Morphological learning for spiking neurons with nonlinear active dendrites

Shaista Hussain

2016

Shaista Hussain. (2016). Morphological learning for spiking neurons with nonlinear active dendrites. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/69184>

<https://doi.org/10.32657/10356/69184>

MORPHOLOGICAL LEARNING FOR SPIKING NEURONS WITH NONLINEAR ACTIVE DENDRITES

SHAISTA HUSSAIN

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2016

ACKNOWLEDGEMENTS

First and foremost, I would like to express my appreciation and sincere thanks to my advisor, Professor Arindam Basu. His tremendous support and encouragement made my PhD an amazing experience. I would like to thank him for showing confidence in my work and allowing me to grow as a researcher. The insightful discussions with him have enabled me to develop the skills of critical and analytical thinking. His enthusiasm towards research has been a constant source of motivation for me and has inspired me to work hard. He has been a great mentor for me and his advice regarding research as well as career have been priceless. I am also very grateful to Professors Justin Dauwels and Shoushun Chen for being the members of my PhD committee and for giving valuable suggestions and feedback which contributed to shaping this thesis work. I wish to express my humble gratitude to the NTU for providing me an opportunity to pursue my PhD under the Research Student Scholarship (RSS) scheme and the School of Electrical and Electronic Engineering for providing all the research facilities and financial support for attending overseas workshops and conferences.

I would like to thank my lab mates, Roshan Gopalakrishnan, Subhrajit Roy, Chen Yi and Enyi Yao for all their help and support. It has been a real pleasure to discuss and work with them on various research projects. I would also like to express my gratitude towards the staff of the Centre for Bio Devices and Signal Analysis (VALENS), where I did most of my PhD work. I would specially like to mention Mdm. Teng Siew Juan and Mr. Lim Cheng Chye for providing the administrative and technical support I needed to run my computer simulations. They have been very friendly and were always ready to help out. The conducive working environment in VALENS contributed a lot to making my PhD journey a delightful experience.

I would like to thank my family for being my support system. My parents, husband and

parents-in-law were immense source of support and encouragement which sustained me thus far. I really appreciate the sacrifices they made on my behalf, particularly my husband, who showed tremendous patience to deal with my routine and late working hours. I would sincerely like to thank him for supporting me by taking care of our infant daughter when I could not be there for her during my paper submissions, exams and presentations. Lastly and most importantly, my daughter, Kulsoom, has been my biggest strength who kept me going during the most difficult times. Her innocence, her baby talk and all her activities cheered me up even during the most gruelling days. I am so thankful to her for not being difficult when I had to leave her at the infant care due to my work. It was the realization that I was not giving her the time she deserved, which helped me stay focussed and complete my PhD research work in a timely manner. I love her for just being there in my life and making it so beautiful.

ABSTRACT

There has been a lack of progress in developing spiking neuron models for pattern classification, which can achieve similar performance as state-of-the-art. To pursue this goal of creating powerful spike-based classifiers, the role of dendrites in neuronal information processing is considered. The neurobiological evidence for dendritic processing has been established in the last few years by neuroscientists across the globe. However, computational models of spiking neurons in machine learning systems have not utilized this mechanism yet. Our work attempts to bridge this gap and explore the possible computational benefits of passive delay and active ionic dendritic mechanisms. A spike-based model for pattern classification is presented which employs neurons with functionally distinct multicompartment dendritic branches. In this model, synaptic integration involves location-dependent processing of inputs on each dendritic compartment, followed by nonlinear processing of the total synaptic input on a dendrite and finally linear integration of the total dendritic output at the soma. This gives the neuron a capacity to perform a large number of input-output mappings.

Firstly, a spiking neuron model is developed based on modifying delays associated with the spikes arriving at an afferent. The application of this model is demonstrated on memorizing spatio-temporal patterns by updating only a few delays corresponding to the most synchronous part of a spike pattern. This model explores the time-based computing approach to design a novel learning algorithm which provides an alternative to the traditional weight-based learning and offers the advantage of simpler hardware implementation without multipliers or digital-analog converters (DACs). The classification accuracy of the system with a load (number of patterns relative to the number of synapses) of up to 2 was shown to be about 80 – 100%. In our pursuit of achieving improved performance and a hardware-friendly learning algorithm, a model is further proposed which consists of nonlinear dendrites and is inspired by the mechanism of

structural plasticity involving correlation-based grouping of synaptic contacts onto dendrites. The model utilizes sparse synaptic connectivity, where each synapse takes a binary value, and learns the optimal input connections of a neuron. The modification of connectivity matrix renders the model suitable for implementation in address-event representation (AER) based neuromorphic systems. A modified margin-based model is also presented, which is shown to result in significant improvement in generalization performance. This performance is found to be comparable with that of standard methods like support vector machine (SVM) and extreme learning machine (ELM) on benchmark data sets and moreover, it is achieved by utilizing 10 – 50% less number of low resolution synapses compared to these algorithms.

The structural learning rule for nonlinear dendrites is also extended to perform multiclass classification and demonstrate the application of this multiclass model on the classification of handwritten digits from the MNIST dataset. For this application, our model is shown to attain comparable performance ($\approx 2\%$ more error) with other spike-based classifiers while using much less synaptic resources, up to 10% of that used by other methods. Two approaches are used to train the model: 1) a branch-specific spike-time-based structural learning rule and; 2) a rate-based version of this rule to reduce training time. The correspondence between these two forms of learning is also established. Enhancements are proposed in this model by developing an adaptive structural scheme to learn the number of dendrites by progressively adding dendrites to the network and simultaneously forming synaptic connections on these dendrites, thereby allocating synaptic resources as per the complexity of the given problem. The model is further enhanced by using an ensemble method which combines several dendritic classifiers to achieve enhanced generalization over individual classifiers. The performance of these enhanced learning algorithms is again demonstrated on the classification of MNIST data, which shows that an ensemble model created with adaptively learned classifiers can attain accuracy of 96.2% which is at par with the best performance reported for any spiking neural network. Moreover, the ensemble has the advantage of using much less number of synapses, about 20% of other spike algorithms.

Finally, the performance benefits of combined learning of delays associated with afferents

and connection pattern of these afferents are investigated. This novel learning algorithm is used by a spiking neuron model comprising nonlinear dendrites with multiple delay compartments on each dendrite to classify spatio-temporal patterns. The multicompartment dendritic neurons are different from our earlier model using lumped dendritic nonlinearity in which all synaptic inputs on a dendrite were lumped together in one compartment. The combined learning rule inspired by the Tempotron rule uses correlation computations to form synaptic connections on specific delay compartments of the nonlinear dendrites. Our neurobiologically realistic multicompartment dendritic model achieves enhanced classification accuracy, which is about 5% higher than that attained by lumped dendrite scheme. Moreover, the Tempotron rule with weights quantized to 4-bits attains about 5% less accuracy than our multicompartment model, which uses binary weights, thereby rendering our proposed learning scheme more appealing for hardware implementation.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	ii
ABSTRACT	iv
LIST OF FIGURES	x
LIST OF TABLES	xvii
CHAPTER	
1. Introduction and Motivation	1
1.1 Motivation and Background	1
1.2 Major Contributions of the Thesis	5
1.3 Organization of the Thesis	6
2. Literature Review	8
2.1 Higher-Order Networks for Nonlinear Classification Problems	8
2.2 Biologically Motivated Higher-Order Networks	10
2.3 Learning Mechanism in Higher-Order Networks of Dendritic Neurons	12
2.3.1 Synaptic Clustering on Dendrites	12
2.3.2 Structural Plasticity	14
2.3.3 Learning with Low Resolution Synapses	15
2.4 Delay Learning for Passive Dendrites	16
3. Spike Pattern Memorization and Classification using Delay Learning Ar-	
chitectures	17
3.1 The DELTRON Model	19
3.1.1 Network Architecture	19
3.1.2 Input Pattern Space	21
3.1.3 Choice of Threshold and the Two types of error	22
3.2 Learning Algorithms	23
3.2.1 Delay Learning Algorithm for Pattern Memorization	23
3.2.2 Delay Learning Algorithm for Pattern Classification	26

3.3	Simulation Results	27
3.3.1	Effect of Learning: V_{max} and t_{max} distribution	28
3.3.2	Pattern Memorization	28
3.3.3	Pattern Classification	31
3.3.4	Effect of non-idealities	34
3.4	Summary	37
3.5	Publications	38
4.	Spike Pattern Classification using a Model of Nonlinear Active Dendrites and Structural Learning	39
4.1	Model of Nonlinear Dendrites	42
4.1.1	Spike-Based Model	44
4.1.2	Reduced Model (RM)	47
4.1.3	Reduced Model with Margin (RMWM)	50
4.2	Theoretical Capacity Predictions	52
4.3	Learning Algorithm	53
4.3.1	Learning Rule for RM	53
4.3.2	Learning Rule for RMWM	56
4.3.3	Branch-specific Spike-time Dependent Structural Plasticity (BSTDSP) Rule	57
4.4	Results	63
4.4.1	Performance of RM	65
4.4.2	Performance of RMWM	72
4.4.3	Performance of BSTDSP Model	77
4.4.4	Performance on Benchmark Datasets	79
4.5	Summary	80
4.6	Publications	81
5.	Optimization and Extensions of the Structural Learning Rule for Dendritic Neurons	83
5.1	Spike Classification with Nonlinear Dendrites & Structural Learning	87
5.1.1	Spike-based Learning Scheme for Multiclass Classification	90
5.1.2	Rate-based Learning Scheme for Multiclass Classification	92
5.1.3	Ensemble Learning Scheme for Multiclass Classification	96
5.1.4	Adaptive Learning Scheme for Multiclass Classification	98
5.2	Results	100
5.2.1	Input Generation	100
5.2.2	Effect of Training Set Size	102
5.2.3	Comparison of Spike-based Learning and Rate-based Learning Schemes	102
5.2.4	Performance of the Ensemble Method	104
5.2.5	Performance of the Adaptive Learning Scheme	106
5.2.6	Test of Noise Sensitivity	107
5.2.7	Analysis of Dendrite Weights: Feature Maps	108

5.2.8	Boosting the Performance using Optimal Neuron Topology .	110
5.2.9	Comparison with Other Spike-based Classification Models .	111
5.3	Summary	113
5.4	Publications	114
6.	Combined Effects of Delay Adaptation and Structural Learning: Initial Explorations	115
6.1	Multicompartment Model of Nonlinear Dendrites	117
6.2	Theoretical Capacity of Multicompartment Model	119
6.3	Learning Algorithm	120
6.3.1	Scheme-2:	121
6.3.2	Scheme-3:	122
6.4	Results	123
6.4.1	Comparison of lumped vs multicompartment models: Effect of number of dendrites	124
6.4.2	Effect of delay per compartment	126
6.4.3	Comparison with the Tempotron rule	127
6.5	Summary	128
6.6	Publications	129
7.	Conclusion and Future Work	130
7.1	Conclusion	130
7.2	Future Work	133
7.2.1	Enhancements in Multicompartment Model	133
7.2.2	Weight Plasticity Combined with Morphological Learning .	134
7.2.3	Multilayer Architecture	134
APPENDICES		135
A.	Validity of the Reduced Models	136
B.	Classification of Event-based MNIST-DVS Dataset	138
BIBLIOGRAPHY		139

LIST OF FIGURES

Figure

2.1	Expected vs actual peak voltage responses to stimulation of synaptic sites on the same dendrite (Reproduced from Figure 4 in [1]) shown with circles. Rest of the curves show polynomial fits to the experimental data.	12
3.1	Delay-based learning model where N synaptic afferents receive incoming spikes fired at time x_i and create EPSP waveforms delayed by d_i , $i = 1, 2, \dots, N$. Spike delays $\mathbf{d} = (d_1, d_2, d_3, \dots, d_N)$ are modified such that membrane potential $V(t)$ crosses the V_{thr}	19
3.2	(a) Membrane potential $V(t)$ for a spike pattern. The horizontal line denotes the maximum value of $V(t)$, i.e. V_{max} at time t_{max} . (b) Probability density distribution of V_{max} for random spike patterns. V_{thr} is set as ΔV units greater than $V_{peak} = 10.2$ for memorization task and the two thresholds V_{thr} and V_{thr-} are set for classification task. (c) Determination of optimal threshold V_{opt} . V_{max} distributions for learnt (red) and new (blue) patterns are obtained. The training threshold V_{thr} used here is 11.7 corresponding to $\Delta V = 1.5$. The point of intersection of the two distribution curves gives $V_{opt} = 10.9$ (dashed vertical line).	23
3.3	Postsynaptic potential due to a spike arriving at a synapse given by kernel K (top). Derivative of K (bottom).	24
3.4	Membrane potential generated by a pattern in class 1 (top row) and in class 2 (bottom row) before and after training show how learning increases V_{max} for the class 1 while reduces it for the class 2. The increased synchrony for class 1 and decreased synchrony for class 2 are shown.	27
3.5	Probability density distribution of V_{max} for a set of 20, 50 and 100 random spike patterns before (blue) and after (red) training showing a clear shift in the peak of the distribution for trained patterns. V_{max} distribution for new patterns presented to the trained network is shown in black.	29

3.6	t_{max} distribution for 100 random spike patterns. The t_{max} values before (white) and after (black) training are spread across the entire pattern duration. $V_{thr} = 10.7$	29
3.7	Memory capacity of the model. The percentage of patterns recalled as a function of different number of patterns used to train the network. As the number of patterns used for training increases the fraction of patterns that the network learns reduces. The effect of learning with different V_{thr} values is also shown.	30
3.8	Total error given by the sum of false positive (FP) and false negative (FN) errors in the pattern memorization task as a function of P and the training threshold V_{thr}	31
3.9	Probability density distribution of V_{max} for 50 patterns learnt to be in class 1 (red) and in class 2 (blue). The effect of learning using different ΔV values on the V_{max} distributions for the two classes is shown. The V_{max} distribution for new unseen patterns presented to the trained network is shown in black. Dashed vertical lines denote the location of V_{peak}	32
3.10	Classification accuracy of the model. The network was trained to classify random patterns belonging to class 1 and class 2. The fraction of correctly classified patterns belonging to both classes is a function of the total number of patterns the network is trained on and the threshold V_{thr} used for each class.	33
3.11	Capacity of the model as a function of the load on the system. The network was trained on P patterns for different number of synapses $N = 50, 100$ and 200 , $\Delta V = 0$. The classification accuracy is plotted against $\alpha = P/N$. The capacity of the model reduces as α increases.	34
3.12	Memory capacity of the model in the presence of noise in t_{max} estimate. A jitter of $\sigma = 2, 5$ and 10 ms was added to the t_{max} values. The pattern memorization capacity drops as higher noise level is introduced by increasing σ . $V_{thr} = 10.7$	35
3.13	Memory capacity of the model in the presence of temporal noise. Gaussian noise with standard deviation 1.5 ms was added to all spike times. The ability of the model to recognize these noisy patterns reduces by $5-15\%$ from its memorization capacity for unperturbed spike patterns.	36
3.14	Memory capacity of the model for incomplete spike patterns. The capacity of the network to recall 50 patterns consisting of spikes missing at some of the $N = 100$ afferents is shown. $V_{thr} = 11.7$	37
4.1	Conversion of binary patterns of spike inputs into binary vectors. An example of how rate encoded spike trains (top left) and single spikes (bottom left) are mapped to the same binary vector (right) is shown.	43

4.2	(a) The major difference between a traditional linearly summing neuron and our neuron with nonlinear dendrites is the lumped nonlinear function $b()$ for every dendritic branch. (b) Architecture of the 2-class pattern classifier that consists of a winner take all (WTA) to compare the outputs of two neurons trained to respond to patterns of classes (+) and (-) respectively by using the Heaviside function $g()$ to generate the output of the model.	44
4.3	(a) The input and output of a dendrite (shown within ellipse) when a rate encoded spike pattern is presented. Time waveform of the input current, $I_{b,in}$ and the output current, $I_{b,out}$ of the branch are shown, where branch nonlinearity $b()$ is a quadratic function. (b) For an input pattern in class (+), the (+) neuron has a higher spiking rate. These output spike trains are input to a WTA which decides the winning neuron.	46
4.4	An example of (+) and (-) neurons with one dendrite each and a squaring nonlinearity. Connections on the branches with 4 synapses each are shown for binary input vector $x = [x_1 \ x_2 \ x_3]^T$	49
4.5	Dendritic nonlinear function $b(z)$ for different values of x_{thr} and b_{sat}	50
4.6	(a) Heaviside step function $g()$. (b) g_{margin} function to enforce a margin δ around the classification boundary.	51
4.7	Spike-based learning for patterns belonging to the two classes: membrane voltage ($V_m(t)$) trace of the (+) neuron is shown for correctly classified patterns for $y^d = 1$ (a) and $y^d = 0$ (b) and for misclassified patterns before learning (c and e) and correct outputs after learning (d and f).	61
4.8	Classification error as measured by the MAE of the L-neuron and NL-neuron. The plot shows the MAE as function of learning iteration for 500 and 1000 training patterns, number of dendrites, $m = 20$	66
4.9	R_W values of the weight correlation matrix (x-axis) for (+) and (-) neurons with $m = 10$ dendrites (y-axis) before and after learning. The histograms plotted by summing the correlation values for each input pair for all the dendrites are shown at the bottom of each R_W matrix.	67
4.10	MAE as a function of the number of dendrites, m . The L-neuron (circles) has higher classification errors than its nonlinear counterpart (squares). The optimal choice of m for total number of synapses, $s = 200$ is $m = 50$. The dashed curve shows the theoretical capacity of the neuron for the same values of m and s	68

4.11	Reciprocal of capacity of a neuron as a function of number of synapses per branch (k) for varying number of branches (m). Capacity increases as the total number of synapses $m \times k$ increases.	70
4.12	MAE as a function of k , the number of synapses per branch in a NL-neuron for binary inputs. 500, 700 and 1000 patterns were trained on neurons with $m = 10, 20$ and 50 branches. The errors decrease as the number of synapses increases.	71
4.13	Testing performance for spike patterns as a function of k . The errors are much higher than that for the corresponding binary input vectors (Figure 4.12(c)). .	71
4.14	Determination of δ for $g_{margin}()$ function. The distribution of α for misclassified spiking inputs. $P = 500$ and $m = 20$	73
4.15	MAE as a function of k for the $g_{margin}()$ function. The training was done with δ_0 set to 25 and the input patterns were tested using $g()$ function. The performance improves for increasing k and m	74
4.16	Architecture showing two neurons generating the output of the model in response to current differences ($I_{PDT}(t) - I_{NDT}(t)$) and ($I_{NDT}(t) - I_{PDT}(t)$). . .	75
4.17	Performance on spike patterns improves as (a) differences in $I_{PDT}(t)$ and $I_{NDT}(t)$ used as input currents for the integrate and fire neurons to counter the high ‘m’ effect; (b) $g_{margin}()$ used to improve the noise tolerance of the learning algorithm; and (c) inclusion of a dendritic nonlinear function $b_{leak}()$ improves the performance at larger k values. $P = 1000$, dashed plots correspond to the errors for binary vectors trained using the $g_{margin}()$ function. . .	76
4.18	Testing performance for single spike patterns ($P = 1000$) as a function of k for $m = 20$. The training (dashed) errors are about 2% less than the testing errors for $\Delta \leq \tau_f$	76
4.19	Membrane voltage $V_m(t)$ of (+) and (-) neurons when a class (+) pattern is presented. The pattern is not classified correctly since the (+) neuron fails to fire.	77
4.20	Classification performance of the BSTDSP algorithm. (a) Spike learning results when trained with (squares) and without (circles) margin. $P = 500$ patterns of single spikes, $m = 20$ (b) Comparison with RMWM learning (circles). The training errors are shown in solid and testing errors with dashed lines. . .	78

5.1	(a) A neuron consisting of nonlinear dendrites characterized by the lumped nonlinear function $b()$ for every dendritic branch. It forms k binary connections from d input afferents on m dendrites. (b) Architecture of a multiclass pattern classifier consisting of nonlinear neurons receiving inputs from PDT and NDT and the neurons connected to a winner take all (WTA) to compare the outputs of the N_C classes. The output of the classifier is equal to 1 for the class corresponding to the highest output o^μ	86
5.2	Ensemble classifier combining the intermediate class-specific outputs o_n^μ to compute the combined class outputs O^μ , which are compared by the WTA to generate the final classifier output $\hat{y}$	97
5.3	Dendritic classifier is trained adaptively by adding dendrites (step-1) and keeping the number of synapses/dendrite (k) as a constant for all neurons. The total number of synapses for the μ^{th} neuron learned in this manner (s_{adapt}^μ) is fixed and the corresponding optimal topology (m_{opt}^μ, k_{opt}^μ inset plot) is theoretically determined (step-2). The length of sides of the rectangle denote m and k values and the area represents the total number of synapses, s	100
5.4	Training and testing error rates for rate-based learning as a function of the training set size. Test performance is measured on 10,000 jittered single spike input patterns with $\Delta = 10$ ms. Each data point is obtained by averaging over 3 training simulation trials and testing on 10 randomly drawn test sets for each learned network; $m = 10$ and $k = 10$	102
5.5	(a) Comparison of performance of rate-based and spike-time-based learning schemes. The test error rates for the spike-based learning rule are about 1.5 times higher than the rate-based test errors; (b) Comparison of test error rate for $P = 200$ patterns learned with rate-based (blue line) and spike-based (red plot) methods as a function of τ_{post} . Each data point is obtained by averaging over 3 training simulation trials and testing on 10,000 single spike inputs with jitter $\Delta = 10$ ms; $m = 10$ and $k = 10$	104
5.6	(a) Ensemble error rate as a function of the number of classifiers combined, N , for 10,000 test samples; $m = 2$ and $k = 10$. (b) Ensemble error rate as a function of the total number of dendrites used, M , showing that a moderate number of dendrites per dendritic tree ($m = 8$) gives reasonably good performance and there is no significant improvement when m is further increased. Training done on $P = 5000$ patterns randomly drawn from the MNIST samples. Each data point is obtained by randomly selecting a subset (N) of the total number of learned classifiers for each case of m (37 for $m = 2$). This process is repeated 10 times and the average of the test errors of the combined classifiers is computed.	105

5.7	(a) The number of dendrites learned using adaptive scheme-1 (top) and the resulting confusion matrix (bottom) showing higher accuracy for easier classes (0, 1, 6) and lower accuracy for difficult digits (8, 9). (b) Number of dendrites learned using adaptive scheme-2 (top) demonstrating all the class neurons adding dendrites when required. Confusion matrix (bottom) showing improved accuracy for all the classes. Training set consists of 20,000 binary digit samples and testing done on 10,000 jittered single spikes ($\Delta = 10$ ms). Model was initialized with $m = 5$ dendrites for all the classes.	107
5.8	Error rate of ensemble consisting of NLD classifiers trained on $P = 20,000$ MNIST samples using the adaptive scheme-2. The ensemble accuracy on 10,000 test patterns is computed for binary vectors and spike inputs with different levels of noise. The accuracy obtained for single spike inputs with time window size $\Delta = 20$ ms is 96.1% by combining 5 classifiers. For Poisson spike inputs, $f_{high} = 250$ Hz and $f_{low} = 1$ Hz. Each data point is obtained by randomly selecting a subset (N) of the total number of learned classifiers (5). This process is repeated 5 times and the average of the test errors of the combined classifiers is computed.	108
5.9	Weight maps of all dendrites of neurons belonging to a particular digit class (top row). Maps in the remaining rows show individual dendrite weights with completely or partially learned features of input digits.	109
5.10	(a) Theoretical capacity of neuron corresponding to digit ‘5’ as a function of number of dendrites (m). The number of dendrites learned adaptively is m_{adapt}^5 and the optimal number of dendrites corresponding to maximum capacity is m_{opt}^5 . (b) Adaptively learned and optimal m values for digit ‘7’. (c) Performance comparison of adaptively learned network (blue) with theoretically determined optimal network (red). Training was done on 20,000 binary digit samples and test performance was measured on 10,000 binary inputs (solid) and jittered single spike input patterns with $\Delta = 10$ ms (dashed). . . .	110
6.1	Architecture of the multicompartment model. (a) A neuron with multiple non-linear dendrites (m), each with multiple compartments (n_d) forming k binary sparse connections from d input lines ($k \ll d$). (b) Binary classifier consisting of PDT and NDT, generating the output y	118
6.2	Random latency spatio-temporal patterns as generated in [2]. P patterns consist of d afferents firing once randomly during stimulus duration T_s	124
6.3	(a) MAE as a function of the number of dendrites (m) for different learning schemes. Inverse capacities are plotted in dashed curves. (b) MAE as a function of the delay per compartment (t_d) for different values of postsynaptic time constant τ_f and $m = 50$. Inverse capacity C_{multi}^{-1} plotted on right-hand y-axis (dashed blue). Each data point is obtained by averaging across 3 simulation trials.	126

A.1	Average dendritic current predicted using the reduced model (y-axis) plotted against actual mean current (x-axis) for different values of $f_{high} \times \tau_f$ and different number of synapses on a dendrite.	137
-----	--	-----

LIST OF TABLES

Table

4.3	Comparison of Δc_{ij} for the BSTDSP rule and RM.	62
4.4	Parameters for Spike-Based Model	65
4.5	Parameters for BSTDSP Model	65
4.6	Specifications of Benchmark Binary Datasets.	80
4.7	Performance Comparison of SVM, ELM and Dendrite Algorithms on Benchmark Datasets.	81
5.3	Parameters for Rate-based Multiclass Model	101
5.4	Parameters for Spike-based Multiclass Model	101
5.5	Comparison with Spike Classifiers on MNIST Data	112
6.3	Parameters for Multicompartment Model	124
6.4	Performance Comparison with the Tempotron Model	128
B.1	Performance on MNIST-DVS dataset	139

Chapter 1

Introduction and Motivation

1.1 Motivation and Background

The broad goal of machine learning is to design algorithms exhibiting automated learning and decision-making behaviour with widespread applications including face recognition, medical diagnosis, fraud detection and weather prediction. Since the human brain excels at tasks such as recognition and classification, it is crucial to try to understand the neural basis of such performance. Moreover, brain operates in low power with a consumption of about 20 W and can perform well even under noisy and imprecise environments. Hence, these features of brain operation can be combined to achieve good performance along with optimized implementation. *The goal of this thesis work is to design brain-inspired machine learning algorithms for pattern classification with the feasibility to be mapped onto hardware.* To pursue this goal, I have combined the neurobiological mechanisms of presence of active ionic conductances in dendrites, discrete synaptic states and sparse cortical connectivity as ingredients of a biologically realistic learning rule. This learning rule is based on the modification of neuron morphology which includes synaptic connectivity diagram, location of synapses on dendrites and the number of dendrites. With this novel learning scheme, I have pursued a shift in the neural network learning paradigms from the traditional weight modification approaches. This proposed method has important implications for future hardware implementation. In this chapter, I first discuss the biological inspiration for the classification algorithms proposed in this thesis and the alternative

learning schemes used in this work. Next, I delve into the motivation for implementing these learning algorithms on dedicated hardware platforms and finally examine the design features of these biologically motivated algorithms desired from the perspective of hardware realization.

Most of the machine learning algorithms have not considered the role of dendrites in neuronal information processing and its implications for the computational power of neurons. From a neurobiological perspective, the signal conduction along dendrites is governed by the passive filtering properties as well as active processing leading to the amplification of synaptic inputs on a dendrite. The traditional neural network models are based on the assumptions that dendrites sum up their synaptic inputs independent of their locations on the dendrite and model a neuron as “point neuron” with single integrative node at the soma where dendrites perform no additional processing except collecting and channeling the synaptic inputs to the soma. This classical view of dendritic integration has been challenged by several modeling and neurophysiological experiments. Rall’s work showed that dendrites act as delay lines, exhibiting spatially asymmetric synaptic integration effects [3] which therefore depends on the location of synapses [4–6] This mechanism contributes to direction selectivity as shown in retinal ganglion cells studied by [7]. Further, experimental evidence has indicated the presence of a variety of voltage-gated ionic conductances in the dendrites of pyramidal neurons. These active ionic mechanisms in the dendrites can trigger local dendritic spikes [8–10] leading to compartmentalization in the dendrites where clusters of coactive inputs can connect [11]. Neuron models designed to exploit this active nonlinear processing in the dendrites have much higher computational power [12] and hence, provide the possibility of attaining enhanced classification performance. This thesis work examines the combined effect of passive and active dendritic processing on the classification performance of neuron models which learned the task using alternative approaches to weight-based learning. These concepts are introduced below.

The main focus of neural networks research has been on weight-based learning algorithms which involve activity-dependent modification of synaptic weights of connections between presynaptic and postsynaptic units. This approach involves encoding of learned information by increasing or decreasing the weights of existing synaptic connections. These learning paradigms

have not considered the modification of synaptic wiring diagram as a source of long-term information storage. In such a scenario, existing connections can get eliminated while new connections can be formed. These changes require structural plasticity involving formation and elimination of synapses which provides an alternative to the weight plasticity. This ability to modify connectivity matrix in a sparsely connected network results in a large number of possible circuits to store learned information [13]. Another mechanism underlying learning in neural networks exploits the temporal encoding of information. Spiking neuron models can solve complex pattern classification problems by utilizing the temporal information encoded in these patterns. One learning strategy for spiking neurons involves tuning the ‘delay’ parameter, which in biological neural networks represents the time elapsed in transmitting information from one neuron to the other and can arise due to processing occurring in synapse, dendrites and axon; and has no analogue in the traditional neural networks. In this thesis work, I have explored these novel biologically relevant schemes of learning in neural networks which have the added advantage of suitability for implementation on hardware platforms.

The need to map these learning algorithms onto hardware systems can be realized by considering that the traditional approach to implement the machine learning algorithms is based on software simulations, which utilize double-precision, noise-free computations that are highly reproducible. However, simulating large networks requires high computational power and vast clusters of computers, which becomes a limitation in a resource constrained environment. The advent of Very Large Scale Integration (VLSI) technology has led to an exponential reduction in transistor size which has enabled the creation of much more complex microprocessors. This has resulted in an explosive growth in computational power, required by the traditional computing techniques to process large amounts of raw data. However, in a bid to make faster computers, these technologies suffer from high power dissipation and statistical variability of the devices which is exacerbated due to their shrinking sizes. Hence, alternative “hardware-centered” approaches of implementing the learning algorithms on dedicated hardware systems are being investigated to overcome these limitations. The inspiration for such hardware designs comes from the fact that human brain is millions of times more power-efficient and far more

compact than computers; it can perform massive high-speed robust computations in parallel; and unlike computers, it exhibits fault tolerance and does not require programming.

This led to the advent of neuromorphic technology, aimed at mimicking the architectural and computational principles of the brain [14], to design chips which can exploit the low power spike-based operation of human brain since this event-driven way of processing leads to minimal power dissipation during long inactive periods [15, 16]. These neuromorphic circuits composed of spiking neural networks also benefit from their large computational power [17] and hence, can be used to solve complex pattern recognition problems that are difficult for standard machine learning methods. Conversely, the neuromorphic systems can be used to understand the working principles of brain by testing ideas about brain function that span across the levels of ion channels to neural micro-circuits and systems. This approach to emulate the neuronal function can enable us to design power-efficient, event-driven, compact hardware systems with spike-based learning algorithms implemented to provide real-time, intelligent solutions for several applications like robotics, Brain-Machine Interface (BMI) and Internet of Things (IoT) sensors. These applications include Dynamic Vision Sensor retina [18] and the AEREAR2 cochlea [19] spike-based sensors that have been developed recently for processing visual and auditory information respectively. One of the basic requirements of such neuromorphic systems for performing complex cognitive tasks is to be able to classify spatio-temporal spike patterns generated by biological neurons as in a BMI application or by event-based sensors as in artificial retina [20]. Hence, spike-based neuromorphic classifiers can be interfaced with these sensors. However, there has been slow progress in the development of these spiking algorithms, which can solve real-world problems with accuracy comparable with the state-of-the-art. A major cause of this problem is the statistical variations in VLSI devices which reduce the accuracy of the synaptic weights leading to reduced classification accuracy than that obtained with computer simulations using double-precision numbers. Another problem with the hardware mapping of spike-based algorithms is the use of large number of synaptic weights, which renders these algorithms impractical for VLSI implementation.

