with the 180 nm CMOS technology and the nominal V_{DD} was set to 1.8 V, the NOR gate output is not precise, which is not critical in this circuit configuration. Fig. 4.13(e) presents the average voltage V_{avg2} for the memristive XOR gate and the inverter threshold. To obtain nominal values for the 180 nm technology, the inverter threshold was increased to approximately 0.8 V. Fig. 4.13(f) shows the XOR gate and XNOR gate outputs. The final XNOR output is "high" for the same inputs and "low" for different inputs. Finally, the outputs of the pattern matchers are averaged and sent to the WTA. Therefore, if the input pixels of the pattern matcher are the same, the average output value corresponding to a particular class increases.

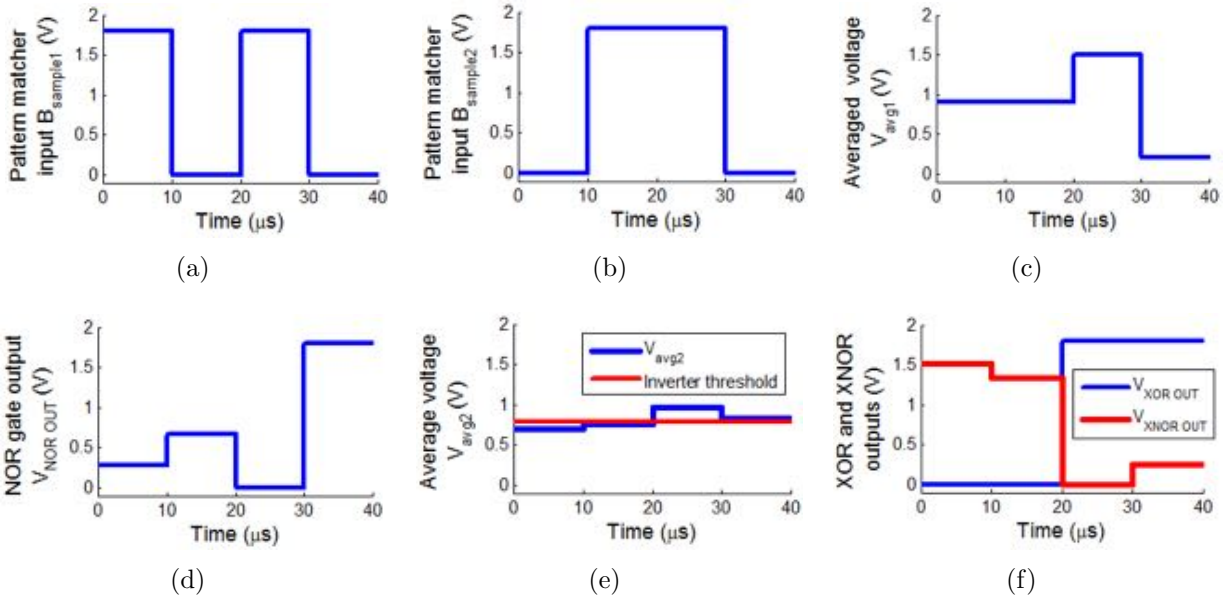


Figure 4.13: Memristive pattern matcher timing diagram: (a) pixel 1 input value $B_{sample1}$, (b) pixel 2 input value $B_{sample2}$, (c) average voltage V_{avg1}, (d) NOR gate output V_{NOROUT}, (e) average voltage V_{avg2} and inverter threshold, and (f) XOR and XNOR outputs.

Power and area calculations Table 4.1 shows a summary of the area and power calculations for the three circuits that were demonstrated in this paper: SP, TM and memristive pattern matcher. The values in Table 4.1 are represented based on consideration of the minimum on-chip area and the worst-case scenario for power dissipation. The maximum amount of power is dissipated when the inputs to the circuit are supplied by the voltage sources with the maximum

Table 4.1: Area and power calculation for the proposed modified HTM design

Configuration		Area (μm^2)	Maximum Power (μW)
HTM SP			
M = 4	N = n×n = 1×1 = 1	19.96	365.88
	N = n×n = 2×2 = 4	20.26	365.88
	N = n×n = 3×3 = 9	20.76	365.88
M = 9	N = 1	44.9	823.23
	N = 4	45.58	823.23
	N = 9	46.70	823.23
M = 16	N = 1	79.83	1463.52
	N = 4	81.03	1463.52
	N = 9	83.03	1463.52
M = 25	N = 1	124.73	2286.75
	N = 4	126.60	2286.75
	N = 9	129.73	2286.75
HTM TM			
$A = 1$, $B = 1$ (for single pixel)		23.85	442.26
Memristive pattern matcher			
2 bit matcher		1.18	69.44

value, which is 1 for the applications discussed in this paper, and all random resistance values of the memristors are set to R_{on}.

The area and power for the SP implementation were calculated for different configurations, i.e., different sizes of the receptor and inhibition blocks. A closer look at the area and power calculations for a particular value of M reveals that values do not change significantly with increasing $n \times n$ size. More precisely, the values of the maximum dissipated power do not change for a given M. This is because an increase in N increases the number of synapses within a single column, which are in turn represented by the memristive devices, which are very compact and do not dissipate significant amounts of power. In contrast, an increase in M results in more columns. A single column require a comparator and an inverter for the processing. Thus, the threshold comparison operation significantly affects on-chip area and power requirements.

Table 4.1 also shows that the pattern matching could be implemented by the compact CMOS-memristor-based circuitry at low power consumption.

Selection of the optimal parameters. In addition to these values, a two-variable analysis was performed to include recognition accuracy as an additional criterion in the selection of optimal values for N and M. This was achieved by performing face recognition analysis on the AR face database [58] with the initial assumption that the optimal delta for TM should be as small as $\Delta = \pm 0.05$. Figure 4.14 illustrates the relation between the n and M circuit parameters and the recognition accuracy results for input images having fixed dimensions of 120×160 bits. As can be seen, the best recognition accuracy for the AR database is achieved with the small values of the n and M parameters. This is because when the inhibition region dimensions increase, the number of bits being suppressed by the inhibition block increases as well. This means that small values for n and m must be selected to increase the number of regions within which decisions are made and to ensure that there is no loss of important features.

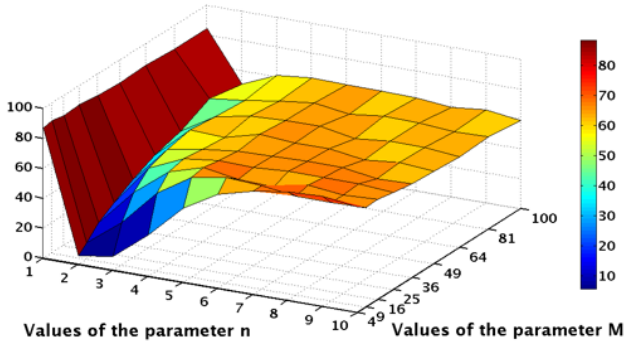


Figure 4.14: Two-variable analysis performed to include recognition accuracy as an additional criterion in the selection of optimal values for the $N = n \times n$ and $M = m \times m$ circuit parameters

Considering the results illustrated in Table 4.1, for a fixed input of $120 \times 160 = 19200$ bits, the optimal circuit parameters were selected to be $N = n \times n = 1$ and $M = m \times m = 4$, for a total of $P = 4800$ processing blocks, a single unit of which is illustrated in Fig. 4.5. Thus, for the proposed design, the SP weights are not trained. Rather, learning is initially achieved via standard filtering; subsequently, mainly within the TM part, the optimal performance is achieved when either all or none of the potentials of the SP part are activated simultaneously.

Figure 4.15 illustrates the analysis that was performed to determine the optimal delta parameter required for TM. This was achieved with the SP having the fixed parameters listed above and performing face recognition for different values of training images. It can be observed that as the number of training images increases, the best recognition accuracy is achieved $\pm\Delta$ with a value of less than or equal to ± 0.1. Hence, the optimal delta value was selected as $\Delta = \pm 0.05$.

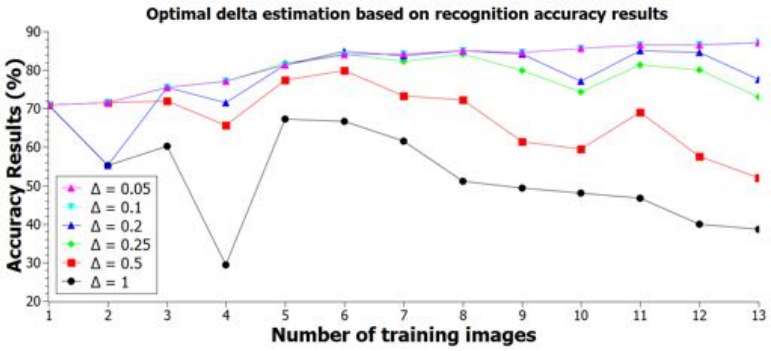


Figure 4.15: Optimal Delta estimation based on recognition accuracy results, with the SP having fixed circuit parameters of $N = 1$, $M = 4$, and $P = 4800$.

End-to-end Evaluations for Applications

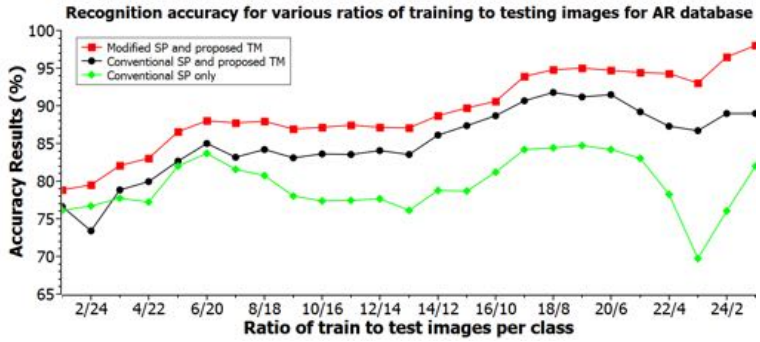
Face Recognition. The algorithm was tested using two human face databases: AR and ORL. The AR database is the largest database having 100 classes of images, meaning that the faces of 100 different people are taken. There are 26 face images per person with different facial expressions, emotions and occlusions such as light, scarves and eye glasses [58]. The other tested database is the ORL database, which includes 40 classes of images containing 10 different images [59]. The database contains different facial expressions and occlusion details in addition to rotated faces up to 20 degrees and 10 percent scale variations [60]. This enables the evaluation of the impact of facial angle and scale changes on face recognition accuracy and on the efficiency of the proposed algorithm. The overall simulation of the system for performance analysis is carried out in MATLAB by utilizing the results of circuit level simulations using the SPICE tool. In all our experiments, the dataset is divided into two distinct sets: training images and testing images. In addition, no overlap is allowed between these sets.

Figure 4.16(a) illustrates the recognition accuracy results for the AR face database for various ratios of training-to-testing images, thereby comparing three different architectures. The first

architecture is based only on the conventional HTM SP presented in [16]. The second architecture is the conventional HTM SP [16] along with the analog HTM TM presented in this paper. The third architecture is the proposed modified HTM SP with the proposed analog HTM TM.

The effectiveness of including the proposed analog TM can be observed from the more accurate results achieved by the two architectures utilizing it in contrast to the pure conventional HTM SP architecture. Next, a comparison of the accuracy results achieved by the modified HTM SP with those achieved by the two other architectures emphasizes the advantage of changing the decision criterion within the inhibition region.

Similarly, Figure 4.16(b) illustrates the same pattern when the same circuit parameters are used for face recognition on the ORL face database for various ratios of training-to-testing images.



(a)

(b)

Figure 4.16: Recognition accuracy results achieved using three different architectures for various ratios of training images to testing images for the (a) AR face database and the (b) ORL face database.

