Learning spike time codes through supervised and unsupervised structural plasticity

Roy, Subhrajit

2016

Roy, S. (2016). Learning spike time codes through supervised and unsupervised structural plasticity. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/67327>

<https://doi.org/10.32657/10356/67327>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**Learning spike time codes through supervised and unsupervised
structural plasticity**

by
Subhrajit Roy

A Thesis
Submitted to the Faculty
of the
NANYANG TECHNOLOGICAL UNIVERSITY
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in
Electrical and Electronic Engineering
2016

Acknowledgment

First and foremost, I want to thank my advisor Dr. Arindam Basu for teaching, guiding and advising me during my time spent in NTU. He is an extremely nice guy with a friendly personality which allowed us to share a great advisor-advisee relationship. I appreciate all his contributions of time, ideas and funding to make my PhD experience more captivating. The joy and enthusiasm he has for his research was contagious and motivational for me. But most importantly, he has taught me, both consciously and unconsciously, how to become a good academic.

Moreover, I would like to express my gratitude to my supervisors and colleagues: Shaista Hussain, Amitava Banerjee, Roshan Gopalakrishnan, Sougata Kar, Enyi Yao, Chen Yi and Aritra Bhaduri of Nanyang Technological University and Lee Wang Wei and Dr. Nitish Thakor of Singapore Institute for Neurotechnology.

I would also like to thank my friends and colleagues for all the achievements and failures we shared together. This experience will never be forgotten. I also thank the administrative and technical staff members of VIRTUS, IC Design Centre of Excellence who have been kind enough to advise and help in their respective roles. Special gratitude to my friends Soumyabrata Dev and Raunaq Pradhan, who started their doctoral journey along with me and have been a constant source of encouragement.

I am eternally grateful to my parents Somnath Roy and Supra Roy, my dear wife Mohar Bandyopadhyay and my brother Soumyajit Roy for their

support throughout the past few years. Their kind understanding and faith in me are what made all these possible.

Finally, I want to acknowledge the financial support from Ministry of Education and Nanyang Technological University.

Abstract

Large-scale spiking neural networks (SNN) are typically implemented on the chip by using mixed analog-digital circuits. While the models of the network components (neurons, synapses and dendrites) are implemented by analog VLSI techniques, the connectivity information of the network is stored in an on-chip digital memory. Since the connectivity information is virtual, the user has full flexibility in reconfiguring the network. When a new task is encountered, a software model of the network is trained in computer and the trained weights are downloaded to the digital memory. Hence, the analog part works in conjunction with the digital part to form a properly weighted SNN suitable for the task at hand. However, very few hardware systems emulating SNN have been reported to solve real-world pattern recognition tasks with performance comparable to their software counterparts. A major challenge in obtaining a classification accuracy comparable to a software implementation of a system on a computer is that statistical variations in VLSI devices diminish the accuracy of synaptic weights. Most of the current neuromorphic systems require high-resolution synaptic weights [1–3] and hence are affected by this problem. In this thesis, for enhanced stability to mismatch and efficient hardware implementation we have considered neurons with binary synapses for recognition of spatiotemporal spike trains. To compensate for the reduced computational power provided by binary synapses we look into more bio-physical models of neurons. Inspired by the nonlinear properties of dendrites in biological neurons, the proposed networks incor-

porate neurons having multiple dendrites with a lumped nonlinearity (two compartment model). We have shown that such a neuron with nonlinear dendrites (NNLD) has higher memory capacity than their linear counterpart and networks employing them provide state-of-the-art performance. Since binary synapses are considered, learning happens through formation and elimination of connections between the inputs and the dendritic branches to modify the structure or morphology of the network. Hence, the learning involves network rewiring (NRW) of the spiking neural network similar to structural plasticity observed in its biological counterparts.

However, for structural plasticity based learning rules high dimensional sparse inputs are required. A popular way to map any input pattern into a high dimensional space enforcing sparse coding is by using non-overlapping binary valued receptive fields. But this method, though suitable for rate coded inputs, cannot be applied to spike-coded inputs. To overcome this issue, taking inspiration from Liquid State Machine (LSM), a popular model for reservoir computing, we have used a spiking neural network reservoir to convert the spike train inputs to a high dimensional sparse spatiotemporal spike encoding. Subsequently, we have used this high dimensional spike data to train our NNLD network having binary synapses through the proposed learning rule. We have shown that compared to a single perceptron using analog weights, this architecture for the readout can attain, even by using the same number of binary valued synapses, up to 3.3 times less error for a two-class spike train classification problem and 2.4 times less error for an input rate approximation task. Even with 60 times larger synapses, a group of 60 parallel perceptrons cannot attain the performance of the proposed dendritically enhanced architecture. Furthermore, we have shown that due to the use of binary synapses, our proposed method is more robust against statistical variations. We have also looked into on-chip implementation of NNLD and NRW learning rule for online training of the proposed readout. This is the first contribution of this thesis.

The second contribution arises from increasing memory capacity of NNLD based networks in recognizing high dimensional spike trains. A morphological learning algorithm inspired by the ‘Tempotron’, i.e., a recently proposed temporal learning algorithm is presented in this thesis. Unlike ‘Tempotron’, the proposed learning rule uses a technique to automatically adapt the neuronal firing threshold during training. Experimental results indicate that our NNLD with binary or 1-bit synapses can obtain similar accuracy as a traditional Tempotron with 4-bit synapses in classifying single spike random latency and pair-wise synchrony patterns. We have also presented results of applying this rule to real life spike classification problems from the field of tactile sensing.

The two earlier contributions are supervised learning rules for training NNLD with binary synapses. The third contribution of this thesis is to provide an unsupervised learning rule for training NNLD. We have proposed a novel Winner-Take-All (WTA) architecture employing an array of NNLDs with binary synapses and an online unsupervised structural plasticity rule for training it. The proposed unsupervised learning rule is inspired by spike timing dependent plasticity (STDP) but differs for each dendrite based on its activation level. It trains the WTA network through formation and elimination of connections between inputs and synapses. To demonstrate the performance of the proposed network and learning rule, we have employed it to solve two, four and six class classification of random Poisson spike time inputs. The results indicate that by proper tuning of the inhibitory time constant of the WTA, a trade-off between specificity and sensitivity of the network can be achieved. We use the inhibitory time constant to set the number of subpatterns per pattern we want to detect. We show that while the percentage of successful trials are 92%, 88% and 82% for two, four and six class classification when no pattern subdivisions are made, it increases to 100% when each pattern is subdivided into 5 or 10 subpatterns. However, the former scenario of no pattern subdivision is more jitter resilient than the

later ones.

Apart from bio-realism and performance, an additional advantage of this method for hardware implementations is that the ‘choice’ of connectivity can be easily implemented exploiting address event representation (AER) protocols commonly used in current neuromorphic systems where the connection matrix is stored in memory. Also, due to the use of binary synapses, our proposed algorithms are less affected by VLSI mismatch.

The algorithms proposed in this thesis can find direct application in classifying spatio-temporal spike patterns arriving from the domain of Brain Machine Interfaces (BMI), Tactile sensors, sensory prosthesis, etc. However, the main contribution of this thesis is that it tends to remove the long-standing bias towards using neural networks with high-resolution weights and weight-update based learning rules. This thesis shall trigger the usage of neural networks with binary synapses and connection-based learning rules for solving pattern recognition tasks in hardware. While traditional networks had to incorporate sparsity by using lesser number of neurons, our learning rules inherently form sparse networks by making sparse connections between inputs and dendrites.

Contents

Acknowledgment	i
Abstract	iii
Contents	vii
List of Figures	xii
List of Tables	xxii
Abbreviations	xxv
1 Introduction	1
1.1 Background and Motivation	1
1.2 Thesis Contribution	7
1.3 Thesis Organization	9
2 Literature Review	11
2.1 Introduction to Spiking Neural Networks	11
2.1.1 Models of computational units	12
2.1.2 Architectures of Spiking Neural Networks	20
2.1.3 Learning in Spiking Neural Networks	21
2.1.4 Spiking Neural Networks in hardware	27
2.2 Introduction to Neurons with nonlinear dendrites	30

2.2.1	The point neuron hypothesis: Structure and Challenges	30
2.2.2	Dendritic branches as key computational elements . . .	31
2.2.3	The two-layer neuron model (2LM)	33
2.2.4	The simplified 2LM using low resolution synapses . . .	37
2.3	Summary	40
3	Liquid State Machine with Dendritically Enhanced Readout	41
3.1	Introduction	42
3.2	Background	45
3.2.1	Theory of Liquid State Machine	46
3.2.2	Previous research on Liquid State Machine	47
3.2.3	Parallel Perceptron Readout and the p -delta learning algorithm	50
3.2.4	Model of Nonlinear Dendrites	54
3.3	LSM-DER architecture and NRW learning rule	55
3.3.1	Spike trains as input to Neurons with active dendrites .	59
3.4	Experiments and Results	61
3.4.1	Problem Description	61
3.4.2	Choice of Parameters	64
3.4.3	Results: Performance of LSM-DER and NRW algorithm	68
3.4.4	Results: Comparison between LSM-DER and LSM-PPR	72
3.5	Discussion of software results	77
3.6	VLSI Implementation for off-chip learning and on-chip testing: Effect of Statistical Variations	80
3.6.1	VLSI Architecture of the Spiking Neural Network . . .	80
3.6.2	Monte Carlo Simulation Results	83
3.6.3	Stability Analysis in Software Simulations	85
3.7	Architectural Exploration and learning rule modification for on-chip online learning	86
3.7.1	Architectural modification of DER for on-chip learning	87
3.7.2	Online NRW learning rule	90

3.8	Summary	93
4	Morphology Optimizing Tempotron	95
4.1	Introduction	96
4.2	Background	97
4.2.1	Theory of Tempotron	97
4.2.2	Previous research on Tempotron	99
4.3	Proposed classifier architecture and MOT learning rule	102
4.3.1	Classifier architecture	102
4.3.2	MOT learning rule	104
4.4	Experiments and Results	108
4.4.1	Problem Description	108
4.4.2	Results: Performance of NNLD trained by MOT	109
4.4.3	Results: Comparison with Tempotron learning	110
4.5	Application example: Classifying Tactile sensing data	112
4.5.1	Task Description	112
4.5.2	Spike Train Generation	113
4.5.3	Results	113
4.6	Discussion	114
4.6.1	Comparison to other supervised spiking neural classifiers	114
4.6.2	Comparison to other classifiers using dendritic processing	115
4.6.3	Hardware implementation of Morphology Optimizing Tempotron	117
4.7	Summary	117
5	An Online Unsupervised Structural Plasticity algorithm for neurons with nonlinear dendrites	119
5.1	Introduction	120
5.2	Previous research on application of STDP in WTA networks .	122
5.3	Proposed architecture and learning rule	124

5.3.1	Winner-Take-All employing Neurons with Nonlinear Dendrites (WTA-NNLD)	124
5.3.2	Spike Timing Dependent Plasticity based Network Rewiring (STDP-NRW) learning rule	127
5.3.3	Specificity and Sensitivity: Role of Inhibitory Time Constant	129
5.4	Choice of Parameters	133
5.5	Experiments and Results	137
5.5.1	Problem Description	138
5.5.2	Case 1: $n_{sub} = 1$	139
5.5.3	Case 2: $n_{sub} > 1$	141
5.6	VLSI Implementation: Effect of statistical variation	145
5.7	Hardware implementation of fitness update	146
5.8	Summary	147
6	Conclusion and Future Work	150
6.1	Conclusion	150
6.2	Future Work	155
6.3	Algorithmic Developments	155
6.3.1	Power efficient Deep Spiking Neural Networks	156
6.3.2	Unsupervised adaptation of reservoir	158
6.3.3	Application of DER and MOT as readouts	160
6.3.4	LSM-DER and MOT as multi-class classifiers	160
6.3.5	Two layer backpropagating network of NNLDs	161
6.4	On-chip implementation of the proposed frameworks	162
6.5	Real world applications	162
A	Estimating capacity of NNLD	164
B	NNLDs as universal approximator of Boolean Functions	167

C	Author's Publications	170
C.1	Journal Papers	170
C.2	Conference Papers	171

List of Figures

1.1	(a) The basic diagram of a neuron (b) The profile of an action potential generated by a neuron. [Source: Wikipedia]	2
2.1	Equivalent circuit diagram of the Hodgkin Huxley model. The model contains two active Na and K channels and one passive leakage channel.	13
2.2	Equivalent circuit diagram of the Leaky Integrate and Fire neuron. A current $I_{in}(t)$ charges the RC circuit. When $V(t)$ becomes larger than the neuron spiking threshold voltage θ , an output pulse is emitted.	15
2.3	This figure shows the effect of pre-synaptic spikes and externally injected input current on the membrane voltage ($V(t)$) and firing threshold ($\theta(t)$) of a SRM neuron. The solid and dotted lines represent the evolution of $V(t)$ and $\theta(t)$ respectively.	16
2.4	The output current $I_{syn}(t)$ of a static and a dynamic synapse which receive the same input spikes are compared against each other. The $I_{syn}(t)$ for both the synapses are computed by summing the exponential current kernels produced for every spike. However, while the amplitude of the kernels remain same for the static synapse, it reduces with a recovery time constant for the dynamic synapse.	18

2.5	The LSM consists of an input layer, a hugely interconnected spiking neural network termed as the liquid followed by a linear classifier stage termed as the readout. The liquid acts as a nonlinear filter by projecting the input to a high dimensional vector of spike trains which are combined at the readout. The network is trained by only tuning the weights of the connections entering the readout.	22
2.6	The multi-layered structure of a Synfire Chain. Volleys of spikes get propagated along the layers. Although there are no recurrent interconnections among the layers, feedback connections may exist within the layers.	22
2.7	An experiment performed by Polsky et al. [4] demonstrating that dendrites of pyramidal neurons are associated with saturating nonlinearities.	34
2.8	The 2LM showing a neuron with with nonlinear dendrites. . .	35
2.9	A neuronal cell with active dendrites as proposed by Mel and Poirazi [5]	38
3.1	The first stage of LSM [6] is the input layer. The input stage is followed by a pool of recurrent LIF neurons whose synaptic connections are not trained. The next stage is a simple linear classifier that is selected and trained in a task-specific manner.	43
3.2	In LSM-PPR, the readout stage is composed of a single layer of perceptrons which do not have any lateral connections. The output of the liquid is connected to each perceptron of this parallel perceptron stage.	51
3.3	A neuron with nonlinear dendrites (NNLD)	55
3.4	In LSM-DER, the readout stage following the liquid is composed of two NNLDs. The final readout output $y()$ is obtained by taking the difference of the output of the two cells and passing it through the function $g()$	56

3.5	An example of input generation for the proposed readout where the spike train $s(t)$ is first convolved with a kernel representing post-synaptic current to produce $s_a(t)$ and then sampled at a desired temporal resolution to give $s_a^f(t)$	60
3.6	Task I is the classification of spike trains. Two different classes of spike trains and the corresponding liquid output when these spike trains are projected into it are shown. Also, a jittered version of a spike train belonging to Class 1 that will be used for testing is depicted.	62
3.7	Task II involves the approximation of a desired function. $r(t)$ is a signal that modulates the 4 input Poisson spike trains. The modulated spike trains are given as input to the liquid and the liquid output is shown. The sum of rates corresponding to the modulated spike trains is also shown	65
3.8	The value of B_N is plotted for all possible values of m	66
3.9	The training error (MAE) obtained by LSM-DER in each iteration is plotted for (a) Task I and (b) Task II when trained on a set of 200 patterns averaged over 10 trials.	69
3.10	The output produced by the LSM-DER after training is superimposed on the target function for a randomly chosen time window and show a very close match.	69

3.11	Keeping the number of synapses per dendritic branch (k) of the NNLDs constant at 10, the average MAE over 10 trials is plotted for (a) Task I and (c) Task II when the number of dendrites (m) is varied. Thus, the total number of synapses connecting the reservoir with the readout increases with m and results in a reduction of error. When the total number of synapses for each NNLD ($s/2$) is kept constant at 70, MAE is plotted for (b) Task I and (d) Task II when number of dendrites are increased. In this case, the error initially reduces, reaches a minima and then starts increasing.	70
3.12	Change in MAE, averaged over 10 trials, when x_{sat} is varied for (a) task I and (b) task II. The dashed lines indicate the MAE for $x_{sat} \rightarrow \infty$	72
3.13	Comparison between MAE vs iterations curve of LSM-DER and LSM-PPR for 200 input patterns averaged over 10 trials for Task I.	73
3.14	Comparison between MAE vs iterations curve of LSM-DER and LSM-PPR for 200 input patterns averaged over 10 trials for Task II.	73
3.15	Convergence Analysis of LSM-DER and LSM-PPR for Task I: The average value of n_0 , n_1 and n_2 obtained for 50, 100 and 200 input patterns averaged over 10 trials.	74
3.16	Generalization Plot: A performance comparison of the training and testing error of LSM-DER and LSM-PPR on Task I for 50, 100 and 200 input patterns averaged over 10 trials	75
3.17	Generalization Plot: A performance comparison of the training and testing error of LSM-DER and LSM-PPR on Task II for 50, 100 and 200 input patterns averaged over 10 trials	76

3.18	Performance Comparison of LSM-DER and LSM-PPR with varying n for Task I (MAE averaged over 10 trials): The number of readout neurons i.e. n is a parameter of LSM-PP. We have obtained the results of LSM-PPR for different values of n . The classification error gradually decreases with increasing n and finally it saturates. The performance of LSM-PPR for different values of n are compared with the results obtained by LSM-DER.	76
3.19	Performance Comparison of LSM-DER and LSM-PPR with varying n for Task II (MAE averaged over 10 trials): The number of readout neurons i.e. n is a parameter of LSM-PP. We have obtained the results of LSM-PPR for different values of n . The error gradually decreases with increasing n and finally it saturates. The performance of LSM-PP for different values of n are compared with the results obtained by LSM-DER.	77
3.20	Histogram showing the distribution of weights of the synapses in the readout for LSM-PPR. It can be noted that most of the synapses have a non-zero weight implying that most of the synapses are active and add to resource consumption.	78
3.21	(a) MAE for Task I is reported by varying τ_s and inter spike interval of the liquid output N . (b) A curve fit to the optimal τ_s for each N is obtained by varying N for both tasks over two orders of magnitude.	80
3.22	Architecture for VLSI implementation of (a) DER and (b) PPR readout schemes for LSM. Schematic diagram of DPI Synapse and squaring block used in these architectures are shown in detail in (c) and (d) respectively.	81

3.23	Stability of LSM-DER and LSM-PPR with respect to different hardware non-idealities are plotted for Task I in (a) and (b) respectively. The constant red line indicates the testing error obtained by the algorithms without any non-ideality. The bars represent the error obtained after inclusion of non-idealities. The rightmost bar marked by (...) represents the error when all the non-idealities are included simultaneously. The MAE is averaged over 10 trials.	84
3.24	Block diagram of Architecture II	88
3.25	Block diagram of Architecture III	89
3.26	Block diagram of Architecture IV	89
3.27	Time line depicting the online NRW learning rule	91
3.28	Setting $P_{MAE} = \frac{P}{4}$, the convergence curves have been shown for different values of P_{conn}	93
4.1	A LIF neuron to be trained by Tempotron learning rule is shown.	98
4.2	The architecture of neuron with nonlinear dendrites (NNLD).	103
4.3	Distribution of V_{max} for 100 patterns (a) before and (b) after training for Task I. V_{max} for patterns in P^+ and P^- are shown in dashed and solid line respectively.	109
4.4	The distribution of V_{max} for a set of (a) 100, (b) 500 and (c) 1000 patterns in task I showing 100%, 94.8% and 84.9% accuracy respectively.	110
4.5	Comparison studies on classification performance, for Tempotron learning and the proposed morphological learning rule for (a) Task I of Random Latency Patterns and (b) Task II of Pair-Wise Synchrony Patterns. The results have been averaged over 10 independent trials.	111
4.6	This figure shows the block diagram for implementing the threshold adaptation technique of MOT.	118

5.1	A spike-based WTA network employing neurons with lumped dendritic nonlinearities as the competing entities. For implementing lateral inhibition, an inhibitory neuron has been included which, upon activation, provides a global inhibition signal to all the NNLD.	125
5.2	An example of the update rule of fitness value ($c_{pj}^n(t)$) is shown. When a post-synaptic spike occurs at t^{post1} the value of $c_{pj}^n(t)$ increases by $b'_j(t^{post1})\bar{e}_i(t^{post1})$. Due to the appearance of a pre-synaptic spike at t^{pre2} , $c_{pj}^n(t)$ reduces by $b'_j(t^{pre2})\bar{f}^n(t^{pre2})$ as shown in the figure.	130
5.3	Specificity is reduced if only one NNLD encodes a pattern based on its first few spikes. As shown, a different pattern with a section resembling the beginning of class 1 pattern may cause neuron N_{f1} to respond.	131
5.4	The pattern memorization capacity of a NNLD (B_N) is plotted as function of the number of dendrites (m) for a fixed number of input dimensions ($d = 100$) and synapses ($s = 100$).	135
5.5	(a) The percentage of successful trials is plotted against $\frac{N}{C}$ for two-class, four-class and six-class classification. The figure shows that as $\frac{N}{C}$ increases the percentage of successful trials also increase and becomes constant after $\frac{N}{C} = 11$. (b) The evolution of CM with the number of epochs for two-class, four-class and six-class classification for $n_{sub} = 1$	138
5.6	For four class classification the above figure shows that out of 44 NNLDs (a) 7 th NNLD recognizes Class 1 pattern (b) Class 2 pattern fires for the 12 th NNLD, (c) 36 th NNLD recognizes Class 3 pattern (d) Class 4 pattern is recognized by 9 th NNLD and the latency for all four of them decreases over epochs until saturation.	139

5.7	(a) The percentage of successful trials is plotted against σ_{jitter}/τ_s for $n_{sub}=1, 5$ and 10. As the number of sub-pattern divisions are increased the jitter/noise robustness of the network decreases. (b) The percentage of successful trials is plotted against σ_{jitter}/τ_s for $n_{sub}=1, 5$ and 10 for patterns having spikes in only 50% of the afferents.	141
5.8	(a) The evolution of CM (averaged over 50 trials) with the number of epochs for two-class, four-class and six-class classification when $n_{sub}=5$. (b) This figure depicts the number of epochs needed for saturation of CM (averaged over 50 trials) vs the number of sub-patterns considered for each pattern (n_{sub}). As n_{sub} increases WTA-NNLD has to train itself for more number of sub-patterns and thus there is an increase in $ep_{sat,avg}$	142
5.9	The input and output of WTA-NNLD has been shown for a particular trial of the four class classification when $n_{sub}=5$. P_1, P_2, P_3 and P_4 represent the patterns of a particular class. The figure depicts that before learning WTA-NNLD produces arbitrary spikes whenever a pattern is presented. After learning, the network produces unique sequence of NNLD spikes for patterns of different classes. This unique sequence acts as an identifier of the pattern class.	143

5.10	Stability of WTA-NNLD trained by STDP-NRW is plotted with respect to different hardware non-idealities for $n_{sub} = 1$ (a) and $n_{sub} = 5$ (b). The constant red line indicates the percentage of successful trials obtained by our method without any nonidealities when $\sigma_{jitter}/\tau_s \approx 0.1$. The bars represent the percentage of successful trials obtained after inclusion of non-idealities. The rightmost bar marked by (...) represents the performance when all the non-idealities are included simultaneously.	145
5.11	In this figure, the information flow for the i^{th} input line connected to the p^{th} synaptic contact point on the j^{th} dendrite of the n^{th} NNLD of the WTA network is shown in (a). On the other hand, (b) shows a mechanism to calculate the correlation coefficient based fitness value of this synapse.	148
6.1	To implement the sparse connectivity matrix,a two level addressing scheme is proposed where the first memory(indexed by the incoming address of the event) holds pointers to valid connection addresses stored in the second memory.	154
6.2	Memory requirement of proposed sparse connectivity scheme is compared with that of a normal fully connected weight memory with 4 bits per weight. The proposed scheme is far more memory efficient for high dimensional inputs($d > 200$).	155
6.3	In this figure, we compare the power dissipation of the conventional and the proposed network when both are implemented in various platforms. The figures clearly depict that the proposed structural plasticity based network dissipates less power than a conventional network across all platforms.	159
6.4	A network employing NNLDs with a single hidden layer is shown. The structure has d inputs, m_H hidden NNLDs and m_L output NNLDs. Each NNLD has m_D dendrites.	161

List of Figures

- A.1 An instance of putting $s = 7$ indistinguishable particles into
 $d = 4$ distinguishable boxes modeled as a sequence of $|$'s and
 $\bullet$'s where $|$ and $\bullet$ denote a wall and a particle respectively. . . 165
- B.1 A parallel layer of n perceptrons with no lateral interconnection. 168
- B.2 A neuron having m dendrites and k synapses. 169

List of Tables

2.1	Comparison of power-efficiency between large-scale neuromorphic and conventional systems	29
3.1	Parameter Values	68
3.2	Monte Carlo Simulation Results of DPI Synapse	82
3.3	Monte Carlo Simulation Results of Square Law circuit	84
3.4	Design parameters of the Circuit Blocks in Figure 3.22	85
3.5	Comparison of DER architectures	90
4.1	Performance comparison of morphological learning rule on NNLD and Tempotron	114
5.1	Analysis of failure statistics	144
6.1	Energy per synaptic event (E_s) for various hardware platforms	158

Abbreviations

2LM	Two-Layer Model
AER	Address Event Representation
ANN	Artificial Neural Networks
AT	After Training
BMI	Brain-Machine Interface
BNN	Biological Neural Networks
CM	Convergence Measure
CMOS	Complementary Metal-Oxide Semiconductor
DAC	Digital-to-Analog Converter
DER	Dendritically Enhanced Readout
DPI	Differential Pair Integrator
DSTDTP	Doublet Spike Timing Dependent Plasticity
DT	During Training
DWM	Domain Wall magnet
ELM	Extreme Learning Machine
ES	Evolutionary Strategies
ESN	Echo State Network
extLIF	Extended Leaky Integrate and Fire
FN	False Negative
FP	False Positive
HH	Hodgkin Huxley
LCP	Local Correlation Plasticity

LIF	Leaky Integrate and Fire
LNN	Living Neuronal Networks
LSM	Liquid State Machine
LSM-DER	Liquid State Machine with Dendritically Enhanced Readout
LSM-PPR	Liquid State Machine with Parallel Perceptron Readout
LTA	Loser-Take-All
LTD	Long Term Depression
LTP	Long Term Potentiation
MAE	Mean Absolute Error
MFCC	Mel Frequency Cepstral Coefficients
MIR	Music Information Retrieval
MNIST	Mixed National Institute of Standards and Technology
MOT	Morphology Optimizing Tempotron
NMDA	N-methyl-D-aspartate
NNLD	Neuron with Nonlinear Dendrites
NRW	Network Rewiring
PPR	Parallel Perceptron Readout
PSP	Post-Synaptic Potential
RBM	Restricted Boltzmann Machine
ReSuMe	Remote Supervised Method
SD	Standard Deviation
SDSM	Separation Driven Synaptic Modification
SDSP	Spike Driven Synaptic Plasticity
SFC	SynFire Chain
SHL	Supervised Hebbian Learning
SKIM	Synaptic Kernel Inverse Method
SNN	Spiking Neural Networks
SRM	Spike Response Model
STDP	Spike Timing Dependent Plasticity
STDP-NRW	Spike Timing Dependent Plasticity based Network Rewiring

Abbreviations

SVM	Support Vector Machine
TSTDP	Triplet Spike Timing Dependent Plasticity
VLSI	Very Large Scale Integration
W-STDP	Weight dependent Spike Timing Dependent Plasticity
WTA	Winner-Take-All
WTA-NNLD	Winner-Take-All employing Neurons with Nonlinear Dendrites

Chapter 1

Introduction

1.1 Background and Motivation

The human brain consists of up to 200 billion neurons and 125 trillion connections between this huge amount of neurons [7], making it the most powerful and intelligent ‘machine’ in the world. Its exceptional power in cognition, learning and decision making is far beyond the reach of any computer or machine ever constructed by us. It is generally believed that the key to the power of the human brain lies in the enormous number of connections or ‘synapses’ between neurons. Despite the highly intricate nature of the brain, it is remarkably power efficient. With a century of great efforts by neuroscientists, we have now attained a basic understanding of the structure, function and communication protocols of the ‘Biological Neural Networks’ (BNN) in a brain. These findings are being taught in the courses of neurobiology, computational neuroscience, etc. in universities and colleges all around the world.

The discovery of new knowledge is often boosted by the advancement of technology. A better understanding of BNN led to the development of their computational models, which started the field of Artificial Neural Networks (ANN) [8] in the 1940s. However, its research went into stagnation in the

Figure 1.1: (a) The basic diagram of a neuron (b) The profile of an action potential generated by a neuron. [Source: Wikipedia]

1970s due to insufficient computer capability of that time. It regained attention later partially because of the advancement of computer power and has been subsequently applied by scientists and engineers to solve problems arising from the field of machine learning and artificial intelligence. After its advent, various types of ANN have been proposed and if we classify them according to the basic computational units (neurons) used, then three different generations can be distinguished. The primary difference between these generations arise from the extent to which they try to model the operation of a biological neuron. A biological neuron, depicted in Figure 1.1(a), collects information from other neurons and sensory inputs in the form of electrical signals through dendrites and integrates them at the soma. If the somatic voltage crosses a threshold, an action potential (Figure 1.1(b)) or ‘spike’ is produced by the neuron which propagates through the axon. The axon is divided into multiple branches ending at synapses. Excited by the action potential, the synapses release chemical neurotransmitters to the dendrites of the other neurons, completing the process of information transmission. The first generation of ANN uses the Rosenblatt’s perceptron [9] as the basic computational unit. These units, often referred to as just ‘perceptrons’ or ‘threshold gates’, encode the presence or absence of action potentials. Hence, they can only provide digital output in the form of 0 or 1. Despite being sim-

ple, they are computationally powerful as it has been shown that multi-layer perceptrons with at least a single hidden layer are universal approximators of Boolean functions [10, 11]. The basic neuron models used in the second generation of ANN apply an ‘activation function’ on a weighted sum of the inputs to provide a continuous set of possible output values. The real-valued output of these neurons are interpreted as the rate of action potentials. It has been proved that networks employing these neurons are capable of approximating any Boolean functions and can do it by using fewer number of neurons [12, 13] than the previous generation networks. In addition, these networks can approximate any continuous function [14, 15]. Although the firing-rate interpretation of the second generation ANN is more bio-realistic than the presence/absence encoding of its predecessor, it is still considered to be simplistic [16]. The main argument arises from the reaction times reported in behavioral experiments. It has been demonstrated that humans can perform pattern analysis and classification in approximately 100 *ms*, despite the fact that it involves 10 synaptic stages from the retina to the temporal lobe [17, 18]. Rolls et al. have reported similar results of visual processing in macaque monkeys [19]. Since the firing rates of neurons involved in these computations are typically below 100 Hz, hence at least 20-30 *ms* would be required just to sample the current firing rate of a neuron. These types of fast cortical computations show that firing-rate coding may be simplistic. In parallel to these revelations, experimental evidences came forward suggesting that biological neurons use the timing of action potentials to encode information [20–24]. These evidences led to the development of a third generation of ANN termed as Spiking Neural Networks (SNN) which uses spiking neurons as the basic computational units. Unlike their predecessors, spiking neurons use time as a resource during computation and communication and are computationally more powerful.

Since the neuronal communication in SNN is in the form of noise-robust, digital pulses or spikes, these networks are also amenable for low-power, low-

voltage very large scale integrated (VLSI) circuit implementations. Hence, in parallel to the progress in theoretical studies of spiking neurons, neuromorphic engineers have been developing low-power VLSI circuits that emulate sensory systems [25–27] and higher cognitive functions like learning and memory [1, 28–30]. Chan et al. [25] presented a circuit consisting of an Address Event Interface and a matched pair of silicon cochlea. They [25] modeled every section of the cochlea as a second order filter which was followed by a simplified inner hair cell circuit and a spiking neural network. Ralph et al. [26] designed and fabricated an arbitrated address-event imager employing 80×60 pixels of $32 \times 30 \mu m$. Unlike conventional imagers which integrate the photocurrent for a fixed time, in [26] the process was inverted by integrating the photocurrent until a fixed voltage i.e. threshold was obtained, which was then followed by the generation of a spike. The magnitude of the light intensity collected by each photosensitive element was inversely proportional to the interspike time interval of the pixel. The readout of each spike was initiated by the individual pixel itself, which ensured that the output bandwidth was allocated according to the pixel output demand. They showed that this method of light intensity encoding provides brighter pixels, reduces power consumption and equalizes the number of integrated photons across light intensity. Lichtsteiner et al. [27] presented a 128×128 CMOS vision sensor, where the pixels operated asynchronously in continuous time to register local relative intensity changes as spike events. Thus, the sensor output was an asynchronous stream of pixel address-events directly encoding luminance (reflectance $\times$ illuminance) changes, thereby reducing data redundancy while preserving accurate timing information.

On the other hand, there is also the need for reconfigurable spiking neural classifiers to serve as an interface to these sensors for extracting and classifying the transmitted spike information. Moreover, with the advent of Brain-Machine Interfaces (BMI), ultra-low power spike train classifiers can be used to decode, for example, motor intentions [31, 32]. These spiking networks

are typically implemented on the chip by using mixed hybrid analog-digital circuits, while the connectivity information of the spiking network is stored in an off-chip digital memory. The neuronal spikes are routed to and from the chip on a shared ‘fast’ digital bus by using an asynchronous communication protocol called the Address-Event-Representation (AER) [33, 34]. Since the connectivity information is virtual, the user has full flexibility in reconfiguring the network. The synaptic weights are usually implemented using on-chip digital to analog converters (DAC) that are shared across synapses. Very few systems with SNN have been reported to solve real-world pattern classification or memorization problems with performance comparable to state-of-the-art systems using algorithmic approaches. A major challenge in obtaining a classification accuracy comparable to a software implementation of a system on a computer is that statistical variations in VLSI devices diminish the accuracy of synaptic weights. Most of the current neuromorphic systems require high-resolution synaptic weights [1–3] and hence are affected by this problem. The typical solution for this problem is to increase transistor size. However, this alone might not be sufficient to guarantee good matching across the chip. For example, [35] demonstrates that $5\mu m \times 5\mu m$ transistor based 5-bit DACs fabricated in $0.35\mu m$ CMOS exhibit only 1.1 effective bits. Calibration techniques can be used to improve the accuracy; however, this incurs significant area penalty due to storage of calibration bits and is unacceptable in large scale systems. The problem of mismatch is also exacerbated for several nanometer scale non-CMOS devices (e.g. memristor [36–38] or domain wall magnets (DWM) [39]) that have potential for use in large scale neuromorphic applications. For example, DWM synapses are expected to have typical programming accuracies of 4-bits which can reduce to 2-3 bits effective resolution due to mismatch [39]. This problem is aggravated by the fact that device mismatch is increasing with continually diminishing transistor sizes. Hence, it has become increasingly important to design networks and algorithms which either exhibit robustness to vari-

ations or utilize low resolution synapses which are better suited for robust implementation in VLSI systems. In [28, 40], authors have proposed systems which show robustness to mismatch while using high resolution synapses. For example, Neftci et al. [28] proposed and simulated a spiking system composed of Restricted Boltzmann Machines (RBMs) which were trained by a learning rule inspired by Spike Timing Dependent Plasticity (STDP). The authors demonstrated that the proposed system is robust to the use of finite-precision synaptic weights with less than 3% decrease in performance for an MNIST [41] recognition task when 5-bit synaptic weights were used instead of 8-bit ones. However, this system used a large number (almost two million) of recurrent synaptic connections thereby making its hardware implementation quite costly. In another study, Schmuker et al. [40] et al. proposed a neuro-morphic hardware which uses population coding to achieve tolerance against variability. However, to provide acceptable performance this system needs to use a large number of high resolution synapses. Although this network provided results comparable to a standard linear classifier, its performance was not close to that of superior nonlinear classifiers like Support Vector Machine (SVM) [42] or Extreme Learning Machine (ELM) [43]. On the other hand, using low resolution or binary synapses provide a simple yet elegant solution to increase robustness. Biological evidences suggesting that synapses can exist only in a small number of states [44, 45] support this solution. Brader et al. [46] proposed a STDP learning algorithm employing bistable synapses which was used for training a network to classify handwritten digits from the MNIST database. A pool of neurons computing in parallel was used to increase the classification performance. However, this method increased the synaptic resource used by a factor equal to the number of neurons in the pool.

In this thesis, we concentrate on the second strategy and explore various solutions by constraining the synaptic weights to low resolution non-negative integers. This results in easier hardware implementation since a

low-resolution non-negative integer weight of W can be implemented by activating a shared binary synapse W times through time multiplexing schemes like address event representation (AER) [33, 34]. However, networks employing only binary synapses tend to have reduced computational power and memory capacity. To circumvent this issue we look into more bio-physical models of neurons. Taking inspiration from the nonlinear dendritic processing observed in biological neurons [5, 47–50] we propose networks employing spiking neurons with nonlinear dendrites (NNLD) and binary synapses. We only consider lumped dendritic nonlinearities, which makes our model structurally similar to the two-compartment or two-layer model [51]. Due to the presence of binary synapses, training of the networks occur through formation and elimination of connections between the inputs and the dendrites. Since the structure of the network evolves during learning, it needs reorganization of the connections between the inputs and the dendrites. Such a type of learning rule can be compared to the structural plasticity [52–55] observed in BNN. In this thesis, we have proposed both supervised and unsupervised classifiers employing NNLDs trained by different types of hardware efficient morphological rules for learning spike time codes. We present the primary contributions of this thesis in the next section.

1.2 Thesis Contribution

The three main contributions of this thesis are listed below:

- **Liquid State Machine with Dendritically Enhanced Readout:**
In the previous section we introduced that networks of NNLD with binary synapses are trained by a structural plasticity based learning rule. During training, the learning rule has to decide which inputs it should connect to a particular dendrite. This makes it a ‘choice’ based mechanism. Such type of morphological learning rules excel when the number of input dimensions is high, since the number of possible choices

increase. A typical way to map any input pattern to a sparse high dimensional space is by using non-overlapping binary valued receptive fields. However, this method is only applicable for rate-coded inputs and cannot be used in our case of spike inputs. To circumvent this problem, we have used a spiking reservoir inspired from Liquid State Machine (LSM) [6] for projecting the input patterns to a sparse high dimensional space. These high dimensional spike trains are then fed into a proposed two-NNLD architecture trained by a novel spike-based structural plasticity rule. We have also presented and analyzed architectures for on-chip implementation of the proposed network and the learning rule.

