Spatio-/Spectro-Temporal Pattern Recognition using Evolving Probabilistic Spiking Neural Networks

Kshitij Dhoble

A thesis submitted to
The Auckland University of Technology
in fulfillment of the requirements
for the degree of

Doctor of Philosophy (PhD)

2013

Primary Supervisor: Prof. Nikola Kasabov
Secondary Supervisor: Prof. Giacomo Indiveri

Knowledge Engineering & Discovery Research Institute
Faculty of Design and Creative Technologies
Auckland University of Technology
Auckland, New Zealand

"I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning."

Kshitij

Kshitij Dhoble

This thesis is dedicated to my parents

Thank you for your unconditional love and support through all my walks of life. For inspiring me to be innovative, for motivating me to try new things, for encouraging me to challenge the status quo, to make new discoveries.

Acknowledgments

I have the pleasure to acknowledge some of the many people who have inspired, supported, and educated me over the past three years. First and foremost, I am very grateful to my primary supervisor Prof. Nikola Kasabov who gave me the opportunity to study PhD degree under his supervision. I wish to express my sincere appreciation to my secondary supervisor Dr. Giacomo Indiveri who has provided me with valuable insight into the neuromorphic hardware; his student Fabio Stefanini for the discussions on artificial silicon retina. I would also like to thank the past and present members of KEDRI at the Auckland University of Technology, who provided me with an inquisitive research environment. I am especially grateful to Nuttapod Nuntalid with whom I was able to implement many methods collaboratively along with productive discussions about spiking neural networks; to Stefan Schlibes for his goofy jokes; to Russel Pears and Ammar Mohhemed for their insights and discussions relating to machine learning; to Harya Widiputra, Haza Nuzly, Raphael Hu and Gary Chen for their discussions on how not to do a PhD degree. Also, many thanks to Diana Kassabova for proofreading my thesis.

Special thanks also to Joyce DMello, the soul of Knowledge Engineering and Discovery Research Institute. I am very indebted to her for the encouraging words and her guidance throughout my studies which are highly appreciated. Many thanks to Hien Nguyen, for love, support and the delicious Vietnamese dishes. Last but not least, I am delighted to thank my parents for their tremendous support throughout my entire education. Their insight, wisdom and emotional support have been invaluable for me. My research has been carried out with the partial financial support of AUT-KEDRI PhD Scholarship.

Contents

List of Figures

List of Abbreviations

AER:	Address Event Representation
ANN:	Artificial Neural Network
BPDC:	Backpropagation-Decorrelation
deSNN:	Dynamic Evolving Spiking Neural Network
deSNNr:	Dynamic Evolving Spiking Neural Network Reservoir
DVS:	Dynamic Vision Sensor
ECOS:	Evolving Connectionist Systems
epSNN:	Evolving Probabilistic Spiking Neural Network
epSNNA-s:	epSNNr Architecture for Spectro-Temporal Data
epSNNA-v:	epSNNr Architecture for Spatio-Temporal Data
epSNNr:	Evolving Probabilistic Spiking Neural Network Reservoir
ESN:	Echo State Network

eSNN:	Evolving Spiking Neural Network
GRN:	Gene Regulatory Network
LIF:	Leaky Integrate-and-Fire Neuron
LSM:	Liquid State Machine
NR:	Noisy Reset Neuron Model
NT:	Noisy Threshold Neuron Model
ROSC:	Rank Order Spike Coding
SDSP:	Spike-Driven Synaptic Plasticity
SNN:	Spiking Neural Network
SNT:	Step-wise Noisy Threshold Neuron Model
SOM:	Self-Organizing Map
SSTD:	Spatio- and Spectro-Temporal Data
STDP:	Spike-timing dependent plasticity
STPR:	Spatio-Temporal Pattern Recognition
TSC:	Temporal Spike Coding

Abstract

Video and audio information is spatio- or spectro- (sound/frequency) temporal in nature and processing of such complex Spatio/Spectro Temporal Data (SSTD) is a challenging task in the machine learning domain. SSTD contains both the spatial (space) and temporal (time) components and most often both these two components are highly correlated.

Due to the existence of high correlations between these two components, it is essential to process them together. However, many of the existing computational methods either process spatial and temporal components separately, or processing them together then the significant correlation information present in the SSTD is not considered. Comparatively, the brain is capable of performing such tasks in a fast and robust manner. Inspired by the innate cognitive functions of our brain, the proposed study investigates how various biological and cognitive aspects such as learning, evolution and neural information processing tasks can be applied to our computational model. We have shown that this enables efficient data acquisition, processing and learning of complex video and audio patterns thereby resulting in improved classification performance.

This thesis proposes novel frameworks and classification methods employing a class of evolving spiking neural networks (eSNN) called dynamic evolving spiking neural networks (deSNN) along with reservoir computing. In our study, we have shown that using the proposed frameworks results in (1) better classification performance when compared to standalone spiking neural network classifiers such as eSNN, (2) better understanding of the data and the problem being solved, (3) faster SSTD processing due to the online one-pass spike-based computational approaches.

All the frameworks and methods proposed in this thesis have been evaluated on synthetic and real world problems. In order to evaluate the efficacy of the new methodology, initially a pilot experiment has been performed as a benchmark test using a synthetic video dataset, followed by experiments on real world problems relating to motion and sound such as human action recognition and heart sound recognition.

"There are billions of neurons in our brains, but what are neurons? Just cells. The brain has no knowledge until connections are made between neurons. All that we know, all that we are, comes from the way our neurons are connected."

Allen (2009)

1 Introduction

Existing statistical and artificial neural networks (ANN) machine learning approaches fail to model the complex spatio-temporal dynamics optimally. Since they either process spatial and temporal component separately the significant correlation information present in the Spatio and Spectro-Temporal Data (SSTD) is lost. Some of the existing methods that do consider both the space and the time component often input the spatio-temporal and spectro-temporal data to the machine learning algorithm in parts i.e. on a frame-by-frame basis. The historical information / event influences future events, hence they share a level of correlation with the future events occurring over time. Many of the traditional machine learning systems fail to consider the entirety of the SSTD pattern, therefore losing the significant spatio-temporal correlation information. This study takes the current spiking neural network based approach in machine learning to new conceptual and operational levels. We have proposed a spiking neural network (SNN) learning approach utilizing the spatio- and spectro-temporal architectures (epSNNA-v and epSNNA-s) that are expected to successfully achieve this holistic integration of spatio- and spectro-temporal events; they can also be further applied to solving real world problems. These architectures allow us to process continuous SSTD in a faster and computationally efficient manner.

1.1 Rationale and Significance of the Study

In spite of the recent research and development of SNN, there is a significant gap in finding the most effective approach and method for processing SSTD. This research aims at addressing this challenge with the development of a new SSTD modeling technique using evolving probabilistic Spiking Neural Networks (epSNN) for AudioVisual pattern recognition, including epSNN reservoir systems (epSNNr).

Initial studies on reservoir systems such as Liquid State Machine (LSM) have shown promising results. Also, the evolving and stochastic nature of our system will allow the model to adapt to new incoming data and learn incrementally. Moreover, the spikebased computation approach is expected to improve the processing time and accuracy considerably.

Video and audio information are spatio- or spectro- (sound/frequency) temporal in nature and processing such complex SSTD is a challenging task in the machine learning domain. Recent studies (Hamed, Kasabov, Shamsuddin, Widiputra, & Dhoble, 2011; Schliebs, Hamed, & Kasabov, 2011; Mohemmed, Schlibes, Matsuda, & Kasabov, 2012) have shown the SNNs capability of processing the SSTD simultaneously while retaining the significant correlation information between the space and time component. Therefore, we hypothesize that spiking neural network based methods and architectures will be capable of processing SSTD from real-world in a fast, one-pass, on-line and efficient manner.

1.2 Definition and Motivation

The human brain studies date back many years. The recent scientific advances (such as electronics, cognitive and computer science) have made it possible to partially emulate the human brain and its innate cognitive ability. Learning is one such innate cognitive ability which has empowered the living animate entities and especially humans with intelligence. It is demonstrated by the ability to acquire new knowledge and skills that enable them to adapt and survive.

The exact working of the human brain is still only partially understood. It is assumed that the human brain's ability to learn arises from the vast number of neurons and their interconnections. The number of connections for a neuron can range from 1000 to 200000 (Haykin, 1994). The latest study by Azevedo et al. (2009) show that the average human brain (weighing 1,508.91 $\pm$ 299.14 g, age $\approx$ 50 years) has on average about 86.1 $\pm$ 8.1 billion (86.1 x 10^{11}) neurons.

The artificial neural networks are modeled after the most basic unit of the brain called neuron. Due to this there is certain similarity between artificial and biological neurons, and thus a biological counterpart for a component can always be found in an artificial neuron. Comparison between the biological and artificial neurons makes similarities between the two more evident.

Although, many existing traditional machine learning methods perform classification tasks much better than the artificial neural network, its ability is still not comparable to the any of the species biological neural network especially the human brain. Since the ultimate goal of machine learning is to carry out cognitive tasks similar to the human brain, it takes its inspiration from it.

Motivated by the brain's ability to learn autonomously by means of biological neural networks, we are looking to develop biologically inspired methods and architectures, that are designed to mimic some aspects of humans' cognitive learning ability. This study aims not only to further new developments in artificial spiking neural networks (SNN), but also to demonstrate the potential of SNN's, for solving real world problems that involve the modeling of SSTD.

1.3 Research Objectives

Considering the fact that some novel, generic and specific spike-based data processing methods are required for SSTD modeling and pattern recognition, the study is divided into several parts as follows: development of novel SNN based classification methods, development of a novel spatio-temporal architecture, development of a novel spectro-temporal architecture, and applications for visual and audio SSTD processing.

Both methods and architectures can be applied to the real world problems either independently or in combination by using spike-based computation approach. Combining the two architectures with the novel classification methods results in hybrid algorithms (deSNNr and learning deSNNr), which are finally employed to work on real world dataset.

Based on the above discussion, we have arrived at the following research objectives:

- Develop some new classification methods for SSTD based on eSNN;
- Develop a new generic eSNN method for SSTD based on reservoir computing;

- Develop a generic method and a system for spatio-temporal pattern recognition using Address Event Representation (AER). This will allow a model to directly utilize input spikes produced by an artificial silicon retina for preprocessing;
- Develop a software simulator for artificial silicon retina. This will allow the usage of visual data that has not been obtained from the artificial silicon retina;
- Develop a generic method and a system for spectro-temporal pattern recognition;
- Through comprehensive experimental analysis, evaluate the classification performance of the architectures and methods in different combinations.
- Demonstrate the feasibility and applicability of the developed generic architecture and methods by applying them to visual and audio SSTD from real world problems.

1.4 Specific Research Questions

Corresponding to the research objectives, the following specific research questions pertaining to this study have been formulated:

- How to improve/extend the existing eSNN method for application on spatio- / spectro-temporal pattern recognition problems?
- Which spike information encoding scheme will be appropriate for SSTD representation?

- Can the reservoir computing approach improve the classification performance? Will the addition of stochastic neuron models further improve the reservoir performance? What is the optimal parameter setting for the neural networks? Can the reservoir states consisting of spikes be directly utilized by SNN methods for processing?
- Will the developed system be capable of processing the entire SSTD set? How will the developed system perform with data having varying spike-times and temporal length (in milliseconds and minutes)?
- Will the system be capable of performing fast, one-pass, on-line learning?

1.5 Scientific Contribution

Figure 1.5.1 presents a visual summary of datasets, classification methods and generic architectures used and developed in the study. The items included under the architectures and methods branches are the main contributions of the study.

Architectures: This study proposes three new generic architectures for spatio-temporal and spectro-temporal pattern recognition (Figure 1.5.1), namely epSNNr, epSNNA-v and epSNNA-s. Each architecture consists of many modules such as reservoir, STDP learning in reservoir, stochastic neuron models, spike encoding module and learning algorithms. In the later chapters, the generic architectures will be explained in more details.

Methods: Four new spiking neural network based generic classifiers are proposed in this thesis. The proposed deSNNm and deSNNs are extensions of

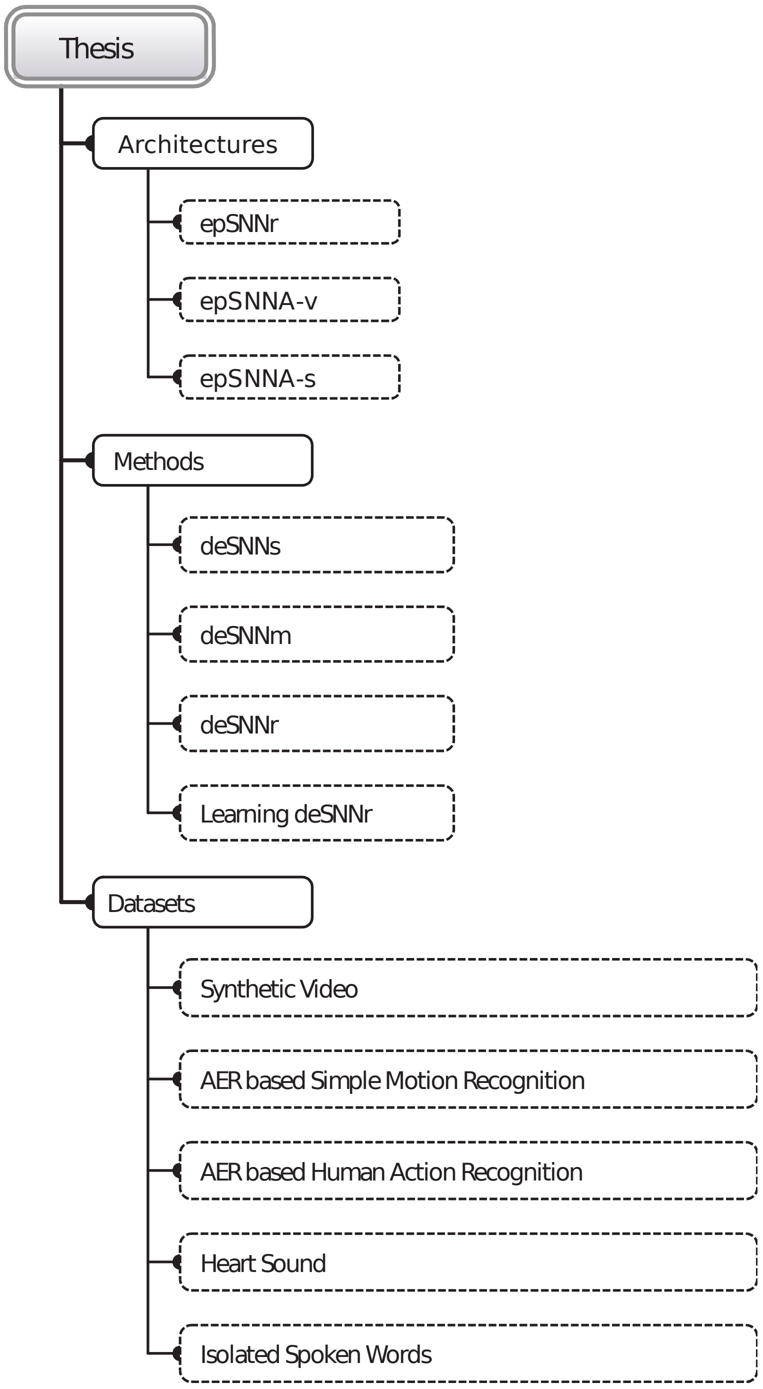


Figure 1.5.1: Architectures and methods developed in this study and datasets used for evaluating their performance.

the eSNN algorithm (S. Wysoski, Benuskova, & Kasabov, 2008). The other two methods called deSNNr and learning deSNNr are hybrid algorithms obtained by combining the earlier proposed architecture and deSNNm/s algorithms.

Datasets: We have used five datasets to evaluate the classification performance of the proposed architectures and methods. It includes one synthetic video dataset for benchmark testing and four real world datasets. The datasets have been described in details in their respective chapters.

1.6 Publications

The content put forth in this thesis was partially published in a number of international conference and journal articles:

- **Dhoble, K.**, Nuntalid, N., Indivery, G., Kasabov, N. (2012) Online Spatio-Temporal Pattern Recognition with Evolving Spiking Neural Networks utilising Address Event Representation, Rank Order, and Temporal Spike Learning, *Proc. WCCI 2012: IEEE World Congress on Computational Intelligence*, IEEE Press, (pp. 17).

- Kasabov, N., **Dhoble, K.**, Nuntalid, N., Indivery, G. (May, 2013) Dynamic evolving spiking neural networks for on-line spatio- and spectro-temporal pattern recognition. *Neural Networks*, Elsevier, Volume 41, (pp. 188-201).

- Kasabov, N., **Dhoble, K.**, Nuntalid, N., & Mohemmed, A. (2011).Evolving probabilistic spiking neural networks for spatio-temporal pattern recognition: A preliminary study on moving object recognition. *ICONIP 2011 - In 18th International Conference on Neural Information Processing*, Springer, Heidelberg. LNCS 7064, pp.230-239, Shanghai, China.

- Mohemmed, A., Matsuda, S., Kasabov, N., & **Dhoble, K.** (2011). Optimization of Spiking Neural Networks with Dynamic Synapses for Spike Sequence Generation using Particle Swarm Optimization. *IJCNN 2011 - In Proceedings of International Joint Conference on Neural Networks*, (pp. 2969-2974), San Jose, California.

- Hamed, H. N. A., Kasabov, K., Shamsuddin, S. M., Widiputra, H., & **Dhoble, K.** (2011). An Extended Evolving Spiking Neural Network Model for Spatio-Temporal Pattern Classification, *IJCNN 2011 - In Proceedings of International Joint Conference on Neural Networks*, (pp. 2653-2656), San Jose, California.

- **Dhoble, K.**, Kasabov, N., Indivery, G. (2013) Dynamic evolving Spiking Neural Networks for Spectro-temporal pattern recognition, *IEEE Transactions on Neural Networks and Learning Systems*, IEEE Press, (in preparation).

1.7 Structure of the thesis

This thesis is organized into twelve chapters. A brief summary of the chapters is presented in this section.

Chapter 1 provides an introduction to the study and its objectives

Chapter 2 presents a literature review covering the theory of neural networks. This is followed by a discussion about the differences between traditional artificial neural networks, artificial spiking neural networks (SNN) and biological neural networks. Various mechanisms of the biological neural networks have also been explained in details in order to show the biological plausibility of the artificial spiking neural network. The main concepts related to spiking neural networks such as neuronal models, spike encoding methods, working memories, learning mechanisms and their applications have been discussed in this chapter.

Chapter 3 reviews recurrent spiking neural network reservoirs structures. It also presents a summary on various reservoir computing approaches.

Chapter 4 proposes new generic architecture called epSNNr. The epSNNr is a Liquid State Machine (LSM) reservoir using various stochastic neural models. A pilot study is carried out using synthetic video dataset to test the feasibility and applicability of the architecture. The classification performance of epSNNr is carried out with various traditional classifiers. Also, we have compared the classification performance of epSNNr using the traditional Leaky Integrate and Fire (LIF) neural model with various stochastic neural models. This pilot study provides us with the feasibility test of using LSM on an entire spatio-temporal data for a pattern recognition task.

Chapter 5 covers the Address Event Representation (AER) approach used in the artificial silicon retina. The workings and advantages of the AER approach for motion recognition are presented. In our spatio-temporal pattern case studies, we have utilized the data obtained from the artificial silicon retina. Therefore, we have discussed the attributes of the data obtained through this method. Also, developed as a part of this study, a software simulator of AER is presented.

Chapter 6 contains one of the main contributions of this thesis which is a novel dynamic evolving spiking neural network (deSNN) method. It is an extension of the evolving spiking neural network (eSNN) proposed by S. Wysoski, Benuskova, and Kasabov (2008). This chapter introduces the eSNN method which is a part of evolving connectionist systems (ECOS) (Kasabov et al., 1998; Kasabov, 2002, 2003; Watts, 2009), followed by the characteristics and specifics of the two deSNN classifiers.

Chapter 7 presents how deSNN classifiers are applied on AER data obtained from the artificial silicon retina. This feasibility study provides us with a working proof on the ability of the deSNN method. Since the output of the AER based artificial silicon retina is in the form of spikes, we show that deSNN method is able to carry out direct spike-time computation instead of the traditional frame-by-frame based computation. Also, a classification performance comparison is carried out between spiking neural network classifiers consisting of eSNN, deSNN and a feed-forward network with spike-driven synaptic plasticity learning rule (SDSP-SNN).

Chapter 8 presents a novel spiking neural network architecture for spatio-temporal pattern recognition (epSNNA-v) along with two new methods namely deSNN reservoir (deSNNr) and learning deSNNr. All the earlier proposed methods, stochastic neuronal models, AER spike encoded data and the reservoir are incorporated into this generic architecture.

Chapter 9 evaluates the performance of the proposed epSNNA-v architecture for spatio-temporal data. Real world human action recognition data acquired from the AER silicon retina is used for performance comparison.

Chapter 10 proposes a new generic architecture for spectro-temporal pattern recognition (epSNNA-s). The proposed generic architecture epSNNA-s is different from epSNNA-v in terms of the spike information encoding scheme.

Chapter 11 evaluates the performance of the proposed epSNNA-s architecture two case studies, namely Heart Sound dataset and Isolated Spoken Words dataset.

Chapter 12 concludes the thesis by summarizing the achievements of the work and contains the overall conclusion. Neuromorphic hardware implementation and applications are also briefly discussed as future directions.

1.8 Summary

This chapter provides the background, motivation, research objectives and research questions addressed in this research.

More background information on the problem of SSTD modeling and pattern recognition can be found from the web page of EV FP7 Marie Curie funded project EvoSpike (http://ncs.ethz.ch/projects/evospike) led by Prof. Nikola Kasabov and Prof. Giacomo Indiveri, in which the candidate (myself) also took part.

The next chapter reviews the main aspects of spiking neural networks (SNN) which are used in this study to derived the generic SNN architectures and methods.

"In the study of brain functions we rely upon a biased, poorly understood, and frequently unpredictable organ in order to study the properties of another such organ; we have to use a brain to study a brain."

Corning and Balaban (1968)

2 Why use Spiking Neural Networks for SSTD?

In order to justify our methodology, it is necessary to become acquainted with various approaches to modeling Spatio / Spectro-Temporal Data (SSTD) and some SNN principles along with the expected benefits and possible inherent limitations. Here we have outlined the past and present research relevant to our study and have explained how our research addresses some of the issues in the

machine learning domain.
In the following subsection, we provide a brief introduction to the SSTD modeling techniques that have been researched and the inherent limitations they contain.

2.1 What is SSTD and why it is difficult to process it?

Video and audio information is spatio- or spectro- (sound/frequency) temporal in nature and processing of such complex Spatio-/Spectro-Temporal Data (SSTD) is a challenging task in the machine learning domain. SSTD contains both the spatial (space) and temporal (time) component and most often both these components are highly correlated.

Due to the existence of strong correlations between these two components, it is essential to process them together. However, many of the existing computational methods either process spatial and temporal components separately or when processing them together the significant correlation information present in the SSTD is ignored. However, the brain is capable of performing such tasks in a fast and robust manner. Inspired by the innate cognitive functions of the brain, the proposed study investigates how various biological and cognitive aspects such as learning, evolution and neural information processing can be applied to our computational model.

2.2 SSTD Processing Approaches

Many of the traditional machine learning systems fail to consider the entirety of the SSTD pattern, therefore losing the significant information about the spatio-temporal correlation. This study takes the innovative work in machine learning done at KEDRI to new conceptual and operational levels. We have proposed a novel machine learning approach utilizing the epSNNr architecture that is expected to successfully achieve this holistic integration of spatio-temporal events. The new approach would allow us to process continuous SSTD in a faster and computationally efficient manner.