Table 4.2 illustrates the results when classifying test images of the AR database into one of the four categories under the three different architectures. The setup divided the AR database images into 13 training and 13 testing images for each class. For architectures utilizing the

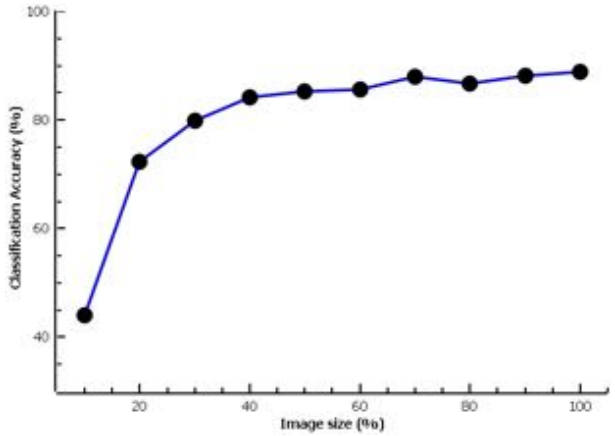


Figure 4.17: The impact of image resolution or generally scaling on the classification accuracy. The 100% image size in x-axis corresponds to 120 × 160 pixels.

proposed TM, training images were used to create a single class map, which was then used to determine the class of the testing images. For the architecture that is based only on the conventional HTM SP proposed in [16], the training images are initially averaged and only then fetched to the SP to produce a single training template, which was then used to determine the class of the testing images. This setup is implemented to enable fair comparison of the three architectures.

It can be observed that the addition of the proposed analog TM increases the capability of the system to differentiate different categories by almost 7%. Moreover, the change in the decision criterion within the inhibition region, which is the replacement of the conventional SP circuitry by the modified SP circuitry, results in an additional increase in the accuracy of approximately 4%.

Figure 4.17 shows the impact of image scaling on the recognition accuracy using the proposed HTM architecture. It can be observed that with increase in resolution there is an increase in the accuracy for the optimal set of parameters.

Speech Recognition. The proposed system was also verified on a speech recognition application. The TIMIT database [61] was used to estimate the capability of the proposed system to recognize quickly varying temporal patterns. Although the database consists of a vast number of unique words, few of them, mostly articles and prepositions, are repeated enough to perform

Table 4.2: Results of classification of the test images into each category of the AR database under the three different architectures using a single template or a class map for each class, consisting of 13 training images and 13 testing images

Architecture	Emotions	Light conditions	Occlusions (glasses)	Occlusions (scarf)	**Total**
Conventional HTM SP [16]	77.50%	91.00%	84.33%	53.33%	**76.54%**
Conventional HTM SP [16] with the proposed TM	84.25%	96.33%	85.67%	67.67%	**83.48%**
Proposed (Modified) HTM SP with the proposed TM	85.50%	97.67%	91.33%	74.33%	**87.21%**

training of the system. Hence, only two samples, which are usually used for speaker recognition, are used in this analysis as two separate classes. Two classes are created by combining instances of the *sa1* sample and instances of the *sa2* sample, which are given as

sa1: *She had your dark suit in greasy wash water all year.*

sa2: *Don't ask me to carry an oily rag like that.*

Then, the training set is created by combining 50 instances, and the testing set is created by combining 15 instances of each class, providing a total of 130 instances to process by the system.

Because the proposed system is constructed to process input data in the form of an image, speech waveforms are initially preprocessed using the Perceptual Linear Prediction (PLP) feature extraction method. Images representing $12th$-order PLP features of sample waveforms without RASTA filtering were obtained according to the procedure and codes described in [62]. As a result, speech samples were converted into images without significant degradation of temporal details. These images were then fetched to the proposed system to perform a recognition procedure similar to that used to perform face recognition.

Figure 4.18 illustrates the recognition accuracy results for speech recognition with initial waveforms having either no additive white Gaussian noise or AWGN with SNR being equal to 20, 10, or 5 dB. It should be noted that because the PLP feature extraction method produces output images having dimensions of 420×560 bits, which is much larger than the dimensions of the

AR images, the M parameter was adjusted accordingly to $M = 49$ receptor blocks within a single inhibition region. With the intent to increase the dimensions within which the decision rule of the Modified SP is performed, this results in an increase in recognition accuracies and in noise robustness.

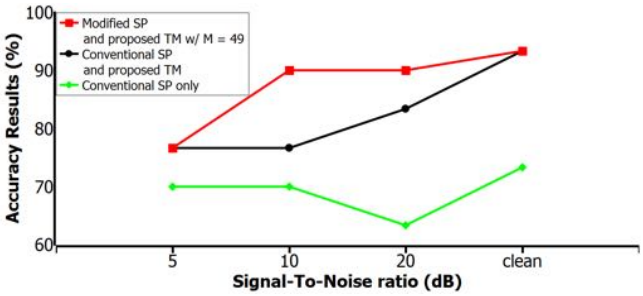


Figure 4.18: Speech recognition accuracies obtained under the three different architectures for various SNRs using the PLP feature extraction method as preprocessing

4.3.4 Discussion and Comparison

Result evaluation

In this work, we presented the circuit-level design of a HTM system for pattern recognition that incorporates the optimized architectures of the HTM SP and HTM TM. The simulation results on the optimum values of the M and N parameters for the processing of P blocks for fixed image dimensions revealed that the highest accuracy is achieved with the lowest M and N values. Thus, for image processing on images with dimensions of 120x160, the SP circuit will consist of $P = 4800$ processing blocks, each having $M = 4$ columns with $N = 1$ synaptic connection per column. As a result, using Table 1, the total on-chip area and power consumption for the implementation of the proposed SP circuit for image processing are 0.096 mm^2 and 1756 mW, respectively. Compared to the previous works [16, 53], one of the major advantages of the proposed HTM architecture is the scalability of the design. Because the previous implementations of the HTM SP were based on the crossbar architecture, the sneak path problem limited the application of HTM to small-scale tasks. However, the use of memristors and averaging circuits in the proposed HTM design ensures the scalability of the

architecture and the application of HTM to large-scale problems. Consideration of the recent publications on the topic of HTM proves the feasibility of the proposed design.

One of the recent designs of the SP was presented in our earlier work [16]. The design [16] is based on parallel memristive crossbar arrays and presents the implementation of the conventional SP algorithm. In the current work, we demonstrated the superiority of the modified HTM over the conventional HTM in terms of achieved recognition and classification accuracy results. In addition to the improved accuracy results that were demonstrated in Section V, the proposed design of the SP circuit offers a reduction in the time required for the processing of an image. This is because the crossbar implementation requires 3 cycles for operation. The first two cycles are related to the write operation of the memristive crossbar. The two-step write technique is applied to write the high and low values on the memristive devices, which will represent strong and weak synaptic connections. The third cycle is referred to as the read operation, during which the connectedness of the synapse is checked. In the research work [16], the best simulation results were demonstrated for a crossbar design that is based on the 6x6 inhibition block. This implies that for the processing of an input image having $120 \times 160 = 19200$ bits, the memristor-crossbar architecture for the SP [16] will be composed of 4 parallel crossbar slices each having 9 synapses (9 rows) within each of 533 serial columns. The sneak path leakage current issue associated with the crossbar structure does not allow for operations within each of the serial columns to be performed in parallel. As a result, considering the switching speed of memristive devices of 10 ns as well as 3 cycles for the operation of the proposed architecture, the minimum time required for the processing of an image will be $3\,cycles \times 10ns \times 533\,serial\,columns = 15990ns \approx 16\mu s$. In contrast, the design proposed in this work allows the processing to be performed in parallel. The delay in the circuit is attributed to the memristor switching time of 10 ns and the amplifier and inverter response times of approximately 2 ns. Thus, the response time of the circuitry will be significantly lower.

Another hardware design of an SP was presented by Streat et al. [45]. The proposed non-volatile architecture [45] was implemented in the VHDL and demonstrated a high level of classification accuracy (91.89%). This design showed requirements in terms of area footprint of $104.26mm^2$ and power consumption for an 8-channel model of the SP of $64.394mW$. The

significant reduction in the on-chip area requirement for the implementation of the proposed design can be noticed and is due to the use of nano-scale memristive devices and 180-nm TSMC CMOS technology. However, the use of amplifiers introduces a significant portion of the power dissipation. This problem can be solved by designing a better circuit for the comparison operation. In addition, the increase in the number of synaptic connections N will reduce the total number of amplifiers required for processing and consequently will reduce the power consumption. However, this might lead to a reduction in recognition accuracy because, as mentioned earlier, a large N will reduce the number of regions in which the decision is made upon the importance of the features.

Advantages of the Memristive Implementation of HTM

Comparing to the digital implementations of the HTM algorithm, the advantages of the analog memristive HTM implementation are related to the processing speed, small on-chip area and adjustable power dissipation. The full hardware implementation of HTM would allow to process visual data faster, in contrast to the digital implementations that are limited by the processing frequency of FPGAs and the sampling rate of analog-to-digital and digital-to-analog converters.

The memristive devices used in the HTM design for various components, such as amplifiers, pattern matcher and averaging circuits, ensure the reduction of the on-chip area of the HTM, in comparison to the implementation that are based on resistors or fixed digital FPGAs, VHDL and VLSI implementations. Also, if the digital HTM implementations have fixed area and power dissipation, the analog circuit for HTM can be adjusted. For example, the high power components, such as operational amplifiers, can be replaces by the circuit components with lower power dissipation.

Limitations of Memristive HTM Hardware Implementation

The HTM SP algorithm is simplified to implement it using analog hardware. Such parameters as more than 1 active columns after overlap calculation, update of the boosting factor, complete

Hebbian-based learning have not been implemented yet. In recently proposed studies [42, 17, 16, 63], the number of winning columns in the overlap phase of HTM SP is limited to 1 because of the overlap parameter is selected using the traditional WTA circuits. However, in the original algorithm the desired number of winning columns can be selected. To improve this limitation, the implementation of the HTM SP with adjustable number of outputs in the WTA circuit is required.

In addition, in most of the implementations the boost factor parameter is not taken into account or represented as 1. This is appropriate for the visual data processing applications [42, 17, 16, 63]. However, the original HTM algorithm implies that the boosting factor should be updated [18]. The analog hardware implementation of the boosting factor update may open the possibility of the successful application of HTM not only for the pattern recognition and visual data processing, but also the other applications, such as prediction making.

The existing hardware implementation of the HTM TM proposed in [17] is not scalable. The HTM TM circuit is presented for a single pixel implementation. The real time implementation of such hardware circuits involves the consideration of the trade-offs between the parallel and sequential processing of the input data. For the parallel processing of the HTM TM circuits, the on-chip area and power is large and it is difficult to scale the architecture. However, the processing speed of such circuit is high, comparing to the implementation of the equivalent digital circuits or sequential processing of the input data using the same HTM TM design. The sequential processing requires the additional sequence control circuit.

Open Problems

One of the most important problems in the hardware implementation of HTM is the continuous hardware training, learning and parameter update process. The research study [18] states that the boosting factor in the HTM SP should be updated based on the average-activity of the mini-columns considering T previous inputs. The appropriate number of T is selected to be 1000, which means that 1000 update cycles in analog hardware is required. Considering that the learning and update process in [17] is performed using analog memristive memories [43], the real

on-chip implementation of the stable multilevel memristive memory with the long lifetime is one of the challenges in the analog hardware implementation of the memristive HTM. Therefore, the robust analog hardware implementation for online learning and training and of HTM is required. The fully analog learning circuit for the HTM will remove the necessity of the offline software-based or FPGA-based training process, which would improve the processing speed. Also, the implementation of the efficient and robust analog memory unit is necessary to achieve the maximum processing speed and efficient temporal data storage. The implementation of the on-chip analog memory unit for the memristive HTM will remove the necessity to have the separate storage unit, which will improve the processing speed and reduce required on-chip areas and power consumption.