- **Morphology Optimizing Tempotron:** Inspired by Tempotron [56], we have proposed a novel structural plasticity influenced spike-timing-based learning rule for training a NNLD directly receiving high dimensional inputs. We have also proposed a novel threshold adaptation mechanism which allows the NNLD to modify its firing threshold during the training phase. In addition, this learning rule is optimized for memory capacity. We have applied the proposed technique to real life spike classification problems arising from the field of tactile sensing.
- **Unsupervised Structural Plasticity:** Apart from the two supervised learning algorithms listed above, we have proposed a novel neuro-inspired online unsupervised learning rule to train a Winner-Take-All (WTA) network composed of spiking NNLDs. Moreover, we have presented a morphological learning rule for training the proposed architecture which is inspired from the STDP learning mechanism but varies for each dendrite depending on its activation level.

1.3 Thesis Organization

The thesis is organized in the following manner. In **Chapter 1** we provide the background and motivation of our work. This is followed by a list of the primary contributions and the thesis organization.

Chapter 2 has two parts - the first part introduces SNN, provides description of spiking neurons and synapses, gives an overview of the SNN architectures, throws some light on the various learning techniques used to train them and provides some background on the VLSI implementation of SNN. In the second part we introduce the concept of neurons with nonlinear dendritic processing, iterate the challenges faced by the point-neuron hypothesis, throw some light on the biological evidences supporting the use of nonlinear dendrites in computational models and present two versions of the two-compartment or two-layer neuron model having high and low resolution synapses.

In **Chapter 3** we introduce the LSM and propose the architecture and learning rule used by Liquid State Machine with Dendritically Enhanced Readout (LSM-DER). We also present the performance of LSM-DER when applied to classification and regression tasks. Possible architectures for on-chip implementation of LSM-DER are also proposed and analyzed.

In **Chapter 4** we give a brief idea about the Tempotron learning rule and propose the framework of Morphology Optimizing Tempotron (MOT). The results of MOT for classifying both synthetic and real world high dimensional spike trains are also presented.

Chapter 5 starts with a brief survey of WTA circuits trained by STDP learning rule. Then we propose our WTA architecture composed of NNLDs and the STDP inspired online unsupervised morphological learning rule for training it. We present the performance of our network in unsupervised classification of multi-class Poisson spike trains. The robustness of the network to VLSI mismatch has also been analyzed.

Chapter 6 concludes the whole thesis. Some future works on the devel-

opment of novel architectures and learning rules for NNLD networks and their application to real world pattern recognition problems are also discussed.

Chapter 2

Literature Review

In this chapter, we provide a literature review of the concepts on which our work is based on. As pointed out in the previous chapter, the proposed work in this thesis will revolve around applying structural plasticity based learning rules to networks of spiking neurons with nonlinear dendrites and binary synapses. Hence, we divide this chapter primarily into two sections. While the first section introduces the concepts related to spiking neural networks (SNN), the second section throws some light on neurons with nonlinear dendrites.

2.1 Introduction to Spiking Neural Networks

Spiking Neural Networks (SNN) represent a special class of Artificial Neural Networks (ANN) in which neuronal communication occurs in the form of noise-robust, digital pulses or ‘spikes’. In ANN history, these are considered to be the third generation of neural networks and known to be more bio-realistic and computationally powerful than their predecessors. The primary difference between SNN and earlier generations of ANN such as perceptrons, sigmoidal neurons, radial basic function, etc. is that the input is current and its output is spiking or firing time respectively. A synapse takes spike

as input and converts it to current for subsequent neurons. Thus unlike its predecessors, spiking neurons do not fire at each propagation cycle, but only when its membrane potential reaches a certain value (threshold). When a neuron emits a spike, a signal is generated which gets distributed to other neurons thereby increasing or decreasing their potential according to the signal [57].

It has been presented that SNN can transmit and receive considerable amount of information via the relative timing of a few spikes [57–60]. This trait makes SNN especially suitable for tasks demanding fast and efficient computation such as motion detection, reactive control, etc. Also, SNN is well suited for problems where relevant information is stored in the timing of input signals such as tracking, speech recognition, etc. It has been analytically proved that spiking neurons are computationally more powerful than perceptrons and sigmoidal gates [59,61] and networks employing spiking neurons can be applied to all tasks solvable by non-spiking networks [62]. These interesting properties and advantages of SNN have attracted researchers to this field.

In this section, we first describe the basic computational units of SNN which is followed by some insight on various neural network architectures that can be implemented using these units. Next, we present some learning methodologies used to train SNN and finally conclude this section by providing some instances of VLSI implementations of SNN.

2.1.1 Models of computational units

Similar to earlier generations of ANN, SNN are composed of only two different types of units - neurons and synapses. In the following portion we provide some popular models for implementing these units.

Figure 2.1: Equivalent circuit diagram of the Hodgkin Huxley model. The model contains two active Na and K channels and one passive leakage channel.

Models of spiking neurons

A spiking neuron integrates all the input currents injected or fed from different synapses over time and whenever the membrane potential reaches a threshold, the neuron emits a spike and resets the membrane voltage. The dynamics of the membrane potential can be modeled by differential equations or integrals at different levels of detail which gives rise to multiple neuron models. We provide short descriptions of three popular models: Hodgkin-Huxley (HH), Leaky Integrate-and-Fire (LIF) and Spike Response Model (SRM).

Hodgkin-Huxley Model Hodgkin and Huxley performed experiments on a giant squid axon and proposed in [63] a neuron model explaining the ionic mechanisms behind the initiation and generation of spikes in neurons. They

discovered the presence of three different types of ion channels: an active sodium (Na) channel, an active potassium (K) channel and a passive leakage channel primarily carried by chloride ions. The active Na and K channels are described by time and voltage dependent conductances $g_{Na}(V(t), t)$ and $g_K(V(t), t)$ and the reversal potentials E_{Na} and E_K respectively where $V(t)$ is the membrane potential of the neuron. The maximum conductance of the Na and K channel are given by $\bar{g}_{Na}$ and $\bar{g}_K$ respectively which denotes all channels are open. The probability that the Na channel is open is governed by additional variables m and h whereas the probability that the K channel is open is guided by n . The passive leakage channel is characterized by the voltage independent conductance $g_L(t)$ and reversal potential E_L . The electrical equivalent circuit combining these mechanisms is given in Figure 2.1. Thus, the following equation describes the Hodgkin-Huxley model:

$$C \frac{dV(t)}{dt} = -g_L(V(t) - E_L) - \bar{g}_{Na}m(t)^3h(t)(V(t) - E_{Na}) - \bar{g}_Kn(t)^4(V(t) - E_K) + I_{in}(t) \quad (2.1)$$

where $I_{in}(t)$ and C represent the combined input current to the neuron and the capacitance of the leaky capacitor characterizing the HH circuit respectively.

Leaky Integrate-and-Fire Model The Leaky Integrate-and-Fire neuron [57] is the most simple, popular and widely used model of a formal spiking neuron. In this model the neuron's membrane is modeled as a simple RC circuit. It consists of a resistor R in parallel to a capacitor C . The membrane voltage is denoted by $V(t)$ and when it reaches a spiking threshold θ , a digital pulse is emitted by the neuron and $V(t)$ is reset to V_{rest} . If the neuron is injected with an input current $I_{in}(t)$ then the dynamics of its membrane

Figure 2.2: Equivalent circuit diagram of the Leaky Integrate and Fire neuron. A current $I_{in}(t)$ charges the RC circuit. When $V(t)$ becomes larger than the neuron spiking threshold voltage θ , an output pulse is emitted.

potential can be mathematically described as:

$$\begin{aligned} RC \frac{dV(t)}{dt} &= -V(t) + R \times I_{in}(t) \quad \text{when } V(t) < \theta \\ V(t) &\rightarrow V_{rest} \quad \text{when } V(t) \geq \theta \end{aligned} \quad (2.2)$$

Figure 2.2 shows the circuit diagram for a LIF neuron. Further mechanisms can be incorporated within this model to make it a bit closer to biological operations. For example, it is possible to add an absolute refractory period Δ_{abs} just after $V(t)$ reaches V_{rest} following a spike. During the absolute refractory period, $V(t)$ is clamped to V_{rest} and the leaky integration process starts after a time period of Δ_{abs} . Moreover, a generalized version of LIF known as Extended Integrate-and-Fire model (extLIF) has also been proposed [57], the most famous instance of which is the Quadratic LIF model.

Figure 2.3: This figure shows the effect of pre-synaptic spikes and externally injected input current on the membrane voltage ($V(t)$) and firing threshold ($\theta(t)$) of a SRM neuron. The solid and dotted lines represent the evolution of $V(t)$ and $\theta(t)$ respectively.

Spike Response Model The Spike Response Model is a generalization of the LIF model introducing richer post spike dynamics such as refractoriness and frequency adaptation. However, unlike LIF which is modeled by differential equations, SRM is formulated using filters. Moreover, the parameters of SRM are only dependent on the time elapsed ($t - \hat{t}$) since last spike $\hat{t}$ and not on voltage. The membrane voltage $V(t)$ of neuron i modeled by SRM is guided by the following rules:

1. In absence of stimuli, $V(t)$ is at the resting potential V_{rest} .
2. Whenever a pre-synaptic spike arrives at neuron i from neuron j at time t_j^f , a post-synaptic potential described by $\epsilon_{ij}(t - \hat{t}_i, s)$ is generated where $s = t - t_j^f$ and f is the spike index.
3. The response of neuron i to an injected input current $I_{in}(t)$ is described by a kernel function $\kappa(t - \hat{t}_i, s)$
4. If $V(t)$ rises to the threshold θ , a spike is generated by neuron i which is described by the kernel function $\eta(t)$. $\eta(t)$ also models the spike

afterpotential. Unlike LIF, the firing threshold may depend on the time since last post-synaptic spike i.e. $\theta = \theta(t - \hat{t}_i)$.

The above rules can be combined to form an equation describing the SRM model:

$$V(t) = V_{rest} + \eta(t - \hat{t}) + \sum_j w_{ij} \sum_f \epsilon_{ij}(t - \hat{t}_i, s) + \int_0^\infty \kappa(t - \hat{t}_i, s) I_{in}(t - s) ds \quad (2.3)$$

where $\hat{t}_i$ is the time of last spike emitted by neuron i and w_{ij} is the weight of the synaptic connection between neuron j and i . Figure 2.3 shows a schematic interpretation of Equation 2.3.

Models of synapses

In SNN, synapses take spikes or digital pulses as input and produce current output. Typically two types of synapses are implemented in a SNN: static and dynamic. The primary difference between the two is that while for a static synapse the post-synaptic current (PSC) is same for every spike it sees, the amplitude of PSC in case of a dynamic synapse depends on the spike train it has encountered so far. If a spike train is given as input to a static synapse then its output is given by:

$$I_{syn}(t) = A \sum_f e^{-\frac{t-t^f}{\tau_{syn}}} u_s(t - t^f) \quad (2.4)$$

where A , τ_{syn} , t^f , f and $u_s(t)$ denote the constant synaptic amplitude, time constant, time of spikes, spike index and step function respectively.

The computational model of dynamic synapse [64] is more elaborate and can be described by combination of two mechanisms: depression and facilitation. The synaptic depression is formulated with three parameters: absolute synaptic efficacy (A), utilization of synaptic efficacy (U) and recovery from

Figure 2.4: The output current $I_{syn}(t)$ of a static and a dynamic synapse which receive the same input spikes are compared against each other. The $I_{syn}(t)$ for both the synapses are computed by summing the exponential current kernels produced for every spike. However, while the amplitude of the kernels remain same for the static synapse, it reduces with a recovery time constant for the dynamic synapse. depression (τ_{rec}). The fraction U of the synaptic efficacy used by an incoming spike becomes instantaneously unavailable for subsequent use and recovers with a time constant of τ_{rec} . The fraction of available synaptic efficacy is denoted by R . On the other hand, synaptic facilitation is included in the

model as a pulsed increase in U by each spike. Let us denote U is the parameter that applies to the first spike in a train and u denotes the utilization of synaptic efficacy over time. u decays with a single exponential, τ_{facil} , to its resting value U . From a resting state of the synapse, all of the synaptic efficacy is available, and the fraction that remains immediately after the first spike in a train is:

$$R_1 = 1 - U \quad (2.5)$$

During the spike train, each pre-synaptic spike uses further fractions of R at the time of its arrival. R therefore constantly changes because of subsequent use by spikes, recovery of the unavailable synaptic efficacy with a time constant of τ_{rec} , and the pulsed increase in u caused by each spike. R for consecutive spikes in the train is then

$$R_{f+1} = R_f(1 - u_{f+1})e^{(-\frac{\Delta t}{\tau_{rec}})} + 1 - e^{(-\frac{\Delta t}{\tau_{rec}})} \quad (2.6)$$

where Δt is the time interval between the f^{th} and $(f + 1)^{th}$ spike and

$$u_{f+1} = u_f e^{(-\frac{\Delta t}{\tau_{facil}})} + U(1 - u_f e^{(-\frac{\Delta t}{\tau_{facil}})}) \quad (2.7)$$

Hence, the synaptic response generated by a dynamic synapse for an input spike train is given by:

$$I_{syn}(t) = \sum_f AR_f(t)u_f(t) \quad (2.8)$$

Figure 2.4 shows a comparison of the post-synaptic currents produced by a static and a dynamic synapse when provided with the same input spike train. Moreover, this figure shows the evolution of $R_f(t)$ and $u_f(t)$ through time.

2.1.2 Architectures of Spiking Neural Networks

In the previous sub-sections we introduced the structure and working principle of basic computational units of SNN. Now we will provide some insight on the construction and organization of networks employing these units. Similar to non-spiking neural networks, SNN network topologies can be classified into three categories: Feedforward, Recurrent and Hybrid networks. Below we give short description of these networks.

Feedforward networks In this topology the data transmission from input to output neurons is one-dimensional. Though the data flow can occur over multiple layers of neurons, no feedback connections are present amongst neurons. Inspired by biological neural systems, where feedforward network topologies are observed in peripheral regions, researchers have used feedforward SNN to construct low-level sensory systems of vision [65], touch [66] and olfaction [67]. The most popular learning rule for training non-spiking multilayered feedforward networks namely backpropagation has also been upgraded by introducing the notion of time [68–70], for direct application in training feedforward SNN.

Recurrent networks In this topology individual or a group of neurons interact in presence of feedback connections. Feedback connections allow the network to possess internal memory and demonstrate a dynamic temporal behavior. Thus, spiking recurrent networks can approximate any dynamical systems [71] and are computationally more powerful than feedforward networks. However, supervised training of recurrent networks is difficult compared to feedforward networks [72]. Due to the inherent ability of recurrent spiking networks to implement memory they have been used for the construction of associative [73–75] or working [76–78] memories. Moreover, spiking neurons with lateral inhibitions are used for decorrelating signals [79] or to implement competitive or Winner-Take-All type networks [80, 81] for

decision making tasks [82].

Hybrid networks This topology is constructed by combining both feed-forward and recurrent networks. Hybrid networks employ multiple subpopulations where some have strictly feedforward connections while others have recurrent connections. The interactions among the subpopulations can be either unidirectional or reciprocal. The two most widely known Hybrid networks are Liquid State Machine (LSM) [6] and Synfire Chain (SFC) [83]. LSM, proposed by Maass et al. in [6], is an architecture which shares the advantages of recurrent networks while avoiding the difficulties associated with its training. The LSM, as shown in Figure 2.5, is composed of an input layer, a huge and sparsely interconnected pool of spiking neurons termed as the ‘liquid’ and a memoryless readout. The synapses in liquid have fixed and random weights which do not require any training. On the other hand, the synapses in the readout are trained in accordance with the problem at hand. Since one of the proposed works in this thesis uses the LSM framework, we shall provide details of LSM in Chapter 3. Synfire Chain, depicted in Figure 2.6, is composed of multiple pools of neurons in which a volley of spikes propagate synchronously from one pool to the subsequent one. This is similar to a feedforward architecture. However, unlike a feedforward architecture, there may be recurrent connections within the pools.

2.1.3 Learning in Spiking Neural Networks

The learning techniques for training neural networks can be classified into three broad categories: unsupervised, supervised and reinforcement learning. In neural networks, learning happens through modification of synaptic efficacies - a process known as synaptic plasticity. Below we look into some learning methods employed in SNN.

Figure 2.5: The LSM consists of an input layer, a hugely interconnected spiking neural network termed as the liquid followed by a linear classifier stage termed as the readout. The liquid acts as a nonlinear filter by projecting the input to a high dimensional vector of spike trains which are combined at the readout. The network is trained by only tuning the weights of the connections entering the readout.

Figure 2.6: The multi-layered structure of a Synfire Chain. Volleys of spikes get propagated along the layers. Although there are no recurrent interconnections among the layers, feedback connections may exist within the layers.

Unsupervised Learning

Unsupervised or self-supervised learning refers to a scenario where training of the network is done on unlabeled data. Evidences of unsupervised learning have been observed in the brain [84,85] and a huge amount of work has been done on modeling this experimental data to develop interesting spike-based unsupervised learning rules [86,87]. For example, biological recordings from

the hippocampal and neocortical pyramidal cells [88,89] have shown that the order and relative timing of pre- vs. post-synaptic spikes induce different synaptic plasticities. Typically, a pre-synaptic spike occurring before a post-synaptic spike induces potentiation of the synaptic efficacy, while the reverse order of spikes induce depression. This protocol is termed as the Spike Timing Dependent Plasticity (STDP) rule [88,89]. In some cases, the reverse phenomenon can be encountered which is known as anti-STDP rule [90]. Both STDP and anti-STDP rule have been widely studied in [91–94]. In [57,95], Gerstner et al. presented a general mathematical formula encompassing STDP, anti-STDP and other forms of synaptic plasticity. Depending on the values of its parameters, this formula describes different plasticity rules. It is given by:

$$\begin{aligned} \frac{d}{dt}w_{ij}(t) = a_0 + S_j(t) & \left[a_1^{pre} + \int_0^\infty a_2^{pre,post}(s)S_i(t-s)ds \right] \\ & + S_i(t) \left[a_1^{post} + \int_0^\infty a_2^{post,pre}(s)S_j(t-s)ds \right] \end{aligned} \quad (2.9)$$

where $w_{ij}(t)$ is the efficacy of the synaptic connection between neuron j and i , $S_j(t) = \sum_f \delta(t-t_j^f)$ and $S_i(t) = \sum_n \delta(t-t_i^n)$ are the pre- and post-synaptic spike trains respectively (t_j^f and t_i^n denote the pre- and post-synaptic spike times with f and n being the spike indices), a_0 , a_1^{pre} and a_1^{post} are parameters and $a_2^{pre,post}$ and $a_2^{post,pre}$ are kernel functions.

The two most popular STDP based rules are doublet STDP (DSTDP) [96,97] and triplet STDP (TSTDP) [98] which are described below:

Doublet STDP In DSTDP the total weight change of the synaptic connection between a pre-synaptic neuron and post-synaptic neuron is guided by the timing of pre- and post-synaptic spikes according to:

$$\begin{aligned}\Delta w^+ &= A^+ e^{(-\frac{\Delta t}{\tau_+})} \quad \text{if } \Delta t > 0 \\ \Delta w^- &= -A^- e^{(\frac{\Delta t}{\tau_-})} \quad \text{if } \Delta t \leq 0\end{aligned}\tag{2.10}$$

where $\Delta t = t_{post} - t_{pre}$ is the time difference between a single pair of post- and pre-synaptic spikes and A_+ and A_- are parameters defining the amplitude of weight change. Equation 2.10 shows that a potentiation or increase in synaptic weight occurs when a post-synaptic spike is emitted within a pre-determined time window (specified by τ_+) after the arrival of a pre-synaptic spike. On the other hand, a depression or decrease in synaptic weight takes place when a pre-synaptic spike is preceded by a post-synaptic spike in a time window specified by τ_- . The amount of potentiation or depression depends on the time difference between the two spikes and the potentiation and depression amplitude parameters A_+ and A_- .

Triplet STDP Some experimental evidences [99, 100] have pointed out that DSTDP fails to reproduce the outcomes involving higher order spike patterns such as triplet and quadruplets of spikes. It also fails to account for the observed dependence on repetition frequency of pairs of spikes. To resolve the above mentioned issues, DSTDP model was extended to include spike triplets resulting in the TSTDTP model which could sufficiently reproduce physiological experiments. The TSTDTP rule is written as a function of difference in timing spikes as,

$$\begin{aligned}\Delta w^+ &= e^{(-\frac{\Delta t_1}{\tau_+})}(A_2^+ + A_3^+ e^{(-\frac{\Delta t_2}{\tau_y})}) \quad \text{if } t = t_{post} \\ \Delta w^- &= -e^{(\frac{\Delta t_1}{\tau_-})}(A_2^- + A_3^- e^{(-\frac{\Delta t_3}{\tau_x})}) \quad \text{if } t = t_{pre}\end{aligned}\tag{2.11}$$

where A_2^+ and A_2^- respectively, denote the amplitude of the weight change whenever there is a pre-post pair or a post-pre pair. Similarly, A_3^+ and A_3^- denote the amplitude of the triplet term for potentiation and depression,

respectively. $\Delta t_1 = t_{post} - t_{pre}$, $\Delta t_2 = t_{post}(n) - t_{post}(n - 1)$ and $\Delta t_3 = t_{pre}(f) - t_{pre}(f - 1)$ are time difference between combinations of pre- and post-synaptic spikes. τ_+ , τ_- , τ_y and τ_x are time constants.

Supervised Learning

Supervised learning in non-spiking neural networks is typically done by applying gradient-descent rules [101]. However, due to the discontinuous nature of SNN, gradient descent which is a derivative based method cannot be directly applied to it. This created a need for developing indirect methods or taking special assumptions for applying gradient descent to SNN. One such method was proposed by Bohte et al. [68, 69]. The proposed algorithm was termed as SpikeProp since it was inspired from the backpropagation algorithm used to train non-spiking multi-layered networks. In SpikeProp, each neuron is allowed to fire only once per propagation cycle and its dynamics after emitting a spike are ignored thereby eliminating the discontinuous nature of the system. Thus, the learning problem of SpikeProp can be stated as follows: given a sequence of input spike trains $S^{in}(t)$, synaptic weights have to be learnt such that the i^{th} neuron emits a spike at t_i^d . In the years following its proposal, SpikeProp has been examined in [102–105]. In [56] Gutig et al. presented the Tempotron learning rule - a spike-based binary classifier derived from the gradient descent rule. In contrast to SpikeProp, the desired firing times are not specified in Tempotron and the algorithm chooses the best time to emit a spike depending on the problem. The learning process involves finding an optimal set of weights which minimizes a cost function measuring the deviation of the membrane voltage from the firing threshold for misclassified patterns.

Carnell et al. [106] showed that popular techniques from linear algebra can be used for supervised learning of spatio-temporal spike trains. Given a set of input patterns $S_{in}(t)$ and a set of adjustable synaptic weights, the task was to approximate a target pattern $S_d(t)$. The authors have proposed few

algorithms to solve this task. In the first method they use Gram-Schmidt process [107] to find an orthogonal basis for the subspace spanned by a set of input series $S_{in}(t)$. After calculating the orthogonal basis, the best approximation in the subspace to any given element $S^d(t)$ was determined. The second solution is an iterative method where the projection of error $E = S^d(t) - S^{out}(t)$ onto direction of time series $S_{in}^i(t)$ is evaluated, where $S^{out}(t)$ denotes the obtained output spike train and the index i is chosen randomly in each iteration. The algorithm is terminated when $\text{norm}(E)$ is sufficiently small.

Another way to train SNN is by using statistical methods [108–110]. The method proposed by Barber et al. in [108] considered discrete time systems which was later extended to continuous time case by Pfister et al. [109, 110]. These models are derived from a probabilistic viewpoint in the sense that the learning rule tries to find an optimal set of synaptic weights by maximizing the likelihood of observing a post-synaptic spike train having the desired spike times, given the post-synaptic membrane potential at the location of the synapse.

Moreover, evolutionary algorithms have found their way into the training of spiking neural networks. In this method, the learning problem is formulated as a continuous optimization problem and subsequently solved by using evolutionary strategies (ES) [111]. Inspired from the inherently unsupervised Hebbian learning, Legenstein et al. [112] proposed a spike-based supervised hebbian learning rule (SHL). In SHL the networks are trained through Hebbian learning guided by a supervisory signal which is applied to the neuron as synaptic or intracellularly injected currents. Ponulak et al. [113] have proposed a modified version of SHL termed as Remote Supervised Method (ReSuMe) which provides stabler solutions than SHL and can be used for complex pattern recognition tasks.

Reinforcement Learning

It is a type of learning technique in which the algorithm tries to find the ideal behavior in a certain context, by maximizing a numerical reward signal. Although reinforcement learning had been prevalent in the machine learning community for quite some time, evidences supporting the presence of such type of learning in brain have only been found recently. Researchers have discovered that the activity of midbrain dopaminergic neurons follow reward based learning similar to reinforcement learning [114]. Based on these observations various spike-based reinforcement learning algorithms have been proposed in [115–119]. A general formula describing most of these models is given below [120]:

$$\frac{d}{dt}w_{ij}(t) = c_{ij}(t)d(t) \quad (2.12)$$

where w_{ij} is the strength of the synaptic connection between the j^{th} and the i^{th} neuron, $c_{ij}(t)$ is an eligibility trace of the connecting synapse which gathers weight changes proposed by STDP and $d(t) = h(t) - h_o(t)$ computes the concentration of the neuromodulatory signal $h(t)$ around its mean value $h_o(t)$. Equation 2.12 connects reinforcement and STDP learning theory and is applicable to both feedforward and recurrent networks.

2.1.4 Spiking Neural Networks in hardware

In recent past, numerous low power hardware implementation of SNN have been proposed [121–127]. This field, a sub-field of neuromorphic engineering [128], has gained popularity among researchers due to the following advantages:

1. Hardware systems implementing SNN are event (spike) based, so the active power dissipation is proportional to firing activity.
2. Neuromorphic systems typically use low power analog circuits for build-

ing computational blocks and reliable digital schemes for communication among these blocks, thereby exploiting the advantages offered by both of these domains.

3. These systems often use an asynchronous multiplexing technique called address event representation (AER) where the connection matrix, for communication among analog blocks, is stored in a configurable digital memory. Thus, AER equips neuromorphic systems with arbitrary connection flexibility.

In Section 2.1.1, we presented few popular models of spiking neurons and synapses. There has been significant work on implementing these models in hardware. We invite the reader to look into [129] and [130] for excellent overviews of circuit implementation of spiking neurons and synapses respectively. Neuromorphic systems are primarily of two types: large scale general purpose systems with a huge number of neurons and synapses but precluding synaptic plasticity and small scale custom systems supporting on-chip learning but having fewer neurons and synapses. The pioneer platforms concentrating on the former type include University of Manchester’s SpiNNaker [123, 131], IBM’s TrueNorth chip [124] and Stanford University’s Neurogrid [125]. SpiNNaker intends to enhance the performance of software simulations by integrating 18 mobile processors onto a single die [123, 131]. The IBM TrueNorth chip uses digital electronics to build basic cores emulating simple neural networks which can be connected to form larger structures [124]. The Neurogrid also takes an analog approach and optimizes the transistor count by sharing synapse and dendritic tree circuits [125]. Compared to traditional platforms such as CPUs and GPUs, these spiking hardware systems require much lower power to operate. In Table 2.1 we compare the average power density required by these systems and traditional systems. These results depict that large-scale hardware implementations of SNNs are well-suited for low-power applications.

On the other hand, there has been significant work on the later type of

Table 2.1: Comparison of power-efficiency between large-scale neuromorphic and conventional systems

	TrueNorth	Neurogrid	SpiNNaker	GPU [132]	CPU [124]
Average Power Density	20 mW/cm ²	50 mW/cm ²	1000 mW/cm ²	25 W/cm ²	75 W/cm ²

systems implementing on-chip synaptic plasticity. In the recent past, various systems have been proposed emulating STDP. For example, a feedforward network with STDP learning was proposed in [133] which successfully implements the targeted synchrony detection. However, this system requires a lot of power and large area to operate due to the presence of power hungry biasing circuits and huge capacitors respectively. These disadvantages were tackled in [134] which employed an array of neurons with bistable STDP synapses. Moreover, unlike the system proposed in [133], it was reconfigurable since it utilized the AER communication protocol. Further optimization of area for STDP learning has been done by Schemmel et al. in [135]. Arthur et al. [136] proposed a neuromorphic chip which has on-chip SRAM cells for storing a binary state of the synaptic efficacies which were updated according to STDP learning rule. Koickal et al. [137] proposed an adaptive olfactory neural network trained by on-chip STDP learning. An on-chip implementation of Hopfield network trained by STDP for associative memory was proposed in [138]. Seo et al. [121] developed a highly efficient neural network with 256 neurons and 64000 synapses for implementing associative memories and solving pattern recognition tasks [121]. An on-chip implementation of weight dependent STDP (W-STDP) was presented in [29] which was able to exploit the physical constraints of CMOS transistors. Hence, in comparison to previous STDP learning chips, the proposed system had lesser area and power requirements. In addition to STDP, other forms of synaptic plasticity have also been implemented in neuromorphic hardware. For exam-

ple, an on-chip implementation of Spike Driven Synaptic Plasticity (SDSP) was presented by Mitra et al. [30] and used for complex pattern recognition tasks. A BCM like voltage dependent rule termed as Local Correlation Plasticity (LCP) has been implemented by Mayr et al. [122, 139]. Few other works throw some light on the implementation of more bio-realistic models of STDP [140, 141].

2.2 Introduction to Neurons with nonlinear dendrites

Dendrites are branches projecting from the cell body of neurons which serve as a platform for synaptic communication among neurons in a biological neural network. Since the beginning of twentieth century, researchers have looked into local dendritic integration from both experimental and computational viewpoint. Recent biological findings have suggested that dendrites are capable of locally integrating synaptic inputs in a nonlinear manner [142]. This enables a single neuron to perform a series of complicated computations such as sound localization [143], coincidence detection [144], collision avoidance [145], etc. In this section, we look into computational models of pyramidal neurons with nonlinear dendrites. We consider pyramidal cells since these are the most abundant neurons in the cerebral cortex and are often associated with higher cognitive and emotional processes in brain [146, 147].

2.2.1 The point neuron hypothesis: Structure and Challenges

The traditional view of neuronal operation is that excitatory and inhibitory synaptic currents arriving across the dendritic arbor are integrated at the soma, where the net current determines the neuron's firing rate and timing. The classical view suggested that dendrites have no inherent process-

ing abilities and are present only to increase the neuron’s receptive surface area [148, 149]. Thus, many multineuron architectures modeling various aspects of brain function had kept dendritic processing out of the equation [86, 150–159]. These studies have encouraged the development of many dendrite less “point neuron” architectures in artificial neural network literature. Although both spiking and non-spiking point neuron architectures have found considerable success for pattern recognition and have been broadly accepted, the model itself faces two primary challenges:

1. Neurophysiological data indicates that dendrites can generate “active” responses similar to spikes with varying degrees of locality and based on a variety of ionic mechanisms [4, 160–164].
2. Biophysical studies have shown that a neuron’s cable properties promote spatially restricted synaptic interactions [49, 50, 165–169].

These two observed phenomena suggest that neurons are capable of performing local nonlinear computations simultaneously within multiple dendrites - an operation which the point neuron model is unable to perform.

2.2.2 Dendritic branches as key computational elements

The discovery of voltage-dependent ion channels with distinctive densities and biophysical properties across the axo-somato-dendritic axis demonstrates that both the soma and dendrites have an active part in forming the neuronal excitability. The conductance and distribution of the several voltage-dependent ion channels present in the dendritic branches vary according to the type of neuron and dendrite. Thus, in addition to the morphological features and passive properties of dendrites, the integration of synaptic inputs also depend on the expression profile and kinetics of their active conductances.

Early theorists have suggested several mechanisms of local nonlinear computations in dendrites and techniques for combining outputs from multiple

dendrites at the soma. For example, Koch et al. [47] proposed the concept of a subregion or “subunit” within the dendrite where local synaptic computations can occur with little interaction with other subregions. They also demonstrated that the divisive interaction between a shunting or “silent” inhibitory synapse and an excitatory synapse could provide a logical ‘AND’-‘NOT’ operation, with the inhibitory signal acting as a local “veto” signal. They proposed that this logical operation, performed in various different “subunits” of a dendrite, could implement a biophysical mechanism of direction selectivity [170] in line with the work of Barlow and Levick [171]. The authors of [172] showed that voltage-dependent sodium ion channels could implement logical ‘AND’ and ‘OR’ operations between neighbouring synapses. They also proposed that by combining these operations, a dendritic branch can implement a hierarchical Boolean logic function. In [48], Rall et al. demonstrated that mixtures of active and passive spines could facilitate complex and diverse nonlinear interactions between inputs and different branches of a dendrite. Zador et al. [173] demonstrated that voltage-dependent sodium ion channels could implement a ‘XOR’ function between inputs projected to two different sites of a dendrite. The authors of [174–176] proposed the “sigma-pi” unit, in which the dendrites produced a set of low-order product terms between groups of interacting synapses which were subsequently combined at the soma. Two other models having local dendritic computations are “clusteron” [49, 177–180] which is a neuron model exploiting the fact that dendrites containing N-methyl-D-aspartate (NMDA) and voltage-dependent sodium ion channels demonstrate cluster sensitivity i.e. excitatory synapses activated in multiple clusters can produce a strong dendritic output than diffusely delivered inputs and “contextron” [181] which is a simplified neuron model where a set of dedicated modulatory inputs enables different subregions in a dendrite in different time instances.

The several active properties of dendrites enhance the computational power of pyramidal neurons and introduce spatio-temporal specificity to

their function output [142]. Experiments suggest that in pyramidal neurons, synaptic inputs are combined linearly when they are located on different dendrites and nonlinearly when located within the same dendritic branch [4, 50, 182, 183]. To demonstrate this phenomenon, Polsky et al. [4] performed two sets of experiments. In the first set, they put two electrodes near a basal dendrite of a 5-layer pyramidal neuron. The electrodes were first stimulated individually (black traces in Figure 2.7) and then simultaneously (red traces in Figure 2.7). The blue traces of Figure 2.7 show the arithmetic sum of the individual responses. The figure depicts that the combined response for simultaneous excitation was (a) linear for weak inputs, (b) supralinear for medium inputs and (c) sublinear for strong inputs. In the second set of experiments, the authors placed the two electrodes near different branches and showed that the dendritic response for simultaneous excitation is approximately linear for (d) weak, (e) medium or (f) strong inputs. The results of these experiments strongly suggest the presence of a saturating nonlinearity in dendrites of pyramidal neurons. Oviedo et al. [184] presented that depending on the spatial distribution of incoming signals the inputs could be integrated sublinearly. This results in the formation of distinct dendritic subunits which can combine inputs in a quasi-independent manner. This compartmentalization of information processing equips the neurons with a second processing layer that boosts the computational capacity of the neuron by at least an order of magnitude compared to that of a thresholding point neuron [5]. Most of the models mentioned above suggest that a neuron calculates a sum of nonlinear terms, with each term representing the interaction between a group of neighbouring synaptic inputs.

2.2.3 The two-layer neuron model (2LM)

The simplest model capturing the essence of nonlinear dendritic computation in a neuron is the two-layer model (2LM) [51]. The assumptions introduced in 2LM for computational simplicity are:

Figure 2.7: An experiment performed by Polsky et al. [4] demonstrating that dendrites of pyramidal neurons are associated with saturating nonlinearities.

1. It describes the behavior of a subtree consisting of a uniform set of thin dendrites projecting from a central node.
2. The subtree consists of only unbranched terminal dendrites. This is supported by the fact that most of the excitatory synapses connect to the long and unbranched sections of a PN thin dendrite [185, 186].
3. Inspired from cable theory suggesting that signal communication within

Figure 2.8: The 2LM showing a neuron with with nonlinear dendrites.

a dendrite is efficient whereas inter-dendritic communication is poor [47, 173, 187], the 2LM assumes that communication within a dendrite is ideal and there is no communication among dendrites.

Figure 2.8 shows the 2LM where the input x_i is connected to the j^{th} dendrite via a synaptic connection of weight w_{ij} . Each dendrite can be described by a linear-nonlinear (LN) cascade which performs a linear weighted sum of the inputs given to it and passes it to a nonlinear function $b(\cdot)$ describing the dendritic nonlinearity. In 2LM only lumped dendritic nonlinearities are considered. The output of the dendrites z_j are summed at the soma with branch weights W_j to form the total somatic current I_s . I_s is then fed into the neuronal activation function $g(\cdot)$ to obtain the final network output y . The output the 2LM model can be mathematically described as:

$$y = g\left(\sum_j W_j b\left(\sum_i w_{ij} x_i\right)\right) \quad (2.13)$$

It is to be noted that the 2LM is isomorphic to a feedforward artificial neural network with a single layer of hidden neurons and one output neuron.

The lumped dendritic nonlinearities $b()$ are equivalent to the hidden neurons interposed between the input and output layers. Thus the theories and learning rules developed for artificial neural networks [101] can be applied directly to the 2LM. Another advantage of the 2LM is that it establishes a connection between the function of dendrites and neurophysiological data. For example, the receptive fields of V2 “complex cells” are usually modeled as a sum-of-subunits structure [188–190] similar to the 2LM. A similar method of pooling multiple subunits is contemplated to account for increasing spatial invariance of receptive fields along the visual cortical form processing pathway [191–195], thereby increasing the possibility that dendritic subunits are the location for a neuron’s functional subunits [47, 97, 167, 196–201]. Moreover, it has been shown in [5, 200, 202, 203] that the partitioning of dendrites into discrete subunits facilitates the analysis of memory and processing capacity. Poirazi et al. [182] studied in detail the specific type of dendritic nonlinearities which might best mimic the observed neuronal firing activities.

The 2LM assumes that dendrites are locationless i.e. it performs an weighted sum of the inputs independent of their location within the subunit. It is criticized due to this assumption, since many biological evidences suggest it to be simplistic. It has been shown that unlike uniform cables, dendrites display highly asymmetric summation and attenuation properties [165, 166, 170, 187, 204–206] which are not captured in the 2LM. However, simulating and training networks constructed with neuron models including these properties would be computationally very intensive. Thus, the 2LM does as a good trade off between computational complexity and biological significance.

In the following sub-section, we show the architecture and learning rule of a particular type of 2LM proposed by Mel and Poirazi [5] which considers binary synapses.

2.2.4 The simplified 2LM using low resolution synapses

Mel and Poirazi [5] proposed an interesting version of the 2LM which uses low resolution instead of high resolution synapses. The inclusion of low resolution synapses in their model was motivated by the following observations:

1. Recent experiments showed that the strength of synaptic transmission at cortical synapses can experience considerable fluctuations “up” and “down” representing facilitation and depression respectively, or both, when excited with short synaptic stimulation and these dynamics are distinctive of a particular type of synapse [207–210]. This kind of short-time dynamics is in contrary to the traditional connectionist models assuming high resolution synaptic weight values and conveys that synapses may only have few states.
2. Moreover, another experimental research on long-term potentiation (LTP) in the hippocampus region of the brain revealed that excitatory synapses may exist in only a small number of long-term stable states, where the continuous grading of synaptic efficacies observed in common measures of LTP may exist only in the average over a huge population of low resolution synapses with randomly staggered thresholds for learning [44].