Hidden Markov Models (HMM) is one of the popular statistical approaches that is widely used for processing temporal information (Rabiner, 1989). It is often used either with traditional neural networks (Trentin & Gori, 2001) or on its own (Waibel et al., 1989; Poppe, 2010). However, it has an inherent limitation when defining the HMM for more than a single independent variable. This means that they can only be defined for a process that is a function of a single variable, such as time or one-dimensional position (Trentin & Gori, 2001; Turaga et al., 2008), rendering them incapable of optimally learning from SSTD patterns which are two-dimensional in nature. Also, there are other emerging approaches such as deep machine learning which involves the combination of Deep Belief Networks (DBNs - Generative Model) and Convolutional Neural Networks (CNNs - Discriminative Model) (Arel, Rose, & Karnowski, 2010). The proposed DBNs model nevertheless carries out learning in a frame by frame manner, rather than learning the entire SSTD patterns. On the other hand, Gerstner and Kistler (2002b) state that the brain-inspired SNN has the ability to learn spatio-temporal patterns by using trains of spikes (which are spatio-temporal events). Furthermore, the 3D topology of the spiking neural network reservoir allows us to capture the whole SSTD patterns at any given time points. The neurons in this reservoir system transmit spikes via synapses that are dynamic in nature, collectively forming a SSTD memory (Maass & Zador, 1999; Maass & Markram, 2002). Often, learning rules such as Spike-Time-Dependent-Plasticity (STDP) (Legenstein, Naeger, & Maass, 2005) are commonly utilized in SNN models.

Recently, several SNN models and their applications have been developed by numerous research groups (Verstraeten et al., 2007; Buonomano & Maass, 2009; Maass et al., 2002; Brader et al., 2007; Bohte & Kok, 2005; Natschlager & Maass, 2002) as well as by our research group at KEDRI (Kasabov, 2007; S. Wysoski et al., 2010; Soltic & Kasabov, 2010a; Kasabov, 2010a; Schliebs, Kasabov, & Defoin-Platel, 2010; Soltic & Kasabov, 2010b; S. Wysoski et al., 2008; Kasabov, 2010b). However, they process the SSTD as a sequence of static feature vectors extracted from segments of data, without utilizing the SNN's capability of learning whole SST patterns.

2.3 A brief History of Neural Networks

The study of human anatomy, especially brain studies date back thousands of years. The recent advances in science (such as electronics, cognitive and computer science) have allowed us to partially emulate the human brain and its innate cognitive ability. This section introduces the general underlying principles of neural networks as well as the strengths and weakness of different types of neural network and how they relate to each other. Some of the most significant neural network designs are presented in details below.

In late 18 century, based on the neuroanatomical findings led by the neurobiologists such as Gerlach (1858), Nissl (1858), and Waldeyerin (1863) (Swanson, 2000; Nissl, 1894), Alexander Bain presented the first neural network in his 1873 book entitled "Mind and Body. The Theories and Their Relation" (Wilkes & Wade, 1997).

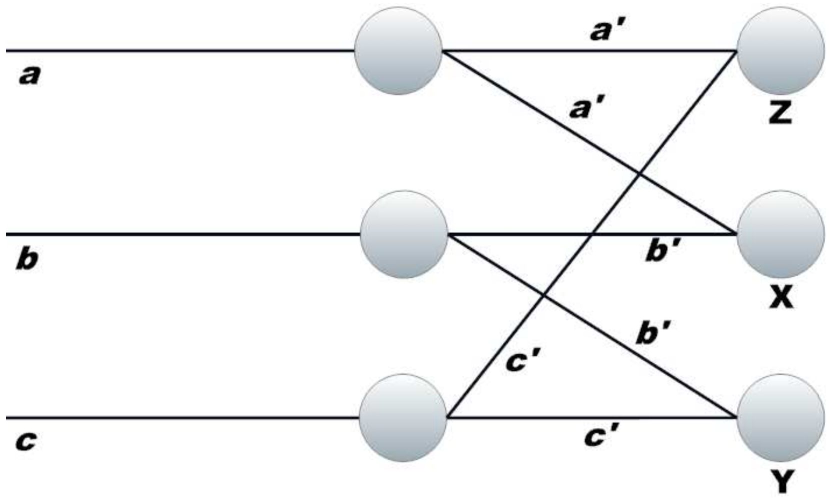


Figure 2.3.1: Bain's summation threshold network. It illustrates the way in which the connections in a neural network can channel activation in different directions. The fiber a branches into two a', a'; the fiber b into b', b'; and c branches into c', c'. One of the branches a' of a unites with one of the branches b' of b, in a cell X; b' and c' unite in Y; a', c' in Z. Adapted from Wilkes and Wade (1997).

In 1943, Warren McCulloch and Walter Pitts designed and built a primitive artificial neural network using simple electric circuits that formed the basis for modern era of neural network research (McCulloch & Pitts, 1943; Haykin, 1994).

There was significant development after the publication of 'The Organization of Behavior: A Neuropsychological Theory' by Hebb (1949). It introduced theoretical concepts such as cell assembly, phase sequence, and Hebb synapse that set forth his hebbian learning rule. The major point brought forward was the strengthening of neural pathways after each use. This rule is especially applicable for the more bio-plausible spiking neural networks. Hebb (1949) findings further reinforced McCulloch-Pitts's theory on neurons and their functions (Haykin, 1994).

With the emergence of computers in the 1950s, the neural models were ported from hardware to the digital realm. Alan Lloyd Hodgkin and Andrew Huxley described a scientific model of a spiking neuron in 1952 (Hodgkin & Huxley, 1952a, 1952b, 1952c). They explained the ionic mechanisms underlying the initiation and propagation of action potentials in the squid giant axon. Hodgkin-Huxley model is widely regarded as one of the great achievements of 20th-century biophysics that describes how action potentials in neurons are initiated and propagated (Hodgkin & Huxley, 1952a, 1952b, 1952c).

In 1954 Marvin Minsky carried out research on neural networks (Minsky, 1954) that was presented in his doctoral dissertation titled "Theory of Neural-Analog Reinforcement Systems and its Application to the Brain-Model Problem" (Harmon, 1962). In 1957, John von Neumann anticipated that computer design based on brain had great prospects and could be implemented by using telegraph relays or vacuum tubes to emulate simple neuron functions.

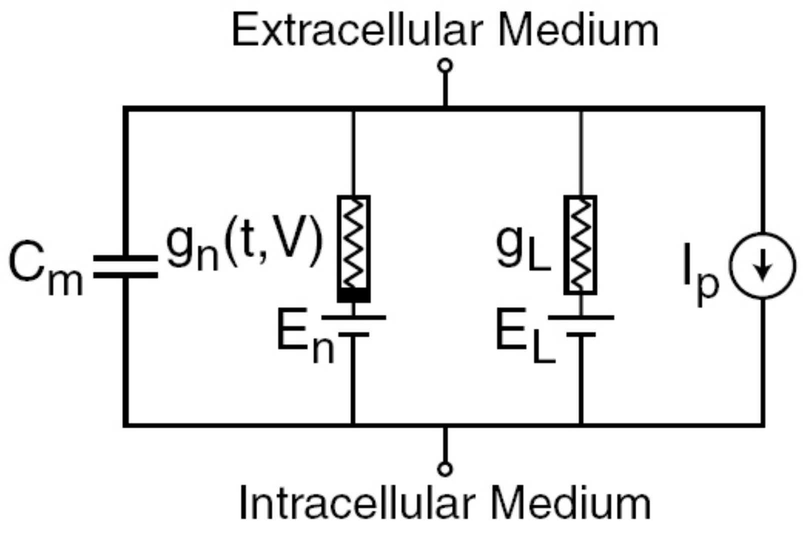


Figure 2.3.2: Hodgkin-Huxley type models represent the biophysical characteristic of cell membranes. The lipid bilayer is represented as a capacitance (C_m). Voltage-gated and leak ion channels are represented by nonlinear (g_n) and linear (g_L) conductances, respectively. The electrochemical gradients driving the flow of ions are represented by batteries (E), and ion pumps and exchangers are represented by current sources (I_p). Adapted from Hodgkin and Huxley (1952c).

This idea led to the invention of the von Neumann machine (Boahen, 2007). Later, Minsky went on to publish the first paper on artificial intelligence entitled "Steps Towards Artificial Intelligence" (Minsky, 1961).

In 1958, based on the idea of McCulloch-Pitt's theory and research done on the fly's eye, a neurobiologist named Frank Rosenblatt worked on the idea of perceptron. He built the first artificial neural network realized in hardware (Masters, 1993).

In 1960, Bernard Wildrow and Marcian Hoff developed the Adaptive Linear Neuron or later known as Adaptive Linear Element (ADALINE) and Multiple

Adaptive Linear Neuron (MADELINE) models which were the first neural networks applied to real problems. ADALINE is a convergent type single layer neural network based on the McCulloch-Pitts neuron consisting of a weight, a bias and a summation function. It's used for prediction involving binary patterns as its input and one output. Similarly, for problems requiring multiple outputs, multiple sets of ADALINEs are used in parallel and this model is known as MADELINE. These models introduced the then novel least mean square (LMS) error training algorithm, also called the Widrow-Hoff Delta Rule (Widrow & Lehr, 1990). ADALINE perceptrons learning procedure is attractor driven (positive reinforcement) in which a convergent subcircuit output value is specified as the goal for each pattern. Therefore, it is necessary for the subcircuit to learn the proper weight values to produce that goal value for any input patterns. Later perceptrons (negative reinforcement type) used misclassification errors instead of using a defined goal for each subcircuit (Widrow & Hoff, 1988; Widrow & Lehr, 1990; Widrow, 2005). In the standard McCulloch-Pitts based perceptron, the weighted sum of the inputs is passed to the activation (transfer) function and its output is used for adjusting weights. Whereas in ADALINE, based on weighted sum of the inputs, the weights drawn from subcircuits are adjusted in the learning phase.

Kohonen introduced self-organizing maps (SOMs) in the 1970s, also commonly known as Kohonen networks (Kohonen, 1989; Kohonen & Honkela, 2007). SOM is an artificial neural network used for visualization and analysis of high-dimensional data in an unsupervised setting. It uses a neighborhood function to preserve the topological properties of the high dimensional input space when projecting it into a low dimensional discretized space called Kohonen map (Kohonen, 1989).

Hopfield (1982) introduced a new form of recurrent artificial neural network as content-addressable memory systems (associative memories) with binary threshold units. This form of ANN is now also known as Hopfield network. The Hopfield networks convergence properties could be analyzed due to the introduction of the energy function. A Hopfield network can be used as an associative memory through Hebbian learning (Hopfield, 1988; Sulehria & Zhang, 2007; Haykin, 1994).

Powell (1987) invented the Radial-Basis Function (RBF) network in 1985. The idea for RBF originates from older pattern recognition techniques (Tou & Gonzalez, 1974) such as potential function, clustering, functional approximation, mixture models and spline interpolation. The RBF employs a clustering algorithm to find the most prominent clusters in the input hyperspace of multivariate data. It then linearly combines hyperspheres around these clusters to determine the classifications of specific input patterns based on previously classified training patterns (Powell, 1987). RBF networks are able to model complex mappings due to their nonlinear approximation properties, whereas perceptrons can only model by means of multiple intermediary layers (Haykin, 1994).

First introduced by Bryson and Ho (1975), the Backpropogation neural networks gained recognition through the work of David E. Rumelhart, Geoffrey

E. Hinton and Ronald J. Williams in 1986 (Russell & Norvig, 1995; Rumelhart, Hinton, & Williams, 1986). They extended the Widrow and Hoff's delta rule to networks with multiple hidden layers by means of generalized delta rule. This model was based on the perceptron and came to be known as the Back Propagation Network. Back Propagation requires differentiable activation (transfer) function. It is one of the most widely used artificial neural network model.

The underlying mechanisms of the Back Propagation paradigm, which solved the problem of training hidden neuron layers, was actually discovered earlier by Paul Werbos in 1974 (Werbos, 1974, 1994), and Parker and LeCun (Parker, 1985; LeCun, 1985, 1986) in 1985, but it was Rumelhart (Rumelhart, Hinton, & Williams, 1986) who made it universally known.

The Boltzmann machine is a type of stochastic recurrent neural network introduced in 1986 by Geoffrey Hinton and Terry Sejnowski (Hinton & Sejnowski, 1986). The Boltzmann machine is derived from Hopfield network but with sophisticated elaborations like the inclusion of annealing and stochastic processes used for tasks such as pattern completion; therefore it is used for solving pattern classification problems with noisy, incomplete data (Hinton & Sejnowski, 1983b, 1983a, 1986). Similar to Hopfield network, the Boltzmann machine has an energy function defined for the network but it is different from the Hopfield network as it is stochastic in nature (Ackley, Hinton, & Sejnowski, 1987).

In 1990 Jeff Elman presented a simple recurrent neural network, which is a feed-forward network modified by one or more feedback connections (Elman, 1991). The "Elman net" functionality produces a pattern-holding reservoir over a certain period of time, therefore allowing detection of temporal patterns in time series.

More recently spiking neural networks (SNN) (Gerstner & Kistler, 2002a; Izhikevich, 2003) have been developed. SNNs belong to the third generation of neural networks. Their mechanism is more realistic in terms of their spiking processes resembling the biological neurons (Maass & Bishop, 1999). In the following section we explain why SNNs are advantageous over traditional neural networks.

2.4 Spiking Neural Networks

The exact workings of the human brain still remain a mystery to the scientific community. It is assumed that the human brain's capability arises from the vast number of neurons and their interconnections. The number of connections for a neuron can range from 1000 to 200000 (Haykin, 1994). The latest study by Azevedo et al. (2009) shows that the average human brain (weighing 1,508.91 $\pm$ 299.14 g, age $\approx$ 50 years) has on average about 86.1 $\pm$ 8.1 billion (86.1 x 10^{11}) NeuN-positive cells ("neurons").

We do not understand the exact workings of the brain yet. In fact, understanding the functioning of a single neuron and its chemical synapses in itself proves to be much more complex than previously assumed.

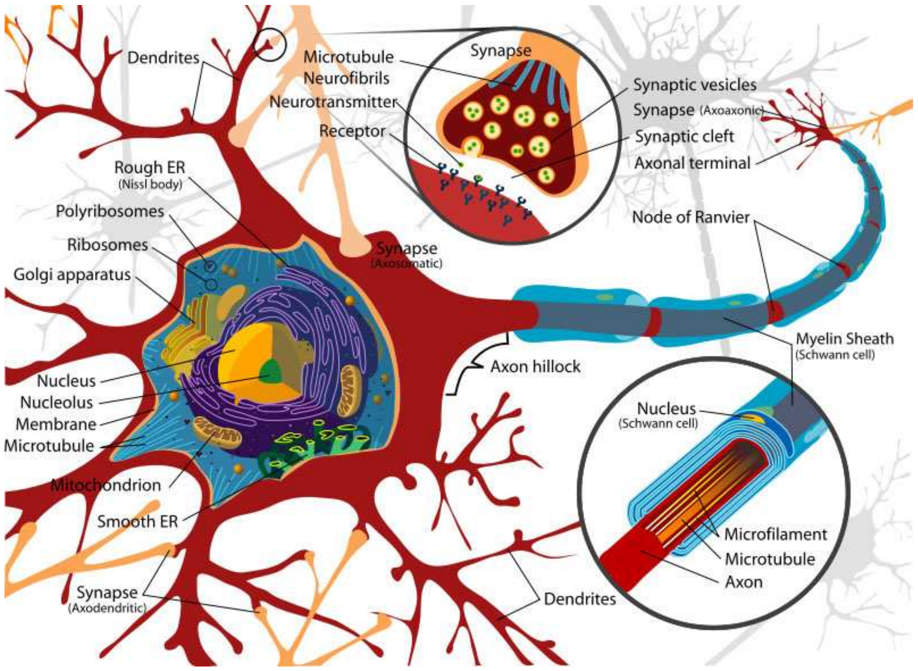


Figure 2.4.1: Schematics of Biological Neuron illustrating signal propagation and synapsis. Taken from Wikipedia (2010)

The artificial neural networks are modelled after the most basic components of the brain, which is the neurons. There is a strong similarity between artificial and biological neurons, and thus a biological counterpart for a component can always be found in an artificial neuron. When comparing the biological and artificial neurons, similarities between the two become more evident.

SNNs are made up of artificial neurons that use trains of spikes to represent and process pulse-coded information (Maass & Bishop, 1999). The biologically realistic information processing capability of spiking neural networks will allow the development novel neural models that enhances the ability to solve various problems. Gerstner and Kistler (2002a) stated that, in order to avoid any prior assumptions on neural computation, the processing and exchange of information between neurons should be carried out at the level of spikes. This resulted in the emergence of spiking neural networks as the new generation of neural network models that are imperative to the computational functionality.

In biological neural networks, neurons are connected at synapses and electrical signals (spikes) pass information from one neuron to another. SNNs are biologically plausible and offer some means for representing time, frequency, phase and other features of the information being processed. This allows incorporating spatio-temporal information in communication and computation, similarly to real neurons. There are many models of biological spiking neurons such as Hodgkin-Huxley's model (Hodgkin & Huxley, 1952c), Spike Response Models (SRM) (Gerstner, 1995; Kistler, Gerstner, & van Hemmen, 1997; Gerstner & Kistler, 2002a), Integrate-and-Fire Models (Maass & Bishop, 1999; Gerstner & Kistler, 2002a), Izhikevich models (Izhikevich, 2004, 2006, 2007; Izhikevich & Edelman, 2008). Although numerous models of SNNs and their applications have been developed, they have not been successfully used for solving large scale, complex AI problems of classification, temporal and string sequence pattern recognition and associative memory (Kasabov, 2010b). According to Kasabov (2010b), since "the spiking processes in biological neurons are stochastic by nature it would be appropriate to look for new inspirations to enhance the current SNN models with probabilistic parameters" (Schliebs, Defoin-Platel, & Kasabov, 2009; Kasabov, 2009).

2.4.1 Spiking Neural Networks Versus Traditional Neural Networks

Although SNNs can perform all the afore mentioned tasks of traditional neural networks, they have some additional capabilities in a range of domains such as;

1. Spatio-temporal domain (e.g. time series).
2. Complex domains requiring massive scale networks with several thousand neurons (e.g. VLSI).
3. Domains requiring biologically derived models (biological fidelity).

Maass and Bishop (1999) state that compared to traditional ANNs, SNN requires fewer neurons to accomplish the same task (Maass & Bishop, 1999; Gerstner & Kistler, 2002a), it is more biologically plausible (Maass & Bishop, 1999) and has the capability of approximating any function. Moreover, since SNN uses spikes instead of analog values to communicate between neurons, they can be multiplexed (as binary codes) thereby requiring less time and space to compute.

SNNs are made up of artificial neurons that use trains of spikes to represent and process pulse coded information (Maass & Bishop, 1999). SNN use trains of spikes for internal information representation (Huguenard, 2000).

2.5 Neuronal Models

Artificial spiking neural networks (SNN's) are characterized by:

- neuronal models;
- encoding information into spikes;
- learning algorithms;
- network structure and connectivity.

The counterpart or the abstraction of the biological neuron can always be found in the spiking neuron models. In order to understand why and how the SNN neuron models work, we have first provided a brief explanation of the workings of a biological neuron followed by the artificial neuron models.

2.5.1 Biological Neurons

The basic characteristics and parts of the neurons are identical to other cells; however, they are specialized cells that have the ability to gather and transmit electrochemical signals (i.e. communicate/pass messages to each other) over a particular distance. As illustrated in figure 2.4.1, a neuron has three basic parts:

Cell body: This part consists of cell components such as nucleus also sometimes referred to as the "control center" (which contains genetic material), endoplasmic reticulum and ribosomes (for building proteins from amino acids.) and mitochondria (generates adenosine triphosphate (ATP), used as source of chemical energy).

Axons: These are long nerve fibers that carry electrochemical impulses (action potentials) away from the cell body or soma. Within the brain, nonmyelinated neurons (axons not sheathed in myelin) exist.

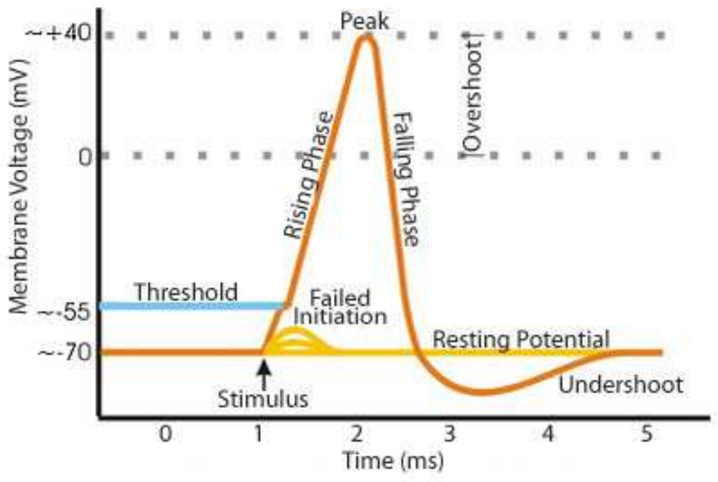
Dendrites: These nerve endings are branchlike projections of the cell that conduct the action potentials received from other neural cells to its cell body, or soma. Depending on the type of neuron, dendrites can exist on one or both ends of the cell.

2.5.2 Action potential

An action potential (nerve impulse) results from a brief change of membrane potential across an excitable membrane in a neuron that is produced due to the voltage-gated ion channels activity in the membrane (Purves et al., 2008). In order to understand how the initiation and transmission of nerve impulses take place in neurons, we need to understand the neurons structure and mechanism in details. The neurons cell membrane is made up of phospholipids. It is arranged in two-layer lipid sandwich with the electrically charged polar heads exposed to water and the polar tails sticking near each other, barricading the cell from the outside water-soluble or charged particles (such as ions). How exactly the charged particles get into the cells is explained in the next section.

Ion Channels

Ions are charged atoms (or water soluble molecules) that are able to move through channels that are present across the cell membranes (phospholipids) bilayer. Sodium, potassium, calcium and chloride ions have their own specific permeable channels in the cell membrane allowing the cell to maintain composition that is different from the outside cells. The selective permeability



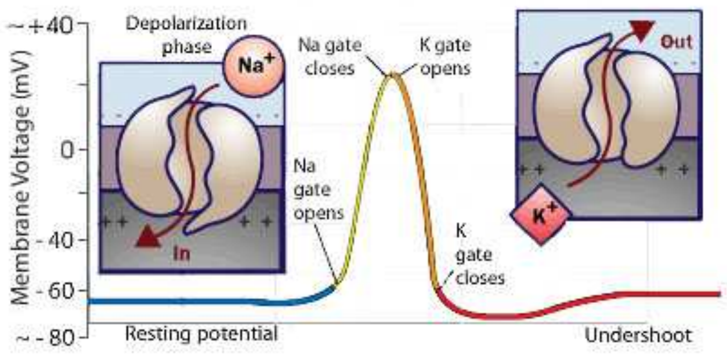


Figure 2.5.1: Action potential schematics: an ideal action potential shows its various phases as the action potential passes a point on a cell membrane. Adapted from Wikipedia (2010).

of the cell membrane allows the maintenance of outside concentration of sodium to be 10 times higher than inside and inside potassium concentration to be approximately 20 times higher than cells outside. This creates a negative electrical charge (of about $\sim$ 70 mV to $\sim$ 1 mV) inside the cell. The cell membrane is semi-permeable, i.e. *leaky*, to the tiny positively charged sodium and potassium ions in the cells outside. This difference in charge between the cells inside and outside creates an affinity. The selective ion pumps spanning across the cell membrane uses ATP as energy for exchange of the sodium and potassium ions across the cell membrane generating a tiny electrical current (as seen in Figure 2.5.1).

Nerve Signals

The action potential (i.e. nerve signal) is generated from the sodium and potassium ions coordinated movement across the cell membrane. The details on how the action potential initiates is discussed below.

Action Potential Initiation:

As mentioned previously, the inside of a cell is negatively charged and has a resting membrane potential of approximately -70 to -80 mV. This chemical or electrical imbalance between the inside and outside cell fluid opens up some of the sodium ion channels in the cell membrane. The transfer of the positively charged sodium (+Na) ions into the cell results in depolarization (reduced negative charge inside the cell). When the depolarization reaches a certain threshold value, it opens more sodium ion channels. Consequently this triggers an action potential due to more inflow of sodium ions in the nerve cell. The high concentration of +Na ions (inside the nerve cell) reverses the local membrane potential. At this point as seen in Figure 2.5.1, the electrical potential inside the cell goes to about $\sim +40$ mV. At this electrical potential, the sodium ion channel closes resulting in sodium inactivation (inflow of sodium ion stops) and simultaneously triggers the opening of the potassium ion channels. The outflow of positive potassium ions repolarises the cell returning it toward the resting membrane potential and causes the shutdown of potassium ion channels. Although, the membrane potential overshoots the resting potential, the ion balance is quickly restored by the sodium-potassium pump bringing the membrane potential to its resting level. This ion flux triggers an action potential (Hodgkin & Huxley, 1952c).