To solve the analog HTM scalability problems, the analysis of the parasitics and leakage currents of the large memristive circuits is required. The scalability of the memristive crossbar array for analog HTM design should be tested and the limitations of the number and side of the HTM input should be identified. Also, the performance of the memristive HTM SP and TM in terms of recognition and classification accuracy should be tested on the real memristive crossbars considering influence of the parasitic effects and leakage currents on the HTM performance.

The realistic implementation of memristive HTM circuits required at least the involvement of the large scale circuits containing the devices that induce the memristive behavior, such as resistive switching memories. However, all existing memristive devices have a limitations in terms of the number of possible memristive states. For example, the analog implementations of the memristive pattern matcher and low voltage amplifier involving the memristors that replace the resistors in the circuits [17] are practically very difficult to implement with the existing memristive devices. As the implementation of these circuits require the memristors to be programmed to exact level of resistance, the implementation of such circuits with the existing two state memristive devices is complex. One of the possibility is to use the set of memristive devices with R_{on} and R_{off} states connected in parallel and in series. However, the real behavior of such configuration should be tested considering leakage currents and parasitics, because the parallel and series connection of the memristors is different than in the resistors. The current flow in memristors should be considered. Memristive analog HTM implementation

can involve the resistive switching memories that have been used for the implementations of the neural networks [64], however the HTM behavior will not be accurate due to the limited number of memristive states. The maximum number of states that can be achieved by memristor is 16 for GST memristors [65]. However, the area of such memristors is larger and GST memristor technology is not always compatible with the CMOS technology.

In addition, the real memristive devices are not always accurate and involve the problem of switching stochasticity. Two real memristors of the same technology can react differently to the same voltage applied across the memristor for the switching [66]. The switching time and level may vary or, according to the theories of probabilistic behavior of the memristor, the switching may not occur. Such behavior of memristive devices may effect the performance accuracy of the HTM SP and HTM TM significantly. For example, in the HTM TM more training images and training cycles may require to achieve the same accuracy and system level HTM performance due to these variabilities in the behavior of the memristor.

4.3.5 Summary

This work presented several novelties in the area of a brain-inspired machine learning algorithm known as HTM. We proposed a simplified algorithm for SP realization based on averaging operations as well as its possible implementation as the memristor-CMOS circuit. In addition, we reconsidered the concepts of TM and demonstrated an analog memristive memory array for its hardware implementation. We discussed both algorithms and circuit implementations in detail and demonstrated the accuracy and efficiency of the proposed methods for face and speech recognition applications. We achieved an average accuracy of 87.21% for face recognition and 92% for speech recognition. To process images with dimensions of 120×160, the calculated area and power requirements for the proposed HTM SP design implementation are 0.096 mm^2 and 1756 mW, respectively. The proposed HTM TM circuit design for a single pixel requires 23.85 μm^2 of area and 0.442 mW of power.

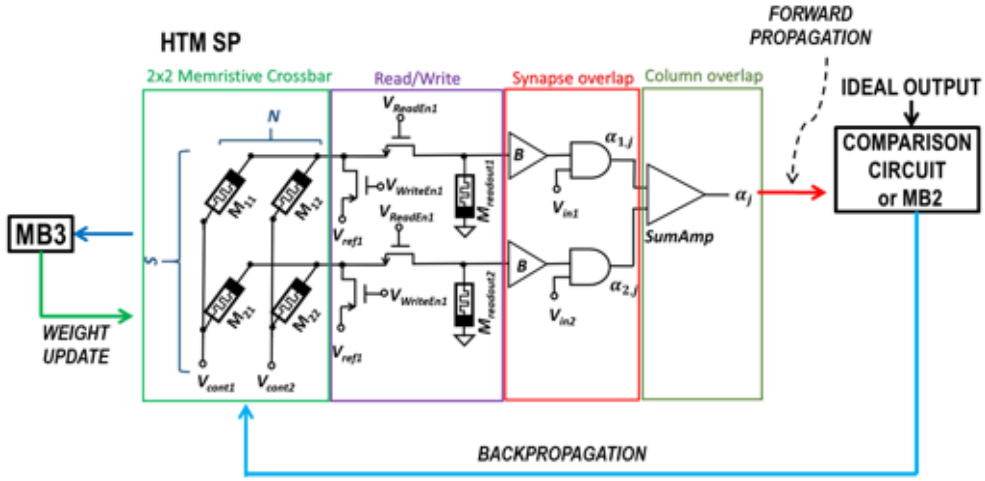
4.4 HTM with backpropagation learning

The other learning architecture, where the proposed circuit can be used is HTM. There are several hardware implementations proposed for the HTM SP and HTM TM [63, 16, 17], however the learning process in the HTM SP and the learning and update process in the HTM TM are limited. In addition, the learning stage of the HTM SP has not been implemented on hardware yet. This stage can include either update process based on Hebb's rule or the backpropagation update of the HTM SP weights [18].

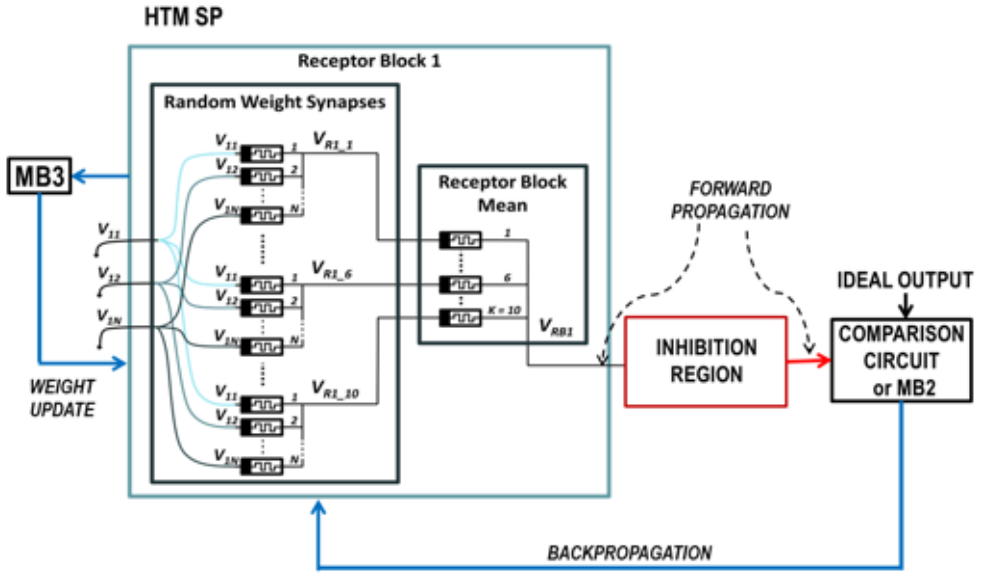
There are two main analog architecture for the HTM SP: conventional HTM SP [16] and modified HTM SP [17]. The application of the proposed learning circuits for both of the architectures is shown in Fig.4.19. Fig.4.19(a) illustrates the application of the proposed learning architecture for the conventional HTM circuits. After the forward propagation through the HTM SP, the HTM SP output is compared to the ideal HTM output. In the conventional HTM SP circuit, the calculated error from the comparison circuit or MB2 is fetched back to the memristive crossbar to calculate the error in the weights, and the weights are updated. Fig.4.19(b) shows the application of the proposed circuits for the modified HTM SP architecture. The conventional HTM SP architecture consists of the receptor and inhibition blocks. The weights of the synapses are located in the receptor block. After the comparison of the HTM SP output to the ideal output, the error is propagated back through the receptor block and MB3, and the memristive weights are updated.

4.5 Feature extraction without learning in the HTM SP

Due to the ability to encode of the input data and produce SDRs, the HTM shows a good performance even without the learning stage for particular applications. In this work, we investigate the initialization stage of the HTM SP and proposed the rule-based deterministic approach instead of the random weight approach for the initial weight assignment. The main purpose of the rule-based approach is to connect the input to the HTM weights, which allows to preserve natural sparsity and structural information from the inputs. Moreover, we propose



(a)



(b)

Figure 4.19: Analog hardware implementation of the (a) conventional HTM SP algorithm and (b) modified HTM SP algorithm with backpropagation learning stage.

the hardware implementation for the rule-based approach and compare it with the conventional random weight approach in terms of power dissipation and on-chip area requirements. Also, we test the system level implementation of the proposed approach on the face recognition problem and show the improvements in the recognition accuracy [43, 17].

Algorithm of the rule-based HTM SP

To improve the initialization phase of the HTM SP, we proposed the rule-based approach for the weights assignment instead of the uniform weight distribution. In the rule-based approach, we establish the connection between the input space and the synaptic permanence values (weights of the synapses). The Eq. 4.14 shows how synaptic permanence weights are assigned in the rule based approach. Eq. 4.14 is used instead of Eq. 4.3 and Eq. 4.2.

$$\mathbf{S}_{ij} = \begin{cases} 1 & \text{if} \quad j \in PI(i) \text{ and } PI(i) > mean(PI) \\ 0 & \text{otherwise} \end{cases} \tag{4.14}$$

In the rule-based approach, the synaptic permanence value is assigned based on the mean value of the inputs within the input space region with the potential connections. If the input is greater than the mean of the the inputs within this neighborhood, the synaptic permanence $\mathbf{S}_{ij}$ is 1, otherwise $\mathbf{S}_{ij} = 0$.

In this work, we focus on the first three phases of the HTM SP: initialization, overlap and inhibition. The Algorithm 2 summaries the proposed approach. Lines 2-18 represent the HTM SP initialization stage, lines 20-22 refer to the overlap stage, and lines 24-27 correspond to the inhibition stage of the HTM SP.

4.5.1 Hardware implementation of the rule-based HTM SP

In this work, we proposed the analog hardware implementation of the rule-based approach for the HTM SP. If the tradition hardware implementation of the HTM SP is based on the

Algorithm 2 The HTM SP algorithm

▷ HTM SP initialization

Define the size of input neighborhood with potential connections, x_i^c, γ, ρ, η, θ_c, size of the local inhibition region, θ_s

Determine ϕ by multiplying the average number of connected input spans of all the SP mini-columns by the number of mini-columns per inputs.

$z_{ij} \sim U(0,1)$

if $\forall x_j \in (x_i^c, \gamma)$ **then**

    $\imath(x_j; x_i^c, \gamma) = 1$

for $\imath(x_j; x_i^c, \gamma)$ and $(z_{ij} < \rho)$ **do**

    $PI(i) = j$

if $j \in PI(i)$ and $PI(i) > mean(PI)$ **then**

    $\mathbf{S}_{ij} = 1$

else

    $\mathbf{S}_{ij} = 0$

$\mathbf{B}_{ij} = \mathbf{S}_{ij}$

for $|y_i - y_j| < \phi, i \neq j$ **do**

    $\mathbf{N}_i = j$

▷ HTM SP oveplap

$o_i = \beta_i \mathbf{B}_{ij} \mathbf{Z}_j$

for $j \in \mathbf{N}(i)$ **do**

    $\mathbf{NO}(i) = o_j$

▷ HTM SP inhibition

if $(o_i \geq \text{prctile}(\mathbf{NO}(i), 1-s))$ and $(o_i \geq \theta_s)$ **then**

    $\alpha_i = 1$

else

    $\alpha_i = 0$

memristive circuits, the rule based approach is based on the CMOS circuits. The proposed receptor block is shown in Fig.4.20.