They showed that neurons with active dendrites i.e. individual dendrites equipped with lumped nonlinearity and low resolution synapses possess higher storage capacity than their linear counterparts. The proposed neuron model, depicted in Figure 2.9, has m dendritic branches connected to it with each branch having k synaptic contact points of weight 1. Note that although the first stage of this model is visually similar to the 2LM model depicted in Fig. 2.8, it has the following constraints. If $\mathbf{x}$ is a d dimensional input vector to this system, then each synaptic contact point is excited by any one of the d afferents where $d \gg k$. The model allows multiple connections of one input dimension to a single dendrite but restricts the number of

Figure 2.9: A neuronal cell with active dendrites as proposed by Mel and Poirazi [5]

connections per dendrite to k . The output response of j^{th} dendritic branch is calculated as a nonlinear weighted sum of the currents of the k synaptic points connected to the branch and is given by $z_j = b(\sum_{i=1}^d w_{ij} x_i)$, where $b(\cdot)$ is the dendritic nonlinearity modeled as a nonlinear activation function, $w_{ij} \in \{0, 1, \dots, k\}$ is the number of times the i^{th} input line is connected to the j^{th} dendritic branch and x_i is the value at the i^{th} input line. The number of synaptic contact points per dendrite are fixed to k by enforcing $\sum_{i=1}^d w_{ij} = k$ for each j . We also define $v_j = \sum_{i=1}^d w_{ij} x_i$ which is the weighted sum of the currents of the k synaptic points of branch j and is input to the branch's dendritic nonlinearity. Combining (non-weighted summation unlike Fig. 2.8) all the dendritic responses, the output $a_N(\mathbf{x})$ of the neuronal cell is given by:

$$a_N(\mathbf{x}) = \sum_{j=1}^m z_j = \sum_{j=1}^m b(v_j) = \sum_{j=1}^m b\left(\sum_{i=1}^d w_{ij} x_i\right). \quad (2.14)$$

The authors constructed a supervised classifier employing two such neuronal cells and evaluated its performance on binary classification tasks involving boolean patterns. The classifier is trained by modifying the morphology of the network i.e. by updating the connections between the inputs and dendrites guided by an iterative learning rule derived from the stochastic gradient descent rule. One of the neuronal cells is termed as the positive cell having output $a^+(x)$ and other negative cell with output $a^-(x)$. The overall classifier output is given by $y(x) = g(a^+(x) - a^-(x))$ where $g(y) = 1/[1 + \exp(-y/0.05)]$ is a global output nonlinearity. Since, the positive and negative cells are trained to produce a higher output for different classes of patterns, the learning of the two cells is complimentary. The training mechanism consisted of the following steps:

1. At every iteration, a random set T of n_T synapses is selected for probable replacement. The performance index ϕ_{pj} for the p^{th} synaptic contact point of the j^{th} dendrite is calculated for each synapse in the set n_T as $\phi_{pj} = \langle x_i b'_j g'(y)(t - g(y)) \rangle$ for the positive cell and $\phi_{pj} = - \langle x_i b'_j g'(y)(t - g(y)) \rangle$ for the negative cell where x_i is the input line connected to this synaptic contact point. Here, $\langle \rangle$ indicates averaging over the training set. The synapse with the lowest value of ϕ_{ij} in T denoted as T_{min} is labeled for replacement.
2. The replacement synapse is chosen from a candidate set R having n_R of the d input lines. The set R is created by placing $n_R (\leq d)$ unique randomly selected input lines on the branch with the lowest ϕ_{pj} synapse via synapses with weight 1. These are called 'silent synapses' since they do not contribute PSP at the soma. The performance index ϕ_{pj} is calculated for all of these silent synapses and the best performing synapse (maximum ϕ_{pj}) is used to replace T_{min} .
3. The above steps are repeated until no further improvement in performance is observed.

2.3 Summary

In the first part of this chapter we have introduced various aspects of spiking neural networks and in the second part we have presented concepts related to neurons with nonlinear dendrites. In Section 2.1 we have first looked into popular models of the basic computational units of SNN i.e. spiking neurons and synapses. Next, we have presented how these units can be combined to form various SNN architectures. Subsequently, we have shown different types of learning rules to train these architectures. We have concluded this section by giving some instances of hardware implementations of spiking networks. In the following Section 2.2 we have first shown the challenges and limitations of traditional neuron models used in software and hardware implementation of neural networks. The fact that these neurons do not have nonlinear dendritic elements has been criticized and in the following subsection biological evidences supporting the presence of nonlinear or active dendrites have been presented. Next, we have presented the computational model of a non-spiking neuron having nonlinear dendrites and high resolution synapses termed as the 2LM. We have also presented the structure and learning rule of a 2LM neuron using low resolution synapses. This chapter explained various concepts on which the work proposed in this thesis is based. In the next three chapters we will explore the contributions of this thesis.

Chapter 3

Liquid State Machine with Dendritically Enhanced Readout

In this chapter, we describe a new neuro-inspired, hardware-friendly readout stage for the liquid state machine (LSM), a popular model for reservoir computing. Compared to the parallel perceptron architecture trained by the p -delta algorithm, which is the state of the art in terms of performance of readout stages, our readout architecture and learning algorithm can attain better performance with significantly less synaptic resources making it attractive for VLSI implementation. Inspired by the nonlinear properties of dendrites in biological neurons, our readout stage incorporates neurons having multiple dendrites with a lumped nonlinearity (two compartment model). The number of synaptic connections on each branch is significantly lower than the total number of connections from the liquid neurons and the learning algorithm tries to find the best ‘*combination*’ of input connections on each branch to reduce the error. Hence, the learning involves network rewiring (NRW) of the readout network similar to structural plasticity observed in its biological counterparts. We show that compared to a single perceptron using

analog weights, this architecture for the readout can attain, even by using the same number of binary valued synapses, up to 3.3 times less error for a two-class spike train classification problem and 2.4 times less error for an input rate approximation task. Even with 60 times more synapses, a group of 60 parallel perceptrons cannot attain the performance of the proposed dendritically enhanced readout. An additional advantage of this method for hardware implementations is that the ‘choice’ of connectivity can be easily implemented exploiting address event representation (AER) protocols commonly used in current neuromorphic systems where the connection matrix is stored in memory. Also, due to the use of binary synapses, our proposed method is more robust against statistical variations.

3.1 Introduction

In Chapter 1.1, we discussed that the problem faced by current neuromorphic systems is the requirement of high resolution tunable synaptic weights. Due to the presence of nonidealities in VLSI implementations, the accuracy of these weights decrease which leads to reduced performance. An architectural solution is partly provided by the Liquid State Machine (LSM) [6], which requires training of very few weights while the others can be random. The LSM network, depicted in Figure 3.1, consists of three stages: an input layer which projects the input pattern into the the second stage or the liquid, which is a recurrent neural network (RNN) with randomly weighted, mostly local interconnections performing the task of mapping the input to internal states. These states are then used by the third stage or the readout circuit to provide the overall system output. The readout is trained in a task-specific way which requires updating the weights of the synapses connecting the liquid and the readout. Hence, this does not entirely eliminate the need for high-resolution, non-volatile, tunable weights.

RNNs in general, though computationally very powerful, need updating

Figure 3.1: The first stage of LSM [6] is the input layer. The input stage is followed by a pool of recurrent LIF neurons whose synaptic connections are not trained. The next stage is a simple linear classifier that is selected and trained in a task-specific manner.

of the weights of the synapses forming the network. Though several learning techniques for modifying the synaptic weights have been proposed [211–215], an optimal solution is yet to be found. In this context, LSM provides a big advantage since only the weights of the synapses connecting the RNN to the readout need to be modified in an LSM, while the interconnections within the RNN pool are fixed. This is clearly useful for VLSI implementations since tunability or high resolution are not required for most weights. However, the number of tunable weights needed in the readout stage for good performance may become a bottleneck. The state of the art readout stage of LSM using a single layer network as readout, is usually composed of a single layer of parallel perceptrons (we denote LSM with a parallel perceptron readout as LSM-PPR) that are trained by the p -delta algorithm [6, 216–219]. For this readout stage of LSM-PPR, the number of tunable weights will be very high and equal to $L \times n$ (provided all liquid neurons are connected to the readout), where L and n are the number of liquid and readout neurons respectively, thereby making it infeasible for low-power smart sensors. To decrease the number of tunable synapses, neuromorphic systems often use an asynchronous multiplexing technique called address event representa-

tion (AER) where the connection matrix is stored in a configurable digital memory. Using AER, it is possible to have a large number of synapses for readout; however, the huge power dissipated in accessing memory for every spike makes this solution infeasible for low-power applications.

In this chapter, we propose a novel architecture and training procedure for the readout stage of LSM that is inspired by the nonlinear processing properties of dendrites and structural plasticity (forming or breaking of synapses, re-routing of axonal branches etc.) in biological neurons. This solution was motivated by the fact that the liquid neurons in LSM produce sparse, high dimensional spike train outputs which is a key requirement in our work as discussed in Chapter 1.2 and presented in [220]. The method proposed in this chapter, which we refer to as LSM with dendritically enhanced readout (LSM-DER), has the following benefits:

- It can reduce the error as compared to p -delta which is the current state of the art algorithm for the training of LSM readout [221], [222].
- It uses an order of magnitude less synaptic resources than parallel perceptrons while achieving comparable performance thus making it feasible for implementation in low-power smart sensors.
- The synapses connecting the liquid and readout can even have binary values without compromising performance. This is also very useful in hardware implementations since it removes the need for high resolution weights.

A dendritic neuron with network rewiring (NRW) rule for classifying high dimensional binary spike rate patterns was presented in [220]. The primary differences between this work and the one proposed in [220] are as follows:

- We present a modified NRW rule that can be applied to arbitrary spike trains and not only rate encoded inputs.
- The modified rule can be used for solving approximation problems.

- For the first time, we demonstrate a dendritic neuron can be used as a readout for LSM.
- We demonstrate the stability of this architecture with respect to parameter variations making it suitable for low-power, analog VLSI implementations.

In the following section, we shall present a review about LSM, parallel perceptrons, the p -delta learning rule and provide details about the neuron model having nonlinear dendritic processing. Section 3.3 provides the LSM-DER algorithm in detail and also proposes the NRW algorithm. The first part of Section 3.4 discusses the classification and approximation tasks used to demonstrate the efficacy of the proposed architecture. Section 3.4 then provides the performance of LSM-DER on these problems and compares it with that of the traditional LSM-PPR. This section also sheds light on the dependance of the performance of LSM-DER on several key parameters. In Section 3.6, we present an architecture of DER which is suitable for on-chip testing when the training has been done in software and the trained connections are imported to the chip. In this section we will also present the robustness of the algorithm to variations in parameters, a quality that is essential for its adoption in low-power, sub-threshold neuromorphic designs which are plagued with mismatch. In Section 3.7 we look into possible architectural implementations of DER such that it can be used for on-chip training. We also provide a modification of the NRW learning rule which allows it to accommodate real time learning. Lastly, we summarize the work presented in this chapter in Section 3.8 and provide some concluding remarks.

3.2 Background

In this section, we will first present, for the sake of completeness, some of the theory about the operation of the LSM, parallel perceptron readout and

the p -delta training algorithm. Then, we shall introduce our dendritically enhanced readout stage constructed of two NNLD and the corresponding NRW algorithm for training it.

3.2.1 Theory of Liquid State Machine

LSM is a reservoir computing method developed from the viewpoint of computational neuroscience by Maass et al. It supports real time computations by employing a high dimensional heterogeneous dynamical system which is continuously perturbed by time varying inputs. The basic structure of LSM is shown in Figure 3.1. It comprises three parts: an input layer, a reservoir or liquid and a memoryless readout circuit. The liquid is a recurrent interconnection of a large number of biologically realistic Leaky Integrate and Fire neurons (LIF) using dynamic synaptic connections in the reservoir. The readout is implemented by a feed-forward neural network that does not possess any lateral interconnections. The LIF neurons of the liquid are connected to the neurons of the readout. The liquid does not create any output but it transforms the lower dimensional input stream to a higher dimensional internal state. These internal states act as an input to the memory less readout circuit which is responsible for producing the final output of the LSM.

If $u(t)$ is the input to the reservoir of a machine M , then the liquid neuron circuit can be represented mathematically as a liquid filter L^M which maps the input function $u(t)$ to the internal states $x^M(t)$ as:

$$x^M(t) = (L^M u)(t) \quad (3.1)$$

The next part of LSM i.e. the readout circuit takes these liquid states as input and transforms them at every time instant t into the output $y(t)$ given

by:

$$y(t) = f^M(x^M(t)) \quad (3.2)$$

The liquid circuit is general and does not depend on the problem at hand whereas the readout is selected and trained in a task-specific manner. Moreover, multiple readouts can be used in parallel for extracting different features from the internal states produced by the liquid. For more details on the theory and applications of LSM, we invite the reader to refer to [6].

3.2.2 Previous research on Liquid State Machine

After LSM's inception, many research works have been published which focus on either the improvement of the LSM framework or its applications in various real world problems. We provide a brief survey of such works.

Research on improving the LSM framework

Zhang et al. [223] proposed a digital implementation of LSM for online pattern recognition tasks. The authors presented a local learning rule where synaptic weight updates happen according to the pre- and post-synaptic firings. The proposed rule does not require any global interaction across the network and hence lends itself to efficient parallel VLSI implementation. In [224], a novel Spike Timing Dependent Plasticity (STDP) based learning rule for generating a self-organized liquid was proposed. The authors showed that LSM with STDP learning rule provides better performance than LSM with randomly generated liquid. Han et al. [225] have studied in detail the effect of the distribution of synaptic weights and synaptic connectivity in the liquid on LSM performance. In addition, the authors have proposed a genetic algorithm based rule for the evolution of the liquid from a minimum structure to an optimized kernel with a optimal number of synapses and high classification accuracy. The authors of [226] have used the Fishers Discrim-

inant Ratio as a measure of the Separation Property of the liquid. Then they have used this measure in an evolutionary framework to generate a liquid with suitable parameters, such that the performance of the readout gets optimized. Schliebs et al. [227] have proposed an algorithm to dynamically vary the firing threshold of the liquid neurons in presence of neural spike activity such that LSM is able to achieve both a high sensitivity of the liquid to weak inputs as well as an enhanced resistance to over-stimulation for strong stimuli. Wojcik et al. [228] simulated the LSM by forming the liquid with Hodgkin-Huxley neurons instead of LIF neurons as used in [6]. They have done a detailed analysis on the influence of cell electrical parameters on the Separation Property of this Hodgkin-Huxley liquid. In [229], Notley et al. have proposed a learning rule which updates the synaptic weights in the liquid by using a tri-phasic STDP rule. Frid et al. [230] have proposed a modified version of LSM which can successfully approximate real valued continuous functions. Instead of providing spike trains as input to the liquid, they have directly provided the liquid with continuous inputs. Moreover, they have also used neurons with firing history dependent sliding threshold to form the liquid. In [231], the authors have shown that LSM in its normal form is less robust to noise in data but when certain biologically plausible topological constraints are imposed then the robustness can be increased. The authors of [232] have presented a liquid formed with stochastic spiking neurons and termed their framework as pLSM. They have shown that due to the probabilistic nature of the proposed liquid, in some cases pLSM is able to provide better performance than traditional LSM. Ortman et al. [233] have used Living Neuronal Networks (LNNs) as reservoir of LSM and presented a detailed study on its pattern separability. In [234], the authors have proposed novel techniques for generating liquid states to improve the classification accuracy. First, they have presented a state generation technique which combines the membrane potential and firing rates of liquid neurons. Second, they have suggested to represent the liquid states in frequency do-

main for short-time signals of membrane potentials. Third, they have shown that combination of different liquid states lead to better performance of the readout. El-Laithy et al. [235] have demonstrated how LSM is able to process various input data as a varying set of transiently stable states of collective activity. Kello et al. [236] have presented a self tuning algorithm which is able to provide a stable liquid firing activity in presence of a varied range of input. A self-tuning algorithm is used with the liquid neurons which adjusts the post-synaptic weights in such a way that the spiking dynamics remain between sub-critical and super-critical. In [237], the authors have proposed a learning rule termed as ‘Separation Driven Synaptic Modification (SDSM)’ for training the liquid, which is able to construct a suitable liquid in fewer generations than random search.

Research on real world application of LSM

LSM has been applied to many real world problems arising from the diverse domain of science and engineering. Zhang et al. [217] have used LSM to efficiently solve the problem of planning a robot’s path in less time while avoiding obstacles. Verstraeten et al. [238] have employed LSM for identifying isolated spoken digits. They have studied the performance of LSM for two different speech front ends, namely Mel Frequency Cepstral Coefficients (MFCC) and Lyon Passive Ear model, and also for three different speech to spike coding method- the classical Poisson spike coding, the BSA filter coding scheme and encoding by a Leaky Integrate and Fire (LIF) neuron. Goodman et al. [239] have successfully applied LSM in identifying the boundaries of spatio-temporal patterns in stockpile surveillance data. They have shown that LSM has the potential to solve the spoken phoneme recognition task. In [240], the authors have used LSM as a domain constrained activity classification system to classify the inertial sensor data obtained from horse riders to recognize activities of interest within equestrian sport. Probst et al. [241] have used LSM to mimic a piece of layer IV of rat somatosensory cortex,

and subsequently trained the readout of LSM to move a rolling ball towards a target by controlling the inclination angles of the surface that is holding the ball. In [218], the authors have used time delayed synapses to connect the liquid to the readout and used this framework for recognizing spoken language. Elmir et al. [242] have employed LSM for automatic fingerprint recognition. They have selected the widely recognized FVC2004 and Finger Cell databases and have shown that LSM is capable of providing comparable performance to state-of-the-art algorithms in less time, thereby making it suitable for real time recognition. The authors of [243] have applied LSM in the field of Music Information Retrieval (MIR) for music classification and clustering. They have shown that LSM can be successfully employed for classifying musical styles (for example, ragtime vs. classical) and it is capable of distinguishing music from note sequences without temporal structure.

3.2.3 Parallel Perceptron Readout and the p -delta learning algorithm

In the section, we present the state-of-the-art parallel perceptron readout (PPR) and provide a detailed description of the p -delta learning rule used to train it.

Parallel Perceptron Readout

The readout stage employed in LSM-PPR is a layer of parallel perceptrons. A single layer composed of a finite number of perceptrons, each receiving the same input, is called a parallel perceptron as shown in Figure 3.2. A single perceptron with input $\mathbf{x}=[x_1, x_2..x_d]$, considering that the constant bias has been converted to one input, computes a function $f : \mathbb{R}^m \rightarrow \{-1, 1\}$ defined as:

Figure 3.2: In LSM-PPR, the readout stage is composed of a single layer of perceptrons which do not have any lateral connections. The output of the liquid is connected to each perceptron of this parallel perceptron stage.

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} \geq 0 \\ -1 & \text{otherwise} \end{cases} \quad (3.3)$$

where $\mathbf{w} \in \mathbb{R}^d$ is the synaptic weight vector. To distinguish the neuronal functions in parallel perceptron readout (PPR) from dendritically enhanced readout (DER), we denote them by f^{PPR} and f^{DER} respectively. Let us now consider a parallel perceptron layer having n number of perceptrons with outputs $f_1^{PPR}, f_2^{PPR}, \dots, f_n^{PPR}$ where $f_i^{PPR} : \mathbb{R}^m \rightarrow \{-1, 1\}$. Then the output of the parallel perceptron readout is given by:

$$\hat{o} = g(p) = g\left(\sum_{i=1}^n f_i^{PPR}(\mathbf{x})\right) \quad (3.4)$$

where $p = \sum_{i=1}^n f_i^{PPR}(\mathbf{x}) \in [-n, \dots, n]$ and $g : \mathbb{Z} \rightarrow \mathbb{R}$ is the squashing function. The function $g()$ is chosen according to the type of computation.

The p -delta learning algorithm

PPR is trained by a simple yet efficient method termed as p -delta rule proposed by Auer et al. in [216]. This simple rule can be utilized to approximate any boolean and continuous functions. Two advantages of this rule over traditional back propagation algorithm is that it is required to modify the weights of only a single layer of synapses and the computation and communication of high precision analog values are not required. We shall present some of the salient features of the algorithm here while inviting the reader to refer to [216] for details.

The p -delta learning rule has two constituents, the first of which is the traditional delta rule that is employed to modify the weights of a ‘subset’ of the individual neurons constituting the parallel perceptron layer. The next constituent is a rule which determines the subset. This rule states that the delta rule should be applied to those particular neurons which gives either the wrong output or the right output but with a small margin. Next, these two steps of the algorithm are discussed individually.

Obtaining Correct Outputs Let $\mathbf{x}$, $\hat{o}$ and o be the input, output and the desired output respectively. Furthermore, let $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_n$ be the synaptic weight vectors of the n perceptrons. If ϵ is the desired accuracy, then the output is considered correct if $|\hat{o} - o| < \epsilon$. In this case, the weights need not be modified. On the other hand, if $\hat{o} > o + \epsilon$, then the output is larger than expected. Thus, to reduce $\hat{o}$, the number of weight vectors with $\mathbf{w}_i \cdot \mathbf{x} \geq 0$ are to be reduced. Application of the traditional delta rule to such a weight vector will give the update $\mathbf{w}_i \leftarrow \mathbf{w}_i + \eta \Delta_i$ where η is the learning rate and $\Delta_i = -\mathbf{x}$.

Proceeding similarly for the case where $\hat{o} < o - \epsilon$, we can arrive at the

general update rule $\mathbf{w}_i \leftarrow \mathbf{w}_i + \eta \Delta_i$ where Δ_i is given by:

$$\Delta_i = \begin{cases} -\mathbf{x} & \text{if } \hat{o} > o + \epsilon \text{ and } \mathbf{w}_i \cdot \mathbf{x} \geq 0 \\ +\mathbf{x} & \text{if } \hat{o} < o - \epsilon \text{ and } \mathbf{w}_i \cdot \mathbf{x} < 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.5)$$

Output stabilization In the preceding description, the weight vector of a particular perceptron is updated only when its output is incorrect. Hence, after the completion of the training phase, there are usually some weight vectors for which $\mathbf{w}_i \cdot \mathbf{x}$ is very close to 0. This implies a small perturbation of the input $\mathbf{x}$ is capable of changing the sign of $\mathbf{w}_i \cdot \mathbf{x}$ thereby reducing the generalization capabilities and the stability of the network output. In order to stabilize the output, the above rule needs to be modified in such a way that $\mathbf{w}_i \cdot \mathbf{x}$ remains away from 0. Thus, a new parameter γ was introduced as the margin, i.e. the training tries to ensure $|\mathbf{w}_i \cdot \mathbf{x}| > \gamma$.

The final learning rule described in [216] that incorporates all of these concepts is given by:

$$\mathbf{w}_i \leftarrow \mathbf{w}_i + \eta \begin{cases} (-\mathbf{x}) & \text{if } \hat{o} > o + \epsilon \text{ and } \mathbf{w}_i \cdot \mathbf{x} \geq 0 \\ (+\mathbf{x}) & \text{if } \hat{o} < o - \epsilon \text{ and } \mathbf{w}_i \cdot \mathbf{x} < 0 \\ \mu(+\mathbf{x}) & \text{if } \hat{o} \leq o + \epsilon \text{ and } 0 \leq \mathbf{w}_i \cdot \mathbf{x} < \gamma \\ \mu(-\mathbf{x}) & \text{if } \hat{o} \geq o - \epsilon \text{ and } -\gamma < \mathbf{w}_i \cdot \mathbf{x} < 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.6)$$

$$\mathbf{w}_i \leftarrow \mathbf{w}_i / \|\mathbf{w}_i\| \quad (3.7)$$

The parameters involved with p -delta algorithm can be gradually modified with the progress of learning as presented in [216] and implemented in the LSM toolbox described in [244].

3.2.4 Model of Nonlinear Dendrites

Mel and Poirazi [5] showed that neurons with active dendrites i.e. individual dendrites equipped with lumped nonlinearity possess higher storage capacity than their linear counterpart. Such a neuron with nonlinear dendrites (NNLD), depicted in Figure 3.3, has m dendritic branches connected to it with each branch having k excitatory synaptic contact points. If $\mathbf{x}$ is an input vector to this system, then each synapse is excited by any one of the d dimensions of the input vector where $d \gg k$. The output response of j^{th} dendritic branch is calculated as a nonlinear weighted sum of the currents of the k synaptic points connected to the branch and is given by $z_j = b(\sum_{i=1}^d w_{ij}x_i)$, where $b()$ is the dendritic nonlinearity modeled as a nonlinear activation function, $w_{ij} \in \{0, 1, \dots, k\}$ is the weight of the i^{th} input line connected to the j^{th} dendrite and x_i the input arriving at i^{th} input line. w_{ij} is a nonnegative integer denoting the number of times the i^{th} input line is connected to the j^{th} dendritic branch. This can be easily implemented in hardware by using binary synapses exploiting AER protocol as discussed in Chapter 1. The number of synaptic contact points per dendrite are fixed to k by enforcing $\sum_{i=1}^d w_{ij} = k$ for each j . We also define $v_j = \sum_{i=1}^d w_{ij}x_i$ which is the weighted sum of the currents of the k synaptic points of branch j and is input to the branch's dendritic nonlinearity. Combining all the dendritic responses, the overall output $f(\mathbf{x})$ of the NNLD is given by:

$$f(\mathbf{x}) = \sum_{j=1}^m z_j = \sum_{j=1}^m b(v_j) = \sum_{j=1}^m b\left(\sum_{i=1}^d w_{ij}x_i\right). \quad (3.8)$$

where $f()$ denotes the neuronal current-frequency conversion function.

LSMs are typically employed to solve two types of tasks: classification and regression. For both these tasks, we employ two NNLDs and calculate the output by noting the difference of the output of the two NNLDs. The

Figure 3.3: A neuron with nonlinear dendrites (NNLD)

overall output of the circuit is given by

$$y = g[f_+(\mathbf{x}) - f_-(\mathbf{x})] \quad (3.9)$$

3.3 LSM-DER architecture and NRW learning rule

In LSM-DER, the liquid is followed by the two NNLD architecture depicted in Figure 3.4. Here, we denote the output of each cell as $f_{+/-}^{DER}$ with the added superscript DER to contrast with the earlier figure for LSM-PPR. The earlier readout circuit consisting of the parallel perceptrons has now been replaced with the above mentioned circuit. The advantages of this architecture over LSM-PPR [6] are:

- The LSM-DER algorithm attains less error than LSM-PPR for both classification and approximation problems as shown in detail in Section III.
- If there are L liquid neurons, then LSM-PPR required $L \times n$ synapses

Figure 3.4: In LSM-DER, the readout stage following the liquid is composed of two NNLDs. The final readout output $y()$ is obtained by taking the difference of the output of the two cells and passing it through the function $g()$.

for connecting the liquid to the readout whereas LSM-DER can achieve comparable performance with far fewer synapses.

- LSM-PPR requires analog synaptic weights whereas LSM-DER can achieve better performance even with binary synapses.

For training the two-cell architecture, a much simpler and hardware friendly version of the learning algorithm proposed in [5] (discussed in detail in Section 2.2.4) has been employed. The key point of the NRW learning algorithm is removing a synapse which is contributing most to the classification errors and replacing it with a new synapse formed from a different afferent line. Note that, since in LSM the spike train output of the liquid is converted to a continuous valued signal by convolving it with a kernel, error minimization in continuous domain is applicable for the readout. During training the NNLDs, a global teacher signal o is also presented indicating the desired output to be obtained on the application of a particular input sample. If error $e = (o - y)$, then according to the gradient-descent algorithm the weight modification,

Δw_{ij} is given by

$$\begin{aligned}\Delta w_{ij} &= -\frac{\partial e^2}{\partial w_{ij}} \\ &= 2 \langle (o - y) \frac{\partial y}{\partial w_{ij}} \rangle \\ &= 2 \langle (o - y) \frac{\partial g(f_+(x) - f_-(x))}{\partial w_{ij}} \rangle\end{aligned}\tag{3.10}$$

where $\langle . \rangle$ signifies averaging over the entire training set. Hence, the weight modification for the positive and the negative NNLDs are $\Delta w_{ij} = 2 \langle (o - y)g'b'_j x_{ij} \rangle$ and $\Delta w_{ij} = -2 \langle (o - y)g'b'_j x_{ij} \rangle$ respectively where g' and b' are the derivatives of $g()$ and $b()$ respectively. However, we have used binary synapses in this work for robust hardware implementation. This implies our NRW learning procedure does not need any weight modification but requires the formation and elimination of synapses. Hence, we consider a fitness parameter $\phi_{pj} = \Delta w_{ij}$ for the p^{th} synaptic contact point of the j^{th} dendrite where the i^{th} input line is connected to guide this process. Thus, an existing synapse with a low value of ϕ_{pj} needs to be eliminated. The learning procedure dictates the replacement of this poorly performing synapse with a synapse from a randomly chosen replacement set having a high value of ϕ_{pj} . The NRW learning rule thus creates a morphological change of the dendrites guided by the gradient descent rule. From the viewpoint of hardware implementation, we have further simplified the learning rule to use a performance index:

$$c_{pj} = \langle x_{ij} b'_j (o - y) \rangle\tag{3.11}$$

for the positive cell and $c_{pj} = - \langle x_{ij} b'_j (o - y) \rangle$ for the negative cell. This is simpler since it does not require computing derivatives of $g()$ as in [5]. The second difference from [5] is the particular functional form of dendritic nonlinearity $b()$ that we used to simplify c_{pj} further. The dendritic

nonlinearity $b()$ used in our simulations was $b(x) = x^2/x_{thr}$ implying $b'(x) = 2x/x_{thr}$. Hence, by ignoring the constant multiplicative term $2/x_{thr}$, we finally get $c_{pj} = \langle x_{ij}v_j(o - y) \rangle$ for the positive cell. This function can be easily implemented on-chip without needing extra calculations to obtain derivative of $b()$ since v_j is already being computed for normal operation. Though we have not implemented spike based learning in this work, the motivation of choosing this learning rule is to utilize circuits that can compute these correlations [34] on-chip in future.

However, a square law output of each dendritic branch will result in unrealistically large values for large inputs. In earlier work [5], the authors had used nonlinearities like $b(x) = x^{10}$ which also have the same issue and will definitely be a problem in VLSI implementations, either analog or digital. Hence, we included a saturation level, x_{sat} at the output such that for $b(x) > x_{sat}$, $b(x) = x_{sat}$. This models a more realistic scenario in hardware and also leads to power saving in analog hardware. Incidentally, biological neurons also exhibit a saturating nonlinearity [4] due to similar constraints.

The outputs of the liquid were applied as input pattern to the setup of Figure 3.4. Thus, the dimension of the input pattern is now equal to the number of liquid neurons. For each m branches, k synaptic contacts with weight 1 were formed by randomly selecting afferents from one of the $d = L$ input lines, where L is the number of liquid neurons. The learning process comprised the following steps in every iteration:

1. For each pattern of the entire training set, the outputs of both the NNLDs i.e f_+^{DER} and f_-^{DER} were calculated. This was followed by computing the overall response of the classifier as per Equation 3.9.
2. After each application of the entire training set, the Mean Absolute Error (MAE) was calculated as $\sum |(o - y)|/P$, where P is the size of the training set.
3. A random set T of n_T synapses were selected for probable replace-

ment. The performance index c_{pj} corresponding to p^{th} synapse of the j^{th} branch was calculated for each synapse in the set n_T for both the NNLDs.

4. The synapse with the lowest value of performance index (c_{pj}) in T was labeled for replacement with the synapse with the highest value of performance index from an another randomly chosen replacement synapse set R having n_R of the d input lines. The set R was created by placing n_R ‘silent’ synapses from d input lines on the branch with the lowest c_{pj} synapse. They do not contribute to the calculation in step (1).
5. The synaptic connections were updated if the replacement led to a decrease in MAE. If there are no such reduction of MAE, a new replacement set R is chosen. If max_{loc} such choices of R do not reduce MAE, it is assumed that the algorithm has stuck to a local minima and connection changes are made in an attempt to escape the local minima even if it increases the MAE .
6. The above mentioned steps were repeated for a max_{iter} number of iterations after which the algorithm is terminated and the connection corresponding to the best minima is saved as the final connection.

3.3.1 Spike trains as input to Neurons with active dendrites

The output of the liquid neurons are spike trains, while the earlier theoretical description of the NNLDs had inputs $\mathbf{x} \in \mathbb{R}^d$. Thus, to use the NNLDs to act as readout, we need a method to transform spike trains to act as inputs to them. This process is described next.

Figure 3.5: An example of input generation for the proposed readout where the spike train $s(t)$ is first convolved with a kernel representing post-synaptic current to produce $s_a(t)$ and then sampled at a desired temporal resolution to give $s_a^f(t)$.

Suppose, the arbitrary spike train $s(t)$ is given by:

$$s(t) = \sum_{t_f} \delta(t - t_f) \quad (3.12)$$

where t_f indicate the spike firing times. One way to convert this to an analog waveform is to convolve it with a low-pass filtering kernel. We choose a fast rising, slow decaying kernel function, $h(t)$, that mimics post-synaptic current (PSC) waveform and is popularly used in computational neuroscience [56].

The specific form of the function we use is given by:

$$h(t) = I_0(e^{-\frac{t}{\tau_s}} - e^{-\frac{t}{\tau_f}}) \quad (3.13)$$

where τ_f and τ_s denote the fast and slow time constants dictating the rise and fall times respectively and I_0 is a normalizing factor. Hence, the final filtered analog waveform, $s_a(t)$ corresponding to the spike train $s(t)$ is given by:

$$s_a(t) = \sum_{t_f < t} h(t - t_f) \quad (3.14)$$

Finally, to train the NNLDs in the readout, we need a set of discrete numbers which we obtain by sampling $s_a(t)$ at a desired temporal resolution T_s . The sampled waveform, $s_a^f(t)$ is given by:

$$s_a^f(t) = s_a(t) \sum_i \delta(t - iT_s) \quad (3.15)$$

Therefore, for the L spike trains produced by the liquid neurons, if the temporal duration of the spike trains are T , then it will result in a total of $\lfloor T/T_s \rfloor$ samples $\mathbf{x}_i \in \mathbb{R}^L$. For our simulations, we have chosen the temporal resolution $T_s = 25$ ms. The whole process is shown for a spike train output from one random liquid neuron in Figure 3.5.

3.4 Experiments and Results

3.4.1 Problem Description

In this sub-section, we describe the two tasks used to demonstrate the performance of our algorithm. The reason for this choice is that both of these are standard problems that are shown in the original publication on LSM [6] and are included as examples in the LSM toolbox [244]. Also, we chose one

Figure 3.6: Task I is the classification of spike trains. Two different classes of spike trains and the corresponding liquid output when these spike trains are projected into it are shown. Also, a jittered version of a spike train belonging to Class 1 that will be used for testing is depicted.

task to be a classification and the other to be an approximation since they are representative of the class of problems solved by LSM.

Task I: Classification of spike trains

The first benchmark task we have considered is the Spike Train Classification problem [244]. The generalized Spike Train Classification problem includes q arrays of e Poisson spike trains having frequency f and length T_{max} which are labeled as templates 1 to q . These spike trains are used as input to the LSM, and the readout is trained to identify each class. Next, a jittered version of each template is generated by altering each spikes within the template by

a random amount that is stochastically drawn from a Gaussian distribution with zero mean and standard deviation STD . This STD is termed as the *jitter*. Given a jittered version of a particular spike train, the task is to correctly identify the class from which it has been drawn. In this chapter, we have considered $q = 2$, $e = 1$, $f = 20$ Hz, $STD = 4$ ms and $T_{max} = 0.5$ sec. Figure 3.6 shows an example instance of two classes of spike trains and the output of the liquid when jittered versions of these spike trains are injected into it. The figure also shows a jittered input spike train to be used for testing. Equal but different number of patterns are used for training and testing.

Task II: Retrieval of sum of rates

The next task is more difficult than the previous classification problem. The job of the network is to produce at its output the sum of firing rates of the input spike trains averaged over a past time window. e Poisson spike trains are injected into the liquid, the firing rates of which are modulated by a randomly chosen function $A + B\sin(2\pi ft + \alpha)$ normalized into the range $[0, 1]$ to form $r(t)$. The parameters A , B and f were drawn randomly from the following intervals: A $[0$ Hz, 30 Hz] and $[70$ Hz, 100 Hz], B $[0$ Hz, 30 Hz] and $[70$ Hz, 100 Hz], f $[0.5$ Hz, 1 Hz] and $[3$ Hz, 5 Hz]. The phase was fixed at $\alpha = 0$ deg. To generate the test input by a different distribution than the training examples the values of A , B and f in the testing case were kept as 50 Hz, 50 Hz and 2 Hz respectively. The values of these parameters for the testing case were chosen in such a way that they lie in the middle of the gaps between the two intervals used for these parameters during training. At any point of time t , the job of the network is then to give as output the normalized sum of rates averaged over the interval $(t - D - W, t - D)$ where the width of the interval is W and D is the delay. In our case we have taken $e = 4$, $W = 30$ ms and $D = 0$ i.e. no delay. Figure 3.7 shows the 4 input Poisson spike trains, the function $r(t)$, the modulated spike trains and the

liquid output when the modulated spike trains are projected into the liquid. The bottom plot in Figure 3.7 shows the target function i.e. the sum of rates averaged over the last 30ms.

3.4.2 Choice of Parameters

The values of the parameters used by the LSM-DER architecture and NRW learning rule are reported in Table 3.1. We shall next discuss the procedure for selecting these parameters.

Total number of synapses per NNLD (s): The number of synapses required for connecting the liquid to the readout in case of LSM-PPR is $L \times n$ where L and n are the number of liquid neurons and number of perceptrons in the readout stage respectively. To demonstrate that LSM-DER uses synaptic resources efficiently, we employ for LSM-DER the same number of synaptic resources used by LSM-PPR when $n = 1$ i.e. L number of synapses. As described earlier, DER comprises two NNLDs (to eliminate the need for negative weights); thus each has half of the total synapses i.e. $\frac{L}{2}$ synapses are allocated for the positive cell and $\frac{L}{2}$ for the negative cell.

Number of dendrites per NNLD (m): In [5] a measure of the pattern memorization capacity, B_N , of the NNLD (Figure 3.3) has been provided by counting all possible functions realizable as:

$$B_N = \log_2 \left(\binom{k+d-1}{k} + m - 1 \right) \text{bits} \quad (3.16)$$

where m , k and d are the number of dendrites, the number of synapses per dendrites and the dimension of the input respectively for this NNLD. Since the readout of LSM-DER employs two such opponent cells, the overall capacity is twice this value.