To mitigate the effect of device mismatch, there is a need to design models and algorithms

which either exhibit tolerance to variations or use low-resolution synapses for robust implementation in VLSI systems. As shown in a study, a neuromorphic classifier achieved performance comparable with standard machine learning linear classifier and exhibited tolerance against variability by using population coding [21]. A limitation of this model is the large number of high resolution weights used to attain the reported performance. Similarly, the spike-based classifiers consisting of Restricted Boltzmann Machine (RBM) constructed with integrate and fire neurons, use a large number of recurrent synaptic connections [15, 22]. **However, in a recent study, spiking Deep Belief Networks (DBNs) have been shown to tolerate very low bit precision of weights, down to four bits, which results in more than 50% of the synapses getting eliminated with very little loss in accuracy [23]. Hence, the use of such spiking neuron models with low resolution weights and sparse synaptic connections can be an ideal candidate for hardware implementation.** However, the use of low resolution synapses with only one or two bits of long-term information has severe implications for the storage capacity of networks working as classifiers or associative memories [24]. Therefore, in order to take advantage of the simple and robust implementation of networks with binary weights, a solution to deal with the reduced storage capacity of binary synapses is needed. The use of active dendritic mechanisms in such networks gives rise to nonlinear units in addition to the somatic nonlinearity thereby increasing the computational capacity of the network. Moreover, since learning binary weights corresponds to finding the optimal connectivity pattern, hence a structural plasticity rule is well suited for learning in networks of binary synapses. The structural learning scheme is also appealing from the perspective of implementation in neuromorphic systems using the asynchronous communication protocol called the (AER) [25–28]. An AER-based system can store connectivity information in an off-chip digital memory which gives the user flexibility to reconfigure the network connectivity.

1.2 Major Contributions of the Thesis

In summary, this discussion has focused on the design of a spike-based learning algorithm which combines the biological inspiration to emulate the performance of human brain with the

feasibility of power-efficient hardware implementation. In this thesis, a bio-plausible, hardware-friendly learning rule was developed which utilized the mechanism of correlation-based structural plasticity to choose the best connectivity pattern for a given classification task in concert with the delay modification to learn the location of connections on a dendrite. Margin-based learning was introduced to enhance the generalization performance of the model. A spiking neural network architecture was presented which consisted of dendrites with passive filtering and models of active ion channels and further employed sparsely connected low resolution synapses. In this work, two forms of learning approaches were adopted to train the dendrite-specific connections of spiking neurons, 1) spike-based learning which involved a variant of spike-timing dependent plasticity (STDP), and 2) rate-based learning which used mean rate inputs to reduce training time and then transferring the learned connections to a spiking neuron model. Finally, an adaptive version of the classification algorithm was developed which involved learning the number of dendrites along with the connectivity matrix of these dendrites, thereby allocating synaptic resources as per the complexity of the given problem.

1.3 Organization of the Thesis

The thesis is organized as follows.

Chapter 2 gives a literature review to describe the underlying concepts of this thesis work and summarize the research on spike-based neuromorphic classifiers.

In chapter 3, a model for spike pattern memorization and classification based on delay adaptation is presented. The model architecture comprises a spiking neuron with passive filtering effects. A learning algorithm is proposed which modifies the temporal delays arising due to filtering effects and utilizes this learning mechanism to memorize and classify spatio-temporal patterns.

In chapter 4, a spike-based pattern classification algorithm based on structural plasticity is discussed. This model uses multiple dendrites endowed with active ionic mechanisms and learning based on modifying connections instead of weights. The connectivity matrix consists of sparse synaptic connections with binary weights.

In chapter 5, extension and enhancements of the model described in Chapter 4 are presented. The extension to multiclass architecture is discussed. Further, the use of an adaptive scheme to learn the number of dendrites in the network and an ensemble method to improve the generalization by combining several dendritic classifiers is demonstrated.

In chapter 6, the combined effect of structural learning and dendritic delay learning in a spiking neuron model is demonstrated. The model utilized both passive and active dendritic mechanisms to process neuronal information in a multicompartment model consisting of non-linear dendrites.

Chapter 7 gives a conclusion of the thesis and discusses the potential avenues which can be explored in future studies.

Chapter 2

Literature Review

In this chapter, a review of the methods used to solve complex pattern classification problems will be provided and the neurobiological relevance of these models which led to the development of several biologically motivated higher-order models will also be presented. Further, the learning mechanisms in these models which provide alternatives to the traditional weight learning algorithms will be discussed.

2.1 Higher-Order Networks for Nonlinear Classification Problems

A wide range of machine learning methods have been proposed to determine the decision boundaries in a pattern classification task. A simple feedforward neural network consisting of a single layer of threshold logic unit (TLU) like perceptrons, can solve only linearly separable pattern classification problems. The perceptron has limited capability since many real-world problems are nonlinearly separable and may require higher-order combinations of some of the input components to solve the problem. One way to perform nonlinear classification is by applying nonlinear functions of the input to a single adaptive element. Specht did some of the earlier work in this area, which involved using polynomial functions to determine nonlinear decision surfaces [29]. This method known as the polynomial discriminant method (PDM) implements a non-parametric classifier by estimating probability density function for each input class as a single polynomial, where the weights for cross-products and power terms are determined using a non-iterative process. Finally, it uses the Bayes decision rule to perform classification.

Another method used to generate higher-order functions of the input components is a functional link network (FLN). FLNs are single-layer networks, which estimate an unknown function $\sum_i w_i \phi_i(x)$, where x is the input vector and $\phi_i(x)$ are predefined basis functions such as products of inputs or sinusoidal functions [30]. These methods have the advantage that the networks can realize a wide variety of nonlinear functions by the intrinsic high-order mapping of the inputs, which is achieved by adapting a single element. However, the choice of nonlinear preprocessing function is critical to solving a particular problem and therefore, requires *a priori* knowledge of the problem at hand [31]. Moreover, the number of terms required during training grows rapidly with the number of inputs. Polynomial neural networks (PNN) based on the evolutionary strategy given by Ivakhnenko [32] have been developed to combat this problem [33]. This approach involves generating populations or layers of neurons and then selecting those giving the best classification results and hence the polynomial of optimum complexity.

Another type of nonlinear classifier consists of multiple adaptive elements. A multilayer network can be considered as a single-layer network with trainable preprocessors that learn the higher-order terms needed for a nonlinear classification problem. A sigma-pi network consists of a hidden layer which computes the product of inputs, followed by a summing unit [34]. The output of a “sigma-pi” unit is given by the following equation:

$$y = \sigma(w_0 + \sum_j w_j x_j + \sum_{j,k} w_{jk} x_j x_k + \sum_{j,k,l} w_{jkl} x_j x_k x_l + \dots) \quad (2.1)$$

where $\sigma()$ is a sigmoid-like activation function, x_j is the j th component of input vector x , $w_{jkl} \dots$ is a weight which captures the correlation between product of input components $x_j, x_k, x_l, \dots$ and the output of the unit; and w_0 is an adjustable threshold. These higher-order networks can be trained using fast learning schemes like Hebbian learning rule or perceptron type learning. However, a disadvantage of this approach is that there is a combinatorial explosion of terms and hence weights needed to accomodate all possible 2^N correlation terms among N inputs, where $x \in R^N$. This exponential increase in the number of higher-order terms with the input dimension limits the applicability of this approach. A method used to reduce the complexity of higher-

order networks involves using *a priori* knowledge of the problem such as geometric invariances [35, 36] to restrict the number of units or terms needed to achieve a desired degree of accuracy. In these studies, invariances were implemented using higher-order networks which eliminated the terms incompatible with the invariance constraint, leading to a simpler architecture of the network.

Durbin and Rumelhart [37] used product units which represent any generalized higher-order term and can learn which one to represent. The output of the product units is given by $y = \prod_{i=1}^N x_i^{p_i}$ where the value of p_i , which can take fractional or negative values, is learnt using gradient descent. Therefore, product units provide much more generality in terms of allowing ratios as well as polynomial terms and can be incorporated in layered networks along with thresholded summing units. Another higher-order network is the pi-sigma network (PSN) which computes the *products of sums* of input components instead of *sums of products* as in sigma-pi networks [38]. A PSN overcomes the problem of combinatorial increase in the number of weights associated with higher-order networks while maintaining the capability of nonlinear mapping and fast learning. The neurobiological relevance of these models can be understood by considering that the multiplicative interactions among neighbouring synaptic inputs can underlie local dendritic computations such that these local dendritic regions act as product units whose outputs are then summed at the soma.

2.2 Biologically Motivated Higher-Order Networks

Mel and his group presented several computational studies to elucidate the role of dendrites which can be incorporated in neuron models rendering them computationally more powerful. Mel hypothesized that a single neuron acts like a powerful multilayered network which performs nonlinear operations within spatially separated dendritic branches [39]. The idea of dendritic segments serving as functional subunits [40] has important implications for the computational capacity of cortical tissue, which was demonstrated by obtaining much larger memory capacity for neurons with nonlinear subunits as compared to that with linear subunits [12]. In a more recent study, a two-layer model was proposed [41] in which different sets of inputs targeting

functionally distinct dendrites are first processed nonlinearly and then the dendritic outputs are linearly integrated at the soma. These results were also obtained in [42] to show that the response of a pyramidal cell model calibrated with experimental data can be described as a sum of nonlinear subunit responses.

The idea that the input-output behaviour of a complex dendritic tree can be reduced to a simple algebraic formula consisting of sum-of-squares is supported by an earlier compartmental model for disparity tuning in complex cells of primary visual cortex [43]. This model consisting of simple-cell-like subunits mapped onto active dendrites, behaves like a hypothetical cell with squaring subunits. In another study, the firing rate of a complex biophysical pyramidal cell model was consistent with the mean rate predicted by an abstract two-layer neural network with sigmoidal subunits representing the thin terminal dendrites [41]. Hence, these and other studies [44, 45] suggest that from a biophysical perspective, a pyramidal cell with varying density, spatial distribution and kinetics of voltage-gated ion channels can be well approximated by an abstract model of sum of nonlinear subunits.

Electrical recordings from pyramidal neurons, the principal cell type in cortex, have shown that dendrites contain a large number and types of voltage-gated ion channels which contribute to active responses including active backpropagation of axonal spikes into the dendritic tree and dendritic spikes mediated by Na^+ and/or Ca^{2+} channels [10] as well as N-methyl-D-aspartate (NMDA) receptor channels [46]. In order to investigate the particular form of nonlinear function representing the input-output behaviour of a dendrite, we can look at the experimental results from a study in which confocal imaging was combined with focal synaptic stimulation to examine the rules of subthreshold synaptic integration in thin dendrites of pyramidal neurons [1].

Figure 2.1 shows the measured and expected peak responses when two nearby sites on the same branch were stimulated simultaneously, where the expected response is the arithmetic sum of the responses to individual stimulations. The solid plot marked with circles corresponds to the measured responses. This curve has been reproduced from the data presented in Figure 4 in [1] for single-pulse stimulation. Therefore, experimental evidence suggests that the dendritic response to synaptic inputs on that branch is governed by a saturating expansive nonlinearity. In

Figure 2.1: Expected vs actual peak voltage responses to stimulation of synaptic sites on the same dendrite (Reproduced from Figure 4 in [1]) shown with circles. Rest of the curves show polynomial fits to the experimental data.

order to determine the form of subunit nonlinearity, polynomials have been used in our work to fit the experimental data because it is easier to implement polynomial functions in a hardware system. The polynomials used are $y = \beta x^2$, $y = \beta x^3$ and $y = \beta x^5$ with saturating effects, where the constant β was chosen to get the best fit. As shown in Figure 2.1, the quadratic function provides the best fit as compared with degree 3 and degree 5 polynomials tested here. Hence, for the few data points obtained from biological measurements, the squaring function can be considered as a valid choice for the dendritic nonlinearity used in this thesis work.

2.3 Learning Mechanism in Higher-Order Networks of Dendritic Neurons

2.3.1 Synaptic Clustering on Dendrites

Several computational models have hypothesized that synapses on a dendrite are activated in clusters [34, 44, 45, 47]. Mel and Koch [47] argued that the output of a cluster of synaptic

inputs on a dendrite can be AND-like, i.e. selective for coactivation of all the inputs. Therefore, threshold operation can act as the output of a dendrite and moreover, it is more biologically plausible for modeling dendritic mechanisms than pure multiplication. Mel proposed an abstract model neuron called the “clusteron” which uses Hebbian learning to map low-order polynomial functions of inputs by selecting similarly activated synaptic contacts from a set of afferent axons onto a dendritic subunit [44].

Several experimental studies have also demonstrated that clustering of synapses on dendritic branches might be a substrate for learning. The first demonstration that synaptic clustering might be a substrate for learning was given in the barn owl auditory system [48]. The authors reported that behavioural experience drives changes in the clustering of axo-dendritic contacts. A related study has shown that new spines emerge in clusters along dendrites as a result of the repetitive activation of the cortical circuitry during motor learning [49]. Synaptic clustering was also reported from a calcium imaging study which demonstrated that locally convergent inputs in spontaneously active networks lead to frequent synchronization of adjacent spines thereby facilitating dendritic compartmentalization [50]. The mechanism of nonlinear dendritic integration leads to regenerative events including dendritic spikes, which have a stronger influence on neuronal firing than the activation of synapses located on different dendrites [51]. This evidence suggests clustering of coactive inputs within a dendritic subregion, which can lead to compartmentalization of various forms of signalling in dendrites, like electrical, biochemical and cellular [11]. A recent study showed activity-dependent fine-tuning of synaptic inputs having temporally correlated activity in a developing neuron [52].

In addition to the structural clustering of synapses on dendrites, a novel form of plasticity at the dendritic level is provided by the excitability of the dendritic branches. A study by [53] showed that experience in enriched environments can lead to compartmentalized increase in dendro-somatic coupling. A model based on branch-specific STDP rule was proposed by [54], where the weight change of a synapse depends on the local dendritic branch potential. This learning mechanism involved competition between dendritic branches which enabled a single neuron to solve the binding problem.

2.3.2 Structural Plasticity

The biological basis of synaptic aggregation on a dendritic branch can be understood by considering that a tangle of axons within a dendrite's receptive field facilitates the formation of clusters of coactivated synapses on local regions of dendritic arbor [47]. The process of constructing these synaptic clusters involves alterations to the cortical 'wiring diagram' which requires structural plasticity at the axo-dendritic interface. The mechanism of structural plasticity includes synaptogenesis, axonal and dendritic growth and remodeling, and also retraction and reformation of dendritic spines [13, 55]. Several experimental studies have provided evidence for the formation and elimination of synapses in the adult brain [56, 57], including activity-dependent pruning of weaker synapses during early development [58]. Large-scale growth and remodeling of axonal and dendritic branches has been found to occur in adult brain within days [59]. In vivo imaging studies have shown that synaptic rewiring mediated by re-routing of whole axonal branches to different postsynaptic targets takes place in the mature cortex [60].

The structural changes provide an alternative form of plasticity in addition to synaptic weight changes associated with a classical Hebbian scheme. However, these two modes of long-term memory might operate on different timescales, where changes in synaptic efficacy could happen in seconds to hours while structural changes might operate over longer periods of days to months. The points in cortical space where an axon passes at sufficient proximity ($< 2 \mu\text{m}$) to a dendrite are called *potential synapses* such that a newly formed dendritic spine or terminal bouton will bridge the gap between them [61]. A measure of the number of potential synapses, known as the filling fraction of the network determines the extent to which synaptic rewiring can change the network structure. The information storage capacity associated with structural plasticity lies in this ability to change wiring diagram in a sparsely connected network, which provides a large number of functionally distinct circuits available to encode information [13] and therefore has important implications for the computational properties of the network [12]. A structural learning algorithm utilizing sparse connectivity can address the problem of weight explosion in higher-order neural networks which is an attractive feature for

implementing this learning network on hardware. This learning scheme can be rendered even more hardware-friendly by using low resolution synaptic weights.

2.3.3 Learning with Low Resolution Synapses

There is accumulating experimental evidence that biological synapses exist in only a small number of states which can be restricted to even two states [62, 63]. The use of synapses with only one or two bits of long-term information has severe implications for the storage capacity of networks working as classifiers or associative memories with capacity for binary synapses reducing by more than half as compared to the capacity using continuous-valued synapses [24]. However, the use of binary synapses provides a simple and robust solution to the problem of parametric variations due to mismatch in devices. Moreover, it is difficult to store the continuous-valued synaptic weights accurately for long time scales and hence, binary weights are important for hardware implementations of learnable networks.

Some studies have presented learning algorithms as biological solutions to deal with the reduced storage capacity of networks with binary synapses. A stochastic learning algorithm proposed in [24] was shown to learn with discrete weights with an arbitrarily high probability, where the number of neurons in the network is large enough. In a similar work, a stochastic spike-driven synaptic plasticity rule was used to train a network of binary synapses [64]. A pool of output neurons was used to calculate the classification accuracy by a voting mechanism which results in a large number of synapses being used. In comparison to these studies, a model based on structural plasticity rule for a sparsely connected network of binary synapses can lead to reduced number of weights. The use of dendritic nonlinearity can yield higher computational power thereby alleviating the problem of reduced capacity of binary weights. The use of passive dendritic mechanisms in neuron models can give rise to further enhancement of storage capacity by utilizing the spike timing information.

2.4 Delay Learning for Passive Dendrites

Most of the neural network research has primarily focused on the use of somatic nonlinearity utilizing the tunable weights of synapses. Hence, there is a need for modeling and understanding the computational properties of other components of our neurons: the axons and dendrites which have been largely ignored till now. This is also facilitated by recent experimental and computational work which has shed light on possible computational roles of these structures. The research on spiking neural networks has led to the emergence of a new paradigm in neural networks, which has garnered a lot of interest lately. Several recent studies have presented spiking neural networks to implement biologically consistent neural and synaptic mechanisms [65–67]. As shown by Izhikevich, spiking neural networks with axonal delays have immense information capacity [68]. These networks can exhibit a large number of stereotypical spatio-temporal firing patterns through a combination of STDP and axonal propagation delays.

Learning schemes based on modifying delays can be utilized to read out these firing patterns. Several biological mechanisms are known to change the transmission delay between two neurons, which has implications for learning and memory [69]. A spiking neural network with adjustable delays can compute a much richer class of functions than a threshold circuit with the same number of programmable weights [17]. This has spurred a renewed interest in the possible role of delays and has even led to analog VLSI implementations of delay models of axons [70, 71]. The earlier implementations [70, 71] need ‘N’ delay storage elements to memorize a single ‘N’ dimensional pattern. Several delay learning schemes have been presented for non-spiking networks [72–74] and some of them have been used in applications like word recognition [75]. In the context of spiking neurons and pulse coupled networks, delay adaptation was implemented in [76, 77] for biologically motivated networks using standard analog hardware elements. Hence, these research findings provide a promising opportunity to design powerful pattern classification algorithms utilizing delay modification.

Chapter 3

Spike Pattern Memorization and Classification using Delay Learning Architectures

List of Acronyms

Acronym	Description
EPSP	Excitatory Postsynaptic Potential
TTFS	Time-to-First-Spike
VLSI	Very Large Scale Integration
FP	False Positive
FN	False Negative

List of Variables

Variable	Description
N	Number of synaptic afferents

Variable	Description
T	Pattern duration
$\mathbf{x}$	Input spike pattern
d	Delay vector
$K(t)$	PSP kernel
V_0	Normalization constant for $K(t)$
τ, τ_s	Fall and rise time constants of PSP respectively
$V(t)$	Membrane voltage
V_{thr}	Threshold voltage
V_{max}, t_{max}	Maximum of $V(t)$, time of V_{max}
V_{peak}	Peak of V_{max} probability distribution
V_{opt}	Optimal threshold
y	Neuron output

In this chapter, the use of temporal coding for learning spike patterns is demonstrated. This learning scheme explores an alternative to weight-based approaches by utilizing delay-based learning. A computational model is presented which can learn spatio-temporal spike patterns by modifying the delays associated with the spikes arriving at a synaptic afferent. Compared to most earlier delay learning implementations [70, 71] that need ‘N’ delay storage elements to memorize a single ‘N’ dimensional pattern and adjust the delay parameters until all the impulses are coincident [76, 77], this method modifies only a few delays to memorize the most ‘salient’ part of each pattern. Here ‘salient’ refers to that part of a spatio-temporal pattern which has maximum synchrony or the largest number of coincidental spikes when observed at the soma of the post-synaptic neuron. This delay learning rule is similar to the one presented in [78] with two differences: it does not use the nonlinear membrane voltage dependent weighting term and involves a single time-based delay adjustment instead of an integral over a time period. More importantly, there is no discussion on the memory capacity of such networks in [78] with the

Figure 3.1: Delay-based learning model where N synaptic afferents receive incoming spikes fired at time x_i and create EPSP waveforms delayed by d_i , $i = 1, 2, \dots, N$. Spike delays $\mathbf{d} = (d_1, d_2, d_3, \dots, d_N)$ are modified such that membrane potential $V(t)$ crosses the V_{thr} .

authors having demonstrated the memorization of a single pattern only. The computational architecture of the delay learning model is described next, followed by the learning algorithm and finally the simulation results and a summary of the chapter.

3.1 The DELTRON Model

3.1.1 Network Architecture

Figure 3.1 depicts the architecture of the DELTRON that comprises an integrate and fire neuron at the output and N excitatory synapses that receive spiking inputs. The name is derived due to similarity in the learning rule with an earlier architecture called Tempotron [2]. Each of the incoming spikes create a delayed excitatory postsynaptic potential (EPSP) that gets linearly summed at the soma. In the bio-physical world, such delays could be attributed to synaptic processes [78] or dendritic propagation times [79]. If the summed membrane potential crosses a threshold, V_{thr} , the integrate and fire neuron generates a spike and resets the membrane voltage

[80]. We want to develop a learning rule that can modify the delays associated with each input so that only a certain desired set of P patterns can fire the neuron by making the membrane potential cross the threshold.

In this work, the application of the DELTRON in classifying or memorizing patterns in the case where there is exactly one spike on each input i at a random time x_i within a fixed time period T , i.e, $x_i \in [1, T]$, $i = 1, 2, \dots, N$, is considered. This case corresponds to applying the DELTRON to classify signals coming from a sensor employing Time-to-First-Spike (TTFS) encoding [81–83]. Time based encoding is becoming popular recently due to the reduced supply voltage (leading to lower voltage headroom) and increased speeds (leading to higher temporal resolution) in today’s deeply scaled VLSI processes; hence, the DELTRON will be very useful as a back end processor for all such temporal encoding sensory systems. Formally, the membrane voltage $V(t)$ can be expressed as a sum of the EPSPs generated by all incoming spikes as:

$$V(t) = \sum_i K(t - t_i) \quad (3.1)$$

where $K : R \rightarrow R$ is the EPSP kernel function, d_i is the delay of the i^{th} branch and $t_i = x_i + d_i$, $i = 1, 2, \dots, N$. The vector $\mathbf{x} = (x_1, x_2, \dots, x_N)$ constitute a spike pattern presented to the network. In this work, the fast rising and slowly decaying PSP kernel is used, given by $K(t) = V_0(\exp[-(t)/\tau] - \exp[-(t)/\tau_s])$, where τ is the synaptic current fall time constant and τ_s the synaptic current rise time constant. Our analysis is quite general and is applicable to other forms of the function K as well. As mentioned earlier, the integrate and fire output neuron elicits a spike when the voltage $V(t)$ crosses the threshold voltage V_{thr} . Let n_{spk} denote the number of spikes fired by the output neuron for the presentation of a pattern. Then, the operation of the neuron is described as:

$$\begin{aligned} \text{If } V(t) &\geq V_{thr}, \\ n_{spk} &\rightarrow n_{spk} + 1 \end{aligned} \quad (3.2)$$

The final output of the network, y , is a logical variable having a value of 1 if the pattern is recognized. This operation can be defined by:

$$\begin{aligned} y &= 1 \text{ if } n_{spk} > 0 \\ &= 0 \text{ otherwise} \end{aligned} \quad (3.3)$$

In other words, the pattern is declared as recognized if the neuron fires one or more spikes in the entire duration.

3.1.2 Input Pattern Space

It was earlier mentioned that the input spike pattern to the network is $\mathbf{x} = (x_1, x_2, \dots, x_N)$ where $x_i \in [1 \ T]$, $i = 1, 2, \dots, N$. For any real world inputs, there is a finite precision Δt at which an input x_i can change. Without loss of generality, let us assume that $\Delta t = 1$ so that x_i are integers (otherwise, we can always re-define $T' = T/\Delta t$ and have $x_i \in [1 \ T']$ be integers). Then, the total number of possible patterns, P_T is given by $P_T = T^N$. However, some of these patterns are very similar to each other and should be considered to belong to the same category. For example, a TTFS imager presented with the same scene will produce slightly different spike times on different trials due to noise inherent in such systems. Hence, two patterns x and y are defined to belong to the same category if:

$$\|x - y\|_\infty \leq s, s \in N \quad (3.4)$$

This results in a total of $(2s + 1)^N$ patterns in each category. In some applications, sometimes spikes may be missing on several inputs. To accommodate such cases, a pattern can be defined to also belong to a category if it is exactly same as one of the existing patterns in that category but has m missing spikes. Then, the total number of patterns in each category for cases where the maximum number of missing spikes is M is given by $P_{cat} = \sum_{m=0}^M {}^N C_m (2s + 1)^N$, where ${}^N C_m$ is the total number of ways in which m out of the total N spikes may be missing. This leads to the total number of categories being $P_C = P_T / P_{cat}$. This provides an upper bound on the total

number of patterns P that the network can be trained to recognize. For the parameter values we use ($s \ll T$, $N \gg 1$, $P \ll P_C$), the probability of two out of P randomly selected patterns to fall in the same category is extremely small. Hence, unless explicitly mentioned, whenever the capacity of the network to memorize or classify patterns is mentioned in the rest of the chapter, it means patterns belonging to different categories.

3.1.3 Choice of Threshold and the Two types of error

The DELTRON suffers from two types of errors when used in pattern memorization or classification tasks: false positive (FP) and false negative (FN). Suppose the network is trained to respond to patterns of class 1 by producing $y = 1$. In the classification case, it is additionally trained to respond to patterns of class 2 with $y = 0$. For pattern memorization, this step is not there and the entire population of patterns excluding those in class 1 act as a class 2 of distractor patterns—hence this is a more difficult task. Now, if a pattern of class 1 is presented during testing and the network responds with $y = 0$, it makes a FN error. On the other hand, if a pattern of class 2 is presented and the network produces $y = 1$, it is a FP error.

The threshold for the output neuron is an important parameter in deciding the balance between FP and FN errors. This is achieved by first computing the probability distribution of the maximum value (V_{max}) of membrane voltage due to random input spike patterns. As an example, Figure 3.2(a) shows V_{max} for a random spike pattern while Figure 3.2(b) depicts the probability distribution of V_{max} (parameters of this simulation are given in Section 3.3). As shown in Figure 3.2(b), V_{thr} is set as a value ΔV units higher than the value V_{peak} at which the probability distribution is maximum, i.e. $V_{thr} = V_{peak} + \Delta V$. It would seem that larger values of ΔV are better since it reduces FP errors. For example, setting V_{thr} to 16 would almost eliminate FP errors. However, this leads to the problem of ‘trainability’—the network can no longer successfully memorize a large number of patterns and makes many FN errors. In that case, the optimal threshold (V_{opt}) is that which minimizes sum of FP and FN errors as shown in Figure 3.2(c). V_{max} distributions are obtained after training the network (red) and when new unseen patterns are presented to the trained network (blue). The V_{max} value at the point of intersec-

Figure 3.2: (a) Membrane potential $V(t)$ for a spike pattern. The horizontal line denotes the maximum value of $V(t)$, i.e. V_{max} at time t_{max} . (b) Probability density distribution of V_{max} for random spike patterns. V_{thr} is set as ΔV units greater than $V_{peak} = 10.2$ for memorization task and the two thresholds V_{thr} and V_{thr-} are set for classification task. (c) Determination of optimal threshold V_{opt} . V_{max} distributions for learnt (red) and new (blue) patterns are obtained. The training threshold V_{thr} used here is 11.7 corresponding to $\Delta V = 1.5$. The point of intersection of the two distribution curves gives $V_{opt} = 10.9$ (dashed vertical line).

tion of these two distributions gives us the optimal threshold, V_{opt} . The strategy we have taken throughout our work is to keep a large threshold value during training ($V_{thr,TR} = V_{thr}$) and use the optimal threshold during recall.

For the classification case, there is another threshold $V_{thr-} = V_{peak} - \Delta V$ and the network is trained to reduce V_{max} lesser than V_{thr-} for patterns in class 2. Again, the optimal threshold is used during testing. For the sake of simplicity, in the rest of the chapter, the training threshold is referred to as V_{thr} with the implicit understanding that the optimal threshold is always used during testing.

3.2 Learning Algorithms

3.2.1 Delay Learning Algorithm for Pattern Memorization

The delay-based learning is similar to the learning in the tempotron [2] except that weights were modified in that case whereas delays are modified in our case. The training set for the

Figure 3.3: Postsynaptic potential due to a spike arriving at a synapse given by kernel K (top). Derivative of K (bottom).

memorization task consisted of P input spike patterns and the memory capacity of the network was computed as the number of patterns learnt. A spike pattern is said to be learnt by the network if output $y = 1$ in response to that pattern. The model was not trained on distractor patterns for this task. Since the training and testing patterns sets are the same in a memorization task, the problem of overfitting arises. However, in our case the use of low-pass kernel function K provides some degree of resilience to overfitting. The network learns spike patterns by modifying the N -dimensional delay vector $\mathbf{d}$. The learning algorithm consists of the following steps:

- (1) Delay vector $\mathbf{d} = (d_1, d_2, d_3, \dots, d_N)$ is initialized by choosing the value of each d_i from a uniform distribution between 0 and d_{init} .
- (2) In every iteration of the training process, a spike pattern is presented to the network.
- (3) $V(t)$ is computed for the pattern. Maximum value of $V(t)$; V_{max} is found.
- (4) If $V_{max} > V_{thr}$, the spike pattern is learnt. Delays are not modified.
- (5) If $V_{max} < V_{thr}$, time t_{max} at which $V(t)$ attains its maximum value (V_{max}) is found.
- (6) The error function is chosen as $E = V_{thr} - V(t_{max})$ since a pattern is learnt if $V(t)$ crosses V_{thr} at least once and t_{max} is the time when it is easiest to do it. Δd is calculated using

gradient descent on the error function as shown below.

$$E = V_{thr} - V_{max} \quad (3.5)$$

$$\begin{aligned} &= V_{thr} - V(t_{max}) \\ \Delta d_i &= -\frac{\partial E}{\partial d_i} = \frac{\partial V(t_{max})}{\partial d_i} \\ &= \frac{\partial [\sum_j K(t_{max} - x_j - d_j)]}{\partial d_i} + \frac{\partial V(t_{max})}{\partial t_{max}} \frac{\partial t_{max}}{\partial d_i} \\ &= -K'(t_{max} - x_i - d_i), \text{ since } \frac{\partial V(t_{max})}{\partial t_{max}} = 0 \end{aligned} \quad (3.6)$$

where K' indicates the derivative of K . Figure 3.3 shows the PSP kernel K and its derivative.

- (7) Delays are modified according to $\mathbf{d} = \mathbf{d} + \eta \Delta \mathbf{d}$ only if the number of learnt patterns increases, where η is the learning rate. The boundary values of delay are 0 and T .
- (8) If the number of patterns learnt by the model does not increase in 20 consecutive learning iterations, then it is assumed that the algorithm has encountered a local minimum. In an attempt to escape the local minimum, delays are modified in spite of no change in the number of patterns learnt by the network. The last minimum which gave the highest number of learnt patterns is remembered.
- (9) Learning consisting of steps 2-8 is stopped if any of the following conditions occurs
 - (a) All patterns are learned completing the learning process, which is the more frequent outcome when trained on small number of patterns.
 - (b) 100 local minima are encountered, which refer to 100 continuous instances of learning algorithm getting trapped in a status of no change in the learning, after which there is no further increase in the number of learnt patterns. This is the more frequent outcome when large number of patterns are being learned. The delays corresponding to the best minimum are the final learnt delays.

The DELTRON maximizes its memory capacity by modifying only *some* of the delays preferentially over the others. This choice is guided by the spikes arriving within a time window before t_{max} . The length of this window is governed by the kernel $K(t)$.

3.2.2 Delay Learning Algorithm for Pattern Classification

The delay-modification learning scheme has also been utilized for training our model on a classification task. Two sets of random spike patterns were generated, each set consisting of P patterns. Each spike pattern $\mathbf{x} = (x_1, x_2, \dots, x_N)$ consists of spike times x_i randomly drawn from a uniform distribution between 1 and T as in the first experiment. These two training sets of spike patterns were arbitrarily assigned to class 1 and class 2. The network was trained such that the output neuron fires at least one spike (network output $y = 1$) in response to the patterns of class 1 and fails to fire for patterns in class 2. The learning algorithm is similar to that used in the memory capacity experiment with an extra step for patterns in class 2. The input patterns belonging to the class 1 and class 2 are presented to the network. The delays are modified when a pattern of class 1 is presented and $V_{max} < V_{peak} + \Delta V$ or when a pattern of class 2 is presented and $V_{max} > V_{peak} - \Delta V$. The thresholds for training the patterns in the two classes: V_{thr} for class 1 and V_{thr-} for class 2 are shown in Figure 3.2(b). Different ΔV values were used for training. $\Delta \mathbf{d}$ is calculated as before and delays are modified as: $\mathbf{d} = \mathbf{d} + \eta \Delta \mathbf{d}$ if the pattern belongs to class 1 and $\mathbf{d} = \mathbf{d} - \eta \Delta \mathbf{d}$ if the pattern belongs to class 2. After training is completed, the classification performance of the network is tested by presenting all the patterns on which it was trained. The optimal threshold V_{opt} to test the classification performance of the network is set to V_{peak} . Therefore, a pattern in class 1 is correctly classified if it fires the output neuron by satisfying the condition $V_{max} > V_{peak}$ and a pattern in class 2 is correctly classified if it satisfies $V_{max} < V_{peak}$.