In the proposed architecture, the memristive mean calculation block and CMOS comparator circuit correspond to the initialization phase on the rule-based HTM SP approach. The memristive mean calculation block calculates the average of the inputs in the mean of the inputs from the set of the potential inputs. The voltage V_{mean} refers to the threshold for assigning the potential inputs as connected or disconnected. The CMOS comparator circuit compares the input of the particular column with the mean of all columns from the potential inputs. If the input is greater than the threshold, the output of the comparator V_{comp} is low and vice versa. The CMOS analog switch block refers to the implementation of the overlap stage of the HTM SP. The voltage $V_{out} = V_{in1}$, if the comparator output V_{comp} is low, which corresponds to the

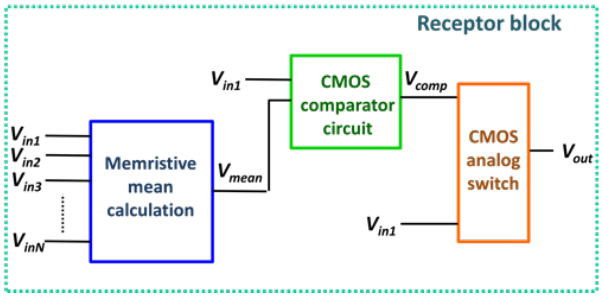


Figure 4.20: The HTM SP receptor block structure for the rule-based approach.

case when the column is connected. The voltage $V_{out} = 0$, when V_{comp} is high, which means that the columns is disconnected. The voltage V_{out} refers to the overlap value of the column.

4.5.2 System level implementation

In this work, we apply the HTM SP with two different initialization stage approach for the face recognition problem. The overall system implementation of the face recognition module with the HTM SP is illustrated in Fig.4.21. The input RGB images are read by the image sensor and applied to the input data controller. In this stage, the sampling process occurs if it is required and the sampled images are preprocessed. In this method, we use only RGB to gray-scale conversion as a preprocessing step. In the existed HTM SP face and speech recognition systems [17], the standard deviation filter is applied in the preprocessing stage to improve the recognition process. However, in this work, we show the effect of the different approaches for the initialization stage; therefore, we remove the filtering stage to obtain the actual results from the HTM SP.

After the controller, the image is applied to the HTM SP stage, which performs the encoding of the image and outputs the sparse binary image with the preserved important image features. The output data controller controls, where the images are directed in the training and testing stages. In the training stage, the output from the HTM SP is preserved in the training template storage. The training continues until all image class templates are preserved. In the testing stage, the output data controller directs the images into the comparison circuit. The comparison circuit can be implemented as a memristive pattern matcher, which compares all templates

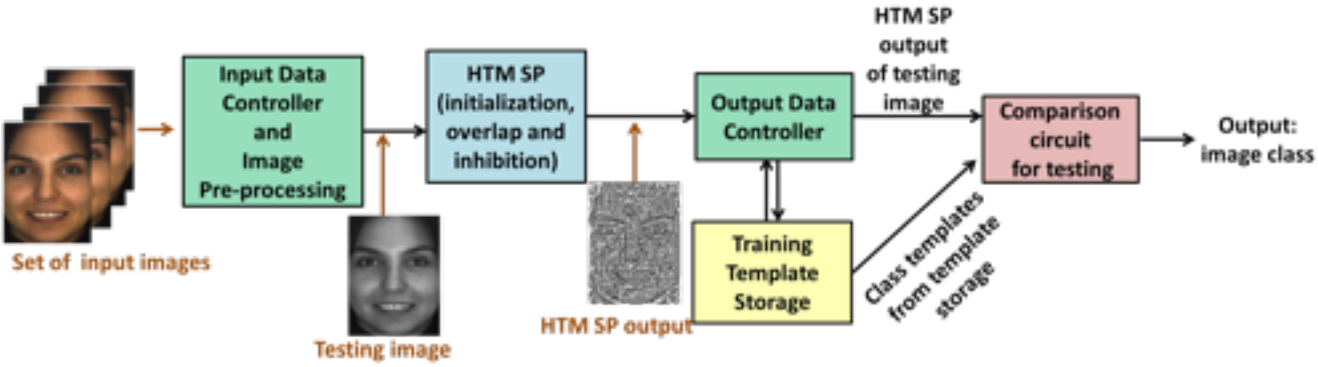


Figure 4.21: The overall system implementation of the face recognition module with the HTM SP.

form the training template storage with the current input image. Finally, the image class is determined.

The algorithmic implementation of the face recognition system approach is shown in Algorithm 3. In this algorithm, line 2 refers to the preprocessing stage, lines 3-17 refer to the HTM SP processing, lines 18-19 correspond to the training phase and lines 20-22 shows the testing (recognition) phase.

4.5.3 Results

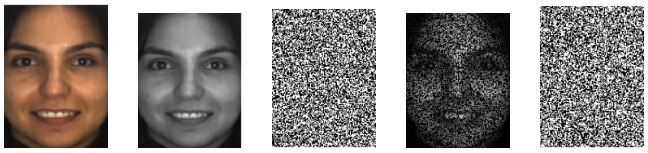
System level simulation

The experiments for the system level simulation were performed in MATLAB for 3 different databases: AR, ORL and YALE. The AR database contains 100 classes of faces with 26 face images per class with various natural variation and occlusions [58]. The ORL database includes 40 classes with 10 image per class with occlusions, scale variations and rotations [59]. The YALE database contains 15 classes with 11 images per class including different facial expressions and natural variabilities [67]. For the experiments in this work, 50% of the images with used for training and the other 50% for testing. The exemplar images for the random weight approach are shown in Fig.4.22, and for the rule-based approach in Fig.4.23.

The recognition accuracy of the random weight and rule base approaches with the variation

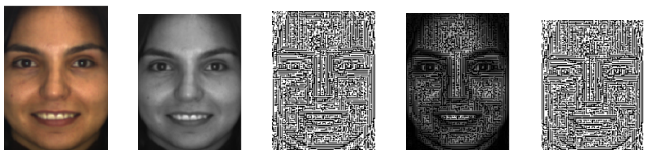
Algorithm 3 System level implementation of HTM

```
1: Define neighborhood N
2: x = grayscale(x)                                   ▷ PRE-PROCESSING
3: for p inhibition regions do                        ▷ HTM SP
4:     for k image blocks do
5:         for all i ∈ W do
6:             if x(i) ⩾ mean(x(i) ∈ N) then
7:                 W(i) = 1
8:             else
9:                 W(i) = 0
10:        image.block(j) = W(j) × image.block(j)
11:    threshold.block = mean(y.image.blocks)
12:    for y image blocks do
13:        if image.block(y) > threshold.block then
14:            inhibition.region(y) = 1
15:        else
16:            inhibition.region(y) = 0
17:    x(p) = inhibition.region(p)
18: if training phase then
19:    Store image to the training template
20: else if testing phase then
21:    Compare image to all atored templates
22:    Determine image class
```



(a) (b) (c) (d) (e)

Figure 4.22: Simulation results for the random weight approach: (a) input image, (b) grayscale image, (c) binary weights, (d) HTM SP overlap output and (e) HTM SP inhibition output.



(a) (b) (c) (d) (e)

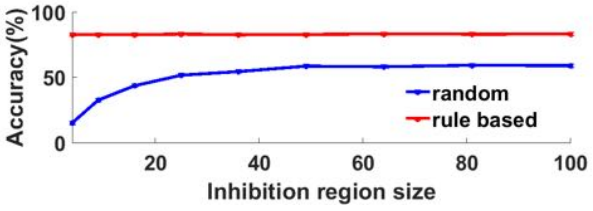
Figure 4.23: Simulation results for the rule-based approach with 2 inputs in the receptor region: (a) input image, (b) grayscale image, (c) binary weights, (d) HTM SP overlap output and (e) HTM SP inhibition output.

of the size of the inhibition region is shown in Fig.4.24. Fig.4.24(a) illustrates the simulation results for AR database, Fig.4.24(b) for ORL database and Fig.4.24(c) for YALE database. The rule-base approach improves face recognition accuracy for AR and ORL databases. However,

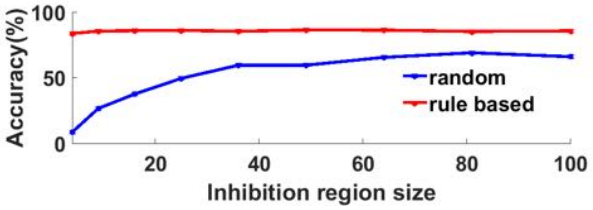
Table 4.3: The average and maximum recognition accuracies for different databases for traditional random weight and proposed rule based approaches.

Dataset	**Random weight approach**		**Rule based approach**	
	mean accuracy	maximum accuracy	mean accuracy	maximum accuracy
AR	47.992 %	59.231 %	82.855%	83.231%
YALE	91.852 %	98.667 %	85.538 %	86.308 %
ORL	49.056 %	69.000 %	85.5389 %	86.308 %

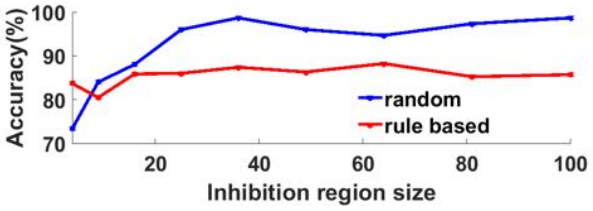
for the YALE database, the recognition accuracy is decreased. This can be explained by the small number of classes and face samples in the YALE database. The average and maximum recognition accuracies for two approaches are compared in Table 4.3.



(a)



(b)



(c)

Figure 4.24: Simulation results for the face recognition for two methods for different databases: (a) AR, (b) ORL and (c) YALE.

Analog hardware simulation

The simulation of the proposed rule-based approach was performed in SPICE for TSMC 180nm CMOS technology. Fig.4.25 illustrates the timing diagram for the proposed rule-based receptor

Table 4.4: Comparison of the random weight and rule base approaches in terms of the on-chip area and maximum power consumption of a single receptor block.

Approach	**On-chip area**	**Power dissipation**
Random weight approach	$0.125\mu m^2$	$42.92pW$
Rule-based approach	$13.31\mu m^2$	$135\mu W$

block, shown in Fig.4.20. Fig.4.25(a) shows the inputs in the receptor block. Fig.4.25(b) illustrates the main input and the total mean of all the inputs. This main input is compared with the mean in the following stages. Fig.4.25(c) shows the comparator circuit output and Fig.4.25(d) illustrates the final output of a single receptor block.

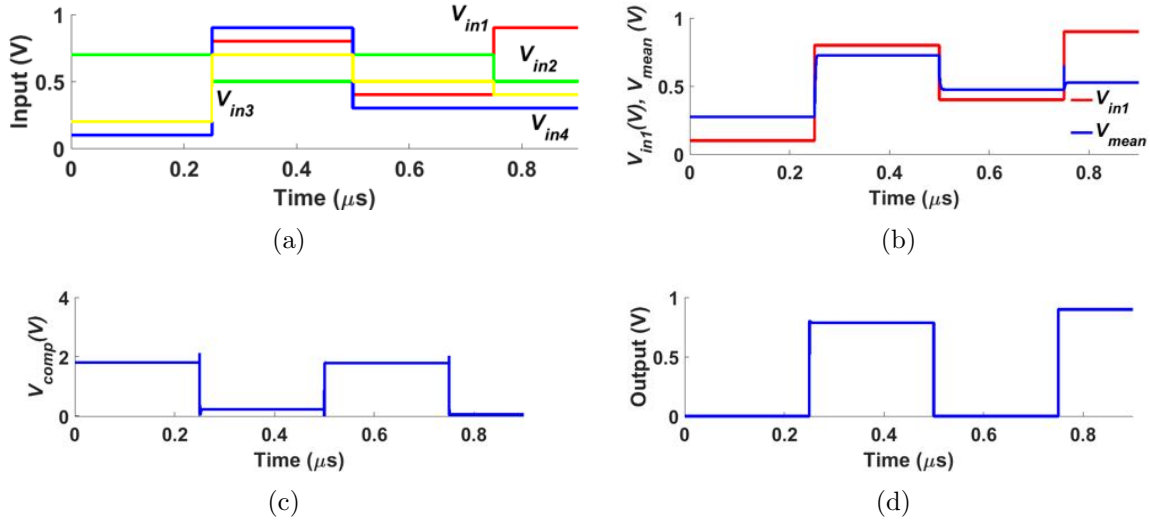


Figure 4.25: Timing diagram for the proposed receptor block for the rule-based HTM approach: (a) inputs from the neighborhood, (b) main input and mean of the inputs, (c) comparator output and (d) receptor block output.