Action Potential Propagation

After the action potential is passed on to the next area of the cell membrane, the previous area enters into a refractory period (depolarizes again) preventing the backward movement of the action potential. Thus the action potential is conducted only in one direction at the speed of 10 to 100 meters per second through the axon depending on the type of neuron (Hodgkin & Huxley, 1952a, 1952b, 1952c). Since the size of action potential remains the same, it is assumed that the information is encoded by the frequency (pulse or rate code is still debatable) of action potentials (Maass & Bishop, 1999).

Synaptic Transmission

Synapsis is the gap between neurons where the communication occurs. Synaptic transmission is also known as neurotransmission that occurs due to the propagation of action potential within the synapses through neurotransmitters. The neuron that sends the action potential is known as presynaptic neuron and the receiving neuron cell is called the postsynaptic neuron. There are several types of neurotransmitters such as glutamate, serotonin, acetylcholine, norepinephrine, dopamine, gamma-amino butyric acid (GABA) etc. Also, a nerve cell may have synapses on it from excitatory presynaptic neurons and from inhibitory presynaptic neurons. In addition to neurotransmitters, there are neuropeptides that modulate the effects of groups of neurotransmitters and single neurotransmitters, over varying time scales (from milliseconds to days). Some of the neuropeptides are released from axonal terminals directly into the synaptic void, while others are released from hormone glands. This leads to a collective management of the control mechanisms that direct the action of intracellular ion flux.

As to how the synaptic transmission process takes place, neurotransmitter serotonin will be used as an example. The presynaptic cell makes serotonin (5-hydroxytryptamine, 5-HT) from the amino acid tryptophan and stores it in vesicles in its end terminals. When an action potential (called the presynaptic spike) passes down the presynaptic cell into its axonal terminals, the vesicles containing serotonin gets stimulated (by the action potential) to fuse with the cell membrane and releases the neurotransmitter into the synaptic cleft. The released serotonin traverses across the cleft and binds with receptors (ion channels) on the membrane of the postsynaptic cell and causes depolarization in the postsynaptic cell. This is caused due to the influx of positive Na ions from the extracellular fluid, which thereby charges the intracellular fluid of the dendrite. As explained previously, when depolarizations reach a threshold level, a new action potential will be propagated in that cell. However, some neurotransmitters hyperpolarizes the postsynaptic cell (more negatively charged) inhibiting the formation of action potentials. The remaining serotonin molecules in the synaptic cleft are then destroyed by enzymes such as monoamine oxidase (MAO) and catechol-o-methyl transferase (COMT). Also some of the neurotransmitters are reabsorbed by the presynaptic cell (reuptake), where MAO and COMT destroy the absorbed serotonin molecules. This prepares the synapse to receive another action potential. The soma, however, remains temporarily charged for a longer period than the dendrites, allowing it to work like an accumulation buffer (leaky reservoir) for new voltage promulgations from the dendrites (Hodgkin & Huxley, 1952c, 1952b, 1952a; Arbib, 2003; Amari & Kasabov, 1998; Purves et al., 2008).

2.5.3 Artificial Spiking Neuron Models:

Many of the traditional ANN's neural models consisted of synaptic weights and activation / transfer function. The artificial neuron abstraction could be simply expressed in mathematical form as

$$y_j = \phi(\sum_i w_{ij} x_i) \tag{2.1}$$

where y_j and x_i are the neuronal output and input signals respectively, ϕ is the activation function and w_{ij} represents the synaptic connection weight between neurons i and j. Biological neurons, however, are described by ion currents that are transmitted through the cell membrane when neurotransmitters activate the ion channels in the cell. In order to simulate a biologically realistic neuron, many models have been proposed. The next section provides a short description on some of these neuron models.

Hodgkin-Huxley Model:

Based on the experiments on the giant axon of the squid, Hodgkin and Huxley found three different ion channels: sodium, potassium and a CI^- ions leak current (Hodgkin & Huxley, 1952b, 1952a, 1952c). The flow of ions through the cell membrane is controlled by voltage-dependent ion channels as explained previously. In mathematical terms, this model can be described as an electric circuit (see Fig.2.3.2) having a capacitance (C), batteries (E), and current sources (I). Thus, as seen in Fig.2.3.2, the current applied over time ($I(t)$), may be distributed as a capacitive current (I_C) which charges the capacitor (C), and the current (I_k) of each ion channel is:

$$I(t) = I_C(t) + \sum_k I_k(t) \tag{2.2}$$

where the $\sum_k$ represents the sum of all ion channels. The capacitor (C) can be defined as $C = QIu$, where Q and u are the charge and voltage across the capacitor. Therefore, charging capacitive current can be represented as

$$I_C = C\frac{du}{dt} \tag{2.3}$$

Hence, according to Eq.2.2 and Eq.2.3:

$$C\frac{du}{dt} = -\sum_k I_k(t) + I(t) \tag{2.4}$$

Therefore, Eq.2.4 can be used to represent Hodgkin-Huxley's three ion channel model as:

$$\sum_k I_k(t) = g_{Na}m^3h(u - E_{Na}) + g_K n^4 h(u - E_K) + g_L(u - E_L) \tag{2.5}$$

where E_{Na}, E_K and E_L are reversal potentials obtained from empirical experiments. The gating variables m, n and h evolve according to the differential equations

$$x = \alpha_x(u)(1 - x) - \beta_x(u)x \tag{2.6}$$

where x represents m, n or h and α_x, β_x denote exponential function that can be adjusted in order to simulate a specific neuron. It can be seen that the Hodgkin-Huxley model can reproduce electrophysiological measurements very accurately. However, due to the model's complexity, it is computationally expensive, making it inappropriate for large networks of spiking neurons.

Leaky Integrate and Fire Model (LIF)

A LIF neuron is a simplified Hodgkin-Huxley model where all the ion channels are represented with a single current (Stein, 1967). Therefore, according to Eq.2.3 and $I_R = u/R$ (Ohm's law) we get

$$I(t) = \frac{u(t)}{R} + C\frac{u}{dt} \tag{2.7}$$

On introducing a time constant $\tau_m = RC$ and R in Eq.2.7, we yield the standard form

$$\tau_m \frac{du}{dt} = -u(t) + RI(t) \tag{2.8}$$

where, u is the membrane potential, $I(t)$ represents input current, τ_m is membrane time constant and R represents (soma membrane) resistance. Apart from the stimulation by the external current $I(t) = I_{ext}(t)$ over time, in a network the neurons can also be stimulated by presynaptic neuron j. The synaptic input of neuron i is the weighted sum over all the current generated by the presynaptic neurons and can be represented as :

$$I_i(t) = \sum_j w_{ij} \sum_f \alpha(t - t_j^{(f)}) \tag{2.9}$$

where, weight w_{ij} reflects the strength of the synapsis from neuron j to neuron i, $t_j^{(f)}$ represents the firing time of neuron j, while α represents the time course of the postsynaptic current. The simplicity of the LIF model makes it suitable for use in large scale networks allowing efficient simulation.

Spike Response Model (SRM)

In Spike Response Model (SRM) (Lapicque, 1907; Hill, 1936; Brillinger, 1992; Gerstner, 1995; Keat, Reinagel, Reid, & Meister, 2001; Jolivet, Lewis, & Gerstner, 2004), the state of the neuron i is defined by a single parameter $u_i(t)$ (membrane potential). The LIF model can be considered as a special case of the general SRM that defines the spike dynamics. The u_i continues to be at resting value $u_{rest} = 0$ until it receives a spike. On reaching the membrane threshold ϑ, the neuron fires and the membrane potential resets to its resting potential. Assuming that the neuron i has fired its last spike at time $\hat{t}_i$, then the evolution of u_i after firing can be expressed as

$$u_i(t) = \eta(t-\hat{t}_i)+\sum_j w_{ij}\sum_f \varepsilon_{ij}(t-\hat{t}_i, t-\hat{t}_j^{(f)})+\int_0^\infty \kappa(t-\hat{t}_i, s)I_{ext}(t-s)ds \tag{2.10}$$

where the spike time of presynaptic neuron j is denoted by $t_j^{(f)}$, η is a function that describes the form of action potential and after potential, ε denotes the time course of the postsynaptic potential and w_{ij} represents synaptic efficacy. The linear response of the membrane for external input current I_{ext} is represented by kernal function κ. Compared to the LIF model, the membrane threshold of SRM is not fixed but may depend on $t - \hat{t}_i$, hence

$$\vartheta \longrightarrow \vartheta(t - \hat{t}_i) \tag{2.11}$$

The SRM is ideal for simulating large number of neurons in a network due to its simplicity. Also, compared to LIF neuron, SRM allows to cover refractoriness. The refractoriness is important because it allows forward propagation of emitted spike. This refractory period is due to the delay in closing of voltage-gated potassium channels that opened in response to depolarization. This delay causes extra potassium conductance and results in hyper polarization.

Fast Integrate and Fire Model

The system is based on SpikeNet introduced in (Delorme, Perrinet, & Thorpe, 2001; Delorme & Thorpe, 2003; S. G. Wysoski & Benuskova, 2006; S. Wysoski et al., 2008). The postsynaptic potential dynamics for neuron i can be expressed as

$$u_i(t) = \sum_j mod^{order(j)} w_{ij} \tag{2.12}$$

where $order(j)$ determines the firing rank of neuron j, w_{ij} denotes synaptic efficacy and $mod \in [0, 1]$. The membrane potential u_i fires on reaching threshold ϑ and resets to resting potential after which the neuron is disabled. This neuron model is often used in image and speech recognition tasks due to its low computational costs.

Kasabov's probabilistic neuronal model

Models of probabilistic neurons have been proposed in several studies, e.g. in the form of dynamic synapses (Hamed, Kasabov, & Shamsuddin, 2010), the stochastic integration of the post-synaptic potential (Verstraeten et al., 2007) and stochastic firing thresholds (Norton & Ventura, 2009). In (Kasabov, 2010b) a probabilistic neuronal model is introduced that has three probabilistic parameters that extend the LIF model:

- $p_{cj,i}(t)$ is the probability that a spike emitted by neuron n_j will reach neuron n_i at a time moment t through the connection between n_j and n_i ;

- $p_{sj,i}(t)$ is the probability of the synapse $s_{j,i}$ to contribute to the post synaptic potential $PSPi(t)$ after the latter has received a spike from neuron n_j;

- $p_i(t)$ is the probability parameter for the neuron n_i to emit an output spike at time t, once the total post-synaptic potential $PSPi(t)$ has reached a value above the PSP threshold (a noisy threshold).

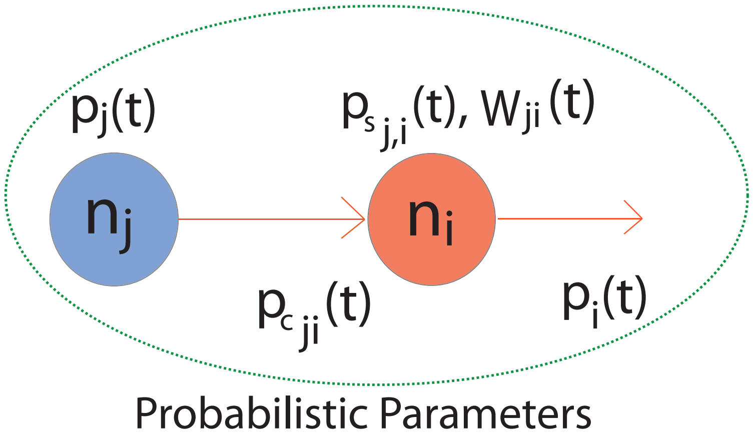


Figure 2.5.2: The figure shows the probabilistic parameters used to extend the LIF model. This probabilistic neuron model was introduced by Kasabov (2010b).

As a partial case, when all or some of the probability parameters are fixed to "1", the pSNM can be reduced to the LIF model. The LIF neuron is arguably the most popular model for simulating spiking networks. It is based on the idea of an electrical circuit containing a capacitor with capacitance C and a resistor with resistance R, where both C and R are assumed to be constant. The model dynamics are described by the following differential equation:

$$\tau_m \frac{du}{dt} = -u(t) + RI(t) \tag{2.13}$$

The constant τ_m is called the membrane time constant of the neuron. Whenever the membrane potential u crosses a threshold v from below, the neuron fires a spike and its potential is reset to a resting potential u_r. It is noteworthy that the shape of the spike itself is not explicitly described in the traditional LIF model. Only the firing times are considered to be relevant. We will introduce here only three types of probabilistic models considering only the third probability parameter $p_i(t)$ of the probabilistic model from (Kasabov, 2010b; Gerstner & Kistler, 2002a). The rest of the probability parameters are not considered in this study or are assumed to be set to 1.

2.6 Methods for encoding information into spikes

There has been a lot of argument on how exactly neurons communicate and decode the spike information into conceptual elements. The neural code can be seen as a spike train traversing with varying frequency and numbers between the neurons. Maass and Bishop (1999) and Gerstner and Kistler (2002a) state that it is assumed that the information is encoded and decoded either by the pulse or rate code or both. This means that various parameters such as spike densities, spike variability, mean firing rate and correlation with nearby spikes are derived based on a given spike train.

A single spike may carry a lot of information but is usually indecipherable if taken out of context since the brain operation follows a parallel processing design therefore making any single spike a part of a parallel collection of spike trains that can have both spatial and temporal relevance. The spatial and temporal information can be deciphered by several different decoding schemes as explained in the next section.

2.6.1 Coding/Decoding schemes

Rank Order Coding

This type of coding counts the relative timing of spikes (latency) from a reference signal to the time when the neuron spikes (Thorpe & Gautrais, 1998). Rank order coding does not consider the precise timing information and depends only on the order in which the spikes arrive. According to Thorpe and Gautrais (1998), due to the sparse coding that only considers the absence of spikes, a rank order code can transmit up to $log2(N!)$ bits of information under conditions where each input neuron can only emit one spike, but is not better than temporal coding. Maass and Bishop (1999) state that rate codes can follow three different averaging procedures such as rate as a spike count (average over time), rate as spike density (average over several runs) and rate as population activity (average over several neurons/population code).

Phase Code

Phase-of-firing code combines the spike count code with a time reference based on oscillations. This means that neuronal spike trains can encode information in the phase of a pulse with respect to the background oscillation (Gerstner &

Kistler, 2002a). According to Maass and Bishop (1999), in the hippocampus and other areas of the brain, oscillations of global variable are common and the neuronal spike trains can encode the information in phase of a pulse with respect to the oscillation occurring in the background.

Temporal Coding

In this the information is encoded by the neuron through precise timing of spikes, or action potential, on a millisecond time scale. Some examples of temporal coding include Spike-timing-dependent plasticity (STDP) and Spike-Driven Synaptic Plasticity (SDSP) learning mechanism that make use of precise spike timing.

Correlated-Firing Code

The code uses spikes from other neurons as a reference for pulse code. The spikes are supposed to occur within a specific time window after the commencement of a reference signal or an oscillation (Panchev & Wermter, 2004; Maass & Bishop, 1999). The topography of self-organizing maps, or liquid state machines, can be represented as spike trains using Correlated-Firing Code.

Synchrony Code

This code uses the synchrony between two or more neurons which could signify a unique event or convey information that is not present in the firing rate of the neuron. However, the timing of spike does not have to be exact, but should fall within the temporal binding window (Panchev & Wermter, 2004). Neurons that represent the same stimulus (pattern or event) may fire synchronously (Maass & Bishop, 1999; Izhikevich, 2006).

Rate Code:

This code counts the average number of spikes within a specific time window (Mean Firing Rate). Gerstner and Kistler (2002a) states that there are mainly three different averaging procedures. So rate code can either mean an average of spikes over time, or an average of spikes over several repetitions of the experiment (by means of summed spike density reported in a Peri-Stimulus-Time Histogram) , or an average of spikes over a population of neurons.

2.6.2 Neuromorphic Cognitive Systems

The neuromorphic cognitive systems consists of analog/digital Very-Large-Scale Integration (VLSI) architecture that utilizes silicon based hardware to mimic the neuro-biological features of nervous system, producing asynchronous event-based spike signals. Neuromorphic engineering is an interdisciplinary domain that takes inspiration from biology, computer science, mathematics, engineering and cognitive psychology to build artificial neural systems, such as vision and auditory systems. In our research we are focused on audio-visual systems and therefore below we provide a brief introduction to the artificial

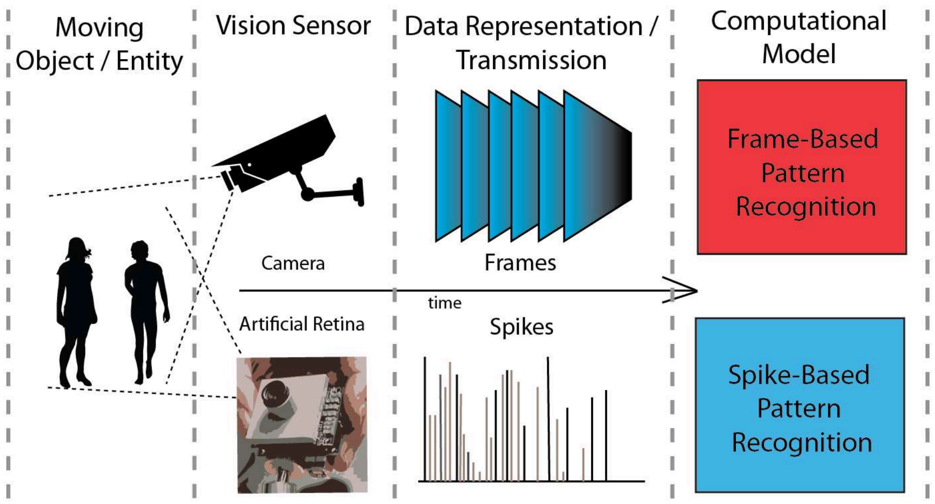

Moving Object / Entity
Vision Sensor
Data Representation / Transmission
Computational Model
Frame-Based Pattern Recognition
Camera
Frames
time
Artificial Retina
Spikes
Spike-Based Pattern Recognition

Figure 2.6.1: This conceptual diagram illustrates the difference between the Frame-based (top) and Spike-based (bottom) vision sensor, data representation and processing system. Depending on the computational model, the Frame-based method may involve computationally intensive pre-processing steps such as feature selection and spike encoding, thereby rendering the Spike-based method faster and computationally less intensive.

silicon retina and cochlea. The data that is used in one of our case studies is acquired from these artificial retina and cochlea. The neuromorphic hardware system that is used has been designed and developed by the Institute of Neuroinformatics, Zurich. In the following section we provide a brief explanation on the workings of these neuromorphic hardware devices.

Artificial Silicon Retina

Many of the real time machine vision systems have an inherent limitation of processing information on a frame by frame basis (see Figure 2.6.1). Lichtsteiner and Delbrück (2005) state that this often results in processing of redundant information present both within and across the frames. However, this drawback is addressed by an Address Event Representation (AER) artificial silicon retina. This neuromorphic vision hardware generates events corresponding to the changes in log intensity. The artificial silicon retina mimics aspects of our biological vision system which utilizes asynchronous spike events captured by the retina (Indiveri, 2008). This allows fast and efficient processing since it discards the irrelevant redundant information by capturing only the spatio-temporal information that corresponds to the temporal changes in log intensity (see Fig.2.6.2). For pixel illumination I, the operation for the continuous form can be defined as:

$$\frac{d}{dt} log I = \frac{dI/dt}{I} \tag{2.14}$$

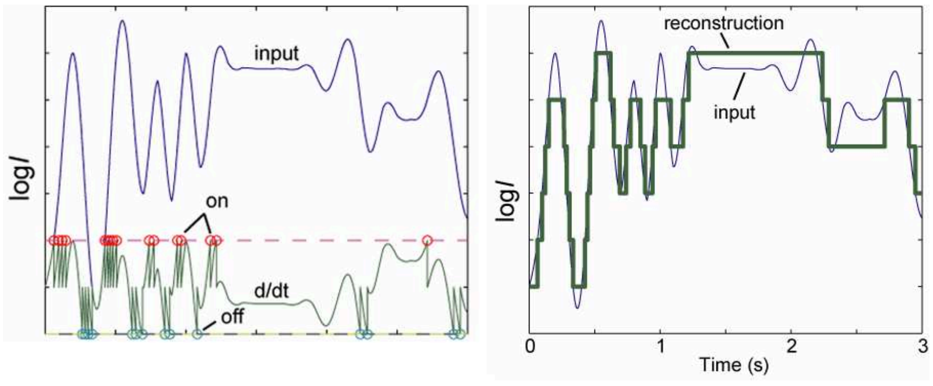


Figure 2.6.2: These two plots show the idealized pixel encoding and reconstruction of video data. The ON and OFF events represent significant changes in log I. It can be seen that changes greater than the threshold generate events, while changes that are smaller than the threshold are still represented internally in the differentiator. Adapted from Lichtsteiner and Delbrück (2005).

Artificial Silicon Cochlea

The artificial silicon cochlea consists of a pair of cochleae and a microphone with address event system similar to that of the artificial retina. It functions by extracting the interaural temporal difference from a far-field source (Chan et al., 2007), allowing tasks such as sound localization to be performed.

2.6.3 SNN Simulation Tools

Since the field of SNN is rather wide and many degrees of biological inspiration exist, there is no all-purpose programming framework that allows the simulation of all possible versions of SNN.

Nevertheless some general libraries of software exist that simulate a specialized class of spiking neurons. An excellent survey about currently available software can be found in Brette et al. (2007). So far I have considered the following options for simulating SNNs and AER spike encoding method:

- MvaSpike: This is a C++ library for event-based simulation neurons (Brette et al., 2007). The simulator is well documented with tutorials, examples, articles and implementation codes used by other researchers/scientists. Therefore, this software is a promising platform for our research.

- SpikeNet: This software library simulates simplified IF neurons and uses a graphical user interface along with an advanced simulation engine (Delorme & Thorpe, 2003). This library is designed for real time visual pattern recognition based applications (Thorpe, Guyonneau, Guilbaud, Allegraud, & VanRullen, 2004). The software is available for free public use.

- NEURON, GENESIS: These software packages for nerve simulation also provide an advanced feature-rich platform for analyzing complex neuronal compartment models. It is an open source and platform-independent software providing high biological fidelity for neuronal models (Hines, 1994; Bower, Beeman, & Hucka, 2002).

- BRIAN: This is another open source, python based software package that delivers an advanced and feature-rich platform for creating complex SNN models. Considering our proposed architecture requirements, this set of tools would be the best choice for a simulator (Goodman & Brette, 2009).

- OpenCV: This is an open source, C++ based software package that delivers a feature-rich platform for real time machine vision. There is a python wrapper available for this machine vision library. Considering our proposed architecture requirements, this is one of the tools that would be the best choice for real time (visual) data acquisition and pre-processing (Bradski, 2000).

Special consideration is needed for the implementation of the learning algorithms. Usually the event driven simulators are only able to simulate a network of neurons but do not allow a training or learning process.

2.7 Summary

In this chapter we justify the use of spiking neural networks for spatio-/spectro-temporal data followed by a literature review covering the theory of neural networks. After this, we discuss the differences between traditional artificial neural networks, artificial spiking neural networks (SNN) and biological neural networks. Various mechanisms of the biological neural networks have also been explained in details in order to show the biological plausibility of the artificial spiking neural network. The main components of spiking neural networks such as neuronal models, spike encoding methods, working memories, learning mechanisms and their applications have also been discussed in this chapter. In the next chapter we provide a literature review on spiking neural network reservoirs.

"Our ultimate objective is to make programs that learn from their experience as effectively as humans do. We shall say that a program has common sense if it automatically deduces for itself a sufficient wide class of immediate consequences of anything it is told and what it already knows."

McCarthy (1963)

3 Spiking Neural Network Reservoirs: A review

In our study, one of the Reservoir Computing (RC) approaches called Liquid State Machine (LSM) is the core component in all our proposed architectures and also in some of the methods. Therefore, it is necessary to have a dedicated chapter that explains the various instantiations of RC. Furthermore, we provide a brief review on previous research relating to LSM and their applications.

Following this, we provide justification and some possible enhancements that could improve the separability of the LSM liquid.

RC refers to the implementation, design, training and analysis of recurrent spiking neural networks and also the recurrent networks comprising of tanh or any other nonlinear units. In a nutshell, its consists of various approaches for designing the network structure, its synaptic connections and training of biologically realistic artificial spiking neural networks. It is presented briefly in the next section.