This learning process enables us to achieve increased synchrony for patterns in class 1 and decreased synchrony for patterns in class 2. An example of this is shown in Figure 3.4. The details of the simulation setup and parameters are provided in the next section; this figure gives an intuitive feeling for the effect of delay changes on $V(t)$. The delays are modified such that

Figure 3.4: Membrane potential generated by a pattern in class 1 (top row) and in class 2 (bottom row) before and after training show how learning increases V_{max} for the class 1 while reduces it for the class 2. The increased synchrony for class 1 and decreased synchrony for class 2 are shown.

a learnt pattern in class 1 (top), has more synchronized coincident spikes at the soma to allow membrane voltage, V to exceed V_{peak} while a trained pattern in class 2 (bottom) is characterized by asynchronous spikes at the soma, which result in V falling below V_{peak} .

3.3 Simulation Results

In all the experiments described here, the number of inputs (N) is 100, the temporal length of a pattern (T) is 400 ms and delays d_i were initialized from a uniform distribution between 0 and 50 ms ($d_{init} = 50$ ms). The parameters used for training are as follows: $V_0 = 2.12$, $\tau = 15$ ms, $\tau_s = \tau/4$, initial learning rate $\eta_0 = 5$ for the first 500 iterations and then reduced by 0.5 after every 500 iterations. η_0 is chosen such that the maximum change in delay in one iteration is much smaller than T . The peak location of the V_{max} distribution was estimated to be $V_{peak} = 10.2$, as shown in Figure 3.2(b). Different V_{thr} values were used for training by varying ΔV .

3.3.1 Effect of Learning: V_{max} and t_{max} distribution

The first simulation results are shown in Figure 3.5. The distribution of V_{max} before and after training was obtained to see how delay learning changes V_{max} values of spike patterns being learned. The solid blue curves denote the probability distribution of V_{max} for $P = 20, 50$ and 100 spike patterns before the network is trained while the red curves with dots show the distribution after training. There is a clear shift in the peak of the distribution towards V_{thr} indicating that the DELTRON is able to learn to respond preferentially to these spike patterns by shifting the V_{max} values closer to the V_{thr} . The black dashed curves depict the response of the trained network to a new random set of P patterns showing that its response to new, not learnt patterns is relatively unchanged. Each row corresponds to the V_{max} distribution for a different V_{thr} value. The V_{thr} values used were relative to the peak value of V_{max} distribution before training as shown in Figure 3.2(b). V_{thr} was set to $10.7, 11.2$ and 11.7 corresponding to $\Delta V = 0.5, 1.0$ and 1.5 units respectively. Figure 3.5 shows that as V_{thr} increases, the separation between the V_{max} distributions before and after training also increases indicating that the network learns the patterns by setting the V_{max} values closer to the V_{thr} . For larger values of V_{thr} , less number of “background” patterns are wrongly recognized to be in the memory and therefore, lower FP errors are incurred.

Figure 3.6 shows the distribution of t_{max} , the time at which $V = V_{max}$. As it can be seen that the t_{max} distribution before delay learning (white) is evenly spread out across time. After training, the t_{max} distribution (black) does not change considerably which implies that the delays are modified such that the EPSPs due to a spike pattern learn to arrive at times closer to the initial t_{max} value for that pattern, thereby not changing the t_{max} values significantly.

3.3.2 Pattern Memorization

In the next experiment, memory capacity of the network was calculated. It was trained on $P = 10, 20, 30, 50, 70$ and 100 random spike patterns using different values of V_{thr} as used in the first experiment. V_{opt} values corresponding to a certain value of P and V_{thr} were determined as discussed earlier (Figure 3.2(c)) and were used to test how many of the trained patterns are

Figure 3.5: Probability density distribution of V_{max} for a set of 20, 50 and 100 random spike patterns before (blue) and after (red) training showing a clear shift in the peak of the distribution for trained patterns. V_{max} distribution for new patterns presented to the trained network is shown in black.

Figure 3.6: t_{max} distribution for 100 random spike patterns. The t_{max} values before (white) and after (black) training are spread across the entire pattern duration. $V_{thr} = 10.7$.

recalled by the network. As shown in the Figure 3.5, the V_{max} distribution for learnt patterns shifts towards V_{thr} used for training. Therefore, as V_{thr} increases, V_{opt} , which lies at the point of intersection of the learnt and background V_{max} distributions, also increases. Figure 3.7 shows

Figure 3.7: Memory capacity of the model. The percentage of patterns recalled as a function of different number of patterns used to train the network. As the number of patterns used for training increases the fraction of patterns that the network learns reduces. The effect of learning with different V_{thr} values is also shown.

the percentage of patterns memorized by the network averaged over 10 repetitions. As discussed earlier, when trained on small number of patterns, $P = 20$ and $V_{thr} = 10.7$, the learning process completes by memorizing all the patterns, which is the exit condition (a) in the step (9) of learning algorithm described in Section 3.2.1. When trained on $P = 50$ patterns, the learning algorithm stops as in condition (a) 4 out of 10 times it was trained and terminates by encountering 100 local minima (exit condition (b)) 6 out of 10 times. For large number of patterns, $P = 100$, the learning algorithm exits by encountering condition (b) for all the 10 runs. The network memorizes 90-100% of the patterns when it is trained on small number ($P = 10 - 50$) of patterns with V_{thr} set to 10.7. The maximum number of patterns learned by the network depends on the number of synapses on which spike patterns arrive. In our simulations, the network having $N=100$ afferents can memorize a maximum of about 84% of the patterns for $P = 100$, i.e. about 84 random spike patterns. When higher threshold is used for training, more number of coincident EPSPs are needed for V_{max} to exceed the higher V_{thr} , thereby making the delay learning more difficult. This results in less number of learnt patterns and therefore a drop in the memory capacity for higher V_{thr} values, maximum number of patterns memorized by the

Figure 3.8: Total error given by the sum of false positive (FP) and false negative (FN) errors in the pattern memorization task as a function of P and the training threshold V_{thr} .

network reduces from a mean value of 84 to 64 for the highest $V_{thr} = 11.7$ used.

Figure 3.8 shows the total error given by the sum of FP and FN errors, incurred by the learning algorithm in the memorization task. As the number of patterns increases, the total error increases which is mainly due to the higher FN error for larger number of patterns. This has already been shown in Figure 3.7, where the memory capacity drops and therefore the FN error increases with the increase in P . For small number of patterns, $P = 10 - 20$, the total error is less for higher V_{thr} and therefore higher V_{opt} values. This is due to low FP errors incurred when higher threshold is used to recall the patterns. For moderate and large number of patterns, $P > 30$ the total errors are similar across different V_{thr} because as FP error reduces with the increase in ΔV , the FN errors start to rise due to the decreased ability of the network to learn large number of patterns at higher thresholds.

3.3.3 Pattern Classification

The network performance in the classification task was determined by training the network on random spike patterns belonging to class 1 and class 2. As mentioned earlier, a pattern belonging to class 1 is said to be correctly classified if $V_{max} > V_{peak}$ and a class 2 pattern is

Figure 3.9: Probability density distribution of V_{max} for 50 patterns learnt to be in class 1 (red) and in class 2 (blue). The effect of learning using different ΔV values on the V_{max} distributions for the two classes is shown. The V_{max} distribution for new unseen patterns presented to the trained network is shown in black. Dashed vertical lines denote the location of V_{peak} .

correctly classified if $V_{max} < V_{peak}$. $V_{peak} = 10.2$ and ΔV values used were 0, 0.2 and 0.4. The V_{max} distributions for the learnt patterns ($P = 50$) in class 1 (red) and class 2 (blue) for different ΔV are shown in Figure 3.9. It can be seen that the separation between the two learnt distributions increases with ΔV , shifting towards the training thresholds V_{thr} and V_{thr-} for the respective classes. When new patterns are presented to the network trained on classification task, the V_{max} distribution (black) obtained is same as the distribution for the learnt patterns before training. As shown in Figure 3.10, 90-100% patterns are correctly classified when the network is trained on approximately 140 patterns, 70 patterns belonging each to class 1 and class 2. As the number of patterns are increased, the network correctly classifies a maximum of about 80% of the 200 patterns it is trained on. When ΔV is increased, the training thresholds for classifying random spike patterns belonging to the two classes shift further away from the peak of the initial V_{max} distribution, increasing for class 1 and decreasing for class 2. In this case, it is expected that the criteria for classifying the patterns in the two classes would be met more easily and therefore, the classification accuracy should be same or higher than the case when $\Delta V = 0$. However, the accuracy decreases slightly because higher V_{thr} also imposes a limit on

Figure 3.10: Classification accuracy of the model. The network was trained to classify random patterns belonging to class 1 and class 2. The fraction of correctly classified patterns belonging to both classes is a function of the total number of patterns the network is trained on and the threshold V_{thr} used for each class.

the ‘trainability’ of the network.

As discussed above, the classification ability of the model depends on the number of input patterns P and ΔV . However, this capacity changes with the number of synapses N . Therefore, the capacity of DELTRON was computed as a function of the parameter $\alpha = P/N$, where α is a measure of the load on the system [2]. The performance of the network was tested on classifying spike patterns for different values of $N = 50, 100$ and 200 by varying the number of patterns for each case and $\Delta V = 0$. As shown in Figure 3.11, the classification capacity is plotted with respect to α for different N . It can be seen that the model can learn to classify patterns with 95-100% accuracy for $\alpha \leq 1$. Further, the capacity reduces to about 80% as the load on the system increases to $\alpha = 2$.

It can also be concluded that the performance of the network on the classification task is better than that on the memory task. As shown in Figures 3.7 and 3.10, the delay trained network can memorize about 50 random spike patterns with 90 – 100% accuracy while it can classify about 140 patterns belonging to two classes with similar accuracy levels. The reason for this is that the memory task can be considered as a classification task where the entire background population is the second class. Hence, it is natural that the network will mis-classify more

Figure 3.11: Capacity of the model as a function of the load on the system. The network was trained on P patterns for different number of synapses $N = 50, 100$ and 200 , $\Delta V = 0$. The classification accuracy is plotted against $\alpha = P/N$. The capacity of the model reduces as α increases.

patterns.

3.3.4 Effect of non-idealities

The noise tolerance of the delay learning scheme was also tested. These experiments tested the degree of resilience of our model to the overfitting problem of a pattern memorization task. First, noise in the t_{max} estimate was introduced by adding Gaussian noise with zero mean and standard deviation σ ms. The pattern memorization capacity of the network was computed using the delay learning scheme in which noisy estimates of t_{max} were used to modify delays. As shown in Figure 3.12, memory capacity drops in the presence of noise in t_{max} as compared to the case when ideal estimate of t_{max} is used ($\sigma = 0$). As σ is increased, the estimate of t_{max} drifts further away from its ideal value, leading to a more difficult and time consuming learning process. Hence, for small number of patterns and moderate values of σ (~ 5 ms) the capacity is almost unchanged and only when σ is close to the synaptic current fall time constant τ , the

Figure 3.12: Memory capacity of the model in the presence of noise in t_{max} estimate. A jitter of $\sigma = 2, 5$ and 10 ms was added to the t_{max} values. The pattern memorization capacity drops as higher noise level is introduced by increasing σ . $V_{thr} = 10.7$.

capacity degrades drastically.

Next, the capacity of our delay learning model to recall patterns was estimated, when presented with incomplete or noisy versions of the trained input patterns. Jittered versions of spike patterns were generated by adding Gaussian noise to each spike time. A jitter of 1.5 ms was added to all the spike patterns. The ability of the network to recognize noisy patterns was determined by calculating the memory capacity in the same way as before. The network was first trained on random spike patterns. In practice, it is useful to set the threshold during recall to be slightly less than V_{opt} so that jittered patterns can be recognized at the cost of slightly increased FP errors. So, threshold has been set to the minimum of $(V_{thr} - 0.2)$ and V_{opt} in order to calculate how many jittered versions of the trained unperturbed spike patterns are recalled by the network. For small to moderate number of patterns ($P = 10 - 50$) and $V_{thr} = 10.7$, V_{opt} estimated is usually higher than $(V_{thr} - 0.2)$ and therefore this lower threshold was used for recalling noisy patterns. This introduces small FP errors in the range of $0.5-2\%$. For larger number of patterns and higher V_{thr} , it is found that V_{opt} is much lower than V_{thr} and therefore the choice of V_{opt} to recall jittered patterns seems reasonable with no further increase in the FP errors.

Figure 3.13: Memory capacity of the model in the presence of temporal noise. Gaussian noise with standard deviation 1.5 ms was added to all spike times. The ability of the model to recognize these noisy patterns reduces by 5-15% from its memorization capacity for unperturbed spike patterns.

As shown in the Figure 3.13, the memory capacity drops by about 5-15% in the presence of temporal noise. The reduction in memorization capacity for noisy patterns is higher for lower V_{thr} ($= 10.7$) while the capacity does not deteriorate by much for higher threshold ($V_{thr} = 11.7$). For example, when trained on 100 spike patterns, the capacity reduces by about 10% for $V_{thr} = 10.7$ while it reduces by only about 4% for $V_{thr} = 11.7$. The reason for this phenomenon can be explained by the fact that the separation between the V_{max} distributions for learnt and background spike patterns is small for low V_{thr} as shown in Figure 3.5. A jittered version of the trained spike pattern has a higher chance of falling in the V_{max} distribution for the background patterns at lower V_{thr} than at higher V_{thr} . Therefore, higher number of noisy versions of learnt unperturbed patterns become unrecognizable at lower thresholds.

Finally, the performance of the model was evaluated when presented with incomplete versions of the trained input patterns. Figure 3.14 shows the percentage of patterns memorized as a function of the number of spikes missing in the spike pattern. The network was trained on 50 random spike patterns consisting of 100 spikes and $V_{thr} = 11.7$ and then 1, 2, 3, 4 and 5% of afferents were randomly selected such that no spikes arrive at these afferents. The threshold for recalling patterns is set to $\min[(V_{thr} - 0.2), V_{opt}]$ as in the last experiment. As it can be seen

Figure 3.14: Memory capacity of the model for incomplete spike patterns. The capacity of the network to recall 50 patterns consisting of spikes missing at some of the $N = 100$ afferents is shown. $V_{thr} = 11.7$.

that the memory capacity reduces from the case when complete patterns were being recalled (0% missing spikes), from a mean value of 74% of complete patterns to about 70% of incomplete patterns, when 1 spike is missing. This drop in the network capacity when recognizing incomplete patterns is related to the probability that the missing spikes arrive close to the t_{max} corresponding to each pattern and also to the difference ($V_{max} - V_{thr}$) for that pattern. Hence, the reduction in memory capacity for 1-2 missing spikes is not very significant. As more number of spikes are removed, the patterns become unrecognizable and the capacity reduces further.

3.4 Summary

This chapter presented a new learning algorithm for spiking neural networks, called the DELTRON, which involves delay modification instead of weight modification, to memorize or classify spatio-temporal spike patterns. The DELTRON can remember more patterns than other delay-based networks by modifying a few delays to remember the most ‘salient’ or synchronous part of every spike pattern. The simulations of memory and classification capacity of the DELTRON for different random spatio-temporal spike patterns were presented. The noise tolerance of the learning algorithm was also demonstrated by computing the memory capacity for noisy

spike patterns and missing spikes. The advantage of this architecture over traditional weight-based ones is simpler hardware implementation without multipliers or DACs as well as being suited to time-based computing. However, the capacity of this algorithm is lesser than weight modification ones because modifying delays can increase V_{max} by a limited amount only.

3.5 Publications

1. **Shaista Hussain**, Arindam Basu, Mark Wang, and Tara J. Hamilton, “Delay Learning Architectures for Memory and Classification,” *Neurocomputing*, vol. 138, pp. 14-26, 2014.
2. **Shaista Hussain**, Arindam Basu, Mark Wang, and Tara Julia Hamilton, “DELTRON: Neuromorphic Architectures for Delay based Learning,” *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 304-307, 2012.

Chapter 4

Spike Pattern Classification using a Model of Nonlinear Active Dendrites and Structural Learning

List of Acronyms

Acronym	Description
NLD	Nonlinear Dendrite
L/NL-neuron	Linear/Nonlinear Neuron
RM	Reduced Model
RMWM	Reduced Model with Margin
PDT	Positive Dendritic Tree
NDT	Negative Dendritic Tree
PSC	Postsynaptic Current
WTA	Winner Take All
MAE	Mean Absolute Error
RF	Receptive Field
STDP	Spike-Timing Dependent Plasticity

Acronym	Description
RSTDTP	Reverse Spike-Timing Dependent Plasticity
BSTDSP	Branch-specific Spike Time Dependent Structural Plasticity
SVM	Support Vector Machine
ELM	Extreme Learning Machine

List of Variables

Variable	Description
m	Number of dendrites in a NL-neuron
k	Number of synapses per dendrite
s	Total number of synapses
d	Number of input components
P	Number of input patterns
$b()$	Dendritic nonlinear function
$g()$	Heaviside step function
x	Binary vector input
$z_j(x)$	Total synaptic activation on j^{th} dendrite
$z_{syn,ij}$	Average synaptic activation at a synapse
$a_{L/NL}(x)$	Activation level of L/NL-neuron
x_{thr}	Dendritic threshold
b_{sat}	Saturation level of dendrite
$b_{leak}()$	Modified dendritic nonlinearity
$g_{margin}()$	Margin-based output function
δ	Margin value
y	Classifier output

Variable	Description
y^d	Desired output
T	Pattern duration
Δ	Time window
T_{syn}	Time of single spike
w_{ij}	Weight for i^{th} synapse on j^{th} dendrite
t_k	Spike times
f_{high}, f_{low}	High and low firing rates of rate encoded Poisson spike train inputs
$K(t)$	PSC kernel
I_0	Normalization constant for $K(t)$
τ_f, τ_r	Fall and rise time constants of PSC respectively
$I_{in}(t)$	Input current of a neuron
$\tilde{I}_{in}$	Average input current to a neuron
$I_{b,in}^j(t)$	Dendritic input current
$I_{b,out}^j(t)$	Dendritic output current
$V_m(t)$	Membrane voltage
V_{thr}	Threshold voltage
V_{st}	Subthreshold voltage
u	Hyperpolarization variable
τ_v, τ_u	Time constants governing $V_m(t)$ and u respectively
$s(t)$	Presynaptic spike train
$r(t)$	Postsynaptic spike train
$\bar{s}, \bar{r}$	Presynaptic and postsynaptic traces respectively
τ_{pre}, τ_{post}	Fall and rise time constants of $\bar{s}$ and $\bar{r}$ respectively
n_{spk}	Spike count of neuron
$\Delta c_{ij}, c_{ij}$	Correlation value for a pattern, correlation averaged over all patterns

Variable	Description
γ	Constant to balance potentiation and depression
C_{NL}	Capacity of NL-neuron
B	Number of distinct branch functions

After exploring the passive dendritic delay mechanisms and a learning scheme to modify these delays in the last chapter, the next step involves the use of active ionic mechanisms in dendrites. Computational modeling studies have shown that neuronal cells having active nonlinear dendrites can perform complex classification tasks much more accurately than their counterparts with linear subunits when the inputs are high-dimensional binary patterns [12]. The presence of nonlinear dendrites provides a neuron with many nonlinear functions to better approximate the decision boundary of a complex classification problem. In this chapter, a bio-plausible neuron model with nonlinear dendrites to mimic the active dendritic processing is introduced. A structural learning rule for the dendritic neurons with binary synapses is proposed and the use of this correlation-based structural rule is demonstrated for a binary classification task. The chapter includes a spike-based classification model of neurons with dendritic nonlinearity, followed by a reduced model for faster training and a margin-based reduced model for improving the generalization performance. Further, a branch-specific spike-timing-based implementation of the structural learning rule is also proposed. The simulation results comparing the performance of different learning schemes and a comparison with standard machine learning classifiers on a set of real-world binary classification datasets are presented and finally a summary of the chapter is given.

4.1 Model of Nonlinear Dendrites

In this work, the dendritic model of [12] is extended to perform supervised classification of binary patterns consisting of rate encoded spike train inputs or patterns of single spikes.

Figure 4.1: Conversion of binary patterns of spike inputs into binary vectors. An example of how rate encoded spike trains (top left) and single spikes (bottom left) are mapped to the same binary vector (right) is shown.

Two types of spike inputs - mean rate and single spike patterns are considered. These spiking inputs encode binary patterns of numbers denoted as $x \in R^d$ where each component $x_i \in \{0, 1\}$. The first type is binary patterns of mean rate inputs which are defined such that for a spike train $s(t) = \sum_k \delta(t - t_k)$ arriving at an afferent, where $\delta(t)$ is a delta function and spike times $\{t_k\} \in [0, T]$ (T is the duration of the pattern), if the mean firing rate of the spike train (averaged over T) is higher than a threshold, the corresponding x_i is set to 1; else it is set to 0 (Figure 4.1). The second type of spike inputs are binary patterns of single spikes in which $x_i = 1$ if the i^{th} afferent fires a single spike and $x_i = 0$ if it fails to fire. Furthermore, all spikes arrive within a small time window (Δ), i.e. $t_k \in [T_{syn} - \frac{\Delta}{2}, T_{syn} + \frac{\Delta}{2}]$ where T_{syn} is a fixed time between $[0, T]$. The binary inputs x are referred to as **binary vectors** in the rest of the chapter. To reduce the training time, we have adopted the strategy of training the model on binary vectors x and testing on the original spike train patterns. Next, the full spike-based model is described in Section

Figure 4.2: (a) The major difference between a traditional linearly summing neuron and our neuron with nonlinear dendrites is the lumped nonlinear function $b()$ for every dendritic branch. (b) Architecture of the 2-class pattern classifier that consists of a winner take all (WTA) to compare the outputs of two neurons trained to respond to patterns of classes (+) and (-) respectively by using the Heaviside function $g()$ to generate the output of the model.

4.1.1 followed by its reduced versions.

4.1.1 Spike-Based Model

A model of neuronal cells having lumped dendritic nonlinearity is proposed. The nonlinear neuron (NL-neuron) consists of a simplified neuron with m dendritic branches and k excitatory synapses on each branch (Figure 4.2(a)). Each synapse is driven by one of the d components of the input. The constraint of $k \ll d$ is enforced in our model for several reasons: first, choosing the best ' k ' connections out of ' d ' possibilities is the basis of our learning rule; second, sparse connections reduce synaptic resources and third, this provides a means of regularization by reducing the number of free parameters.

Let us consider the input spike train at the i^{th} synapse of the j^{th} branch given by:

$$s_{ij}(t) = \sum_{t_{ij}^s} \delta(t - t_{ij}^s) \quad (4.1)$$

where t_{ij}^s denotes the times at which spikes arrive at this synapse. The leaky integrate and fire

neuron model is used to generate the spike output using the following differential equation for the membrane voltage:

$$C \frac{dV_m(t)}{dt} + \frac{V_m(t)}{R} = I_{in}(t) \quad (4.2)$$

$$\text{If } V_m(t) \geq V_{thr}, V_m(t) \rightarrow 0;$$

$$n_{spk} \rightarrow n_{spk} + 1$$

where V_{thr} is the threshold for firing and n_{spk} is the spike count, which is initialized to 0 before every pattern presentation. The input current to the neuron, $I_{in}(t)$ is the sum of all dendritic currents, given by

$$I_{in}(t) = \sum_j I_{b,out}^j(t) \quad (4.3)$$

$$I_{b,out}^j(t) = b(I_{b,in}^j(t)) \quad (4.4)$$

$$I_{b,in}^j(t) = \sum_i w_{ij} \left(\sum_{t_{ij}^s < t} K(t - t_{ij}^s) \right) \quad (4.5)$$

Here $I_{b,out}^j(t)$ is the branch output current, $I_{b,in}^j(t)$ is the branch input current given by the sum of currents generated from all synapses on the j^{th} branch, w_{ij} is the weight of the i^{th} synapse formed on the j^{th} branch, $i \in \{1, \dots, d\}$, $j \in \{1, \dots, m\}$ such that $\sum_i w_{ij} = k$ where $w_{ij} \in Z_+$ which is the set of non-negative integers and $b()$ denotes the nonlinear activation function of the dendritic branch. The postsynaptic current (PSC) generated by the input spikes at an afferent is given by $K(t - t_{ij}^s) = K(t) \star s_{ij}(t)$ where the kernel function $K(t)$ [2] is given by:

$$K(t) = I_0 (\exp(-t/\tau_f) - \exp(-t/\tau_r)) \quad (4.6)$$

where I_0 is the normalization constant; and τ_f and τ_r are the fall and rise time constants of the PSC respectively.

The use of this model is investigated in a supervised classification of patterns belonging to two classes denoted as (+) and (-). Figure 4.2(b) shows the classifier that uses a two-input

Figure 4.3: (a) The input and output of a dendrite (shown within ellipse) when a rate encoded spike pattern is presented. Time waveform of the input current, $I_{b,in}$ and the output current, $I_{b,out}$ of the branch are shown, where branch nonlinearity $b()$ is a quadratic function. (b) For an input pattern in class (+), the (+) neuron has a higher spiking rate. These output spike trains are input to a WTA which decides the winning neuron.

WTA [84, 85] circuit to compare the outputs of a pair of one excitatory (+) and one inhibitory (-) neuron trained to preferentially respond to patterns of class (+) or (-) respectively. The WTA circuit is used to select the winning neuron by generating the binary-valued output of the classifier, y as follows:

$$\begin{aligned} \text{If } n_{spk}^+ > n_{spk}^-, y &= 1 \\ \text{else } y &= 0 \end{aligned} \quad (4.7)$$

Figure 4.3(a) shows the input and output of a dendrite (ellipse) when a rate encoded spike pattern is presented. Sample waveforms denote the time evolution of $I_{b,in}(t)$ and $I_{b,out}(t)$ for the dendrite for a case where $b()$ is a squaring function. Figure 4.3(b) shows an example case where the firing rate of the neuron trained on mean rate encoded spike patterns of class (+) is higher for an input belonging to that class, i.e. $n_{spk}^+ > n_{spk}^-$. These output spike trains of the (+) and (-) neurons are used as inputs to the WTA which gives the classification output.

4.1.2 Reduced Model (RM)

For the two specific cases of spike trains considered here (mean rate inputs and single spike patterns), the above model can be simplified to reduce the time required for the training process. The validity of reduced model is discussed in Appendix A. First, the details of the RM are presented and then the improved modified form proposed in this work is described. The average activation level ($z_{syn,ij}$) at a synapse is defined as:

$$z_{syn,ij} = \frac{1}{T} \int_0^T \sum_{t_{ij}^s} K(t - t_{ij}^s) dt \quad (4.8)$$

where T is the duration of a pattern. Therefore, the average input current to a neuron is given by:

$$\tilde{I}_{in} = \sum_j b(\sum_i w_{ij} z_{syn,ij}) \quad (4.9)$$

Here $z_{syn,ij}$ is directly proportional to x_{ij} which is the input arriving at the i^{th} synapse on the j^{th} branch, and can be written as $z_{syn,ij} = K_{syn} x_{ij}$. Therefore, the RM gives the average activation level of a NL-neuron as $a_{NL} = \hat{f}(\tilde{I}_{in})$, where $\hat{f}$ is the current-to-frequency function of the neuron (e.g. threshold linear functions can be used to approximate the expected firing rate of integrate and fire neurons [80]). For simplicity, a function with unity slope and zero threshold is used, leading to $\hat{f}(\tilde{I}_{in}) = \tilde{I}_{in}$ (choosing other values of slope will not affect the result of classification where only a comparison is made between output of the two neurons). Therefore, the activation level a_{NL} at the soma is given by:

$$a_{NL} = \sum_j b(\sum_i w_{ij} z_{syn,ij}) \quad (4.10)$$

Similarly, for a linear neuron (L-neuron), the output of the soma is defined as $a_L = \sum_j \sum_i w_{ij} z_{syn,ij}$. In Figure 4.2(a), the L-neuron is drawn in a manner similar to the NL-neuron to emphasize that the only difference between the two models is the absence or presence of the nonlinearity $b()$. Also, in all simulations both the linear and nonlinear neurons have the same total number of

synapses (or synaptic resource), $s = m \times k$ so that we can compare performance of the two neuron types. Finally, the output of the classifier can be computed by modeling the WTA as an ideal comparator to get:

$$y_{L/NL} = g(a_{L/NL}^+ - a_{L/NL}^-) \quad (4.11)$$

where $g()$ is a Heaviside step function that gives an output 1 if the argument $(a_{L/NL}^+ - a_{L/NL}^-)$ is positive and 0 otherwise. The subscript L/NL of the variables denotes the corresponding output of a linear/nonlinear neuron. For convenience, we will drop this subscript in the remainder of the chapter with the understanding that y or a refers to y_L/y_{NL} or a_L/a_{NL} in the context of linear/nonlinear neurons.

The nonlinear dendritic subunits in the RM act as higher-order units which can capture nonlinear input-output relationships as in higher-order networks [34, 35, 38]. These nonlinear mappings can be realized by polynomial functions which are useful nonlinearities used to represent correlations among input components. A squaring function is used for the dendritic nonlinearity, $b(z_j) = z_j^2/x_{thr}$, where $z_j = \sum_i w_{ij}z_{syn,ij}$ is the total synaptic activation on the j^{th} branch. With this branch function, we can write the output of a NL-neuron as $a_{NL} = \frac{K_{syn}^2}{x_{thr}} \sum_j (\sum_i w_{ij}x_{ij})^2$. This nonlinearity is chosen for its easy implementation in hardware using current-mode circuits [86, 87]. For our case of binary input vectors and binary synapses, setting $x_{thr} \geq 2$ ensures that at least two synapses on a nonlinear dendrite (NLD) need to be active ($z_j \geq 2$) to evoke a supra-linear response. The use of a quadratic nonlinearity yields first and second-order terms in the decision function of the classifier. This is illustrated in the example shown in Figure 4.4, which shows (+) and (-) neurons with one dendrite each and input connections as shown. For the binary input vector $x = [x_1 \ x_2 \ x_3]^T$ where $x_i \in \{0, 1\}$, the two dendritic functions can be written as $(2x_1 + x_2 + x_3)^2$ and $(2x_2 + 2x_3)^2$, which generate cross-correlation terms x_1x_2 , x_2x_3 and x_1x_3 . Finally, the discriminant function $h(x)$ can be written as the difference of the outputs of (+) and (-) neurons.

Figure 4.4: An example of (+) and (-) neurons with one dendrite each and a squaring nonlinearity. Connections on the branches with 4 synapses each are shown for binary input vector $x = [x_1 \ x_2 \ x_3]^T$.

$$\begin{aligned}
 h(x) &= (2x_1 + x_2 + x_3)^2 - (2x_2 + 2x_3)^2 \\
 &= 4x_1^2 - 3x_2^2 - 3x_3^2 + 4x_1x_2 + 4x_1x_3 - 6x_2x_3
 \end{aligned}$$

where $h(x) = 0$ gives the decision boundary in the 3-dimensional pattern space. If the dendritic nonlinearity is a polynomial of any order, n , it will generate terms like $x_i^p x_j^q$ in each dendritic output function, where p and q are positive integers. Since for binary inputs $x_i^p = x_i$ and $x_i^p x_j^q = x_i x_j$, therefore the connection matrix learned by the neurons (or equivalently the decision boundary) with any polynomial dendritic nonlinearity reflects the correlations present in the inputs. In other words, the (+) neuron should make connections reflecting those correlations present in patterns belonging to class (+) but not in class (-) and vice versa. This will be shown later in Section 4.4.1.1.

The use of a squaring nonlinearity for each dendritic branch will result in unrealistically large values for large inputs and large power dissipation in the corresponding analog VLSI hardware. Hence, a saturation level is introduced such that for $b(z_j) \geq b_{sat}$, $b(z_j) = b_{sat}$. Figure 4.5 shows the nonlinear function $b(z)$ for different x_{thr} values, with (dashed) and without (solid) the

Figure 4.5: Dendritic nonlinear function $b(z)$ for different values of x_{thr} and b_{sat} .

b_{sat} term. This saturating dendritic activation function also conforms well with the electrophysiological measurements of synaptic summation within dendrites of pyramidal neurons [1]. Fits using saturating polynomial functions on the experimental data measured in pyramidal neurons in the cited study were presented in Figure 2.1.