Table 4.4 compares the on-chip area and power dissipation for random weight and rule-based approaches.

4.5.4 Discussion

The proposed rule-based approach outperforms the traditional HTM random weight approach. This can be explained by the fact that the rule-based approach that draws the correlation between the HTM SP weights to the input space. The main goal of the HTM SP is to create the SDR from the input. However, the facial images contain the natural sparseness. The rule

based approach ensures the preservation of this natural sparseness of the images, which results in the increase of the recognition accuracy. In addition, this allows to preserve the structural information from the images, such as edges.

The hardware implementation of the rule-based approach required larger on-chip area and power consumption, comparing to the traditional random weight method. However, to achieve high recognition accuracy in the rule-based approach, the image filtering stage is not required, which is performed on the separate software. Moreover, the rule-based approach does not require the programming of the memristors to the random weights, which can be achieved combining either software-based or mixed-signal random number generation approach. The programming of the memristors requires additional time and reduces the processing speed. Also, the high accuracy of the rule-based approach result allows to remove the learning phase from the HTM SP, which can be implemented using digital or analog circuits and requires a significant amount of extra power and on-chip area [17].

4.5.5 Summary

The hardware implementation of a rule-based approach for the initialization phase of the HTM SP has been proposed. The proposed rule-based approach allows to achieve significant increase in recognition accuracy. The maximum accuracy is approximately 86%, which is equivalent to the processing of the HTM SP with the learning phase. The on-chip area and power requirements to implement the rule-based initialization phase of the HTM SP are $13.31\mu m^2$ and $135\mu W$ for a single receptor block, respectively.

4.6 Conclusion

In this chapter, the learning process of HTM was investigated. The learning stages of HTM SP and HTM TM were considered. The modified architecture of HTM SP was proposed. The analog architecture of HTM TM was designed. The possibility to implement to backpropagation

learning for HTM SP learning stage was introduced. The rule-based approach of HTM SP without learning was shown.

Chapter 5

Applications of the analog backpropagation learning circuit

5.1 Introduction

In this chapter, it is demonstrated how this learning circuit can be used in different biologically inspired learning architectures, such as Neural networks, and Long-Short Term Memory (LSTM), for training of the memristive networks. The proposed circuits ensure the implementation of online learning. The additional circuit and activation functions are introduced to ensure the implementation of various biologically inspired architectures.

5.2 Additional circuits and activation functions

5.2.1 Analog architecture

The proposed analog memristive backpropagation learning circuits can be used for various applications and learning architectures, such as neural networks, HTM and LSTM hardware implementations. To apply the proposed backpropagation circuits for various architectures,

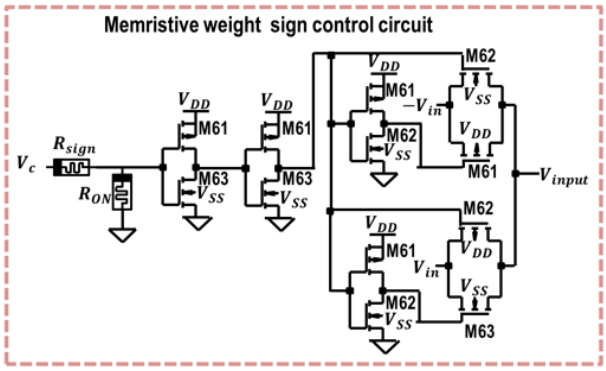


Figure 5.1: Memristive weight sign control circuit that can be integrated to the crossbar to control the weight of the synapse or applied as an external circuit with a separate memristors to store the sign of the weight.

the implementation of additional functional blocks and activation functions is required. We proposed the implementation of the memristive weight sign control circuit, tangent function, and approximate sigmoid and tangent.

As the neural network weights can be both positive and negative and negative weight cannot be produced by the memristor, the implementation of the additional weight control circuit is required. For each negative weight, the sign of the input voltage is changed. There are 2 possible ways to implement the sign. One of the solutions is to store the sign for each sequence in the external storage unit and apply it to the circuit with the weight normalization circuit. The other solution is to store the sign of each weight in the additional memristive crossbar elements. We propose a memristive weight sign control circuit shown in Fig.5.1.

The sign of each memristor in the crossbar is stored in the addition memristive crossbar or in separate memristors as R_{ON} or R_{OFF}. The memristor storing the sign of the weight is followed by the analog sign read circuit. There are three possible solutions. The first is to implement a single analog sign read circuit and switch it between the memristors in the crossbar, which requires additional on-chip area. The second solution is to have a single circuit for all the memristors in the sign crossbar, connect the circuit to different memristors one at a time and read sign in a sequential manner, which reduces the processing time. The last and most effective solution is to implement the number of sign read circuits equivalent to the number of rows in a crossbar, which allows to read the sign of all the memristors in a single column. This allows to achieve the trade-off between the required area and power and processing time. In Fig.5.1,

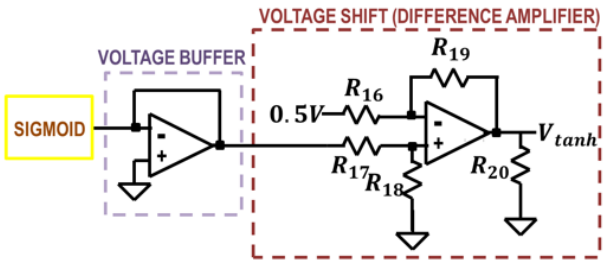
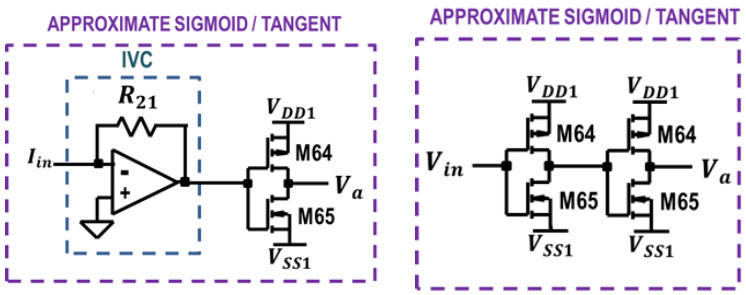


Figure 5.2: Implementation of the tangent function based on the sigmoid circuit from Fig.4.23(d). This architecture allows to implement a single circuit for both sigmoid and tangent functions in a multilayer neural network or another learning architecture and switch between these two functions.

the sign of the memristor representing the weight is stored in the memristor R_{sign}. When the sign is read $V_c = 1.25V$ is applied. If R_{sign} is set to R_{ON}, the output of the analog switch V_{sign} is positive and vice versa. The weight sing read circuit acts as a switch. If $R_{sign} = R_{ON}$, the voltage V_c is above the switch threshold and it selects and outputs the positive voltage V_{in}, which is the input voltage to the crossbar. If $R_{sign} = R_{OFF}$, the voltage drops and V_c is below the switch threshold and the switch outputs the voltage $-V_{in}$. The parameters of the transistors are the following: $M_{61} = 0.18\mu/0.72\mu$, $M_{62} = 0.18\mu/10.36\mu$ and $M_{63} = 0.18\mu/0.36\mu$. The transistors in the circuit have an underdrive voltage $V_{DD} = 1V$.

The implementation of different activation functions can be performed with the analog circuits shown in Section **??**. To implement the tangent function, the sigmoid function can be adjusted. The use of the same sigmoid circuit allows to build a single circuit for both of the functions and switch between the sigmoid and tangent implementations when it is required. The implementation of the tangent function is shown in Fig.5.2. The sigmoid and buffer part remain the same as in the sigmoid implementation and the voltage shift circuit based on the difference amplifier is added. The difference amplifier is based on the same OpAmp shown in Fig.4.23(e) with $R_{16} = 10k\Omega$, $R_{17} = 1k\Omega$, $R_{18} = 2.5k\Omega$, $R_{19} = 15k\Omega$ and $R_{20} = 1k\Omega$. The circuit shifts the voltage level of the sigmoid and allows to implement tangent function with the same circuit.

The implementation of an approximate sigmoid and tangent functions can be done with a simple thresholding circuits shown in Fig.5.3. There are options: current-control and voltage control approximate functions. The current control circuit is shown in Fig.5.3(a). The input current I_{in} is applied to the current to voltage converter based on the OpAmp with $R_{22} = 20k\Omega$



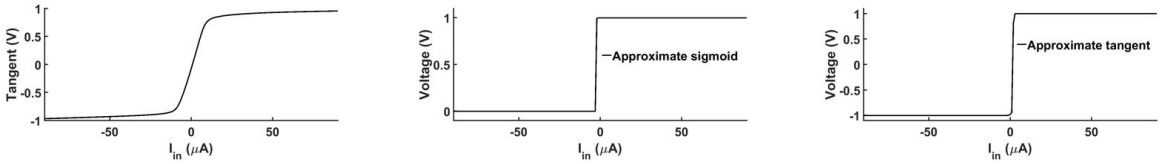
(a) (b)

Figure 5.3: Implementation of the approximate sigmoid and approximate tangent functions driven by: (a) input current and (b)input voltage. To implement the sigmoid and tangent, the voltage levels V_{DD1} and V_{SS1} are varied.

and inverted by the inverter with $M64 = 0.18\mu/0.36\mu$ and $M65 = 0.18\mu/1.72\mu$. The W/L ratio of $M64$ and $M65$ can be adjusted depending of the required transition part between high and low value of approximate sigmoid and tangent functions. The voltages V_{DD1} and V_{SS1} are different for sigmoid and tangent implementations. For the approximate sigmoid $V_{DD1} = 1V$ and $V_{SS1} = 0V$, which means that the transistors have an under-drive voltage level for TSMS 180nm CMOS technology. In the approximate tangent implementation, $V_{DD1} = 1V$ and $V_{SS1} = -1V$. The voltage-controlled sigmoid and tangent can be implemented by the simple thresholding circuits with two inverters shown in Fig.5.3(b). The W/L transistor ratios of $W_{66}-W_{69}$ and voltage levels of V_{DD1} and V_{SS1} can be adjusted to obtain a required amplitude, range and transition region for the sigmoid and tangent.

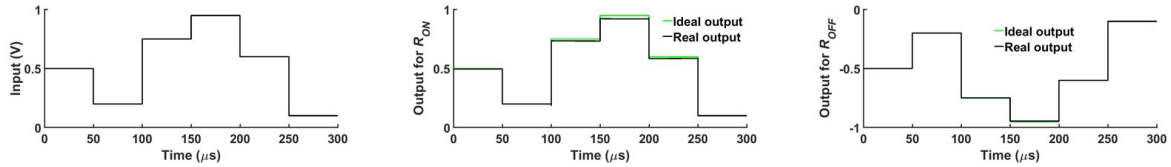
5.2.2 Simulation results

The simulation results for additional activation functions are shown in Fig.5.4. Fig.5.4 (a) represents the simulation of the proposed tangent function. Fig.5.4 (b) and Fig.5.4 (c) illustrate the simulation of current driven approximate sigmoid and tangent, respectively. The timing diagram for the memristive weight sign control circuit is shown in Fig.5.5. Fig.5.5 (a) represents the positive input voltage. Fig.5.5 (b) illustrates the ideal output and real memristive weight sign control circuit output for R_{ON}, when the weight is positive. Fig.5.5 (c) illustrates the ideal output and real output of the proposed weight sign control circuit for R_{OFF}, when the weight is negative.