3.1 Introduction to Reservoir Computing

Reservoir computing is a neural network based computational framework for computation where the input signals are fed into a dynamical system called reservoir resulting in mapping of the input to a higher dimension. Then the readout function is used to read the states / dynamics of the reservoir for imposing an input output mapping.

Reservoir computing includes a number of independently found approaches based on this fundamental idea, namely Liquid State Machines, Echo State Networks, Backpropagation Decorrelation and Temporal Recurrent Networks. The reservoir comprises a group of recurrently connected neurons. The connectivity is generally random, and the units are typically nonlinear. On the whole, the activity in the reservoir is driven by the input and is also influenced by the past. The reservoir's dynamical input output mapping provides a crucial benefit over the simple time delay neural networks. This approach theoretically allows for real time computation on continuous input streams in parallel. Each neuron is stimulated by time varying inputs from external sources as well as from other neurons. The recurrent connectivity turns the time varying input into a spatio temporal pattern of activations in the network nodes (Maass et al., 2002).

The reservoir system is partially biologically plausible, since parts of the cerebral cortex have been found to carry out sensory integration in small and homogeneous columns of neurons (Kandel, Schwartz, & SzJessell, 1991). However, it remains unknown, how the brain is able to perform extract features tasks from these dynamic networks. Furthermore, Gerstner and Kistler (2002a) and Abbott and Nelson (2000) state that since a reservoir has fading memory and input separability, by means of a readout function, the liquid state machine can be proved to be a universal function approximator using Stone Weierstrass theorem. In the following section, we give a brief review of the approaches taken for reservoir computing and its applications.

3.2 Types of Reservoirs

Reservoir computing is a recently coined term and it subsumes a number of independently found instantiations of this fundamental idea, such as:

1. Echo State Networks (Jaeger, 2001b)
2. Liquid State Machines (Natschläger, Maass, & Markram, 2002; Maass et

al., 2002)

3. Decorrelation-Backpropagation Learning (Steil, 2004)

4. Temporal Recurrent Neural Network (Dominey, 1995)

Due to the computational methods feasibility for practical application and for being an explanatory model for some of the processes in our brain, the reservoir computing for spiking neural networks is now regarded as an established paradigm.

The core concept of reservoir computation has been independently established a number of times in various scientific domains, resulting in a number of reservoir computing varieties (Lukoševičius & Jaeger, 2009). Some of the popular flavors are briefly explained in the next section.

3.2.1 Echo State Networks

Echo State Networks (ESNs) is one of the pioneering Reservoir Computing (RC) approaches first proposed by Jaeger (2001b, 2007). For training the ESNs, a linear readout function is often sufficient for achieving good performance when employed for practical applications. This is because of the algebraic properties inherent in the recurrent neural network that is ESNs. The term dynamical reservoir refers to untrained recurrent neural network component of ESNs, and its reservoir states are termed echoes since the state reflects / represents the input history (Jaeger, 2001b). Apart from the above understanding, what makes ESNs different from other reservoir computing flavors is the use of ”weighted sum and nonlinearity” type of simulated analog-valued neurons such as tanh() nonlinearity function. Employing leaky-integrate and fire (LIF) neurons in ESNs have now become a common practice (Jaeger, Lukoševičius, Popovici, & Siewert, 2007). Since the readout from the echo state networks is linear, most often for batch training, a liner regression method is used for computing the output weights and similarly computationally inexpensive method such as least squares algorithms are employed for the online training approach (Jaeger & Haas, 2004). Most of the initial studies on ESN (Jaeger, 2001b, 2001a, 2002; Jaeger et al., 2003; Jaeger & Haas, 2004) were carried out for machine learning and nonlinear signal processing applications.

3.2.2 Liquid State Machines

Liquid State Machines (LSMs) was proposed by Maass, Natschläger, and Markram (2002) and is another pioneering reservoir method that was developed independently and at the same time as Echo State Networks. Compared to the ESNs which were framed on the basis of theoretical computational principles, the Liquid State Machine was developed on the basis of computational neuroscience. The foundation of Liquid State Machine allows the reservoir system to correspond to the computational properties of neural microcircuits (Maass et al., 2002; Maass, Natschlager, & Markram, 2003; Natschläger, Markram, & Maass, 2003; Maass, Natschläger, & Markram, 2004).

Due to this biological plausibility of LSM reservoirs, they employ biologically realistic models of dynamic synaptic connection and spiking leaky integrate-and-fire neurons. The synaptic connections among the neurons are biologically motivated and therefore LSM reservoirs often follow the topological and metric constraints seen in biological spiking neural network systems.

In Liquid State Machine reservoir, the name liquid comes from the analogy to dropping an object such as a stone into a still body of water or other liquid. The dropped object or stone causes perturbation or ripples on the surface of the liquid. Hence, the dropping of stone into the liquid can be seen as a conversion to spatio-temporal pattern of liquid displacement. Often, spike trains are used as an input to an LSM reservoir.

As a readout for LSMs, potentially anything such as linear regression, multilayer feed-forward neural networks, spiking or sigmoid neurons can be used (Maass et al., 2002). Since the output is in spikes, real-valued outputs are obtained by employing mechanisms for averaging spike trains.

LSM are often computationally expensive and difficult to implement due to the use of biologically plausible spiking neuron models with dynamic synaptic connections. The huge number of parameters associated with the neuron models and networks often makes it a challenging task to manually fine tune in order to obtain satisfying results for real world applications. Due to this reason, ESNs are more widely used for engineering application of recurrent neural networks.

On the other hand, spiking neuron models are able to perform more sophisticated information processing when compared to Echo State Networks which can emulate only the mean firing rates of biological neurons. This is because the spiking neural networks model also considers the spike firing time derived from the input signal information (Maass et al., 2002; Maass, Joshi, & Sontag, 2006). Also, due to its biological plausibility, many mechanisms from biological neural networks can be easily incorporated into the LSM reservoir.

Polychronization

Polychronization refers to the group of neurons participating in a consistent spatio-temporal firing pattern (Izhikevich, 2006). Synchronization can be considered as a special case of polychronization where a group of neurons tend to fire at the same time.

The theoretical number of possible coding schemes is quite high. Some of the statistical encoding schemes are: maximum spike rate using a specific time window, synchrony coefficient of a neuronal population, latency in percent of Maximum spike rate, standard deviation of interspike interval, mean interspike interval divided by latency etc. Maass and Bishop (1999); Gerstner and Kistler (2002a) state that how exactly the biological neuron encodes and decodes the information from and into conceptual elements is still controversial or unknown.

Previous Works

Various research groups have used Liquid State Machines (LSM's) for solving real world problems or have otherwise carried out research to understand the characteristics of Liquid State Machines under various conditions. A brief review of previous works is presented below.

Along with the spectro-temporal pattern recognition domain, Verstraeten, Schrauwen, and Stroobandt (2005) used LSM to recognize isolated word. For performance comparison, the results of LSM where compared to the state-of-the-art Hidden Markov Model (HMM) having Mel-Frequency Cepstral Coefficients (MFCC) front end. The authors also compared various spike information representation schemes such as HopfieldBrody, MFCC and Lyon Passive Ear model, considered as a preprocessing step. As an initial step, Ben's Spike Algorithm (BSA) (Schrauwen & Van Campenhout, 2003) was used to encode the sound into spike trains. Verstraeten, Schrauwen, and Stroobandt (2005) study showed the performance of LSM similar to Hidden Markov Model (HMM) having MFCC front end. Also, the authors show that LSM is robust against input noise whereas HMM is sensitive. The original design of HMM allows only processing of discretely sampled data while much of real-world information is continuous in nature (in time as well as magnitude). Furthermore, additional tasks such as speaker identification or word separation cannot be performed on the same input using HMMs.

Also, in studies by Pape, de Gruijl, and Wiering (2008) and by Ju, Xu, and VanDongen (2010), the authors have used LSM for music recognition. The latter study has also demonstrated real-time applicability of LSM for music style recognition. In both studies, the authors have used Fast Fourier Transform (FFT) to convert the sound data into vectors of 64 frequency bands with their respective amplitudes followed by normalization. Also, simple readout functions such as Recurrent Neural Network (Elman Network), K-Nearest Neighbour (KNN) or perceptrons are used for classification tasks.

Along with the spatio-temporal pattern recognition domain especially relating to visual information, Burgsteiner et al. (2005) used LSM for predicting movements from real-world images. Also, Grzyb et al. (2009) used alternative neuron models (such as integrate-and-re, resonate-and-re, FitzHugh-Nagumo, Morris-Lecal, Hindmarsh-Rose and Izhikevichs models) in LSM. Here the authors have used Gabor representations of facial expressions (in the form of image sequence) as an input to LSM. They also used HMM and multi-layered perceptrons as a LSM readout function for facial expression recognition task.

What is missing?

From the above studies, it can be seen that LSM performs very well on Spatio / Spectro-Temporal data (SSTD). However, none of the studies have explored the applicability of LSM on continuous real-word data. Especially in the case of visual data, it is often presented to the LSM in the form of a sequence of images rather than a whole spatio-temporal sequence. In our study, we hypothesize that presenting the entire visual data to the LSM is not only feasible but will also result in better classification performance since the correlation between the spatial and temporal component of the data will be strongly maintained. In our study, we want to find out what the LSM performance would be on the entire data that has varying temporal lengths ranging from milliseconds to minutes. Furthermore, most of the previous studies have used traditional machine learning algorithms. We propose an SNN based classifier as a readout function. This will show the applicability of SNN methods on real word problems, and will allow on-line one-pass computation since the SNN classifier can directly process the spikes from LSM. Furthermore, it will be entirely a spike-time based computational approach.

In a study performed in our research group (Schliebs, Nuntalid, & Kasabov, 2010), the authors have suggested that the use of stochastic neural models to construct probabilistic LSM could potentially result in better classification performance. However, the study was performed using spike inputs generated by Poisson process. In our study, we will also test this hypothesis by constructing probabilistic LSMs and applying them on real world problems.

Lastly, in majority of the LSM studies, Hebbian learning rules such as Spike Timing Dependent Plasticity (STDP) are either applied at the input to LSM connections or at the LSM to readout connections. We hypothesize that the combination of dynamic synapses in liquid and STDP (i.e. a learning reservoir) will allow the network to adapt its activity to the inputs it receives. This should in turn result in a better classification performance.

3.2.3 Backpropagation-Decorrelation

The concept of backpropagation-decorrelation reservoir has been taken from the classical recurrent neural network methods where error backpropagation is used for performance optimization.

In a study by Atiya and Parlos (2000), the analysis of recurrent neural networks weight dynamics when used with Atya-Parlos recurrent learning (APRL) algorithm (Atiya & Parlos, 2000) showed that the hidden weights change slowly compared to the output weights of the network. In another study (Schiller & Steil, 2005), it was shown that when there is a single output the weight changes are column-wise coupled. In a nutshell, when APRL is coupled with recurrent neural networks, this results in a slowly adapting reservoir with fast adapting output. Based on the above findings the BackPropagation-DeCorrelation (BPDC) was proposed by Steil (2004), producing an iterative and online recurrent neural network training method.

BPDC utilizes the same neuron model present in echo state networks. The advantage of BPDC is in its fast convergence times, therefore allowing it to quickly adapt to rapid signal changes (Steil, 2004).

3.2.4 Temporal Recurrent Networks

Peter F. Dominey's research focuses on cortico-striatal circuits in the human brain (Dominey, 1995; Dominey et al., 2003, 2006). Even though his research aims at understanding the complex neural structures and their functions rather than theoretical computational principles, he was the one who clearly defined the reservoir computing principles, stating, "... there is no learning in the recurrent connections [within a subnetwork corresponding to a reservoir], only between the State [i.e., reservoir] units and the Output units. Second, adaptation is based on a simple associative learning mechanism ..." (Dominey & Ramus, 2000). Dominey and Ramus (2000), coined the term Temporal Recurrent Network for the neural reservoir module. The algorithm implied by Dominey can be regarded as a form of Least Mean Squares algorithm. He also states that the simulated recurrent prefrontal network is dependent upon the fixed randomized recurrent connections (Dominey, 2005). The reservoir computing researchers and Dominey were not aware of each other's research findings until early 2008.

3.3 Summary

In this chapter we have given a brief description of reservoir computing and the various flavors of reservoirs. As per the research objectives discussed in chapter 1, we stated that one of our aims was to demonstrate SNNs potential to solve real world problems. Taking this into consideration, we have opted to use the Liquid State Machine as a reservoir computing approach. In chapter 4, we conduct a pilot study demonstrating the classification capability of LSM reservoir on a spatio-temporal dataset. Also, we incorporate various stochastic spiking neuron models in the LSM reservoir. The experiment setting, dataset and results are explained in more details in chapter 4.

4

A novel evolving probabilistic SNN reservoir architecture for SSTD

In this chapter, we present a novel architecture called evolving probabilistic SNN reservoir architecture (epSNNr) and carry out classification performance test on synthetic video dataset. The proposed epSNNr consists of probabilistic Liquid State Machine (LSM), which is constructed using various stochastic neuron models. The hypothesis that probabilistic LSM can potentially have better classification performance than LSM without stochastic neuron models or without LSM will be tested by using synthetic spatio-temporal data.

4.1 Aim of the study

Video information is spatio-temporal (ST) in nature and the problem of ST pattern recognition (STPR) is a challenging task in the machine learning domain. Existing statistical and artificial neural networks machine learning approaches fail to model the complex ST dynamics optimally, since they either process spatial and temporal components separately or integrate them together in a simple way, losing the significant correlation information present in the ST data. Many of the existing methods process data on a frame-by-frame basis, rather than as whole spatio-temporal patterns.

Hidden Markov Models (HMM) is among the most popular statistical approaches and is widely used for processing time series (Rabiner, 1989). HMMs are often used either with traditional neural networks (Trentin & Gori, 2001) or on their own (Poppe, 2010). However, HMM have some limitations when used for multiple time series that have also spatial components (Turaga et al., 2008).

There are other emerging approaches such as deep machine learning which involves the combination of Deep Belief Networks (DBNs - Generative Model) and Convolutional Neural Networks (CNNs - Discriminative Model) (Arel et al., 2010). The proposed DBNs model nevertheless carries out learning in a frame by frame manner, rather than learning the entire STD patterns.

The brain inspired SNN have the ability to learn spatio-temporal patterns by using trains of spikes (which are spatio-temporal events) (Gerstner & Kistler, 2002b). Furthermore, the 3D topology of a spiking neural network reservoir has the potential to capture a whole STD pattern at any given time point. The neurons in this reservoir system transmit spikes via synapses that are dynamic in nature, collectively forming an ST memory (Maass & Markram, 2002). Often, learning rules such as Spike-Time-Dependent-Plasticity (STDP) (Legenstein et al., 2005) are commonly utilized in SNN models.

Recently, several SNN models and their applications have been developed by numerous research groups (Maass et al., 2002),(Natschlager & Maass, 2002) as well as by our research group (Kasabov, 2007), (Schliebs, Kasabov, & Defoin-Platel, 2010), (Kasabov, 2010b). However, they still process ST data as a sequence of static feature vectors extracted from segments of data, without utilizing the SNN's capability of learning whole ST patterns.

In order to address the limitations of the current machine learning techniques for ST pattern recognition from continuous ST data, we have developed a novel SNN architecture called evolving probabilistic SNN reservoir (epSNNr).

The aim of this study is to demonstrate the feasibility of the proposed novel architecture for continuous ST modeling and pattern recognition utilizing epSNNr. More specifically, in this study we show that:

1. The epSNNr approach is more accurate and flexible than using standard SNN;

2. The use of probabilistic neuronal models is superior to the traditional deterministic SNN models, including a better performance on noisy data in this particular dataset.

In order to demonstrate the feasibility of the proposed novel architecture, we have evaluated our approach on a synthetic video dataset.

4.2 The proposed epSNNr Architecture

The proposed epSNNr architecture can be described by the following characteristics:

- it uses a probabilistic model of a neuron;

- it captures in its internal space ST patterns from data that can be classified in an output module;

The design of the overall epSNNr architecture is illustrated in Figure 4.2.1, where the data acquisition part represents the video and/or audio data stream along with the spike encoding module. The data processing module represents several components/modules where dimensional transformation and learning take place.

The connections between neurons are initially set by using a Gaussian function centered at each spatially located neuron, so that closer neurons are

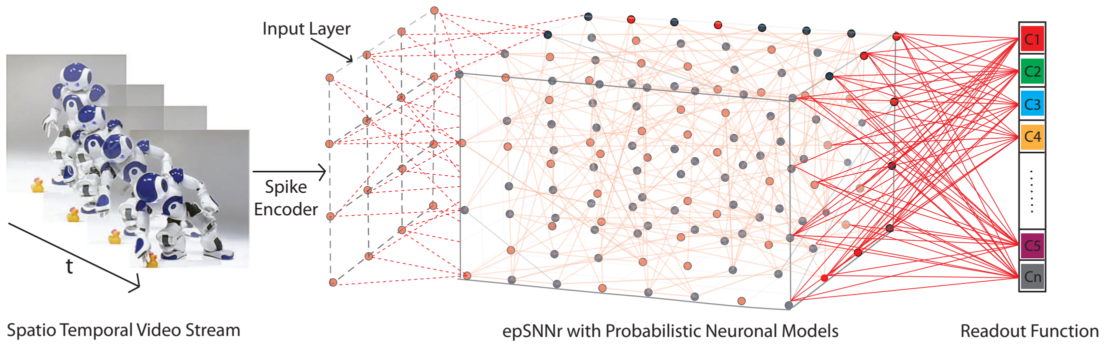

Input Layer
Spike Encoder
t
C1
C2
C3
C4
C5
Cn
Spatio Temporal Video Stream
epSNNr with Probabilistic Neuronal Models
Readout Function

Figure 4.2.1: A generic epSNNr architecture for spatio-temporal data modeling and pattern recognition.

connected with a higher probability. The input information is transformed into trains of spikes before being submitted to the epSNNr. Continuous value input variables can be transformed into spikes using different approaches:

- population rank coding (S. Wysoski et al., 2010),(Kasabov, 2007),(Schliebs, Kasabov, & Defoin-Platel, 2010);
- thresholding the input value, so that a spike is generated if the input value is above a threshold;
- thresholding the difference between two consecutive values of the same variable over time as it is in the artificial cochlea and artificial retina devices (Schliebs, Nuntalid, & Kasabov, 2010; Hamed et al., 2010).

The input information is entered in the epSNNr continuously and its state is evaluated after an entire input stream (a sample) is entered, rather than after every single time frame.

The epSNNr uses a probabilistic neural model as explained in the next section. The current state of the epSNN 'reservoir' $S(t)$ is captured in an output module. For this purpose dynamically created spatio-temporal clusters

$C_1, C_2, \dots C_k$ of close (both in space and time) neurons, can be used. The state of each cluster C_i at a time t is represented by a single number, reflecting on the spiking activity at this time moment of all neurons in the cluster, which is interpreted as the current spiking probability of the cluster. The states of all clusters define the current reservoir state $S(t)$. In the output function, the cluster states are used differently for different tasks.

4.3 Probabilistic neuronal models in the epSNNr as extensions of the LIF model

Models of probabilistic neurons have been proposed in several studies, e.g. in the form of dynamic synapses (Hamed et al., 2010), the stochastic integration of the post-synaptic potential (Verstraeten et al., 2007) and stochastic firing thresholds (Norton & Ventura, 2009). In (Kasabov, 2010b) a probabilistic neuronal model is introduced that has three probabilistic parameters to extend the LIF model:

- $p_cj, i(t)$ is the probability that a spike emitted by neuron n_j will reach neuron n_i at a time moment t trough the connection between n_j and n_i ;
- $p_sj, i(t)$ is the probability of the synapse $s_{j,i}$ to contribute to the post synaptic potential $PSPi(t)$ after the latter has received a spike from neuron n_j;
- $p_i(t)$ is the probability parameter for the neuron n_i to emit an output spike at time t, once the total post-synaptic potential $PSPi(t)$ has reached a value above the PSP threshold (a noisy threshold).

As a partial case, when all or some of the probability parameters are fixed to "1", the pSNM can be reduced to LIF. The LIF neuron is arguably the best known model for simulating spiking networks. It is based on the idea of an electrical circuit containing a capacitor with capacitance C and a resistor with resistance R, where both C and R are assumed to be constant. The model dynamics are described by the following differential equation:

$$\tau_m \frac{du}{dt} = -u(t) + RI(t) \tag{4.1}$$

The constant τ_m is called the membrane time constant of the neuron. Whenever the membrane potential u crosses a threshold v from below, the neuron fires a spike and its potential is reset to a resting potential u_r. It is noteworthy that the shape of the spike itself is not explicitly described in the traditional LIF model. Only the firing times are considered to be relevant. We will introduce here only three types of probabilistic models considering only the third probability parameter $p_i(t)$ of the probabilistic model from (Kasabov, 2010b). The rest of the probability parameters are not considered in this study or are assumed to be set to 1.

We define a *noisy reset* (NR) model that replaces the deterministic reset of the potential after spike generation with a stochastic one. Let $t^{(f)} : u(t^{(f)}) = v$ be the firing time of a LIF neuron, then

$$\lim_{t \to t^{(f)}, t > t^{(f)}} u(t) = N(u_r, \sigma_{SR}) \tag{4.2}$$

defines the reset of the post-synaptic potential. $N(u_r, \sigma_{SR})$ is a Gaussian distributed random variable with mean μ and standard deviation σ. Variable σ_{SR} represents a parameter of the model.

We define two stochastic threshold models that replace the constant firing threshold v of the LIF model with a stochastic one. In the *step-wise stochastic threshold* (SNT) model, the dynamics of the threshold update are defined as

$$\lim_{t \to t^{(f)}, t > t^{(f)}} v(t) = N(v_0, \sigma_{ST}) \tag{4.3}$$

Variable σ_{ST} represents the standard deviation of the Gaussian distribution N and is a parameter of the model. According to Eq.4.2, the threshold is the outcome of a v_0-centered Gaussian random variable which is sampled whenever the neuron fires. We note that this model does not allow spontaneous spike activity. More specifically, the neuron can only spike at time $t^{(f)}$ when also receiving a pre-synaptic input spike at $t^{(f)}$. Without such a stimulus a spike output is not possible. The *continuous stochastic threshold* (NT) model updates the threshold continuously over time. Consequently, this model allows spontaneous spike activity, i.e, a neuron may spike even at a time when pre-synaptic input spike is absent. The threshold is defined as an Ornstein-Uhlenbeck process (Maass & Zador, 1999):

$$\tau_v \frac{dv}{dt} = v_0 - v(t) + \sigma_{CT} \sqrt{2\tau_v \xi(t)} \tag{4.4}$$

where the noise term ξ corresponds to Gaussian white noise with zero mean and unit standard deviation. Variable σ_{CT} represents the standard deviation of the fluctuations of $v(t)$ and is a parameter of the model. We note that $v(t)$ has an overall drift to a mean value v_0 , i.e. $v(t)$ reverts to v_0 exponentially with rate τ_v, the magnitude being in direct proportion to the difference $v_0 - v(t)$.

In this chapter we explore the feasibility of using the above three probabilistic (stochastic) neuron models in an epSNNr for a simple moving object recognition task.

4.4 epSNNr Pilot Study: Synthetic Video Dataset

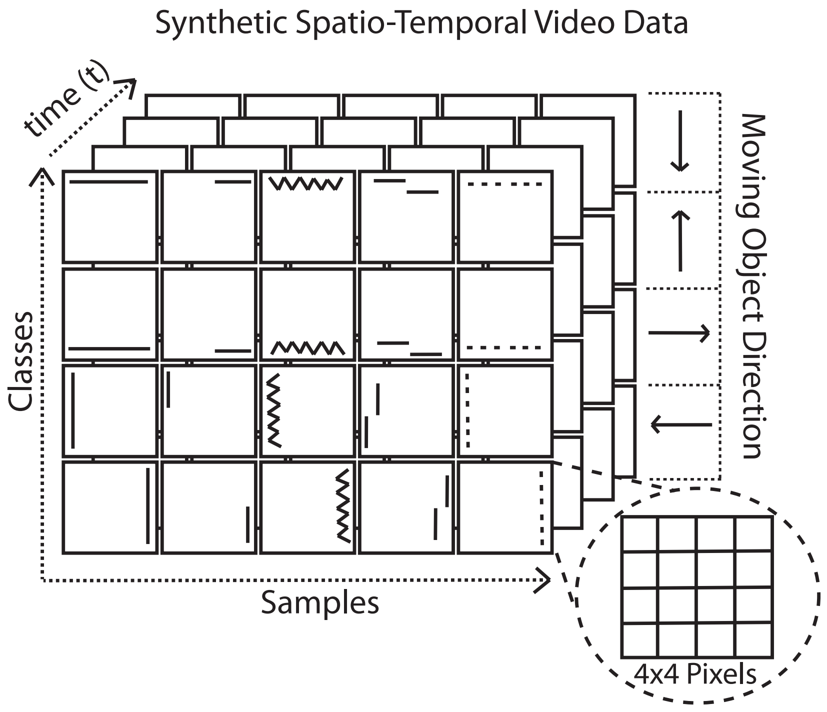


Figure 4.4.1: The figure illustrates the synthetic video dataset. There are four classes corresponding to the 4 different directions of the object movement where each class consists of five samples. The arrows represent the direction in which the objects are moving.