4.1.3 Reduced Model with Margin (RMWM)

It was found that when RM is tested with noisy spike inputs, the classification error is about 5 times higher than the training error. Therefore, modifications to this model are proposed with the objective of enhancing the robustness of the model to noisy inputs. This modified method, known as RMWM, uses a margin function $g_{margin}()$ (Figure 4.6(b)) during training. We can define $g_{margin}() : \Re \rightarrow \Re$, where $\Re$ is the set of real numbers, as given below:

$$\begin{aligned}
 g_{margin}(\alpha) &= 1 \text{ if } \alpha \geq \delta \\
 &= 0 \text{ if } \alpha \leq -\delta \\
 &= \frac{0.5}{\delta} \alpha + 0.5 \text{ otherwise}
 \end{aligned} \tag{4.12}$$

where α denotes $a^+ - a^-$. The Heaviside step function $g()$ (Figure 4.6(a)) is only used during the testing phase. This $g_{margin}()$ function enforces a margin δ around the classification boundary

Figure 4.6: (a) Heaviside step function $g(\cdot)$. (b) g_{margin} function to enforce a margin δ around the classification boundary.

and as shown in Section 4.4.2.2, this margin improves the generalization performance similar to margin maximization in SVMs [88].

We found that the choice of δ depends on the size of the neuron measured by the number of branches and the number of synapses on each branch. The parameter δ can also be varied to achieve a trade-off between the training and the generalization performance, as shown in [64]. A larger δ will give better generalization ability and reduce training performance while smaller values of δ will give optimal performance on the training set at the cost of reduced generalization performance. Therefore, an adaptive method was used to set the value of δ by assigning an initial high value δ_0 and then reducing it gradually as discussed in Section 4.3.2.

Next, a modification to the dendritic nonlinearity used in RM is proposed. This was motivated by the saturating nature of the $b(\cdot)$ function and the fact that branch outputs can get saturated (Figure 4.5) leading to reduced classification performance of the model. This problem becomes more severe as the number of synaptic connections on each branch increases, which increases the average input to $b(\cdot)$. Hence, $b(\cdot)$ is modified so that a constant signal proportional to the mean synaptic activation of a branch is subtracted from the total excitatory synaptic activation of each branch. The constant signal is similar to the signal from the inhibitory population

in the model proposed by [64]. The new dendritic nonlinearity $b_{leak}()$ is defined as:

$$b_{leak}(z_j) = g(z_j - z_{leak,j}) \frac{(z_j - z_{leak,j})^2}{x_{thr}} \text{ if } b_{leak}(z_j) < b_{sat}$$

$$= b_{sat} \text{ otherwise} \quad (4.13)$$

where g is the Heaviside step function and $z_{leak,j}$ is the average synaptic activation on the j^{th} branch corresponding to the initial random connections, given by $z_{leak,j} = P(x_i = 1) \times z_{syn,ij} \times k$, where $P(x_i = 1)$ is the probability that $x_i = 1$ and k is the number of synapses on every branch of the neuron. Since, we consider a neuron with all the branches having same number of synapses, therefore, average activation on a branch before learning, $z_{leak,j}$ is same for all branches. Hence, we drop the branch subscript j and denote $z_{leak,j}$ as z_{leak} in the rest of the chapter. Here, z_{leak} can be regarded as an inhibitory signal to balance the excitation on each dendritic branch.

4.2 Theoretical Capacity Predictions

For both the RM and RMWM, the theoretical capacities of the L- and NL-neurons can be predicted by estimating the number of distinct input-output functions that can be expressed by these neurons as a function of the total number of synapses, s , and the number of distinct input lines, d . We wanted to see if the added flexibility provided by the nonlinear dendrites allows us to relax the resolution of weights used. To do this, binary weights are used such that $w_{ij} = 1$ if a connection exists and 0 otherwise. Therefore, the capacity of these neurons can be given by the combinatorial expressions to count the number of input-output mappings, i.e. all possible ways in which input lines can connect to synapses on dendrites resulting in distinct memory fields. These expressions were derived in [12] and are provided here for reference. For a L-neuron, it is given by the number of ways in which s synaptic sites can be drawn from d input lines with replacement, equal to $\binom{s+d-1}{s}$. The spatial location of the s synapses is not considered. For a NL-neuron, the number of distinct branch functions available by drawing k synapses from d afferents with replacement are counted, given by $B = \binom{k+d-1}{k}$ and then the number of ways in which m branches of a NL-neuron can be drawn from B possible functions, calculated

as $\binom{B+m-1}{m}$, where the total number of synapses, $s = m \times k$ is kept the same as that for L-neuron to allow direct comparison of capacities. The theoretical capacity calculated from these expressions is compared with the classification errors measured by training linear and nonlinear neurons on binary classification tasks as described in Section 4.4.1.1.

4.3 Learning Algorithm

In this section, a learning rule is described which attempts to achieve classification performance in agreement with the theoretical capacity of a neuron, i.e. for a fixed number of synapses, minimum classification error is obtained when the predicted capacity is high. This learning rule is motivated by the feasibility of the equivalent hardware implementation. The goal of the learning process is to train the L- and NL-neurons so that the classifier output is $y = 1$ and $y = 0$ for input samples from the positive and negative classes respectively. A binary teacher signal, y^d ($= 1$ or 0) guides the learning process by indicating the desired output during the training phase. The learning process can be regarded as a combinatorial search problem of selectively forming synaptic connections on each dendritic subunit. This learning rule is a supervised or error modulated form of Hebbian learning, which is based on the change in the connection strength of a synapse as a function of the presynaptic and postsynaptic activities. However, our learning algorithm involves the formation and elimination of synapses instead of weight modification, since we use binary weights.

4.3.1 Learning Rule for RM

To guide the learning process, a fitness function c_{ij} is calculated for each synapse as the correlation between synaptic input and the output of the branch on which the synapse is located and modulated by the classification error. This correlation function is based on the fitness function used by [12], which was derived from a stochastic gradient descent learning rule. Since, (+) and (-) neurons learn to respond preferentially to patterns belonging to class (+) and (-) respectively, the learning for the two neurons is complementary to each other. The fitness

values for each pattern, Δc_{ij} are computed using the following equations:

$$\text{For (+) class neuron, for every pattern } \Delta c_{ij}^+ = x_{ij} b_j \text{sgn}(y^d - y) \quad (4.14)$$

$$\text{For (-) class neuron, for every pattern } \Delta c_{ij}^- = -x_{ij} b_j \text{sgn}(y^d - y) \quad (4.15)$$

where b_j denotes the output of the j^{th} branch given by $b_j = b(z_j)$ and $\text{sgn}()$ is the signum function with a value of 1 for $y^d > y$, -1 for $y^d < y$ and 0 for $y^d = y$. Note that the error term $\text{sgn}(o - y)$ only modulates the sign of Δc_{ij} . The c_{ij} values are then calculated over an epoch as:

$$c_{ij}^+ = \langle \Delta c_{ij}^+ \rangle \quad (4.16)$$

$$c_{ij}^- = \langle \Delta c_{ij}^- \rangle \quad (4.17)$$

where $\langle . \rangle$ indicates averaging over the entire training set.

The error term ensures that only wrongly classified patterns contribute to c_{ij} . This implies that the connection changes based on the input correlations, have minimal effect on the patterns already learned, which agrees with the basic adaptive learning principle of *minimal disturbance* [31]. The connection changes are done by using the following logic: a synapse with the smallest c_{ij} value corresponds to a poorly-performing synapse and is a candidate for replacement. Since we operate with fixed synaptic resources, this synapse needs to be replaced by another one. To do this, a set of silent synapses are first formed on the chosen branch as possible candidates for the new connection. The silent synapses are randomly selected from the set of input lines in addition to the existing synapses and do not contribute to the dendritic activity. The silent synapse with the highest c_{ij} is chosen as the replacement. Hence, grouping of connections on a dendrite is modified, guided by a Hebbian-like learning rule. This type of synaptic clustering has been observed experimentally in [52] showing activity-dependent clustering of synaptic inputs on developing hippocampal dendrites. Moreover, our calculation of c_{ij} can be implemented easily in hardware since it needs only local information while the global error signal is a binary value.

The input connections for a neuron with m branches and k synaptic contacts per branch were initialized by randomly selecting afferents from one of the d input lines with weight $w = 1$. Our model was trained on P input pattern vectors (x) belonging to (+) and (-) classes. The learning process comprises the following steps in every iteration:

- (1) The activation of each dendrite, z_j is determined and then the sum of all branch responses is calculated as in Equation (4.10) for both (+) and (-) neurons, giving $a^+(x)$ and $a^-(x)$ for all P patterns. The overall output of the classifier is computed according to Equation (4.11).
- (2) The error is calculated by averaging the absolute error $|o - y|$ over the entire batch of P patterns, which is described as mean absolute error (MAE). Since in our case, y^d and y are binary-valued, the absolute error is always 1 for every misclassified pattern and hence, the MAE is equal to the fraction of misclassified patterns.
- (3) A random set T consisting of $n_T (< s)$ synapses is selected as candidates for replacement.
- (4) The synapse with the lowest c_{ij} in T is replaced by the best-performing (maximum c_{ij}) synapse in a random replacement set R consisting of $n_R (< d)$ synapses. The set R was created by placing n_R ‘silent’ synapses from d input lines on the branch with the lowest c_{ij} synapse. They do not contribute to the calculation in step (1).
- (5) Synaptic connections are modified if the replacement led to either a reduction or no change in MAE. If the MAE increased with the replacement, a new replacement set R is created. If the MAE does not decrease after repeating this step $n_{ch} (= 100)$ times, we assume a local minimum is encountered. In this case, a replacement with the last choice of R is done, even if it increases MAE in an attempt to escape the local minimum. The connection matrix corresponding to the local minimum that gave the least error is saved.
- (6) Steps (1) to (5) are repeated until either all the patterns have been memorized or $n_{min} (= 100)$ local minima are encountered. At this point, the learning process is stopped. The

connection patterns corresponding to the best minimum become the final learned synaptic connections of the neuron.

The fraction of synapses to be replaced in each iteration is similar to a learning rate parameter in the traditional weight modification algorithm. A minimum of one out of randomly selected n_T out of all s synapses is replaced in each iteration for smooth convergence to ideal topologies. The largest possible size (n_R) of the replacement set is equal to d , the dimension of the input. Using $n_R = d$ should give better optimization results but this increases computational cost. Therefore, n_R as a fraction of d is chosen, which is a good trade-off between the computational cost and quality of the solution to the optimization problem. Finally, the RM trained on binary input vectors is tested on spiking inputs using the equivalent spike-based model and the connection matrix after training. Here, we wanted to test how well our reduced abstract learning rule can transfer the learned performance to classify the original spike patterns or the noisy spike inputs when the temporal dynamics of a spike-based model are introduced.

4.3.2 Learning Rule for RMWM

The learning algorithm for the RMWM is the same except that the fitness values are calculated as:

$$\text{For (+) class neuron, for every pattern } \Delta c_{ij}^+ = x_{ij} b_{leak,j} \text{sgn}(y^d - y) \quad (4.18)$$

$$\text{For (-) class neuron, for every pattern } \Delta c_{ij}^- = -x_{ij} b_{leak,j} \text{sgn}(y^d - y) \quad (4.19)$$

where, $y = g_{margin}(\alpha)$. The fitness values c_{ij} are calculated by taking the average of Δc_{ij} for all patterns as shown in Equations (4.16) and (4.17). The value of margin δ for the $g_{margin}()$ function as given in Equation (4.12) is determined using the following steps:

- (1) Training is first performed using the $g()$ function and then the corresponding spike inputs are presented to the model.
- (2) The values of α are recorded for all cases in which the network misclassified the spiking inputs.

- (3) A margin of δ_0 given by the highest value of α for a wrongly classified pattern is introduced around the classification boundary to reduce the misclassified cases.
- (4) δ value is reduced to 80% of δ_0 if the learning algorithm gets stuck in the same local minimum for 5 consecutive times and is reduced to 80% of its present value everytime this condition is encountered.

After the completion of training, the testing is done in the same way as in the case of RM, on spike inputs using the spike-based model.

4.3.3 Branch-specific Spike-time Dependent Structural Plasticity (BSTDSP) Rule

Our abstract models (RM and RMWM) have used the c_{ij} values to guide structural plasticity; however, no specific spike time based mechanism to achieve this computation has been shown so far. In other words, all learning is done on an abstract model while only the testing has involved spike trains. This section presents a possible spike time based rule that enables online calculation of c_{ij} using an anti-Hebbian rule for excitatory synapses. Such “reverse STDP” (RSTDP) rules have also been widely reported in biological measurements [89–91]. Our spike-based learning rule is inspired from a recent study in which RSTDP in concert with hyperpolarization of the postsynaptic neuron or spike-frequency adaptation was used to realize variants of the perceptron learning rule [92, 93] by modifying synaptic weights. The algorithm enables associative learning where a teacher signal is present during the learning process. In our case, a similar concept is used but in contrast to [92], instead of modifying weights, the change in c_{ij} is calculated. Another major difference in our work compared to [92] is that the learning rule for a synapse is modulated by its dendritic branch activation level and is hence termed branch-specific. This concept of branch specific plasticity has also been observed to a certain degree in biology [53, 94] and has been used to model feature binding in a single neuron [54]. Finally, there is a question of what physical variable in biology might represent c_{ij} ? There is evidence that calcium concentration in spines is a correlation sensitive, spike-time dependent signal and has been implicated as a guide for structural plasticity [95]. The correlations are

computed in a branch-specific manner as discussed next in our BSTDSP model.

4.3.3.1 BSTDSP: Model Details

The basic architecture is similar to the nonlinear model shown in Figure 4.2(b) consisting of (+) and (-) neurons, and the outputs of the neurons connected to a WTA circuit. The dynamics of the membrane potential $V_m(t)$ is as follows:

$$\tau_V \frac{dV_m}{dt} = (u - V_m) + I_{in}(t) \quad (4.20)$$

$$\tau_u \frac{du}{dt} = -u \quad (4.21)$$

$$\text{If } V_m \geq V_{thr}, V_m \rightarrow V_{reset};$$

$$u \rightarrow u_{reset}$$

$$u_{reset} = V_{reset} < 0$$

$$n_{spk} \rightarrow n_{spk} + 1$$

where u denotes a hyperpolarization variable which relaxes back to 0 with a time constant τ_u and is set to u_{reset} after a postsynaptic spike; $I_{in}(t)$ is the input current given by Equations (4.3)-(4.5); and τ_V and τ_u are the time constants governing the fast and slow dynamics of the membrane voltage and hyperpolarization respectively. Note that the differential equation for $V_m(t)$ in Equation 4.2 only had one state variable while Equation 5.9 has two. This allows us to separate the time scale for synaptic current based depolarization and post spike hyperpolarization i.e. recovery from hyperpolarization, which is much slower. It will be shown later in Section 4.3.3.2 that this hyperpolarization allows us to build a margin for classification. The output of the model is computed as:

$$y = g(n_{spk}^+ - n_{spk}^-) \quad (4.22)$$

The presynaptic spike train $s_{ij}(t)$ is given by Equation (4.1) and the postsynaptic spike train

is denoted by:

$$r(t) = \sum_{t_{post}} \delta(t - t_{post}) \quad (4.23)$$

where t_{post} is the postsynaptic spike time. These pre- and postsynaptic spikes also drive exponentially decaying memory traces—the presynaptic trace $\bar{s}$ and the postsynaptic trace $\bar{r}$ given by:

$$\bar{s} = \exp(-t/\tau_{pre}) \quad (4.24)$$

$$\bar{r} = \exp(-t/\tau_{post}) \quad (4.25)$$

4.3.3.2 BSTDSP: Learning Rule and Parameter Choice

To understand the learning process intuitively, let us consider the case of 2-class synchronous single spike inputs such that each afferent either fails to fire or fires a spike at a fixed time T_{syn} . During learning for $y^d = 1$, a teacher signal forcing a postsynaptic spike (at time $t = 1$ ms) is present for the (+) neuron and absent for the (-) neuron. For $y^d = 0$, vice versa conditions for the teacher signal exist.

Our structural plasticity rule used to calculate the correlation value c_{ij} , is analogous to the RSTDP rule such that if a postsynaptic spike arrives before a presynaptic spike, then the correlation value for that synapse is increased. The condition for decreasing correlations does not use postsynaptic spike—instead, the relevant event is when the membrane voltage $V_m(t)$ crosses a subthreshold voltage V_{st} from below, where $0 < V_{st} < V_{thr}$. These subthreshold crossing events occurring at times t_{st} are denoted by $r_{st}(t) = \sum_{t_{st}} \delta(t - t_{st})$. Depression happens when presynaptic spike time t_{ij}^s occurs before t_{st} . The reason for using $r_{st}(t)$ instead of $r(t)$ is to enforce a margin as will be explained later in this section. These plasticity conditions are used in a branch-specific manner to compute the $\Delta c_{ij}(t)$ values for each pattern presentation, which are

utilized to change connections on each branch. The learning rule can be written as:

$$\text{For (+) class neuron, for every pattern } \Delta c_{ij}^+(t) = I_{b,out}^{j+}(t) \bar{r}^+(t) s_{ij}^+(t) - \gamma I_{b,out}^{j+}(t) \bar{s}_{ij}^+(t) r_{st}^+(t) \quad (4.26)$$

$$\text{For (-) class neuron, for every pattern } \Delta c_{ij}^-(t) = I_{b,out}^{j-}(t) \bar{r}^-(t) s_{ij}^-(t) - \gamma I_{b,out}^{j-}(t) \bar{s}_{ij}^-(t) r_{st}^-(t) \quad (4.27)$$

where γ is a constant used to balance potentiation and depression and ensure that $c_{ij} = 0$ when a pattern is learned. The term $I_{b,out}^j(t)$ (from Equation (4.4)) indicates branch specificity and is one major difference from [92]. The correlation values averaged over all the patterns can be written as: $c_{ij}^+ = \langle \Delta c_{ij}^+(t) \rangle$ and $c_{ij}^- = \langle \Delta c_{ij}^-(t) \rangle$. The connection changes based on c_{ij} values are done as discussed in Equations (4.14) and (4.15) in Section 4.3.1.

In order to understand how the c_{ij} values computed for the BSTDSP rule approximate the c_{ij} values of the RM learning (Equations (4.14) and (4.15)), let us look at examples of all the cases – when the model predicts the correct or wrong output (y) for $y^d = 1$ or $y^d = 0$ class of input pattern. Figure 4.7 shows the membrane voltage $V_m(t)$ of the (+) neuron under these different conditions. We consider cases where $x_{ij} = 1$ since $\Delta c_{ij} = 0$ for both neurons when $x_{ij} = 0$. Further, we assume that there are n active synapses out of k synapses on the j^{th} dendrite ($z_j = n$) for each of the cases considered. Therefore, for $y^d = 1$ and $y = 1$, the correlation values can be written as:

$$\begin{aligned} \Delta c_{ij}^+ &= I_{b,out}^j(T_{syn}) \bar{r}(T_{syn}) - \gamma I_{b,out}^j(T_{rise}) \bar{s}_{ij}(T_{rise}) \\ \Delta c_{ij}^- &= 0 \end{aligned} \quad (4.28)$$

where the presynaptic spike arrives at time T_{syn} and $V_m(t)$ crosses V_{st} at time T_{rise} ; $I_{b,out}^j(T_{syn}) = b(z_j) = b(n)$. We assume that $\tau_{pre} \gg \tau_V$ which lets us write $I_{b,out}^j(T_{rise}) \approx I_{b,out}^j(T_{syn}) = b(n)$. Therefore, Δc_{ij} for this and other cases can be simplified as shown in Table 4.3.

The Δc_{ij} values of the two models can be equated for the correctly classified cases to com-

Figure 4.7: Spike-based learning for patterns belonging to the two classes: membrane voltage ($V_m(t)$) trace of the (+) neuron is shown for correctly classified patterns for $y^d = 1$ (a) and $y^d = 0$ (b) and for misclassified patterns before learning (c and e) and correct outputs after learning (d and f).

Table 4.3: Comparison of Δc_{ij} for the BSTDSP rule and RM.

Cases	BSTDSP		RM	
	Δc_{ij}^+	Δc_{ij}^-	Δc_{ij}^+	Δc_{ij}^-
$y^d = 1, y = 1$ (Figure 4.7(a))	$b(n)\exp(-T_{syn}/\tau_{post}) - \gamma b(n)\exp(-T_{rise}/\tau_{pre})$	0	0	0
$y^d = 0, y = 0$ (Figure 4.7(b))	0	$b(n)\exp(-T_{syn}/\tau_{post}) - \gamma b(n)\exp(-T_{rise}/\tau_{pre})$	0	0
$y^d = 1, y = 0$ (Figure 4.7(c))	$b(n)\exp(-T_{syn}/\tau_{post})$	$-\gamma b(n)\exp(-T_{rise}/\tau_{pre})$	$b(n)$	$-b(n)$
$y^d = 0, y = 1$ (Figure 4.7(e))	$-\gamma b(n)\exp(-T_{rise}/\tau_{pre})$	$b(n)\exp(-T_{syn}/\tau_{post})$	$-b(n)$	$b(n)$

pute the value of γ as given by:

$$\gamma = \frac{\exp(-T_{syn}/\tau_{post})}{\exp(-\bar{T}_{rise}/\tau_{pre})} \quad (4.29)$$

Here, $\bar{T}_{rise}$ can be estimated by taking the average of all the times at which $V_m(t)$ crosses V_{st} for different patterns. From the misclassified cases, we can write that:

$$[\Delta c_{ij}]_{BSTDSP} = \exp(-T_{syn}/\tau_{post})[\Delta c_{ij}]_{RM} \quad (4.30)$$

Therefore, c_{ij} values for spike-based learning are smaller than the c_{ij} values calculated for the RM. Since connection changes are done on the basis of the ranking of c_{ij} , therefore Equation (5.25) suggests that for the two models, the learning process comprising modifying connections should be the same for a given set of input patterns and for the same initial connections. This will remain true if the assumptions used to derive the above relationship are satisfied, i.e. $\tau_{pre} \gg \tau_V$ and that the estimate of T_{rise} is close to the actual value of T_{rise} for each pattern.

Figures 4.7(d) and 4.7(f) show the correct classification outputs after learning where teacher signals are not available. Hence for $y^d = 1$, in the absence of initial hyperpolarization, $V_m(t)$ starts from 0 and can easily cross V_{thr} thereby providing robustness against noise. During learning, $V_m(t)$ is encouraged to cross a lower threshold V_{st} , which ensures a margin for learning

class (+) patterns, given by:

$$\delta_{spike}^+ = (V_{st} - V'_{reset}) - V_{thr} \quad (4.31)$$

where V'_{reset} is the voltage at the time T_{syn} when an initial postsynaptic spike is present. For class (-) patterns, $V_m(t)$ does not cross V_{thr} after learning while during learning $V_m(t)$ is reduced below V_{st} providing a desired margin of:

$$\delta_{spike}^- = (V_{thr} - V_{st}) \quad (4.32)$$

For an unbiased classifier, the margins for the two classes are equal. Using this condition, the values of V_{st} and V_{reset} can be found. The value of the desired margin $\delta_{spike} = \delta_{spike}^{+/-}$ is determined by using the spike equivalent value of margin, δ (Section 4.1.3) after learning. This is done by computing ΔV_m , which is the difference in the membrane voltage of the (+) and (-) neurons (analogous to $a^+ - a^-$) when a single synapse is activated. This value multiplied by δ gives δ_{spike} since in the binary input case the synaptic strength is normalized to 1. Since δ is a function of m and k , different values of δ_{spike} are used, which entails changing the values of parameters V_{st} and V_{reset} for different neuron configurations.

4.4 Results

Firstly, the procedure for training the RM and RMWM is discussed, followed by the method used to test the performance of these models.

- (1) Training on binary vectors: To demonstrate the learning process of the model, random patterns were generated in a manner similar to [12]. The input patterns were chosen from a d_o ($= 40$) dimensional spherical Gaussian distribution. These patterns were randomly assigned to the positive ($y^d = 1$) and negative ($y^d = 0$) classes. The d_o -dimensional input patterns were mapped into a high d ($= 400$) dimensional space to incorporate sparse neural coding by using $n_{rf} = 10$ non-overlapping binary-valued receptive fields (RFs) along each dimension. Hence, binary vectors $x \in \{0, 1\}^d$ were generated such that the

probability of any one RF neuron being active is 0.1, i.e. $P(x_i = 1) = p = 0.1 \forall i$. The binary patterns thus generated were used to train either the RM (Section 4.1.2) or the RMWM (Section 4.1.3). The connection matrix learned after training process is used for testing the model. This method has the advantage of shorter simulation times as compared to the learning on spike inputs. It can be used in practice as well (if the desired input patterns are known) where the learned connection matrix can be downloaded to a neuromorphic chip.

- (2) Testing on noisy spike inputs: The noise sensitivity of the RM and RMWM was tested on noisy spiking versions of the binary input vectors used to train the models. Henceforth testing the model will refer to testing the noise tolerance of the model except for the classification of benchmark datasets, where the generalization performance is evaluated. We introduce noise in the inputs by using Poisson spike trains or jittered single spikes. We used Poisson spike trains because we wanted to model a realistic case where our classifier receives inputs from a noisy spiking sensor or from biological neurons in the case of brain-machine interfaces. The two types of spiking inputs used for testing are generated as: 1) for rate encoded spike patterns, binary values of 1 and 0 are mapped to Poisson spike trains with mean firing rates of f_{high} and f_{low} respectively, similar to [96] and 2) single spike patterns are generated by mapping binary input ' $x_i = 1$ ' to a single spike arriving randomly within $[T_{syn} - \Delta/2, T_{syn} + \Delta/2]$ and ' $x_i = 0$ ' to no spike, where T_{syn} is a fixed time within the stimulus duration T . The testing on spike patterns utilizes the spike-based model (Section 4.1.1) with the learned connection matrix.

The results obtained are organized as follows:

- (1) Firstly, the classification performance of RM is discussed, showing that the performance of RM on noisy spike trains is much worse than the training performance on binary vectors. Hence, RM suffers from poor noise tolerance.
- (2) Secondly, the margin-based RMWM is shown to lead to improvements in both the training performance and the noise sensitivity.

- (3) Next, the classification results from the BSTDSP rule are presented, where the performance of the abstract RMWM is compared with that of our bio-realistic BSTDSP model.
- (4) Finally, the performance of RMWM is compared against support vector machine (SVM) and extreme learning machine (ELM) methods on 3 different databases.

Simulation results are presented for $P = 500, 700$ and 1000 input patterns using $m = 10, 20$ and 50 dendrites and $k = 5, 10, 15, 25$ and 50 synapses on each dendrite, unless stated otherwise. The learning parameters are $n_R = n_T = 25$. The parameters for testing on spike patterns using the spike-based model and the parameters for the BSTDSP model are summarized in Tables 4.4 and 4.5 respectively. These parameter values were chosen in accordance with the validity of the RM as discussed in Section A. All the results are obtained by averaging across 5 simulation trials.

Table 4.4: Parameters for Spike-Based Model

f_{high}	f_{low}	τ_r	τ_f	I_0	V_{thr}	R	C	T
250 Hz	1 Hz	2 ms	8 ms	2.12	10 mV	10 M Ω	5 nF	200 ms

Table 4.5: Parameters for BSTDSP Model

τ_V	τ_u	τ_{pre}	τ_{post}	T_{syn}
4 ms	80 ms	10 ms	50 ms	100 ms

4.4.1 Performance of RM

First, the results of training the RM with binary vectors are presented and then testing its robustness to noisy spike inputs.

4.4.1.1 Training Performance

(I) Comparison of L- and NL-neurons

In the first experiment, the performance of L- and NL-neurons is compared in a binary classification task. Both the classifiers were trained on $P = 500$ and 1000 input patterns

Figure 4.8: Classification error as measured by the MAE of the L-neuron and NL-neuron. The plot shows the MAE as function of learning iteration for 500 and 1000 training patterns, number of dendrites, $m = 20$.

and their performances were compared. As shown in Figure 4.8, the MAEs decrease for both neurons as learning progresses. However, the NL-neuron achieves a lower error than its linear counterpart and this difference in error grows as the classifier is trained on larger number of patterns as seen from the performance on 1000 versus 500 training patterns.

In order to understand the effect of learning on the connection matrix, it was investigated whether the input connections learned by a neuron are able to capture the correlations present in the input patterns. If x_m^+ and x_m^- denote the m^{th} binary vector belonging to class (+) and (-) respectively, then $[(x_m^+).(x_m^+)^T]$ computes the second-order correlations present in the (+) pattern. $R = [\sum_{m=1}^{P^+} (x_m^+).(x_m^+)^T]/P^+ - [\sum_{m=1}^{P^-} (x_m^-).(x_m^-)^T]/P^-$ is a $d \times d$ matrix for d -dimensional patterns with d^2 entries of which the $d(d+1)/2$ values in the upper triangle and the diagonal are unique. P^+ and P^- denote the total number of patterns in class (+) and (-) respectively. The entries in R can be ordered according to decreasing magnitude which provides a unique ranking $O = \{(r_1, c_1), (r_2, c_2), \dots\}$ of the $d(d+1)/2$ entries in the upper triangle and the diagonal, where r_i and c_i denote row and column index respectively. Lower ranks correspond to high positive value entries in R , which

Figure 4.9: R_W values of the weight correlation matrix (x-axis) for (+) and (-) neurons with $m = 10$ dendrites (y-axis) before and after learning. The histograms plotted by summing the correlation values for each input pair for all the dendrites are shown at the bottom of each R_W matrix.

denote $x_p x_q$ combinations that are specific to pattern class (+) and not in pattern class (-). Similarly, high ranks correspond to high negative value entries in R for those $x_p x_q$ combinations that uniquely define pattern class (-). Similarly, $[(W_j^+).(W_j^+)^T]$ gives the correlation matrix for the weights on a dendrite of the (+) neuron, where W_j^+ is the d -dimensional weight vector of the j^{th} dendrite, $W_j^+ \in \mathbb{Z}_+^d$, where $\mathbb{Z}_+^d$ is a d -dimensional set of non-negative integers.

Figure 4.9 shows how the input correlations learned by the (+) and (-) neurons match with those present in the input patterns. In this example, neurons with $m = 10$ dendrites and $k = 5$ synapses per dendrite were trained to classify $P = 100$ binary patterns where $d = 20$. R_{W_j} is obtained by ordering the entries in the correlation matrix $[(W_j).(W_j)^T]$ according to O and is plotted for each dendrite of the (+) and (-) neurons before and after training. Each pixel shows the correlation value between a particular pair of input connections formed on a dendrite. For $d = 20$, the total number of unique $x_p x_q$ combinations is equal to 210, and hence x-axis ranges from 1 – 210 and y-axis ranges from 1 – 10 for 10 dendrites in each neuron. The histograms obtained by plotting the sum of correlations

Figure 4.10: MAE as a function of the number of dendrites, m . The L-neuron (circles) has higher classification errors than its nonlinear counterpart (squares). The optimal choice of m for total number of synapses, $s = 200$ is $m = 50$. The dashed curve shows the theoretical capacity of the neuron for the same values of m and s .

for each connection pair for all the dendrites are shown below each correlation matrix. Figures 4.9(a) and 4.9(c) show that the initial connections formed on the (+) and (-) neurons pick up the input correlations randomly, as evident from the correlation values evenly distributed along the x-axis. As seen in Figure 4.9(b), the (+) neuron has a much higher clustering of connections corresponding to the lower ranked $x_p x_q$ combinations unique to (+) patterns while it learns very few of the correlations $x_p x_q$ with high negative values (higher ranked) unique to the (-) patterns. Similarly, the clustered appearance toward the right end of x-axis in Figure 4.9(d) indicates that the (-) neuron learns to capture more of the correlations present in the (-) patterns and fewer of those in (+) patterns. Hence, the proposed learning algorithm enables the connection matrix to pick out unique correlations in the data. For the binary classifier consisting of a pair of neurons, it has been shown that each neuron prefers patterns belonging to a particular class by mimicking the input correlations present in that class.

(II) Effect of number of dendrites

Next, the performance of the model is tested as the number of dendrites, m is varied. The

total number of synaptic resources was kept constant at $s = 200$. Figure 4.10 shows the MAE versus m . The circles represent the errors for the L-neuron with $m = 1$ and the NL-neuron errors are denoted by squares ($m > 1$). The performance of both types of neurons decreases with larger number of input patterns. However, the L-neuron performs much worse than the NL-neuron with classification errors of 19% against 9% for 1000 patterns. Further the NL-neuron errors decrease as the number of branches increases, which can be explained by the fact that as m increases, a larger number of dendritic functions can be selected from the B possible functions obtained by drawing k synapses from d input lines (Section 4.2). These nonlinear branch functions can well approximate the nonlinear classification boundary of the problem as explained in the example in Section 4.1.2. As the number of dendrites continues to increase, the number of branch functions that can be drawn from B functions reduces if m increases beyond a certain value. As seen in Figure 4.10, the optimal value of m is 50 for $s = 200$. The theoretical capacity of the neuron is also plotted on the right-hand y-axis (dashed), calculated by taking the logarithm of the combinatorial expressions given in Section 4.2. As the capacity increases, more patterns can be learned and can be correctly classified. Hence, classification errors should decrease, which is in accordance with the empirical results. Similar to the measured classification errors, the theoretical capacity also peaks around $m = 50$ and further reduces as m increases. Therefore, as the number of dendrites grow, the classification performance of a NL-neuron with fixed synaptic resources (s) first increases to a peak value and then falls with further addition of dendrites.