Figure 3.7: Task II involves the approximation of a desired function. $r(t)$ is a signal that modulates the 4 input Poisson spike trains. The modulated spike trains are given as input to the liquid and the liquid output is shown. The sum of rates corresponding to the modulated spike trains is also shown

Figure 3.8: The value of B_N is plotted for all possible values of m

In our case $d = L$ and $s = \frac{L}{2}$. Since $s = m \times k$, for a fixed s the all possible values which m can take are the factors of s . Thus, we can calculate the value of B_N for all possible values of m which are shown in Figure 5.4. It is evident from the curve that the capacity is maximum when $m = 14$. But, in our simulations, we found that the learning algorithm cannot train the NNLDs to attain this maximum capacity in a reasonable time due to the huge number of possible wiring configurations to choose from. Hence, as a compromise between capacity and trainability, we chose $m = 7$ in the following simulations.

Number of synapses per branch (k): After s and m have been set, the value of k can be calculated as $k = \frac{s}{m}$.

The slow (τ_s) and fast time constant (τ_f): The fast time constant (τ_f) usually takes a small value in hardware realizations and is not tuned. As for the slow time constant τ_s , if its value is too small, then the post synaptic

current due to individual spikes die down rapidly and thus temporal summation of separated inputs do not take place. On the other hand large values of τ_s renders all spikes effectively simultaneous. So, in both extremes the extraction of temporal features from the liquid output is impaired. A detailed discussion on the selection of τ_s with respect to the inter spike interval (ISI) of the liquid output is given in Section 3.5.

Threshold of nonlinearity: x_{thr} : It can be seen that the value of x_{thr} is different for the two tasks. For selecting x_{thr} , we need to note that the operating principle of the NRW learning rule is to favor those connection topologies where correlated inputs for synaptic connections on the same branch. The nonlinear function $b()$ should give a supralinear output when more than one synaptic inputs are co-activated. For the nonlinear function $b(x) = \frac{x^2}{x_{thr}}$ used here, $b(x) > x$ for $x > x_{thr}$. Hence, the choice of x_{thr} is given by:

$$\overline{I_{syn}} < x_{thr} < 2\overline{I_{syn}} \quad (3.17)$$

where $\overline{I_{syn}}$ denotes the average post-synaptic current from an active synapse. Performing this calculation for a large number of input patterns, we obtained the values of $\overline{I_{syn}} = 1.65$ for Task I and $\overline{I_{syn}} = 5.34$ for Task II. Thus, in our case we chose the value of x_{thr} as 1.8 for Task I and 7 for Task II.

Moreover, an extensive study on the performance of the algorithm due to the variation of x_{sat} will also be provided. For fair comparison, the number of liquid neurons used for both LSM-DER and LSM-PPR was 140. In case of LSM-PPR, the readout stage consists of $n = 40$ neurons which implies the use of 140×40 synapses to connect the liquid neurons to the array of parallel perceptrons. Unless otherwise mentioned, these are the default values of parameters in this chapter.

Choice of $g()$: For Task I, one of the NNLDs was trained to respond to '+' patterns and the other to '-' patterns. Thus, $g()$ in this case is

Table 3.1: Parameter Values

Parameters	Description	Task I	Task II
m	Dendrites per NNLD	7	7
k	Synapses per dendrite	10	10
L	Number of Liquid neurons	140	140
τ_s	Slow time constant	7.5ms	7.5ms
τ_f	Fast time constant	30ms	30ms
n_T	Number of synapses in the target set for probable replacement	15	15
n_R	Number of synapses in the replacement set	25	25
max_{iter}	Maximum number of iterations	1000	1000
max_{loc}	Maximum number of iterations required to declare local minima	30	30
x_{thr}	Threshold of nonlinearity	1.8	7
x_{sat}	Saturation Level	75	75

$signum() : \mathbb{R} \rightarrow \{0, 1\}$ which operates on the combined activity of the two neurons to decide the category (+ or -) of the input pattern. The output of the $signum$ function i.e. y can take a value of either 1 or 0 implying that the pattern belongs to '+' or '-' category respectively. For Task II, $g()$ is given by:

$$g(z) = \frac{1}{1 + \exp(-(z/2))} \quad (3.18)$$

3.4.3 Results: Performance of LSM-DER and NRW algorithm

The proposed readout is separately trained for Task I and Task II. Similar to the method followed in [6, 244], we calculate mean absolute error (MAE) by averaging the error in approximation or classification across all the patterns. We first demonstrate the convergence of the NRW algorithm by plotting the error against iterations for both the tasks in Figure 3.9. The decrease in the error with iterations proves the successful operation of the NRW algorithm in

Figure 3.9: The training error (MAE) obtained by LSM-DER in each iteration is plotted for (a) Task I and (b) Task II when trained on a set of 200 patterns averaged over 10 trials.

finding new suitable connections. Moreover, Fig. 3.10 shows both the target function and the output of the readout stage (during a randomly selected time window) of LSM-DER for a test pattern in Task II. The figure shows that the readout is able to approximate the desired sum of rates very closely.

Similar to [5], it is important to understand the variation in error for

Figure 3.10: The output produced by the LSM-DER after training is superimposed on the target function for a randomly chosen time window and show a very close match.

Figure 3.11: Keeping the number of synapses per dendritic branch (k) of the NNLDs constant at 10, the average MAE over 10 trials is plotted for (a) Task I and (c) Task II when the number of dendrites (m) is varied. Thus, the total number of synapses connecting the reservoir with the readout increases with m and results in a reduction of error. When the total number of synapses for each NNLD ($s/2$) is kept constant at 70, MAE is plotted for (b) Task I and (d) Task II when number of dendrites are increased. In this case, the error initially reduces, reaches a minima and then starts increasing.

both tasks as the architecture of the LSM-DER (characterized by m and k) is varied. We have performed two different experiments to study this dependence. In the first experiment, the number of dendritic branches (m) is varied while keeping the number of synaptic connections per branch (k) at

a constant value of 10 (other parameters are fixed at the values mentioned in table 3.1). This results in an increase in the total number of synaptic connections $s = m \times k$ from the liquid to each readout neuron as m is increased. The results for this experiment are shown in Figure 3.11(a) and (c) for tasks I and II respectively. As expected, the MAE reduces with increasing m since an increase in m with constant k results in more possible functions.

In the second experiment, we vary m while keeping the total number of synapses allocated to a NNLD, s constant at a value of 70. The results for this procedure are plotted in Figure 3.11 (b) and (d) for tasks I and II respectively. The capacity of the readout of LSM-DER (B_N) is also plotted in these figures. The figures show that as m increases B_N first increases and attains a maxima after which B_N decreases. The MAE should have been the lowest when B_N attains the maximum value i.e. at $m = 14$. But, this does not happen in practice and MAE is lowest for $m = 7$. We suspect this is because at $m = 10$ and $m = 14$, the total number of distinct input-output functions that can be implemented by rewiring becomes too large and the NRW algorithm easily gets trapped in several equivalent local minima. We are currently trying to develop better optimization strategies to overcome this issue.

In our next experiment, we have analyzed the performance of LSM-DER with the variation of x_{sat} . Figure 3.12(a) and (b) demonstrates the dependence of MAE on x_{sat} for tasks I and II respectively. As expected, small values of x_{sat} lead to higher error since the branch outputs saturate and cannot encode changes in input. The ideal situation, like [5], is defined when there is no x_{sat} i.e. the dendritic branches produce a square law output without an upper bound and is denoted by the dashed lines in the figure.

(a)

(b)

Figure 3.12: Change in MAE, averaged over 10 trials, when x_{sat} is varied for (a) task I and (b) task II. The dashed lines indicate the MAE for $x_{sat} \rightarrow \infty$.

3.4.4 Results: Comparison between LSM-DER and LSM-PPR

Next, we compare the performance of LSM-DER with LSM-PPR. In Figure 3.13 and Figure 3.14, the profiles of MAE during the training procedure associated with Tasks I and II respectively are shown for LSM-DER and LSM-PPR. It is evident from Figure 3.13 that during Task I the training error for LSM-PPR reduces swiftly and saturates before LSM-DER converges to its minimum error. On the other hand, LSM-DER takes more number of iterations to reach the minimum error but the minimum training error obtained by it is less than that of LSM-PPR. Thus, from the two curves of Figure 3.13 we can mark three significant points:

Figure 3.13: Comparison between MAE vs iterations curve of LSM-DER and LSM-PPR for 200 input patterns averaged over 10 trials for Task I.

1. n_0 : Number of iterations at which the error for LSM-PPR saturates.
2. n_1 : Number of iterations at which the error curve of LSM-DER and LSM-PPR intersects for the first time.
3. n_2 : Number of iterations required by LSM-DER to achieve minimum error.

Figure 3.14: Comparison between MAE vs iterations curve of LSM-DER and LSM-PPR for 200 input patterns averaged over 10 trials for Task II.

Figure 3.15: Convergence Analysis of LSM-DER and LSM-PPR for Task I: The average value of n_0 , n_1 and n_2 obtained for 50, 100 and 200 input patterns averaged over 10 trials.

We have compared the convergence in training for LSM-DER and LSM-PPR when the number of input patterns are 50, 100 and 200. The average values n_0 , n_1 and n_2 obtained for the three cases are depicted in Figure 3.15 for Task I. From this figure, we can see that $n_0 < n_2$ for all cases implying faster convergence for LSM-PPR. However, we can note that $n_0 \approx n_1$ implying LSM-DER can always achieve same error as LSM-PPR in roughly similar number of iterations; it takes more time for LSM-DER to find *better* solutions.

The earlier plots suggest LSM-DER can attain better error in training—however, error during testing is more important to show the ability of the system to generalize. These generalization plots depicted in Figure 3.16 and Figure 3.17 for Tasks I and II respectively show that for each case, LSM-DER outperforms LSM-PPR for different number of training patterns.

LSM-PPR also requires the setting of another parameter: n denoting the number of readout neurons. Till now, LSM-DER has been compared with LSM-PPR keeping $n = 40$. Next, we vary the number of readout neurons of

Figure 3.16: Generalization Plot: A performance comparison of the training and testing error of LSM-DER and LSM-PPR on Task I for 50, 100 and 200 input patterns averaged over 10 trials

LSM-PPR and compare the results with LSM-DER. The outcome of these experiments are shown in Figure 3.18 and Figure 3.19. In both the figures, the training and testing error of LSM-DER is plotted as a constant line and is compared with the results of LSM-PPR for different values of n . During this experiment, we chose the values of n as 1, 10, 20, 30, 40, 50 and 60 thereby covering the most basic $n = 1$ case to advanced cases employing more number of readout neurons. Note that the synaptic resource consumed by LSM-DER is same as the case for $n = 1$ or a single perceptron. As evident from the figures, the error attained by LSM-PPR decreases with the increase in value of n but finally becomes saturated at a value higher than the error attained by LSM-DER. The graph also shows that the saturation starts approximately when $n = 40$ and this also explains why we have chosen this value of n while comparing our LSM-DER algorithm to LSM-PPR. From these plots, we can conclude that LSM-DER attains 3.3 and 2.4 times less error than LSM-PPR with same number of high resolution weights in tasks I and II respectively. Also, LSM-PPR requires 40 – 60 times more synapses to achieve similar performance as LSM-DER in Task II.

Till now, for our simulations the number of liquid neurons L was kept as 140. Now, we will look into the effect of increasing liquid size on the

Figure 3.17: Generalization Plot: A performance comparison of the training and testing error of LSM-DER and LSM-PPR on Task II for 50, 100 and 200 input patterns averaged over 10 trials

Figure 3.18: Performance Comparison of LSM-DER and LSM-PPR with varying n for Task I (MAE averaged over 10 trials): The number of readout neurons i.e. n is a parameter of LSM-PP. We have obtained the results of LSM-PPR for different values of n . The classification error gradually decreases with increasing n and finally it saturates. The performance of LSM-PPR for different values of n are compared with the results obtained by LSM-DER.

performance and convergence of LSM-DER and LSM-PPR for Task I. For $L = 560$, LSM-DER provides a testing error of 0.079 and $n_2 = 1472$. On the other hand for $L = 560$, LSM-PPR provides a testing error of 0.129 and the average number of iterations it takes to saturate while training is (n_0) is 628. For $L = 1120$ LSM-DER and LSM-PPR provides testing error of 0.076 and

Figure 3.19: Performance Comparison of LSM-DER and LSM-PPR with varying n for Task II (MAE averaged over 10 trials): The number of readout neurons i.e. n is a parameter of LSM-PP. We have obtained the results of LSM-PPR for different values of n . The error gradually decreases with increasing n and finally it saturates. The performance of LSM-PP for different values of n are compared with the results obtained by LSM-DER.

0.12 respectively. In this case, n_2 for LSM-DER is 2402 and n_0 for LSM-PPR is 914. Hence, we see that for higher dimensional inputs, the NRW rule is still able to find suitable connections but requires a larger number of iterations.

3.5 Discussion of software results

It is surprising that even after using 140×40 synapses, LSM-PPR cannot attain the performance of LSM-DER that utilizes only 140 synapses. To verify if this is indeed an improvement, we need to check if LSM-PPR is really using up all the allocated synapses. Thus we provide a histogram in Figure 3.20 which shows the final weight distribution of the synapses for Task I when trained on 200 patterns. From the figure it is clear that most of the weights are non-zero thereby confirming that a large portion of the 140×40 synapses are indeed active and being used. This is surprising since the function counting method (used to predict the capacity of NNLD) would predict a large capacity for the parallel perceptron case as well. Essentially, each of the per-

Figure 3.20: Histogram showing the distribution of weights of the synapses in the readout for LSM-PPR. It can be noted that most of the synapses have a non-zero weight implying that most of the synapses are active and add to resource consumption.

ceptrons behave akin to a dendrite—in fact, they have more number of analog weights and hence better capacity. The differences between the DER and PPR readout are in the threshold nonlinearity and the training algorithm. Also, the sampled inputs $s'_a(t)$ for this stage are derived through a different convolution kernel. To tease apart the contribution of each difference to the poor performance, we consider two separate cases for training LSM-PPR on Task I. In case I, we use $n = 14$ but instead of the threshold nonlinearity, we use a saturating square non-linearity similar to LSM-DER. In this case, LSM-PPR is able to reduce the average MAE from 0.2169 to 0.1331 indicating the advantage of preserving some analog information at the output of each perceptron. In case II, we explore the impact of different states $s'_a(t)$. We keep $n = 1$ for LSM-PPR and use our convolution kernel instead of the one in the LSM toolbox. In this case, the average MAE reduces to 0.1360 showing

the importance of choosing the kernel carefully. Next, combining both these modification, LSM-PPR trained by p -delta could achieve an average MAE of 0.0706 that is comparable to LSM-DER. This demonstrates that when PPR is equipped with only the stronger components of DER (better nonlinearity and convolutional kernel) but not its weaker components (binary synapses), PPR still gives similar performance as DER thereby showing the superiority of our learning rule.

Given the importance of the convolution kernel for generating $s'_a(t)$ as described earlier, we now provide some insight to its choice. The state generation method discussed in Section 3.3.1 requires two parameters : τ_s and τ_f . τ_f is the fast time constant which takes a small positive value in hardware implementations and is typically not tuned. τ_s is the slow time constant responsible for integration across temporally correlated spikes and we will analyze its effect on the performance of LSM-DER. If there are L liquid neurons and the mean firing rate of the each liquid neuron is μ_f , then the mean ISI across the entire liquid output is given by $N = 1/(L \times \mu_f)$. Intuitively, we expect $\tau_{s,opt}$, the optimal value of τ_s , to be correlated with this quantity since it has to be long enough to integrate information of temporally correlated spikes across all the liquid neurons. In Fig.3.21(a), the MAE attained by LSM-DER for Task I is shown for different values of τ_s when N is varied by changing the value of L . A similar result is obtained by changing the value of μ_f keeping L as constant and is not shown here to avoid repetition. From this figure, we see a strong correlation between N and the best value of τ_s , $\tau_{s,opt}$. A similar set of simulations are also done for Task II. The best τ_s is then chosen for each value of N and plotted in Fig.3.21(b). It can be seen that for both tasks and with N spanning two orders of magnitude, $\tau_{s,opt}$ can be well described by the fit $\tau_{s,opt} = 52.83N - 3.1$.

Figure 3.21: (a) MAE for Task I is reported by varying τ_s and inter spike interval of the liquid output N . (b) A curve fit to the optimal τ_s for each N is obtained by varying N for both tasks over two orders of magnitude.

3.6 VLSI Implementation for off-chip learning and on-chip testing: Effect of Statistical Variations

This section contains the description of a VLSI architecture to implement DER and PPR, Monte Carlo simulation results of the key sub-circuits to show their statistical variability and incorporation of these statistical variations in Matlab analytical models to analyze the stability of the algorithms. These circuits are designed in AMS 0.35 μm CMOS technology.

3.6.1 VLSI Architecture of the Spiking Neural Network

The VLSI architectures for implementing DER and PPR readouts for LSM are shown in Figure 3.22(a) and (b) respectively. For both cases, AER is used to provide the synaptic connections. For DER, there is one shared synapse for every dendritic branch while for PPR, there is one shared synapse per perceptron. The input spikes (output of the liquid) are applied to the circuit through an address decoder while Differential Pair Integrator (DPI) circuits

Figure 3.22: Architecture for VLSI implementation of (a) DER and (b) PPR readout schemes for LSM. Schematic diagram of DPI Synapse and squaring block used in these architectures are shown in detail in (c) and (d) respectively.

are used to implement synaptic function. For the DER case, there are m dendritic branches connected to a **NEURON** block through m Square Law Nonlinear circuits. The output of the spiking neuron can be converted to the analog output y by considering the spike rate averaged over a pre-defined time period. For the case of classification, a winner-take-all circuit can be used instead of two neurons. The VLSI architecture for PPR with $\frac{m}{2}$ perceptrons is the combination of $\frac{m}{2}$ DPI circuits with current source outputs (for positive weights) and $\frac{m}{2}$ with current sink outputs (for negative weights). For each perceptron, if the current from the positive weight DPI is higher, the output voltage gets pulled high and vice versa. These voltages are the inputs to the **N-SUM** block that computes a sum of these voltages (perceptron outputs) and passes it through a nonlinear function $g()$ to obtain $\hat{o}$ as presented in Eqn. 3.4. Compared to the **NEURON** or **N-SUM** blocks, the synapses and square law blocks are more numerous. Hence, their individual sizes are

also kept small leading to them being more susceptible to variations. Next, we shall briefly describe circuit implementations of these blocks and show simulations of the effect of statistical variations.

Differential Pair Integrator Synapse

The circuit schematic shown in Figure 3.22(c) is a DPI circuit that converts pre-synaptic voltage pulses into post-synaptic currents ($I_{syn}(t)$). In this circuit, transistors operate in subthreshold regime. As mentioned in [245], controlling the bias voltages V_w and V_{tau} we can set $I_w \gg I_\tau$, which simplifies this nonlinear circuit to a canonical first-order low pass filter. The bias voltages V_{thr} , V_w can effectively control the weight of the synapse (maximum output synaptic current, denoted by I_0) and V_{tau} controls the fall time constant (τ_s) of the output current. For an input spike arriving at t_i^- and ending at t_i^+ to the DPI synapse, the rise time of I_{syn} is very small. The discharge profile can be modeled by the Equation 3.19.

$$I_{syn}(t) = I_0 e^{(-\frac{t-t_i^+}{\tau_s})} \quad (3.19)$$

where $\tau_s = \frac{C_{SYN} U_T}{\kappa I_\tau}$, κ is the subthreshold slope factor, and U_T is the thermal voltage.

Table 3.2: Monte Carlo Simulation Results of DPI Synapse

$\mu[I_0]$	$\sigma[I_0]$	$\frac{\sigma}{\mu}[I_0]$	$\mu[\tau_s]$	$\sigma[\tau_s]$	$\frac{\sigma}{\mu}[\tau_s]$
4.21 nA	570 pA	13 %	912.5 μs	91.9 μs	10.1 %
1.06 nA	127 pA	12 %	903.1 μs	90.7 μs	10.0 %
7.74 nA	875 pA	11 %	4116 μs	413.6 μs	10.1 %
1.06 nA	127 pA	12 %	18335.7 μs	1832.7 μs	10.0 %

Saturating Square Law Nonlinear circuit

We have designed the current mode squaring circuit given in Figure 3.22(d) as described in [220]. Transistors M_{S2} , M_{S1} , M_{S3} and M_{S5} form a translinear loop. Hence, the current through M_{S5} is expressed as given in Equation 3.20. The transistor M_{S5} is biased to pass a maximum current of I_{sat} (set by V_{BSAT}).

$$I_{out} = \frac{I_{in}^2}{I_{thr}} \quad (3.20)$$

I_{thr} is the dc current through M_{S4} set by its Gate voltage (V_{BTHR}). Due to process parameter mismatch between the transistors M_{S2} , M_{S1} , M_{S3} and M_{S5} , the output current deviates from the exact relationship given in Equation 3.20. Since variation of threshold voltage (ΔV_{th}) dominates other sources of variation in the subthreshold regime, the translinear loop equation for Figure 3.22(d) can be re-written as

$$I'_{out} = e^{\frac{(\Delta V_{th1} + \Delta V_{th2} - \Delta V_{th5} - \Delta V_{th3})}{U_T}} \times \frac{I_{in}^2}{I_{thr}} \quad (3.21)$$

For the sake of modeling this multiplicative mismatch, the nonideality term $e^{\frac{(\Delta V_{th1} + \Delta V_{th2} - \Delta V_{th5} - \Delta V_{th3})}{U_T}}$ in the Equation 3.21 is denoted by a parameter c_{ni} , and hence the output of the square block can be written as $I'_{out} = c_{ni} \times I_{out}$.

3.6.2 Monte Carlo Simulation Results

We have performed Monte-Carlo simulation of the DPI synapse and Square Law Nonlinear circuits considering transistor mismatch. The objective of the Monte-Carlo simulation is to capture the variation of I_0 , τ_s , c_{ni} from one dendritic branch (or perceptron) to the other. Some of the representative results of the statistical simulation are listed in Table 3.2 and 4.1. For dif-

Figure 3.23: Stability of LSM-DER and LSM-PPR with respect to different hardware non-idealities are plotted for Task I in (a) and (b) respectively. The constant red line indicates the testing error obtained by the algorithms without any non-ideality. The bars represent the error obtained after inclusion of non-idealities. The rightmost bar marked by (...) represents the error when all the non-idealities are included simultaneously. The MAE is averaged over 10 trials.

ferent settings of variable bias parameters (V_{thr} , V_{tau} , V_w and V_{BTHR}), we obtained worst case variation of I_0 and I'_{out} as 13% and 18% respectively. The parameter $\frac{\sigma}{\mu}[I'_{OUT}]$ given in Table 4.1 can also be written as $\frac{\sigma}{\mu}[c_{ni}]$.

Table 3.3: Monte Carlo Simulation Results of Square Law circuit

I_{thr}	I_{IN}	$\mu[I'_{OUT}]$	$\sigma[I'_{OUT}]$	$\frac{\sigma}{\mu}[I'_{OUT}] = \frac{\sigma}{\mu}[c_{ni}]$
30 nA	20 nA	12.15 nA	2.20 nA	18.0 %
30 nA	60 nA	99.70 nA	13.6 pA	14.0 %
30 nA	100 nA	243.1 nA	26.4 pA	11.1 %

The impact of global process variation on circuit performance has been neglected here, because performance drift due to global variations can be eliminated by tuning the bias parameters. The device sizing for the circuit blocks are given in Table 3.4 for reference.

Table 3.4: Design parameters of the Circuit Blocks in Figure 3.22

Synapse Block in Figure 3.22 (c)	Square Block in Figure 3.22 (d)
$\frac{W}{L} (M_{d1}, M_{d2}, M_{d3}) \equiv \frac{8\mu}{1\mu},$ $\frac{W}{L} (M_{d4}) \equiv \frac{0.40\mu}{0.35\mu}, \frac{W}{L} (M_{d5}) \equiv \frac{10\mu}{1\mu}$ $C_{syn} = 1.1pF$	$\frac{W}{L} (M_{s1}, M_{s2}) \equiv \frac{8\mu}{2\mu},$ $\frac{W}{L} (M_{s3}, M_{s5}) \equiv \frac{8\mu}{2\mu},$ $\frac{W}{L} (M_{s4}, M_{s6}) \equiv \frac{16\mu}{2\mu}$

3.6.3 Stability Analysis in Software Simulations

To analyze the stability of the algorithms, the statistical variations described above are incorporated during the testing phase of the simulation. The non-idealities are included only in the testing phase (and not during the training phase) because in the actual implementation, the training will be done in software and the trained connections will be downloaded directly to the chip. Figure 3.23 shows the performance of both LSM-DER and LSM-PPR when the non-idealities are included for Task I. Since all the variations are across branch, so for fair comparison the PPR architecture used in this analysis has the same number of dendrites as that of the LSM-DER structure used. Moreover, we have used the same convolution kernel for state generation in both cases. In Figure 3.23, the bars corresponding to τ_s , c_{ni} , and I_0 denote the performance degradation when statistical variations of τ_s , c_{ni} and I_0 are included individually. Finally, to imitate the true scenario we consider the simultaneous implementations of all the non-idealities, which is marked by (...). In the software simulations, the $\frac{\sigma}{\mu}$ of τ_s and I_0 has been taken to be the worst case scenario as displayed in Table 3.2 i.e. 10.1% and 13% respectively. Similarly, the $\frac{\sigma}{\mu}$ of c_{ni} in the software simulations has been taken to be the worst case scenario as portrayed in Table 4.1 i.e. 18%. From Figure 3.23 it is evident that when all the variations are included, then the MAE of LSM-DER and LSM-PPR increases by 0.0233 and 0.0470 respectively. This concludes that the modifying connections of binary synapses in LSM-DER results in more robust VLSI implementations compared to the adaptation of high resolution weights LSM-PPR.

3.7 Architectural Exploration and learning rule modification for on-chip online learning

It is a long standing debate in Machine Learning domain [246] on whether to train a network by batch or online learning. Traditionally online learning is chosen over batch learning when faster training of the network is required. But, we have slightly different motivations for considering online learning. Firstly, since we consider on-chip learning, to perform batch training of the neural network implemented in the chip, the entire data set has to be stored in the chip itself. This requires a lot of storage space. On the other hand if online learning is done, storage of ‘mini-batches’ of the entire data set will suffice. Secondly, batch learning requires averaging over the entire data set to find the ‘true gradient’, but since the hardware implementation of the neural network is done by analog circuits, so averaging over the entire data set is not possible due to time constant limitations. In that case, the ‘true gradient’ needs to be found on the part of the data set which the averaging circuit is able to average on. Thirdly, since these neural network chips will be used as embedded sensors in real world scenarios where the input data distribution is changing over time, online learning, unlike batch learning, is capable of tracking the changes in the data and providing good approximation results.

In this section, we will focus on developing low-overhead architectures for on-chip implementation of Dendritically Enhanced Readout(DER) facilitating on-chip training and will also propose an online variant of the NRW rule for real-time training.

3.7.1 Architectural modification of DER for on-chip learning

Architecture I

First, let us consider the hardware requirements for on-chip learning using the architecture described in Section 3.3. Since we randomly select any of the $m \times k$ number of synapses to be in T and use the corresponding c_{pj} , we need to have provision in hardware for calculating c_{pj} for all the synapses in the positive and the negative cells. After labelling the minimum synapse for replacement (using loser-take-all (LTA) blocks for comparing c_{pj}), the replacement set R has to be formed by placing nR ‘silent’ synapses on the branch with the minimum c_{pj} synapse. Since we do not know a priori which dendrite will have the minimum c_{pj} synapse, each branch has to be equipped with nR silent synapses with their own c_{pj} calculator and a winner-take-all (WTA) is used to find the one with maximum c_{pj} . Thus, the number of c_{pj} calculators required are $N_c = (k \times m + nR \times m)$ per cell.

Architecture II

As a first step to reduce the number of c_{pj} calculators, we note that only n_T of the $m \times k$ c_{pj} are used in any step. Hence, to form the target set T , we create a set of ‘Copy synapses’ denoted by C per cell by keeping one new synapse equipped with a c_{pj} calculator on each dendrite. It is formed by randomly taking one synapse from each dendrite of the cell. Thus, $n_T = m$ in this case. Even though we marginally increase the number of synapse circuits, we drastically reduce number of c_{pj} calculators. The input to the copy synapse on one branch (entries in an off-chip SRAM storing AER connections [247]) is same as one of the existing synapse inputs on that branch. The formation of the silent synapse set is same as the previous architecture. Thus, the number of c_{pj} calculators required per cell is $N_c = (m + nR \times m)$. The architecture of the positive cell is shown in Figure 3.24.

Figure 3.24: Block diagram of Architecture II

Architecture III

As an immediate next step we try to reduce the number of c_{pj} calculators utilized by the silent synapses. The reason behind the usage of $nR \times m$ c_{pj} calculators per cell was that we did not know beforehand on which dendrite the nR silent synapses were to be placed. Two methods can be employed to circumvent this problem. In Method I, instead of keeping the silent synapses in the dendrite containing the minimum c_{pj} synapse for the current epoch (presentation of all the input patterns), we place them in the dendrite containing the minimum c_{pj} synapse for the previous epoch. This technique ensures that we have prior knowledge of where to place the silent synapses and so we would only need nR number of silent synapses per cell each having a c_{pj} calculator. Thus, now we only need $N_c = m + nR$ number of c_{pj} calculator circuits per cell. This architecture of the positive cell is portrayed in Figure 3.25.

Figure 3.25: Block diagram of Architecture III

Figure 3.26: Block diagram of Architecture IV

Table 3.5: Comparison of DER architectures

Arch.	N_c	MAE	Remarks
I	$k \times m + nR \times m$ (119)	0.0912	Least MAE= MAE_{min} , Highest $N_c = N_{c,max}$
II	$m + nR \times m$ (56)	0.0946	$MAE \approx MAE_{min}$, $N_c < N_{c,max}$
III	$m + nR$ (14)	0.1088	$MAE = 1.1930 MAE_{min}$, $N_c \ll N_{c,max}$
IV	$k + nR$ (17)	0.1094	$MAE = 1.1996 MAE_{min}$, $N_c \ll N_{c,max}$

Architecture IV

In Method II for reduction of silent synapse calculators, we place the silent synapses randomly on any one of the dendrites. In this case, the target set formation technique used for Architecture II cannot be used. The Target Set T in this method comprises of all the synapses present in the randomly chosen dendrite for silent synapse placement making $n_T = k$. So, for each cell we require k and nR c_{pj} calculators for the copy synapse and the silent synapse set respectively. The architecture of the positive cell is portrayed in Figure 3.26.

Software simulations have been done for each of the above four architectures for the Spike Train Classification task [6, 248] and the average Mean Absolute Error (MAE) for 10 trials has been provided in Table 3.5. From Table 3.5, it is evident that Architecture III gives a good trade off between performance and hardware overhead by using 8.5 times less c_{pj} calculators than Architecture I while hurting performance by only 1.76%.

Table 3.5 summarizes the key points of the above four architectures.

3.7.2 Online NRW learning rule

Here, we present a modified NRW learning rule suitable for online learning of P patterns. One practical constraint when the c_{pj} calculator is implemented

using analog circuits is that the time duration t_{sp} over which the averaging circuit operates is of the order of milliseconds, which may even be smaller than the duration T_p of a single pattern encountered in real world pattern sets. Hence, we can never find maximum or minimum of $c_{pj} = \text{sgn}(t - y) \int_0^{PT_p} (x_i b'_j)$. Our proposed rule takes into account such cases as well.

Figure 3.27: Time line depicting the online NRW learning rule

1. After presentation of P_{sub} number of patterns of the entire data set P , a set L_{sub} is formed by choosing the input dimensions where at least one spike has occurred. The initial connection of both the cells of DER are initialized from the set L_{sub} .
2. The Target Set T and Replacement Set R is initialized from the set L_{sub} .
3. A sub-Target Set T_{sub} and a sub-Replacement Set R_{sub} (which are subsets of T and R respectively) is formed for each cell every t_{sp} seconds. T_{sub} and R_{sub} comprises of the entries of T and R respectively which received a non-zero inputs in the last t_{sp} seconds.
4. The averaging circuit finds $c'_{pj} = \text{sgn}(t - y) \int_0^{t_{sp}} (x_i b'_j)$ for the synapses in T_{sub} and R_{sub} —an estimate of the true c_{pj} used in the earlier algorithm. The WTA and LTA circuits calculate the minimum c_{pj} synapse from T_{sub} and maximum c_{pj} synapse from R_{sub} respectively. This step has a

loss in information since we no longer have information to get the true c_{pj} .

5. To make up for information lost in the earlier step, we introduce a voting mechanism to find most probable maxima and minima. After presentation of P_{conn} number of patterns, the synapse that has been selected as the minimum for the most number of times is tagged as the candidate for replacement. This synapse is replaced by the silent synapse which had maximum c_{pj} value for most number of times. Thus the network rewiring learning takes place after P_{conn} number of input patterns have been presented. The MAE is checked for the next P_{MAE} patterns and if the average P_{MAE} does not decrease, the connections are switched back.
6. Steps 3-6 are repeated max_{iter} number of times after which the algorithm is terminated and the connection corresponding to the best performance is saved as the final connection.

Figure 3.27 provides a visual representation of the learning mechanism. In the extreme case when $P_{conn} = P$, the connection changes are made after the presentation of the entire batch. When P_{conn} is large, we have more information available to make better decisions about the replacement leading to more stable convergence of the algorithm. However, the connection changes in this case are less frequent and so the training becomes slow. On the other hand, when P_{conn} is small the frequency of connection changes increase and so the training is fast at the cost of convergence issues. This trend can be observed in Figure 3.28 where the convergence curves of the learning rule, averaged over 10 trials, has been shown for different values of P_{conn} with fixed P_{MAE} . We have also obtained the MAE for different values of P_{MAE} keeping the value of P_{conn} constant at $\frac{P}{4}$. The average MAE over 10 trials when P_{MAE} has been kept as $\frac{P}{8}$, $\frac{P}{4}$ and $\frac{P}{2}$ are 0.1317, 0.1127 and 0.1098 respectively. These results show the characteristic tradeoffs involved in online

Figure 3.28: Setting $P_{MAE} = \frac{P}{4}$, the convergence curves have been shown for different values of P_{conn} .

learning. When $P_{conn} = P$ and $P_{MAE} = P$, the MAE obtained is 0.1075 i.e. we have achieved similar performance as compared to batch learning, though we have used an averaging circuit with 4200 times lesser time constant. Our voting mechanism has artificially stretched the time constant of the circuit thereby making this possible.

3.8 Summary

In this chapter, we have proposed a novel architecture (LSM-DER) and an efficient learning rule (NRW) for the readout stage of Liquid State Machine. Inspired by the nonlinear properties of dendrites in biological neurons, the readout neurons of LSM-DER employ multiple dendrites with lumped nonlinearities. The results depict that LSM-DER along with NRW is able to achieve better performance (3.3X less error in classification and 2.4X less error in approximation) than the state-of-the-art LSM-PPR with same number of synapses thus making it suitable for VLSI implementations. More-

over, unlike LSM-PPR which uses high precision analog synaptic weights, LSM-DER uses binary synapses, thus being very advantageous for hardware implementations. Since the synapses are binary valued, the NRW learning rule chooses the best possible connection matrix between inputs and the dendritic branches. Also, we have shown that the proposed architecture is robust against statistical variation of parameters, a feature essential for VLSI implementations. Furthermore, we have looked into various strategies which can be employed for on-chip implementation of DER for on-chip online training. We have tried to find an architecture which is capable of optimizing both area and power, while maintaining acceptable performance. In this search we have devised an optimized architecture, termed in this chapter as Architecture III, which uses 8.5 times less resources than the non-optimized architecture (Architecture I) while having a performance degradation of only 1.76%. We have also devised a novel learning rule for DER which is able to train the DER network for online learning. Although this learning rule uses an averaging circuit with 4200 times lesser time constant rather than its batch counterpart, still it provides similar MAE due to our unique voting mechanism. In the next chapter we will propose another morphological learning mechanism for NNLD based on the Tempotron learning rule which directly classifies high dimensional spatio-temporal spike trains. Compared to the work (LSM-DER) proposed in this chapter, the subsequent learning rule is optimized for memory capacity.

Chapter 4

Morphology Optimizing Tempotron

In this chapter, a neuron with nonlinear dendrites (NNLD) and binary synapses that is able to learn temporal features of spike input patterns is considered. Since binary synapses are considered, learning happens through formation and elimination of connections between the inputs and the dendritic branches to modify the structure or “morphology” of the NNLD. A morphological learning algorithm inspired by the ‘Tempotron’, i.e., a recently proposed temporal learning algorithm-is presented in this work. Unlike ‘Tempotron’, the proposed learning rule uses a technique to automatically adapt the NNLD threshold during training. Experimental results indicate that our NNLD with 1-bit synapses can obtain similar accuracy as a traditional Tempotron with 4-bit synapses in classifying single spike random latency and pair-wise synchrony patterns. Hence, the proposed method is better suited for robust hardware implementation in the presence of statistical variations. We also present results of applying this rule to real life spike classification problems from the field of tactile sensing.

4.1 Introduction

Though the representation of stimulus by the neurons in our brain is a topic of much ongoing research and debate, it is widely believed that the timing of the action potentials or spikes fired by these neurons carry important information [56]. Spike latency codes i.e. delays in the spike time after stimulus presentation have been suggested for tactile, olfactory and retinal systems [249]. They are also thought to offer significant advantages in terms of reducing power needed for communicating spikes as well as allowing rapid processing of inputs. Hence, neuromorphic engineers, who aim to mimic the brain's processing capabilities in silicon, have also been interested in spike timing based neural networks.

In this chapter, a hardware friendly morphological learning rule to train a NNLD with binary synapses for classifying spatiotemporal high dimensional spike patterns is presented. Previous studies have shown that NNLD can be successfully applied to learn both mean rate encoded inputs [5, 220] and spike-timing information [247](described in Chapter 3). Although in Chapter 3 we have proposed a spike time based learning rule for training a NNLD architecture, however the rule was not optimized for memory capacity. Here we propose a novel memory capacity optimized spike-timing-based learning rule for training NNLD. Our work is inspired by the Tempotron learning rule for spiking neurons [56]. However, unlike the Tempotron learning rule that requires weights with high resolution, the proposed network uses low-resolution non-negative integer weights and learns through modifying connections of inputs to dendritic branches. As presented in Chapter 1, this can be easily implemented in hardware by using binary synapses following the AER communication protocol. Furthermore, the spiking threshold of the neuron employed in Tempotron is fixed throughout the learning. On the other hand, the proposed method is equipped with a threshold adaptation mechanism trying to optimize the number of false positives and false negatives.