(a) (b) (c)

Figure 5.4: Simulation of additional activation functions versus current: (a) tangent, (b) approximate current driven sigmoid and (c) approximate current driven tangent.



(a) (b) (c)

Figure 5.5: Timing diagram for memristive weight sign control circuit implementation: (a) input to the circuit, (b) ideal output and real output for R_{ON} (when the weight is positive) and (c) ideal and real outputs for R_{OFF} (when the weight is negative).

5.3 Neural networks with backpropagation

There is a number of architectures where the proposed learning circuit can be used. The complete analog learning and training circuits for most of the architectures and networks has not been implemented on analog hardware. One of the most important configurations is a neural network. There are different architectures and types of the neural networks that can use the proposed learning circuit without making a significant modification of the proposed design, such as deep neural networks, binary neural networks and multiple neural networks.

Table 5.1 represents the calculation of the on-chip area and maximum power dissipation for separate components for the analog leaning circuit implementation and additional components and activation functions. In addition, the example of area and power dissipation for a small crossbar is shown.

Table 5.1: Power consumption and on-chip area calculation for the separate circuit components.

Circuit component	**Power consumption**	**On-chip area**
Weight sign control circuit	$195.1\mu W$	$16.64\mu m^2$
Approximate current driven sigmoid/tangent	$52.9mW$	$2118.00\mu m^2$
Approximate voltage driven sigmoid/tangent	$41.2pW$	$0.40\mu m^2$

Summary

We present the analog circuit implementation of additional components and activation functions that can be used to implement various learning architectures. Moreover, we show the hardware implementation of various learning architectures with the proposed backpropagation learning, including deep neural network, binary neural network, multiple neural network, conventional and modified HTM SP and LSTM.

5.3.1 Binary neural network with backpropagation learning

Introduction

We propose the analog hardware implementation of BNN with backpropagation and binary weights that can be scaled to implements deep BNN. The overall performance of the system is tested for different number of neural network layers. The increase in the number of hidden layers and connections between them represented by the memristive crossbar and analog implementation of different activation functions allows to build binary deep neural network. The proposed network is tested using MNIST and IRIS databases.

Background

There are several implementations of BNN. In this work, we focus on the implementation of BNN with binary weights trained with the backpropagation algorithm. In the training stage, the weights are updated with the backpropagation algorithm with gradient descent. After the training, the weights are binarized before the recognition and classification process. In general, the backpropagation algorithm consists of four steps: forward propagation, backpropagation to the output layer, backpropagation to the hidden layers and the update of weights.

In the implemented algorithm, the weights are binarized to a certain values w_{low} and w_{high} and their negative alternatives $-w_{low}$ and $-w_{high}$ according to the Eq.5.1. All weights in each

weight matrix are rounded to the closest positive or negative weight.

$$\Delta w = \begin{cases} w_{high} & \text{if} \quad round(w) = w_{high} \\ w_{low} & \text{if} \quad round(w) = w_{low} \\ -w_{low} & \text{if} \quad round(w) = -w_{low} \\ -w_{high} & \text{if} \quad round(w) = -w_{high} \end{cases} \tag{5.1}$$

Such binarization is useful for the hardware implementation of the BNN using memristive crossbar approach, where only high and low value of memristor is possible to achieve.

Hardware implementation of BNN

The proposed hardware implementation of the BNN is shown in Fig.5.6. The design consists of the memristive crossbars, activation function, sequence control unit, storage and normalization unit, weight sign control unit, neural network training unit and weight update unit. Each input image is reshaped and feed into the crossbar rows to V_{in1} and up to V_{ink}, where k is the number of pixels in a single image. The number of the rows in the crossbar corresponds to the number of the inputs neurons to the crossbar and the number of columns refers to the number of output neurons in the BNN.

The design of BNN is based on the binary memristive crossbar representing the weights between the layers of the neurons in the neural network. The output of the crossbar is the current in the transistor M_r. The currents from each column of the crossbar are read sequentially one column at a time. The transistors are controlled by the control signal V_c, which switches the transistors ON and OFF. To ensure that the power consumption of the circuit is not too high, the maximum V_c switches on the transistor is restricted to $1V$. In order to keep the current through the transistor proportional to the applied voltage signal, the following condition should be fulfilled: $V_{DC} \leqslant V_{GS} - V_t$, where V_t is the transistor threshold. We adjust the overall circuit to TSMC 180nm CMOS process, and the NMOS transistor that is used in the crossbar has the threshold level $V_t = 0.35V$. Therefore, the maximum drain voltage of the transistor to

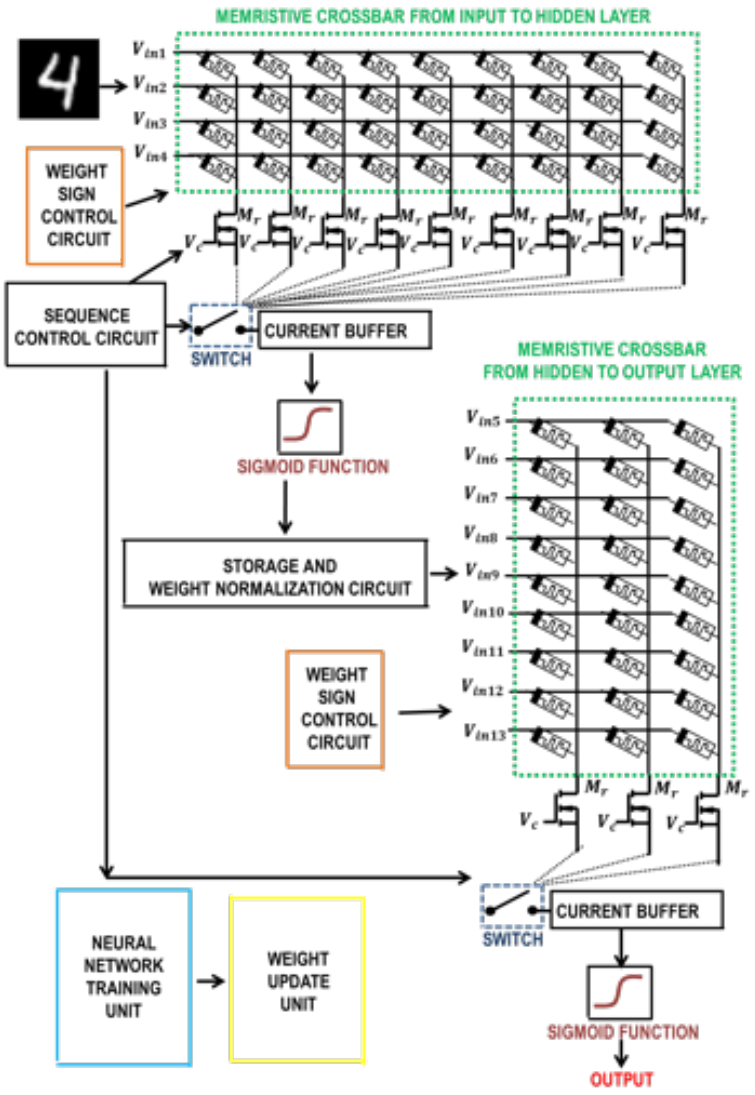


Figure 5.6: Overall architecture of the proposed BNN implementation

ensure linear operation is $V_D = 0.65V$ and $V_{in} < 0.65V$. For this work, the HP memristor model is used and the analog circuits and transistor parameters are adjusted to the following values of memristors: $R_{on} = 3k$ and $R_{off} = 62k$ with the threshold level of 1V. To avoid the summation of the current to exceed the range of the *sigmoid* function, the input voltage levels are normalized between $0V$ and $0.1V$. This range was obtained by analyzing the range of output currents flowing through a single memristor for R_{on} and R_{off} regions shown in Fig.5.7.

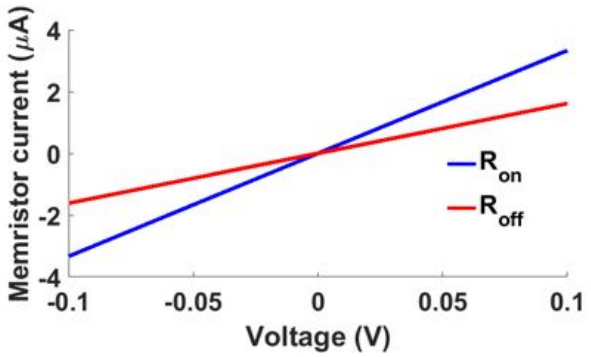


Figure 5.7: Output current range for a single memristor for R_{on} and R_{off}

Binary neural network with backpropagation

Binary neural network can be implemented with the proposed circuit using two stage memristors in the memristive crossbar representing the weights. The implementation of the binary neural network is shown in Fig.5.8. In the binary neural network, the forward propagation process and backpropagation process are the same as in a three layer neural network. However, due to the limitations of the binary weights, the direct update of the weights after the error calculation will not provide hight accuracy results. We suggest to store the value of the change in error in the external storage and training units in time. And after a certain time period of the training update the weights. This method can improve the accuracy results for the classification problems using binary neural networks. In addition, binary neural networks provide high accuracy results for only a particular set of problems.

5.3.2 Deep neural network with backpropagation learning

The proposed configuration for deep neural network with the proposed memristive analog learning circuits is shown in Fig.5.9. The deep neural network configuration contains N+1 layers and N crossbars correspond to the synapses between the layers. In the forward propagation process MB1 is used. MB1 can be modified to implement various activation functions. The backpropagation process through the output layer of deep neural network in performed by MB2 and MB4, and the backpropagation through hidden layers is done by MB3. In the update process MB4 and MB3 are applied. The blocks MB2, MB3 and MB4 in each layer can be modified

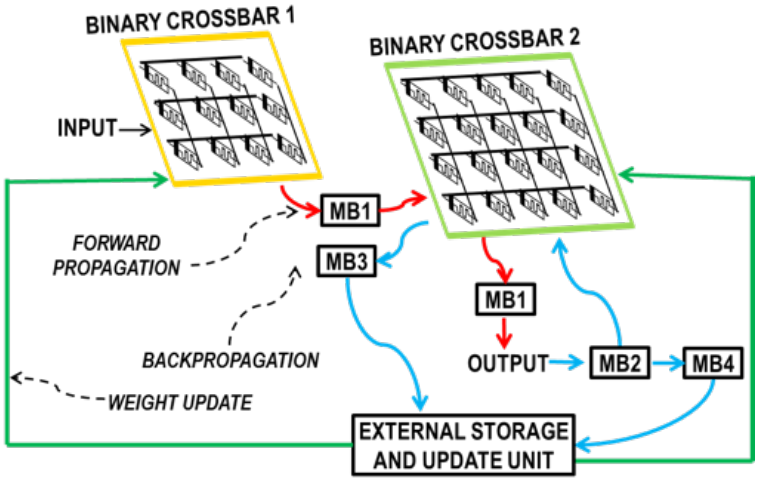


Figure 5.8: Three layer binary neural network with backpropagation learning. The crossbars contain memristors that can be programmed only for R_{ON} and R_{OFF} stages. To improve the accuracy and the performance of the network, the change in error is stored in the external storage and update unit. The crossbar weights are updated based on several iterations in time.