(III) Effect of number of synapses per dendrite

The performance of NL-neuron was also determined as a function of number of synapses per branch (k) since it is a free parameter for an AER-based hardware system that can be easily adjusted. The reciprocal of theoretical capacity of a neuron as a function of k is plotted in Figure 4.11. It shows that for a neuron with a fixed number of dendrites, the inverse capacity decreases and therefore capacity increases as the number of synapses per dendrite is increased and as more dendrites are added, the neuron can learn more patterns.

Figure 4.11: Reciprocal of capacity of a neuron as a function of number of synapses per branch (k) for varying number of branches (m). Capacity increases as the total number of synapses $m \times k$ increases.

Figure 4.12 shows classification errors measured empirically. The errors decrease as k increases for a fixed value of m , which is due to the presence of higher number of total synapses $s = m \times k$. The larger number of patterns are more difficult to train and hence lead to higher errors. As larger number of dendrites are used, the errors decrease further and for a fixed number of synapses per branch, for example $k = 25$, the classification error for 1000 patterns reduces from 11.2% for $m = 10$ by a factor of 2 for $m = 20$ and of 6.5 for $m = 50$. Therefore, the empirical results agree with the theoretical capacity predictions that is, capacity increases with higher number of synapses, more patterns can be correctly classified.

4.4.1.2 Noise Sensitivity Performance

The spike-based model with the same connection matrix was used to classify spike patterns of mean firing rates. The classification performance of RM on spike patterns was measured as a function of k . Figure 4.13 shows that the classification errors for spiking inputs are much higher than that for the mean rate inputs on which the model was trained. It can be seen that the testing errors on noisy spike train inputs are 2 – 5 times larger than that of the training errors for binary

Figure 4.12: MAE as a function of k , the number of synapses per branch in a NL-neuron for binary inputs. 500, 700 and 1000 patterns were trained on neurons with $m = 10, 20$ and 50 branches. The errors decrease as the number of synapses increases.

Figure 4.13: Testing performance for spike patterns as a function of k . The errors are much higher than that for the corresponding binary input vectors (Figure 4.12(c)).

input vectors (Figure 4.12(c)). In contrast to the training errors decreasing with k , it is observed that the testing errors increase as k is increased.

The increased errors can be attributed to the fact that as number of branches (m) increases, the linear summation of dendritic branch outputs leads to a very high average value of the input

currents I_{in}^+ and I_{in}^- ($I_{in} = \sum_{j=1}^m I_{b,out}^j = \sum_{j=1}^m I_{b,in}^2 / I_{thr}$) to the (+) and (-) neurons respectively. This causes the small differences between I_{in}^+ and I_{in}^- to become indistinguishable for the (+) and (-) neurons due to the finite precision of the spike-based model used for testing as compared with the infinite precision of the abstract RM used for training. It can also be seen from Figure 4.13 that the errors increase as the number of synapses per branch increases. This behaviour of the model can be explained by the fact that the use of higher dendritic threshold I_{thr} to prevent branch saturation at larger k reduces the I_{in}^+ and I_{in}^- currents due to a spike pattern and therefore leads to decreased discriminability of the input currents to the spike-based (+) and (-) neurons. Next, the use of RMWM in rectifying these issues is investigated.

4.4.2 Performance of RMWM

Here, the results of training the RMWM with binary vectors are presented and then testing its noise sensitivity.

4.4.2.1 Training Performance with $g_{margin}()$ function

We present the results of applying the method discussed in Section 4.3.2 to determine the value of δ for the $g_{margin}()$ function. The model was trained on $P = 500$ binary input patterns and then tested on the spike versions of these patterns. The α values for all the misclassified inputs were recorded. This experiment was repeated for NL-neurons with different number of branches and number of synapses per branch ranging from $k = 5$ to 50. The probability density distribution (pdf) of α as a function of number of synapses on each dendrite is shown in Figure 4.14 for $m = 20$ and three values of k . We can see that the peak of the pdf of α varies with both k and m (Effect of ‘m’ not shown in the Figure). Therefore, for a neuron with 20 branches and around 5 – 50 synapses per branch, δ_0 can be set to around 20 and then gradually reduced during training. Similar δ values can be used for training the NL-neurons with the same values of m and k on a classification dataset consisting of binary input patterns.

Next the $g_{margin}()$ function (Figure 4.6(b)) was utilized to test if the performance of the classifier improves. The model was trained on 1000 patterns for different values of k . The

Figure 4.14: Determination of δ for $g_{margin}()$ function. The distribution of α for misclassified spiking inputs. $P = 500$ and $m = 20$.

$g_{margin}()$ function was used during the training process with $\delta_0 = 25$. After the training was completed, the memorized patterns were recalled using $g()$ function. As shown in the Figure 4.15, the classification errors reduce for all neuron configurations. As compared to training errors of RM (Figure 4.12(c)), the errors corresponding to $g_{margin}()$ are smaller by factors of 2 – 5 and for large k values (> 25), the errors reduce drastically by factors of more than 10 – 20. Hence, the improved classification performance is observed because the model is trained to generate a decision boundary with a margin δ . We will also present the results of improved noise tolerance achieved using the $g_{margin}()$ function.

4.4.2.2 Improving Noise Sensitivity

Here, the RMWM is utilized to improve the noise sensitivity by rectifying the problems discussed in Section 4.4.1.2 in a step-wise manner.

- (a) In order to balance the effect of high average values of the input currents for large values of m , the following strategy was adopted. The model architecture was modified such that both (+) and (-) neurons receive inputs from all the dendrites, which are split into excitatory and inhibitory dendritic trees, also referred to as positive dendritic tree (PDT)

Figure 4.15: MAE as a function of k for the $g_{margin}()$ function. The training was done with δ_0 set to 25 and the input patterns were tested using $g()$ function. The performance improves for increasing k and m .

and negative dendritic tree (NDT) respectively (Figure 4.16). In biological neural networks, the dendritic trees can have both excitatory and inhibitory synapses. However, a simplified scheme is used in which input currents for the two neurons are given by:

$$I_{in}^+(t) = I_{PDT}(t) - I_{NDT}(t) \quad (4.33)$$

$$I_{in}^-(t) = I_{NDT}(t) - I_{PDT}(t) \quad (4.34)$$

where $I_{PDT}(t)$, $I_{NDT}(t)$ are the currents generated by the PDT and NDT, calculated using Equation 4.3. As shown in Figure 4.17(a), the reduction in classification error is most significant, by a factor of 1.2 for $m = 10$ as compared to when testing results were obtained without the differential input (Figure 4.13). The errors reduce for smaller k while for larger k , the performance still drops with increasing k . The dashed plots correspond to the training performance of the RMWM as shown in Figure 4.15. As described next, the RMWM proposes to achieve noise tolerance to spiking inputs which is close to this optimal training performance obtained by the model.

Figure 4.16: Architecture showing two neurons generating the output of the model in response to current differences ($I_{PDT}(t) - I_{NDT}(t)$) and ($I_{NDT}(t) - I_{PDT}(t)$).

- (b) Next, the $g_{margin}()$ function is used to train the model on binary input vectors and then tested the performance on spiking inputs. This approach was used to improve the noise tolerance to the spike train versions. The errors computed using a NL-neuron are shown in Figure 4.17(b). As seen, the classification errors reduce for all combinations of m and k by factors of 1.5 – 6.5 of that of errors obtained using RM and the performance levels match with the training performance (dashed plots) more closely than in Figure 4.17(a). These results suggest that training the model with $g_{margin}()$ function improves the robustness to noise by introducing a margin around the classification boundary. As seen in Figure 4.17(b), for $m = 50$, the errors tend to rise with increasing k as observed in the earlier results. Next, the approach used to solve this problem is discussed.
- (c) Here, the use of dendritic nonlinearity, $b_{leak}()$ defined in Equation (4.13) in improving the performance for spike patterns at larger k values is tested. As shown in Figure 4.17(c), the errors at larger k (15 – 50) are smaller by factors of 1.3 – 2.5 as compared to errors obtained with the nonlinearity $b()$ (Figure 4.17(b)). However, there is no significant change in the performance for smaller k values. This method further improves the test performance on noisy spiking inputs and the overall reduction in the classification errors for RMWM is by a factor of about 1.5 – 10 of that of test errors for RM.

Figure 4.17: Performance on spike patterns improves as (a) differences in $I_{PDT}(t)$ and $I_{NDT}(t)$ used as input currents for the integrate and fire neurons to counter the high ‘m’ effect; (b) $g_{margin}()$ used to improve the noise tolerance of the learning algorithm; and (c) inclusion of a dendritic nonlinear function $b_{leak}()$ improves the performance at larger k values. $P = 1000$, dashed plots correspond to the errors for binary vectors trained using the $g_{margin}()$ function.

Figure 4.18: Testing performance for single spike patterns ($P = 1000$) as a function of k for $m = 20$. The training (dashed) errors are about 2% less than the testing errors for $\Delta \leq \tau_f$.

4.4.2.3 Performance on Single Spike Patterns

The performance of the model was also tested on patterns of single spikes. The input connections were learned using the RMWM on binary input vectors. The testing was done on 1000 single spike patterns arriving within a time window of $\Delta = 0.5\tau_f$, τ_f , $2\tau_f$ and $3\tau_f$, where τ_f is

Figure 4.19: Membrane voltage $V_m(t)$ of (+) and (-) neurons when a class (+) pattern is presented. The pattern is not classified correctly since the (+) neuron fails to fire.

the fall time constant of the postsynaptic waveform (Equation (5.7)) and $T_{syn} = 100$ ms. The results shown in Figure 4.18 suggest that the model generalizes well on single spike patterns with testing errors about 1.2 – 4 times that of the training errors (dashed) when $\Delta \leq \tau_f$, where $\tau_f = 8$ ms. This condition was found to be satisfied for different values of $\tau_f = 8, 16$ and 24 ms, the results for $\tau_f = 16$ and 24 ms are not shown in Figure 4.18. As Δ increases, the testing errors increase because the spikes arriving at the synapses on the same dendrite are now more spread out in time leading to a reduced dendritic response as compared to the response learned with binary vectors.

4.4.3 Performance of BSTDSP Model

The performance of our BSTDSP rule discussed in Section 4.3.3.1 was evaluated on a two-class classification of spike patterns. The architecture used here is similar to the two neurons model shown in Figure 4.16 where the input current into each neuron is the difference of currents with opposite signs. For training, random binary patterns of single spikes were used with $\Delta = 0$. In this case, a low value of V_{thr} ($= 0.1$ mV) was chosen such that for the initial random connections, a neuron evokes a spike for about 65% of the input patterns. The reason for choosing a low V_{thr} value can be understood from Table 4.3 where for the RM, any misclassi-

Figure 4.20: Classification performance of the BSTDSP algorithm. (a) Spike learning results when trained with (squares) and without (circles) margin. $P = 500$ patterns of single spikes, $m = 20$ (b) Comparison with RMWM learning (circles). The training errors are shown in solid and testing errors with dashed lines.

fication leads to Δc_{ij} values (last two rows of Table 4.3) of opposite signs which contribute to the connection change in both (+) and (-) neurons after an epoch. This strategy of modifying both neurons limits the changes required in each neuron as compared to the case when only one neuron contributes at a time. Figure 4.19 shows a case where a (+) pattern is not classified correctly. However, only (+) neuron has non-zero Δc_{ij} (Δc_{ij}^+ value for $y^d = 1$ and $y = 0$ in Table 4.3) while $\Delta c_{ij}^- = 0$ because the (-) neuron does not fire a spike due to high V_{thr} . The use of a lower V_{thr} will result in both Δc_{ij}^+ and Δc_{ij}^- to be non-zero and of opposite signs.

The classification performance of the spike-based algorithm is shown in Figure 4.20. The training errors are plotted as a function of k for $P = 500$ patterns. The model was also tested on noisy versions of the patterns of single spikes by adding a uniform random noise within the time window $\Delta = \tau_f$ around the spike time T_{syn} . Figure 4.20(a) shows the performance when margin was not introduced, i.e. $\delta_{spike} = 0$ (circles) and when margin was introduced by using $\delta_{spike} > 0$ (squares), corresponding to different k values. It can be seen that both training and noise testing performances improve significantly as a result of margin-based learning, similar to the results obtained for RMWM learning (Section 4.4.2). The errors for noisy spike inputs with margin classifier reduce by up to a factor of 5 as compared to learning without margin.

Margin-based learning using spike inputs was compared with the RMWM learning on mean rate inputs as shown in Figure 4.20(b). The results show that the spike-timing based learning yields training errors (solid, squares) which are similar to that of training errors for RMWM (solid, circles). The test performance on noisy patterns of single spikes for the two instances of learning is shown with dashed plots. The test errors for spike learning are higher than the errors for RMWM learning by about 2.5 times and this factor increases with k . These differences in the performance of BSTDSP from RMWM can be attributed to the differences in Δc_{ij} calculations for the two schemes, which are not exactly same. As discussed in Section 4.3.3.2, the value of γ as calculated in Equation (5.17) is based on the assumption that $\tau_{pre} \gg \tau_V$ which is not perfectly true in our case. This also results in the estimate of T_{rise} used for calculating γ to be different from the actual T_{rise} for each pattern. Hence, the values of Δc_{ij} are slightly different from the ideal values as shown in Table 4.3. Another discrepancy between the learning for BSTDSP and RMWM can arise from the fact that the margin setting is not exact. As discussed in Section 4.3.3.2, the margin for spike-based learning was set using $\delta_{spike} = \Delta V_m \times \delta$ where a wrong estimate of the constant ΔV_m will mean that δ_{spike} and δ are not truly analogous to each other, leading to different results. However, it can be seen that even with these differences in the learning for the two models, our proposed bio-realistic spike-based learning can achieve comparable errors with the more precise learning in the abstract model presented earlier.

4.4.4 Performance on Benchmark Datasets

Finally, the performance of the proposed algorithm was tested on real-world benchmark classification tasks and compared the results with the performance of the SVM and ELM methods. The binary datasets include datasets on breast cancer (BC), heart disease (HEART) and ionosphere (ION). These datasets (Table 4.6) are taken from the UCI Machine Learning Repository [97]. The performance of our improved RMWM with structural learning rule was evaluated for both binary vector patterns as well as spiking inputs, where each sample corresponding to a feature was mapped into ten non-overlapping density-matched RFs, which were generated from the probability distribution of all samples of that particular feature. The resulting binary

samples were then converted into spike patterns by using the method discussed in Section 4.4.

Table 4.6: Specifications of Benchmark Binary Datasets.

Datasets	# features	# training samples	# testing samples
BC	9	222	383
HEART	13	70	200
ION	34	100	251

The performance measures like number of neurons, number of weights and classification accuracy for different algorithms are given in Table 4.7 to compare the classification performance as a function of the computational complexity of the network used. The results for SVM and ELM are taken from [98]. The number of nonlinear dendrites of a neuron is analogous to the number of support vectors of the SVM and the number of hidden neurons for the ELM. The performance of different classifiers is also compared on the basis of synaptic resources used. The total number of weights in an N -input network (SVM and ELM) using M neurons is given by $N \times M + M$. The total number of weights in the dendritic model with m dendrites and k synapses per dendrite for each neuron in the classifier is given by $s = m \times k$. The classification accuracy obtained in this work for spike inputs is slightly less (0.1 – 0.8%) than that for binary input vectors. The reduced accuracy is probably due to the noise in the Poisson spike trains and the effects of the integration and fire process of the neuron. As the results suggest the proposed model uses a similar number of dendritic processing units as the number of neurons used by SVM and ELM and achieves generalization performance within 1 – 2% of that of these algorithms. Moreover, our method achieves this performance by utilizing 10 – 50% fewer binary synapses, particularly for HEART and ION datasets, therefore rendering the model feasible for hardware implementation.

4.5 Summary

In this chapter, a structural learning rule for neurons with nonlinear dendrites was presented and used for binary classification of spike patterns, with the aim of implementation in neuro-

Table 4.7: Performance Comparison of SVM, ELM and Dendrite Algorithms on Benchmark Datasets.

Datasets	SVM			ELM			Dendrite Model			
	#neurons	#wts	Acc.	#neurons	#wts	Acc.	#dendrites	#wts	Acc. (Binary)	Acc. (Spike)
BC	24	240	96.61	66	660	96.35	20	204	96.01	95.93
HEART	42	588	75.5	36	504	76.5	10	104	75.3	74.53
ION	43	1505	91.24	32	1120	89.64	50	404	89.22	88.96

morphic hardware. The main contributions in this chapter include (1) using a correlation-based structural learning rule for binary synapses which is more amenable to hardware implementation; (2) developing and testing a reduced model (RM) to minimize the training time and a reduced model with margin (RMWM) which led to a significant improvement in the noise sensitivity performance of our model; (3) proposing a branch-specific spike time dependent structural plasticity (BSTDSP) rule tying it to the well-known spike-based learning protocols; and (4) finally, comparing the generalization performance of our model with state-of-the-art nonlinear classifiers on real-world classification tasks which showed that our neurally-inspired model gives similar performance to these other methods by utilizing much less synaptic resources.

4.6 Publications

1. **Shaista Hussain**, Shih-Chii Liu, and Arindam Basu, “Hardware-Amenable Structural Learning for Spike-based Pattern Classification using a Simple Model of Active Dendrites,” *Neural Computation*, vol. 27, no. 4, pp. 845-897, 2015.
2. Subhrajit Roy, Arindam Basu, and **Shaista Hussain**, “Hardware efficient, Neuromorphic Dendritically Enhanced Readout for Liquid State Machines,” *IEEE Biomedical Circuits and Systems (BioCAS)*, pp. 302-305, 2013.
3. Enyi Yao, **Shaista Hussain**, Arindam Basu, and Guang-Bin Huang, “Computation using Mismatch: Neuromorphic Extreme Learning Machines,” *IEEE Biomedical Circuits and*

Systems Conference (BioCAS), pp. 294-297, 2013.

4. **Shaista Hussain**, Roshan Gopalakrishnan, Arindam Basu, and Shih-Chii Liu, “Morphological Learning: Increased Memory Capacity of Neuromorphic Systems with Binary Synapses Exploiting AER Based Reconfiguration,” *International Joint Conference on Neural Networks (IJCNN)*, pp. 1-7, 2013.

Chapter 5

Optimization and Extensions of the Structural Learning Rule for Dendritic Neurons

List of Acronyms

Acronym	Description
NLD	Nonlinear Dendrite
NL-neuron	Nonlinear Neuron
PDT	Positive Dendritic Tree
NDT	Negative Dendritic Tree
PSC	Postsynaptic Current
WTA	Winner Take All
TTFS	Time-to-First-Spike
RBM	Restricted Boltzmann Machine
STDP	Spike-Timing Dependent Plasticity
RSTDP	Reverse Spike-Timing Dependent Plasticity
BSTDSP	Branch-specific Spike Time Dependent Structural Plasticity

List of Variables

Variable	Description
m	Number of dendrites in a NL-neuron
k	Number of synapses per dendrite
s	Total number of synapses
d	Number of input components
M	Total number of dendrites
N_C	Number of classes
P	Number of input patterns
m_{opt}, k_{opt}	Theoretically optimal values of m and k respectively
m_{adapt}	Number of dendrites learned using adaptive learning scheme
$b()$	Dendritic nonlinear function
$g()$	Heaviside step function
x	Binary vector input
$z_j(x)$	Total synaptic activation on j^{th} dendrite
$z_{syn,ij}$	Average synaptic activation at a synapse
$a(x)$	Output of PDT/NDT
$o(x)$	Difference of outputs of PDT and NDT
$g_{margin}()$	Margin-based output function
δ	Margin value
y	Classifier output
y^d	Desired output
T	Pattern duration
T_{syn}	Time of single spike
Δ	Jitter in single spike

Variable	Description
w_{ij}	Weight for i^{th} synapse on j^{th} dendrite
t_{ij}	Spike times at i^{th} synapse on j^{th} dendrite
f_{high}, f_{low}	High and low firing rates of rate encoded Poisson spike train inputs
$I_{DT}(t)$	Currents generated by PDT/NDT
$I_{D,j}(t)$	Output current of j^{th} dendrite of PDT/NDT
$K(t)$	PSC kernel
I_0	Normalization constant for $K(t)$
τ_f, τ_r	Fall and rise time constants of PSC respectively
$I_{in}(t)$	Input current of a neuron
$V_m(t)$	Membrane voltage
V_{thr}	Threshold voltage
V_{st}	Subthreshold voltage
u	Hyperpolarization variable
τ_v, τ_u	Time constants governing $V_m(t)$ and u respectively
$s_{ij}(t)$	Presynaptic spike train
$r(t)$	Postsynaptic spike train
$\bar{s}, \bar{r}$	Presynaptic and postsynaptic traces respectively
τ_{pre}, τ_{post}	Fall and rise time constants of $\bar{s}$ and $\bar{r}$ respectively
n_{spk}	Spike output of neuron
$\Delta c_{ij}, c_{ij}$	Correlation value for a pattern, correlation averaged over all patterns
γ	Constant to balance potentiation and depression
C_{NL}	Capacity of NL-neuron
B	Number of distinct branch functions

Figure 5.1: (a) A neuron consisting of nonlinear dendrites characterized by the lumped nonlinear function $b()$ for every dendritic branch. It forms k binary connections from d input afferents on m dendrites. (b) Architecture of a multiclass pattern classifier consisting of nonlinear neurons receiving inputs from PDT and NDT and the neurons connected to a winner take all (WTA) to compare the outputs of the N_C classes. The output of the classifier is equal to 1 for the class corresponding to the highest output o^μ .

In this chapter, the structural learning rule for binary classification, introduced in Chapter 4, is extended to perform multiclass classification of spike patterns. Moreover, the learning rule is optimized by proposing an adaptive scheme which learns the number of dendrites in the network by progressively adding dendrites, thereby allocating synaptic resources according to the complexity of a given problem. This adaptive structural learning model is further optimized to achieve enhanced classification performance by developing an ensemble method which combines several individually trained dendritic classifiers. The chapter includes description of the model and learning algorithms, demonstration of performance of different learning schemes on classification of handwritten digit samples and a comparison with the results obtained using other spike-based classification algorithms, followed by a summary of the chapter.

5.1 Spike Classification with Nonlinear Dendrites & Structural Learning

A margin-based neuron model with nonlinear dendrites (NLDs) for spike pattern classification was proposed in Chapter 4. The model comprises nonlinear neurons having lumped dendritic nonlinearity where a nonlinear neuron (NL-neuron) consists of multiple (m) dendritic branches with each branch governed by its nonlinearity $b()$ and k excitatory synapses on each branch driven by one of the d input components (Figure 5.1(a)). The model uses binary weight for the i^{th} synapse on the j^{th} dendrite, $w_{ij} \in \{0, 1\}$, where $i \in \{1, \dots, d\}$, $j \in \{1, \dots, m\}$. The advantage of such binary weight connections are in hardware implementations since they require much less memory resources, are more resistant to mismatch in analog implementations and also enable easy digital implementations as done in [99]. Moreover, sparse connectivity is enforced on each dendritic branch ($k \ll d$) and allow the learning to choose the best ‘k’ connections on each branch. Hence, we can write:

$$\sum_{i=1}^d w_{ij} = k \text{ for } j = 1, \dots, m \quad (5.1)$$

This model was used to perform supervised binary classification of spike patterns. In this chapter, the binary classification model is extended to perform multiclass classification of spike patterns belonging to N_C classes. The multiclass classifier consists of (+) and (-) neurons corresponding to all N_C classes. The inputs to the neurons for class μ are received from a pair of excitatory and inhibitory dendritic trees, also referred to as positive (PDT) and negative dendritic tree (NDT) respectively, each with m dendrites. These input currents for class μ neurons are given by:

$$I_{in}^{\mu+}(t) = I_{PDT}^{\mu}(t) - I_{NDT}^{\mu}(t) \quad (5.2)$$

$$I_{in}^{\mu-}(t) = I_{NDT}^{\mu}(t) - I_{PDT}^{\mu}(t) \quad (5.3)$$

where $\mu \in \{1, \dots, N_C\}$ and $I_{PDT}^{\mu}(t)$, $I_{NDT}^{\mu}(t)$ are the currents generated by the PDT and NDT of class μ , calculated by taking the sum of dendritic output currents $I_{PD,j}^{\mu}(t)$ and $I_{ND,j}^{\mu}(t)$ respec-

tively, which can be expressed as nonlinear functions of the total synaptic current on a dendrite. To make notations simpler for the ease of reading, we drop the superscript μ for the μ^{th} class in the rest of the chapter except where the outputs of two or more classes are compared. Hence, the PDT/NDT currents for class μ neuron are denoted as $I_{DT}(t)$ and the output current of the j^{th} dendrite of PDT/NDT as $I_{D,j}(t)$. Therefore, the equations for $I_{DT}(t)$ can be written as:

$$I_{DT}(t) = \sum_j I_{D,j}(t) \quad (5.4)$$

$$= \sum_j b\left(\sum_i w_{ij} \left(\sum_{t_{ij} < t} K(t - t_{ij})\right)\right) \quad (5.5)$$

$$= \sum_j b(z_j(t)) \quad (5.6)$$

Here $K(t)$ denotes the postsynaptic current (PSC) generated by the spike input at a synapse of the PDT or NDT of class μ and is given by [2]:

$$K(t - t_{ij}) = I_0(\exp[-(t - t_{ij})/\tau_f] - \exp[-(t - t_{ij})/\tau_r]) \quad (5.7)$$

where t_{ij} denotes the times at which spikes arrive at this synapse; I_0 is the normalization constant; and τ_f and τ_r are the fall and rise time constants of the PSC respectively; $z_j(t)$ is the total synaptic activation on the j^{th} dendrite and $b(z_j(t))$ is the dendritic nonlinear function given by:

$$b(z_j(t)) = g(z_j(t) - z_{leak,j}(t))(z_j(t) - z_{leak,j}(t))^2 \quad (5.8)$$

Here, $g()$ is a Heaviside step function that gives an output 1 at all times where the argument $(z_j(t) - z_{leak,j}(t))$ is positive and 0 otherwise and $z_{leak,j}(t)$ is the average synaptic activation on the j^{th} branch corresponding to the initial random connections. The use of $z_{leak,j}(t)$ term serves to balance the excitation on each branch by subtracting the mean activation level on that branch, which can be regarded as a signal from a pool of inhibitory neurons. Finally, the spike output n_{spk} of a neuron of class μ , receiving the input current $I_{in}(t)$, is generated using the leaky integrate and fire neuron model described next.

The dynamics of the membrane potential $V_m(t)$ of the (+) and (-) neurons of class μ is explained through the equations given below:

$$\tau_V \frac{dV_m}{dt} = (u - V_m) + I_{in}(t) \quad (5.9)$$

$$\tau_u \frac{du}{dt} = -u \quad (5.10)$$

If $V_m \geq V_{thr}$, $V_m \rightarrow V_{reset}$;

$u \rightarrow u_{reset}$

$u_{reset} = V_{reset} < 0$

$n_{spk} \rightarrow n_{spk} + 1$

where u denotes a hyperpolarization variable which relaxes back to 0 with a time constant τ_u and is set to u_{reset} after a postsynaptic spike and τ_V and τ_u are the time constants governing the fast and slow dynamics of the membrane voltage and hyperpolarization respectively. Here, the variables $V_m(t)$, u and n_{spk} are used to describe the dynamics of both (+) and (-) neurons of class μ and the input current $I_{in}(t)$ can be computed using Equations (5.2) and (5.3).

As shown in Figure 5.1(b), the outputs of all the (+) and (-) neurons are connected to a WTA circuit to generate the overall classifier output as:

$$y_\mu = g(o_{spk}^\mu - o_{spk}^\nu), o_{spk}^\nu \geq o_{spk}^\xi, \forall \nu, \xi \neq \mu \quad (5.11)$$

where μ, ν, ξ are integers in the range $[1, N_C]$ and y_μ is the binary-valued μ^{th} component of the N_C -dimensional output y , $o_{spk}^\mu = n_{spk}^{\mu+} - n_{spk}^{\mu-}$; $n_{spk}^{\mu+}$ and $n_{spk}^{\mu-}$ are the spike counts of the (+) and (-) neurons of class μ respectively.

In the following sections, spike-time-based and rate-based learning schemes for our multi-class classification model with NLDs are developed. For these schemes using binary synapses, structural learning rules are presented, which involve the modification of connection matrix instead of connection weights. The spike-time-based learning requires significant processing time and therefore, to mitigate this problem we have used rate-based training using the reduced

model, described in Section 4.1.2 for binary classification. With the use of faster rate-based learning, we further propose an ensemble method which combines the outputs of several NLD classifiers in order to achieve performance gain over individual classifiers. An adaptive learning scheme is also developed which learns to allocate the required number of dendrites to a neural network along with learning the connection patterns on these dendrites. Next, the spike-based learning rule is described.

5.1.1 Spike-based Learning Scheme for Multiclass Classification

A biologically realistic branch-specific spike time dependent structural plasticity (BSTDSP) rule for binary classification was proposed in Chapter 4. Our learning rule is used to find the bio-realistic correlation values c_{ij} . The correlation values needed for structural plasticity rule are obtained from an online spike-based learning rule. Here, an extension of BSTDSP to multiclass classification is presented and later in Section 5.1.2, the correspondence between rate-based and spike-based learning approaches is demonstrated.

The multiclass BSTDSP is derived using the example of synchronous single-spike inputs such that each afferent either fails to fire or fires a spike at a fixed time T_{syn} and these spike inputs are assigned to N_C classes. The single-spike input is denoted by the presynaptic spike train $s_{ij}(t)$ at the i^{th} synapse of the j^{th} dendrite, and the postsynaptic spike train is denoted by $r(t)$ given by:

$$r(t) = \sum_{t_{post}} \delta(t - t_{post}) \quad (5.12)$$

where t_{post} is the postsynaptic spike time. These pre- and postsynaptic spikes also drive exponentially decaying memory traces—the presynaptic trace $\bar{s}$ and the postsynaptic trace $\bar{r}$ given by:

$$\bar{s} = \exp(-t/\tau_{pre}) \quad (5.13)$$

$$\bar{r} = \exp(-t/\tau_{post}) \quad (5.14)$$

If a pattern belonging to class μ is presented, a teacher signal forcing a postsynaptic spike

(at time $t = 1$ ms) is present for the (+) neuron and absent for (-) neuron of class μ . For the remaining $(N_C - 1)$ classes, opposite conditions for the teacher signal exist, i.e. teacher signal is absent for (+) neuron and present for (-) neuron.

The spike-based structural plasticity learning rule involves computing correlation values c_{ij} for each synapse belonging to the PDT and NDT corresponding to class μ . This learning scheme is analogous to the reverse-STDP rule such that if a postsynaptic spike arrives before a presynaptic spike, then the correlation value for that synapse is potentiated. The condition for depression does not use postsynaptic spike—instead, the relevant event is when the membrane voltage $V_m(t)$ crosses a subthreshold voltage V_{st} from below, where $0 < V_{st} < V_{thr}$. These subthreshold crossing events occurring at times t_{st} are denoted by $r_{st}(t) = \sum_{t_{st}} \delta(t - t_{st})$. The correlation value is decreased when presynaptic spike time t_{ij}^s occurs before t_{st} . The reason for using $r_{st}(t)$ instead of $r(t)$ is to enforce a margin as will be explained later in this section. Finally, the change in correlation value $\Delta c_{ij}(t)$ for every pattern presentation is computed using these plasticity conditions in a branch-specific manner. Hence, the learning rule for the PDT and NDT belonging to the μ^{th} class can be written as:

For PDT of class μ :

$$\Delta c_{ij}^{PDT}(t) = I_{D,j}^{PDT}(t) \bar{r}^{PDT}(t) s_{ij}^{PDT}(t) - \gamma I_{D,j}^{PDT}(t) \bar{s}_{ij}^{PDT}(t) r_{st}^{PDT}(t) \quad (5.15)$$

For NDT of class μ :

$$\Delta c_{ij}^{NDT}(t) = I_{D,j}^{NDT}(t) \bar{r}^{NDT}(t) s_{ij}^{NDT}(t) - \gamma I_{D,j}^{NDT}(t) \bar{s}_{ij}^{NDT}(t) r_{st}^{NDT}(t) \quad (5.16)$$

where γ is a constant used to balance the potentiation and depression and ensure that $c_{ij} = 0$ when a pattern is learned. The correlation values averaged over all the patterns can be written as: $c_{ij}^{PDT} = \langle \Delta c_{ij}^{PDT}(t) \rangle$ and $c_{ij}^{NDT} = \langle \Delta c_{ij}^{NDT}(t) \rangle$, where $\langle . \rangle$ indicates average calculated over an epoch. It was shown in Chapter 4 that by assuming the presynaptic time constant τ_{pre} to be much greater than the membrane time constant τ_V , the value of γ can be computed as:

$$\gamma = \frac{\exp(-T_{syn}/\tau_{post})}{\exp(-T_{rise}/\tau_{pre})} \quad (5.17)$$

where the presynaptic spike arrives at time T_{syn} and $V_m(t)$ crosses V_{st} at time T_{rise} . The connection changes are done on the basis of c_{ij} values using the same process as discussed for rate-based learning. The use of a subthreshold voltage V_{st} leads to a margin δ_{spike} for classifying spike patterns. A desired margin δ_{spike} for a particular class μ can also be preset, which can then be used to calculate the class specific values of V_{st} and V_{reset} as explained in Chapter 4. The class margins suited for a particular problem can be assigned by using margins set for rate-based learning. The value of the desired margin, δ_{spike} is determined by using the spike equivalent value of δ . This is done by computing the difference in the membrane voltage of the neuron (analogous to $a_{PDT} - a_{NDT}$) when a single synapse is activated. This value ΔV_m multiplied by δ gives δ_{spike} , since in the binary input case the synaptic strength is normalized to 1.