The organization of this chapter is as follows: a brief description and previous work on Tempotron is presented in Section 4.2. In Section 4.3 the architecture of the classifier and the morphological learning algorithm for training it are introduced. Two spike time based binary classification tasks are discussed in Section 4.4.1 and performance of our method in comparison to Tempotron is shown in Section 4.4.2, Section 4.4.3 and Section 4.5. Finally, our work is compared with other classifiers in Section 4.6 and then a summary of this chapter is provided in Section 4.7.

4.2 Background

In this section, for the sake of completeness, we provide a brief description of the Tempotron learning rule. This is followed by a short review of the research works focusing on it.

4.2.1 Theory of Tempotron

The tempotron, proposed by Gütig and Sompolinsky in [56], is a simple biologically plausible supervised learning rule for training a LIF neuron. The learning rule involves modifying the synaptic weights associated with the neuron such that the trained neuron produces a spike for one class of patterns while remaining silent for the other. Thus, the tempotron rule trains a LIF neuron to be a binary classifier of spatio-temporal spike patterns. The two core assumptions of this rule are : 1) before the presentation of each input pattern, the membrane voltage of the LIF neuron is at resting potential and 2) after the neuron produces a spike, the incoming spikes of the current pattern have no further effect i.e. even if the neuron would produce more than one spike, the remaining ones following the first spike are synthetically removed.

A basic diagram of the neuron is shown in Figure 4.1. The d -dimensional input patterns of duration T secs are provided to the neuron through d

Figure 4.1: A LIF neuron to be trained by Tempotron learning rule is shown. Depending on the time of arrival of the incoming spikes, the synapses get activated and generate post-synaptic potentials (PSPs). The membrane voltage, computed by taking a weighted linear sum of these PSPs, is given by:

$$V(t) = V_{rest} + \sum_{i=1}^d w_i \sum_{\forall g \text{ s.t. } t_g^i < t} K(t - t_g^i) \quad (4.1)$$

where w_i is the weight of the i^{th} input line, K denotes the post-synaptic potential kernel and t_g^i (g is the spike index) are times of incoming spikes on the i^{th} afferent. In Tempotron [56] the synaptic kernel is a normalized PSP of the form:

$$K(t - t_g) = V_0(\exp[-(t - t_g)/\tau_s] - \exp[-(t - t_g)/\tau_f]) \quad (4.2)$$

where the parameters τ_s and $\tau_f = \tau_s/4$ denote the decay time constants of membrane integration and synaptic current respectively. Moreover V_{rest} is taken to be 0 which reduces Equation 4.1 to:

$$V(t) = \sum_{i=1}^d w_i \sum_{\forall g \text{ s.t. } t_g^i < t} K(t - t_g^i) \quad (4.3)$$

For a given pattern if the membrane voltage $V(t)$ crosses the neuronal firing threshold V_{thr} , a single spike is emitted and V_{out} is interpreted to take a value of 1. When the neuron does not produce a spike, V_{out} is set to be 0.

In tempotron learning rule a cost function measuring the deviation of the maximum membrane voltage (V_{max}) from V_{thr} for misclassified patterns is minimized. It is given by:

$$E = \begin{cases} V_{thr} - V_{max}, & \text{if pattern in } P^+ \text{ is presented} \\ V_{max} - V_{thr}, & \text{if pattern in } P^- \text{ is presented} \end{cases} \quad (4.4)$$

where V_{max} is the maximal value of post-synaptic potential $V(t)$ at the time t_{max} , i.e., $V_{max} = V(t_{max})$ and P^+ and P^- are the two classes of patterns. Application of gradient descent rule to the space of synaptic weights for minimizing E yields that for misclassified patterns the synaptic weights should be updated according to the following rule:

$$\Delta w_i = \begin{cases} \lambda \sum_{\forall g \text{ s.t. } t_g^i < t_{max}} K(t_{max} - t_g^i), & \text{if pattern in } P^+ \text{ is presented} \\ -\lambda \sum_{\forall g \text{ s.t. } t_g^i < t_{max}} K(t_{max} - t_g^i), & \text{if pattern in } P^- \text{ is presented} \end{cases} \quad (4.5)$$

where $\lambda > 0$ is the learning rate and w_i is updated only for misclassified patterns.

4.2.2 Previous research on Tempotron

After its proposition in 2006 [56], Tempotron has been widely studied and used by spiking neural network researchers in the last few years from various viewpoints. First, we throw some light on the works which have proposed modifications of the Tempotron learning rule. In [250], Ambard et al. have proposed a modified Tempotron learning rule having better generalization

performance. They have termed the modified rule as “Voltage-Margin Tempotron”, which introduces a margin around the LIF threshold during training. The margin forces the different class of patterns to move away from the threshold thereby increasing the separation between them. The margin is gradually increased during learning until a larger margin cannot be crossed anymore. Rotman [251] et al. have studied the performance of Tempotron in presence of synaptic unreliability and short-term dynamics as observed in biological neural networks. They have discovered that although Tempotron is capable of learning spike time codes in absence of synaptic reliability, it provides a significantly reduced performance. They have looked into possible strategies and proposed that using clustered spike bursts instead of a single spike transmission significantly increases classification performance. Moreover, they have shown that using dynamic synapses instead of static synapses can further improve classification performance. Finally, they have shown that performance can be enhanced even further by using a mixture of facilitating and depressing synapses. In Tempotron, learning happens through online minimization of a cost function in the space of synaptic efficacies. Inspired by this mechanism, Hussain et al. [252] have proposed an algorithm called DELTRON which minimizes the same cost function used by Tempotron, but in the space of axonal delays. The authors have provided a VLSI implementation of DELTRON, showing that their architecture is simpler as compared to the ones implementing weight modifications. However, the memory capacity of DELTRON is slightly lower than its weight based counterparts.

Second, we look into some previous applications of Tempotron for spatio-temporal pattern recognition. In [253], Zhao et al. have proposed an event or spike based neuromorphic system for recognizing human postures which has three parts: a feature map construction stage, a motion symbol detection stage for generating feature spikes and a classifier stage composed of LIF neurons trained by the Tempotron rule. The authors have generated a spike-based human posture dataset by employing an AER vision sensor and

compared the performance of the proposed systems against two other biologically inspired algorithms namely HMAX scheme [254] and the Serre model proposed in [255]. They have shown that the tempotron based classifier provides much better result than HMAX and gives comparable performance to Serre model while taking much less time. Moreover, they have presented the performance of their system on Mixed National Institute of Standards and Technology (MNIST) [41] and MNIST dynamic vision sensor (MNIST-DVS) [256] dataset. Furthermore, they have calculated that as a result of the Tempotron learning rule, the computation time used by the classifier is linear with respect to the number of incoming spikes and Tempotron neurons. In [257] Gütig et al. have employed a LIF neuron trained by Tempotron rule for complex visual recognition tasks. The authors have collected experimental neuronal firing data from the retinal ganglion cells of salamander and subsequently fed it directly to the LIF neuron. They have shown that the trained neuron is capable of performing sophisticated visual detection tasks while remaining invariant to polarity and strength of the image contrast. The superior performance of such a simple model on complex visual tasks demonstrate the power of Tempotron rule and spike-based temporal processing. Huajin et al. [258] have proposed a visual digit recognition system which uses Tempotron for classification. The proposed model consists of an encoding layer followed by a supervised classifier. They have used the encoding layer to transform the images into spatio-temporal spike trains through ganglion cells. The generated spike data was then fed into a LIF neuron trained by Tempotron which modeled the classifier stage. They have shown that the proposed system gives superior performance on the MNIST dataset.

Third, researchers have been looking into Tempotron from a theoretical viewpoint. In [259] Baldassi et al. presented a mathematical analysis of a discrete-time Tempotron model having discrete synapses. They discovered that the model is capable of learning random spatio-temporal patterns at a learning rate which saturates the information theoretic bounds. They have

also proposed novel local and distributed learning algorithms which are based on message passing methods and unlike Tempotron they do not follow the gradient descent rule. They have analyzed the pattern storing capability of the proposed rule both in continuous and discrete time domain. Rubin et al. [260] presented a theoretical study of the computational power of Tempotron. First, they have analytically proven that the capacity (maximum number of patterns per synapse that can be correctly classified) of the neuron is independent of the number of input synapses. Second, they have thrown some light on the dependency between various time scales in the system dynamics and classification performance of Tempotron. Finally, they have analyzed the complex geometric solution space for Tempotron rule. Florian et al. [261] have compared Tempotron with another spiking neural learning algorithm namely Remote Supervised Method (ReSuMe) and showed them to be quasi-equivalent to each other.

4.3 Proposed classifier architecture and MOT learning rule

In this section we propose the architecture of the classifier used and the Morphology Optimizing Tempotron (MOT) learning rule.

4.3.1 Classifier architecture

Similar to 3.3, the structure of NNLD considered here is characterized by m identical dendritic branches and k excitatory synaptic contacts per branch. For each branch, the synaptic contact is formed by one of d dimensions of input afferents where $d \gg k$. At the relevant times governed by incoming spikes, the synapses are activated and the membrane voltage is calculated by

Figure 4.2: The architecture of neuron with nonlinear dendrites (NNLD).
 weighted sum of PSPs as follows:

$$V(t) = \sum_{j=1}^m b(v_j(t)) = \sum_{j=1}^m b \left(\sum_{i=1}^d w_{ij} \sum_{\forall g \text{ s.t. } t_g^i < t} K(t - t_g^i) \right) \quad (4.6)$$

where w_{ij} is the weight of the i^{th} input line connected to the j^{th} branch, $v_j(t)$ is the input to the j^{th} dendritic nonlinearity, $b(\cdot)$ is the nonlinear activation function of the dendritic branch which is characterized by $b(v_j(t)) = v_j(t)^2 / x_{thr}$, K denotes the post-synaptic potential kernel and t_g^i (g is the spike index) are times of incoming spikes on the i^{th} afferent. The weights seen by the inputs lines are $w_{ij} \in \{0, 1, \dots, k\}$ i.e. nonnegative integers which can be implemented in hardware by a shared binary synapses using the AER protocol. To suppress unrealistically large values, we include a saturation level x_{sat} at the output of each dendrite such that for $b(v_j(t)) > x_{sat}$, $b(v_j(t)) = x_{sat}$. Similar to 3.3, we allow each input afferent to make multiple synaptic connections on the same dendritic branch but limit the total number of connections per branch by enforcing $\sum_{i=1}^d w_{ij} = k$ for each j . In this work, we consider normalized PSP of the form described by Eqn. 4.2.

For binary classification, our classifier consists of a single NNLD as shown in Figure 4.2 where the final output voltage V_{out} is interpreted to take a value of either “1” or “0” depending on whether the summed membrane voltage $V(t)$ is crossing a threshold voltage (V_{thr}). Similar to [56], this indicates that a neuron fires at least one spike if $V(t)$ crosses (V_{thr}); otherwise it remains quiescent. After the neuron fires a spike, it is returned to the refractory state and the rest of the spikes in the incoming pattern do not affect the computation.

4.3.2 MOT learning rule

As pointed out in 4.1, the training of this structure involves two simultaneous steps - learning the connections between inputs and dendrites and learning the neuronal firing threshold (V_{thr}).

Learning the connections

Since we have binary synapses, a morphological learning rule that can modify the connections between afferent lines and synapses is needed. The inspiration of this work comes from the Tempotron learning rule [56] that learnt the temporal feature of random spike patterns in two classes by updating its weights so that the neuron fires a spike for patterns in class P^+ and stays silent for the other class. It was shown in [56] that the Tempotron rule endows a neuron with larger classification capacity than a perceptron with same number of synapses. Hence, we also start with a cost function that measures the deviation between the maximum membrane voltage (V_{max}) generated by misclassified patterns and V_{thr} defined as:

$$E = \begin{cases} V_{thr} - V_{max}, & \text{if pattern in } P^+ \text{ is presented} \\ V_{max} - V_{thr}, & \text{if pattern in } P^- \text{ is presented} \end{cases} \quad (4.7)$$

where V_{max} is the maximal value of post-synaptic potential $V(t)$ at the time t_{max} , i.e., $V_{max} = V(t_{max})$. According to the gradient-descent method, the change in synaptic efficacy for the first case is calculated by:

$$\begin{aligned}
 \Delta w_{ij} &= - \frac{\partial E}{\partial w_{ij}} \\
 &= - \frac{\partial \left(V_{thr} - \left(\sum_{j=1}^m b \left(\sum_{i=1}^d w_{ij} \sum_{\forall g \text{ s.t. } t_g^i < t_{max}} K(t_{max} - t_g^i) \right) \right) \right)}{\partial w_{ij}} \quad (4.8) \\
 &= b'(v_j(t_{max})) \sum_{\forall g \text{ s.t. } t_g^i < t_{max}} K(t_{max} - t_g^i)
 \end{aligned}$$

where $b'(\cdot)$ denotes the derivative of $b(\cdot)$. The gradient for second case can be calculated similarly and we do not show it here for brevity. As mentioned earlier, since we consider binary weights, i.e., $w_{ij} = 1$ if a connection exists and 0 otherwise, we cannot directly modify the weights by adding the Δw_{ij} term derived here. Instead, the term Δw_{ij} in Equation 4.8 is reinterpreted as a correlation term $c_{pj}(= w_{ij})$ for the p^{th} synaptic contact point on the j^{th} dendritic branch where the i^{th} input line is connected and is used to guide the process of swapping connections. At every iteration of the learning process, the synapse with the lowest c_{pj} averaged over an entire batch of patterns from a randomly chosen target set will be replaced (weight changed from 1 to 0) with the highest c_{pj} synapse in a candidate replacement set similar to 3.3. To keep the chapter self-contained, the mechanism of the learning process is outlined briefly below:

1. The learning process starts with random initialization of the connection matrix between input afferents and dendritic branches.
2. In each iteration of the training process, the activation of synapses on each dendritic branch are determined and the cell membrane output voltage ($V(t)$) in 4.6 is calculated for all the input patterns.

3. From the calculated $V(t)$, the maximum membrane voltage (V_{max}) is observed and classification result is determined, i.e., the patterns are correctly classified if $V_{max} > V_{thr}$ for P^+ and $V_{max} < V_{thr}$ for P^- .
4. A random set T of n_T synapses having weight 1 was targeted for possible replacement. For all misclassified patterns, the correlation term, c_{pj} is calculated for each synapse in T and averaged over the entire pattern set,
5. The poorest-performing synapse (minimum c_{pj}) in T , T_{min} is chosen for replacement.
6. To aid the replacement process, a randomly chosen set R containing n_R of the d afferent lines is forced to make silent synapses with weight 1 on the dendritic branch of T_{min} . These synapses are “silent” since they do not contribute PSP to the computation of $V(t)$ —so they do not alter the classification when the same pattern set is re-applied. But now c_{pj} is calculated for synapses in R and T_{min} is replaced with the best-performing synapse (maximum c_{pj}) in R .
7. The learning from step (2) to (6) continues until all the patterns are correctly learnt (or) the maximum number of iteration is reached.

Learning the threshold

Since we do not have an arbitrary multiplicative weight in our neural model, the range of maximum voltages obtainable from our model in response to a fixed temporal spike pattern is limited. This is similar to the problem faced in [252]. Hence, improper selection of threshold may largely degrade the classification performance since a very large V_{thr} ($= m \times k \times K_{max}$ for example) may never be crossed by $V(t)$. To solve this issue we have proposed two methods to set the threshold. In the first method, we have determined the value of V_{thr} by noting the maximum value of $V(t)$, i.e., V_{max} at time t_{max}

for a large number of random input spike patterns and connection matrices. The resultant probability distribution of V_{max} was used to determine the optimal threshold V_{thr} . V_{thr} was set to be the voltage corresponding to the peak of this probability distribution function. Since this mechanism does not involve the learning of V_{thr} , it remains constant during learning and is referred to as “NNLD (static)”. In the second method, we have proposed an automatic mechanism for adapting V_{thr} during training. This technique involves updating the value of V_{thr} after each iteration which is guided by the following formula:

$$\Delta V_{thr} = \eta(w_{fp}FP - w_{fn}FN) \quad (4.9)$$

where FP , FN , w_{fp} , w_{fn} and $\eta > 0$ are the number of false positives, number of false negatives, weightage associated with false positive error, weightage associated with false negative error and threshold learning rate respectively. In this chapter, we keep $w_{fp} = w_{fn} = 1$. Equation 4.9 is responsible for balancing the number of false positives and false negatives. When $FP > FN$, the number of negative patterns incorrectly classified as positive patterns is more than the number of positive patterns incorrectly classified as negative patterns so to balance FP and FN Equation 4.9 increases the value of V_{thr} . On the other hand, when $FP < FN$ Equation 4.9 diminishes the value of V_{thr} . The rate of threshold adaptation is controlled by the threshold learning rate η . Since the value of V_{thr} is adapted during the training phase, we refer it as “NNLD (adaptive)”. The morphological learning rule described in Section 4.3.2 combined with the adaptive threshold mechanism comprises the MOT learning rule.

4.4 Experiments and Results

4.4.1 Problem Description

In this sub-section, we describe the two tasks used to demonstrate the performance of our algorithm. The reason for this choice is that both of these are standard problems shown in [56] and facilitates comparison.

Task I: Classifying Random Latency Patterns

The first task is binary classification of single spike random latency patterns [56]. To perform the task, P spike patterns were generated and randomly assigned to one of the two classes P^+ (Class 1) or P^- (Class 2). Each spike pattern $X = (x_1, x_2, \dots, x_d)$ consists of d afferents, where each of them spiked only once at a time drawn independently from a uniform distribution between 1 and T ms.

Task II: Classifying pairwise Synchrony Patterns

To examine the ability of our algorithm to learn correlations in multiple spikes, another data set that consists of pairwise synchrony events in each pattern is generated. In this data set, all the d afferents are grouped into $(d/2)$ pairs and afferents in a given pair fire single spike patterns synchronously. Since synchronous events occur at random, uniformly distributed times for both class of patterns, class information is ingrained solely in the patterns of synchrony; neither spike counts nor spike timing of individual neurons carry any information relevant for the classification task. This task was representative of spike synchrony-based sensory processing.

For both Task I and II, we have kept d as 500.

Figure 4.3: Distribution of V_{max} for 100 patterns (a) before and (b) after training for Task I. V_{max} for patterns in P^+ and P^- are shown in dashed and solid line respectively.

4.4.2 Results: Performance of NNLD trained by MOT

Throughout the experiment, the design parameters m , k , x_{thr} , x_{sat} , τ and T , were chosen as 100, 5, 1, 100, 15 *ms* and 400 *ms* respectively. As discussed in Section 4.3.2, the firing threshold V_{thr} is adapted and so it is randomly initialized before training.

To start with Task I, the NNLD in Figure 4.2 was trained on a small number ($= 100$) random latency patterns as generated in Section 4.4.1. The results in Figure 4.3 (a) and (b) show that the proposed method can efficiently perform the classification task. A clear separation between Class 1 and Class 2 shows that the proposed method is able to respond to the random single spike latency patterns by shifting V_{max} closer and farther to the V_{thr} for each pattern in classes 1 and 2 respectively.

The NNLD was also trained for larger number of input patterns (500 and 1000 patterns) as presented in Figure 4.4 (b) and (c). It shows that the proposed method can perform the classification task quite well by achieving accuracies of 95.58% ($SD = 0.54\%$) and 86.57% ($SD = 0.72\%$) respectively for these cases. Next, we performed the experiment of Task II with our network. It was observed that the classification performances (100%, 100%

Figure 4.4: The distribution of V_{max} for a set of (a) 100, (b) 500 and (c) 1000 patterns in task I showing 100%, 94.8% and 84.9% accuracy respectively.

and 99.71 %($SD = 0.1\%$) accuracy for 100, 500 and 1000 patterns) are much better than that of performance in random latency patterns of Task I. Hence, our method can identify the extra information embedded in synchrony of neural firings. It also reveals that the learning rule does not depend only on a single synchronous pair but on a huge number of synchronous pairs.

4.4.3 Results: Comparison with Tempotron learning

Next, the performance of proposed method is compared with the Tempotron learning [56] that learns spike time patterns by using weight updates. The number of synapses used by Tempotron is equal to the number of input afferents d . Since we are interested in the performance of these algorithms in their hardware implementations plagued by mismatch, we consider the performance of the Tempotron when its weight is quantized at different resolutions. Further, we do the quantization in two ways: either after training

Figure 4.5: Comparison studies on classification performance, for Tempotron learning and the proposed morphological learning rule for (a) Task I of Random Latency Patterns and (b) Task II of Pair-Wise Synchrony Patterns. The results have been averaged over 10 independent trials.

(AT) or as a step during training (DT). The first method corresponds to the case where weights trained in software version of the algorithm are downloaded to the hardware while the second method is analogous to performing training on-chip. Furthermore, for comparison between the two threshold selection mechanism as described in Section 4.3.2, we have also calculated the value of V_{thr} for the first method by equating it to the voltage corresponding to the peak location of V_{max} distribution over 10000 random input spike patterns. We term this as $V_{thr,static}$. Figure 4.5 depicts that the performance obtained by morphological learning with adaptive threshold (second method) is superior to that of learning with fixed $V_{thr,static}$. Moreover, the comparison results in Figure 4.5 (a) also show that the Tempotron using floating-point numbers achieves better performance compared to the proposed method. However, when the high resolution weights are quantized at 2-bit level, its performance is worse than the proposed method. Also, it can be seen that the performance is better when the quantization is performed within the learning loop. This is to be expected since the learning algorithms can now try to correct this quantization error as well.

Only at 4-bit quantization level, the classification performance of Tempotron (93.05% ($SD = 0.55\%$) and 89.14% ($SD = 0.7\%$) accuracy for 500 and 1000 patterns) in Task I is comparable to our proposed method using 1-bit

or binary weights. This underlines the importance of our proposed method in robustly implementing spike timing based classifiers using low-resolution analog synapses available in nano-scale CMOS or non-CMOS devices. Similarly, the classification performance of NNLD for Task II with 1-bit binary weights in Figure 4.5 (b) is comparable to performance of Tempotron at 4-bit quantization which shows about 100 % and 98.12%(0.12%) accuracy respectively for 500 and 1000 patterns.

4.5 Application example: Classifying Tactile sensing data

Till now the proposed algorithm has been applied to classify synthetic synchrony patterns. But, to check whether the algorithm can be applied to solve pattern recognition tasks, it is used to classify Tactile information. The patterns which have been till now presented for classification had one spike per afferent but real world scenarios may have multiple spikes per afferent.

4.5.1 Task Description

A detailed description of the task can be found in [262]—here, we give a brief description for completeness. The task requires a flexible, stretchable and conformable tactile sensor array made up of conductive fabric which is used for data collection. Two glass spheres (indenters) of diameter 65 mm and 105 mm were indented onto this sensor array by a suitably programmed 6-axis robotic arm. The indentation force used was 4N, as measured using a load cell placed below the sensor having an accuracy of 0.01N. The signal recording was started just before placing the indenter onto the sensor array and stopped before removing the indenter. Between consecutive indentations, a 5 second pause was provided so that the sensors were able to recover. A total of 100 recordings were taken per indenter. The collected data were converted to

spikes as described in the next sub-section. For each indenter, 60 randomly chosen recorded data was used for training both the proposed algorithm and Tempotron. The remaining 40 recordings were used for testing.

4.5.2 Spike Train Generation

The conversion of the analog data recorded by the sensor array to spike trains involves the following steps. First, the analog data is converted to digital output by applying a fixed threshold of 0.5. This digital output is inverted to generate another set of digital data. Each channel of the digital data and its inverted version are passed through Leaky Integrate and Fire Neuron to generate two sets of spike trains. These spike trains are combined to form the spike response to be given as an input to the algorithms. The usage of both the digital data and its inverted version for spike generation ensures both low to high and high to low transitions are captured by a change in firing activity. The combined spike train patterns given as input to the algorithms consist of 130 afferents. Thus, we allot 130 synapses for both Tempotron and NNLD.

4.5.3 Results

The performance of the proposed method on this application is compared with Tempotron algorithm at different quantization levels in Table 4.1. The results, averaged over 10 independent trials, show that the proposed algorithm is able to achieve an accuracy of 96.54% ($SD = 0.6543\%$). Although Tempotron without quantization performs better (97.06% ($SD = 0.5123\%$)) than the proposed learning rule on NNLD, but after quantization, at least 6 bits of weight resolution is needed by Tempotron to match our performance with 1 bit weights.

Table 4.1: Performance comparison of morphological learning rule on NNLD and Tempotron

Cases	Accuracy	
	Mean	SD
Morph. lear. on NNLD (adaptive V_{thr})	96.54	0.6573
Morph. lear. on NNLD (for V_{thr_static})	95.64	0.6473
Tempotron (no quantization) [56]	97.06	0.5123
Tempotron (6-Bit quantization after training)	95.64	0.7614
Tempotron (6-Bit quantization during training)	96.26	0.6822
Tempotron (4-Bit quantization after training)	78.87	1.8371
Tempotron (4-Bit quantization during training)	85.92	1.3846
Tempotron (2-Bit quantization after training)	51.56	3.765
Tempotron (2-Bit quantization during training)	55.88	2.8731

4.6 Discussion

Here, we compare our method with other reported algorithms and other possible variants of morphological learning.

4.6.1 Comparison to other supervised spiking neural classifiers

In recent years, several supervised learning algorithms have been proposed such as Tempotron [56], ReSuMe [113], SpikeProp [69] and Chronotron [263] for training spiking neurons. From a pattern recognition viewpoint, these algorithms can be classified into two types. The general theme of the first type of algorithms is that a desired output spike train is specified prior to learning for each class of patterns. SpikeProp, ReSuMe and Chronotron are examples of this type, among which SpikeProp can only produce a single output spike whereas the others are capable of producing multi-spike train. In the second type, no such desired output spike train is specified beforehand and the algorithms choose the best time to spike for each pattern during training. Tempotron and the proposed algorithm are examples of this type which chooses t_{max} (defined in Sec. 4.3.2) as the time to spike. We have already compared

the performance of our algorithm with Tempotron. We now choose ReSuMe as a representative of the first type of algorithms and compare its performance with the proposed learning rule for the two tasks described in Sec. 4.4.1. For ReSuMe, the neuron had to fire one spike at $t_+ = 350ms$ for P^+ patterns and at $t_- = 450ms$ for P^- patterns. The classification performance of ReSuMe (89.58%(SD=1.24%)) and 82.65%(SD=2.22%) accuracy for 500 and 1000 patterns) in Task I is worse compared to the proposed method. Similarly, for Task II, the proposed method performs much better than ReSuMe learning rule (96.74%(SD = 0.8467%) and 92.12%(SD = 1.1654%) accuracy for 500 and 1000 patterns).

4.6.2 Comparison to other classifiers using dendritic processing

Here we first compare our method with another dendritic algorithm proposed by Wu et al. in [202]. Unlike [202] which considered only mean rate encoded inputs, our learning rule can be applied to arbitrary spike trains.

Second, we compare the proposed algorithm with the structural plasticity rule proposed in Chapter 3 namely Dendritically Enhanced Readout (DER). In Chapter 3, this structure has been used as the readout stage of Liquid State Machine. The primary difference of the proposed algorithm with DER is in the number of data points per pattern to be memorized for a particular task. If we consider there are P^+ and P^- patterns of Class 1 and Class 2 respectively, then the number of data points to be memorized by the proposed algorithm for memorization tasks like Task I are $P_{m_NNLD} = T \times P^- + 1$ while the number of data points to be memorized by DER are $P_{m_DER} = T \times (P^+ + P^-)$ where T is the number of time points per pattern. So, DER has to memorize almost two times more data points than the proposed algorithm for the same number of patterns thereby reducing its memory capacity.

Third, we have also compared our performance with another recently pro-

posed dendritic algorithm termed as Synaptic Kernel Inverse Method (SKIM) [264]. The difference mentioned above in the context of DER also applies to SKIM. Apart from that, SKIM also uses much more resources than the proposed NNLD which we will show next. We compare the two methods on Task I for 100 patterns in which case our proposed method has 100% accuracy. The neural network architecture used in [264] consists of N pre-synaptic neurons which connect to an output spiking neuron, via synaptic connections to its M dendritic branches. The weights of these synapses are random and fixed. These synapses along with a subsequent nonlinearity projects the input to a higher dimension thereby increasing separability. The dendritic branches sum the synaptic input currents, and the output from the dendritic branches are summed at the soma of the output neuron. The weights of the connection between dendritic branches and the soma are learnt by Moore pseudo inversion method [264]. Thus, for this network $N \times M$ synaptic resources are required for connecting the N pre-synaptic neurons to M post-synaptic dendritic branches and M synapses are required for connecting the M dendritic branches to the single output spiking neuron. The number of input spiking neurons are equal to the number of afferents in the input data which in our case is 500. Since we have used 500 synapses for NNLD, so initially we keep the number of post-synaptic dendrites in SKIM as 1 ($M = 1$) to match the number of resources used by us. But for $M = 1$, SKIM does not perform well and provides only 50% accuracy. When the number of dendritic branches are increased, SKIM provides better results and finally is able to provide 100% accuracy when $M = 360$. So to provide equivalent result as the proposed algorithm SKIM requires 361 times more resources than our proposed NNLD.

4.6.3 Hardware implementation of Morphology Optimizing Tempotron

We discussed the hardware implementation of a two-cell NNLD architecture in Sec. 3.6 and Sec. 3.7. MOT is implemented by using only one NNLD and hence it requires lesser resources than LSM-DER. For off-chip training and on-chip testing, this single cell MOT architecture can be easily implemented by the resources described in Sec. 3.6 and Sec. 3.7. However, for the on-chip implementation of MOT, an extra circuit is needed for implementing the threshold adaptation technique.

Our aim is to implement Eqn. 4.9 proposed in Sec. 4.3.2. A block diagram to implement this equation is depicted in Fig. 4.6. First, $w_{fp}FP - w_{fn}FN$ is calculated and its absolute value is multiplied by η using a multiplier shown in Fig. 4.6. The output of the multiplier is given as input to an adder, the other input of which is the current V_{thr} . The sign of $w_{fp}FP - w_{fn}FN$ determines whether an *Add* or a *Sub* signal will be generated. If $w_{fp}FP - w_{fn}FN \geq 0$, an *Add* signal is produced, otherwise a *Sub* signal is generated. While the *Add* signal indicates that $\eta(w_{fp}FP - w_{fn}FN)$ will be added to the current V_{thr} , the *Sub* signal indicates that $\eta(w_{fp}FP - w_{fn}FN)$ will be subtracted from it.

The threshold adaptation circuit proposed here along with the circuit components discussed in Sec. 3.6 and Sec. 3.7 are enough to build a full working MOT architecture.

4.7 Summary

This chapter presents a morphological learning rule inspired by Tempotron that can be used to find the optimal morphology of neurons with nonlinear dendrites (NNLD) and binary synapses. The learning rule includes a novel threshold adaptation technique. To see the effectiveness of the proposed method, the NNLD trained with morphological rule is used to solve

Figure 4.6: This figure shows the block diagram for implementing the threshold adaptation technique of MOT.

two classification tasks and one application problem. The results depict that our proposed method with 1 bit weights can achieve comparable performance to tempotron learning rule with 4-bit to 6-bit quantized weights. In Chapter 3 and Chapter 4 we have proposed two supervised structural plasticity rules. In the next chapter we explore the training of NNLD networks with an online unsupervised learning rule. Although our learning rule is influenced by STDP, however unlike it, we use binary synapses and introduce dendritic integration.

Chapter 5

An Online Unsupervised Structural Plasticity algorithm for neurons with nonlinear dendrites

In Chapter 3 and Chapter 4 we proposed two supervised classifiers - LSM-DEER and MOT for training neurons with nonlinear dendrites (NNLD) and binary synapses. In this chapter we will propose a novel Winner-Take-All (WTA) architecture employing NNLDs with binary synapses and an online unsupervised structural plasticity rule for training it. The proposed learning rule is inspired by spike timing dependent plasticity (STDP) but differs for each dendrite based on its activation level. It trains the WTA network through formation and elimination of connections between inputs and synapses. To demonstrate the performance of the proposed network and learning rule, we employ it to solve two, four and six class classification of random Poisson spike time inputs. By proper tuning of the inhibitory time constant of the WTA we can set the number of subpatterns per pattern we want to detect before training. We show that while the percentage of suc-

successful trials are 92%, 88% and 82% for two, four and six class classification when no pattern subdivisions are made, it increases to 100% when each pattern is subdivided into 5 or 10 subpatterns. However, the former scenario is more jitter resilient than the later cases.

5.1 Introduction

The WTA is a computational framework in which a group of recurrent neurons cooperate and compete with each other for activation. The computational power of WTA [254, 265, 266] and its function in cortical processing [254, 267] have been studied in detail. Various models and hardware implementations of WTA have been proposed for both rate [268–275] and spike based [276–278] neural networks. In recent past, researchers have looked into the application of STDP learning rule on WTA circuits. The performance of competitive spiking neurons trained with STDP has been studied for different types of input such as discrete spike volleys [279–281], periodic inputs [282, 283] and inputs with random intervals [278, 284, 285].

In this chapter, for the first time we are proposing a Winner-Take-All (WTA) network which uses neurons with NNLD and binary synapses as the basic computational unit. This architecture, which we refer to as Winner-Take-All employing Neurons with NonLinear Dendrites (WTA-NNLD), uses a novel branch-specific Spike Timing Dependent Plasticity based Network Rewiring (STDP-NRW) learning rule for its training. The authors of [286] have presented a branch-specific STDP rule for batch learning of a supervised classifier constructed of NNLDs. The primary differences of the work proposed in this chapter with [286] are:

- We present an unsupervised learning rule for training a WTA network.
- We propose an online learning scheme where the connection modifications occur after presentation of each pattern.

In this chapter, we consider spike train inputs with patterns occurring at random order which is same as the type presented in [278]. The primary differences between our work and the one proposed in [278] are:

- Our WTA network is composed of neurons with nonlinear dendrites instead of traditional neurons with no dendrites.
- Unlike the network proposed in [278] that requires high resolution weights, the proposed network uses low-resolution non-negative integer weights which can be implemented in hardware by using binary synapses using the AER protocol [33, 34].
- In [278], though the neurons were allowed to learn and respond to sub-patterns but there was no actual guideline or control parameter to set the number of sub-patterns to be learned. Here we utilize the slow time constant of the inhibitory signal to select the number of sub-patterns we want to divide a pattern into.

In the following section, we will present a overview of NNLD, propose the WTA-NNLD architecture and STDP-NRW learning rule and show how the inhibitory slow time constant can be used to select sub-patterns within a pattern. Then we shall provide guidelines on selecting the parameters associated with WTA-NNLD and STDP-NRW. In Section 5.4 we will describe the classification task considered in this chapter which will be followed by the results in Section 5.5. We will also present the robustness of the proposed method to variations in parameters in Section 5.6, a quality that is essential for its implementation in low-power, sub-threshold neuromorphic designs which are plagued with mismatch. We will discuss possible hardware implementations in Section 5.7 and conclude the chapter in Section 5.8 by discussing the implications of our work and future directions in the last section.

5.2 Previous research on application of STDP in WTA networks

In this section, we throw some light on contemporary research works using unsupervised STDP learning rule to train WTA networks. Delorme et al. [279] have proposed a biologically plausible visual system framework which uses Rank Order Coding to store information. Rank Order Coding uses the order in which a neuron's inputs fire for encoding information. The proposed framework has two layers where the first layer models the retina and the second layer simulates a group of V1 neurons organized as multiple retinotopic homogeneous maps. Whenever a V1 neuron fires, it inhibits neurons at equivalent positions in the other maps thereby implementing the WTA mechanism. Natural images scenes were presented as input to the retinal layer and its output spike waves were transmitted to the V1 layer. The V1 layer was trained by a standard STDP learning rule. The authors have shown that after passing thousands of images through their architecture, the neurons of V1 layer are capable of implementing orientation like receptive fields. Masquelier et al. [281] have proposed a four layer hierarchical feed-forward SNN (S1-C1-S2-C2) for image recognition in which simple cells (S) gain selectivity from a linear sum operation, while complex cells (C) gain invariance from a nonlinear max pooling operation. Hence, this architecture models the increasing complexity and invariance observed along the ventral pathway [287]. The neurons of S1, C1 and S2 are arranged in retinotopic maps. The S1 neurons detect edges, C1 maps subsample S1 maps by taking the maximum response over a square neighborhood, S2 cells are selective to intermediate complexity visual features and C2 cells take the maximum response of S2 cells over all positions and scales. Lastly, a classifier takes the C2 cells' output and produces a decision class. The WTA mechanism was implemented in C1 layer through local lateral inhibition and STDP learning rule was used to train the connections between C1 and S2. The authors have

demonstrated the superior performance of their network for face and motor-bike recognition. Gerstner et al. [282] have looked into the effect of local Hebbian learning in a network of spiking neurons following the Spike Response Model from a theoretical viewpoint. They have shown that despite long post-synaptic and dendritic integration times in spiking networks, local Hebbian learning makes them capable of storing periodical spatio-temporal patterns with a time resolution of $1ms$. Yoshioka [283] has studied the construction and retrieval of associative memories in networks made of Hodgkin-Huxley type spiking neurons. A global inhibition was used to implement WTA and the network was trained by STDP learning rule for periodical spike patterns. The author also presented a theoretical analysis of the perfect memory retrieval state and showed it to be similar to the obtained numerical results. In [278], Masquelier et al. have proposed a network having an array of LIF neurons with local inhibitory connections. Multiple distinct high dimensional input patterns repeating at random intervals were presented to the network, while it was trained by STDP learning rule. They have shown that distinct neurons could pick up different parts of the input patterns by firing at different latencies. Hence, the network was able to learn subpatterns within the presented input patterns. To explain the computational function which emerges when a WTA network is trained by STDP, Nessler et al. have presented some theoretical analysis and results in [284]. They have shown that these networks provide the fundamental components of Bayesian computation by yielding representations of posterior distributions for hidden causes of high-dimensional spatio-temporal spike train inputs through the firing probabilities of the neurons. While the authors of [284] have looked into Bayesian inference, Kappel et al. [285] have shown that the spike trains emitted from a WTA network trained by STDP can be viewed as samples from the state space of a Hidden Markov Model (HMM).

5.3 Proposed architecture and learning rule

In this section, we first present the architecture of WTA-NNLD. This is followed by a description of the STDP-NRW learning rule. Lastly, we throw some light on the role of inhibitory time constant in balancing the specificity and sensitivity of the network.