The synthetic video data set (see Fig.4.4.1) consists of four different classes with five samples in each class. Each class corresponds to the objects trajectory

/ movement (from up to down, left to right, down to up and right to left). Moreover, from Figure 4.4.1 it can be seen that each of the samples belonging to the same class has varying amount of noise (distorted shapes). In total there are 20 video sequences in the dataset. Each of the videos have a frame rate of 25 frames per second with time span averaging around 4 seconds. All video sequences are then resized to $4 \times 4 \times 4$.

Our goal is to apply our method for action recognition of moving objects. This particular synthetic data set was designed to test the architectures capability of classifying moving objects based on their trajectory/motion. Furthermore, this synthetic dataset is also used to test the model's feasibility for handling continuous spatio-temporal data stream where the epSNNr is provided with inputs of multiple spikes (i.e. 3-dimensional inputs).

4.5 Design of the experiment

Similar to the study presented in (S. Wysoski et al., 2010), we use the population rank encoding method for transforming the continuous value input variables into spikes. These spikes are then fed to the epSNN reservoir which results in liquid responses.

It can be seen (from Fig.4.5.1) that there are sharp peaks in the peristimulus time histograms (PSTH). This is due to occurrence of spikes after every repetition. These spikes are also known as reliable spikes and are useful for training the algorithms in order to map a particular reservoir response to a desired class label. Figure 4.5.1) shows the raster plot and PSTH produced by

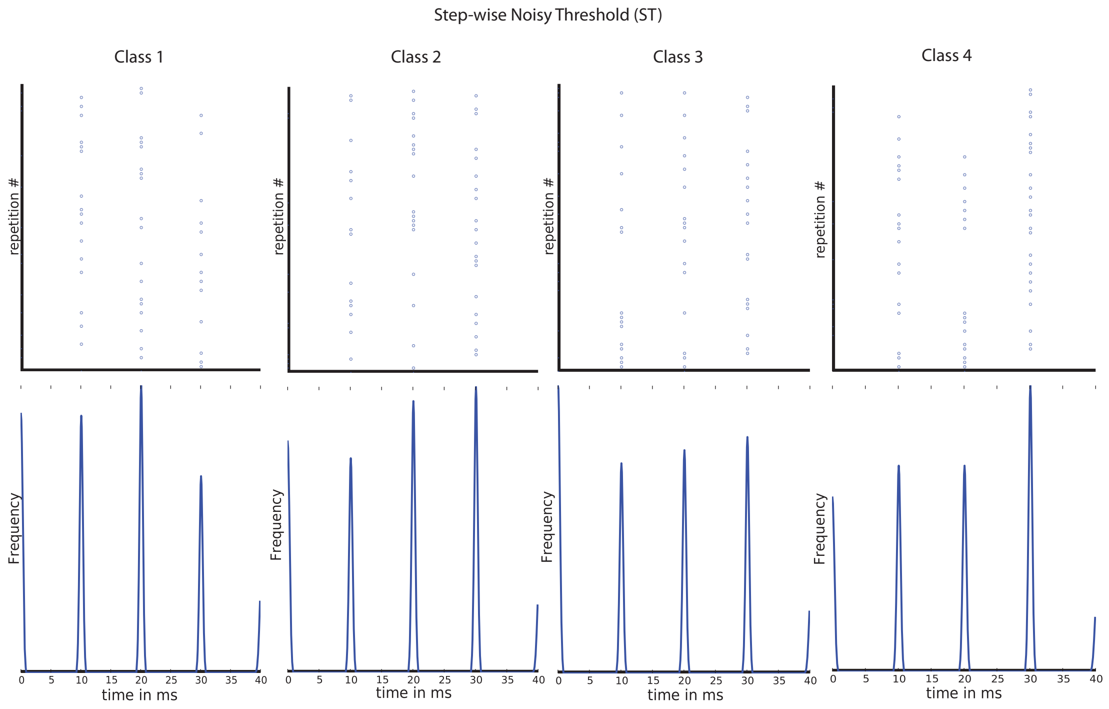


Figure 4.5.1: The figure shows the raster plots and PSTH of 4 typical states for the 4 classes produced by Step-wise Noisy Threshold (SNT). The top row shows the raster plot of the neural response of epSNNr with SNT probabilistic neurons recorded in 64 repetitions. The bottom row presents the corresponding smoothed PSTH for each raster plot. Each column corresponds to 4 different classes as indicated by the plot labels.

Step-wise Noisy Threshold probabilistic neuronal model for a particular instance belonging to one of the four different classes. On acquiring these liquid responses from the last layer of epSNN reservoir, they are concatenated as state vectors according to their corresponding classes. After transforming these liquid responses to state vectors, they are used for training and testing the classifiers.

For our pilot experiment, we have used five different types of classifiers as readout functions. These are Naive Bayes (NB), Multi-Layered Perceptron (MLP), Radial Basis Function (RBF), Decision Tree Induction Algorithm (J48) and Support Vector Machine (SVM). Default parameter settings are used for each of the classifiers in all our experiments. For MLP, the learning rate is been set to 0.3, with 64 hidden nodes for 500 epochs. RBF kernel was used for SVM with gamma value of 0.0 and weights set to 1. As for the J48, the confidence factor used for pruning is 0.25 and the minimum number of instances per leaf is set to 2.

Due to the sparsity of the data samples in each class, we have used the leave-one-out cross-validation method for the training and testing of all five classifiers. This allows us to test all samples while being unbiased and with minimum variance. The experiment is run 10 times and the obtained test results are averaged. Moreover, no pre-processing steps such as feature selection are applied on the synthetic video dataset.

One of the aims of this study is to investigate the feasibility of epSNNr for spatio-temporal video pattern recognition using different probabilistic neuron models. We have tested our synthetic video dataset with three probabilistic neuron models, namely Noisy Reset (NR), Step-wise Noisy Threshold (SNT) and Continuous Noisy Threshold (NT) along with the standard Leaky Integrate and Fire (LIF) neuron model. In order to continuously feed three dimensional inputs to the reservoir, the dimensions of the input layer are set to 4×4. This input layer dimensions are the same as those of the synthetic video data. Therefore, there is one input neuron for each pixel at a time.

4.6 epSNNr parameter settings

The LSM network topology is constructed using 140 neurons (N) arranged in a three-dimensional topology of $10 \times 7 \times 2$ neurons. The neurons (N) is made up of excitatory (N_{ex}) and inhibitory (N_{inh}) neurons with $4:1$ ratio respectively. The simulation time for the reservoir is set to 350 ms, since the maximum time scale for the synthetic video data is 3.5 seconds. These 3.5 seconds video timescale is internally represented by the LSM as 350 ms.

Table 4.6.1: The following table presents the parameter settings that are used in our experimental setup for the neuron models and LSM

Parameters	*Value/s*
For AER Spike Encoder	
Threshold	± 0.17
Video Scale	14 x 10 px
Simulation Time	350 s
For Neuron	
Time Constance	10 ms
Reset Potential	0 mV
Initial Firing Threshold	10 mV
Standard Deviation of reset fluctuation	3 mV
Standard Deviation of Step-wise Firing Threshold	2 mV
Standard Deviation of Continuous Firing Threshold	1 mV
For LSM	
Simulation Time	350 ms
Number of Neurons	140
Excitatory to Inhibitory Neuron Ratio	4:1
Input Neurons Connection Probability	0.2
Input Neurons Connection Weight	3.99 mV
Time-bin for Liquid Responses	10 ms

4.7 Experimental results and discussions

In theory, a readout function $f(L^N)$ can be of any type. Therefore, in our pilot experiment, we use five different traditional classifiers as the readout functions, which are namely Naive Bayes (NB), Multi-Layered Perceptron (MLP), Radial Basis Function (RBF), J48 Decision Tree and Support Vector Machine (SVM). In order to continuously feed 3 dimensional input to the reservoir, we assemble the input neurons such that it has the same dimensions. Therefore, there is one input neuron for each pixel/spike at a time and these neurons are connected to one of the faces of the LSM having the same dimensions as the input data. For applying the readout function (classifiers) on the liquid states $X(t)$, we have concatenated all the liquid states from different time points. Similarly, we have concatenated all the time points of the synthetic spatio-temporal video data and directly apply it on the classifier/s, i.e. without the reservoir. Default parameter settings are used for each of the classifiers in all our experiments. Moreover, no preprocessing steps such as feature selection were applied. Due to the large number of the data samples in each class, we have opted for the 10-fold cross validation method. Table 4.7.1 provides the classification accuracy where static connections and AER On Spike Events are utilized in the LSM.

From Table 4.7.1, it can be seen that epSNNr approach is more accurate and flexible than using standard SNN. Also on an average, the probabilistic neuronal models performed 7.09% better than the traditional deterministic LIF neuron model. Furthermore, when compared to the results obtained by the classifiers without the reservoir, the epSNNr approach average performance is

Table 4.7.1: The table presents the Classification Accuracy (Acc.) and Standard Deviation (Std. Dev.) for five different methods, namely Naive Bayes (NB), Multi-Layered Perceptron (MLP), Radial Basis Function (RBF), J48 Decision Tree and Support Vector Machine (SVM). The classification was carried out on LSM states generated by three probabilistic neuron models, namely Noisy Reset (NR), Step-wise Noisy Threshold (SNT) and Continuous Noisy Threshold (NT) along with the standard Leaky Integrate and Fire (LIF) neuron model.

Methods	Without Reservoir	With LSM Reservoir			
(Classifiers)	Acc.(%)/Std. Dev.	LIF Model	NR Model	NT Model	SNT Model
NB	36.45 % ± 8.3073	55.09 % ± 0.1667	65.00 % ± 9.4786	75.00 % ± 22.9640	78.39 % ± 6.6023
MLP	50.00 % ± 15.9344	56.59 % ± 0.1784	100.00 % ± 0.000	100.00 % ± 0.0000	100.00 % ± 0.0000
RBF	55.00 % ± 8.1490	93.75 % ± 10.8253	96.25 % ± 5.5902	96.25 % ± 3.4233	93.75 % ± 6.2500
J48	36.25 % ± 6.8465	52.42 % ± 0.1677	63.60 % ± 11.9486	61.25 % ± 16.7705	63.92 % ± 17.2511
SVM	46.25 % ± 12.1835	81.25 % ± 19.2638	80.10 % ± 19.3137	83.75 % ± 17.4553	77.50 % ± 18.0061

37.55% higher. We assume that this is due to the epSNNr's ability to naturally process spatio-temporal data streams when compared to traditional methods. Also, the probabilistic neuron models further enhance the separability of the reservoir. The advantage of probabilistic neural model has been well established in previous studies (Schliebs, Nuntalid, & Kasabov, 2010) and it is also apparent from our experiment. From Table 4.7.1, it can be seen that our proposed epSNNr approach performs very well, especially with classifiers such as MLP and RBF for this particular dataset.

This particular pilot study shows epSNNr's capability of handling continuous multiple spike injections using probabilistic neuron model. Moreover, the recognition rates for our system are compared to the results obtained by the classifiers without the reservoir, the average performance of the epSNNr approach is significantly higher. This suggests that the use of probabilistic neuronal models is superior in several aspects when compared with the traditional deterministic SNN models, including a better performance on noisy data. However, further study on the behavior of the epSNNr architecture under different conditions is needed and more experiments are required to be carried out on benchmark action recognition video datasets. A working reservoir implementation (in python) is provided in Appendix A, Section A.4.1.

4.8 Summary

In this chapter we introduce a novel epSNNr architecture. Also, some specific characteristics and the robustness of the method are studied. Furthermore, guidelines for the configuration of parameters are derived for the epSNNr architecture. The epSNNr presented in this chapter by itself is not evolving. However, integration of evolving readout functions in its architecture such as deSNN (introduced in chapter 6) makes it evolving in nature. The stochastic aspect (introduced by stochastic neuron models) in the LSM reservoir makes it novel from other SNN's performing RC. The findings of this study have been published in Kasabov, Dhoble, Nuntalid, and Mohemmed (2011). We also have found that it is somewhat inconvenient to map the states of the LSM reservoir into vectors for the traditional readout function. Therefore, in later chapters we introduce improved architectures that use SNN-based readout functions. This will allow the readout function to directly utilize spike output produced by epSNNr architecture. And lastly, we also found that probabilistic LSM can potentially have a better classification performance. In the next chapter we introduce the AER and its implementations.

"The eye sees only what the mind is prepared to comprehend."

Henri-Louis Bergson (1859-1941)

5 Address Event Representation (AER): a method and its implementation

In this chapter we present the analysis of the spikes output produced by a neuromorphic hardware, artificial silicon retina, under different light conditions. All of the AER based data used in our study are obtained under the same light conditions/settings, hence it it important to show the amount of inherent noise and the characteristics of the AER data. Moreover, we also present a software simulator developed for encoding spatio-temporal (visual) data into spikes using AER method and its spikes output.

The neuromorphic cognitive system consists of analog/digital Very-Large-Scale Integration (VLSI) architecture that utilizes silicon based hardware to mimic the neuro-biological features of the nervous system, producing asynchronous event-based spike signals. Neuromorphic engineering is an interdisciplinary domain that takes inspiration from biology, computer science, mathematics, engineering and cognitive psychology to build artificial neural systems, such as vision and auditory systems.

Since our research, is focused on audio-visual systems, we provide a brief introduction on the artificial silicon retina. The AER data used in our spatio-temporal pattern recognition case studies is acquired from the artificial retina. This neuromorphic hardware system has been designed and developed by the Institute of Neuroinformatics, Zurich (Delbruck, 2007). The following section provides a detailed explanation on the workings of these neuromorphic hardware devices based on which the Artificial Silicon Retina (DVS128) software simulator was developed.

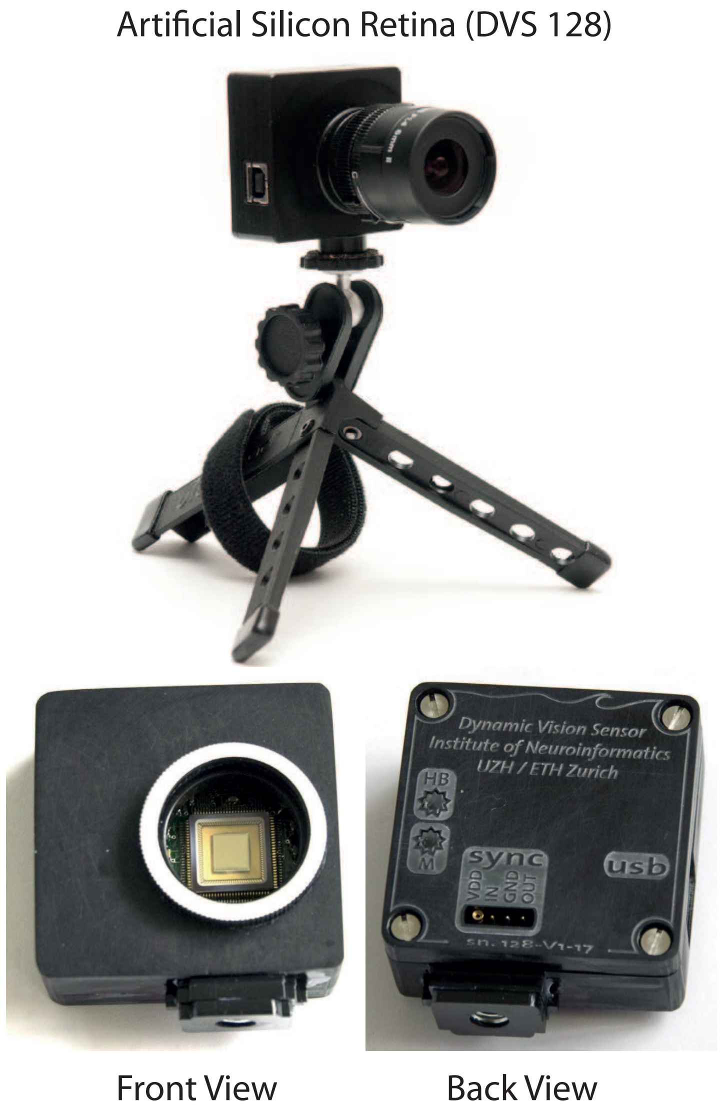


Figure 5.1.1: The figure shows the artificial silicon retina (DVS 128). This neuromorphic hardware was used to acquire data for experiments in our study.

5.1 Artificial Silicon Retina - Neuromorphic Hardware utilizing AER

5.1.1 AER Noise

For determining the level of noise under fluorescent and natural light conditions, seven samples were collected for both light conditions. 1200ms was the time

(a) AER recording environment under artificial (fluorescent) light conditions

(b) AER recording environment under natural light conditions

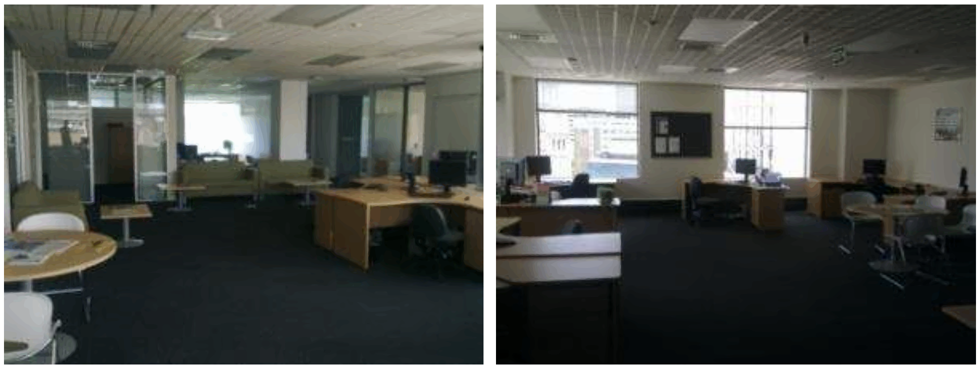


Figure 5.1.2: The figure shows the two different lighting conditions under which the AER data was recorded for the Human Action Recognition Dataset using Artificial Silicon Retina (DVS128).

period considered for all samples. 16000 neurons were used to represent the 57×57 pixel area (see Figure 5.1.4) of the Artificial Silicon Retina. The spikes (generated due to noise and collected from the Artificial Silicon Retina shown in Figure 5.1.1) in both light settings were averaged. The results (see Figure 5.1.5) show that on average, under fluorescent light conditions the level of noise is about 87.71% ± 11.41 and under natural light conditions the level of noise is around 50.00% ± 17.05. This extra noise under the fluorescent light conditions is due to flickering of the light source (tube lights). Since the Artificial Silicon Retina's capture rate is faster than standard cameras, it is able to pick up

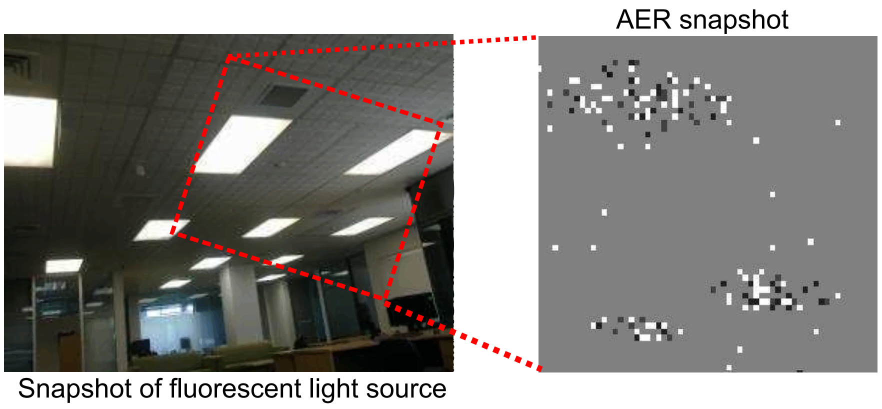


Figure 5.1.3: The figure shows the noise captured by the Artificial Silicon Retina when focused on fluorescent light source such as a tube light. The photo on the left shows the actual image and the plot on the right shows the AER snapshot.

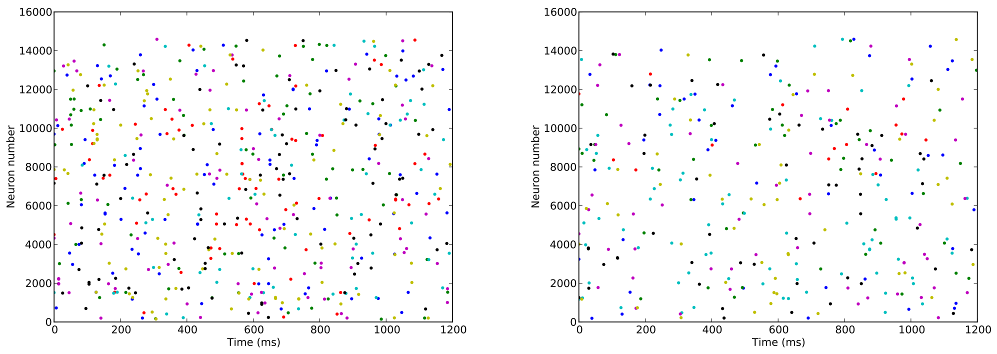


Figure 5.1.4: The figure shows the raster plot, i.e. the spikes that represent noise captured by the Artificial Silicon Retina when focused on fluorescent light source such as a tube light. The seven different spike colors represent noise from seven samples. The plot on the left is the raster plot for noise under fluorescent light conditions and figure on the right is the raster plot for noise under natural light conditions.

flicker / fluctuation of light (see Figure 5.1.3). This noise, however, can easily be reduced or eliminated by adjusting the bias settings and introducing a filter. Since when recording the data for human action recognition we did not adjust the bias setting or apply any filter, it is necessary to perform the comparison on the level of noise introduced under both light conditions. The actual light setting for both conditions / cases can be seen in figure 5.1.2.

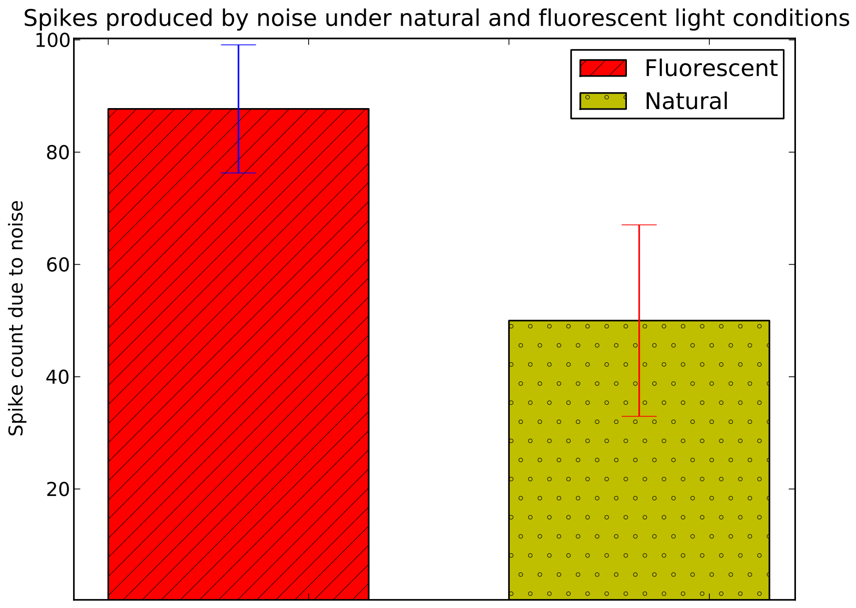


Figure 5.1.5: This bar graph compares the noise level from AER data acquired under fluorescent (left) and natural (right) light conditions.