5.1.2 Rate-based Learning Scheme for Multiclass Classification

The rate-based learning rule for NLD model is valid for rate encoded inputs like Poisson spike trains and place/synchrony encoded single spike patterns. As discussed in Chapter 4, this validity results from the fact that for such spike inputs, the average synaptic activation $z_{syn,ij} = \frac{1}{T} \int_0^T \sum_{t_{ij}} K(t - t_{ij}) dt$ is directly proportional to the input arriving at that synapse, x_{ij} , where T is the pattern duration. This reduced model was developed to improve the training time and it uses encoded binary vectors obtained by mapping spike train with “high” firing rate or single spike to binary value ‘1’ and spike train with “low” firing rate or absence of spike to binary value ‘0’. Here, we present a rate-based NLD model for multiclass classification consisting of PDT and NDT corresponding to each of the N_C classes. The output of the PDT and NDT of class μ to input x is given by:

$$a_{PDT}(x) = \sum_{j=1}^m b \left(\sum_{i=1}^d w_{PDT,ij} x_{PDT,ij} \right) \quad (5.18)$$

$$a_{NDT}(x) = \sum_{j=1}^m b \left(\sum_{i=1}^d w_{NDT,ij} x_{NDT,ij} \right) \quad (5.19)$$

The difference of outputs of the positive and negative dendritic trees of each class, $o(x) = (a_{PDT}(x) - a_{NDT}(x))$, are then used as inputs to the WTA circuit which computes the overall output and therefore decision of the classification task, $y(x)$ using a similar logic as in Equation (5.11).

$$y_{\mu}(x) = g(o^{\mu}(x) - o^{\nu}(x)), o^{\nu}(x) \geq o^{\xi}(x), \forall \nu, \xi \neq \mu \quad (5.20)$$

During training, the classifier output is computed using a margin-based output function, $g_{margin}(x)$ instead of $g(x)$ to enforce a margin around the classification boundary. The $g(x)$ function is only used during the testing phase to calculate the output. Therefore, the classifier output during training calculated using the $g_{margin}(x)$ function is given by:

$$\begin{aligned} y_{\mu}(x) &= g_{margin}(o^{\mu}(x) - o^{\nu}(x)), o^{\nu}(x) \geq o^{\xi}(x) \\ y_{\nu}(x) &= g_{margin}(o^{\nu}(x) - o^{\mu}(x)), \forall \nu, \xi \neq \mu \end{aligned} \quad (5.21)$$

The function $g_{margin}()$ is defined as:

$$\begin{aligned} g_{margin}(\alpha) &= 1 \text{ if } \alpha \geq \delta \\ &= 0 \text{ if } \alpha \leq -\delta \\ &= \frac{0.5}{\delta} \alpha + 0.5 \text{ otherwise} \end{aligned} \quad (5.22)$$

where the margin is set using different values of the parameter δ for different classes to which input patterns belong. The value of margin δ for each class is determined using the following steps:

- (1) Multi-class model is first trained using the $g(x)$ function, where the output of the model is calculated using Equation 5.20. Connection matrices for all neurons are saved.
- (2) Cross validation set patterns are presented and the values $\alpha_{\mu\nu} = (o^{\nu}(x) - o^{\mu}(x))$ are recorded for all the cases for which patterns from class μ are misclassified as belonging to class ν .

- (3) The set of $\alpha_{\mu\nu}$ values for each class $\mu \in \{1, \dots, N_C\}$ are saved. The margin for class μ , δ^μ is set to the highest value of $\alpha_{\mu\nu}$.
- (4) δ^μ value is reduced to 80% of its present value whenever learning algorithm gets stuck in the same local minimum for 5 consecutive times.

The learning process which is based on the mechanism of structural plasticity involves formation and elimination of synaptic connections. The connections are modified by computing a metric based on correlation values c_{ij} for each synapse of the PDT and NDT. Hence, the learning rule for the rate-based learning scheme is given by:

For class μ , for PDT:

$$c_{ij}^{PDT} = \langle x_{ij}^{PDT} b_j^{PDT} \text{sgn}(y_\mu^d - y_\mu) \rangle \quad (5.23)$$

For class μ , for NDT:

$$c_{ij}^{NDT} = - \langle x_{ij}^{NDT} b_j^{NDT} \text{sgn}(y_\mu^d - y_\mu) \rangle \quad (5.24)$$

where the desired output y^d is available as the teacher signal during the training phase and is a N_C -dimensional binary vector consisting of $(N_C - 1)$ zeros and a 1 corresponding to the class to which the input pattern belongs; $\text{sgn}()$ is the signum function with a value of 1 for $y_\mu^d > y_\mu$, -1 for $y_\mu^d < y_\mu$ and 0 for $y_\mu^d = y_\mu$; and the output y is computed using Equation (5.21).

The connection changes are done by using the following logic: a synapse with the smallest c_{ij} value corresponds to a poorly-performing synapse and is a candidate for replacement. To replace this synapse, a set of silent synapses are first formed on the chosen branch as possible candidates for the new connection. The silent synapse with the highest c_{ij} is chosen as the replacement. At the start of the learning process, the input connections for all the PDT and NDT corresponding to N_C classes each with m dendritic branches, k synaptic contacts per branch and $s = m \times k$ total synapses are initialized by randomly selecting afferents from among d input lines with weight $w_{ij} = 1$. Training set consisting of P input patterns (x) belonging to N_C classes is presented. The learning process comprises the following steps in every iteration consisting of

the presentation of all P patterns:

- (1) The misclassification error rate is calculated by taking the average of the fraction of patterns for which $y_\mu \neq y_\mu^d$ for any $\mu \in \{1, \dots, N_C\}$.
- (2) For each dendritic tree of class μ , a random set T consisting of n_T ($< s$) synapses is selected as candidates for replacement.
- (3) The synapse with the lowest c_{ij} in T is targeted for replacement and the dendrite j_T on which it is located is identified. A random replacement set R is created by placing n_R ‘silent’ synapses from d input lines ($n_R < d$) on the j_T^{th} dendrite. The synapse with the lowest c_{ij} in T is replaced by the best-performing (maximum c_{ij}) synapse in R . The silent synapses do not contribute to the calculation in step (1).
- (4) Synaptic connections are modified if the replacement led to either a reduction or no change in error rate. If the error increased with the replacement, a new replacement set R is created. If the error does not decrease after repeating this step n_{ch} ($= 50$) times, we assume a local minimum is encountered. We then do a replacement with the last choice of R , even if it increases error in an attempt to escape the local minimum. The connection matrix corresponding to the local minimum that gave the least error is saved.
- (5) Steps (1) to (4) are repeated until either all the patterns have been memorized or n_{min} ($= 150$) local minima are encountered. At this point, the learning process is stopped. The connection patterns corresponding to the best minimum become the final learned synaptic connections of the neuron.

After the completion of training, the final learned connections are saved and then used to calculate the error rate on the spike test pattern set for the testing phase. This is done by mapping the learned connections onto an equivalent spiking network by introducing synaptic integration and spike initiation mechanisms. The testing classification output is computed using Equation (5.11).

The spike-based and the rate-based structural learning rules were compared in Chapter 4 and a relationship between the correlation terms $[\Delta c_{ij}]_{spike}$ and $[\Delta c_{ij}]_{rate}$ for the two forms of learning was derived as given below.

$$[\Delta c_{ij}]_{spike} = \exp(-T_{syn}/\tau_{post})[\Delta c_{ij}]_{rate} \quad (5.25)$$

Therefore, any differences between the performance of the two learning methods can be understood on the basis of this relationship. This suggests that as the value of τ_{post} is increased, the correlation values for the rate-based learning, $[\Delta c_{ij}]_{rate}$ well approximate the correlation values for the spike-based learning, $[\Delta c_{ij}]_{spike}$. Since the two forms of learning tend to coincide with increasing τ_{post} , therefore, the faster rate-based approach is used instead of the computationally prohibitive spike-based learning scheme for the enhancements proposed in the model.

5.1.3 Ensemble Learning Scheme for Multiclass Classification

An ensemble method is also used in this work, where several classifiers when combined together yield better classification accuracy than any of the single classifiers in the ensemble. Our ensemble model consists of individually trained NLD classifiers which are then combined to classify novel test patterns. Since, previous research has demonstrated that combining identical classifiers does not produce any gain over individual classifiers [100], hence an ensemble is created by combining several classifiers which are individually trained and disagree on their predictions. The complementary information about the novel patterns obtained from different classifiers can be exploited to produce a more accurate overall output. In order to generate different predictions for different classifiers, individual networks were initialized with different random synaptic connections. Figure 5.2 shows the basic framework for the classifier ensemble scheme. It consists of N individually trained multiclass classifiers as members of the ensemble. Each classifier generates the intermediate output, $o_n^\mu(x)$, which is the difference in outputs of PDT and NDT for class μ of the n^{th} classifier. The intermediate outputs are combined in a class-specific manner to give $O^\mu(x) = \sum_{n=1}^N o_n^\mu(x)$. Finally, the ensemble output is generated

Figure 5.2: Ensemble classifier combining the intermediate class-specific outputs o_n^μ to compute the combined class outputs O^μ , which are compared by the WTA to generate the final classifier output $\hat{y}$.

using:

$$\hat{y}_\mu(x) = g(O^\mu(x) - O^\nu(x)), O^\nu(x) \geq O^\xi(x), \forall \nu, \xi \neq \mu \quad (5.26)$$

The performance of ensemble model gives us an insight into the number of component classifiers needed in an ensemble. However, it is impractical to use a large number of classifiers in an ensemble due to the long training time required and also because it leads to increased synaptic resources. Therefore, we use the ensemble created with a few classifiers as a trade-off between trainability and accuracy. Moreover, it is not clear what level of complexity is required for individual classifiers and whether an ensemble of perceptrons will perform better than that of complex dendritic trees. To address this problem, an adaptive learning rule is developed for allocating the number of dendrites according to the difficulty of the problem being solved as well as the difficulty to learn a specific class. This scheme is discussed next.

5.1.4 Adaptive Learning Scheme for Multiclass Classification

For the learning schemes described above, a fixed number of dendrites (m) was chosen and assigned to all the dendritic trees in the model. Since, it is difficult to choose an optimal value of m that matches the complexity of a given problem, an algorithm is proposed in which the number of dendrites corresponding to each class is learned during the training process. The value of m is adapted for each class dendritic trees independently while the number of synapses per branch, k , is kept constant throughout. Hence, the learning process generates the relative sizes of the connection matrix for each neuron as well as the optimal sparse connections within that matrix. The learning rule used is same as that for the rate-based scheme with fixed m except that all the computations in every iteration are done for the current value of m for each neuron. Additional steps are included in the learning algorithm to adapt the value of m . After making a correlation-based connection change, we check if the learning process for a particular class μ has slowed down using the following steps:

- (1) If the error rate for class μ , which is calculated as the average of the number of patterns for which $y_\mu \neq y_\mu^d$, does not decrease after n_{ch} ($= 50$) iterations, the learning process for class μ has encountered a local minimum. In this case, a new dendrite is added to both PDT and NDT corresponding to class μ , if the error is the highest amongst all the N_C classes.
- (2) This is done by drawing k random connections from the d input lines to represent a new dendrite. The connection matrix is updated by appending the new dendrite to the existing matrix as a new row. The value of m for class μ is incremented according to:

$$\begin{aligned} m^{PDT} &\rightarrow m^{PDT} + 1 \\ m^{NDT} &\rightarrow m^{NDT} + 1 \end{aligned} \tag{5.27}$$

- (3) After adding a dendrite, error rate for cross validation set is computed. If this validation error increases in the last 3 dendrite addition steps, the learning process is stopped. The

connections are frozen and saved for the testing phase.

The above steps are added after the step (4) of the rate-based learning algorithm in Section 5.1.2 as steps 4.1, 4.2 and 4.3. All the steps from (1)-(4.3) are then repeated until learning stops by one of the stopping conditions discussed in step (5).

This adaptive learning method is guided by the level of difficulty of each neuron's classification task and moreover, it is used to learn the relative size of each neuron and not the optimal neuron topology. While performing this adaptation, we keep the number of synapses per branch, k , to be a constant for all neurons and just vary the number of dendrites per neuron, m . Hence, at the end of the adaptation, both PDT and NDT of the μ^{th} class have a total number of synapses s_{adapt}^μ given by $s_{adapt}^\mu = m_{adapt}^\mu \times k$, where m_{adapt}^μ is the number of dendritic branches learned for the μ^{th} class neurons after adaptation. This is shown as the step-1 in Figure 5.3, where the topology of a class neuron is denoted by the rectangle with its sides representing m and k values.

However, from theoretical considerations of function counting as shown in [12], the theoretical capacity of a neuron with nonlinear dendrites with a fixed value of total number of synapses (s_{opt}) is maximum for a relatively large value of m_{opt} and small value of k_{opt} . This theoretical capacity was given by the combinatorial expression derived by counting all possible ways in which d afferents can connect to synapses on dendrites resulting in distinct memory fields [12]. Hence, for a neuron with NLDs, the capacity C_{NL} in bits was calculated as:

$$C_{NL} = 2\log_2 \binom{B+m-1}{m} \quad (5.28)$$

where $B = \binom{k+d-1}{k}$ is the number of distinct branch functions. As shown in the inset plot for the optimal topology determination step in Figure 5.3, C_{NL} plotted as a function of m has a maximum at m_{opt} for a fixed value of s_{opt} . These derivations were also discussed in Chapter 4. We can now use this theory to change the structure of the μ^{th} neuron to optimize capacity while preserving the same number of synapses. Hence, by setting $s_{opt}^\mu = s_{adapt}^\mu$ (fixing the area of rectangle as shown in step-2 in Figure 5.3), the theoretical optimal topology (m_{opt}^μ, k_{opt}^μ) is determined such that $s_{opt}^\mu = s_{adapt}^\mu = m_{opt}^\mu \times k_{opt}^\mu$. The use of this approach to boost the capacity

Figure 5.3: Dendritic classifier is trained adaptively by adding dendrites (step-1) and keeping the number of synapses/dendrite (k) as a constant for all neurons. The total number of synapses for the μ^{th} neuron learned in this manner (s_{adapt}^μ) is fixed and the corresponding optimal topology (m_{opt}^μ, k_{opt}^μ inset plot) is theoretically determined (step-2). The length of sides of the rectangle denote m and k values and the area represents the total number of synapses, s .

of our adaptively learned dendritic network is demonstrated in Section 5.2.8.

5.2 Results

5.2.1 Input Generation

The performance of our multiclass classification NLD model is demonstrated on the MNIST dataset consisting of grayscale images of handwritten digits belonging to one of the $N_C = 10$ classes (0 to 9). It is a benchmark machine learning dataset used previously to test the performance of many classification algorithms [101]. The input patterns consisting of 28x28 images were converted into $d = 784$ dimensional binary vectors by thresholding. The training set consists of 20,000 patterns randomly selected from the full MNIST dataset, with equal number of samples of each digit. The testing set consists of a total of 10,000 patterns. In case of adaptive learning scheme, 20% of the training patterns comprise the cross validation set which is used to compute the error whenever a dendrite is added to the network and finally to stop adding dendrites when the cross validation error does not reduce further. The learning procedure is stopped when either all the patterns are correctly classified or when 150 minima are encountered. For different training runs 150 minima are encountered in different number of learning iterations

Table 5.3: Parameters for Rate-based Multiclass Model

d	m	k	n_R	n_T	T_{syn}
784	10	10	25	25	100 ms

Table 5.4: Parameters for Spike-based Multiclass Model

τ_V	τ_u	τ_{pre}	τ_{post}	V_{thr}	T	T_{syn}
5 ms	200 ms	10 ms	200 ms	0.1 mV	200 ms	100 ms

and for each data point generated, this training process is repeated 3 times.

The binary input vectors are used for rate-based training. For spike-based training, patterns of single spikes are used, which are generated by mapping binary input ‘ $x_i = 1$ ’ to a single spike arriving at $T_{syn} = 100$ ms and ‘ $x_i = 0$ ’ to no spike, where the stimulus duration $T = 200$ ms. Testing for both forms of rate-based and spike-based learning is done on two types of spike inputs: 1) place/synchrony code of single spikes with jittered spikes arriving within a time window $[T_{syn} - \frac{\Delta}{2}, T_{syn} + \frac{\Delta}{2}]$ corresponding to binary input ‘1’ and no spike for ‘0’ input and different amounts of jitter Δ ; and 2) rate encoded Poisson spike trains with mean firing rates of f_{high} and f_{low} mapped to binary inputs ‘1’ and ‘0’ respectively. The Poisson spike inputs have only been used to test the noise sensitivity of the model. The parameters used for the rate-based and spike-based models are given in **Table 5.3** and **Table 5.4** respectively, unless stated otherwise.

The significance of using these spike inputs can be understood by considering that single spike representations are commonly used in Time-to-First-Spike (TTFS) imagers [83, 102]. These imagers when presented with binary images such as in the MNIST dataset used in this work, generate a cluster of spikes corresponding to the white pixels and another cluster much later in time corresponding to the black pixels. Such place/synchrony code is also abundantly used in neuroscience [103]. Moreover, rate encoded outputs are commonly available from neuromorphic sensors such as the artificial cochlea in [104]. Such rate encoded Poisson spike trains are also often used to test the performance of neuromorphic classifiers as demonstrated in [15, 64, 105].

Figure 5.4: Training and testing error rates for rate-based learning as a function of the training set size. Test performance is measured on 10,000 jittered single spike input patterns with $\Delta = 10$ ms. Each data point is obtained by averaging over 3 training simulation trials and testing on 10 randomly drawn test sets for each learned network; $m = 10$ and $k = 10$.

5.2.2 Effect of Training Set Size

In the first experiment, the effect of training set size on the classification performance is measured by training the model on $P = 200 - 20,000$ patterns using the rate-based learning scheme. Each training accuracy is obtained by averaging over 3 trials and the testing accuracy is computed by averaging across 10 presentations of 10,000 test patterns for each of the learned network, where testing is done on jittered single spike inputs with $\Delta = 10$ ms. The training was done on a network consisting of $m = 10$ dendrites in the PDT and NDT of all $N_C = 10$ classes. Hence, the total number of dendrites used is $M = 200$. As shown in Figure 5.4, for the rate-based learning, the training error increases while the testing error on spike versions of the inputs reduces as the training set size increases. It is clear from this result that as more data is added, it becomes difficult to memorize the data. However, the generalization performance improves with more training data.

5.2.3 Comparison of Spike-based Learning and Rate-based Learning Schemes

The performance of the spike-based and rate-based learning schemes is compared. Since spike-time-based learning takes very long and the training time increases with the size of train-

ing set, therefore, the spike model is trained on smaller number of training patterns, 200 – 1000. For both forms of learning schemes, the networks consisted of $m = 10$ dendrites in the PDT and NDT of all $N_C = 10$ classes. Hence, the total number of dendrites used by both models is $M = 200$. Figure 5.5(a) shows the comparison between the performance of spike-based and rate-based learning schemes. Both forms of learning exhibit the trend of decreasing test error with the training set size. However, the spike-based learning method results in about 1.5 times higher errors than that for rate-based learning. This difference in performance of the two methods can be attributed to three reasons: 1) Firstly, the relationship between these two methods (Equation (5.25)) suggests that as the value of τ_{post} is increased, the correlation values for the two learning methods become more similar. Therefore, we need to repeat our spike learning simulations with higher values of τ_{post} to achieve greater agreement between the two forms of learning; 2) Secondly, our estimation of γ in Equation (5.17) is based on the assumption that $\tau_{pre} \gg \tau_V$. Hence, smaller τ_{pre} will result in a non-ideal value of γ which will give different results for the two learning schemes; and 3) Finally, the discrepancy in results can arise if the margin setting is not exact. As discussed above, the margin for spike-based learning was set using $\delta_{spike} = \Delta V_m \times \delta$, where a wrong estimate of the constant ΔV_m will mean that δ_{spike} and δ are not truly analogous to each other, leading to different results.

Based on this discussion, we attempted to reduce the difference between the performance of rate-based and spike-time-based learning approaches. Therefore, we increased the value of τ_{post} from 200 to 500, 1000 and 2000 ms and τ_{pre} from 10 to 50 ms. As shown in Figure 5.5(b), the test error for $\tau_{post} = 200$ ms corresponds to our prior parameter settings for $P = 200$ patterns (Figure 5.5(a), blue curve). As τ_{post} is increased, the testing performance of spike-based learning rule improves and gradually becomes closer to the testing performance of rate-based method, shown with blue line. We expect that we can achieve further agreement between the two forms of learning by getting the margin values δ_{spike} and δ to be analogous to each other or by determining the desired δ_{spike} directly in the spike domain. Hence, it has been shown that the error rate for spike-based learning becomes similar to that of rate-based learning if τ_{post} is long enough. However, due to the long simulation times of the spike-based learning, rate-based

Figure 5.5: (a) Comparison of performance of rate-based and spike-time-based learning schemes. The test error rates for the spike-based learning rule are about 1.5 times higher than the rate-based test errors; (b) Comparison of test error rate for $P = 200$ patterns learned with rate-based (blue line) and spike-based (red plot) methods as a function of τ_{post} . Each data point is obtained by averaging over 3 training simulation trials and testing on 10,000 single spike inputs with jitter $\Delta = 10$ ms; $m = 10$ and $k = 10$.

learning has been used for the remaining analyses in the chapter.

5.2.4 Performance of the Ensemble Method

Next, the effect of combining several NLD multiclass classifiers in an ensemble is studied. These individual classifiers were trained on $P = 5000$ patterns using rate-based learning rule and tested on 10,000 jittered single spike inputs with $\Delta = 10$ ms. The network consisted of $m = 2, 5, 8, 15$ and 20 dendrites in PDT and NDT for each of $N_C = 10$ classes. Hence, the total number of dendrites used in the ensemble is given by $M = 2 \times m \times N_C \times N$. As shown in Figure 5.6(a), the error rate for $m = 2$ reduces by about 48% when up to 25 classifiers are added in the ensemble. However, the error rate does not change by much or it increases slightly if further classifiers are combined. Moreover, most of the error reduction, 40% out of the total change of 48%, is achieved when first 5 classifiers are added. We have also looked at the effect of size of individual networks in the ensemble. Figure 5.6(b) illustrates this effect where error rate is plotted as a function of the total number of dendrites in the ensemble, M . As shown, the error rate reduces with the number of classifiers for all values of m . For a fixed value of M , $m = 2$ gives the highest error rate, which reduces with larger values of m . However, as m is

Figure 5.6: (a) Ensemble error rate as a function of the number of classifiers combined, N , for 10,000 test samples; $m = 2$ and $k = 10$. (b) Ensemble error rate as a function of the total number of dendrites used, M , showing that a moderate number of dendrites per dendritic tree ($m = 8$) gives reasonably good performance and there is no significant improvement when m is further increased. Training done on $P = 5000$ patterns randomly drawn from the MNIST samples. Each data point is obtained by randomly selecting a subset (N) of the total number of learned classifiers for each case of m (37 for $m = 2$). This process is repeated 10 times and the average of the test errors of the combined classifiers is computed.

increased beyond a certain value ($m = 8$ in this case, the reduction in errors is not significant, and therefore the use of more complex dendritic trees is not leading to significant advantage in terms of performance.

These results indicate that the ensemble model can attain reasonably good performance when the member classifiers are trained using a moderate number of dendrites. In an ideal scenario, a large number of classifiers can be used to achieve significant improvement in performance. However, it is impractical to train several classifiers and hence, only a few classifiers are used which results in most of the performance gain of the ensemble. In case of limited resources (total number of dendrites M), an intermediate level of complexity of the ensemble comprising a moderate number of member classifiers with a moderate number of dendrites in each classifier can be used. Next, we investigate how to determine this intermediate level of complexity for a particular classification problem. For an ensemble created with a few classifiers (3 – 5), we do not know the number of dendrites required by each classifier and also further, if different levels of network complexity underlie representation of different classes to which data belongs. Therefore, the adaptive learning rule is used to automatically learn the number of dendrites

suitable for each class of a given classification problem.

5.2.5 Performance of the Adaptive Learning Scheme

The adaptive learning scheme to learn $P = 20000$ patterns was used by initializing the network with $m = 5$ dendrites in each dendritic tree of a class, and then adaptively increasing m in a class-specific manner. The learning process was stopped when 150 minima were encountered and therefore, each adaptively learned classifier took different number of iterations to complete 150 minima. If dendrites are added to only 5 worst-performing classes whenever their learning slows down, referred to as scheme-1, the accuracy obtained on 10,000 test single spike inputs is 92.1%. The number of dendrites learned by each class in one simulation run is shown by the bar plot at the top of Figure 5.7(a). It can be seen that the digits '2', '3', '5', '8' and '9' use most of the dendrites and hence are most difficult to learn while '0', '1' and '6' are the easier ones requiring only a small number of dendrites. The confusion matrix at the bottom of Figure 5.7(a) shows the classification accuracy when an actual digit (column-wise) is predicted by the model as represented row-wise. It can be seen that the neurons for easier digits '0', '1' and '6' can attain good accuracy by utilizing small number of dendrites whereas the difficult digits like '8' and '9' exhibit lower accuracy, where '9' is mostly misclassified as '4' and '7' having similar features.

A drawback of scheme-1 is that we are not allocating the resources appropriately in this case since adding dendrites to only "difficult to learn" classes might not be helpful beyond a point and adding dendrites to the easier classes also can allow the learning to converge faster and attain better performance. This idea was supported by our results obtained by adding dendrites to all the classes, referred to as scheme-2, which yielded a classification accuracy of 94.2%. The number of dendrites learned by all classes is shown at the top of Figure 5.7(b), indicating that the dendrites are added to not only the difficult classes but the easier digits like '0', '1' and '6' also utilize more resources. The confusion matrix shows that the classification accuracy of almost all the digits increase as compared to scheme-1, thereby contributing to the overall improved performance.

Figure 5.7: (a) The number of dendrites learned using adaptive scheme-1 (top) and the resulting confusion matrix (bottom) showing higher accuracy for easier classes (0, 1, 6) and lower accuracy for difficult digits (8, 9). (b) Number of dendrites learned using adaptive scheme-2 (top) demonstrating all the class neurons adding dendrites when required. Confusion matrix (bottom) showing improved accuracy for all the classes. Training set consists of 20,000 binary digit samples and testing done on 10,000 jittered single spikes ($\Delta = 10$ ms). Model was initialized with $m = 5$ dendrites for all the classes.

5.2.6 Test of Noise Sensitivity

The performance of an ensemble created with classifiers learned using the adaptive scheme-2 is tested in the presence of noisy spike inputs. Figure 5.8 shows the classification accuracy for 10,000 test patterns consisting of single spike inputs with different levels of jitter Δ and Poisson spike inputs with $f_{high} = 250$ Hz and $f_{low} = 1$ Hz. For reference, we also present the testing accuracy obtained when non-spiking binary inputs are used (solid blue), which increases from 94.6% to 96.1% by adding just 3 classifiers to the ensemble and further increases to about 96.4% by adding 2 more classifiers. The accuracy for single spikes with no noise ($\Delta = 0$) is about 0.2 – 0.5% less than that of binary vectors, which can be attributed to the effects of integrate and fire process of the neurons in the model. As noisy single spikes are presented, accuracy reduces further. However, the change is only about 0.5% for $\Delta = 20$ ms as compared to $\Delta = 0$. Further, the performance of the ensemble on spiking inputs with and without noise

Figure 5.8: Error rate of ensemble consisting of NLD classifiers trained on $P = 20,000$ MNIST samples using the adaptive scheme-2. The ensemble accuracy on 10,000 test patterns is computed for binary vectors and spike inputs with different levels of noise. The accuracy obtained for single spike inputs with time window size $\Delta = 20$ ms is 96.1% by combining 5 classifiers. For Poisson spike inputs, $f_{high} = 250$ Hz and $f_{low} = 1$ Hz. Each data point is obtained by randomly selecting a subset (N) of the total number of learned classifiers (5). This process is repeated 5 times and the average of the test errors of the combined classifiers is computed.

become very similar when just 3 dendritic classifiers are combined together. Similarly, for Poisson spike inputs the ensemble yields similar classification accuracy as in the absence of noise. Hence, the ensemble consisting of a few classifiers where each classifier is individually trained using the adaptive learning scheme offers the following advantages: 1) performance gain; 2) feasible to train few classifiers; 3) automatically learned network complexity; 4) noise sensitivity and 5) limited resources due to the use of a few classifiers with adaptively learned network size.

5.2.7 Analysis of Dendrite Weights: Feature Maps

The weights of dendrites were analyzed to understand the features of inputs learned by the nonlinear dendrites. Since our learning rule encourages correlated inputs to be grouped together on the same dendrite, we expected the dendrite weight maps to capture the input correlations. For this purpose, the $d = 784$ -dimensional binary weight vector $w_{.j} = [w_{1j} w_{2j} \dots w_{dj}]$ for the j^{th} dendrite was mapped to a 28×28 matrix consisting of binary elements (since there can be multiple connections from the same afferent on a dendrite, the weight vector can have integer

Figure 5.9: Weight maps of all dendrites of neurons belonging to a particular digit class (top row). Maps in the remaining rows show individual dendrite weights with completely or partially learned features of input digits.

values > 1). For this analysis, 20,000 patterns were trained using the adaptive scheme-2 by initializing the network with $m = 5$ dendrites for each class.

The weight matrices are visualized as shown in Figure 5.9. The first row depicts the mean weights of all the dendrites corresponding to a digit. It can be seen that the dendrites learn the features of digit images presented to the network. The dendrites belonging to a particular digit class together represent the complete digits. The feature maps look pixelated due to sparse integer weights learned by each dendrite. The remaining rows show the weights of 4 individual dendrites, m_1 to m_4 , randomly selected out of the total learned dendrites for each digit. These maps demonstrate that each dendrite learns some features of the input digits. However, some of the dendrites like m_4 of digit '3' and m_3 of digit '7' can learn to represent complete digits, which suggests that some of the dendrites for these classes are redundant and hence, can be removed. A hybrid approach which involves adding new dendrites and pruning the superfluous ones can therefore be used to design the network.

Figure 5.10: (a) Theoretical capacity of neuron corresponding to digit ‘5’ as a function of number of dendrites (m). The number of dendrites learned adaptively is m_{adapt}^5 and the optimal number of dendrites corresponding to maximum capacity is m_{opt}^5 . (b) Adaptively learned and optimal m values for digit ‘7’. (c) Performance comparison of adaptively learned network (blue) with theoretically determined optimal network (red). Training was done on 20,000 binary digit samples and test performance was measured on 10,000 binary inputs (solid) and jittered single spike input patterns with $\Delta = 10$ ms (dashed).

5.2.8 Boosting the Performance using Optimal Neuron Topology

Next, the theoretically optimal configuration of the network is determined which can be used to boost the performance of our structural learning scheme, as discussed in Section 5.1.4. For this analysis, we used one instance of the final number of dendrites learned for each class using the adaptive scheme-2 (Figure 5.7(b)). We then computed the theoretical capacity for each neuron of all classes by fixing the total number of synapses learned by the μ^{th} neuron, s_{adapt}^μ , and varying the m and k values such that $s_{adapt}^\mu = s_{opt}^\mu = m \times k$ (step-2 in Figure 5.3). Figure 5.10(a) shows the theoretical capacity of class ‘5’ neuron plotted as a function of m , where m_{adapt}^5 is the number of dendrites learned using the proposed adaptive method while m_{opt}^5 is the number of dendrites corresponding to the maximum capacity for a neuron with the same $s_{opt}^5 = s_{adapt}^5$ as that of a neuron trained adaptively. The number of dendrites, m_{adapt}^7 learned by our method and the corresponding optimal value, m_{opt}^7 for class ‘7’ neuron are shown in Figure 5.10(b).