5.3.1 Winner-Take-All employing Neurons with Non-linear Dendrites (WTA-NNLD)

We propose a spike based WTA network, depicted in Figure 5.1, which is composed of N NNLDs. Each NNLD consists of m identical branches having lumped nonlinearities with each branch containing k excitatory synaptic contact points of weight 1. If we consider a d dimensional input pattern, then each synapse is driven by any one of these input dimensions where $d \gg k$. We use the Leaky integrate-and-fire model to generate output spikes. Thus, the neuronal membrane voltage of the n^{th} NNLD is guided by the following differential equation:

$$\begin{aligned}
 C \frac{dV^n(t)}{dt} + \frac{V^n(t)}{R} &= I_{in}^n(t) \\
 \text{If } V^n(t) &\geq V_{thr}, V^n(t) \rightarrow 0; \& f^n(t) \rightarrow 1 \\
 \text{else } f^n(t) &= 0
 \end{aligned} \tag{5.1}$$

where $V^n(t)$, $I_{in}^n(t)$ and $f^n(t)$ are the membrane voltage, input current and output spikes of the n^{th} NNLD respectively. V_{thr} is the firing threshold which is kept same for all the NNLDs. Let us denote the input spike train arriving at the i^{th} input line as $e^i(t)$ which is given by:

$$e^i(t) = \sum_g \delta(t - t_i^g) \tag{5.2}$$

Figure 5.1: A spike-based WTA network employing neurons with lumped dendritic nonlinearities as the competing entities. For implementing lateral inhibition, an inhibitory neuron has been included which, upon activation, provides a global inhibition signal to all the NNLD.

where $g = 1, 2, \dots$ is the label of the spike. Then, the input current $I_{in}^n(t)$ to the neuron can be calculated as:

$$I_{in}^n(t) = \sum_{j=1}^m b\left(\sum_{i=1}^d w_{ij} \left(\sum_{\forall g \text{ s.t. } t_i^g < t} K(t - t_i^g) \right)\right) \quad (5.3)$$

where $b(\cdot)$ is the nonlinear activation function of the dendritic branch characterized by $b(z) = z^2/x_{thr}$. Here, x_{thr} describes the behavior of the dendritic nonlinear function by determining whether the dendrite will produce a sublinear, linear or supralinear output for a given input. K denotes the post-synaptic current kernel given by:

$$K(t) = I_0(e^{-\frac{t}{\tau_s}} - e^{-\frac{t}{\tau_f}}) \quad (5.4)$$

where τ_f and τ_s are the fast and slow time constants governing the rise and fall times respectively and I_0 is a normalizing constant. Similar to the algorithms proposed in Chapter 3 and Chapter 4 we consider low resolution non-negative integer weights associated with the input lines so $w_{ij} \in \{0, 1, \dots, k\}$ means the number of times the i^{th} input line is connected to the j^{th} dendritic branch. This results in easier hardware implementation since a low-resolution non-negative integer weight of W can be implemented by activating a shared binary synapse W times through time multiplexing schemes like address event representation (AER) [33,34]. Like our earlier work proposed in previous two chapters, we allow multiple connections of one input dimension to a single dendrite but restrict the number of connections per dendrite to k by enforcing $\sum_{i=1}^d w_{ij} = k$ for each j . The output of the n^{th} NNLD is a spike train and can be denoted as

$$f^n(t) = \sum_a \delta(t - t_n^a) \quad (5.5)$$

where $a = 1, 2, \dots$ is the spike index.

We have modelled the effect of lateral inhibition by providing each NNLD with a global inhibitory current signal $I_{inh}(t)$ supplied by a single inhibitory neuron N_{inh} through synapses. The signal $I_{inh}(t)$ is provided by the inhibitory neuron to all the NNLD whenever any one of them fires an output spike. $I_{inh}(t)$ is modeled as $I_{inh}(t) = K_{inh}(t - t_n^{last})$, when the last post-synaptic spike is produced by the n^{th} NNLD at t_n^{last} . The inhibitory post-synaptic kernel, K_{inh} , is given by:

$$K_{inh}(t) = I_{0,inh}(e^{-\frac{t}{\tau_{s,inh}}} - e^{-\frac{t}{\tau_{f,inh}}}) \quad (5.6)$$

where $\tau_{f,inh}$ and $\tau_{s,inh}$ are the fast and slow time constants dictating the rise and fall times of the inhibitory current respectively and $I_{0,inh}$ sets its amplitude.

5.3.2 Spike Timing Dependent Plasticity based Network Re-Wiring (STDP-NRW) learning rule

Since we consider synaptic contact points of weight 1, we do not have the provision to keep real valued weights associated with them. Hence, to guide the unsupervised learning, we define a correlation coefficient based fitness value $c_{pj}^n(t)$ for the p^{th} synaptic contact point on the j^{th} dendrite of the n^{th} NNLD of the WTA network, as a substitute for its weight. The operation of the network and the learning process comprises the following steps whenever a pattern is presented:

- $c_{pj}^n(t)$ is initialized as $c_{pj}^n(t = 0) = 0 \forall p = 1, 2, \dots, k; j = 1, 2, \dots, m \text{ \& } n = 1, 2, \dots, N$.
- The value of $c_{pj}^n(t)$ is depressed at pre-synaptic and potentiated at post-synaptic spikes according to the following rule:
 1. Depression: If the pre-synaptic spike occurs at the p^{th} synapse on the j^{th} dendritic branch of the n^{th} NNLD at time t^{pre} , then the

value of $c_{pj}^n(t)$ is updated by a quantity $\Delta c_{pj}^n(t^{pre})$ given by:

$$\Delta c_{pj}^n(t^{pre}) = -b'_j(t^{pre}) \bar{f}^n(t^{pre}) \quad (5.7)$$

where $\bar{f}^n(t) = K(t) * f^n(t)$ is the post-synaptic trace of the n^{th} NNLD and $b'()$ denotes derivative of the nonlinear function $b()$.

2. Potentiation: If the n^{th} NNLD of the WTA-NNLD network fires a post-synaptic spike at time t^{post} then $c_{pj}^n(t) \forall p = 1, 2, \dots, k; j = 1, 2, \dots, m$ i.e. for each synapse connected to the n^{th} NNLD is updated by $\Delta c_{pj}^n(t^{post})$ given by:

$$\Delta c_{pj}^n(t^{post}) = b'_j(t^{post}) \bar{e}^i(t^{post}) \quad (5.8)$$

where $\bar{e}^i(t) = K(t) * e^i(t)$ is the pre-synaptic trace of the corresponding input line connected to it.

A pictorial explanation of this update rule of $c_{pj}^n(t)$ is shown in Figure 5.2. Note that for a square law nonlinearity, $b'(z) \propto z$ and hence can be easily computed in hardware.

- During the presentation of the pattern whenever a spike is produced by any of the N excitatory NNLDs, the inhibitory neuron N_{inh} sends an inhibitory signal to all the NNLDs of the WTA.
- After the network has been integrated over the current pattern, the synaptic connections of the NNLDs which have produced at least one spike are modified.
- If we consider that Q out of N NNLDs have produced a post-synaptic spike for the current pattern, then the synaptic connections of the q^{th} NNLD $\forall q = 1, 2, \dots, Q$ is updated by tagging the synapse (s_{min}^q) having the lowest value of correlation coefficient out of the $m \times k$ synapses connected to it for possible replacement.

- To aid the unsupervised learning process, randomly chosen sets R^q containing n_R of the d input dimensions are forced to make silent synapses of weight 1 on the dendritic branch of $s_{min}^q \forall q = 1, 2, \dots, Q$. We term these synapses as “silent” since they do not contribute to the computation of $V^n(t)$ - so they do not alter the classification when the same pattern set is re-applied. The value of $c_{pj}^q(t)$ is calculated for synapses in R^q and s_{min}^q is replaced with the synapse having maximum $c_{pj}^q(t)$ (r_{max}^q) in $R^q \forall q = 1, 2, \dots, Q$ i.e. the input line connected to s_{min}^q is swapped with the input line connected to r_{max}^q .
- All the $c_{pj}^n(t)$ values are reset to zero and the above mentioned steps are repeated whenever a pattern is presented. Here, we define an epoch for C -class classification as a set of patterns consisting of one pattern from each of the C classes- in random order. We define another term l_{mean} as the average of the latencies of the post-synaptic spikes in the network over time period of the last epoch which is given by:

$$l_{mean} = \langle \sum_n \sum_a t_a^n \rangle_1 \quad (5.9)$$

where $\langle \cdot \rangle_1$, n and a denote averaging over one epoch, the n^{th} NNLD and the spike index of the output of the n^{th} NNLD respectively. We note the value of l_{mean} for every epoch and the learning is considered to converge when the value of a ‘Convergence Measure’ (CM) based on l_{mean} reaches saturation. We define our ‘Convergence Measure’ in Section 5.4.

5.3.3 Specificity and Sensitivity: Role of Inhibitory Time Constant

When a pattern is presented to the WTA-NNLD and any one of the N NNLDs produce an output spike, a global inhibition current $I_{inh}(t)$ is injected

Figure 5.2: An example of the update rule of fitness value ($c_{pj}^n(t)$) is shown. When a post-synaptic spike occurs at t^{post1} the value of $c_{pj}^n(t)$ increases by $b'_j(t^{post1})\bar{e}_i(t^{post1})$. Due to the appearance of a pre-synaptic spike at t^{pre2} , $c_{pj}^n(t)$ reduces by $b'_j(t^{pre2})\bar{f}^n(t^{pre2})$ as shown in the figure.

into all the N NNLDs. The slow time constant $\tau_{s,inh}$ of this signal controls the output firing activity of the WTA-NNLD. Typically, a large value of $\tau_{s,inh}$ (w.r.t to T_p) is set, and only one NNLD produces an output spike i.e. patterns of same class are encoded by a single NNLD. The post-synaptic spike latency for a pattern P is defined as the time difference between the start of the pattern and the first spike produced by any one of the N neurons of WTA-NNLD. During training of WTA-NNLD for this case different NNLDs get locked onto different classes of pattern and the latency gradually decreases until the end of the training. Thus, after completion of training, the unique NNLDs which have learned different classes of pattern rely only on the first few spikes (determined by the latency at the end of training) to predict the pattern's class thereby significantly reducing the prediction time [278]. So, the sensitivity of the network is increased. However, the problems with this approach are:

Figure 5.3: Specificity is reduced if only one NNLD encodes a pattern based on its first few spikes. As shown, a different pattern with a section resembling the beginning of class 1 pattern may cause neuron N_{f1} to respond.

- The percentage of successful classifications can be less due to the strict requirement of different neurons firing based only on first few spikes of different patterns (shown in Section 5.5).
- Though the prediction time of a patterns class is significantly reduced, this method neglects most part of the pattern after the first few spikes which may lead to a lot of false detections.

We demonstrate the limitation mentioned in the above point by a simple example in Figure 5.3. Let us consider we are performing C class classification and assume that after the training phase is complete, NNLD N_{f1} respond to patterns belonging to Class 1. NNLD N_{f1} has trained itself to provide an output spike depending on the position of the first few spikes (red spikes in dashed box of Figure 5.3) of the pattern. It neglects the rest of the

pattern while providing a prediction. However, for longer patterns there is a chance that this spike set can occur anywhere inside a random pattern (not belonging to any class or to another class). The same NNLD N_{f1} responds to such patterns by producing a post-synaptic spike. Thus, we see that though trained WTA-NNLD is very sensitive in this case, it loses specificity. On the other hand, if we set a moderate value of $\tau_{s,inh}$, then for a single pattern multiple NNLDs are capable of producing output spikes for one pattern. Hence, patterns of the same class are now encoded by a sequence of successive firing of few NNLDs where each NNLD fires for one sub-pattern. Let n_{sub} be the number of sub-patterns that is set by a proper choice of $\tau_{s,inh}$. Thus the original case of one NNLD firing for each pattern corresponds to $n_{sub} = 1$. In this chapter, for a C class classification we define a successful trial as one in which (a) during the training phase WTA-NNLD learns different unique representations for patterns of different classes and (b) after completion of training and achieving success in (a), the network produces the same representation, when presented with testing patterns, which it had learned during the training phase. When $n_{sub} = 1$ i.e. no pattern subdivisions are made, this unique representation is a different neuron firing for different classes of patterns. Hence, each class of input patterns is now identified by a unique order of neuron firing. When $n_{sub} > 1$, the unique representation is a different sequence of successive NNLDs firing for different classes of patterns. When, $n_{sub} > 1$, we allow the NNLDs to detect subpatterns within patterns. Since in this approach the WTA-NNLD gives weightage to the entire pattern before predicting its class, the number of false detections can be largely reduced. However, this method has a limitation of being less noise robust since one of the many sub-patterns can be easily corrupt by noise (shown in Section 5.5) and fail to produce an unique identifier during testing phase. Hence, the choice of n_{sub} and consequently the inhibitory time constant depends on the amount of temporal jitter in the application.

5.4 Choice of Parameters

The following is an exhaustive list of the parameters used by WTA-NNLD and STDP-NRW:

1. T_p : Duration of a pattern
2. d : Dimension of the input
3. m : Number of dendrites per NNLD
4. k : Number of synapses per dendrite
5. n_R : Number of input dimensions in replacement set
6. τ_s and τ_f : Slow and fast time constant of excitatory current kernel
7. I_0 : Normalization constant of excitatory current kernel
8. $\tau_{s,inh}$ and $\tau_{f,inh}$: Slow and fast time constant of inhibitory current kernel
9. $I_{0,inh}$: Normalization constant of inhibitory current kernel
10. x_{thr} : Threshold of dendritic nonlinearity
11. V_{thr} : Firing threshold voltage of NNLD
12. N : Number of NNLDs in WTA
13. C : Number of classes of patterns

We will now provide some guidelines on choosing the key parameters:

Total number of synapses per NNLD (s) The number of synapses allocated to each NNLD of WTA-NNLD are kept as equal to the dimension (d) of the input patterns. This is done to ensure NNLD uses the same amount of synaptic resources as used by a non-dendritic neuron. Thus, if the proposed network is comprised of N NNLDs then the total number of synaptic resources required are $d \times N$.

Number of dendrites per NNLD (m) In Eqn. 5.10 of Sec. 3.4.2 a measure of the pattern memorization capacity, B_N , of the NNLD (Figure 3.3) has been provided by counting all possible functions realizable as:

$$B_N = \log_2 \left(\binom{k+d-1}{k} + m - 1 \right) \text{bits} \quad (5.10)$$

where m , k and d are the number of dendrites, the number of synapses per dendrites and the dimension of the input respectively for the NNLD. When a new classification problem is encountered, we first note down the value of d , which in turn sets our s since we have considered $s = d$. Since $s = m \times k$, for a fixed s all possible values which m can take are factors of s . We calculate B_N for these values of m by Equation 5.10. The value of m for which B_N attains its maxima is set as m in our experiment. As an example, we show in Figure 5.4 the variation of B_N with m when $d = 100$. It is evident from the curve that the capacity is maximum when $m = 25$ and so in our simulations for classifying 100 dimensional patterns we employ NNLDs having 25 dendrites.

Number of synapses per branch (k) After s and m has been set, the value of k can be computed as $k = \frac{s}{m}$.

The normalization constant (I_0), slow (τ_s) and fast time constant (τ_f) of excitatory PSC kernel The fast time constant (τ_f) and the slow time constant τ_s have been defined in Section 5.3.2. In hardware implementation of a synapse [247] τ_f usually takes a small positive value and is typically not tuned. The slow time constant, τ_s , is responsible for integration across temporally correlated spikes and the performance of the network is dependent on its value. If τ_s takes too small a value, then the post-synaptic current due to individual spikes die down rapidly and thus temporal integration of separated inputs do not take place. On the other hand large values of τ_s renders all spikes effectively simultaneous. So, in both extremes the extrac-

Figure 5.4: The pattern memorization capacity of a NNLD (B_N) is plotted as function of the number of dendrites (m) for a fixed number of input dimensions ($d = 100$) and synapses ($s = 100$).

tion of temporal features from the input pattern is hampered. In Sec. 3.5, we have provided a mathematical formula for calculating $\tau_{s,opt}$, the optimal value of τ_s , with respect to the inter spike interval (ISI) of the input pattern for which optimal performance of the network is obtained. If we are considering d dimensional patterns and the mean firing rate of each dimension is μ_f , then the mean ISI across the entire pattern is given by $\mu_{ISI} = 1/(d \times \mu_f)$. We can then set $\tau_{s,opt}$ according to the formula

$$\tau_{s,opt} = 52.83\mu_{ISI} - 3.1 \quad (5.11)$$

In our simulations, we keep τ_f as $\tau_f = \frac{\tau_s}{10}$. As the weight of all the synaptic contact points are 1 so to keep the amplitude of the PSC for a spike as 1 we set $I_0 = 1.4351$.

Threshold of nonlinearity (x_{thr}) During the training of WTA-NNLD, the STDP-NRW rule preferably selects those connection topologies where correlated inputs for synaptic connections are connected to the same branch. Thus, the lumped dendritic nonlinearity $b(x) = \frac{x^2}{x_{thr}}$ should give a supralinear output only when correlated input dimensions are connected to the dendrite. To ensure this we keep the value of x_{thr} equal to the average input to the nonlinear function in case of random connections. We create numerous instances of dendrites having k synapses and calculate the average input to the nonlinear function, $b_{in,avg}$, for the pattern set at hand. Then we set the value of x_{thr} as, $x_{thr} = b_{in,avg}$.

V_{thr} of NNLD The NNLD should provide a post-synaptic spike only when correlated inputs have been connected to its dendrites. We consider a NNLD having m dendrites and k synapses and create numerous instances of random connections to these synapses. We measure the average value of the maximum membrane voltage produced when this NNLD is integrated over the pattern duration for all these instances given by $\langle (V)_{max} \rangle$ and set $V_{thr} = \langle (V)_{max} \rangle$.

The normalization constant ($I_{0,inh}$), slow ($\tau_{s,inh}$) and fast ($\tau_{f,inh}$) time constant of $I_{inh}(t)$ The post-synaptic firing activity of the WTA-NNLD network is dependent on $\tau_{s,inh}$ and $I_{0,inh}$. To simulate the hardware scenario we set $\tau_{f,inh}$ to a small value given by $\tau_{f,inh} = \frac{\tau_{s,inh}}{10}$. To set $I_{0,inh}$ and $\tau_{s,inh}$, we first excite WTA-NNLD with ep_{ini} epochs of patterns prior to training and calculate the average excitatory current ($I_{e,av}$) to the NNLDs as:

$$I_{e,av} = \frac{1}{ep_{ini}CT_p} \int_{t-ep_{ini}CT_p}^t \frac{1}{N} \sum_{n=1}^N I_{in}^n(t) dt \quad (5.12)$$

The idea is to generate a $I_{inh}(t)$ which if provided by N_{inh} at the beginning of a subpattern, decays exponentially to $I_{e,av}$ at the end of the subpattern

i.e. after time T_{sub} has elapsed. This ensures that once a post-synaptic spike is generated by a NNLD in a particular T_{sub} time window other NNLDs are unable to fire during that same T_{sub} time window. Assuming $\tau_{f,inh} \ll \tau_{s,inh}$ we can derive that the required $I_{inh}(t)$ is implemented by setting $\tau_{s,inh}$ as:

$$\tau_{s,inh} = \frac{T_{sub}}{\ln\left(\frac{I_{0,inh}}{I_{e,av}}\right)} \quad (5.13)$$

Note that $\tau_{s,inh}$ has an inverse logarithmic relation to $I_{0,inh}$.

Convergence Measure (CM) The formula for calculating CM , applicable to both $n_{sub} = 1$ and $n_{sub} > 1$, for detecting the convergence of learning is given by:

$$CM = \frac{l_{mean}}{n_{sub}} - \frac{(n_{sub} - 1) T_{sub}}{2} \quad (5.14)$$

Note that for $n_{sub} = 1$, $CM = l_{mean}$ and so it computes the time-to-first spike for patterns averaged over an epoch. For $n_{sub} > 1$, CM calculates the average time-to-first spike from the beginning of each subpattern of C patterns of an epoch. We consider the learning has converged when the value of CM saturates.

5.5 Experiments and Results

In this section, we will describe the classification task considered in this chapter. To show how the classification performance generalizes to multi-class we will consider two, four and six class classification. We will be showing the performance of WTA-NNLD and STDP-NRW for three values of n_{sub} given by $n_{sub} = 1, 5$ and 10 .

Figure 5.5: (a) The percentage of successful trials is plotted against $\frac{N}{C}$ for two-class, four-class and six-class classification. The figure shows that as $\frac{N}{C}$ increases the percentage of successful trials also increase and becomes constant after $\frac{N}{C} = 11$. (b) The evolution of CM with the number of epochs for two-class, four-class and six-class classification for $n_{sub} = 1$.

5.5.1 Problem Description

The benchmark task we have selected to analyze the performance of the proposed method is the Spike Train Classification problem [244]. In the generalized Spike Train Classification problem, C arrays of h Poisson spike trains having frequency f and length T_p are present which are labeled as classes 1 to C . Jittered versions of these templates are created by altering the position of each spike within the templates by a random amount that is randomly drawn from a Gaussian distribution with zero mean and standard deviation σ_{jitter} . The network is trained by these jittered versions of spike trains, and the task is to correctly identify a pattern's class. In this chapter, unless otherwise mentioned, we have considered $h = 100$ and Poisson spike trains are present in each afferent, $f = 20$ and $T_p = 0.5$ sec and varied C and σ_{jitter} . Inspired from [278], we also consider the scenario when $h/2$ randomly chosen afferents do not contain any spikes, while the remaining $h/2$ afferents are Poisson spike trains.

Figure 5.6: For four class classification the above figure shows that out of 44 NNLDs (a) 7th NNLD recognizes Class 1 pattern (b) Class 2 pattern fires for the 12th NNLD, (c) 36th NNLD recognizes Class 3 pattern (d) Class 4 pattern is recognized by 9th NNLD and the latency for all four of them decreases over epochs until saturation.

5.5.2 Case 1: $n_{sub} = 1$

In this case we have $T_{sub} = T_p$, so one NNLD is capable of firing only once when a pattern is presented. Considering $\sigma_{jitter} = 0$, we have varied the number of NNLDs and noted the percentage of successful trials which is depicted in Figure 5.5(a). To make the horizontal axis invariant of the number of classes, we have taken $\frac{N}{C}$ as the horizontal axis. From the figure we can

conclude that the percentage of successful trials gradually increases with an increase in $\frac{N}{C}$ and finally becomes constant after $\frac{N}{C} = 11$. Thus, unless otherwise mentioned, we will keep $N = 11 \times C$ when $n_{sub} = 1$. It can be seen from Figure 5.5(a) that the percentage of successful trials cannot go beyond 92 %, 88 % and 82 % for two, four and six class classification respectively.

Earlier in Section 5.4 we have proposed that the learning converges when the CM saturates. For $n_{sub} = 1$, $CM = l_{mean}$ is the average of the time-to-first spikes for patterns in an epoch. As an example we consider a particular trial of four-class classification and show in Figure 5.6 that during training, the latencies of the four NNLDs, N_{f1} , N_{f2} , N_{f3} and N_{f4} which uniquely recognize the four class of patterns gradually reduce until reaching a saturation point. Moreover, in Figure 5.5(b) we show the epochwise evolution of CM averaged over 50 trials for two-class, four-class and six-class classification. It is evident from the figure that the value of CM decreases thereby showing that the algorithm is favoring correlated inputs such that the post-synaptic spikes can occur faster and finally saturates after some epochs have passed. We denote the number of epochs taken by the algorithm for saturation of CM as ep_{sat} and note its value for 50 trials. The average value of ep_{sat} for 50 trials, $ep_{sat,avg}$, is then computed to be $ep_{sat,avg} = 149, 157$ and 165 for two-class, four-class and six-class classification respectively. Moreover, this phenomenon clearly indicates that while the WTA-NNLD network is being trained by STDP-NRW learning rule, C unique NNLDs which have locked onto the C different classes of pattern, are trying to recognize the start of repeating patterns for different classes. Figure 5.5(b) also suggests that after the training has stopped, these C unique NNLDs, instead of looking at the whole pattern of duration $T_p = 500ms$, can now look only at the starting $44.8ms$, $53.1ms$ and $56.1ms$ of the patterns for $C = 2, 4$ and 6 respectively to predict its class.

Let us now consider the effect of jitter and we show in Figure 5.7(a) the performance of the proposed method when the amount of jitter is varied.

Figure 5.7: (a) The percentage of successful trials is plotted against σ_{jitter}/τ_s for $n_{sub} = 1, 5$ and 10 . As the number of sub-pattern divisions are increased the jitter/noise robustness of the network decreases. (b) The percentage of successful trials is plotted against σ_{jitter}/τ_s for $n_{sub} = 1, 5$ and 10 for patterns having spikes in only 50% of the afferents.

Next, we look into the performance of the proposed method when patterns with 50% empty afferents are considered. Figure 5.7(b) depicts the results obtained by our network for this case with varying amounts of jitter.

5.5.3 Case 2: $n_{sub} > 1$

Next, we consider $n_{sub} = 5$ i.e. we divide each pattern into 5 sub-patterns by setting $\tau_{s,inh}$ and $I_{0,inh}$ as per Equation 5.13. For C class classification the maximum number of subpatterns can be $C \times n_{sub}$ so we set $N = C \times n_{sub}$ in this case i.e. we keep $N = 10, 20$ and 30 for two, four and six class classification respectively. Considering $\sigma_{jitter} = 0$, the evolution of CM with epochs for two, four and six class classification averaged over 50 trials is shown in Figure 5.8(a). Moreover, the value of $ep_{sat,avg}$ (averaged over 50 trials) is found out to be 210, 221 and 230 when $C = 2, 4$ and 6 respectively. Unlike Case 1, here we consider a pattern as an unique firing sequence of few NNLDs. As an example, we consider a particular trial of four class

Figure 5.8: (a) The evolution of CM (averaged over 50 trials) with the number of epochs for two-class, four-class and six-class classification when $n_{sub} = 5$. (b) This figure depicts the number of epochs needed for saturation of CM (averaged over 50 trials) vs the number of sub-patterns considered for each pattern (n_{sub}). As n_{sub} increases WTA-NNLD has to train itself for more number of sub-patterns and thus there is an increase in $ep_{sat,avg}$.

classification and look into the first and last 3 epochs during its training. It is evident from Figure 5.9 that during the first 3 epochs, WTA-NNLD produces arbitrary sequence of spikes. However, it can be seen that after the training of the network is complete, WTA-NNLD produces different firing sequences for different patterns while producing the same sequence when same patterns are encountered. WTA-NNLD trained by this method produces a 100% accuracy in recognizing different patterns by producing its unique firing sequence for two, four and six class classification. The performance of the network with varying intensity of jitter is provided in Figure 5.7(a) (spikes present in all afferents) and Figure 5.7(b) (spikes present in only half of the afferents) which depict that the $n_{sub} = 5$ case is less jitter resilient than the $n_{sub} = 1$ case.

We further increase the resolution of pattern subdivision by decreasing $\tau_{s,inh}$. We consider $n_{sub} = 10$ and following the principle of $n_{sub} = 5$, the number of NNLDs employed for $n_{sub} = 10$ are 20, 40 and 60 for two-class, four-class and six-class classification. This approach also provides 100% ac-

Figure 5.9: The input and output of WTA-NNLD has been shown for a particular trial of the four class classification when $n_{sub} = 5$. P_1 , P_2 , P_3 and P_4 represent the patterns of a particular class. The figure depicts that before learning WTA-NNLD produces arbitrary spikes whenever a pattern is presented. After learning, the network produces unique sequence of NNLD spikes for patterns of different classes. This unique sequence acts as an identifier of the pattern class.

curacy in providing a unique sequence of firing whenever a particular pattern is encountered when $\sigma_{jitter} = 0$. However, the performance of the network falls rapidly with the increase of σ_{jitter} as shown in Figure 5.7(a) and Fig-

ure 5.7(b). We also show the evolution of CM with the epochs in Figure 5.8(a). Furthermore, the number of epochs needed for convergence of CM in this case is much more than the previous cases as depicted in Figure 5.8(b). We conclude that dividing a pattern into too many sub-patterns hamper the network performance.

Next we delve a bit further and show the statistics of causes for the failure of the system in producing successful trials. A trail may fail if either condition (a) or (b) (described in Section 5.3.3) is not satisfied. We denote the failure of condition (a) as $F1$. Note that for a trial, condition (b) can fail if a pattern is misclassified as a pattern of another class (denoted as $F2$) or as a random pattern (denoted as $F3$). Table 5.1 shows the statistics of failed trials for $n_{sub} = 1, 5$ and 10 when $\sigma_{jitter}/\tau_s \approx 0.1$. Moreover, we test our network with random patterns and note the cases where a learnt unique representation is produced for a random input pattern i.e. a false positive error occurs. The percentage of false positive errors produced for $n_{sub} = 1, 5$ and 10 when $\sigma_{jitter}/\tau_s \approx 0.1$ are 8%, 0% and 0% respectively. Note that no false positive errors occur for $n_{sub} = 5$ and 10 since it is highly unlikely for a random pattern to make a sequence of neuron firing same as any of the learnt representation.

Table 5.1: Analysis of failure statistics

Case	$n_{sub} = 1$	$n_{sub} = 5$	$n_{sub} = 10$
$F1$	12%	2%	6%
$F2$	2%	2%	2%
$F3$	2%	4%	8%

Figure 5.10: Stability of WTA-NNLD trained by STDP-NRW is plotted with respect to different hardware non-idealities for $n_{sub} = 1$ (a) and $n_{sub} = 5$ (b). The constant red line indicates the percentage of successful trials obtained by our method without any nonidealities when $\sigma_{jitter}/\tau_s \approx 0.1$. The bars represent the percentage of successful trials obtained after inclusion of non-idealities. The right-most bar marked by (...) represents the performance when all the non-idealities are included simultaneously.

5.6 VLSI Implementation: Effect of statistical variation

In this section we analyze the stability of our algorithm to hardware non-idealities by incorporating the statistical variations of the key sub-circuits. The primary sub-circuits needed to implement our architecture are synapse, dendritic squaring block, neuron and c_{pj}^n calculator. While the variability of synapse circuit is modeled by mismatch in the amplitude (I_0) and time constant (τ_s) of the synaptic kernel function, the variability of the squaring block is captured by a multiplicative constant (cb_{ni}) as shown in Section 3.6. We do not consider the variation of inhibitory current kernel since it is global and only a single instance is present in the architecture. In Chapter 3.6, we proposed the circuits for implementing the synapse and squaring block of NNLD and performed Monte Carlo analysis to find their variabilities. We

presented that the $\frac{\sigma}{\mu}$ of I_0 , τ_s and cb_{ni} for the worst case scenario are 13%, 10.1% and 18% respectively. The mismatch of the LIF neuron circuit proposed in [288] was captured by variations in the firing threshold V_{thr} , the $\frac{\sigma}{\mu}$ of which was computed to be 12.5%. Lastly, the nonidealities of the c_{pj}^n calculator block is modeled as a multiplicative constant (cc_{ni}). Monte Carlo analysis of the c_{pj}^n calculator block revealed that its $\frac{\sigma}{\mu}$ for the worst case is 18%. Figure 5.10 shows the performance of the proposed method when these nonidealities are included in the model for $n_{sub} = 1$ and $n_{sub} = 5$ when $\sigma_{jitter}/\tau_s \approx 0.1$. The bars corresponding to I_0 , τ_s , cb_{ni} , and cc_{ni} denote the performance degradation when statistical variations of I_0 , τ_s , cb_{ni} , and cc_{ni} are included individually. The results of Figure 5.10(a) and Figure 5.10(b) depict that the performance of the proposed algorithm is most affected by τ_s and cc_{ni} and least by cb_{ni} . Finally, to mimic the proper hardware scenario we consider the simultaneous implementations of all the nonidealities, which is marked by (...). The (...) bars show that there is an 8% and 6% decrease in performance for $n_{sub} = 1$ and $n_{sub} = 5$ respectively.

5.7 Hardware implementation of fitness update

In Sec. 5.3.2, we described the proposed STDP-NRW learning rule. The primary component of our learning is a STDP-like update of the correlation coefficient based fitness value $c_{pj}^n(t)$ for the p^{th} synaptic contact point on the j^{th} dendrite of the n^{th} NNLD of the WTA network. If we consider that the i^{th} input line i.e. $e_i(t)$ is connected to this contact point, then implementation of the fitness update mechanism is depicted in Fig. 5.11. In Fig. 5.11(a), we show the information flow from the input afferent ($e^i(t)$) to the neuronal output ($f^n(t)$). The auxiliary variables $e^i(t)$ and $f^n(t)$ and their convolved versions $\bar{e}^i(t)$ and $\bar{f}^n(t)$ are shown. Moreover, we show the input and output of the dendritic nonlinearity i.e. $v_j(t)$ and $z_j(t)$ respectively. Note that, since

we have considered square-law nonlinearity, $b'_j(t) \propto v_j(t)$. The copy synapse is required to generate a copy of the signal $\bar{e}^i(t)$ such that the operation of the original path remains unaffected when it is used.

Figure 5.11(b) depicts the mechanism of fitness update. At the start of the pattern, the voltage V_C across the capacitor C is reset by using S and V_{mid} . After that V_C increases for post-synaptic spikes and decreases for pre-synaptic spikes as follows. Whenever a pre-synaptic spike/pulse occurs at time t^{pre} , the switch corresponding to $e^i(t)$ closes and V_C reduces by a amount $\Delta V_C(t^{pre}) \propto b'_j(t^{pre})\bar{f}^n(t^{pre}) \propto v_j(t^{pre})\bar{f}^n(t^{pre})$. This term can be calculated by mirroring currents $v_j(t)$ and $\bar{f}^n(t)$ from the information path of Fig. 5.11(a) and passing them through a multiplier.

On the other hand, whenever a post-synaptic spike/pulse occurs at time t^{post} , the switch corresponding to $f^n(t)$ closes and V_C increases by an amount $\Delta V_C(t^{post}) \propto b'_j(t^{post})\bar{e}^i(t^{post}) \propto v_j(t^{post})\bar{e}^i(t^{post})$. This term can be easily computed by mirroring currents $v_j(t)$ and $e^i(t)$ and passing them through a multiplier. Note that a pulse extender might be required if the spike widths are not reliable enough to trigger the above mechanisms.

At the end of the pattern, V_C becomes a measure of the correlation coefficient based fitness value $c_{pj}^n(t)$. Hence, the synapse with the lowest fitness value is identified by comparing the value of V_C 's at the end of a pattern.

5.8 Summary

In this chapter, we have proposed a new neuro-inspired Winner-Take-All architecture (WTA-NNLD) and a STDP inspired dendrite specific structural plasticity based learning rule (STDP-NRW) for its training. Motivated by recent biological evidences and models suggesting nonlinear processing properties of neuronal dendrites we employ neurons with nonlinear dendrites to construct our WTA architecture. Moreover, we consider binary synapses instead of high resolution synaptic weights. Thus our learning rule, instead of

Figure 5.11: In this figure, the information flow for the i^{th} input line connected to the p^{th} synaptic contact point on the j^{th} dendrite of the n^{th} NNLD of the WTA network is shown in (a). On the other hand, (b) shows a mechanism to calculate the correlation coefficient based fitness value of this synapse.

weight updates, trains the network by modification of the connections between input and synapses. We have also provided a method by which the

number of subpatterns per pattern learned by WTA-NNLD can be controlled. WTA-NNLD encodes patterns of different classes by either activity of distinct NNLDs or by a distinct sequence of NNLD firings. To demonstrate the performance of WTA-NNLD and STDP-NRW, we have considered two, four and six class classification of 100 dimensional Poisson spike trains. We can conclude from the result that the slow time constant of inhibitory signal ($\tau_{s,inh}$) can be properly set to obtain a tradeoff between specificity and sensitivity of the network. In the next chapter, we will conclude this thesis and list some future directions in which we will venture with the work proposed in this thesis.

Chapter 6

Conclusion and Future Work

In the last three chapters, we have proposed two supervised and one unsupervised structural plasticity inspired learning rules for training neurons with nonlinear dendrites (NNLD) and binary synapses. In this chapter, we will provide some concluding remarks and show few directions in which we will carry this work forward.

6.1 Conclusion

In the last few years, various VLSI implementations of artificial neural networks i.e. neuromorphic systems have been proposed which either mimic sensory or processing functions of the brain. The later type of systems are usually trained by neuro-inspired learning algorithms for implementing memory and solving pattern recognition tasks. Typically learning is done on a computer and the trained high resolution synaptic weights are downloaded in these systems. However, due to the presence of nonidealities in hardware, accuracy of the synaptic weights decrease and hence these systems cannot match their software counterparts in performance. This problem can be tackled by using either VLSI designs which exhibit robustness to variations or low resolution synapses instead of high resolution ones. Systems of the former

type [28, 40] usually require large number of synapses to provide acceptable performance which makes their hardware implementation very costly. The adoption of low resolution or binary synapses provide a simple yet efficient solution for handling VLSI mismatch and hence we concentrate on proposing such systems in this thesis. However, since the computational power of a network is reduced by using binary synapses, we consider nonlinear dendritic processing for the neurons used in the proposed algorithms. Biological evidences accumulated over the last decade support the use of neurons with nonlinear dendrites and binary synapses. Since we consider binary synapses, the learning happens through formation and elimination of synapses i.e. by a restructuring of the input lines connected to the neuron. This type of morphological learning is comparable to structural plasticity observed in biology. During the training phase the learning rule selectively chooses the input lines which it should connect to the dendrites. Such a type of choice based mechanism is favoured by high dimensional inputs and this inspired us in employing the spiking reservoir used by Liquid State Machine (LSM) to project low dimensional inputs into high dimensional spaces. This led to the development of our first framework entitled Liquid State Machine with Dendritically Enhanced Readout (LSM-DER) trained by a novel Network Rewiring (NRW) learning rule which provides the following advantages:

1. Compared to the state-of-the-art LSM framework namely Liquid State Machine with Parallel Perceptron Readout (LSM-PPR), LSM-DER provides superior performance for both classification and approximation problems typically solved by LSM variants.
2. While LSM-PPR uses high resolution synapses, synapses used by LSM-DER are binary and hence suitable for robust hardware implementation.
3. LSM-DER can attain comparable performance to LSM-PPR with far fewer synapses. Moreover, when the same number of tunable synapses

are allocated to both of them, LSM-DER provides 3.3X less error in classification and 2.4X less error (when using same number of synaptic resources) in approximation.

4. The proposed area and power optimized structures for on-chip implementation of LSM-DER provide results comparable to its software framework, when trained by an online NRW learning rule having 4200 times lesser time constant than its batch counterpart.

Our next work, inspired by the Tempotron learning rule, was directly (without a reservoir) used for recognizing high dimensional spike trains. Unlike our previous work DER, the proposed rule is optimized for memory capacity. This algorithm termed as the Morphology Optimizing Tempotron (MOT) offers the following advantages:

1. MOT with 1-bit synaptic weights provides performance comparable to Tempotron using 4 or 6-bit weights.
2. Unlike Tempotron, MOT uses a novel mechanism to adapt the firing threshold of an NNLD during the training phase.
3. Compared to another supervised classifier using spiking neurons called Remote Supervised Method (ReSuMe), MOT provides 7-10% better performance.