5.2 Artificial Silicon Retina - Software Simulator utilizing AER

Based on the Artificial Retina Hardware (AER) hardware, developed in The Institute for Neuroinformatics (INI), Zurich, we have developed a software simulator that allows us to capture video and convert it to spike trains in an off-line batch manner or in real time (see Figure 5.2.1). The simulator was implemented in python using OpenCV libraries (Bradski, 2000). The KEDRI's AER software simulator algorithm is given in Appendix A, Section A.1. This serves as an alternative to the artificial silicon retina, thereby allowing us to perform pilot experiments with our developed system.

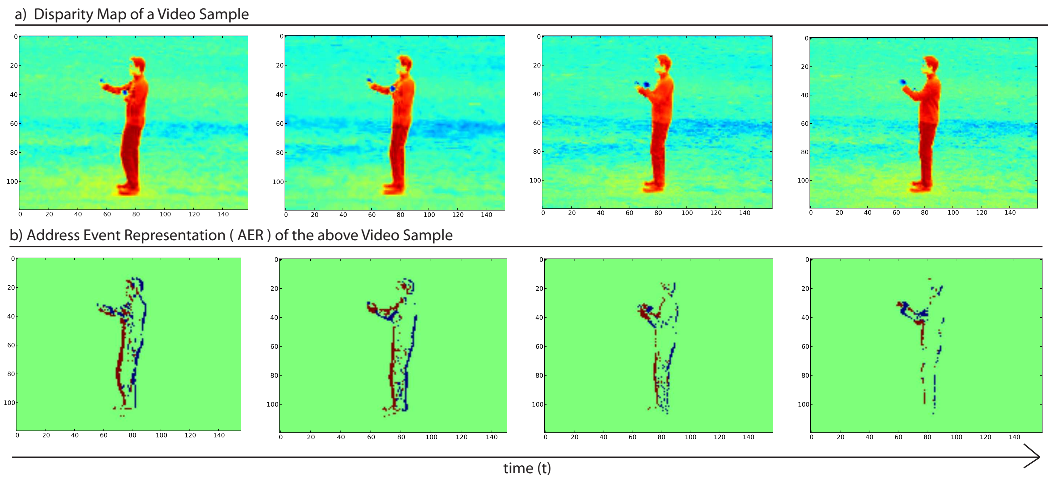


Figure 5.2.1: (a) shows the disparity map of a video sample from KTH dataset (Schuldt et al., 2004). (b) shows the Address Event Representation (AER) generated by our simulator for Video Sample shown in (a). Here the red and blue color represent the On and Off events respectively.

Many of the real time machine vision systems have an inherent limitation of processing information on a frame by frame basis. Lichtsteiner and Delbrück (2005) state that this often results in processing of redundant information present both within and across the frames (see Fig.5.2.2). However, this drawback is addressed by an Address Event Representation (AER) artificial silicon retina.

This neuromorphic vision hardware generates events corresponding to the changes in log intensity. The artificial silicon retina mimics some aspects of our biological vision system which utilizes asynchronous spike events captured by the retina (Indiveri, 2008). This allows fast and efficient processing since it discards the irrelevant redundant information by capturing only that spatio-temporal information corresponding to the temporal changes in log intensity (see Figures 5.2.2 & 5.2.1). For pixel illumination I, the operation for the continuous form can be defined as

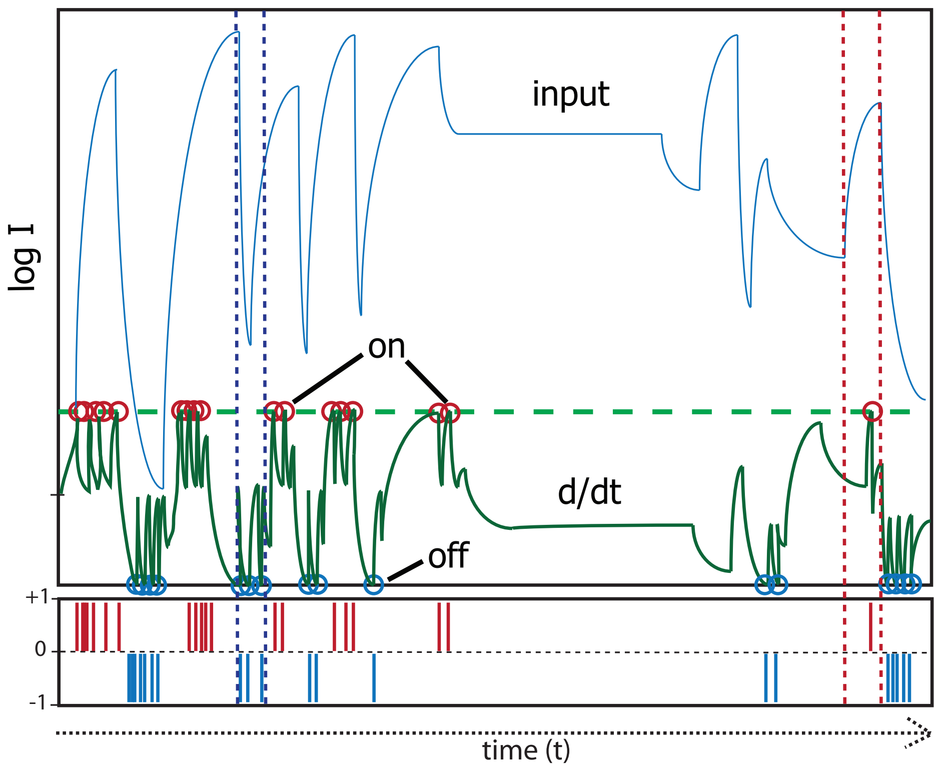


Figure 5.2.2: These figure shows the idealized pixel encoding and reconstruction of video data. The ON and OFF events represent significant changes in log I. It can be seen that the changes are greater than the threshold generated events. Adapted from (Lichtsteiner & Delbrück, 2005).

$$\frac{d}{dt}logI = \frac{dI/dt}{I} \tag{5.1}$$

We have successfully created a simulator that allows capturing of video and converting it to spike trains in real time (see Fig. 5.2.3). This serves as an alternative to the artificial silicon retina thereby allowing us to perform pilot experiments on our developed system.

Figure 5.2.3 shows the AER On and Off spike events for three human actions namely Boxing, Hand Clapping and Jogging. All three actions are performed by subject 1. It can be seen that the On and Off spike events for each of the

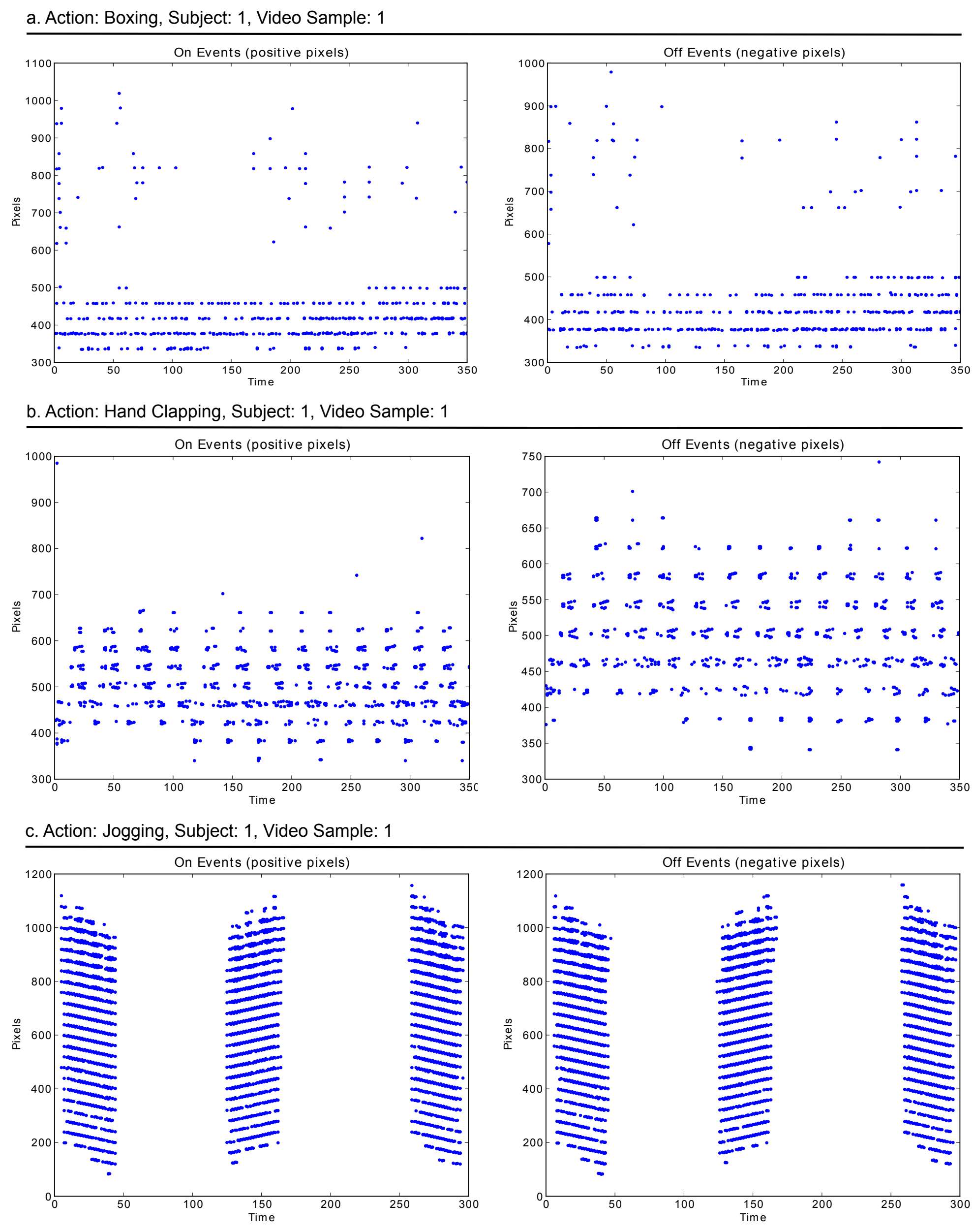


Figure 5.2.3: The figure shows the On and Off spike events for three human actions, namely a) Boxing, b) Hand Clapping and c) Jogging. It can be seen that the On and Off event map for each of the actions are quite similar especially for actions such as a) Boxing and b) Jogging. All the above three actions are performed by subject 1. At this stage we notice that the AER spike encoding for each of the actions are very much distinctive.

actions performed by the subject are very much similar. Also, each action produces spiking events that are highly distinctive from each other. However, from Figure 5.2.4, it can be seen that similar actions such as Walking, Running and Jogging produce On and Off spike events that are visually quite similar. It can be observed that different subjects performing the same action produce characteristically different spiking events. This is because the subjects do not perform the actions in an exact manner and each of them has a peculiar way of performing the same action.

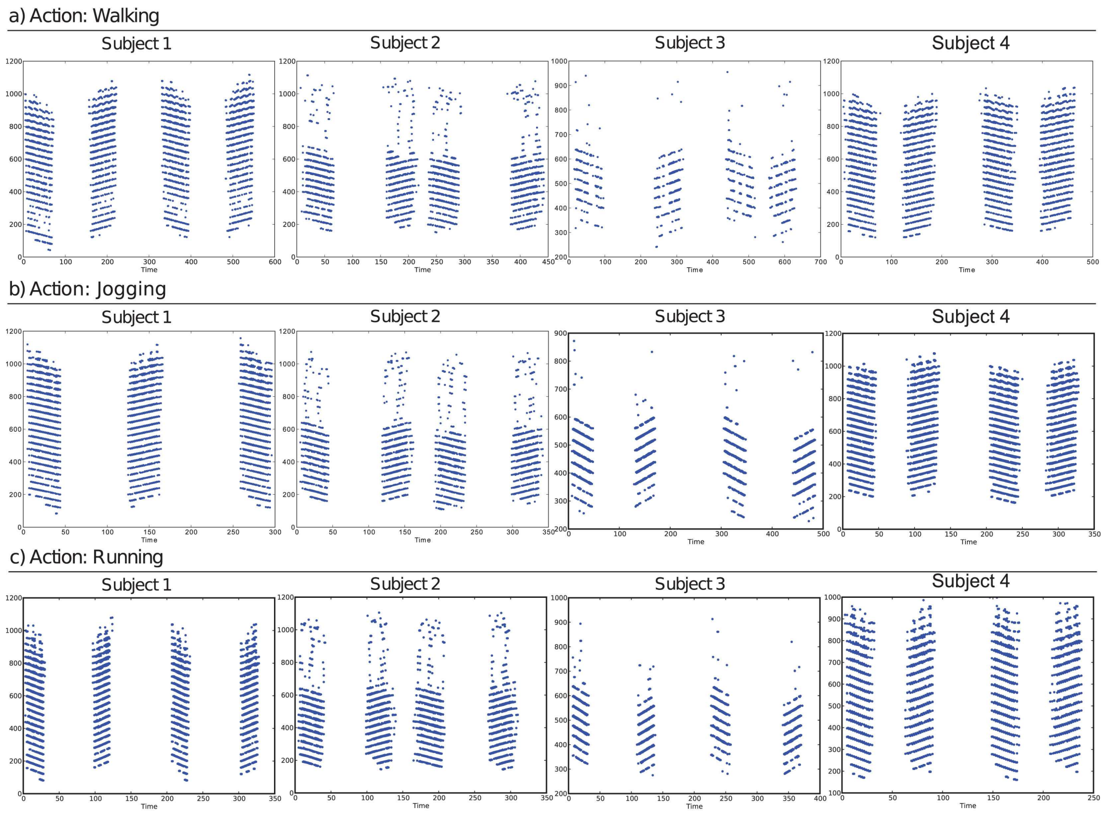


Figure 5.2.4: The figure shows that similar actions such as Walking, Running and Jogging produce spike events that are visually alike. In this figure we have only shown 'On' spike event map for 4 different subjects for easy comparison. It can be observed that different subjects performing the same action produce characteristically different spiking events.

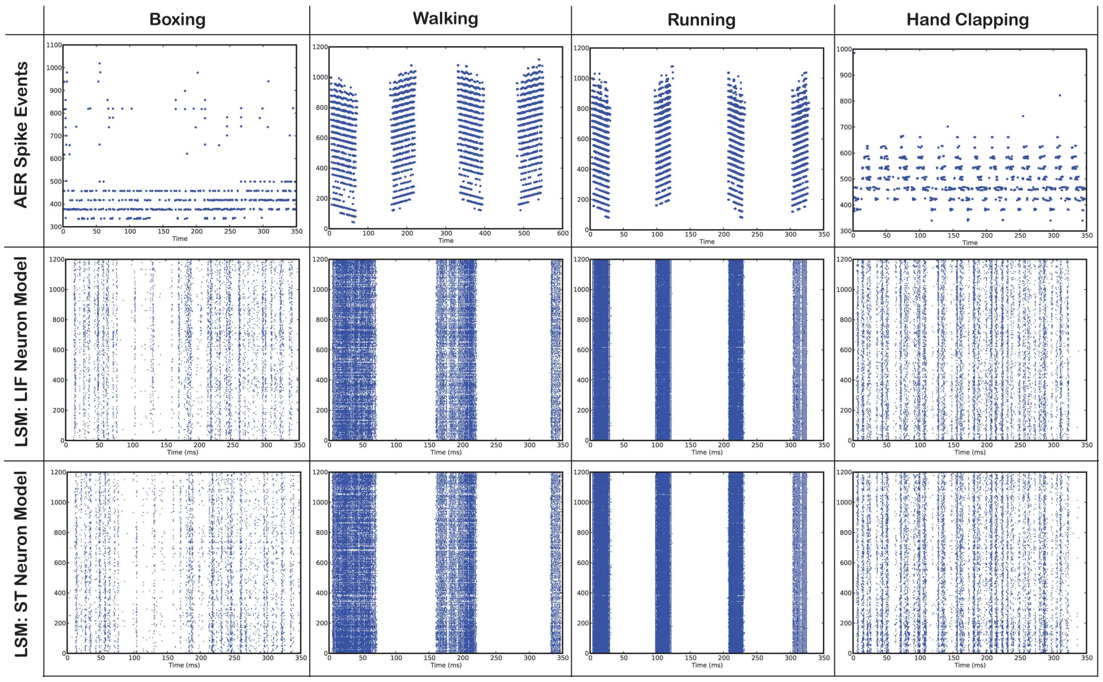


Figure 5.2.5: The above figure shows the spiking events for four actions, namely boxing, walking, running and hand clapping performed by the same subject. For each of the four activities the corresponding AER spike events, spikes from LSM using LIF neuron model and spikes from LSM using ST neuron model are presented. The x-axis represents time in milliseconds for LSM and time in seconds for AER. The y-axis represents the number of neurons

The Figure 5.2.5 shows the spiking events for four actions, namely boxing, walking, running and hand clapping performed by the first subject. For each of the four activities the corresponding AER spike events, spikes from LSM using LIF neuron model and spikes from LSM using SNT neuron model are presented. It can be observed that unique spiking patterns are produced by LSM for each of the actions performed by subject 1. Moreover, when visually comparing the spiking activities produced by the standard Leaky Integrate and Fire (LIF) and probabilistic Stepwise Noisy Threshold (SNT) neuron model, at this stage no peculiar difference between them can be identified.

5.3 Summary

In this chapter we have introduced the AER approach that is utilized by the artificial silicon retina (DVS-128). We have also shown the raster plot for output generated by the software simulator created in this study which utilizes AER spike encoding scheme. Since in later chapters we have used the data obtained from artificial silicon retina, this initial test allows us to confirm the feasibility of using the AER generated spikes in our frameworks and methods. In the next chapter, we introduce a new evolving spiking neural network based classifier / method.

6 Dynamic Evolving Spiking Neural Network (deSNN): a new generic method

So far we have proposed a novel SNN-based architecture (epSNNr) and an AER-based spike information encoding method. Now we need a SNN based classifier that can also be used as a readout function for the epSNNr. Considering the fact that novel and generic spike-based spatio/spectro-temporal data processing methods are required for modeling and pattern recognition, in this chapter, we propose one such novel method called dynamic evolving SNN (deSNN).

6.1 Evolving Spiking Neural Networks (eSNN)

In general, eSNN use the principles of evolving connectionist systems (ECOS) (Kasabov et al., 1998; Kasabov, 2002), where neurons are created (evolved) incrementally to capture clusters of input data either in an unsupervised way, e.g. DENFIS (Kasabov & Song, 2002), or in a supervised way, e.g. EFuNN (Kasabov, 2001). All developed models of ECOS type, from simple ECOS (Watts, 2009), to eSNN (Kasabov, 2007), and then to the introduced in this study dynamic eSNN (deSNN), have been guided by the following seven main principles (Kasabov, 2002):

1. They evolve in an open space;
2. They learn in on-line, incremental mode, possibly through one pass of incoming data propagation through the system;
3. They learn in a life-long learning mode;
4. They learn both as individual systems and as an evolutionary population of systems;
5. They use constructive learning and have evolving structures;
6. They learn and partition the problem space locally, thus allowing for a fast adaptation and tracing the evolving processes over time;

7. They facilitate different types of knowledge, mostly a combination of memory-based, statistical and symbolic knowledge.

6.1.1 Evolving Spiking Neural Network (eSNN)

The rank order (RO) learning rule used in the eSNN is based on the assumption that most important information of an input pattern is contained in earlier arriving spikes (Thorpe & Gautrais, 1998). It establishes a priority of inputs (synapses) based on the order of the spike arrival on these synapses for a particular pattern, which is a phenomenon observed in biological systems as well as an important information processing concept for some STPR problems, such as computer vision and control (Thorpe & Gautrais, 1998). RO-based spike coding (ROSC) and RO learning makes use of the extra information of spike (event) order.

RO learning utilizes ROSC and was introduced in (Thorpe & Gautrais, 1998). It has several advantages when used in SNN, mainly: fast learning (as it uses the extra information of the order of the incoming spikes) and asynchronous data entry (synaptic inputs are accumulated into the neuronal membrane potential in an asynchronous way). The RO learning is most appropriate for AER input data streams as the events and their addresses are entered into the SNN one by one, in the order of their occurrence (Kasabov, Dhoble, Nuntalid, & Indiveri, 2012).

The eSNN structure and a supervised learning algorithm based on the RO were introduced in (Kasabov, 2007; S. Wysoski et al., 2010). They make use of the integrate-and fire (IF) model of a neuron (Gerstner & Kistler, 2002b) (Figure 6.1.1). eSNN evolve their structure and functionality in an on-line manner, and based on the incoming data. For every new input pattern, a new neuron is dynamically allocated and connected to the input neurons (feature neurons). The neurons connections are established using the RO rule for the neuron to recognize this pattern (or a similar one) as a positive example. The neurons represent centers of clusters in the space of the synaptic weights. In some implementations similar neurons are merged (Kasabov, 2007; S. Wysoski et al., 2010). That makes it possible to achieve a very fast learning in an eSNN (only one pass may be necessary), both in a supervised and in an unsupervised mode (Kasabov, Dhoble, Nuntalid, & Indiveri, 2012).

The post-synaptic potential of a neuron i at a time t is calculated as:

$$PSP(i,t) = \sum mod^{order(j)} W_{j,i} \tag{6.1}$$

where: mod is a modulation factor; j is the index for the incoming spike at synapse j, i and $w_{j,i}$ is the corresponding synaptic weight; $order(j)$ represents the order (the rank) of the spike at the synapse j, i among all spikes arriving from all m synapses to the neuron i. The $order(j)$ has a value 0 for the first spike and increases according to the input spike order. An output spike is generated by neuron i if the $PSP(i,t)$ becomes higher than a threshold $PSPTh(i)$.

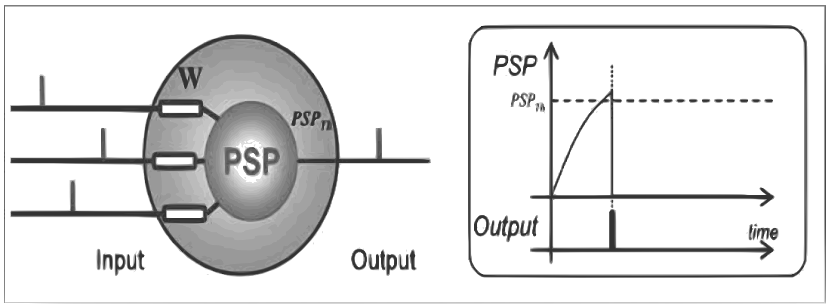


Figure 6.1.1: Integrate-and-fire neuron with RO learning

During the training process, for each training input pattern (sample, example) a new output neuron is created and the connection weights are calculated based on the order of the incoming spikes. In the eSNN, the connection weights of on-line created connections between a neuron n_i , representing an input pattern of a known class, and an activated input (feature) neuron n_j, are established using the RO rule (Thorpe & Gautrais, 1998):

$$\Delta W_{j,i} = mod^{order(j,i(t))} \tag{6.2}$$

After the whole input pattern (example) is presented, the threshold of the neuron n_i is defined to make this neuron spike when this or a similar ST pattern (example) is presented again in the recall mode. The threshold is calculated as a fraction (C) of the total PSP:

$$PSP_{max} = \sum_{j=1}^{m} \sum_{t=1}^{T} (mod^{order(j,i(t))} W_{j,i(t)}) \tag{6.3}$$

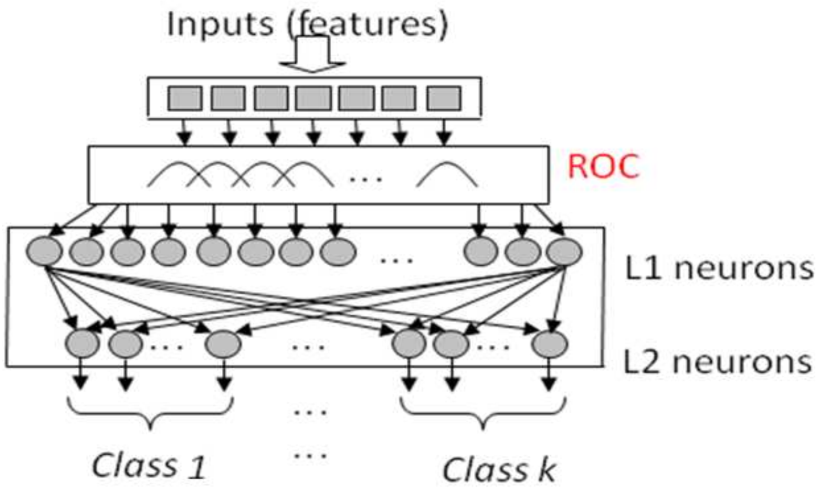


Figure 6.1.2: eSNN for classification using population coding of inputs. Taken from (S. Wysoski et al., 2010)

$$PSP_{Th} = C.PSP_{max} \tag{6.4}$$

If the connection weight vector of the trained neuron is similar to the one of an already trained neuron in a repository of output neurons for the same class, the new neuron will merge with the most similar one, averaging the connection weights and the threshold of the two neurons (Kasabov, 2007; S. Wysoski et al., 2010). Otherwise, the new neuron will be added to the class repository. The similarity between the newly created neuron and a training neuron is computed as the inverse of the Euclidean distance between weight matrices of the two neurons. An example of an eSNN for classification is given in Figure 6.1.2 (S. Wysoski et al., 2010; Kasabov, 2007). The recall procedure can be performed using different recall algorithms as explained below.

a. The first recall algorithm is when RO is used for a new input pattern (for recall, test) (Eq.6.2) and the connection weight vector for this input is compared with the patterns of existing neurons for which the output class is established during training. The closest neuron is the winner and defines the class of the new input pattern. This algorithm uses the principles of transductive reasoning (Q. Song & Kasabov, 2005) and nearest neighbor classification (Cover & Hart, 1967). It compares synaptic weight vectors of a new neuron that captures a new input pattern and existing ones. We will denote this model as eSNNs.

b. A modification of the above algorithm is when spikes of the new input pattern are propagated as they arrive to all trained neurons and the first one that spikes (its PSP is greater than its threshold) defines the class. The assumption is that the neuron that best classifies the input ST pattern will spike earlier. This eSNN is denoted as eSNNm.