We then trained a network with neurons consisting of m_{opt}^μ dendrites on 20,000 MNIST samples using our structural learning rule. Figure 5.10(c) shows the comparison between the per-

formance of our adaptively learned network (blue) and the corresponding theoretically optimal network (red). These classifiers were combined in ensembles and tested on 10,000 non-spiking binary inputs and jittered single spike patterns. The results show that the optimal topology can achieve about 0.5 – 0.9% higher accuracy than the adaptively learned configuration. Moreover, the use of optimal m and k values also helps to generate the highest accuracy of our model, 96.6%, obtained on the binary test inputs. Further advantage of using a network with optimal topology can be understood by comparing these results with the accuracies depicted in Figure 5.8. We can see that the ensemble of 3 classifiers with optimal network topology can attain higher accuracy than the ensemble of 5 adaptively learned classifiers. Hence, theoretical capacity predictions of the optimal network configuration can be used to boost the classification performance of our structural learning rule while using same synaptic resources as the adaptive case.

5.2.9 Comparison with Other Spike-based Classification Models

Finally, the performance of our model with other spike classification models consisting of a network of spiking neurons is compared. The spike classifiers considered here use the two main approaches of rate-based and spike-based learning. The models consisting of spiking RBMs map the weights learned using offline rate-based scheme onto spiking neural network in [15] and use a spike-based event-driven learning rule based on STDP in [22]. In [64], a stochastic spike-driven synaptic plasticity rule was used to train a network of binary synapses and classification accuracy determined by voting over a pool of neurons.

Table 5.5 compares the number of neurons, number of dendrites used by our dendritic models, total number of nonlinearities in the model (#NL), synapses used and the accuracy attained by these classification algorithms on MNIST dataset. The number of neurons mentioned is the total number of neurons in the network excluding those in the input layer. For our models, the number of NLDs are also shown while, for the other networks, all synaptic currents sum up linearly implying the use of only one linear dendritic branch per neuron and no NLDs. The number of training examples used by [22, 64] and our method is 20,000 while the training set

Table 5.5: Comparison with Spike Classifiers on MNIST Data

Model	#Neurons	#Dendrites	#NL	#Synapses	Accuracy
[15] (Poisson spikes)	1010	No dendrites	1010	647,000	94.09%
[22] (Poisson spikes)	540	No dendrites	540	412,000	91.9%
[64] (Poisson spikes)	150	No dendrites	150	117,600	96.5%
Adaptive NLD (Single spikes)	20	440	460	6720	94.2%
Ensemble NLD (Single spikes)	100	2312	2412	35285	96.1%
Optimal NLD (Single spikes)	100	3960	4060	20163	96.4%

in [15] consisted of 120,000 samples generated by introducing small random translations, rotations and scalings in the original MNIST training examples. The number of test patterns used by all the models is 10,000.

The results suggest that the NLD model can achieve accuracy comparable with other algorithms by utilizing significantly less, which is about 2 – 7% of the total number of synapses used by the other methods. Moreover, the use of low resolution integer weights in our work in contrast with the high resolution weights used by spiking Restricted Boltzmann Machine (RBM) models [15, 22] renders our structural learning rule for NLDs more feasible for implementing spike classification in hardware. A recent study showed that a two-layer spiking Deep Belief Network (DBN) having 642,150 synapses can tolerate reduced bit precision of the weights up to 4 bits by maintaining the network performance at 94.95% [23]. The reduced precision leads to about 50% of the weights becoming zero and hence, getting pruned. However, the number of synapses used is still much higher (more than 50 times) than the number of 1-bit weights used by our model achieving similar performance. Further, the ensemble of NLD classifiers yielded 96.1% accuracy by using rate-based learning and testing on Poisson spike inputs. Moreover, the use of optimal network topology, which was determined using the theoretical capacity cal-

culations, enabled us to attain even higher accuracy by reducing the synaptic resources by about 1.7 times. These results are at par with the best performance achieved by [64]. It is important to note that both models in [64] and our work use binary synapses; however, we obtain similar performance as in their work by training an ensemble of only a few classifiers resulting in still less number of synaptic resources, about 20% of that used in [64]. We also expect that our learning rule will be more amenable for hardware implementation since the dendritic polynomial non-linearity is much simpler than implementation of a full neuron as in the population of output neurons in [64]. Therefore, the use of much less number of synaptic resources with binary weights and a more simpler learning rule render our NLD model more hardware-friendly.

5.3 Summary

This chapter proposed extensions and enhancements in the structural learning rule for neuron with nonlinear dendrites. The binary classifier of Chapter 4 was extended to a multiclass classifier by employing additional neurons with one opponent pair corresponding to each class. The main contributions of this chapter are: 1) branch-specific spike-time-based structural learning method, followed by; 2) rate-based multiclass learning algorithm; 3) an ensemble method combining the outputs of several dendritic classifiers showing that for ensembles of NLDs with the same number of synapses, having a single dendritic branch (perceptron) or having too many dendrites are sub-optimal; the optimal case is a moderate size of the dendritic tree; 4) an adaptive structural learning scheme to learn the required moderate number of dendrites; 5) evaluation of different learning schemes on classification of handwritten digit samples; 6) analysis of dendrite weights to visualize the features learned by individual dendrites, and finally; 7) comparison of the performance of our model with the results obtained using other spike-based classification algorithms showing that our model can achieve performance which is at par with other methods by using much less synaptic resources. The results showing classification performance of our method on recordings of MNIST digits using a dynamic vision sensor, are provided in the Appendix B.

5.4 Publications

1. **Shaista Hussain** and Arindam Basu, “Multi-class Classification by Adaptive Network of Dendritic Neurons using Structural Plasticity,” *Frontiers in Neuroscience*, vol. 10, no. 113, 2016.
2. **Shaista Hussain**, Shih-Chii Liu, and Arindam Basu, “Improved Margin Multi-Class Classification Using Dendritic Neurons with Morphological Learning,” *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2640-2643, 2014.

Chapter 6

Combined Effects of Delay Adaptation and Structural Learning: Initial Explorations

List of Acronyms

Acronym	Description
NLD	Nonlinear Dendrite
NL-neuron	Nonlinear Neuron
PDT	Positive Dendritic Tree
NDT	Negative Dendritic Tree
PSP	Postsynaptic Potential
MAE	Mean Absolute Error

List of Variables

Variable	Description
m	Number of dendrites in a NL-neuron
k	Number of synapses per dendrite

Variable	Description
s	Total number of synapses
d	Number of input components
P	Number of input patterns
n_d	Number of delay compartments
t_d	Time delay per compartment
w_{pqr}	Weight for q^{th} synapse on r^{th} compartment of p^{th} dendrite
W_p	Weight matrix for p^{th} dendrite
T_d^r	Total delay at r^{th} compartment
$b()$	Dendritic nonlinear function
T_s	Pattern duration
t_q	Spike times at q^{th} afferent
$K(t)$	PSP kernel
V_0	Normalization constant for $K(t)$
τ_f, τ_r	Fall and rise time constants of PSP respectively
$V_p(t)$	Membrane voltage generated on p^{th} dendrite
$V_D(t)$	Voltage on a dendritic tree
V_{out}	Voltage difference of PDT and NDT
V_{thr}	Threshold voltage
V_{max}, t_{max}	Maximum of V_{out} , time of V_{max}
y	Neuron output
y^d	Desired output
$\Delta c_{pqr}, c_{pqr}$	Correlation value of a synapse for a pattern, correlation averaged over all patterns
C, C_{multi}	Capacity of NL-neuron without and with delay compartments
B, B_{multi}	Number of distinct branch functions of NL-neuron without and with delay compartments

This chapter investigates the combined effects of delay adaptation and structural learning in a neuron model consisting of dendrites endowed with passive filtering presented in Chapter 3 and active ionic mechanisms described in Chapters 4 and 5. Hence, each nonlinear dendrite in this model is divided into multiple location-dependent delay compartments, where the structural learning will lead to the aggregation of coactive inputs on a dendrite while the delay modification will enable these inputs to optimally connect to specific dendritic compartments. The formation of synaptic connections on spatially sensitive multicompartment dendrites can result in increased storage capacity as compared to the capacity of a model with single compartment lumped dendrites, as shown in this chapter. Multicompartment dendritic model is introduced in this chapter, followed by different learning schemes used to train this model to perform spike pattern classification. The performance of these schemes is compared and finally, a summary of the chapter is given.

6.1 Multicompartment Model of Nonlinear Dendrites

A multicompartment model of dendrites is presented where it is assumed that the nonlinear ionic mechanisms are not available within these compartments and only at the proximal ends of dendrites. Therefore, only passive signal propagation is considered along dendrites resulting in time delay t_d per compartment. A delay learning scheme was proposed in Chapter 3 for a spiking neuron model which can be considered as comprising a single dendrite lacking nonlinear processing where spike inputs arrive with delays associated with the afferents connecting on this dendrite.

Let us consider a neuron with m NLDs which receive excitatory synaptic contacts from d input afferents forming on one of the n_d delay compartments on each dendrite (Figure 6.1(a)). The model utilizes binary weights such that for the synapse formed from the q^{th} afferent on the r^{th} compartment of the p^{th} dendrite, $w_{pqr} \in \{0, 1\}$, where $p \in \{1, \dots, m\}$, $q \in \{1, \dots, d\}$ and $r \in \{1, \dots, n_d\}$.

Figure 6.1: Architecture of the multicompartment model. (a) A neuron with multiple nonlinear dendrites (m), each with multiple compartments (n_d) forming k binary sparse connections from d input lines ($k \ll d$). (b) Binary classifier consisting of PDT and NDT, generating the output y .

The weight matrix for a dendrite is denoted by W_p , which is a $d \times n_d$ matrix. Here, the compartment number denoted by r is counted outwards from the soma. Therefore, for the compartment nearest to the soma known as the most proximal compartment, $r = 1$ and for the compartment farthest away from the soma or the most distal compartment, $r = n_d$. We use a convention where the first compartment has zero delay, $t_d|_{r=1} = 0$. Sparse connections are enforced on each dendrite such that:

$$\sum_q \sum_r w_{pqr} = k, \text{ where } k \ll d \quad (6.1)$$

If t_q denotes the time at which a spike arrives at the q^{th} afferent, membrane voltage generated due to the afferents connected on the p^{th} dendrite can be written as:

$$V_p(t) = \sum_q w_{pqr} \sum_{t_q < t} K(t - t_q - T_d^r) \quad (6.2)$$

where the total delay at the r^{th} compartment $T_d^r = t_d \times (r - 1)$ and the postsynaptic potential (PSP) generated by the input spikes at a synapse is denoted by the kernel function $K(t)$ used in Section 3.1.1:

$$K(t) = V_0(\exp(-t/\tau_f) - \exp(-t/\tau_r)) \quad (6.3)$$

V_0 is the normalization constant; and τ_f and τ_r are the fall and rise time constants respectively. The dendritic output voltage given by $b(V_p(t))$ is summed across all dendrites to give the voltage on a dendritic tree, written as:

$$V_D(t) = \sum_p b(V_p(t)) \quad (6.4)$$

where $b()$ is the dendritic nonlinear function. The soma of the neuron receives inputs from positive dendritic tree (PDT), denoted by (+) and negative dendritic tree (NDT), denoted by (-). These inputs are then combined ($V_{out} = V_D^+(t) - V_D^-(t)$) to generate the neuron output, y . This scheme can be used to perform supervised binary classification of spike patterns (Figure 6.1(b)) where the output of the classifier y is defined as:

$$\begin{aligned} \text{If } V_{max} \geq V_{thr}, y &= 1 \\ \text{else } y &= 0 \end{aligned} \quad (6.5)$$

where $V_{max} = \max(V_D^+(t) - V_D^-(t))$; $V_D^+(t)$, $V_D^-(t)$ are the voltages of the PDT and NDT respectively and V_{thr} is the threshold voltage.

6.2 Theoretical Capacity of Multicompartment Model

The theoretical capacity of a lumped nonlinear dendrite (NLD) model can be estimated as the number of possible ways in which input afferents can connect to synapses on dendrites resulting in distinct memory fields. As discussed earlier in Section 4.2, the combinatorial expression to count the number of distinct I-O mappings is given by:

$$C = \log_2 \left[\binom{B+m-1}{m} \right] \quad (6.6)$$

where $B = \binom{k+d-1}{k}$ is the number of distinct branch functions available by drawing k synapses from d afferents with replacement and then drawing m branches from B possible branch functions.

For the multicompartment model, k synapses on a dendrite can connect to n_d compartments in n_d^k ways. Therefore, the number of distinct branch functions can be written as:

$$B_{multi} = B \times n_d^k \quad (6.7)$$

and the total number of I-O mappings for a multicompartment model is given by:

$$C_{multi} = \log_2 \left[\binom{B_{multi} + m - 1}{m} \right] \quad (6.8)$$

6.3 Learning Algorithm

The reduced model for mean rate inputs with lumped dendritic nonlinearity as proposed in Chapter 4 was extended for the classification of spatio-temporal spike patterns, which can be described by Equations (6.1)-(6.5) except two differences arising due to the lumped nature of dendrites in the model:

- (1) Weights are denoted by w_{pq} , where the weight of a dendrite w_p is a d -dimensional vector satisfying the sparsity constraint, i.e. $\sum_q w_{pq} = k$.
- (2) Time delay $t_d = 0$ and hence $T_d^r = 0$.

This spike pattern classifier used a structural learning scheme inspired by the Tempotron [2] rule to learn branch-specific connection patterns. A correlation-based learning rule was employed to learn the groups of input afferents which form connections on a dendrite. The connection changes were done by defining a correlation term c_{pq} for each synapse averaged over all patterns. For binary classification of P random spatio-temporal patterns with target output $y^d = 0$ or 1 , correlation functions $\Delta c_{pq}^+(t)$ and $\Delta c_{pq}^-(t)$ for PDT and NDT for a single

pattern can be written as:

$$\Delta c_{pq}^+(t) = K(t_{max} - t_q^+)b(V_p^+(t_{max}))sign(V_{thr} - V_{max}) \quad (6.9)$$

$$\Delta c_{pq}^-(t) = -K(t_{max} - t_q^-)b(V_p^-(t_{max}))sign(V_{thr} - V_{max}) \quad (6.10)$$

where t_{max} is the time at which $(V_D^+(t) - V_D^-(t)) = V_{max}$; t_q^+ and t_q^- are the spike times at the q^{th} afferent forming connections on the p^{th} dendrite of PDT and NDT respectively and dendritic voltage $V_p^{+/-}(t) = \sum_q w_{pq}^{+/-} \sum_{t_q^{+/-} < t} K(t - t_q^{+/-})$. $sign()$ is the signum function with a value of 1 for $V_{thr} > V_{max}$, -1 for $V_{thr} < V_{max}$ and 0 for $V_{thr} = V_{max}$. Hence, correlations c_{pq}^+ and c_{pq}^- are obtained by averaging $\Delta c_{pq}^+(t)$ and $\Delta c_{pq}^-(t)$ across all P input patterns. Note that the error term $sign(V_{thr} - V_{max})$ only modulates the sign of $\Delta c_{pq}(t)$. We will refer to this learning algorithm as scheme-1 for the purpose of comparison with the schemes proposed in this work. In our present work, the scheme-1 is extended to incorporate the formation of connections on multiple compartments of each dendrite. The two methods adopted for this purpose are explained next.

6.3.1 Scheme-2:

This learning scheme involves learning the connections first using scheme-1 by assuming a lumped dendritic model. The delays associated with these connections formed on each dendrite are then learned by using the following steps:

- (1) All delays are initialized to zero. For the dendritic input voltage:

$$V_p(t) = \sum_q w_{pq} \sum_{t_q < t} K(t - t_q - T_d^q) \quad (6.11)$$

where w_{pq} are the learned weights using scheme-1 and the total delay applied to the synaptic connection from the q^{th} afferent on the p^{th} dendrite, T_d^q is initialized to zero.

- (2) Keeping the weights w_{pq} fixed, delay T_d^q is modified in each learning iteration if the classifier output y (Equation (6.5)) is not equal to the actual output y^d for the input pattern presented during this iteration.

- (3) The error incurred if $y \neq y^d$ is given by $E = V_{thr} - V_{max}$. The delays are modified by performing gradient descent on E with respect to the delay T_d^q :

$$\Delta T_d^q = -\frac{\partial E}{\partial T_d^q} \quad (6.12)$$

- (4) Therefore, delays for the connections on PDT and NDT are modified according to:

$$T_d^{q+} = T_d^{q+} + \eta K'(t_{max} - t_q^+ - T_d^{q+}) b(V_p^+(t_{max})) \text{sign}(V_{thr} - V_{max}) \quad (6.13)$$

$$T_d^{q-} = T_d^{q-} - \eta K'(t_{max} - t_q^- - T_d^{q-}) b(V_p^-(t_{max})) \text{sign}(V_{thr} - V_{max}) \quad (6.14)$$

where $K'()$ is the derivative of kernel $K()$ with respect to T_d^q and η is the learning rate. This delay learning approach is similar to the DELTRON learning algorithm discussed in Chapter 3.

- (5) The delays are modified until either all the patterns are correctly classified or if the error does not decrease after 100 iterations.

6.3.2 Scheme-3:

This method learns both the dendrite and the compartment of the dendrite on which a connection forms. The learning rule is similar to that of scheme-1 except that this new rule involves computing the correlation function c_{pqr} for a synaptic connection (q) formed on specific compartment (r) of a dendrite (p). Hence, the correlations for the PDT and NDT in response to a single pattern:

$$\Delta c_{pqr}^+(t) = K(t_{max} - t_q^+ - T_d^{r+}) b(V_p^+(t_{max})) \text{sign}(V_{thr} - V_{max}) \quad (6.15)$$

$$\Delta c_{pqr}^-(t) = -K(t_{max} - t_q^- - T_d^{r-}) b(V_p^-(t_{max})) \text{sign}(V_{thr} - V_{max}) \quad (6.16)$$

Therefore, correlations are computed by $c_{pqr}^+ = \langle \Delta c_{pqr}^+(t) \rangle$ and $c_{pqr}^- = \langle \Delta c_{pqr}^-(t) \rangle$ where $\langle . \rangle$ indicate average across all P input patterns. The learning process comprises the following

steps:

- (1) The weights w_{pqr} are initialized by forming random connections from the d input lines on the first compartment ($r = 1$) of all dendrites and therefore, delay $T_d^r = 0$.
- (2) In every learning iteration, error is calculated by averaging the absolute error $|y^d - y|$ over the entire batch of P patterns, which is described as mean absolute error (MAE).
- (3) A random set T consisting of n_T ($< s$, where s is the total number of synapses) synapses is selected as candidates for replacement for both PDT and NDT. Synapse with the lowest c_{pqr} in T is eliminated.
- (4) A random replacement set R is created by selecting n_R ($< d$) silent synapses from d input lines. Correlations c_{pqr} are computed by placing these synapses on each compartment ($r = 1, \dots, n_d$) of the dendrite with the lowest c_{pqr} synapse as found in step (3). The lowest c_{pqr} synapse is then replaced by the best-performing (maximum c_{pqr}) synapse in R .
- (5) Synaptic connections are modified if the replacement led to either a reduction or no change in MAE. If the MAE increased with the replacement, a new replacement set R is created. If the MAE does not decrease after repeating this step n_{ch} ($= 20$) times, we assume a local minimum is encountered. We then do a replacement with the last choice of R , even if it increases MAE in an attempt to escape the local minimum. The connection matrix corresponding to the local minimum that gave the least error is saved.
- (6) Steps (2) to (5) are repeated until either all the patterns have been memorized or n_{min} ($= 100$) local minima are encountered.

6.4 Results

The performance of our multicompartment model was evaluated on random latency spatio-temporal patterns, also used in Chapter 3. Since we compare our results with that of the Temptron [2], our model is trained on the stimulus set used in their study. A random latency

Figure 6.2: Random latency spatio-temporal patterns as generated in [2]. P patterns consist of d afferents firing once randomly during stimulus duration T_s .

Table 6.3: Parameters for Multicompartment Model

T_s	τ_f	τ_r	V_0	d	P	s	t_d	$T_d^{n_d}$
400 ms	15 ms	3.75 ms	2.12 mV	500	500	500	2 ms	200 ms

spike pattern consists of d afferents, each randomly firing once during pattern duration T_s (Figure 6.2). P such patterns were generated and assigned equally to two classes denoted by output $y^d \in \{0, 1\}$. All the classifiers using different schemes of learning consist of PDT and NDT with m equal to the total number of dendrites and $s = m \times k$ equal to the total number of synapses for both dendritic trees. The results were obtained by averaging each data point across 3 simulation trials. The parameters used for input generation and the model are given in Table 6.3 unless stated otherwise.

6.4.1 Comparison of lumped vs multicompartment models: Effect of number of dendrites

The three schemes discussed in this work are compared on classification of spatio-temporal patterns. We also looked at the effect of number of dendrites (m) on the performance of different schemes by fixing the value of $s = 500$ for all methods. As shown in Figure 6.3(a), the lumped model of scheme-1 results in the highest errors when comparing the three methods. Moreover, the error decreases as m is increased from 10 to 50 and as m is increased further, the error increases. This can be understood by looking at the measure of theoretical capacity, C for NLD

model as discussed in Section 4.2. For a fixed number of synapses, minimum classification error is obtained when the predicted capacity is high and vice versa. The inverse capacity C^{-1} is also plotted on the right-hand y-axis (dashed blue) of Figure 6.3(a), which shows that as m increases, C^{-1} decreases first (C increases and hence error reduces) and then increases (C decreases with error increasing) as m is increased beyond a certain value. The optimal value of m for the scheme-1 classifier is $m = 50$ for $s = 500$, which is in agreement with the optimal m for C^{-1} .

The performance of scheme-2, which involves learning connections first followed by learning delays associated with these connections is much better than that of scheme-1. The mean error reduces by a factor of about 3. Therefore, once the “best” possible connections for a specific classification problem are formed on each dendrite, introducing time delays serves to shift the classification boundary to further reduce the classification error. This can also be explained using the predicted theoretical capacity C_{multi} (Equation (6.8)) for multicompartment model, which depends on the total number of branch functions, B_{multi} (Equation (6.7)). Since B_{multi} is usually much higher than B (lumped scheme) because of the n_d^k factor, hence it leads to much higher capacity for the multicompartment model. Figure 6.3(a) shows C_{multi}^{-1} plotted on the right-hand y-axis in dashed red. The results also suggest a similar relationship with m as exhibited by scheme-1. The optimal m for scheme-2 is also around $m = 50$.

Finally, for scheme-3 which combines the simultaneous learning of connections and their compartment locations, the errors are much lower than that for scheme-1, but about 1.5 times higher than scheme-2. For scheme-3, multiple delay compartments on each dendrite provide high flexibility which results in higher capacity and hence, lower classification errors than that of scheme-1. Moreover, in each learning step, scheme-2 involves modifying only delays while scheme-3 involves modifying both connections and the delays. The much larger pool of choices for replacement synapse in scheme-3 as compared to scheme-2 presents learnability issues for scheme-3, which can take longer to converge. Hence, the classification errors are higher than that for scheme-2 while the optimal value of m is same in the two cases because the theoretical capacities for both schemes using the same number of delay elements are same.

Figure 6.3: (a) MAE as a function of the number of dendrites (m) for different learning schemes. Inverse capacities are plotted in dashed curves. (b) MAE as a function of the delay per compartment (t_d) for different values of postsynaptic time constant τ_f and $m = 50$. Inverse capacity C_{multi}^{-1} plotted on right-hand y-axis (dashed blue). Each data point is obtained by averaging across 3 simulation trials.

6.4.2 Effect of delay per compartment

Next, the effect of varying the time delay per dendritic compartment (t_d) was studied on the classification of spatio-temporal patterns. We used the learning algorithm discussed under scheme-3 and varied t_d while keeping the total delay along a dendrite $T_d^{n_d}$ constant. We also looked at the effect of varying the postsynaptic time constant, τ_f . The number of delay compartments, n_d , which is given by $n_d = T_d^{n_d}/t_d + 1$, decreases as the delay per compartment increases. As shown in Figure 6.3(b), the classification error increases as t_d increases and then saturates for larger values of t_d or smaller number of compartments.

This behaviour can again be related to the theoretical capacity, C_{multi} for a multicompartment model (C_{multi}^{-1} plotted in dashed blue). From Equations (6.7)-(6.8), we can see that C_{multi} is a function of the term n_d^k . Therefore, as t_d increases, n_d decreases leading to reduced C_{multi} and hence higher classification error. As n_d reduces further, though C_{multi} reduces, but for our task of classifying $P = 500$ patterns, an almost constant accuracy can still be achieved using minimal amount of resources. The results also suggest that a lower postsynaptic time constant, τ_f yields better performance for small values of t_d and higher τ_f works well for larger t_d .

6.4.3 Comparison with the Tempotron rule

The performance of our methods was compared with that of the Tempotron learning rule for the classification of spatio-temporal spike patterns. Table 6.4 shows the comparison between the Tempotron model, which modifies high precision synaptic weights, and the proposed learning schemes inspired by the Tempotron. In contrast with the Tempotron, the Deltron model proposed in Chapter 3 modifies synaptic delays, whereas the learning schemes (1-3) proposed in this chapter modify connectivity matrix of spiking neurons with NLDs and the multicompartment model (schemes-2 and -3) also learn the connection delays. We compare the architecture, weight resolution, delay resolution, classification accuracy and theoretical capacity of these models for the same number of synaptic connections ($s = 500$). It can be seen that the Tempotron model with 7-bit weights performs the best at 100% accuracy but its accuracy drops significantly when the weights are quantized to 3 bits. The Deltron yields lower accuracy than the Tempotron because the delay adaptation can change V_{max} by a limited amount as compared to weight adaptation. The lumped dendritic model (scheme-1) has advantage in terms of achieving higher accuracy using 1-bit (0 or 1) weights. Our multicompartment models can attain even higher accuracy by employing dendritic delays, leading to the best performance of 98.6% using 3-bit delays for scheme-2 as compared to the Tempotron which achieves similar accuracy by using higher (5-bit) precision weights.

Table 6.4 also shows a comparison of the theoretical capacities computed for all the models. As discussed in Section 6.2, the presence of delay compartments in multicompartment dendritic models contributes to higher capacity (Equation (6.8)) than a lumped dendritic model (Equation (6.6)) due to the factor of n_d^k . Moreover, for the same number of synapses s for all models, the capacity of Tempotron model with n_w -bit weights can be computed as $n_w \times s$ bits and for Deltron with n_d -bit delays, the capacity is $n_d \times s$ bits. The capacities computed for the Tempotron in Table 6.4 show that the classification accuracy increases as the theoretical capacity of the model increases. Further, the dendritic model with 1-bit weights (scheme-1) has higher capacity than the Tempotron using 3-bit weights and same number of total synapses, which

shows that the use of NLDs leads to higher capacity. The presence of delay compartments on NLDs contributes further to increased capacity, as demonstrated by the dendritic model using 3-bit delays (scheme-2), which has higher capacity and hence, achieves higher classification accuracy than the Tempotron using 3-bit weights. Finally, the multicompartment NLD model using scheme-3 performs worse than scheme-2, both having similar capacities, which is due to the longer computational time required by scheme-3 to converge (Section 6.4.1).

Table 6.4: Performance Comparison with the Tempotron Model

Model	#Dendrites, #Inputs/Dendrite	Weight (Bits)	Delay (Bits)	Acc. (%)	Capacity (Bits)
Tempotron [2] (7-bit)	No dendrites	7	No delays	100	3500
Tempotron (5-bit)	No dendrites	5	No delays	98.8	2500
Tempotron (3-bit)	No dendrites	3	No delays	92.67	1500
Deltron (Chapter 3)	No dendrites	$w_i = 1 \forall i$	8	89.2	4000
Scheme-1	50, 10	1	No delays	94.07	1683
Scheme-2 ($t_d = 1$ ms)	50, 10	1	3	98.6	2433
Scheme-3 ($t_d = 2$ ms)	50, 10	1	7	97.33	3345

6.5 Summary

This chapter presented a novel learning algorithm for classification of spatio-temporal patterns which is based on learning connection matrix for multicompartment dendritic model. This learning rule finds the optimal morphology in terms of the dendrite and the delay compartment on that dendrite to which input afferents connect. A correlation value is computed for each synapse such that a synapse on a particular compartment which is weakly correlated with its dendrite is eliminated and a strongly correlated synapse on the same or a different compartment of the same dendrite is selected as replacement. Two learning schemes were proposed for the multicompartment model in which: 1) connections and then delays are learned; 2) connections and delays are learned simultaneously. The performance of these learning algorithms was evaluated on random latency spatio-temporal patterns and compared the results with the

performance of a model with lumped dendritic nonlinearity and that of the Tempotron learning rule. The results showed that our multicompartment model with binary synapses can achieve higher performance than the Tempotron model using 4-bit weights.

6.6 Publications

1. **Shaista Hussain** and Arindam Basu, “Morphological Learning in Multicompartment Neuron Model with Binary Synapses,” *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016.
2. Subhrajit Roy, Phyto Phyto San, **Shaista Hussain**, Lee Wang Wei, and Arindam Basu, “Learning Spike Time Codes through Morphological Learning with Binary Synapses,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 7, pp. 1572-1577, 2015.
3. Phyto Phyto San, **Shaista Hussain**, and Arindam Basu, “Spike-Timing Dependent Morphological Learning for a Neuron with Nonlinear Active Dendrites,” *IEEE International Joint Conference on Neural Networks (IJCNN)*, pp. 3192-3196, 2014.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This dissertation presented a novel spike pattern classification model which combined features demonstrating biological relevance and feasibility of hardware implementation. For this work, the focus was on the role of passive and active dendritic mechanisms in neuronal information processing and developed a simple learning rule based on alternatives to weight modification, rendering future hardware mapping of the model much simpler. This work was implemented by firstly presenting a learning algorithm which involves modification of temporal delays to memorize or classify spatio-temporal spike patterns. The training algorithm uses a strategy of updating as few delays as possible to memorize a single pattern where the subset of delays modified correspond to the most synchronous part of the pattern. It was shown to classify patterns with about 80 – 100% accuracy for network load $\alpha \leq 2$. For the tougher memorization task where the entire background population can be considered as a second class, it can memorize about 50 patterns with 90% accuracy of recall. The hardware implementation of delay learning is simpler without multipliers or digital-analog converters (DACs) as compared to that of traditional weight modification algorithms. However, the capacity of this algorithm is lesser than weight modification ones because of finite resolution of the delay modification scheme. This limitation led to the next stage of this dissertation work where enhanced classification performance was achieved by utilizing active mechanisms in dendrites while still maintaining the

hardware-friendly aspect of the learning algorithm.

This was accomplished by developing a spike-based neuron model which employed active dendritic processing, mimicked by a saturating square nonlinearity, and using it for classifying high dimensional binary patterns. The synaptic inputs arriving on each dendritic subunit are nonlinearly processed before being linearly integrated at the soma, giving the neuron a capacity to perform a large number of input-output mappings as compared to the scenario when dendrites act as linear units. This was verified in our work, where the nonlinear dendrites led to about 2 – 5 times lesser classification errors than the linear dendrites. Moreover, the model utilizes sparse synaptic connectivity, where each synapse takes a binary value. The connection pattern with binary weights is learned using a simple hardware-friendly, margin enhancing learning algorithm inspired by the mechanism of activity-dependent structural plasticity in biological neurons. Since the learning results in change in connectivity between input afferents and dendrites, it is well suited for implementation in neuromorphic systems using address-event representation (AER). In this work, two forms of structural learning for the dendritic model were presented; 1) rate-based learning for a reduced model, developed to reduce the training time, and; 2) branch-specific spike-based version of the structural plasticity rule inspired by the spike-timing dependent plasticity (STDP). The proposed model was evaluated on benchmark binary classification problems and its performance was compared against that achieved using Support Vector Machine (SVM) and Extreme Learning Machine (ELM) techniques. Our biologically plausible learning rule attained comparable performance while utilizing 10 – 50% less computational resources than the reported standard machine learning methods.

Consequent to designing the structural learning rule for dendritic neurons, extensions and enhancements were introduced in the basic model. The architecture was extended from a binary classification model to a multiclass classifier of spike patterns. The correspondence between spike-based and rate-based learning schemes was demonstrated and therefore, the faster rate-based learning was adopted for further analysis. An adaptive structural learning rule was proposed which works by progressively adding dendrites to a neuron and forming synaptic connections on these dendrites, thereby allocating synaptic resources as per the complexity of the

given problem. Finally, an ensemble method was developed to attain enhanced generalization by combining several dendritic classifiers. The performance of the multiclass model and the proposed enhancements was evaluated on classification of handwritten digits from the benchmark MNIST dataset and compared with other spike classifiers. The learned dendritic weights were analyzed by visualizing the weight maps which showed that the dendrites extract parts of digits as input features for the discrimination task. The proposed model was shown to achieve classification accuracy within 1 – 2% of other reported spike-based classifiers while using much less synaptic resources, up to 10% of that used by other methods. Further, an ensemble classifier created with adaptively learned classifiers can attain accuracy of 96.2% which is at par with the best reported performance and it achieves this by using still much less number of synapses, about 20% of other spike algorithms.