Unlike the previous two algorithms, the final contribution of this thesis is an online unsupervised learning rule. This work proposes a morphological training algorithm termed as Spike Timing Dependent Plasticity based Network Re-Wiring learning rule (STDP-NRW) inspired from STDP. The algorithm is employed to train a Winner-Take-All (WTA) architecture composed of NNLDs with binary synapses. The advantages of this rule are as follows:

1. Apart from using robust binary synapses, the proposed method provides superior performance for classification of high dimensional multi-class Poisson spike trains.
2. The algorithm allows the detection of subpatterns within patterns by tuning the time constant of the global inhibitory neuron of the WTA architecture.
3. Unlike traditional spike-based unsupervised learning rules, the proposed method is extremely resilient to false positive errors.

We will conclude this subsection by comparing the bit requirement of our methods with conventional networks following the method given in [289]. Let us consider a conventional two-layer neural network with d inputs, H hidden layer neurons and C output neurons for C classes of input patterns. The comparable network in our proposed structural plasticity methods have H dendrites and C output neurons. $2m$ out of the H dendrites connect to each of the C output neurons ($H = 2mC$) weights of 1 and hence can be implicitly implemented by accumulators. If we consider that each weight of a conventional network has b bits of resolution, the total number of bits required by the conventional network(NOB_{conv}) is given by:

$$\begin{aligned}
 NOB_{conv} &= b \times H \times d + b \times H \times C \\
 &= b \times H \times (d + C) \\
 &= b \times H \times d \text{ for } C \ll d
 \end{aligned} \tag{6.1}$$

For the proposed case, the connection matrix is of size $d \times H$ though only $k \times H$ entries are non-zero where $k \ll d$. To implement this sparse connectivity efficiently in an address event framework, we propose to use a two tier addressing scheme as shown in Fig. 6.1. Here, the incoming address will be used to index into a pointer array of d entries with $\lceil \log_2(H \times k) \rceil$ bits per entry. An incoming spike address, say i , is used to index into this array and

Figure 6.1: To implement the sparse connectivity matrix, a two level addressing scheme is proposed where the first memory (indexed by the incoming address of the event) holds pointers to valid connection addresses stored in the second memory. read the two consecutive values a_i and a_{i+1} . As shown in the figure, suppose $a_i = p$ and $a_{i+1} = q$. $n_i = a_{i+1} - a_i$ is the number of synapses connected to this input. If $n_i > 0$, then $a_i = p$ is used as a pointer to the p^{th} location in a dendrite address array. This second array has $H \times k$ entries with $\lceil \log_2(H) \rceil$ bits per entry that hold the address of the dendrite where the synapse is located. n_i consecutive values (d_p to d_{q1}) are read as destination addresses to route the spikes. Now, the total memory required by the lookup table in the proposed methods (NOB_{prop}) can be estimated as:

$$NOB_{prop} = d \times \lceil \log_2(H \times k) \rceil + H \times k \times \lceil \log_2(H) \rceil \quad (6.2)$$

The memory requirements for the proposed and conventional methods are compared in Fig. 6.2 by setting $b = 4$, $H = 10^4$ and varying d over a wide range for $k = 16, 32$, and 64 . It can be seen that the proposed method requires much less memory than the conventional case for large values of d when the sparsity is higher while the overhead of having a pointer array is more for small values of d . The crossover typically happens for $d < 200$ for values of k as large as 64 . Since for most practical cases d is much larger, we expect our method to be widely applicable.

Figure 6.2: Memory requirement of proposed sparse connectivity scheme is compared with that of a normal fully connected weight memory with 4 bits per weight. The proposed scheme is far more memory efficient for high dimensional inputs ($d > 200$).

6.2 Future Work

The plans for our future work can be primarily subdivided into three categories:

1. Algorithmic Developments
2. On-chip implementation of the proposed frameworks
3. Real world applications

6.3 Algorithmic Developments

This subsection lists the work we are going to do which are related to development of the algorithmic frameworks proposed in this thesis.

6.3.1 Power efficient Deep Spiking Neural Networks

In this thesis, we have used structural plasticity based learning rules for training networks of NNLDs. However, we identify that structural plasticity is not confined to networks of NNLDs but can be used to train any network composed of binary synapses. Since structural plasticity produces sparse connections, it is more power efficient when compared to a conventional spiking neural network implementation. This is because networks having sparse connections will require less energy in communicating spikes to synapses and creating synaptic currents. In [290], Hasler et al. showed that the energy needed for communication of spikes form a significant portion of the total energy. Hence, networks having sparse connections are inherently more power efficient. To mathematically validate this claim, first we compare the number of synaptic events generated by a sparsely connected network obtained through structural plasticity and a conventional network. Second, we compare the energy required to implement both of them in different hardware platforms.

Let us consider that for both the proposed and the conventional architectures there are L layers of neurons where the i^{th} layer has H_i neurons. Since the conventional architecture is fully connected, each neuron of the i^{th} layer is connected to all the H_{i+1} neurons of the $(i + 1)^{th}$ layer for all values of $i = 1$ to $L - 1$. On the other hand, in the proposed method, structural plasticity forces each neuron of the $(i + 1)^{th}$ layer to receive inputs from only k out of all the H_i neurons present in the i^{th} layer. Now, let us consider the i^{th} and the $(i + 1)^{th}$ layer. If the average firing rate of the neurons of the i^{th} layer is $\bar{n}_i$ then the number of synaptic events that will be created per second for transferring the output of the i^{th} layer to the input of next layer by accessing a digital memory is given by:

$$NEV_{conv,sl} = \bar{n}_i \times H_i \times H_{i+1} \text{ Hz} \quad (6.3)$$

Since L layers of neuron are present, total number of synaptic events in the circuit would be $NEV_{conv} = (L - 1) \times NEV_{conv,sl}$.

In the proposed method, since each neuron of the $(i + 1)^{th}$ layer is allowed to receive input from only k out of the H_i neurons of the i^{th} layer, the total number of connections present in between the i^{th} and the $(i + 1)^{th}$ layer are $k \times H_{i+1}$. Hence, each neuron of the i^{th} layer makes $(k \times H_{i+1})/H_i$ connections to the next layer in average. The number of synaptic events generated per second in between the i^{th} and the $(i + 1)^{th}$ layer is then given by:

$$\begin{aligned} NEV_{prop,sl} &= \bar{n}_i \times H_i \times \frac{k \times H_{i+1}}{H_i} \text{ Hz} \\ &= \bar{n}_i \times k \times H_{i+1} \text{ Hz} \end{aligned} \quad (6.4)$$

Similar to the previous architecture, total number of synaptic events in the circuit would be $NEV_{prop} = (L - 1) \times NEV_{prop,sl}$. By taking the ratio of NEV_{conv} and NEV_{prop} we get:

$$\frac{NEV_{conv}}{NEV_{prop}} = \frac{H_i}{k} \quad (6.5)$$

Power dissipation is directly related to the number of memory accesses which is equal to the number of synaptic events generated. Since typically $H_i \gg k$, $NEV_{conv} \gg NEV_{prop}$ which suggests that structural plasticity based learning rules are power efficient.

Now, we shall compare the power required by the conventional and the proposed network if implemented in low-power general-purpose hardware systems that can be used to implement spiking neural networks. In this study, we consider University of Manchester's SpiNNaker [123, 131], IBM's TrueNorth chip [124], Stanford University's Neurogrid [125], NVIDIA's Tegra K1 processor [291], ETH Zurich's FPGA-based Minitaur [292] and a server CPU platform [291]. Moreover, we consider that for all i , $\bar{n}_i = 250\text{Hz}$, $H_i = d$ and $k = 10$. NEV_{conv} and NEV_{prop} becomes $(L - 1) \times 250 \times d^2 \text{ Hz}$ and

Table 6.1: Energy per synaptic event (E_s) for various hardware platforms

	TrueNorth	SpiNNaker	Neurogrid	Minitaur	Tegra K1	CPU
E_s	26 pJ	20 nJ	94 pJ	80 nJ	2.2 μ J	9.8 μ J

$(L - 1) \times 250 \times 10 \times d$ Hz respectively. We compute the power dissipation across different hardware systems by choosing various values of L and d and report our findings in Fig. 6.3 by varying $L \times d$ as a parameter. The formula for calculating power dissipation is given by:

$$\text{Power dissipation} = \begin{cases} E_s \times NEV_{conv} & \text{for conventional networks} \\ E_s \times NEV_{prop} & \text{for proposed method} \end{cases} \quad (6.6)$$

where E_s is the energy required per synaptic event or spike. The value of E_s for the considered platforms are shown in Table 6.1.

The results clearly depict that the proposed method is more power efficient in all the platforms. These results have inspired us to create, in near future, deep networks with 1-bit synapses that will be trained through structural plasticity. Note that since in TrueNorth the memory is distributed throughout the chip, its memory access technique is the most efficient and hence has the least E_s thereby making it the most power efficient system. However, in Neurogrid and SpiNNaker the memory is implemented in the periphery and hence accessing it requires a bit more E_s that in turn increases the power requirement. Moreover, in MiniTaur the RAM blocks are placed at a corner of the chip and hence requires comparatively more power than TrueNorth, Neurogrid and SpiNNaker.

6.3.2 Unsupervised adaptation of reservoir

LSM-DER, proposed in Chapter 3, does not involve any training in the liquid as it is mathematically difficult to formulate the synaptic weight update

Figure 6.3: In this figure, we compare the power dissipation of the conventional and the proposed network when both are implemented in various platforms. The figures clearly depict that the proposed structural plasticity based network dissipates less power than a conventional network across all platforms.

formula in recurrent networks due to the inherent output feedback associated with supervised learning techniques. Hence, we will look for unsupervised learning rules for training the liquid neurons. In Chapter 5 we proposed a STDP inspired unsupervised learning rule for training WTA networks. We plan to apply this learning rule for adapting the synaptic connections of liquid.

Moreover, till now we have employed NNLDs only in the readout of LSM. In future, we will utilize NNLDs and binary synapses to form the liquid. In addition, we will look into the effects of nonlinear dendrites on the Separation Property of the liquid.

Our aim is to form an improved liquid capable of training itself and having a higher Separation Property than a randomly generated liquid.

6.3.3 Application of DER and MOT as readouts

Apart from LSM, there are other two-stage supervised algorithms such as Extreme Learning Machine (ELM) [43], Echo State Network (ESN) [293], Synaptic Kernel Inversion Method (SKIM) [264], Remotely Supervised Method (ReSuMe) [113], etc. using synapses with random weights as their first stage in various different ways. DER and MOT can find an immediate application as the second stage of these algorithms.

6.3.4 LSM-DER and MOT as multi-class classifiers

In Chapter 3 and Chapter 4 we have shown that LSM-DER and MOT can be successfully applied to solve two-class classification problems. But, not all real world data can be classified into two classes. For example, classification of English alphabets, numerals, set of images, etc. fall under the category of multi-class classification. Hence, we will also look into solving these type of classification problems by LSM-DER and MOT. However, LSM-DER and MOT in their current form cannot be directly applied to solve multi-class

Figure 6.4: A network employing NNLDs with a single hidden layer is shown. The structure has d inputs, m_H hidden NNLDs and m_L output NNLDs. Each NNLD has m_D dendrites.

problems. Hence, we plan to upgrade these two frameworks for handling multi-class data.

6.3.5 Two layer backpropagating network of NNLDs

In this thesis, we have considered only single layer NNLD architectures. However, for complex pattern recognition tasks a single layer may not be enough to produce performance at par with state-of-the-art machine learning classifiers. Hence, we plan to develop NNLD architectures with one or multiple

hidden layers. A basic NNLD architecture with binary synapses and a single hidden layer is depicted in Figure 6.4. In addition, we plan to develop a structural plasticity based learning mechanism inspired from the backpropagation rule for training multi-layer neural networks.

6.4 On-chip implementation of the proposed frameworks

Two key requirements from hardware implementations of neuromorphic systems are low power and area overhead. Hence, we choose the subthreshold analog circuit design paradigm to implement the primary circuits used by DER, MOT and WTA-NNLD. The learning rules can be implemented with the help of FPGA where the blocks communicate among themselves through the AER protocol. The main blocks needed in these designs are DPI synapse, squaring circuit for implementing the dendritic nonlinearity, neuron block and the c_{pj} calculator which have been described in Chapter 3. We have fabricated an array of NNLDs and presented the results in [288]. However, this chip does not support on-chip testing and the learnt connections have to be downloaded from a computer. We are currently testing our NNLD chip which supports on chip structural learning.

6.5 Real world applications

We have thought of some immediate applications of the proposed learning rules on real world datasets. Some of these applications are:

1. **Image Classification:** We plan to apply the multi-class versions of LSM-DER and MOT and the proposed unsupervised learning rule for image classification. Specifically we want to classify the spiking image data from MNIST-DVS [256]. This dataset is generated by showing

the MNIST dataset to a highly sensitive Dynamic Vision Sensor [294] and recording the output spike trains.

2. **Asynchronous decoding of dexterous finger movements:** We will be applying the proposed algorithms for asynchronous decoding of individuated finger and wrist movements of monkeys. The task description can be found in [31].
3. **Isolated Word Recognition:** The proposed algorithms will be used for classifying spoken digits. A detailed description of the task is presented in [238]. The next step in this work would be to classify the spike train outputs of spiking cochlea such as Dynamic Audio Sensor / AER-EAR [25].

In this section, we presented some future works related to the work proposed in this thesis. However, this is not an exhaustive list and we will also explore other avenues in near future.

Appendix A

Estimating capacity of NNLD

Let us consider a NNLD having m dendritic branches with each branch having k excitatory synaptic contact points of weight 1. In addition, let the total number of synaptic resources used by it is denoted as s given by $s = m \times k$. Moreover, it receives d dimensional Boolean patterns as input. On the other hand, we consider a neuron without dendritic branches having the same number of synaptic resources (s) and receiving the same Boolean input patterns. The aim of this appendix is to compare the memory capacity i.e. the number of distinct input output functions that can be memorized by these two classes of neurons and demonstrate the superior computational power of NNLD.

Let us first compute the memory capacity of a neuron without dendrites. The number of distinct input output functions that can be memorized by this cell is equal to the number of ways in which d input dimensions can be distributed across s synapses where repetition is allowed and order does not matter. Repetition is allowed since the same input line can connect to the neuron multiple times and order does not matter as the synapses are considered to be locationless. We will tackle this problem by looking into an isomorphic problem i.e. the same problem appearing in a different guise. The problem is to find the number of ways in which s indistinguishable particles can be placed into d distinguishable boxes i.e. swapping the particles in any

Figure A.1: An instance of putting $s = 7$ indistinguishable particles into $d = 4$ distinguishable boxes modeled as a sequence of $|$'s and $\bullet$'s where $|$ and $\bullet$ denote a wall and a particle respectively.

way is not considered a separate possibility and all that is important are the counts for how many particles are in each box. As depicted in Figure A.1 any configuration can be encoded as a sequence of $|$'s and $\bullet$'s in a natural way. To be valid, a sequence must start and end with a $|$, with exactly $d - 1$ $|$'s and s $\bullet$'s in between; conversely, any such sequence is a valid encoding for configuration of particles in boxes. Thus there are $d + s - 1$ slots between the two outer walls, and we need only choose where to put the r $\bullet$'s, so the number of possibilities is given by $\binom{d+s-1}{s}$. To relate this back to the original problem we can let each box correspond to one of the d dimensions and use the particles as “check marks” to tally how many times each input is chosen. For example, if a certain box contains exactly 4 particles, that means the input line corresponding to that box was selected 4 times. The particles being indistinguishable corresponds to the fact that we do not care about the order in which the input lines are connected. Hence, the solution to the original problem is also $\binom{d+s-1}{s}$. Hence, the capacity in bits of a neuron without dendrites is given by $\log_2 \binom{d+s-1}{s}$.

This idea can be extended for computing the capacity of NNLD. However, as dendrites are present so we will take a two step approach. First, let us compute the number of distinct dendrites that can be created if each dendrite has k synapses. Using the method explained in previous paragraph this is calculated to be $\binom{d+k-1}{k}$. Now we need to select m dendrites out of these, where the same dendrite is allowed to connect multiple times to a neuron

Appendix A. *Estimating capacity of NNLD*

and dendrites are locationless with respect to the neuron, which is given by $\binom{(d+k-1)+m-1}{m}$. Hence, the capacity of a NNLD is given by $\log_2(\binom{(d+k-1)+m-1}{m})$ bits. These calculations clearly depict that the memorization capacity of NNLD is superior as compared to its non-dendritic counterpart.

Appendix B

NNLDs as universal approximator of Boolean Functions

In Appendix A, we discussed about the memory capacity of a NNLD. In this appendix, we shall throw some light on the approximation capabilities of NNLD. We will apply a theorem proposed by Auer et al. [216] for parallel perceptrons (depicted in Figure B.1), to the case of neurons with nonlinear dendritic branches. The architecture and functional input-output relation of parallel perceptrons are described in detail in Section 3.2.3. The theorem states that:

Assume that $h : D \rightarrow [-1,1]$ is some arbitrary continuous function with a compact domain $D \subseteq \mathbb{R}^d$ (for example $D = [-1,1]^d$, and $\epsilon > 0$ is some given parameter. Then there exists a parallel perceptron that computes a function $f : \mathbb{R}^d \rightarrow [-1,1]$ such that $|f(\mathbf{z}) - h(\mathbf{z})| < \epsilon$ for all $z \in D$. Furthermore any Boolean function can be computed by rounding the output of a parallel perceptron (i.e., output 1 if the number p of positive weighted sums is above some threshold else output 0).

Figure B.2 shows a neuron with m nonlinear dendrites with each dendrite

Figure B.1: A parallel layer of n perceptrons with no lateral interconnection. having k synapses and receiving d dimensional inputs. As considered in this thesis, the i^{th} input line is connected to the j^{th} dendrite with a nonnegative integer weight of $w_{ij} \in \{0, 1, \dots, k\}$. The type of dendritic nonlinearity $b(\cdot)$ used in this thesis is given by:

$$\begin{aligned}
 b(x) &= \frac{x^2}{x_{thr}} \quad \text{if } b(x) < x_{sat} \\
 b(x) &= x_{sat} \quad \text{if } b(x) \geq x_{sat}
 \end{aligned} \tag{B.1}$$

Under the assumption $x_{thr} \rightarrow 0$ and appropriate scaling by x_{sat} , $b(x)$ takes the form of threshold nonlinearity of a perceptron. Hence, each branch of a NNLD can be compared to a single perceptron of the parallel perceptron architecture. However, while parallel perceptron uses real-valued synaptic weights, dendrites use synapses with integer weights. It has been proven that, for Boolean inputs, any perceptron (threshold gate) could be implemented with integer weights, and also that the number of bits per weight is $\mathcal{O}(d \log(d))$ [10]. Hence, the computational power of a NNLD and a parallel perceptron are same in case of Boolean inputs. Thus, a NNLD can be considered as an universal approximator of Boolean functions.

Figure B.2: A neuron having m dendrites and k synapses.

Appendix C

Author's Publications

C.1 Journal Papers

1. S. Roy, A. Banerjee and A. Basu, "Liquid State Machine with Dendritically Enhanced Readout for Low-Power, Neuromorphic VLSI Implementations," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 5, pp. 681-695, Oct. 2014.
2. S. Roy, P. P. San, S. Hussain, L. W. Wei and A. Basu, "Learning Spike time codes through Morphological Learning with Binary Synapses," *IEEE Transactions on Neural Networks and Learning Systems*, 2015. (Accepted)
3. S. Roy and A. Basu, "An Online Unsupervised Structural Plasticity Algorithm for Spiking Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*. (Revised and resubmitted)
4. A. Banerjee, A. Bhaduri, S. Roy, S. Kar and A. Basu, "Spiking Neural Classifier with Lumped Dendritic Nonlinearity and Binary Synapses: A Current mode VLSI Implementation and Analysis," *IEEE Transactions on Biomedical Circuits and Systems*. (Revised and resubmitted)

C.2 Conference Papers

1. S. Roy, A. Basu and S. Hussain, “Hardware efficient, neuromorphic dendritically enhanced readout for liquid state machines,” in *Proceedings of the 2013 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, Rotterdam, The Netherlands, Oct. 2013, pp. 302-305.
2. S. Roy, S. Kar and A. Basu, “Architectural exploration for on-chip, on-line learning in spiking neural networks,” in *Proceedings of the 14th International Symposium on Integrated Circuits (ISIC)*, Singapore, Dec. 2014, pp. 128-131.
3. A. Banerjee, S. Kar, S. Roy, A. Bhaduri and A. Basu, “A Current-Mode Spiking Neural Classifier with Lumped Dendritic Nonlinearity,” *IEEE International Symposium on Circuits and Systems (ISCAS)*, Lisbon, Portugal, May 2015.
4. S. Korde, S. Roy, E. Yao and A. Basu, “On-chip Machine Learner for Spike Sorting in Implantable Brain Machine Interfaces (BMI),” *IRC Conference on Science, Engineering and Technology (IRC-SET)*, Singapore, May 2015.

Bibliography

- [1] H. Y. Hsieh and K. T. Tang, “VLSI implementation of a bio-inspired olfactory spiking neural network,” *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 23, pp. 1065–1073, Jul. 2012.
- [2] R. Mill, S. Sheik, G. Indiveri, and S. Denham, “A model of stimulus-specific adaptation in neuromorphic analog VLSI,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 5, pp. 413–419, Oct. 2011.
- [3] T. Koickal, L. Gouveia, and A. Hamilton, “A programmable time event coded circuit block for reconfigurable neuromorphic computing,” in *Computational and Ambient Intelligence* (F. Sandoval, A. Prieto, J. Cabestany, and M. Graa, eds.), vol. 4507 of *Lecture Notes in Computer Science*, pp. 430–437, Springer Berlin Heidelberg, 2007.
- [4] A. Polsky, B. W. Mel, and J. Schiller, “Computational subunits in thin dendrites of pyramidal cells,” *Nature Neuroscience*, vol. 7, no. 6, pp. 621–627, 2004.
- [5] P. Poirazi and B. W. Mel, “Impact of active dendrites and structural plasticity on the memory capacity of neural tissue,” *Neuron*, vol. 29, no. 3, pp. 779–796, Mar. 2001.
- [6] W. Maass, T. Natschläger, and H. Markram, “Real-time computing without stable states: a new framework for neural computation based on perturbations,” *Neural Computation*, vol. 14, pp. 2531–2560, Nov. 2002.
- [7] K. Micheva, B. Busse, N. Weiler, N. O'Rourke, and S. Smith, “Single-synapse analysis of a diverse synapse population: Proteomic imaging methods and markers,” *Neuron*, vol. 68, no. 4, pp. 639 – 653, 2010.

- [8] W. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.
- [9] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychological Review*, vol. 65, pp. 386–408, Nov. 1958.
- [10] S. Muroga, *Threshold Logic and its Applications*. New York: Wiley, 1971.
- [11] E. Alpaydin, *Introduction to Machine Learning, second edition*. The MIT Press, 2 ed., 2009.
- [12] W. Maass, G. Schnitger, and E. Sontag, “On the computational power of sigmoid versus boolean threshold circuits,” in *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, pp. 767–776, Oct 1991.
- [13] B. Dasgupta and G. Schnitger, “The power of approximating: a comparison of activation functions,” in *Advances in Neural Information Processing Systems 5*, pp. 615–622, Morgan Kaufmann, 1993.
- [14] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol. 2, no. 5, pp. 359 – 366, 1989.
- [15] V. Kurkova, “Kolmogorov’s theorem and multilayer neural networks,” *Neural Networks*, vol. 5, no. 3, pp. 501 – 506, 1992.
- [16] M. Oram, M. C. Wiener, R. Lestienne, and B. J. Richmond, “The stochastic nature of precisely timed spike patterns in visual system neural responses,” *Journal of Neurophysiology*, vol. 81, no. 6, pp. 3021–3033, 1999.
- [17] S. J. Thorpe and M. Imbert, “Biological constraints on connectionist modelling,” in *Connectionism in Perspective*, pp. 63–92, Elsevier, 1989.
- [18] D. Perrett, E. Rolls, and W. Caan, “Visual neurones responsive to faces in the monkey temporal cortex,” *Experimental Brain Research*, vol. 47, no. 3, pp. 329–342, 1982.

- [19] E. T. Rolls and M. J. Tovee, "Processing speed in the cerebral cortex, and the neurophysiology of visual backward masking," *Proceedings of the Royal Society*, vol. 257, no. 1348, pp. 9–15, 1994.
- [20] M. Abeles, *Corticonies: Neural circuits of the cerebral cortex*. Cambridge University Press, 1991.
- [21] M. Abeles, H. Bergman, E. Margalit, and E. Vaadia, "Spatiotemporal firing patterns in the frontal cortex of behaving monkeys," *Journal of Neurophysiology*, vol. 70, no. 4, pp. 1629–1638, 1993.
- [22] F. Rieke, D. Warland, R. V. Stevenick, and W. Bialek, *SPIKES: exploring the neural code*. Cambridge: MIT Press., 1996.
- [23] R. Lestienne, "Determination of the precision of spike timing in the visual cortex of anaesthetised cats," *Biological Cybernetics*, vol. 74, no. 1, pp. 55–61, 1996.
- [24] T. J. Sejnowski, "Time for a new neural code?," *Nature*, vol. 376, no. 6535, pp. 21–22, 1995.
- [25] V. Chan, S. Liu, and A. Schaik, "AER EAR: A matched silicon cochlea pair with address event representation interface," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, pp. 48–59, Jan. 2007.
- [26] E. Culurciello, R. E. Cummings, and K. Boahen, "A biomorphic digital image sensor," *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 281–294, Feb. 2003.
- [27] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128x128 120 dB 15 us latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 566–576, Feb. 2008.
- [28] E. Neftci, S. Das, B. Pedroni, K. K. Delgado, and G. Cauwenberghs, "Event driven contrastive divergence for spiking neuromorphic systems," *Frontiers in Neuroscience*, vol. 7, no. 272, pp. 1–14, 2014.
- [29] S. Bamford, A. Murray, and D. Willshaw, "Spike-timing-dependent plasticity with weight dependence evoked from physical constraints," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 6, pp. 385–398, Aug. 2012.

- [30] S. Mitra, S. Fusi, and G. Indiveri, “Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 3, pp. 32–42, Jan. 2009.
- [31] V. Aggarwal, S. Acharya, F. Tenore, H. Shin, R. E. Cummings, M. Schieber, and N. Thakor, “Asynchronous decoding of dexterous finger movements using m1 neurons,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 16, pp. 3–14, Feb. 2008.
- [32] S. Acharya, F. Tenore, V. Aggarwal, R. Etienne-Cummings, M. Schieber, and N. Thakor, “Decoding individuated finger movements using volume-constrained neuronal ensembles in the m1 hand area,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 16, pp. 15–23, Feb. 2008.
- [33] K. A. Boahen, “Point-to-point connectivity between neuromorphic chips using address events,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, pp. 416–434, May 2000.
- [34] S. Brink, S. Nease, P. Hasler, S. Ramakrishnan, R. Wunderlich, A. Basu, and B. Degnan, “A learning-enabled neuron array IC based upon transistor channel models of biological phenomena,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 7, pp. 71–81, Feb. 2013.
- [35] B. L. Barranco, T. S. Gotarredona, and R. S. Gotarredona, “Compact low-power calibration mini-dacs for neural arrays with programmable weights,” *IEEE Transactions on Neural Networks*, vol. 14, pp. 1207–1215, Nov. 2003.
- [36] D. Querlioz, O. Bichler, P. Dollfus, and C. Gamrat, “Immunity to device variations in a spiking neural network with memristive nanodevices,” *IEEE Transactions on Nanotechnology*, vol. 12, pp. 288–295, May 2013.
- [37] K. H. Kim, S. Gaba, D. Wheeler, J. M. Cruz-albrecht, T. Hussain, N. Srinivasa, and W. Lu, “A functional hybrid memristor crossbar-array/CMOS system for data storage and neuromorphic applications,” *Nano Letters*, vol. 12, no. 1, pp. 389–395, 2011.

- [38] G. Indiveri, B. L. Barranco, R. Legenstein, G. Deligeorgis, and T. Pro-dromakis, “Integration of nanoscale memristor synapses in neuromor-phic computing architectures,” *Nanotechnology*, vol. 24, no. 38, 2013.
- [39] M. Sharad, C. Augustine, G. Panogopoulos, and K. Roy, “Spin-based neuron model with domain-wall magnets as synapse,” *IEEE Transac-tions on Nanotechnology*, vol. 11, pp. 843–53, Jul. 2012.
- [40] 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, pp. 2081–2086, 2014.
- [41] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learn-ing applied to document recognition,” *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov. 1998.
- [42] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learn-ing*, vol. 20, no. 3, pp. 273–297, 1995.
- [43] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: Theory and applications,” *Neurocomputing*, vol. 70, no. 13, pp. 489 – 501, 2006. Neural Networks Selected Papers from the 7th Brazilian Symposium on Neural Networks (SBRN ’04) 7th Brazilian Symposium on Neural Networks.
- [44] C. C. H. 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.
- [45] D. H. O’Connor, G. M. Wittenberg, and S. S. H. Wang, “Graded bidi-rectional 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.
- [46] J. Brader, W. Senn, and S. Fusi, “Learning real-world stimuli in a neural network with spike-driven synaptic dynamics,” *Neural Compu-tation*, vol. 19, pp. 2881–2912, May 2007.

- [47] C. Koch, T. Poggio, and V. Torres, “Retinal ganglion cells: A functional interpretation of dendritic morphology,” *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, vol. 298, pp. 227–263, Jul. 1982.
- [48] W. Rall, “Functional insights about synaptic inputs to dendrites,” in *Analysis and Modeling of Neural Systems* (F. Eeckman, ed.), pp. 63–68, Springer US, 1992.
- [49] B. W. Mel, “The clusteron: Toward a simple abstraction for a complex neuron,” in *Advances in Neural Information Processing Systems*, pp. 35–42, 1992.
- [50] P. Poirazi, T. Brannon, and B. W. Mel, “Arithmetic of subthreshold synaptic summation in a model cal pyramidal cell,” *Neuron*, vol. 37, pp. 977–987, Mar. 2003.
- [51] M. Jadi, B. Behabadi, A. Poleg-Polsky, J. Schiller, and B. Mel, “An augmented two-layer model captures nonlinear analog spatial integration effects in pyramidal neuron dendrites,” *Proceedings of the IEEE*, vol. 102, pp. 782–798, May 2014.
- [52] A. K. McAllister, L. C. Katz, and D. C. Lo, “Neurotrophins and synaptic plasticity,” *Annual Review of Neuroscience*, vol. 22, no. 1, pp. 295–318, 1999.
- [53] C. Lüscher, R. A. Nicoll, R. C. Malenka, and D. Muller, “Synaptic plasticity and dynamic modulation of the postsynaptic membrane,” *Nature Neuroscience*, vol. 3, no. 6, pp. 545–550, 2000.
- [54] M. Segal, E. Korkotian, and D. D. Murphy, “Dendritic spine formation and pruning: common cellular mechanisms?,” *Trends in Neurosciences*, vol. 23, no. 2, pp. 53–57, 2000.
- [55] B. Lendvai, E. A. Stern, B. Chen, and K. Svoboda, “Experience-dependent plasticity of dendritic spines in the developing rat barrel,” *Nature* 404, p. 876880, Apr. 2000.
- [56] S. Gutig and H. Sompolinsky, “The tempotron: a neuron that learns spike timing-based decisions,” *Nature Neuroscience*, vol. 9, pp. 420–428, Feb. 2006.

- [57] W. Gerstner and W. Kistler, *Spiking Neuron Models: An Introduction*. New York, NY, USA: Cambridge University Press, 2002.
- [58] W. Maass and C. M. Bishop, eds., *Pulsed Neural Networks*. Cambridge, MA, USA: MIT Press, 1999.
- [59] W. Maass, “Networks of spiking neurons: The third generation of neural network models,” *Neural Networks*, vol. 10, no. 9, pp. 1659 – 1671, 1997.
- [60] R. VanRullen, R. Guyonneau, and S. J. Thorpe, “Spike times make sense,” *Trends in Neurosciences*, vol. 28, no. 1, pp. 1 – 4, 2005.
- [61] W. Maass, “Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons,” in *Advances in Neural Information Processing Systems*, 1997.
- [62] F. Pomulak and A. Kasinski, “Introduction to spiking neural networks: Information processing, learning and applications,” *Acta Neurobiol Exp (Wars)*, vol. 71, no. 4, pp. 409–33, 2011.
- [63] A. L. Hodgkin and A. F. Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve,” *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.
- [64] H. Markram, Y. Wang, and M. Tsodyks, “Differential signaling via the same axon of neocortical pyramidal neurons,” *PNAS*, vol. 95, no. 9, pp. 5323–5328, 1998.
- [65] L. Perrinet, M. Samuelides, and S. Thorpe, “Sparse spike coding in an asynchronous feed-forward multi-layer neural network using matching pursuit,” *Neurocomputing*, vol. 57, pp. 125 – 134, 2004.
- [66] A. Cassidy and V. Ekanayake, “A biologically inspired tactile sensor array utilizing phase-based computation,” in *IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 45–48, Nov. 2006.
- [67] O. Rochel, D. Martinez, E. Hugues, and F. Sarry, “Stereo-olfaction with a sniffing neuromorphic robot using spiking neurons,” in *16th European Conference on Solid-State Transducers (EUROSENSORS)*, Sep. 2002.

- [68] S. M. Bohte, J. N. Kok, and H. L. Poutr'e, "Spike-prop: error-backpropagation in multi-layer networks of spiking neurons," *Neurocomputing*, vol. 1, no. 4, pp. 17–37, 2002.
- [69] S. M. Bohte, H. A. L. Poutr, J. N. Kok, H. A. La, P. Joost, and N. Kok, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, pp. 17–37, 2002.
- [70] J. Xin and M. Embrechts, "Supervised learning with spiking neural networks," in *International Joint Conference on Neural Networks (IJCNN)*, vol. 3, pp. 1772–1777, 2001.
- [71] W. Maass and H. Markram, "On the computational power of recurrent circuits of spiking neurons," tech. rep., The Weizmann Institute for Science, 2002.
- [72] J. Hertz, A. Krogh, and R. Palmer., *Introduction to the Theory of Neural Networks*. Addison-Wesley, 1991.
- [73] W. Gerstner and J. V. Hemmen, "Associative memory in a network of 'spiking' neurons," *Network: Computation in Neural Systems*, vol. 3, no. 2, pp. 139–164, 1992.
- [74] F. T. Sommer and T. Wennekers, "Associative memory in networks of spiking neurons," *Neural Networks*, vol. 14, no. 6-7, pp. 825–834, 2001.
- [75] M. Zamani, A. Sadeghian, and S. Chartier, "A bidirectional associative memory based on cortical spiking neurons using temporal coding," in *International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, July 2010.
- [76] D. J. Amit and G. Mongillo, "Spike-driven synaptic dynamics generating working memory states," *Neural Computation*, vol. 15, pp. 565–596, May 2003.
- [77] G. Mongillo, O. Barak, and M. Tsodyks, "Synaptic theory of working memory," *Science*, vol. 319, no. 5869, pp. 1543–1546, 2008.
- [78] B. Szatmary and E. M. Izhikevich, "Spike-timing theory of working memory," *PLoS Computational Biology*, vol. 6, p. e1000879, Aug. 2010.

- [79] C. Linster and T. A. Cleland, “Decorrelation of odor representations via spike timing-dependent plasticity,” *Frontiers in Computational Neuroscience*, vol. 4, no. 157, 2010.
- [80] E. Lumer, “Effects of spike timing on winner-take-all competition in model cortical circuits,” *Neural Computation*, vol. 12, pp. 181–94, Jan. 2000.
- [81] J. DZ and H. S. Seung, “Fast computation with spikes in a recurrent neural network,” *Physical Review E*, vol. 65, May 2002.
- [82] X. J. Wang, “Decision making in recurrent neuronal circuits,” *Neuron*, vol. 60, no. 2, pp. 215–34, 2008.
- [83] E. Bienenstock, “A model of neocortex,” *Network: Computation in Neural Systems*, vol. 6, no. 2, pp. 179–224, 1995.
- [84] C. Koch, *Biophysics of Computation: Information Processing in Single Neurons*. Oxford University Press, 1999.
- [85] T. V. Bliss and T. Lomo, “Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path,” *Journal of Physiology*, vol. 232, no. 2, pp. 331–56, 1973.
- [86] D. Hebb, *The organization of behavior*. Wiley, 1949.
- [87] E. L. Bienenstock, L. N. Cooper, and P. W. Munro, “Neurocomputing: Foundations of research,” ch. Theory for the Development of Neuron Selectivity: Orientation Specificity and Binocular Interaction in Visual Cortex, pp. 437–455, Cambridge, MA, USA: MIT Press, 1988.
- [88] G. Q. Bi and M. M. Poo, “Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type,” *The Journal of Neuroscience*, vol. 18, no. 24, pp. 10464–10472, 1998.
- [89] H. Markram, J. Luebke, M. Frotscher, and B. Sakmann, “Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps,” *Science*, vol. 275, no. 5297, pp. 213–215, 1997.

- [90] 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. 10420–10429, 2006.
- [91] R. Kempter, W. Gerstner, and J. L. van Hemmen, “Hebbian learning and spiking neurons,” *Physical Review E*, vol. 59, pp. 4498–4514, Apr 1999.
- [92] A. Kepecs, M. C. W. van Rossum, S. Song, and J. Tegner, “Spike-timing-dependent plasticity: common themes and divergent vistas,” *Biological Cybernetics*, vol. 87, no. 5-6, pp. 446–458, 2002.
- [93] G. Q. Bi, “Spatiotemporal specificity of synaptic plasticity: cellular rules and mechanisms,” *Biological Cybernetics*, vol. 87, no. 5-6, pp. 319–332, 2002.
- [94] H. Markram, W. Gerstner, and P. J. Sjöström, eds., *Spike-Timing-Dependent Plasticity: A Comprehensive Overview*. Frontiers Media SA, 2012.
- [95] W. Gerstner and W. M. Kistler, “Mathematical formulations of hebbian learning,” *Biological Cybernetics*, vol. 87, no. 5-6, pp. 404–415, 2002.
- [96] S. Song, K. Miller, and L. Abbott, “Competitive hebbian learning through spike-timing-dependent synaptic plasticity,” *Nature Neuroscience*, vol. 3, pp. 919–926, 2000.
- [97] N. Iannella and S. Tanaka, “Synaptic efficacy cluster formation across the dendrite via stdp,” *Neuroscience Letters*, vol. 403, no. 1-2, pp. 24–29, 2006.
- [98] J. Pfister and W. Gerstner, “Triplets of spikes in a model of spike timing-dependent plasticity,” *The Journal of Neuroscience*, vol. 26, no. 38, pp. 9673–9682, 2006.
- [99] P. Sjöström, G. Turrigiano, and S. Nelson, “Rate, timing, and cooperativity jointly determine cortical synaptic plasticity,” *Neuron*, vol. 32, no. 6, pp. 1149–1164, 2001.