The main advantage of the eSNN is that it is computationally inexpensive and boosts the importance of the order in which spikes arrive to the neuron. This makes the eSNN suitable for on-line learning of mainly static data vectors (for some applications see (Kasabov, 2007; S. Wysoski et al., 2010)). The problem with the eSNN is that there is no mechanism to deal with multiple spikes arriving at different times on the same synapse and representing same spatio-temporal pattern, which is needed for STPR. While the synapses capture long term memory during the learning phase, they have limited abilities (only through the PSP growth) to capture short term memory, which is necessary for complex STPR tasks.

In this section, we propose an extended eSNN model with the use of SDSP learning (Fusi et al., 2000), thus combining the two representations ROSC and STC. This is followed by a demonstration presented in the next chapter that shows the new model deSNN performs better than either the eSNN or the SDSP alone for an STPR problem.

6.2 The proposed Dynamic Evolving Spiking Neural Network (deSNN)

Temporal Spike Coding (TSC) and temporal spike learning are observed in the auditory and visual information processing in the brain as well as in motor control (Morrison, Diesmann, & Gerstner, 2008). Its use in neuro-prosthetics is essential along with applications for a fast, real-time recognition and control of sequence of related processes (Brader et al., 2007). Temporal coding accounts for the precise time of spikes and has been utilized in several learning rules, most popular being Spike-Time Dependent Plasticity (STDP) (S. Song et al., 2000) and SDSP (Fusi et al., 2000), the latter being implemented in an SNN hardware chip (Brader et al., 2007). Temporal coding of information in SNN makes use of the exact time of spikes (e.g. in milliseconds). This is biologically observed in the visual-, auditory-, and pre-frontal cortex and the motor control brain area. Every spike matters and so does its time. The TSC is used in several SNN models and learning algorithms, the most popular ones perhaps being STDP and SDSP as described below.

6.2.1 The Spike Time Dependent Plasticity (STDP) learning rule

The STDP learning rule uses Hebbian form of plasticity in the form of long-term potentiation (LTP) and depression (LTD) (S. Song et al., 2000). Efficacy of synapses is strengthened or weakened based on the timing of post-synaptic action potentials in relation to the pre-synaptic spike (example is given in Figure 6.2.1). If the difference in the spike time between the pre-synaptic and post-synaptic neurons is negative (pre-synaptic neuron spikes first) then the connection weight between the two neurons increases, otherwise it decreases. STDP allows connected neurons to learn consecutive temporal associations from data. Pre-synaptic activity that precedes post-synaptic firing can induce long-term potentiation (LTP); reversing this temporal order causes long-term depression (LTD).

6.2.2 The Fusi's Spike-Driven Synaptic Plasticity (SDSP) Learning Rule

The SDSP is an unsupervised learning method (Fusi et al., 2000) and is a modification of the STDP (Brader et al., 2007). SDSP directs the change of the synaptic plasticity V_{w_0} of a synapse w_0 depending on the time of spiking of the pre-synaptic neuron and the post-synaptic neuron. V_{w_0} increases or decreases depending on the relative timing of the pre- and post-synaptic spikes.

If a pre-synaptic spike arrives at the synaptic terminal before a post-synaptic spike within a critical time window, the synaptic efficacy is increased (potentiation). If the post-synaptic spike is emitted just before the pre-synaptic spike, synaptic efficacy is decreased (depression). This change in synaptic efficacy can be expressed as:

$$\Delta V_{w_0} = \frac{I_{pot}(t_{post})}{C_p} \Delta t_{spk} \;\; if \; t_{pre} < t_{post} \tag{6.5}$$

$$\Delta V_{w_0} = -\frac{I_{dep}(t_{post})}{C_d} \Delta t_{spk} \;\; if \; t_{post} < t_{pre} \tag{6.6}$$

where: Δt_{spk} is the pre- and post-synaptic spike time window. The C_p and C_d are potentiation and depression thresholds of bistable circuits.

The SDSP rule can be used to implement a supervised learning algorithm, where a teacher signal that copies the desired output spiking sequence is entered along with the training spike pattern, but without any change of the weights of the teacher input (refer to figure 6.2.2). In (Brader et al., 2007) the SDSP model has been successfully used to train and test an SNN for the recognition of 293 character (classes). Each character (a static image) is represented as a 2000 bit feature vector, and each bit is transformed into spike rates, with $50Hz$ spike burst to represent 1 and $0Hz$ to represent 0. For each class, 20 different training patterns are used and 20 neurons are allocated, one for each pattern (altogether $5,860$) (Figure 6.2.2) and trained for several hundreds of iterations.

The SDSP model is implemented in the INI analogue SNN silicon chip (Indiveri et al., 2011). The silicon synapses comprise bistability circuits for driving a synaptic weight to one of two possible analogue values (either potentiated or depressed). These circuits drive the synaptic-weight voltage with a current that is superimposed on that generated by the STDP and which can be either positive or negative. If, on short time scales, the synaptic weight is increased above a set threshold by the network activity via the STDP learning mechanism, the bistability circuits generate a constant weak positive current. In the absence of activity (and hence learning) this current will drive the weight toward its potentiated state. If the STDP decreases the synaptic weight below the threshold, the bistability circuits will generate a negative current that, in the absence of spiking activity, will actively drive the weight toward the analogue value, encoding its depressed state. The STDP and bistability circuits facilitate the implementation of both long-term and short term memory.

Figure 6.2.2 illustrates the schematic of the network architecture for a data set consisting of two classes. Brader et al. (2007) explains that the output units are grouped into two pools, selective to stimuli C1 and C2, respectively, and are connected to the input layer by plastic synapses. The output units receive additional inputs from teacher and inhibitory populations (Brader et al., 2007).

While successfully used for the recognition of static patterns, the potential of the SDSP SNN model and its hardware realization have not been fully explored for STPR.

In the following section we extend the existing evolving SNN (eSNN) model (Kasabov, 2007; S. Wysoski et al., 2010), that utilizes rank-order spike coding (ROSC) and RO learning rule described here with temporal spike coding (TSC) representation and TSC learning rules, namely the Fusis SDSP rule (Fusi et al., 2000) to arrive at a new model of a dynamic eSNN (deSNN).

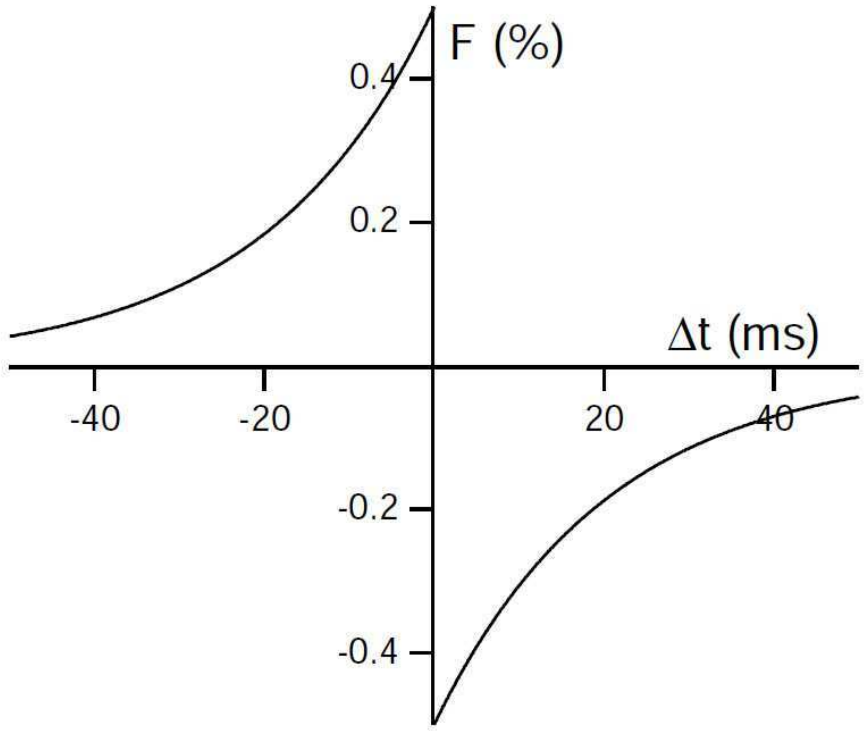


Figure 6.2.1: An illustration of the STDP learning rule. Taken from (S. Song et al., 2000)

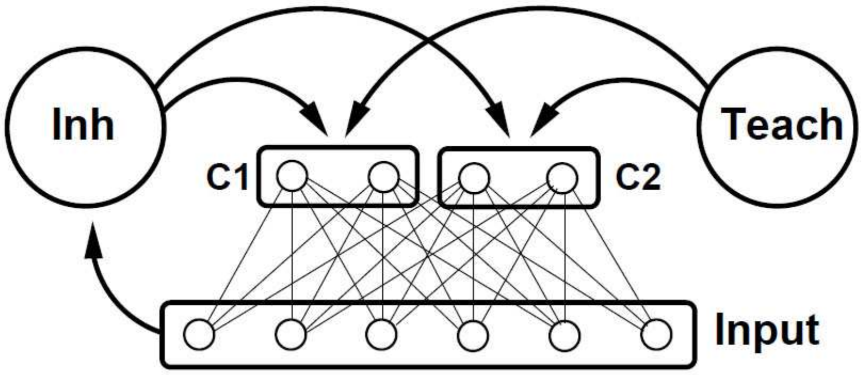


Figure 6.2.2: An example of using SDSP neurons. Taken from (Brader et al., 2007)

6.2.3 The proposed deSNN with RO- and SDSP learning rules

The main disadvantage of the RO learning eSNN is that they adjust their connection weights once only (based on the rank of the first spike), which is appropriate for static pattern recognition, but not for STPR. In the latter case the connection weights need to be further tuned based on following spikes on the same synapse using temporal spike learning.

In the proposed deSNN both the RO and SDSP learning rules are utilized. While the RO learning will set the initial values of the connection weights for a STPR utilizing the existing event order information in an AER dataset, the STDP/SDSP will adjust these connections (in an unsupervised manner) based on following spikes (events) as part of the same spatio-temporal pattern.

The training algorithm of deSNN is presented below.

```
1: SET deSNN parameters (including: Mod, C, Sim and the SDSP parameters)
2: FOR every input STP $i$ represented as AER DO
  2a. Create a new output neuron $j$ for this pattern and calculate the initial
      values of connection weights using the RO learning rule:
      $w_j = (Mod)^{order(j)}$
  2b. Adjust the connection weights $w_j$ for consecutive spikes on the
      corresponding synapses using the SDSP learning rule.
```

```
2c. Calculate $PSP_{max}$
2d. Calculate the threshold value $x_i = PSP_{max(i)} * C$
2e. IF the new neuron $j$ weight vector $w_j$ is similar to the weight vector of an
      already trained output neuron using Euclidean distance and a threshold
      $Sim$, then merge the two neurons:
      $w = w_{new} + w * N/N + 1$, $x = x_{new} + x * N/N + 1$
      where $N$ is the number of all previous merges of the merged neuron
      ELSE
      END IF
3: END FOR (Repeat to all input STP)
```

Two types of deSNN are proposed that differ in the recall algorithm and correspond to the two types of eSNN: eSNNs and eSNNm:

(a) deSNNs

(b) deSNNm

6.3 deSNN Examples

6.3.1 deSNN Example 1

Figure 6.3.1 illustrates the main idea of the deSNN learning algorithm. A single spatio-temporal pattern of four input spike trains is learned by a single output neuron. RO learning is applied to calculate the initial weights based on the order of the first spike on each synapse (shown in red). Then STDP (in this case SDSP) rule is applied to dynamically tune these connection weights. The

SDSP algorithm increases the assigned connection weight of the synapse that is receiving the next spike and at the same time depresses the synaptic connections of synapses that do not receive a spike at this time. Due to a bi-stability drift in the SDSP rule, once a weight reaches the defined High value (resulting in LTP) or Low value (resulting in LTD), this connection weight is fixed to this value for the rest of the training phase. The rate at which a weight reaches LTD or LTP depends upon the set parameter values (Kasabov, Dhoble, Nuntalid, & Indiveri, 2012).

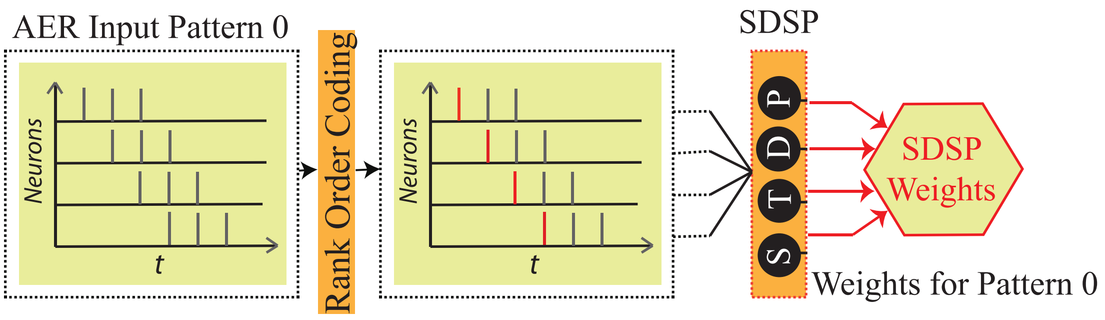


Figure 6.3.1: A simple example to illustrate the main principle of the deSNN learning algorithm.

6.3.2 deSNN Example 2

In this example we consider two spatio-temporal patterns of five inputs each (Table 6.3.1) to be learned in two output neurons and it is explained below.

The synaptic drift caused by the SDSP makes synaptic weights dynamically learn spike time relationships between different input spike trains as part of the same spatio-temporal pattern. The example above is of a very small scale and in reality there are hundreds and thousands of input synapses to a neuron and hundreds and thousands of spikes at each synapse forming a complex spatio-temporal pattern to be learned, described by some statistical characteristics. Even a small synaptic drift can make a difference (Kasabov, Dhoble, Nuntalid, & Indiveri, 2012).

Table 6.3.1: Here we consider two spatio-temporal pattern inputs. The spike time for input pattern 1 and 2 along with the ROC and SDSP values (weights) are provided for this deSNN example.

Pattern 1			**Pattern 2**		
Spike times (ms)	**ROC**	**SDSP**	**Spike times (ms)**	**ROC**	**SDSP**
Input 1: 0.0, 1.0, 2.0, 3.0, 4.0	1	0.998	Input 1: 4.0, 5.0, 6.0, 7.0, 8.0	0.4096	0
Input 2: 1.0, 2.0, 3.0, 4.0, 5.0	0.8	0.798	Input 2: 3.0, 4.0, 5.0, 6.0, 7.0	0.512	0
Input 3: 2.0, 3.0, 4.0, 5.0, 6.0	0.64	0	Input 3: 2.0, 3.0, 4.0, 5.0, 6.0	0.64	0
Input 4: 3.0, 4.0, 5.0, 6.0, 7.0	0.512	0	Input 4: 1.0, 2.0, 3.0, 4.0, 5.0	0.8	0.798
Input 5: 4.0, 5.0, 6.0, 7.0, 8.0	0.4096	0	Input 5: 0.0, 1.0, 2.0, 3.0, 4.0	1	0.998
Time of a pattern presentation T= 8.0 ms					

The connection weights learned in a deSNN represent the input patterns in an internal, computational spatio-temporal space built by the model. How many different input patterns can be learned and discriminated in this space depends on the choice of the model parameters. This issue will be discussed in Chapter 8. As a partial case, the neurons in the deSNNm can be connected to each other with inhibitory connections (the winner takes all WTA, type of connections); so once a neuron fires, it will prevent other neurons from firing (both during recall and training) as it is shown in (Brader et al., 2007).

So far, we have presented the learning phase of a deSNN model. In terms of recall, two types of deSNN are proposed that differ in the recall algorithms. They mainly correspond to the two types of eSNN from previous section, namely eSNNm and eSNNs that are discussed below.

(a) deSNNm: After learning, only the connection weights initially created

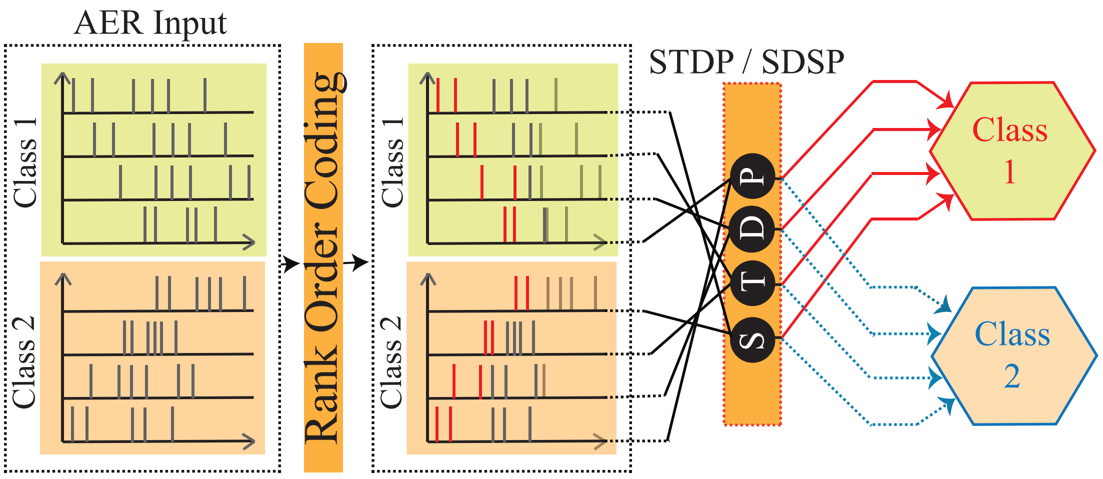


Figure 6.3.2: This figure illustrates example 2 where two spatio-temporal patterns are to be learned by two output neurons.

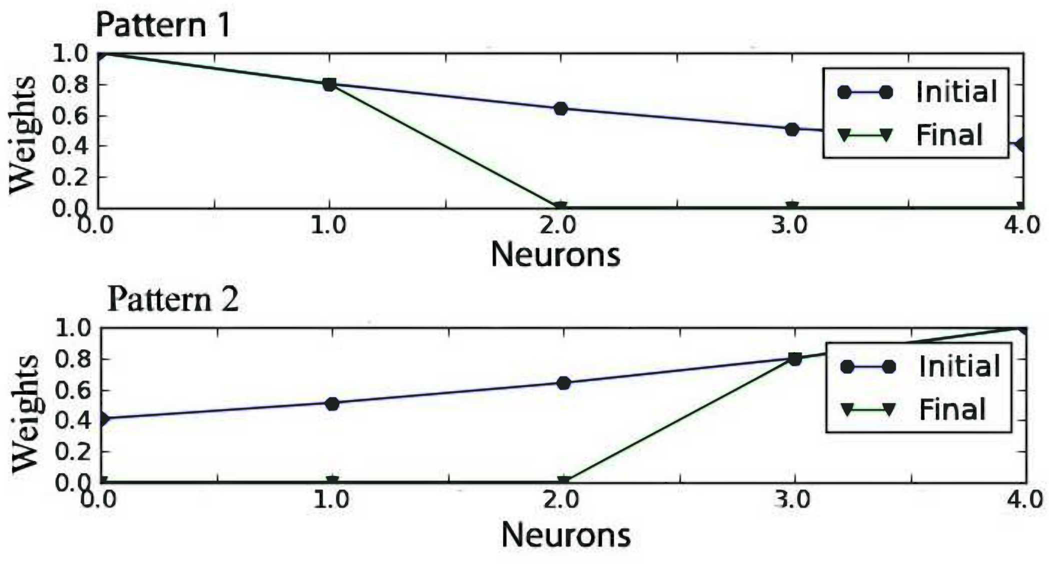


Figure 6.3.3: This figure shows the initial and final synaptic weights for patterns 1 and 2 for the first four neurons. The difference in final weights for the two spatio-temporal pattern can be clearly seen.

with the use of the RO rule are restored as long term memory in the synapses and the model. During recall on a new spatio-temporal pattern the SDSP rule is applied so that the initial synaptic weights are modified on a spike time basis according to the new pattern and using formula 6.5 as it is during the SDSP learning phase. At every time moment t the PSP of all output neurons is calculated. The new input pattern is associated with the neuron i if the $PSP_i(t)$ is above its threshold Th_i (Kasabov, Dhoble, Nuntalid, & Indiveri, 2012). The following formula is used:

$$PSP_i(t) = \sum_{l}^{t} \sum_{j}^{M} f_j(l).w_{j,i}(l) \tag{6.7}$$

where: t represents the current time unit during the presentation of the input pattern for recall; M is the number of the input synapses to neuron i; $f_j(l) = 1$ if there is a spike at time l at synapse j for this input pattern, otherwise it is 0; $w_j, i(l)$ is the efficacy of the dynamic synapse between j and i neurons at time l.

(b) deSNNs: This model corresponds to the eSNNs and is based on the comparison between the synaptic weights of a newly created neuron to represent the new spatio-temporal pattern for recall, and the connection weights of the created during training neurons. The new input pattern is associated with the closest output neuron based on the minimum distance between the weight vectors. As the synaptic weights are dynamic, the distance should be calculated in a different way than the distance measured in the eSNN, possibly using both the initial $w(0)$ and the final $w(T)$ connection weight vectors learned during training and recall. As a partial case, only the final weight vector $w(T)$ can be used (Kasabov, Dhoble, Nuntalid, & Indiveri, 2012).

6.4 Discussion

The chapter presents a new dynamic eSNN model, deSNN, that combines rank-order (RO) and spike-time learning for fast, on-line supervised or unsupervised learning, modeling and pattern recognition of SSTD; it is also suitable for efficient hardware implementation. The model is characterized by the following features:

- one - pass propagation of an SSTD during learning;
- evolving and merging neurons and connections in an incremental, adaptive, life-long learning mode;
- utilizing dynamic synapses that are modifiable during both learning and recall;
- storing history of learning in terms of initial $w(0)$ and final $w(T)$ connection weights in both learning and recall;
- the stored connection weights can be interpreted as clusters of spatio-temporal patterns that can be represented as spatio-temporal fuzzy rules, similar to the rules described in Soltic and Kasabov (2010b)

A demonstration algorithm for deSNN is provided in Appendix A, Section A.3. A major issue for the future development of deSNN models and systems is the optimization of its numerous parameters. The optimization of network parameters and the scalability of the methods is being considered for future work.

6.5 Summary

This chapter explains the characteristics and specifics of the proposed deSNN methods along with some examples that demonstrate the deSNN mechanism. The study has been published in Kasabov, Dhoble, Nuntalid, and Indiveri (2012). In the next chapter, the applicability of the two deSNN methods, namely deSNNm and deSNNs, is tested using real simple motion data captured by the AER artificial silicon retina.