Finally, in order to extract the advantages of both delay learning and structural learning in a spiking neuron model with dendrites, these two learning paradigms were combined and used for the classification of spatiotemporal patterns. This approach led to a multicompartment dendritic model in which structural plasticity rule is used to form sparse synaptic connections with binary weights on location-dependent delay compartments. This novel structural learning scheme, inspired by the Tempotron learning rule, finds the optimal neuron morphology in terms of the dendrite and the delay compartment on that dendrite where input afferents connect. A correlation value is computed for each synapse such that a synapse on a particular compartment which is weakly correlated with its dendrite is eliminated and a strongly correlated synapse on the same or a different compartment of the same dendrite is selected as replacement. Two learning schemes for the multicompartment model were proposed, in which: 1) connections are learned first, followed by delay modification; 2) connections and delays are learned simultaneously. The performance of our learning algorithms was evaluated on random latency spatiotemporal patterns and compared the results with the performance of a model with lumped dendritic subunits and that of the Tempotron. It was shown that the multicompartment schemes lead to classification errors reducing by up to 5% as compared with the lumped dendrite scheme. Further, the Tempotron achieves a classification accuracy which is about 1% higher than our best

result. However, when Tempotron weights are quantized to 4 bits or less, it performs worse than our methods, all of which use binary weights. Hence, our biologically realistic multicompartment model offers optimized performance and is appealing from the perspective of hardware implementation.

7.2 Future Work

7.2.1 Enhancements in Multicompartment Model

In future, the gap between the classification performance attained by the proposed model and the state-of-the-art methods can be bridged by enhancing the model. For the benchmark MNIST dataset, the best classification result achieved so far is 99.06% accuracy using maxout networks [106]. The accuracy obtained in this thesis can be improved by the application of multicompartment dendrite model with the combined delay and structural learning, as proposed in the Chapter 6. The storage capacity of the network can further be increased by including distributed nonlinearity along a dendrite such that the nonlinear output of each compartment serves as input to the next compartment on the dendrite. This scheme is also more bio-realistic from the perspective of real neurons consisting of extended dendritic trees with complex branching patterns.

This model can be enhanced by developing an adaptive learning scheme to prune the redundant or least “salient” synapses. This pruning method combined with the progressive addition of dendrites will yield an optimally sized network as per the complexity of a given classification problem. The network will learn both the number of required dendrites as well as the number of synapses on each dendrite. This approach to obtain the smallest network can also improve the generalization performance. The multicompartment model can also be enhanced to learn the number of compartments needed on a dendrite. An adaptive approach similar to the one used for learning the number of dendrites can be implemented by first fixing the total delay along a dendrite. The number of compartments are initialized to 1 compartment per dendrite, which is same as the lumped dendritic model. As learning slows down, a compartment is added to the

first dendrite and this process is continued until the maximum number of compartments on a dendrite has been included, after which compartments are added to the second dendrite and so forth.

Our model was used to classify spike patterns of a fixed duration (T_s). However, in real-world applications, the duration of an input pattern might not be known beforehand. Hence, for practical applications of our learning algorithms, parallel dendritic trees processing chunks of input patterns can be implemented. These parallel dendrites connect to a winner take all (WTA) circuit. Therefore, if a dendritic tree responds to a specific chunk, it inhibits the other trees, which learn to respond to other chunks of the input.

7.2.2 Weight Plasticity Combined with Morphological Learning

The proposed algorithm in this thesis can also be extended in future by combining structural learning with biologically observed spike-based weight modification rules like STDP to incorporate precise spike timing information for learning both synaptic connections as well as the weights of these synapses on a dendritic branch. This new learning rule can be incorporated by modifying synaptic weights during every pattern presentation and then consolidating the weights over an epoch to make the connection changes. This logic is also consistent with the experimental findings on different timescales of the two forms of plasticity with weight modification operating in seconds to hours while structural changes taking longer. Further inspiration for this approach comes from the study conducted by Poirazi and Mel in which they pointed out that experimental evidences contradict the conventional view of stable, high-resolution synapses and it can be assumed that long-term memory is encoded by synaptic weights averaged over longer timescales [12] which may exist in a small number of states [62].

7.2.3 Multilayer Architecture

Our model consists of a single layer of neurons which are partitioned into positive and negative components corresponding to the excitatory and inhibitory synapses respectively. This can be extended to a multilayer architecture, which can capture the multiple levels of abstraction in

a complex high-dimensional data. This architecture will allow to learn data representations by utilizing multiple nonlinear transformations. This approach known as “Deep Learning” gained momentum due to the work done by Hinton [107]. It relies on learning hierarchy of features where each level of hierarchy represents the input at a different level of abstraction and more abstract higher-level features can be defined in terms of the lower-level ones. Different amounts of abstraction in a data, which makes it easier to learn a particular task, can be generated by varying the number and size of layers of a deep learning architecture. The gradient-based learning in deep supervised multilayer neural networks gets stuck in local minima and as the network becomes deeper due to addition of more layers, the generalization performance worsens. Hence, deep learning algorithms were designed, which first use the unsupervised pre-training to learn the generally useful representations and then fine-tune the layers using supervised training [108]. Hence, the possibility of developing a multilayer model of neurons with nonlinear dendrites using the approach based on deep learning can be explored.

APPENDIX A

Validity of the Reduced Models

The reduced models (RM and RMWM) developed in Chapter 4 employ time-averaged synaptic activation, $z_{syn,ij}$ instead of the actual postsynaptic current generated by a synapse. This approximation can be used to describe the average current for the j^{th} dendrite as in the following equation:

$$\tilde{I}_{b,out}^j = \left\{ \sum_i w_{ij} \left(\frac{1}{T} \int_0^T \left(\sum_{t_{ij}^s} K(t - t_{ij}^s) \right) dt \right) \right\}^2 / x_{thr} \quad (A.1)$$

where the dendritic nonlinearity is a quadratic function. However, the actual dendritic current is given by the following equation:

$$I_{b,out}^j(t) = \left\{ \sum_i w_{ij} \sum_{t_{ij}^s} K(t - t_{ij}^s) \right\}^2 / x_{thr} \quad (A.2)$$

Therefore, the reduced models are valid if the average dendritic current $\tilde{I}_{b,out}^j$ is equal to the average current computed using the spike-based model, i.e.

$$\begin{aligned} \tilde{I}_{b,out}^j &= \frac{1}{T} \int_0^T I_{b,out}^j(t) dt \\ \Rightarrow \left\{ \sum_i w_{ij} \left(\frac{1}{T} \int_0^T \left(\sum_{t_{ij}^s} K(t - t_{ij}^s) \right) dt \right) \right\}^2 &= \frac{1}{T} \int_0^T \left\{ \sum_i w_{ij} \sum_{t_{ij}^s} K(t - t_{ij}^s) \right\}^2 dt \end{aligned} \quad (A.3)$$

Figure A.1: Average dendritic current predicted using the reduced model (y-axis) plotted against actual mean current (x-axis) for different values of $f_{high} \times \tau_f$ and different number of synapses on a dendrite.

The Equation (A.3) will hold true only if the postsynaptic current $\sum_{t_{ij}^s} K(t - t_{ij}^s)$ is a constant function of time. For the PSC waveform to be described by a constant function, both the frequency of a spike train arriving at a synapse (f_{high}) and synaptic current fall time constant (τ_f in Equation (5.7)) are increased. Figure A.1 shows that as $f_{high} \times \tau_f$ increases, the plot of average dendritic current calculated using the reduced models (y-axis) versus the actual values (x-axis) drifts toward the ‘y=x’ line, which indicates that the values predicted by the reduced model become equal to the actual values. The data points correspond to different number of synapses on a branch. These results have guided the choice of f_{high} and τ_f in this work.

APPENDIX B

Classification of Event-based MNIST-DVS Dataset

The performance of adaptive learning algorithm described in Chapter 5 was evaluated on the actual event-based MNIST dataset consisting of dynamic vision sensor (DVS) recordings of different handwritten digits [109]. This dataset was generated by using 10,000 of 28x28 pixel digit images from the original MNIST dataset which were enlarged to three different scales using smoothing interpolation algorithms. These upscaled digit images were displayed on an LCD monitor with slow motion and a 128x128 pixel AER DVS [110] was used to record these moving digits for a total time duration of 2 s. We used 10,000 recordings of moving digits upscaled to scale-4 for our analysis, which were also used to evaluate the performance of the event-driven categorization system proposed in [111]. The training was performed on randomly selected 90% of the total 10,000 recordings while the remaining 10% recordings were used for testing. For training, the event streams for 128x128 DVS recordings were converted to 128x128 pixel images by calculating the total number of events occurring at each pixel location for two different time durations, 100 ms and full length of 2 s. The 128x128 images were then cropped to digit patches by selecting the location of 28x28 squares with maximum events occurring.

The rate-based adaptive scheme was used for training followed by testing on event streams occurring at the 28x28 patch extracted from the original 128x128 DVS recordings. The training procedure was repeated 3 times and the average testing accuracy over 3 trials was determined as shown in **Table B.1**, which also compares with the performance of the event driven system in [111]. The results show that our adaptive dendritic algorithm can achieve comparable testing

accuracy (88.1% for 2 s) with the other reported performance on MNIST-DVS dataset, with higher accuracy attained when longer recordings are used. Moreover, our method yields lower training accuracy than [111] while the testing accuracy is similar, suggesting that our method is more robust to overfitting.

Table B.1: Performance on MNIST-DVS dataset

Time length of recording used	Training Accuracy (%)	Testing Accuracy (%)
100 ms ([111])	98.9	76.9
2 s ([111])	99.1	88.2
100 ms (This work)	93.7	80.2
2 s (This work)	97.3	88.1

BIBLIOGRAPHY

- [1] A. Polsky, B. Mel, and J. Schiller, “Computational subunits in thin dendrites of pyramidal cells,” *Nature Neuroscience*, vol. 7, no. 6, pp. 621–627, 2004.
- [2] R. Gutig and H. Sompolinsky, “The tempotron: a neuron that learns spike timing-based decisions,” *Nature Neuroscience*, vol. 9, pp. 420–428, 2006.
- [3] W. Rall, *Neural Theory and Modeling*, ch. Theoretical significance of dendritic trees for neuronal input-output relations., pp. 73–97. Stanford University Press, Stanford, CA, 1964.
- [4] B. F. Behabadi, A. Polsky, M. Jadi, J. Schiller, and B. W. Mel, “Location-dependent excitatory synaptic interactions in pyramidal neuron dendrites,” *PLoS Computational Biology*, vol. 8, no. 7, 2012.
- [5] M. Jadi, B. F. Behabadi, A. Poleg-Polsky, J. Schiller, and B. W. Mel, “An augmented two-layer model captures nonlinear analog spatial integration effects in pyramidal neuron dendrites,” *Proceedings of the IEEE (Special issue on Computational Neuroscience)*, vol. 102, no. 5, pp. 782–798, 2014.
- [6] T. Branco, B. A. Clark, and M. Hausser, “Dendritic discrimination of temporal input sequences in cortical neurons,” *Science*, vol. 329, no. 5999, pp. 1671–1675, 2010.
- [7] H. B. Barlow and W. R. Levick, “The mechanism of directionally selective units in rabbit’s retina,” *The Journal of Physiology*, vol. 178, no. 3, pp. 477–504, 1965.
- [8] J. C. Magee, “Dendritic integration of excitatory synaptic input,” *Nature Reviews Neuroscience*, vol. 1, p. 181190, 2000.
- [9] M. London and M. Hausser, “Dendritic computation,” *Annual Review of Neuroscience*, vol. 28, pp. 503–532, 2005.
- [10] M. Hausser, N. Spruston, and G. J. Stuart, “Diversity and Dynamics of Dendritic Signaling,” *Science*, vol. 290, no. 5492, pp. 739–744, 2000.
- [11] T. Branco and M. Hausser, “The single dendritic branch as a fundamental functional unit in the nervous system,” *Current Opinion in Neurobiology*, vol. 20, pp. 494–502, 2010.
- [12] P. Poirazi and B. Mel, “Impact of active dendrites and structural plasticity on the memory capacity of neural tissue,” *Neuron*, vol. 29, pp. 779–96, 2001.

- [13] D. B. Chklovskii, B. W. Mel, and K. Svoboda, "Cortical rewiring and information storage.," *Nature*, vol. 431, no. 7010, pp. 782–788, 2004.
- [14] C. Mead, "Neuromorphic electronic systems," *IEEE Proceedings*, vol. 78, no. 10, pp. 1629–36, 1990.
- [15] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, "Real-time classification and sensor fusion with a spiking deep belief network," *Frontiers in Neuroscience*, vol. 7, pp. 1–13, 2013.
- [16] D. Neil and S.-C. Liu, "Minitaur, an event-driven FPGA-based spiking network accelerator," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. PP:(99) 1, 2014.
- [17] W. Maass and M. Schmitt, "On the complexity of learning for spiking neurons with temporal coding," *Information and Computation*, vol. 153, pp. 26–46, 1999.
- [18] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128x128 120 dB 15us latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–76, 2008.
- [19] S.-C. Liu, A. V. Schaik, B. A. Minch, and T. Delbruck, "Asynchronous binaural spatial audition sensor with 2 x 64 x 4 channel output," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 4, pp. 453 – 464, 2013.
- [20] S. Sheik, M. Pfeiffer, F. Stefanini, and G. Indiveri, "Spatio-temporal Spike Pattern Classification in Neuromorphic Systems," *Living Machines*, vol. 8064, pp. 262–273, 2013.
- [21] M. Schmuker, T. Pfeil, and M. P. Nawrot, "A neuromorphic network for generic multivariate data classification," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 111, no. 6, p. 20812086, 2014.
- [22] E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs, "Event-driven contrastive divergence for spiking neuromorphic systems," *Frontiers in Neuroscience*, vol. 7, no. 272, pp. 1–14, 2014.
- [23] E. Stamatias, D. Neil, M. Pfeiffer, F. Galluppi, S. B. Furber, and S.-C. Liu, "Robustness of spiking Deep Belief Networks to noise and reduced bit precision of neuro-inspired hardware platforms.," *Frontiers in Neuroscience*, vol. 9, no. 222, 2015.
- [24] W. Senn and S. Fusi, "Convergence of stochastic learning in pperceptron with binary synapses.," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 71(6 Pt 1):061907, 2005.
- [25] K. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Transactions on Circuits and Systems II*, vol. 47, no. 5, pp. 416–34, 2000.
- [26] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linarez-Barranco, R. Paz-Vicente, F. Gómez-Rodríguez, L. C. nas Mesa, R. Berner, M. Rivas-Pérez, T. Delbruck, S.-C. Liu, R. Douglas, P. Hafliger, G. Jiménez-Moreno, A. Ballcells, T. Serrano-Gotarredona,

- A. Acosta-Jiménez, and B. Linares-Barranco, “CAVIAR: A 45k neuron, 5M synapse, 12G connects/s AER hardware sensory-processing-learning-actuating system for high-speed visual object recognition and tracking,” *IEEE Transactions on Neural Networks*, vol. 20, no. 9, pp. 1417–38, 2009.
- [27] T. Choi, P. Merolla, J. Arthur, K. Boahen, and B. Shi, “Neuromorphic implementation of orientation hypercolumns,” *IEEE Transactions on Circuits and Systems I*, vol. 52, pp. 1049–60, 2005.
- [28] R. Vogelstein, U. Mallik, E. Culurciello, G. Cauwenberghs, and R. Etienne-Cummings, “A multichip neuromorphic system for spike-based visual information processing,” *Neural Computation*, vol. 19, pp. 1–20, 2007.
- [29] D. F. Specht, “Generation of Polynomial Discriminant Functions for Pattern Recognition,” *IEEE Transactions on Electronic Computers*, vol. EC-16, pp. 308–319, June 1967.
- [30] Y.-H. Pao and Y. Takefuji, “Functional-link net computing: theory, system architecture, and functionalities,” *Computer*, vol. 25, no. 5, pp. 76–79, 1992.
- [31] B. Widrow and M. Lehr, “30 years of adaptive neural networks: Perceptron, madaline and backpropagation,” *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1415–1442, 1990.
- [32] A. G. Ivakhnenko, “Polynomial theory of complex systems,” *IEEE Transactions on Systems Man and Cybernetics*, vol. 1, no. 4, pp. 364–378, 1971.
- [33] B. B. Misra, S. C. Satapathy, B. N. Biswal, P. K. Dash, and G. Panda, “Pattern classification using polynomial neural network,” in *IEEE Conference on Cybernetics and Intelligent Systems*, 2006.
- [34] D. E. Rumelhart, G. E. Hinton, and J. L. McClelland, “A general framework for parallel distributed processing,” in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1* (D. E. Rumelhart and J. L. McClelland, eds.), pp. 45–76, MIT Press Cambridge, MA, USA, 1986.
- [35] C. L. Giles and T. Maxwell, “Learning, invariance, and generalization in high-order neural networks,” *Applied Optics*, vol. 26, no. 23, pp. 4972–4978, 1987.
- [36] C. L. Giles, R. D. Griffin, and T. Maxwell, “Encoding Geometric Invariances in Higher-Order Neural Networks,” in *Proceedings of the Neural Information Processing Systems, American Institute of Physics*, pp. 301–309, 1988.
- [37] R. Durbin and D. E. Rumelhart, “Product units: A computationally powerful and biologically plausible extension to backpropagation networks,” *Neural Computation*, vol. 1, no. 1, pp. 133–142, 1989.
- [38] J. Ghosh and Y. Shin, “Efficient higher-order neural networks for classification and function approximation,” *International Journal of Neural Systems*, vol. 3, no. 4, pp. 323–350, 1992.
- [39] B. W. Mel, “Information processing in dendritic trees,” *Neural Computation*, vol. 6, no. 6, pp. 1031–1085, 1994.

- [40] A. Govindarajan, I. Israely, S.-Y. Huang, and S. Tonegawa, “The dendritic branch is the preferred integrative unit for protein synthesis-dependent LTP,” *Neuron*, vol. 69, no. 1, pp. 132–146, 2011.
- [41] P. Poirazi, T. Brannon, and B. Mel, “Pyramidal neuron as two-layer neural network,” *Neuron*, vol. 37, no. 6, pp. 989–999, 2003.
- [42] P. Poirazi, T. Brannon, and B. W. Mel, “Arithmetic of Subthreshold Synaptic Summation in a Model CA1 Pyramidal Cell,” *Neuron*, vol. 37, pp. 977–987, 2003.
- [43] K. A. Archie and B. W. Mel, “A model for intradendritic computation of binocular disparity,” *Nature Neuroscience*, vol. 3, no. 1, pp. 54–63, 2000.
- [44] B. W. Mel, “The clusteron: Toward a simple abstraction for a complex neuron,” in *Proceedings of the Neural Information Processing Systems*, pp. 35–42, Morgan Kaufman, San Mateo, CA, 1991.
- [45] B. W. Mel, “Synaptic integration in an excitable dendritic tree,” *Journal of Neurophysiology*, vol. 70, no. 3, pp. 1086–1101, 1993.
- [46] J. Schiller, G. Major, H. J. Koester, and Y. Schiller, “NMDA spikes in basal dendrites of cortical pyramidal neurons,” *Nature*, vol. 404, no. 6775, pp. 285–289, 2000.
- [47] B. W. Mel and C. Koch, “Sigma-pi learning: On radial basis functions and cortical associative learning,” in *Proceedings of the Neural Information Processing Systems*, pp. 474–481, Morgan Kaufman, San Mateo, CA, 1989.
- [48] T. J. McBride, A. Rodriguez-Contreras, A. Trinh, R. Bailey, and W. M. DeBello, “Learning drives differential clustering of axodendritic contacts in the barn owl auditory system,” *The Journal of Neuroscience*, vol. 28, no. 27, pp. 6960–6973, 2008.
- [49] M. Fu, X. Yu, J. Lu, and Y. Zuo, “Repetitive motor learning induces coordinated formation of clustered dendritic spines in vivo,” *Nature*, vol. 483, pp. 92–95, 2012.
- [50] N. Takahashi, K. Kitamura, N. Matsuo, M. Mayford, M. Kano, N. Matsuki, and Y. Ikegaya, “Locally synchronized synaptic inputs,” *Science*, vol. 335, no. 6066, pp. 353–356, 2012.
- [51] M. E. Larkum and T. Nevian, “Synaptic clustering by dendritic signalling mechanisms,” *Current Opinion in Neurobiology*, vol. 18, pp. 321–331, 2008.
- [52] T. Kleindienst, J. Winnubst, C. Roth-Alpermann, T. Bonhoeffer, and C. Lohmann, “Activity-dependent clustering of functional synaptic inputs on developing hippocampal dendrites,” *Neuron*, vol. 72, pp. 1012–1024, 2011.
- [53] J. K. Makara, A. Losonczy, Q. Wen, and J. C. Magee, “Experience-dependent compartmentalized dendritic plasticity in rat hippocampal CA1 pyramidal neurons,” *Nature Neuroscience*, vol. 12, no. 12, pp. 1485–1487, 2009.

- [54] R. Legenstein and W. Maass, “Branch-specific plasticity enables self-organization of nonlinear computation in single neurons,” *The Journal of Neuroscience*, vol. 31, no. 20, pp. 10787–10802, 2011.
- [55] M. Butz, F. Worgotter, and A. V. Ooyen, “Activity-dependent structural plasticity,” *Brain Research Reviews*, vol. 60, no. 2, pp. 287–305, 2009.
- [56] J. T. Trachtenberg, B. E. Chen, G. W. Knott, G. Feng, J. R. Sanes, E. Welker, and K. Svoboda, “Long-term in vivo imaging of experience-dependent synaptic plasticity in adult cortex,” *Nature*, vol. 420, pp. 788–794, 2002.
- [57] G. W. Knott, A. Holtmaat, L. Wilbrecht, E. Welker, and K. Svoboda, “Spine growth precedes synapse formation in the adult neocortex in vivo,” *Nature Neuroscience*, vol. 9, no. 9, pp. 1117–1124, 2006.
- [58] J. V. L. Be and H. Markram, “Spontaneous and evoked synaptic rewiring in the neonatal neocortex,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 35, pp. 13214–13219, 2006.
- [59] C. Darian-Smith and C. D. Gilbert, “Axonal sprouting accompanies functional reorganization in adult cat striate cortex,” *Nature*, vol. 368, no. 6473, pp. 737–740, 1994.
- [60] D. D. Stettler, H. Yamahachi, W. Li, W. Denk, and C. D. Gilbert, “Axons and synaptic boutons are highly dynamic in adult visual cortex,” *Neuron*, vol. 49, no. 6, pp. 877–887, 2006.
- [61] A. Stepanyants, P. R. Hof, and D. B. Chklovski, “Geometry and structural plasticity of synaptic connectivity,” *Neuron*, vol. 34, no. 2, pp. 275–288, 2002.
- [62] C. C. Petersen, R. C. Malenka, R. A. Nicoll, and J. J. Hopfield, “All-or-none potentiation at CA3-CA1 synapses,” *Proc. Natl. Acad. Sci. USA*, vol. 95, no. 8, pp. 4732–4737, 1998.
- [63] D. H. O’Connor, G. M. Wittenberg, and S. S.-H. Wang, “Graded bidirectional synaptic plasticity is composed of switch-like unitary events,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 27, pp. 9679–9684, 2005.
- [64] J. Brader, W. Senn, and S. Fusi, “Learning real-world stimuli in a neural network with spike-driven synaptic dynamics,” *Neural Computation*, vol. 19, pp. 2881–2912, 2007.
- [65] K. Ramanathan, N. Ning, D. Dhanasekar, G. Li, L. Shi, and P. Vadakkepat, “Presynaptic Learning and Memory with a Persistent Firing Neuron and a Habituating Synapse: a Model of Short Term Persistent Habituation,” *International Journal of Neural Systems*, vol. 22, no. 4, p. 1250015, 2012.
- [66] J. Rossell, V. Canals, A. Morro, and A. Oliver, “Hardware implementation of stochastic spiking neural networks,” *International Journal of Neural Systems*, vol. 22, p. 1250014, Aug 2012.

- [67] W. Wong, Z. Wang, B. Zhen, and S. Leung, "Relationship between applicability of current-based synapses and uniformity of firing patterns," *International Journal of Neural Systems*, vol. 22, p. 1250017, Aug 2012.
- [68] E. Izhikevich, "Polychronization: Computation With Spikes," *Neural Computation*, vol. 18, pp. 245–282, 2006.
- [69] H. Agmon-Snir and I. Segev, "Signal Delay and Input Synchronization in Passive Dendritic Structures," *Journal of Neurophysiology*, vol. 70, no. 5, pp. 2066–2085, 1993.
- [70] R. Wang, C. Jin, A. McEwan, and A. Schaik, "A programmable axonal propagation delay circuit for time-delay spiking neural networks," *Proc. IEEE ISCAS*, pp. 869–872, 2011.
- [71] R. Wang, J. Tapson, T. Hamilton, and A. V. Schaik, "An aVLSI programmable axonal delay circuit with spike timing dependent delay adaptation," in *Proceedings of the International Symposium on Circuits and Systems*, (Korea), May 2012.
- [72] P. Baldi and A. Atiya, "How delays affect neural dynamics and learning," *IEEE Transactions on Neural Networks*, vol. 5, pp. 612–621, 1994.
- [73] U. Bodenhausen and A. Waibel, "The tempo 2 algorithm: Adjusting time-delays by supervised learning," *Advances in Neural Information Processing Systems 3*, R.P. Lippman, J.E. Moody, and D.S. Touretzky Eds., pp. 155–161, 1991.
- [74] D. Tank and J. Hopfield, "Neural computation by concentrating information in time," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 84, pp. 1896–1900, 1987.
- [75] K. Unnikrishnan, J. Hopfield, and D. Tank, "Connected-digit speaker-dependent speech recognition using a neural network with time-delayed connections," *IEEE Transactions on Signal Processing*, vol. 39, no. 3, pp. 698–713, 1991.
- [76] R. Eckmiller and H. Napp-Zinn, "Information processing in biology-inspired pulse coded neural networks," in *Proceedings of International Joint Conference on Neural Networks*, vol. 1, pp. 643–8, 1993.
- [77] H. Napp-Zinn, M. Jansen, and R. Eckmiller, "Recognition and tracking of impulse patterns with delay adaptation in biology-inspired pulse processing neural net (BPN) hardware," *Biological Cybernetics*, vol. 74, pp. 449–453, May 1996.
- [78] H. Huning, H. Glunder, and G. Palm, "Synaptic Delay Learning in Pulse-Coupled Neurons," *Neural Computation*, vol. 10, pp. 555–565, 1998.
- [79] P. Hasler, S. Koziol, E. Farquhar, and A. Basu, "Transistor channel dendrites implementing HMM classifiers," in *Proceedings of the International Symposium on Circuits and Systems*, pp. 3359–62, May 2007.
- [80] W. Gerstner and W. Kistler, *Spiking Neuron Models: Single neurons, Populations, Plasticity*. Cambridge University Press, 2002.

- [81] X. Guo, X. Qi, and J. Harris, "A Time-to-First-Spike CMOS Image Sensor," *IEEE Sensors Journal*, vol. 7, no. 8, pp. 1165–1175, 2007.
- [82] C. Shoushun and A. Bermak, "Arbitrated Time-to-First Spike CMOS Image Sensor With On-Chip Histogram Equalization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 3, pp. 346–357, 2007.
- [83] X. Qi, X. Guo, and J. Harris, "A time-to-first spike CMOS imager," in *Proceedings of the International Symposium on Circuits and Systems*, pp. 824–7, May 2004.
- [84] J. Lazzaro, S. Ruckebusch, M. Mahowald, and C. Mead, "Winner-take-all networks of $O(N)$ complexity," in *Proceedings of the Neural Information Processing Systems*, pp. 703–11, The MIT Press, 1988.
- [85] A. L. Yuille and D. Geiger, "Winner-take-all mechanisms," in *The Handbook of Brain Theory and Neural Networks* (M. Arbib, ed.), pp. 1056–1060, MIT Press, 1995.
- [86] S.-C. Liu, G. Indiveri, T. Delbruck, and R. Douglas, *Analog VLSI: Circuits and Principles*. Bradford Book, MIT Press, ISBN No. 0-262-12255-3, 2002.
- [87] A. Basu, S. Brink, C. Schlottmann, S. Ramakrishnan, C. Petre, S. Koziol, F. Baskaya, C. Twigg, and P. Hasler, "A floating-gate based field programmable analog array," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 9, pp. 1781–94, 2010.
- [88] S. Haykin, *Neural Networks: A Comprehensive Foundation*. MacMillan Publishing Company, 1994.
- [89] J. Letzkus, B. Kampa, and G. Stuart, "Learning rules for spike timing-dependent plasticity depend on dendritic synapse location," *The Journal of Neuroscience*, vol. 26, no. 41, pp. 10420–9, 2006.
- [90] R. Froemke, M. Poo, and Y. Dan, "Spike-timing-dependent synaptic plasticity depends on dendritic location.," *Nature*, vol. 434, pp. 221–225, 2005.
- [91] P. Sjöström and M. Häusser, "A cooperative switch determines the sign of synaptic plasticity in distal dendrites of neocortical pyramidal neurons.," *Neuron*, vol. 51, pp. 227–238, 2006.
- [92] C. Albers, M. Westkott, and K. Pawelzik, "Perfect associative learning with spike-timing-dependent plasticity," in *Advances in Neural Information Processing Systems 26* (C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, eds.), pp. 1709–1717, Curran Associates, Inc., 2013.
- [93] P. D'Souza, S.-C. Liu, and R. H. Hahnloser, "Perceptron learning rule derived from spike-frequency adaptation and spike-time-dependent plasticity," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 107, no. 10, pp. 4722–4727, 2010.
- [94] A. Losonczy, J. Makara, and J. C. Magee, "Compartmentalized dendritic plasticity and input feature storage in neurons," *Nature*, vol. 452, pp. 436–441, 2008.

- [95] M. Helias, S. Rotter, M. Gewaltig, and M. Diesmann, "Structural plasticity controlled by calcium based correlation detection," *Frontiers in Computational Neuroscience*, vol. 2, pp. 1–21, December 2008.
- [96] S. Mitra, S. Fusi, and G. Indiveri, "Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI," *IEEE Transactions on Biomedical Engineering*, vol. 3, no. 1, pp. 32–42, 2009.
- [97] K. Bache and M. Lichman, "UCI Machine Learning Repository." Internet : <http://archive.ics.uci.edu/ml/>, 2013.
- [98] G. Babu and S. Suresh, "Sequential projection-based metacognitive learning in a radial basis function network for classification problems.," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 2, pp. 194–206, 2013.
- [99] P. A. Merolla, J. V. Arthur, R. A.-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, pp. 668–673, August 2014.
- [100] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169–198, 1999.
- [101] Y. L. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of IEEE*, vol. 86, pp. 2278–2324, Nov. 1998.
- [102] S. Chen and A. Bermak, "Arbitrated Time-To-First Spike CMOS Image Sensor Array with On-Chip Histogram Equalization," *IEEE Transactions on VLSI*, vol. 15, pp. 346–57, Mar 2007.
- [103] W. Gerstner and W. M. Kistler, *Spiking Neuron Models Single Neurons, Populations, Plasticity*. Cambridge University Press, Aug 2002.
- [104] V. Chan, S.-C. Liu, and A. van Schaik, "AER EAR: A matched silicon cochlea pair with address event representation interface," *IEEE Transactions on Circuits and Systems I*, vol. 54, no. 1, pp. 48–59, 2007.
- [105] D. Marti, M. Rigotti, M. Seok, and S. Fusi, "Energy-efficient neuromorphic classifiers," *arXiv5:1507.0023*, 2015.
- [106] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proceedings of the 30th International Conference on Machine Learning*, pp. 1319–1327, 2013.
- [107] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
- [108] Y. Bengio, "Learning Deep Architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

- [109] T. Serrano-Gotarredona and B. Linares-Barranco, “The mnist-dvs database. [online].,” 2014.
- [110] T. Serrano-Gotarredona and B. Linares-Barranco, “A 128x128 1.5 % contrast sensitivity 0.9 % FPN 3ms latency 4mW asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers,” *IEEE Journal of Solid-State Circuits*, vol. 48, no. 3, pp. 827–838, 2013.
- [111] B. Zhao, R. Ding, S. Chen, B. Linares-Barranco, and H. Tang, “Feedforward categorization on aer motion events using cortex-like features in a spiking neural network,” *IEEE Transactions on Neural Networks and Learning Systems*, 2015.