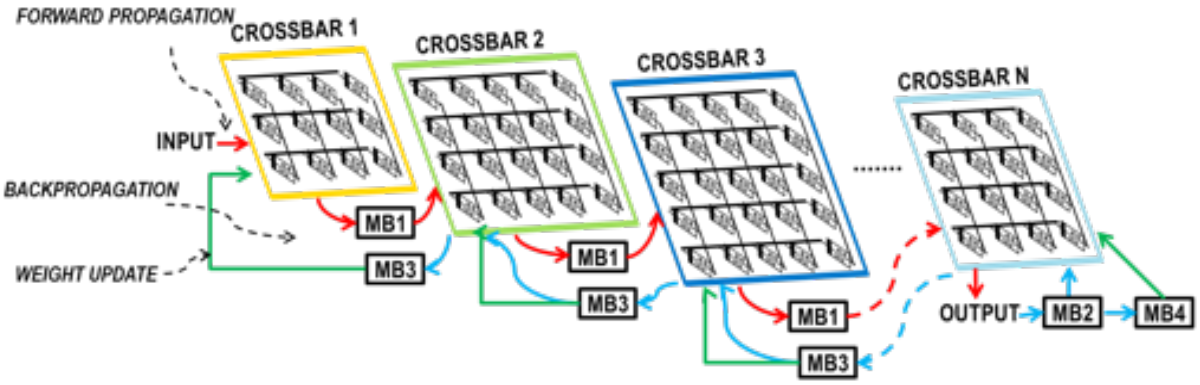


Figure 5.9: Deep neural network implementation with the backpropagation learning. Red arrows correspond to forward propagation process. Blue arrows refer to backpropagation process. Green arrows show the weight update process.

depending on the activation function applied in forward propagation in MB1 for each layer.

5.3.3 Multiple neural network with backpropagation learning

The proposed memristive analog learning circuits can be used for the Multiple Neural Network (MNN) approach. This is useful when several sources of input data are used and the decision on the output depends on the fusion of the results from each data source. Each data source output are fetched to separate crossbar and the outputs of all the crossbars are fetched into the decision layer. The architecture of the multiple neural network with backpropagation learning

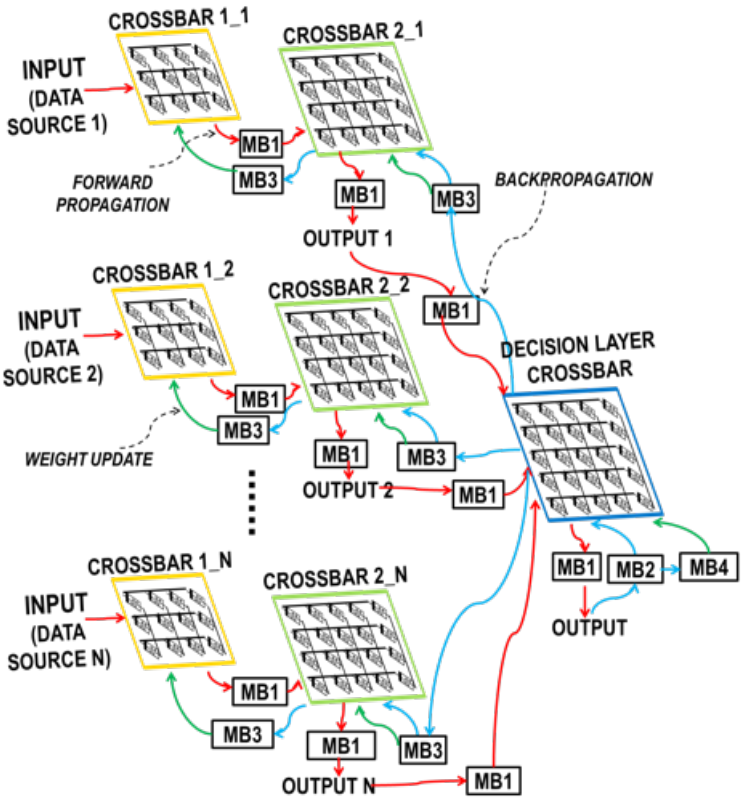


Figure 5.10: Multiple neural network with backpropagation learning. The inputs from different data sources are fetched into different crossbars and the outputs from the crossbars are used as the inputs to the decision layer containing the memristive synapses.

is illustrated in Fig.5.10. We recommend to use this approach, when the neural network inputs are taken from different data sources, such as various sensors in the system. The activation functions of the layer are different for separate crossbars and for the decision layer and depend on the data that is used for the processing. The number of the required backpropagation blocks equals to the number of the crossbars that the network contain.

5.3.4 Simulation results

DNN and BNN results

The system level simulations were performed in MATLAB for MNIST and IRIS database. Table 5.2 shows the performance evaluation of the proposed BNN, comparing to the conventional neural network (NN) with backpropagation. In general, the average accuracy results for BNN are lower than the conventional NN. However, according to the maximum possible accuracy

results, the BNN approach outperforms the conventional NN. The BNN is sensitive to the values of the weights. As the simulations are performed for different sets of binary weight, the average accuracy of the algorithm is lower then for the conventional NN. However, the selection of a particular value of the weights allows to achieve the maximum possible accuracy, which is higher than the performance accuracy of the conventional NN. The overall performance of deep BNN and deep NN is lower, in comparison to the 3 layer NN and BNN. To enhance the performance of deep BNN, the increase of the number of training iterations and selection of the appropriate number of hidden layer neurons is required.

In addition, the performance of the BNN varies depending on number of features, number of neurons in the hidden layer and the value of binary weights. Therefore, to achieve high accuracy with the BNN, the selection of the values of binary weights should be performed considering these parameters. The selection of the weights corresponds to the selection of the range of input voltage levels in the hardware implementation.

Table 5.2: Performance evaluation of the proposed BNN comparing to the conventional neural network.

Method	**Average accuracy**		**Maximum Accuracy**	
	MNIST database	IRIS database	MNIST database	IRIS database
Conventional 3 layer NN with backpropagation	89.9%	96.6%	91%	100%
Proposed 3 layer BNN with backpropagation	**89.6%**	**68%**	**100%**	**100%**
Conventional deep NN with backpropagation (6 layers)	32%	32%	62%	62%
Proposed deep BNN with backpropagation (6 layers)	**22%**	**10%**	**39.9%**	**32%**

5.3.5 Summary

The hardware implementation of the BNN, DNN and MNN has been proposed. We presented the application of the memristive binary crossbar for BNN, DNN and MNN and the design of the hardware units and the analog circuits for activation functions. The performance of the proposed BNN and DNN was evaluated using MNIST and IRIS databased. The average

accuracy that can be achieved with BNN is approximately 90%. The hardware implementation is scalable. Thus, it is possible to implement deep BNN. The area and power requirements for 3 layer BNN with the crossbar of the size of 4×10 are $1072.4mW$ and $4839.9\mu m^2$, respectively.

5.4 LSTM with backpropagation learning

LSTM architecture can also be implemented using the proposed memristive analog backpropagation circuits. The full implementation of LSTM with analog circuits has not been proposed yet. However, the analog implementation of the separate LSTM components have been shown in [24]. Fig.5.11 illustrates the full system level LSTM architecture consisting of the output gate, input gate, write gate and forget gate. The weights of LSTM gates W_i, W_o, W_f and W_c can be stored in the memristive crossbars. The activation functions in the LSTM architecture can be replaced with different variations of MB1. While, the weight update process of the crossbar W_o is performed by MB4 as the update of the output layer, which the update of W_i, W_f and W_c is performed by MB3.

5.5 Summary

In this chapter, the possibility of implementation of analog backpropagation learning circuit based on memristive crossbar has been investigated. The learning architectures, such as Deep Neural Network, Binary Neural Network, Multiple Neural Network and LSTM with the learning stage were proposed. The analog circuits with additional functions were implemented.

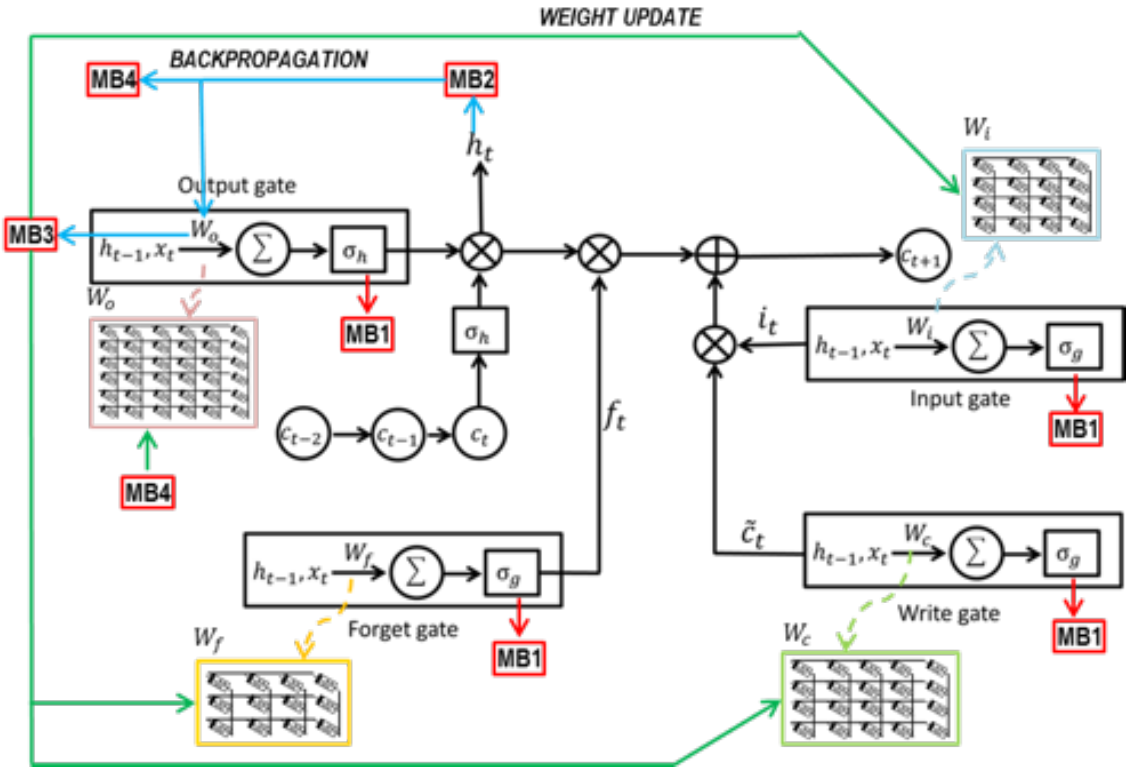


Figure 5.11: Analog memristive hardware implementation of the LSTM algorithm.

Chapter 6

Conclusion

6.1 Contributions and Results

In this thesis, the implementation of learning and training process has been investigated. The implementation of analog learning circuit for the learning process with backpropagation with gradient descent was proposed. The learning circuit is useful for memristive crossbar array based architectures. The learning process of HTM was investigated. The learning parts of HTM SP and HTM TM were studied. The HTM TM learning circuit was proposed. The possibility of rule-based HTM SP implementation without learning was shown. The possibility to integrate the proposed learning circuit to various biologically inspired learning architectures is illustrated. The LSTM, BNN, MNN and DNN architecture with backpropagation learning on hardware are shown. The additional activation function for implementation of various learning circuits using memristive crossbar are shown.

6.2 Open problems

The open problems include:

- Scalability of the analog architecture. The power dissipation and on-chip area of the

proposed circuit should be reduced and the design of analog learning block has to be optimized.

- Scalability of memristive crossbar. The parasitics, sneak path effects and leakage current problems in memristive crossbar should be investigated and the limitations of the crossbar size has to be determined.

- The limitations of memristive devices. Most of the currently available memristive devices have a limit in terms of number of states that can be achieved and problems with integrity with CMOS transistors and silicon chips.

6.3 Future Work

The possible future work may include the following:

- Full control circuit and memory unit for the proposed learning architecture should be implemented.

- The frequency and electromagnetic effects on memristive crossbar should be studied.

- The design of the analog learning circuit should be improved in terms of area and power.

- The possibility of integration of HTM SP with backpropagation should be investigated.

- The full implementation of the proposed biologically inspired architecture with backpropagation learning should be performed.

- The system level implementation of the proposed architectures for various data processing problems has to be performed.

Bibliography

[1] G. Khodabandehloo, M. Mirhassani, and M. Ahmadi, "Analog implementation of a novel resistive-type sigmoidal neuron," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 4, pp. 750–754, April 2012.