- [100] H. Wang, R. Gerkin, D. Nauen, and G. Bi, “Coactivation and timing dependent integration of synaptic potentiation and depression,” *Nature Neuroscience*, vol. 8, no. 2, pp. 187–193, 2005.
- [101] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2nd ed., 1998.
- [102] S. C. Moore, “Back-propagation in spiking neural networks,” Master’s thesis, University of Bath, 2002.
- [103] S. McKennoch, D. Liu, and L. Bushnell, “Fast modifications of the spikeprop algorithm,” in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 3970–3977, 2006.
- [104] B. Schrauwen and J. Van Campenhout, “Extending spikeprop,” in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, vol. 1, p. 475, July 2004.
- [105] O. Booi and H. tat Nguyen, “A gradient descent rule for spiking neurons emitting multiple spikes,” *Information Processing Letters*, vol. 95, no. 6, pp. 552 – 558, 2005.
- [106] A. Carnell, D. Richardson, and B. Uk, “Linear algebra for time series of spikes,” in *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pp. 363–368, 2005.
- [107] G. Arfken, *Mathematical Methods for Physicists*. Orlando, FL: Academic Press, 3 ed., 1985.
- [108] D. Barber, “Learning in spiking neural assemblies,” in *Advances in Neural Information Processing Systems 15*, pp. 149–156, MIT Press, 2002.
- [109] J. P. Pfister, D. Barber, and W. Gerstner, “Optimal hebbian learning: A probabilistic point of view,” in *Artificial Neural Networks and Neural Information Processing ICANN/ICONIP 2003*, vol. 2714, pp. 92–98, Springer Berlin Heidelberg, 2003.
- [110] J. P. Pfister, T. Toyoizumi, D. Barber, and W. Gerstner, “Optimal spike-timing dependent plasticity for precise action potential firing,” *Neural Computation*, no. 6, pp. 131–1348, 2006.

- [111] A. Belatreche, L. P. Maguire, M. McGinnity, and Q. X. Wu, “A method for supervised training of spiking neural networks.,” in *Proceedings of IEEE Cybernetics Intelligence - Challenges and Advances (CICA)*, pp. 39–44, 2003.
- [112] R. Legenstein, C. Naeger, and W. Maass, “What can a neuron learn with spike-timing-dependent plasticity?,” *Neural Computation*, vol. 17, pp. 2337–2382, Mar. 2005.
- [113] F. Ponulak and A. Kasiński, “Supervised learning in spiking neural networks with resume: Sequence learning, classification, and spike shifting,” *Neural Computation*, vol. 22, pp. 467–510, Feb. 2010.
- [114] W. Schultz, “Getting formal with dopamine and reward,” *Neuron*, vol. 36, no. 2, pp. 241–263, 2002.
- [115] R. Florian, “A reinforcement learning algorithm for spiking neural networks,” in *Proceedings of the Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pp. 299–306, Sept 2005.
- [116] D. Baras and R. Meir, “Reinforcement learning, spike-time-dependent plasticity, and the bcm rule,” *Neural Computation*, vol. 19, no. 8, pp. 2245–2279, 2007.
- [117] R. V. Florian, “Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity,” *Neural Computation*, vol. 6, no. 6, pp. 1468–1502, 2007.
- [118] E. Izhikevich, “Solving the distal reward problem through linkage of stdp and dopamine signaling,” *BMC Neuroscience*, vol. 8, no. Suppl 2, 2007.
- [119] E. Vasilaki, N. Frmaux, R. Urbanczik, W. Senn, and W. Gerstner, “Spike-based reinforcement learning in continuous state and action space: When policy gradient methods fail,” *PLoS Computational Biology*, vol. 5, no. 12, p. e1000586, 2009.
- [120] R. Legenstein, D. Pecevski, and W. Maass, “A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback,” *PLoS Computational Biology*, vol. 4, no. 10, p. e1000180, 2008.

- [121] J. Seo, B. Brezzo, Y. Liu, and B. e. a. Parker, “A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons,” in *Proceedings of the IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–4, Sept 2011.
- [122] C. Mayr, J. Partzsch, M. Noack, and R. Schuffny, “Live demonstration: Multiple-timescale plasticity in a neuromorphic system,” in *Proceedings IEEE International Symposium on Circuits and Systems (ISCAS)*, 2013.
- [123] S. B. Furber, F. Gallupi, S. Temple, and L. A. Plana, “The spinnaker project: A massively-parallel computer architecture for neural simulations,” *Proceedings of the IEEE*, vol. 102, 2014.
- [124] P. A. Merolla, J. V. Arthur, and R. A. I. et al., “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science*, vol. 345, no. 6197, pp. 668–673, 2014.
- [125] B. Benjamin, P. Gao, and E. e. a. Mcquinn, “Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations,” *Proceedings of the IEEE*, vol. 102, pp. 699–716, May 2014.
- [126] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri, “A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses,” *Frontiers in neuroscience*, vol. 9, 2015.
- [127] M. R. Azghadi, S. Moradi, D. B. Fasnacht, M. S. Ozdas, and G. Indiveri, “Programmable spike-timing-dependent plasticity learning circuits in neuromorphic VLSI architectures,” *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 12, no. 2, p. 17, 2015.
- [128] C. Mead, *Analog VLSI and Neural Systems*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.
- [129] G. Indiveri, B. L. Barranco, and T. J. H. et al., “Neuromorphic silicon neuron circuits,” *Frontiers in Neuroscience*, vol. 5, May 2011.
- [130] C. Bartolozzi and G. Indiveri, “Synaptic dynamics in analog VLSI,” *Neural Computation*, vol. 19, no. 10, pp. 2581–2603, 2007.

- [131] S. Furber, D. Lester, L. Plana, J. Garside, E. Painkras, S. Temple, and A. Brown, "Overview of the spinnaker system architecture," *IEEE Transactions on Computers*, vol. 62, pp. 2454–2467, Dec. 2013.
- [132] J. Y. Chen, "Gpu technology trends and future requirements," in *Electron Devices Meeting (IEDM), 2009 IEEE International*, pp. 1–6, Dec 2009.
- [133] A. Bofill-I-Petit and A. Murray, "Synchrony detection and amplification by silicon neurons with stdp synapses," *IEEE Transactions on Neural Networks*, vol. 15, pp. 1296–1304, Sep. 2004.
- [134] G. Indiveri, E. Chicca, and R. Douglas, "A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity," *IEEE Transactions on Neural Networks*, vol. 17, pp. 211–221, Jan 2006.
- [135] J. Schemmel, A. Grubl, K. Meier, and E. Mueller, "Implementing synaptic plasticity in a VLSI spiking neural network model," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, 2006.
- [136] J. V. Arthur and K. Boahen, "Learning in silicon: Timing is everything," in *Advances in Neural Information Processing Systems* (ong, ed.), 2006.
- [137] T. Koickal, A. Hamilton, S. L. Tan, J. Covington, J. Gardner, and T. Pearce, "Analog VLSI circuit implementation of an adaptive neuro-morphic olfaction chip," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, pp. 60–73, Jan 2007.
- [138] H. Tanaka, "A CMOS spiking neural network circuit with symmetric/asymmetric stdp function," *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, vol. E92A, no. 7, pp. 1690–1698, 2009.
- [139] C. Mayr, M. Noack, J. Partzsch, and R. Schuffny, "Replicating experimental spike and rate based neural learning in CMOS," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, 2010.

- [140] Y. Meng, K. Zhou, J. Monzon, and C.-S. Poon, “Iono-neuromorphic implementation of spike-timing-dependent synaptic plasticity,” in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 7274–7277, Aug 2011.
- [141] G. Rachmuth, H. Shouval, M. Bear, and C. Poon, “A biophysically-based neuromorphic model of spike rate- and timing-dependent plasticity,” in *Proc. Nat. Acad. Sci. USA*, pp. e1266–e1274, 2011.
- [142] R. A. Silver, “Neuronal arithmetic,” *Nature Reviews Neuroscience*, vol. 11, pp. 474–489, 2010.
- [143] L. A. Jeffress, “A place theory of sound localization,” *Journal of Comparative and Physiological Psychology*, vol. 41, no. 1, pp. 35–39, 1948.
- [144] M. Larkum, J. Zhu, and B. Sakmann, “A new cellular mechanism for coupling inputs arriving at different cortical layers,” *Nature* 398, no. 6725, pp. 338–341, 1999.
- [145] F. Gabbiani, H. Krapp, C. Koch, and G. Laurent, “Multiplicative computation in a visual neuron sensitive to looming,” *Nature* 420, pp. 320–324, 2002.
- [146] G. Elston, “Cortex, cognition and the cell: new insights into the pyramidal neuron and prefrontal function,” *Cerebral Cortex* 13, vol. 13, no. 11, pp. 1124–1138, 2003.
- [147] N. Spruston, “Pyramidal neurons: dendritic structure and synaptic integration,” *Nature Reviews Neuroscience*, vol. 9, no. 3, p. 206221, 2008.
- [148] S. Cash and R. Yuste, “Linear summation of excitatory inputs by cal pyramidal neurons,” *Neuron*, vol. 22, no. 2, pp. 383–394, 1999.
- [149] R. Yuste and R. Urban, “Dendritic spines and linear networks,” *Journal of Physiology*, vol. 98, no. 4-6, pp. 479–486, 2004.
- [150] J. S. Albus, “A theory of cerebellar function,” *Mathematical Biosciences*, vol. 10, no. 1-2, pp. 25–61, 1971.

- [151] A. J. Bell and T. J. Sejnowski, “An information-maximization approach to blind separation and blind deconvolution,” *Neural Computation*, vol. 7, no. 6, pp. 1129–1159, 1995.
- [152] F. Crick, “Neurocomputing: Foundations of research,” ch. Function of the Thalamic Reticular Complex: The Searchlight Hypothesis, pp. 569–575, Cambridge, MA, USA: MIT Press, 1988.
- [153] S. Grossberg, E. Mingolla, and W. D. Ross, “Visual brain and visual perception: how does the cortex do perceptual grouping?,” *Trends in Neurosciences*, vol. 20, no. 3, pp. 106 – 111, 1997.
- [154] K. C. Neokleous and C. N. Schizas, “Computational modeling of visual selective attention,” *Procedia Computer Science*, vol. 7, no. 0, pp. 244 – 245, 2011.
- [155] B. A. Olshausen and D. J. Field, “How close are we to understanding v1?,” *Neural Computation*, vol. 17, no. 8, pp. 1665–1699, 2005.
- [156] A. Pouget, J. M. Beck, W. J. Ma, and P. E. Latham, “Probabilistic brains: Knowns and unknowns,” *Nature Neuroscience*, vol. 16, pp. 1170–1178, 2013.
- [157] G. Raphael, G. A. Tsianos, and G. E. Loeb, “Spinal-like regulator facilitates control of a two-degree-of-freedom wrist,” *The Journal of Neuroscience*, vol. 30, no. 28, pp. 9431–9444, 2010.
- [158] S. M. Stringer and E. T. Rolls, “Learning transform invariant object recognition in the visual system with multiple stimuli present during training,” *Neural Networks*, vol. 21, no. 7, pp. 888–903, 2008.
- [159] Z. Wei, X. J. Wang, and D. H. Wang, “From distributed resources to limited slots in multiple-item working memory: A spiking network model with normalization,” *The Journal of Neuroscience*, vol. 32, no. 33, pp. 11228–11240, 2012.
- [160] M. E. Larkum, J. J. Zhu, and B. Sakmann, “Dendritic mechanisms underlying the coupling of the dendritic with the axonal action potential initiation zone of adult rat layer 5 pyramidal neurons,” *Journal of Physiology*, vol. 533, no. Pt 2, pp. 447–466, 2001.

- [161] M. E. Larkum, T. Nevian, M. Sandler, A. Polsky, and J. Schiller, “Synaptic integration in tuft dendrites of layer 5 pyramidal neurons: A new unifying principle,” *Science*, vol. 325, no. 5941, pp. 756–760, 2009.
- [162] N. Xu, M. T. Harnett, and S. R. W. et al., “Nonlinear dendritic integration of sensory and motor input during an active sensing task,” *Nature*, vol. 492, no. 7428, p. 247251, 2012.
- [163] S. D. Antic, W. L. Zhou, A. R. Moore, S. M. Short, and K. D. Ikonomu, “The decade of the dendritic NMDA spike,” *Journal of Neuroscience Research*, vol. 88, no. 14, pp. 2991–3001, 2010.
- [164] T. Branco and M. Hausser, “Synaptic integration gradients in single cortical pyramidal cell dendrites,” *Neuron*, vol. 69, no. 5, pp. 885–892, 2011.
- [165] 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, p. e1002599, 2012.
- [166] M. Jadi, A. Polsky, J. Schiller, and B. W. Mel, “Location-dependent effects of inhibition on local spiking in pyramidal neuron dendrites,” *PLoS Computational Biology*, vol. 8, no. 6, p. e1002550, 2012.
- [167] K. A. Archie and B. W. Mel, “A model for intradendritic computation of binocular disparity,” *Nature Neuroscience*, vol. 3, pp. 54–63, 2000.
- [168] B. F. Behabadi and B. W. Mel, “Mechanisms underlying subunit independence in pyramidal neuron dendrites,” *Proc. Nat. Acad. Sci. USA*, vol. 111, no. 1, pp. 498–503, 2013.
- [169] A. Borst and T. Euler, “Seeing things in motion: Models, circuits, and mechanisms,” *Neuron*, vol. 71, no. 6, pp. 974–994, 2011.
- [170] C. Koch, T. Poggio, and V. Torre, “Nonlinear interactions in a dendritic tree: Localization, timing, role in information processing,” *Proc. Nat. Acad. Sci. USA*, vol. 80, no. 9, pp. 2799–2802, 1983.
- [171] H. B. Barlow and W. R. Levick, “The mechanism of directionally selective units in rabbits retina,” *Journal of Physiology*, vol. 178, no. 3, p. 477504, 1965.

- [172] G. Shepherd and R. Brayton, “Logic operations are properties of computer-simulated interactions between excitable dendritic spines,” *Neuroscience*, vol. 21, no. 1, pp. 151–165, 1987.
- [173] A. M. Zador, B. J. Claiborne, and T. H. Brown, “Nonlinear pattern separation in single hippocampal neurons with active dendritic membrane,” in *Advances in Neural Information Processing Systems 4*, 1991.
- [174] B. W. Mel, “MURPHY: A robot that learns by doing,” *Proceedings of the Neural Information Processing Systems Conference (NIPS)*, p. 544553, 1987.
- [175] B. W. Mel, “A connectionist model may shed light on neural mechanisms for visually guided reaching,” *Journal of Cognitive Neuroscience*, vol. 3, no. 3, pp. 273–292, 1991.
- [176] 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.
- [177] G. Katona, A. Kasza’s, G. F. Turi, N. Hajos, G. Tamas, E. S. Vizi, , and B. Rozsa, “Roller coaster scanning reveals spontaneous triggering of dendritic spikes in ca1 interneurons,” in *Proc. Nat. Acad. Sci. USA*, vol. 108, pp. 2148–2153, 2011.
- [178] B. W. Mel, “Synaptic integration in an excitable dendritic tree,” *Journal of Neurophysiology*, vol. 70, no. 3, pp. 1086–1101, 1993.
- [179] B. W. Mel, “NMDA-based pattern discrimination in a modeled cortical neuron,” *Neural Computation*, vol. 4, no. 4, pp. 502–517, 1992.
- [180] J. K. Makara and J. C. Magee, “Variable dendritic integration in hippocampal ca3 pyramidal neurons,” *Neuron*, vol. 80, no. 6, pp. 1438–1450, 2013.
- [181] B. W. Mel, “A connectionist learning model for 3-d mental rotation, zoom, pan,” in *Proceedings of the 8th Annual Conference of Cognitive Science Society (CogSci)*, pp. 562–571, 1986.
- [182] P. Poirazi, T. Brannon, and B. Mel, “Pyramidal neuron as two-layer neural network,” *Neuron*, vol. 37, no. 6, p. 989999, 2003.

- [183] A. Losonczy and J. Magee, “Integrative properties of radial oblique dendrites in hippocampal ca1 pyramidal neurons,” *Neuron* 50, pp. 291–307, 2006.
- [184] H. V. Oviedo and A. D. Reyes, “Integration of subthreshold and suprathreshold excitatory barrages along the somatodendritic axis of pyramidal neurons,” *PLoS ONE*, vol. 7, p. e33831, Mar. 2012.
- [185] C. Beaulieu and M. Colonnier, “A laminar analysis of the number of round-asymmetrical and flat symmetrical synapses on spines, dendritic trunks, cell bodies in area 17 of the cat,” *Journal of Comparative Neurology*, vol. 231, no. 2, pp. 180–189, 1985.
- [186] G. N. Elston and J. DeFelipe, “Spine distribution in cortical pyramidal cells: A common organizational principle across species,” *Progress in Brain Research*, vol. 136, pp. 109–133, 2002.
- [187] W. Rall and J. Rinzel, “Branch input resistance and steady attenuation for input to one branch of a dendritic neuron model,” *Biophysics Journal*, vol. 13, no. 7, pp. 648–687, 1973.
- [188] D. H. Hubel and T. N. Wiesel, “Receptive fields, binocular interaction and functional architecture in the cats visual cortex,” *Journal of Physiology*, vol. 160, no. 1, pp. 106–154, 1962.
- [189] D. J. Heeger, “Normalization of cell responses in cat striate cortex,” *Visual Neuroscience*, vol. 9, no. 2, pp. 181–197, 1992.
- [190] J. Touryan, G. Felsen, and Y. Dan, “Spatial structure of complex cell receptive fields measured with natural images,” *Neuron*, vol. 45, no. 5, pp. 781–791, 2005.
- [191] K. Fukushima, “Neocognitron: A self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.
- [192] M. W. Oram, D. I. Perrett, and J. K. Hietanen, “Directional tuning of motion-sensitive cells in the anterior superior temporal polysensory area of the macaque,” *Experimental Brain Research*, vol. 97, no. 2, pp. 274–294, 1993.

- [193] B. W. Mel, “SEEMORE: Combining color, shape, texture histogramming in a neurally inspired approach to visual object recognition,” *Neural Computation*, vol. 9, no. 4, pp. 777–804, 1997.
- [194] C. Cadieu, M. Kouh, A. Pasupathy, C. E. Connor, M. Riesenhuber, and T. Poggio, “A model of v4 shape selectivity and invariance,” *Journal of Phy*, vol. 98, no. 3, pp. 1733–1750, 2007.
- [195] N. C. Rust and J. J. DiCarlo, “Selectivity and tolerance (invariance) both increase as visual information propagates from cortical area v4 to it,” *The Journal of Neuroscience*, vol. 30, no. 39, pp. 12978–12995, 2010.
- [196] J. H. Bollmann and F. Engert, “Subcellular topography of visually driven dendritic activity in the vertebrate visual system,” *Neuron*, vol. 61, no. 6, pp. 895–905, 2009.
- [197] T. Euler, P. B. Detwiler, and W. Denk, “Directionally selective calcium signals in dendrites of starburst amacrine cells,” *Nature*, vol. 418, pp. 845–852, 2002.
- [198] N. L. Iannella, T. Launey, and S. Tanaka, “Spike timing-dependent plasticity as the origin of the formation of clustered synaptic efficacy engrams,” *Frontiers in Computational Neuroscience*, vol. 4, 2010.
- [199] D. Lee, B. J. Lin, , and A. K. Lee, “Hippocampal place fields emerge upon single-cell manipulation of excitability during behavior,” *Science*, vol. 337, no. 6096, pp. 849–853, 2012.
- [200] R. Legenstein and W. Maass, “Branch-specific plasticity enables self-organization of nonlinear computation in single neurons,” *The Journal of Neuroscience*, vol. 31, no. 30, pp. 10787–10802, 2011.
- [201] B. Sivyer and S. R. Williams, “Direction selectivity is computed by active dendritic integration in retinal ganglion cells,” *Nature Neuroscience*, vol. 16, pp. 1848–1856, 2013.
- [202] X. Wu and B. Mel, “Capacity-enhancing learning rules in a medial temporal lobe online learning model,” *Neuron*, vol. 62, no. 1, pp. 31–41, 2009.

- [203] M. S. Goldman, J. H. Levine, G. Major, D. W. Tank, and H. S. Seung, “Robust persistent neural activity in a model integrator with multiple hysteretic dendrites per neuron,” *Cerebral Cortex*, vol. 13, no. 11, pp. 1185–1195, 2003.
- [204] E. T. Vu and F. B. Krasne, “Evidence for a computational distinction between proximal and distal neuronal inhibition,” *Science*, vol. 255, no. 5052, pp. 1710–1712, 1992.
- [205] 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.
- [206] A. Gidon and I. Segev, “Principles governing the operation of synaptic inhibition in dendrites,” *Neuron*, vol. 75, no. 2, pp. 330 – 341, 2012.
- [207] A. Thomson, J. Deuchars, and D. West, “Single axon excitatory postsynaptic potentials in neocortical interneurons exhibit pronounced paired pulse facilitation,” *Neuroscience* 54, no. 2, p. 347360, 1993.
- [208] H. Markram and M. Tsodyks, “Redistribution of synaptic efficacy between neocortical pyramidal neurons,” *Nature* 382, vol. 382, no. 6594, pp. 807–810, 1996.
- [209] J. Varela, K. Sen, J. Gibson, J. Fost, L. Abbott, and S. Nelson, “A quantitative description of short-term plasticity at excitatory synapses in layer 2/3 of rat primary visual cortex,” *The Journal of Neuroscience*, vol. 17, no. 20, pp. 7926–7940, 1997.
- [210] C. M. Hempel, K. H. Hartman, X. J. Wang, G. G. Turrigiano, and S. B. Nelson, “Multiple forms of short-term plasticity at excitatory synapses in rat medial prefrontal cortex,” *Journal of Neurophysiology*, vol. 83, no. 5, pp. 3031–3041, 2000.
- [211] P. Werbos, “Backpropagation through time: what it does and how to do it,” *Proceedings of the IEEE*, vol. 78, pp. 1550 –1560, Oct. 1990.
- [212] A. F. Atiya and A. G. Parlos, “New results on recurrent network training: Unifying the algorithms and accelerating convergence,” *IEEE Transactions on Neural Networks*, vol. 11, pp. 697–709, May 2000.

- [213] G. V. Puskorius and L. A. Feldkamp, “Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks,” *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 279–297, 1994.
- [214] S. Ma and C. Ji, “Fast training of recurrent networks based on the em algorithm,” *IEEE Transactions on Neural Networks*, vol. 9, pp. 11–26, Jan. 1998.
- [215] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, pp. 1735–1780, Nov. 1997.
- [216] P. Auer, H. Burgsteiner, and W. Maass, “A learning rule for very simple universal approximators consisting of a single layer of perceptrons,” *Neural Networks*, vol. 21, no. 5, pp. 786–795, 2008.
- [217] Y. Zhang and K. Wang, “The application of liquid state machines in robot path planning,” *Journal of Computers*, vol. 4, no. 11, pp. 1182–1186, 2009.
- [218] P. Jakimovski and H. R. Schmidtke, “Delayed synapses: An LSM model for studying aspects of temporal context in memory,” in *Proceedings of the 7th International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT)*, pp. 138–144, 2011.
- [219] R. Legenstein, H. Markram, and W. Maass, “Input prediction and autonomous movement analysis in recurrent circuits of spiking neurons,” *Rev Neurosci*, vol. 14, no. 1-2, pp. 5–19, 2003.
- [220] S. Hussain, R. Gopalakrishnan, A. Basu, and S. C. Liu, “Morphological Learning: Increased Memory Capacity of Neuromorphic Systems with Binary Synapses Exploiting AER Based Reconfiguration,” in *IEEE Intl. Joint Conference on Neural Networks (IJCNN)*, pp. 1 – 7, Aug. 2013.
- [221] E. Hourdakis and P. Trahanias, “Improving the Classification Performance of Liquid State Machines Based on the Separation Property,” in *Engineering Applications of Neural Networks* (L. Iliadis and C. Jayne, eds.), vol. 363 of *IFIP Advances in Information and Communication Technology*, pp. 52–62, Springer Berlin Heidelberg, 2011.

- [222] J. Vreeken, “On real-world temporal pattern recognition using Liquid State Machines,” Master’s thesis, Utrecht University (NL), 2004.
- [223] Y. Zhang, P. Li, Y. Jin, and Y. Choe, “A digital liquid state machine with biologically inspired learning and its application to speech recognition,” *Neural Networks and Learning Systems, IEEE Transactions on*, 2015.
- [224] F. Xue, Z. Hou, and X. Li, “Computational capability of liquid state machines with spike-timing-dependent plasticity,” *Neurocomputing*, vol. 122, no. 0, pp. 324–329, 2013.
- [225] H. Ju, J. X. Xu, E. Chong, and A. M. J. Vandongen, “Effects of synaptic connectivity on liquid state machine performance,” *Neural Networks*, vol. 38, pp. 39–51, Feb. 2013.
- [226] E. Hourdakis and P. Trahanias, “Use of the separation property to derive liquid state machines with enhanced classification performance,” *Neurocomput.*, vol. 107, pp. 40–48, May 2013.
- [227] S. Schliebs, M. Fiasché, and N. Kasabov, “Constructing robust liquid state machines to process highly variable data streams,” in *Proceedings of the 22nd International Conference on Artificial Neural Networks and Machine Learning (ICANN)*, pp. 604–611, 2012.
- [228] G. M. Wojcik, “Electrical parameters influence on the dynamics of the hodgkinhuxley liquid state machine,” *Neurocomputing*, vol. 79, pp. 68–74, 2012.
- [229] S. Notley and A. Gruning, “Improved spike-timed mappings using a tri-phasic spike timing-dependent plasticity rule,” in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, Jun. 2012.
- [230] A. Frid, H. Hazan, and L. Manevitz, “Temporal pattern recognition via temporal networks of temporal neurons,” in *Proceedings of the 27th Convention of Electrical Electronics Engineers in Israel (IEEEI)*, pp. 1–4, Nov. 2012.

- [231] H. Hazan and L. M. Manevitz, “Topological constraints and robustness in liquid state machines,” *Expert Syst. Appl.*, vol. 39, pp. 1597–1606, Feb. 2012.
- [232] S. Schliebs, A. Mohemmed, and N. Kasabov, “Are probabilistic spiking neural networks suitable for reservoir computing?,” in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 3156–3163, Jul. 2011.
- [233] R. L. Ortman, K. Venayagamoorthy, and S. M. Potter, “Input separability in living liquid state machines,” in *Proceedings of the 10th International Conference on Adaptive and Natural Computing Algorithms (ICANNGA) - Volume Part I*, 2011.
- [234] F. RhéAume, D. Grenier, and L. Bossé, “Multistate combination approaches for liquid state machine in supervised spatiotemporal pattern classification,” *Neurocomputing*, vol. 74, pp. 2842–2851, Oct. 2011.
- [235] K. El-Laithy and M. Bogdan, “On the capacity of transient internal states in liquid-state machines,” in *Proceedings of the 21st International Conference on Artificial Neural Networks (ICANN) - Volume Part II*, pp. 56–63, 2011.
- [236] C. Kello and M. Mayberry, “Critical branching neural computation,” in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, Jul. 2010.
- [237] D. Norton and D. Ventura, “Improving liquid state machines through iterative refinement of the reservoir,” *Neurocomput.*, vol. 73, pp. 2893–2904, Oct. 2010.
- [238] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout, “Isolated word recognition with the liquid state machine: a case study,” *Inf. Process. Lett.*, vol. 95, pp. 521–528, Sep. 2005.
- [239] E. Goodman and D. Ventura, “Spatiotemporal pattern recognition via liquid state machines,” in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 3848–3853, 2006.
- [240] S. Schliebs and D. Hunt, “Continuous classification of spatio-temporal data streams using liquid state machines,” in *Proceedings of the 19th*

- International Conference on Neural Information Processing (ICONIP)*
- Volume Part IV, pp. 626–633, 2012.
- [241] D. Probst, W. Maass, H. Markram, and M. Gewaltig, “Liquid computing in a simplified model of cortical layer iv: Learning to balance a ball,” in *Proceedings of the 22nd International Conference on Artificial Neural Networks and Machine Learning (ICANN) - Volume Part I*, pp. 209–216, 2012.
- [242] Y. Elmir, Z. Elberrichi, and R. Adjoudjelmir, “Liquid state machine based fingerprint identification,” *Australian Journal of Basic & Applied Sciences*, vol. 5, no. 5, pp. 857–865, 2011.
- [243] H. Ju, J. X. Xu, and A. M. J. Vandongen, “Classification of musical styles using liquid state machines,” in *Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, Jul. 2010.
- [244] T. Natschläger, H. Markram, and W. Maass, *Computer models and analysis tools for neural microcircuits*, ch. 9. Kluwer Academic Publishers (Boston), 2002.
- [245] C. Bartolozzi and G. Indiveri, “Synaptic Dynamics in Analog VLSI,” *Neural Computation*, vol. 19, no. 10, pp. 2581–2603, 2007.
- [246] Y. Yao, “On complexity issues of online learning algorithms,” *IEEE Transactions on Information Theory*, vol. 56, pp. 6470–6481, Dec. 2010.
- [247] S. Roy, A. Banerjee, and A. Basu, “Liquid state machine with dendritically enhanced readout for low-power, neuromorphic VLSI implementations,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, pp. 681–695, Oct. 2014.
- [248] S. Roy, A. Basu, and S. Hussain, “Hardware efficient, Neuromorphic Dendritically Enhanced Readout for Liquid State Machines,” in *Proceedings of the IEEE Biomedical Circuits and Systems (BioCAS)*, pp. 302–305, Nov 2013.
- [249] R. Johansson and I. Birznieks, “First spikes in ensembles of human tactile afferents code complex spatial fingertip events,” *Nature Neuroscience*, vol. 7, no. 3, pp. 170–177, 2004.

- [250] M. Ambard and S. Rotter, “Support vector machines for spike pattern classification with a leaky integrate-and-fire neuron,” *Frontiers in Computational Neuroscience*, vol. 6, 2012.
- [251] Z. Rotman and V. A. Klyachko, “Role of synaptic dynamics and heterogeneity in neuronal learning of temporal code,” *Journal of Neurophysiology*, vol. 110, no. 10, pp. 2275–2286, 2013.
- [252] S. Hussain, A. Basu, M. Wang, and T. Hamilton, “DELTRON: Neuro-morphic architectures for delay based learning,” in *Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 304–307, Dec 2012.
- [253] B. Zhao, R. Ding, S. Chen, B. Linares-Barranco, and H. Tang, “Feed-forward categorization on aer motion events using cortex-like features in a spiking neural network,” *IEEE Transactions on Neural Networks and Learning Systems*, 2014.
- [254] M. Riesenhuber and T. Poggio, “Hierarchical models of object recognition in cortex,” *Nature Neuroscience*, vol. 2, no. 11, pp. 1019–1025, 1999.
- [255] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, “Robust object recognition with cortex-like mechanisms,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 411–426, Mar. 2007.
- [256] T. S. Gotarredona and B. L. Barranco, “The MNIST-DVS database.” [Online]. Available: <http://www2.imse-cnm.csic.es/caviar/MNISTDVS.html>.
- [257] R. Gutig, T. Gollisch, H. Sompolinsky, and M. Meister, “Computing complex visual features with retinal spike times,” *PLoS ONE*, vol. 8, p. e53063, Jan. 2013.
- [258] H. Tang, Q. Yu, and K. Tan, “Learning real-world stimuli by single-spike coding and tempotron rule,” in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, Jun. 2012.

- [259] C. Baldassi, A. Braunstein, and R. Zecchina, “Theory and learning protocols for the material tempotron model,” *Journal of Statistical Mechanics: Theory and Experiment*, p. P12013, Dec. 2013.
- [260] R. Rubin, R. Monasson, and H. Sompolinsky, “Theory of spike timing-based neural classifiers,” *Physical Review Letters*, vol. 105, p. 218102, Nov 2010.
- [261] R. V. Florian, “Tempotron-like learning with resume,” in *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pp. 368–375, Springer Berlin Heidelberg, 2008.
- [262] W. Lee, J. Cabibihan, and N. Thakor, “Bio-mimetic strategies for tactile sensing,” in *Proceedings of the IEEE SENSORS*, pp. 1–4, Nov. 2013.
- [263] R. V. Florian, “The chronotron: a neuron that learns to fire temporally precise spike patterns,” *PLoS ONE*, vol. 7, no. 8, p. e40233, 2013.
- [264] J. Tapson, G. Cohen, S. Afshar, K. M. Stiefel, Y. Buskila, R. Wang, T. J. Hamilton, and A. van Schaik, “Synthesis of neural networks for spatio-temporal spike pattern recognition and processing,” *Frontiers in Neuroscience*, vol. 7, no. 153, 2013.
- [265] W. Maass, “Neural computation with winner-take-all as the only non-linear operation,” in *Advances in Neural Information Processing Systems 12*, pp. 293–299, MIT Press, 2000.
- [266] W. Maass, “On the computational power of winner-take-all,” *Neural Computation*, vol. 12, no. 11, pp. 2519–2535, 2000.
- [267] L. Itti, C. Koch, and E. Niebur, “A model of saliency-based visual attention for rapid scene analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 1254–1259, Nov. 1998.
- [268] S. Kaski and T. Kohonen, “Winner-take-all networks for physiological models of competitive learning,” *Neural Networks*, vol. 7, no. 67, pp. 973 – 984, 1994.
- [269] J. Barnden and K. Srinivas, “Temporal winner-take-all networks: a time-based mechanism for fast selection in neural networks,” *IEEE Transactions on Neural Networks*, vol. 4, pp. 844–853, Sep. 1993.

- [270] Y. He and E. Sanchez-Sinencio, “Min-net winner-take-all CMOS implementation,” *Electronics Letters*, vol. 29, pp. 1237–1239, Jul. 1993.
- [271] J. Starzyk and X. Fang, “CMOS current mode winner-take-all circuit with both excitatory and inhibitory feedback,” *Electronics Letters*, vol. 29, pp. 908–910, May 1993.
- [272] T. Serrano and B. L. Barranco, “A modular current-mode high-precision winner-take-all circuit,” in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 5, pp. 557–560, May 1994.
- [273] G. Indiveri, “Winner-take-all networks with lateral excitation,” *Analog Integrated Circuits and Signal Processing*, vol. 13, no. 1-2, pp. 185–193, 1997.
- [274] G. Indiveri, “A current-mode hysteretic winner-take-all network, with excitatory and inhibitory coupling,” *Analog Integrated Circuits and Signal Processing*, vol. 28, no. 3, pp. 279–291, 2001.
- [275] S. C. Liu, “A normalizing aVLSI network with controllable winner-take-all properties,” *Analog Integrated Circuits and Signal Processing*, vol. 31, no. 1, pp. 47–53, 2002.
- [276] M. Oster, R. Douglas, and S. C. Liu, “Computation with spikes in a winner-take-all network,” *Neural Computation*, vol. 21, pp. 2437–2465, Sep. 2009.
- [277] J. L. McKinstry and G. M. Edelman, “Temporal sequence learning in winner-take-all networks of spiking neurons demonstrated in a brain-based device,” *Frontiers in Neurorobotics*, vol. 7, no. 10, 2013.
- [278] T. Masquelier, R. Guyonneau, and S. J. Thorpe, “Competitive STDP-based spike pattern learning,” *Neural Computa*, vol. 21, pp. 1259–1276, May 2009.
- [279] A. Delorme, L. Perrinet, and S. J. Thorpe, “Networks of integrate-and-fire neurons using rank order coding B: Spike timing dependent plasticity and emergence of orientation selectivity,” *Neurocomputing*, vol. 38, no. 40, pp. 539–545, 2001.

- [280] R. Guyonneau, R. VanRullen, and S. J. Thorpe, “Temporal codes and sparse representations: A key to understanding rapid processing in the visual system,” *Journal of Physiology-Paris*, vol. 98, no. 4-6, pp. 487 – 497, 2004.
- [281] T. Masquelier and S. J. Thorpe, “Unsupervised learning of visual features through spike timing dependent plasticity,” *PLoS Computational Biology*, vol. 3, p. e31. doi:10.1371/journal.pcbi.0030031, 02 2007.
- [282] W. Gerstner, R. Ritz, and J. L. V. Hemmen, “Why spikes? hebbian learning and retrieval of time-resolved excitation patterns,” *Biological Cybernetics*, vol. 69, no. 5-6, pp. 503–515, 1993.
- [283] M. Yoshioka, “Spike-timing-dependent learning rule to encode spatiotemporal patterns in a network of spiking neurons,” *Physical Review E*, vol. 65, p. 011903, Dec 2001.
- [284] B. Nessler, M. Pfeiffer, L. Buesing, and W. Maass, “Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity,” *PLoS Computational Biology*, vol. 9, no. 4, p. e1003037, 2013.
- [285] D. Kappel, B. Nessler, and W. Maass, “STDP installs in winner-take-all circuits an online approximation to hidden markov model learning,” *PLoS Computational Biology*, vol. 10, p. e1003511, Mar 2014.
- [286] S. Hussain, S. C. Liu, and A. 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.
- [287] E. Kobatake and K. Tanaka, “Neuronal selectivities to complex object features in the ventral visual pathway of the macaque cerebral cortex,” *Journal of Neurophysiology*, vol. 71, pp. 856–867, Mar. 1994.
- [288] A. Banerjee, A. Bhaduri, S. Roy, S. Kar, and A. Basu, “A current-mode spiking neural classifier with lumped dendritic nonlinearity,” in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, no. May, 2015.

- [289] S. Hussain and A. Basu, “Multiclass classification by adaptive network of dendritic neurons with binary synapses using structural plasticity,” *Frontiers in Neuroscience*, vol. 10, no. 113, 2016.
- [290] J. Hasler and H. B. Marr, “Finding a roadmap to achieve large neuro-morphic hardware systems,” *Frontiers in Neuroscience*, vol. 7, no. 118, 2013.
- [291] P. S. Paolucci, R. Ammendola, A. Biagioni, O. Frezza, F. L. Cicero, A. Lonardo, M. Martinelli, E. Pastorelli, F. Simula, and P. Vicini, “Power, energy and speed of embedded and server multi-cores applied to distributed simulation of spiking neural networks: ARM in NVIDIA tegra vs intel xeon quad-cores,” *CoRR*, vol. abs/1505.03015, 2015.
- [292] 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. 22, pp. 2621–2628, Dec 2014.
- [293] H. Jaeger and H. Haas, “Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication,” *Science*, vol. 304, no. 5667, pp. 78–80, 2004.
- [294] T. Serrano-Gotarredona and B. Linares-Barranco, “A 128x128 1.5% contrast sensitivity 0.9% fpn 3 μ s latency 4mW asynchronous frame-free dynamic vision sensor using transimpedance amplifiers,” *IEEE Journal of Solid-State Circuits*, vol. 48, no. 3, pp. 827–838, 2013.