[2] I. Kataeva, F. Merrikh-Bayat, E. Zamanidoost, and D. Strukov, "Efficient training algorithms for neural networks based on memristive crossbar circuits," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–8.

[3] E. Zamanidoost, F. M. Bayat, D. Strukov, and I. Kataeva, "Manhattan rule training for memristive crossbar circuit pattern classifiers," in *Intelligent Signal Processing (WISP), 2015 IEEE 9th International Symposium on*. IEEE, 2015, pp. 1–6.

[4] B. Li, Y. Wang, Y. Wang, Y. Chen, and H. Yang, "Training itself: Mixed-signal training acceleration for memristor-based neural network," in *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*. IEEE, 2014, pp. 361–366.

[5] A. M. Hassan, C. Yang, C. Liu, H. H. Li, and Y. Chen, "Hybrid spiking-based multi-layered self-learning neuromorphic system based on memristor crossbar arrays," in *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2017, pp. 776–781.

[6] X. Hu, G. Feng, S. Duan, and L. Liu, "A memristive multilayer cellular neural network with applications to image processing," *IEEE transactions on neural networks and learning systems*, 2017.

[7] R. Hasan and T. M. Taha, "Enabling back propagation training of memristor crossbar neuromorphic processors," in *Neural Networks (IJCNN), 2014 International Joint Conference on*. IEEE, 2014, pp. 21–28.

[8] E. R. Kandel and R. D. Hawkins, "The biological basis of learning and individuality," *Scientific American*, vol. 267, no. 3, pp. 78–86, 1992.

[9] P. Leff, H. Romo Parra, J. C. Calva, R. Acevedo, R. Gutiérrez, and B. Anton, "Synaptic plasticity: Understanding the neurobiological mechanisms of learning and memory. part ii," *Salud Mental*, vol. 24, no. 3, 2001.

[10] D. A. Baxter and J. H. Byrne, "Learning rules from neurobiology," *The neurobiology of neural networks*, pp. 71–105, 1993.

[11] J. Sun, "Cmos and memristor technologies for neuromorphic computing applications," 2015.

[12] 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, 2016.

[13] J. Hawkins and S. Blakeslee, "On intelligence. 2004," *New York St. Martins Griffin*, pp. 156–8.

[14] J. Hawkins, S. Ahmad, S. Purdy, and A. Lavin, "Biological and machine intelligence (bami)," 2016, initial online release 0.4. [Online]. Available: http://numenta.com/biological-and-machine-intelligence/

[15] D. George and J. Hawkins, "A hierarchical bayesian model of invariant pattern recognition in the visual cortex," in *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, vol. 3. IEEE, 2005, pp. 1812–1817.

[16] A. P. James, I. Fedorova, T. Ibrayev, and D. Kudithipudi, "Htm spatial pooler with memristor crossbar circuits for sparse biometric recognition," *IEEE Transactions on Biomedical Circuits and Systems*, vol. PP, no. 99, pp. 1–12, 2017.

[17] O. Krestinskaya, T. Ibrayev, and A. P. James, "Hierarchical temporal memory features with memristor logic circuits for pattern recognition," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. PP, no. 99, pp. 1–1, 2017.

[18] N. Inc., "Hierarchical temporal memory including htm cortical learning algorithms," Tech. Rep., 2006.

[19] N. Ahad, J. Qadir, and N. Ahsan, "Neural networks in wireless networks: Techniques, applications and guidelines," *Journal of Network and Computer Applications*, vol. 68, pp. 1 – 27, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804516300492

[20] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in neural information processing systems*, 2016, pp. 4107–4115.

[21] D. W. Patterson, "Artificial neural networks, theory and applications, prentice hall, singapore, 1996."

[22] M. Sundermeyer, H. Ney, and R. Schlter, "From feedforward to recurrent lstm neural networks for language modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 517–529, March 2015.

[23] K. Greff, R. K. Srivastava, J. Koutnk, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, Oct 2017.

[24] K. Smagulova and A. P. James, "A memristor-based long short term memory circuit," *Proceedings of Oxford Circuits and Systems Conference*, 2017.

[25] D. Soudry, D. Di Castro, A. Gal, A. Kolodny, and S. Kvatinsky, "Hebbian learning rules with memristors," *Israel Institute of Technology: Haifa, Israel*, 2013.

[26] Y. Choe, "Anti-hebbian learning," in *Encyclopedia of Computational Neuroscience*. Springer, 2015, pp. 191–193.

[27] P. M. Sheridan, C. Du, and W. D. Lu, "Feature extraction using memristor networks," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 11, pp. 2327–2336, 2016.

[28] M. R. Azghadi, S. Al-Sarawi, N. Iannella, and D. Abbott, "Design and implementation of bcm rule based on spike-timing dependent plasticity," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*. IEEE, 2012, pp. 1–7.

[29] M. V. Nair and P. Dudek, "Gradient-descent-based learning in memristive crossbar arrays," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–7.

[30] X. Wu, V. Saxena, and K. Zhu, "A cmos spiking neuron for dense memristor-synapse connectivity for brain-inspired computing," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–6.

[31] A. Cohen, R. Douglas, C. Koch, T. Sejnowski, S. Shamma, T. Horiuchi, G. Indiveri *et al.*, "Report to the national science foundation: Workshop on neuromorphic engineering," *Telluride, Colorado, USA (June-July 2004) www. ini. unizh. ch/telluride*, 2004.

[32] T. S. Lande, *Neuromorphic systems engineering: neural networks in silicon*. Springer Science & Business Media, 1998, vol. 447.

[33] I. Kataeva, F. Merrikh-Bayat, E. Zamanidoost, and D. Strukov, "Efficient training algorithms for neural networks based on memristive crossbar circuits," in *2015 International Joint Conference on Neural Networks (IJCNN)*, July 2015, pp. 1–8.

[34] Y. Zhang, X. Wang, and E. G. Friedman, "Memristor-based circuit design for multilayer neural networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2017.

[35] D. Negrov, I. Karandashev, V. Shakirov, Y. Matveyev, W. Dunin-Barkowski, and A. Zenkevich, "An approximate backpropagation learning rule for memristor based neural networks using synaptic plasticity," *Neurocomputing*, vol. 237, pp. 193–199, 2017.

[36] Y. Chauvin and D. E. Rumelhart, *Backpropagation: theory, architectures, and applications*. Psychology Press, 1995.

[37] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[38] S. Xiao, X. Xie, S. Wen, Z. Zeng, T. Huang, and J. Jiang, "Gst-memristor-based online learning neural networks," *Neurocomputing*, 2017.

[39] D. Biolek, Z. Kolka, V. Biolkova, and Z. Biolek, "Memristor models for spice simulation of extremely large memristive networks," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2016, pp. 389–392.

[40] A. M. Zyarah, "Design and analysis of a reconfigurable hierarchical temporal memory architecture," Master's thesis, Rochester Institute of Technology, 1 Lomb Memorial Dr, Rochester, NY 14623, 6 2015.

[41] A. B. Csapó, P. Baranyi, and D. Tikk, "Object categorization using vfa-generated nodemaps and hierarchical temporal memories," in *Computational Cybernetics, 2007. ICCC 2007. IEEE International Conference on.* IEEE, 2007, pp. 257–262.

[42] T. Ibrayev, A. P. James, C. Merkel, and D. Kudithipudi, "A design of htm spatial pooler for face recognition using memristor-cmos hybrid circuits," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2016, pp. 1254–1257.

[43] A. James, A. Irmanova, and T. Ibrayev, "Design of discrete-level memristive circuits for hierarchical temporal memory based spatio-temporal data classification system," *IET Cyber-Physical Systems: Theory & Applications*, 2017.

[44] D. Fan, M. Sharad, A. Sengupta, and K. Roy, "Hierarchical temporal memory based on spin-neurons and resistive memory for energy-efficient brain-inspired computing," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 9, pp. 1907–1919, 2016.

[45] L. Streat, D. Kudithipudi, and K. Gomez, "Non-volatile hierarchical temporal memory: Hardware for spatial pooling," *arXiv preprint arXiv:1611.02792*, 2016.

[46] A. Irmanova and A. P. James, "Htm sequence memory for language processing," in *Poster session presented at IEEE International Conference on Rebooting Computing (ICRC 2017)*, Nov 2017.

[47] Y. Cui, S. Ahmad, and J. Hawkins, "The htm spatial pooler: a neocortical algorithm for online sparse distributed coding," *bioRxiv*, p. 085035, 2017.

[48] ——, "Continuous online sequence learning with an unsupervised neural network model," *Neural computation*, 2016.

[49] D. George and J.Hawkins, "Hierarchical temporal memory: Concepts, theory and terminology," Tech. Rep., 2006.

[50] A. B. Csapo, P. Baranyi, and D. Tikk, "Object categorization using vfa-generated nodemaps and hierarchical temporal memories," in *Computational Cybernetics, 2007. ICCC 2007. IEEE International Conference on*, Oct 2007, pp. 257–262.

[51] W. J. C. Melis, S. Chizuwa, and M. Kameyama, "Evaluation of the hierarchical temporal memory as soft computing platform and its vlsi architecture," in *2009 39th International Symposium on Multiple-Valued Logic*, May 2009, pp. 233–238.

[52] M. Deshpande, "Fpga implementation and acceleration of building blocks for biologically inspired computational models," Master's thesis, Portland State University, 2011.

[53] D. Fan, M. Sharad, A. Sengupta, and K. Roy, "Hierarchical temporal memory based on spin-neurons and resistive memory for energy-efficient brain-inspired computing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–1, 2015.

[54] D. Biolek, Z. Kolka, V. Biolkova, and Z. Biolek, "Memristor models for spice simulation of extremely large memristive networks," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2016, pp. 389–392.

[55] 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.

[56] H. Sato and S. Takagi, "Low-voltage amplifier with improved linearity using triode region mosfet," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2015, pp. 2469–2472.

[57] A. K. Maan, D. A. Jayadevi, and A. P. James, "A survey of memristive threshold logic circuits," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 8, pp. 1734–1746, Aug 2017.

[58] A. Martınez and R. Benavente, "The ar face database," *Rapport technique*, vol. 24, 1998.

[59] R. Ahdid, S. Safi, and B. Manaut, "Approach of facial surfaces by contour," in *Multimedia Computing and Systems (ICMCS), 2014 International Conference on*, April 2014, pp. 465–468.

[60] ——, "Approach of facial surfaces by contour," in *Multimedia Computing and Systems (ICMCS), 2014 International Conference on*, April 2014, pp. 465–468.

[61] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon Technical Report N*, vol. 93, 1993.

[62] D. P. W. Ellis, "PLP and RASTA (and MFCC, and inversion) in Matlab," 2005, online web resource. [Online]. Available: http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/

[63] T. Ibrayev, U. Myrzakhan, O. Krestinskaya, A. Irmanova, and A. P. James, "On-chip face recognition system design with memristive hierarchical temporal memory," *arXiv preprint arXiv:1709.08184*, 2017.

[64] Y. Long, E. M. Jung, J. Kung, and S. Mukhopadhyay, "Reram crossbar based recurrent neural network for human activity detection," in *2016 International Joint Conference on Neural Networks (IJCNN)*, July 2016, pp. 939–946.

[65] D. Kuzum, R. G. 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.

[66] M. Al-Shedivat, R. Naous, G. Cauwenberghs, and K. N. Salama, "Memristors empower spiking neurons with stochasticity," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 5, no. 2, pp. 242–253, 2015.

[67] R. Senthilkumar and R. K. Gnanamurthy, "A detailed survey on 2d and 3d still face and face video databases part i," in *Communications and Signal Processing (ICCSP), 2014 International Conference on*, April 2014, pp. 1405–1409.