7

AER based Simple Motion Recognition with the deSNN method

In this chapter, we present a preliminary experiment used to evaluate the classification performance of deSNN methods using the AER data obtained from the artificial silicon retina hardware.

7.1 Introduction

Evolving spiking neural networks (eSNN) are computational models that evolve new spiking neurons and new connections in the process of learning patterns from incoming data in an on-line mode. With the development of new techniques to capture spatio- and spectro-temporal data in a fast on-line mode, using for example address event representation (AER) such as the implemented one in the artificial retina and the artificial cochlea chips, and with the available SNN hardware technologies, new and more efficient methods for spatio-temporal pattern recognition (STPR) are needed.

This chapter uses a new eSNN model, the dynamic eSNN (deSNN), that utilizes both rank-order spike coding (ROSC), also known as time to first spike, and temporal spike coding (TSC). Each of these representations are implemented through different learning mechanisms, such as RO learning and temporal spike learning - spike driven synaptic plasticity (SDSP) rule. The deSNN model is demonstrated on a small scale moving object classification problem when AER data is collected with the use of an artificial retina camera. The new model is superior to eSNNm, eSNNs and SDSP-SNN in terms of learning time and accuracy for learning. It makes use of the order of spikes input information which is explicitly present in the AER data, while a temporal spike learning rule accounts for any consecutive spikes arriving on the same synapse that represent temporal components in the learned spatio-temporal pattern.

Spatio- and spectro- temporal data (SSTD), that are characterized by a strong temporal component, are the most common types of data collected in many domain areas, including engineering (e.g. speech and video data), bioinformatics (e.g. gene expression data), neuroinformatics (e.g. EEG, fMRI), ecology (e.g. establishment of species), environment (e.g. global warming phenomenon), medicine (e.g. patients risk of disease and recovery data), economics (e.g. financial time series), etc.

However, there is a lack of efficient methods for modeling such data and for spatio-temporal pattern recognition (STPR) that can facilitate the discovery of complex STP from streams of data and the prediction of new spatio-temporal events. The brain-inspired spiking neural networks (SNN) (Gerstner & Kistler, 2002b; Maass & Zador, 1999), are considered the third generation of neural networks. They are a promising paradigm for STPR as these new generation of computational models and systems are potentially capable of modeling complex information processes due to their ability to represent and integrate different information dimensions, such as time, space, frequency, phase, and to deal with large volumes of data in an adaptive and self-organizing manner.

With the development of new techniques to capture spatio-temporal data in a fast on-line mode, e.g. using address event representation (AER), such as the one implemented in the artificial retina chip (Delbruck, 2007) (see example in Figure 5.2.1) and the artificial cochlea chip (Van Schaik & Liu, 2005), and with the advanced SNN hardware technologies (Indiveri et al., 2011), new opportunities have been created for efficient STPR across domain areas.

However, the exploration of the new opportunities still requires efficient and suitable methods.

In the following section, the deSNN is demonstrated on a simple moving object classification problem where data was collected using AER in an artificial retina camera (Delbruck, 2007) (as explained in Chapter 5). A comparative analysis of results obtained with eSNN, deSNN, and an SNN that uses only SDSP learning rule shows the advantage of the proposed deSNN in terms of fast and accurate learning of AER data for STPR.

7.2 Experimental Setting and Results for deSNN

Here we use an AER dataset of a moving object classification collected through a silicon retina camera (Delbruck, 2007). The object is a moving irregular wooden bar in front of the camera. Two classes of movements are recorded, namely crash and no crash. For the crash samples, the object is recorded as it approaches the camera and for no crash other movements such as up/down motion at a fixed distance from the camera are recorded. The size of the recorded area is 7,000 pixels. Each type of movement is recorded 10 times, and five of the samples are used for training and five for testing. The following five models are created, trained and tested: SDSP, eSNNs, eSNNm, deSNNs and deSNNm. The parameter C for the $SDSP$, $deSNNm$ and $deSNNm$ is optimized between 0 and 1 (with 0.1 step). The parameters used in the models are presented in Table 7.2.1 and the classification results along with the number of training iterations are presented in Table 7.2.2.

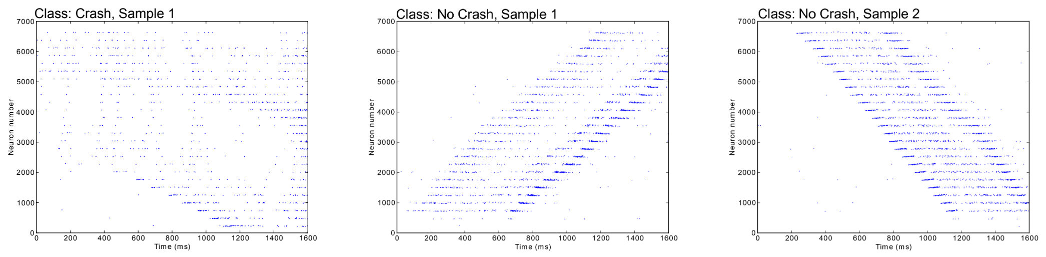


Figure 7.2.1: Raster plots for the AER encoded samples from the *Crash* and *No crash* classes. It can be seen that there is a similarity between the spike trains of Crash class, sample 1 (left figure) & No crash class, sample 2 (right figure).

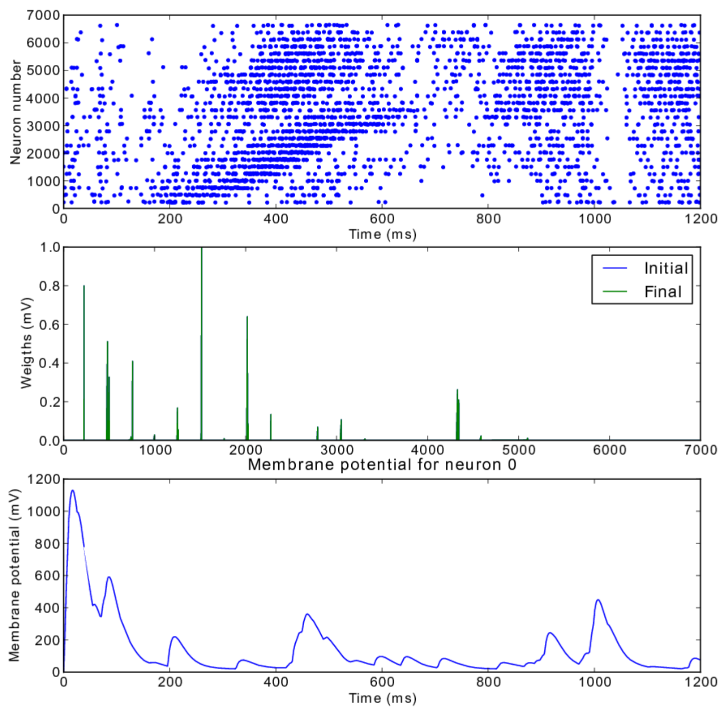


Figure 7.2.2: The figure shows the spike raster plot, weights change and the membrane potential (for neuron 0) for the eSNNs that utilizes rank order without the SDSP dynamics

Table 7.2.1: Parameter settings

For neurons and synapses	
Excitatory synapse time constant	2 ms
Inhibitory synapse time constant	5 ms
Neuron time constant (tau mem)	20 ms
Membrane leak	20 mV
Spike threshold (Vthr)	800 mV
Reset value	0 mV
Fixed inhibitory weight	0.20 volt
Fixed excitatory weight	0.40 volt
Thermal voltage	25 mV
Refractory period	4 ms
For learning related parameters (Fusi)	
Up/Down weight jumps (Vthm)	5 x (Vthr/8)
Calcium variable time constant (tau ca)	5 x (tau mem)
Steady-state asymptode for Calcium variable (wca)	50 mV
Stop-learning threshold 1 (stop if $Vca < thk1$)	1.7 x wca
Stop-learning threshold 2 (stop LTD if $Vca > thk2$)	2.2 x wca
Stop-learning threshold 2 (stop LTP if $Vca > thk3$)	(8 x wca) - wca
Plastic synapse (NMDA) time constant	9 ms
Plastic synapse high value (wp hi)	6 mvolt
Plastic synapse low value (wp lo)	0 mvolt
Bistability drift	0.25
Delta Weight	0.12 x wp hi
Other miscellaneous parameters / values	
Input Size	7000 spike train
Simulation time	1600 ms
mod (for rank order)	0.8

Table 7.2.2: Classification accuracy for SNN-based classifiers. The parameter C for the deSNNm and deSNNm has been optimized between 0 and 1 (with 0.1 step)

SNN Classifiers					
	SDSP SNN	eSNNs	eSNNm	deSNNs	deSNNm
Accuracy (%)	70	40	60	60	90
No. of training iteration	5	one-pass	one-pass	one-pass	one-pass

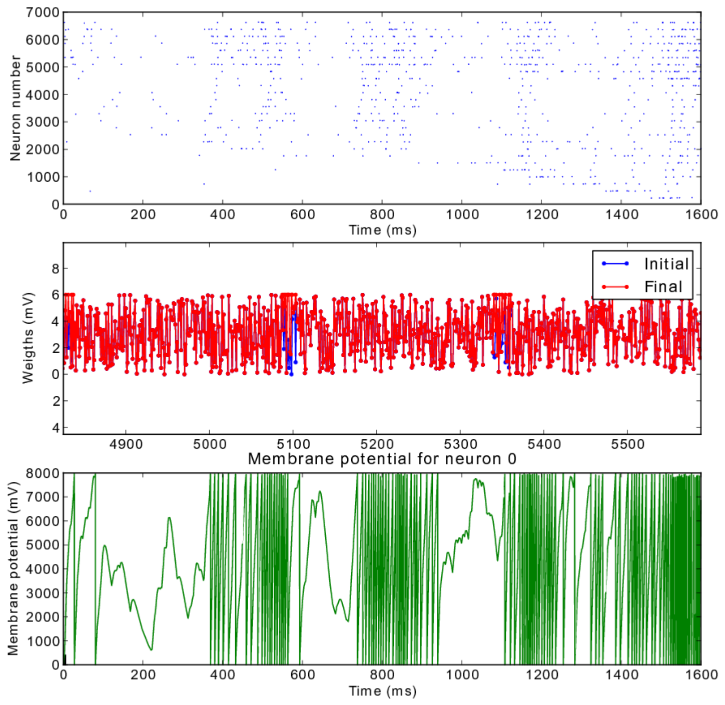


Figure 7.2.3: This figure shows the spike raster plot, weights change and the membrane potential (mV) for the deSNNs. From the weights and the membrane potential graph (of neuron 0) it can be seen that due to the SDSP, the synaptic weights adjustments are faster compared to Fig. 7.2.2, for the sample from the same class

The parameters shown in table 7.2.1 have been set based on trial and error method. Due to large number of parameters, this approach is very time-intensive. Amongst the five online one-pass learning algorithm, deSNNm method performed the best on this particular dataset yielding 90.00 percent accuracy. Other algorithms such as SDSP SNN, DeSNNs, eSNNs and eSNNm gave an accuracy of 70%, 60%, 40% and 60% respectively. The methods incorporating only the rank order coding (eSNNs, eSNNm) did not perform very well (see raster plots for different classes in Figure 7.2.1) because many of the samples from both classes have almost similar spike patterns. The eSNNs especially under-performs due to the absence of SDSP mechanism and dynamic synapses. On the other hand it can be seen that these spike based classifiers are robust to noise and handle the temporal aspect of the data very well. The noise refers to the events/spikes generated by AER in the absence of any motion or due to the flickering of fluorescent tube. Please refer to chapter 5 for more details on noise generated by AER. This experiment demonstrates the feasibility of spike-based classifiers. Compared to the traditional methods, this is a spike-based approach where the need to convert spike times into vectors is eliminated. They learn in on-line, incremental mode, through one pass of incoming data propagation through the system.

7.3 Summary

In this chapter we have shown the applicability of the two deSNN methods, namely deSNNm and deSNNs, on real world simple motion data captured using the AER artificial silicon retina. Due to the size of the input (spike trains), converting them into vectors for use with traditional machine learning algorithms can be computationally expensive or inconvenient. Furthermore, the deSNN has the ability to process the entirety of the spatio-temporal data in an online one-pass manner rather than in the traditional frame-by-frame approach. This study was recently published in Kasabov, Dhoble, Nuntalid, and Indiveri (2012). Chapter 8, introduces a new generic architecture for spatio-temporal pattern recognition (epSNNA-v) followed by its classification performance evaluation on human action recognition dataset obtained from the artificial silicon retina.

8 A Novel epSNNr Architecture for Visual Data (epSNNA-v)

This chapter introduces a new generic architecture called epSNNA-v. The epSNNA-v integrates the methods and architecture proposed in Chapter 4 and Chapter 6 with some existing approaches. Being an entirely spike-time based computational architecture, it complements the AER-based spike information encoding methods (utilized by artificial silicon retina) and the deSNN methods.

Furthermore, Hebbian learning is also introduced in the liquid reservoir. We hypothesize that the introduction of a learning rule such as Spike-Timing dependent plasticity (STDP) into the probabilistic reservoir will increase its separability property, thereby resulting in an increased classification performance.

8.1 A Novel Evolving Probabilistic Spiking Neural Network Architecture for spatio-temporal data (epSNNA-v)

In our study so far we have proposed several classification methods such as deSNNm and deSNNs, and an architecture called epSNNr that utilizes several stochastic neural models such as Noisy Reset (NR), (Continuous) Noisy Threshold (NT) and Step-wise Noisy Threshold (SNT) in the reservoir. Here we consider the earlier proposed methods (deSNN) and architecture (epSNNr), and incorporate them into one generic architecture called evolving probabilistic spiking neural network architecture for spatio-temporal pattern recognition (epSNNA-v). This architecture is based entirely on a spike-based pattern recognition approach that has been briefly discussed in Chapter 2 (see Figure 2.6.1).

Figure 8.1.1 depicts the proposed novel generic architecture that comprises the following modules:

- Data Acquisition Module;

- Transformation Module;
- Learning Module.

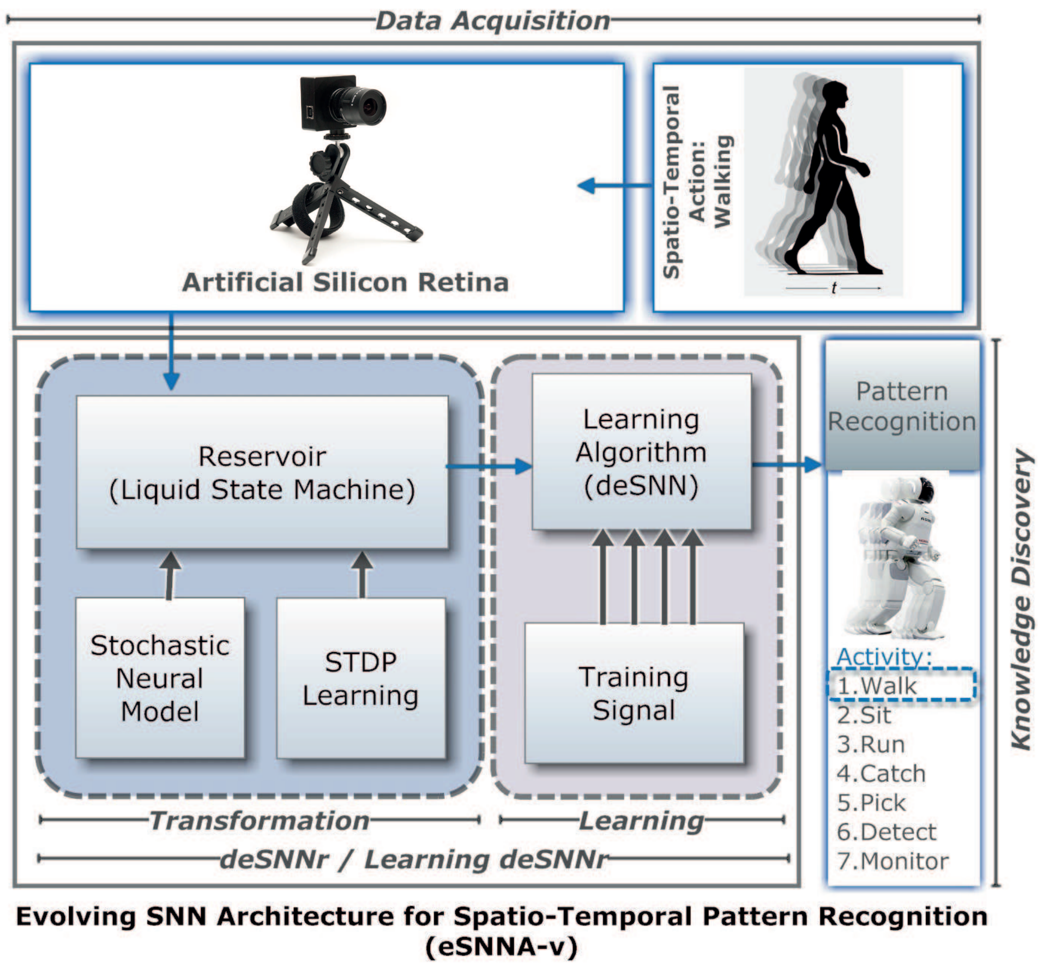


Figure 8.1.1: A novel generic architecture is an evolving probabilistic spiking neural network architecture for spatio-temporal pattern recognition (epSNNA-v). It consists of three mail modules.

The characteristics and specifics of the above mentioned generic architecture modules are briefly explained below.

8.2 Data Acquisition Module

The data acquisition module captures the spatio-temporal data (such as human motion /action) via artificial silicon retina (DVS 128). Alternatively, the earlier mentioned software simulator can be used (see Chapter 5). The simulator generates Address Event Representation (AER) spikes representing the spatio-temporal actions similar to that of the artificial silicon retina (DVS 128). The outputs of the silicon retina and the AER software simulator are both in the form of spike trains. The spike trains are then fed into the transformation module.

8.3 Transformation Module

From Figure 8.1.1, it can be seen that three sub-modules collectively make up the transformation module. These sub-modules are:

- Reservoir;
- Stochastic Neural Models;
- The STDP learning mechanism.

In our study we use Liquid State Machine (LSM) for the reservoir. The neuron models that can be, and have been independently used in our study are the standard Leaky Integrate and Fire (LIF) neurons and three stochastic neural models called Noisy Reset (NR), (Continuous) Noisy Threshold (NT) and Step-wise Noisy Threshold (SNT). Our earlier proposed architecture epSNNr introduced in Chapter 4 was constructed using the stochastic neural models with the LSM reservoir. This transformation module can be regarded to certain extent as the epSNNr architecture. The python implementation of the epSNNr is provided in Appendix A, Section A.4.1.

The transformation module also includes the STDP learning rule that can be incorporated in the LSM reservoir. The integration of the LSM reservoir with Stochastic Neural Models and deSNN algorithm results in a hybrid algorithm which we refer to as the deSNNr. Also, the inclusion of STDP learning in the deSNNr's reservoir results into another hybrid algorithm called learning deSNNr. Here the concept of unsupervised SDSP learning from the deSNN has been applied to learning deSNNr, so the initial synaptic weights are modified on a spike-time basis according to the new pattern during the STDP learning phase. The transformation module is so called because the reservoir acts as a spatio-temporal filter and the module allows the data to be transformed into a higher dimension space. This transformation therefore improves the inter and intra class separability of the data and is especially useful when applied to classification tasks. The spikes obtained from the transformation module is fed into the learning module.

8.3.1 Learning deSNNr

Kempter, Gerstner, and Hemmen (2001) state that applying STDP to a network of dynamic synapses can further improve the computational capability of the system. Based on this hypothesis it is claimed that high dynamics of the liquid reservoir will result in longer-lasting short-term memory and improved separation property while having lower computational costs (Vreeken, 2004). Even though on an individual synaptic level STDP acts as a destabilizing factor (Kempter, Gerstner, & Van Hemmen, 1999), it is claimed that on large scale it has the opposite effect, i.e. STDP helps in regulating network homeostasis (Abbott & Nelson, 2000). This feature is highly desirable. S. Song, Miller, Abbott, et al. (2000) state that introduction of such learning rule into a network with dynamic synapses will allow the network to adapt its activity to the inputs it receives. Therefore, based on the above argument, we hypothesize that introduction of STDP into a probabilistic reservoir (one with stochastic neuron models) should further improve the separability property of the liquid while lowering the computational costs. This hypothesis is tested through various experiments that are presented in later Chapter 9 and Chapter 11. The python implementation of the learning reservoir is presented in Appendix A, Section A.4.2.

8.4 Learning Module

Technically the learning module can incorporate any learning algorithm. However, it is more convenient to use spiking neural network based learning algorithms, since the output from the transformation module is in spikes. In our study, we use the earlier proposed deSNNm and deSNNs methods as the learning algorithms in this module. The function and performance of these methods have been discussed in Chapter 6. The outcome from the learning module results in knowledge discovery.

8.5 Summary

This chapter introduces a new generic architecture called epSNNA-v. The epSNNA-v architecture integrates the earlier proposed methods and architecture with some of the existing approaches. epSNNA-v is based on a novel generic spike-based spatio-temporal pattern recognition approach that easily complements neuromorphic hardware such as the artificial silicon retina. In a nutshell, the proposed architecture is characterized by: one-pass propagation of SSTD during learning; utilizing STDP/SDSP and dynamic synapses in learning deSNNr that are modifiable during both learning and recall; it is realized with an entirely spike-based spatio-temporal pattern recognition approach.

In the following chapter, we present a case study that evaluates the classification performance of the epSNNA-v architecture. The case study makes use of human action recognition (HAR) data captured by the artificial silicon retina.

9 Human Action Recognition with epSNNA-v

Human action recognition in machine learning is traditionally performed by marking particular sequences of images with action labels. The human action pattern recognition has many applications both online and offline in various domains such as human-computer interaction, visual surveillance and video retrieval.

The interest in human action recognition is driven by its potential applicability in many fields. As a video surveillance application, it will allow automatic detection of any possible suspicious activities. In gaming and simulations it will provide the user with a more interactive, intuitive and immersible experience. The automatic detection of a particular human gesture or motion (e.g dance moves) will allow the user to search or retrieve related video more efficiently.

It is a challenging task due to various problems such as the numerous variances in recording settings and action performance.

In this study, we take a spike-based pattern recognition approach using epSNNA-v to solve these problems. Below we discuss the characteristics of the human action recognition data we have used in our case study followed by the performance evaluation of the epSNNA-v architecture.

9.1 Case Study: AER based Human Action Recognition

The Address Event Representation (AER) based Human Action Recognition (HAR) dataset that is used in this case study has been acquired from the artificial silicon retina (DVS-128). The data consists of three human motions/actions, namely boxing, hand waving and hand clapping (i.e. three classes). Each class has 22 samples, where 11 samples were recorded under artificial (fluorescent) light conditions and the other 11 - under natural light conditions. Therefore, in total the dataset consists of 66 samples. As seen in Figure 5.1.5, under fluorescent light conditions the average level of noise is around 87.71% ± 11.41 and under natural light conditions the average level of noise is around 50.00% ±17.05. This extra noise under the fluorescent light conditions is due to flickering of the light source (see Figure 5.1.3). Each sample consists of repetitive human actions, where each of the actions have varying execution time. Since the average length (i.e. the entire video sample length) of the samples is 1600 milliseconds, we considered 1600 milliseconds for our spiking neural network simulation time. Also, 16,000 input neurons are required to represent the 120×120 DVS-128 pixel size.

The difficulty in interpreting the data for classification task can be seen in Figure 9.1.1. Apart for the noise inherent in the dataset, we can also see spatial and temporal differences between the samples belonging to the same classes.

9.2 Experimental settings and results for deSNN

Compared to the previous experiment (see Table 7.2.2) on the AER simple motion recognition, the outcome of this experiment which uses real world dataset is quite different. From Table 9.2.2, it can be seen that the eSNNs that utilizes rank order (RO) learning performs significantly better than eSNNm, when compared to the previous experiment of AER-based simple motion recognition. However, when we introduce Fusi's Spike Driven Synaptic Plasticity (SDSP) (i.e. deSNNs), it under-performs when compared to eSNNs. On the other hand, it is the opposite for eSNNm, where the introduction of Fusi's SDSP (i.e. deSNNm) enhances its performance on this particular dataset resulting in an accuracy of 70.20%±5.0011. This shows that Fusi's SDSP, which is actually a supervised learner, complements the Rank Order Coding (ROC)

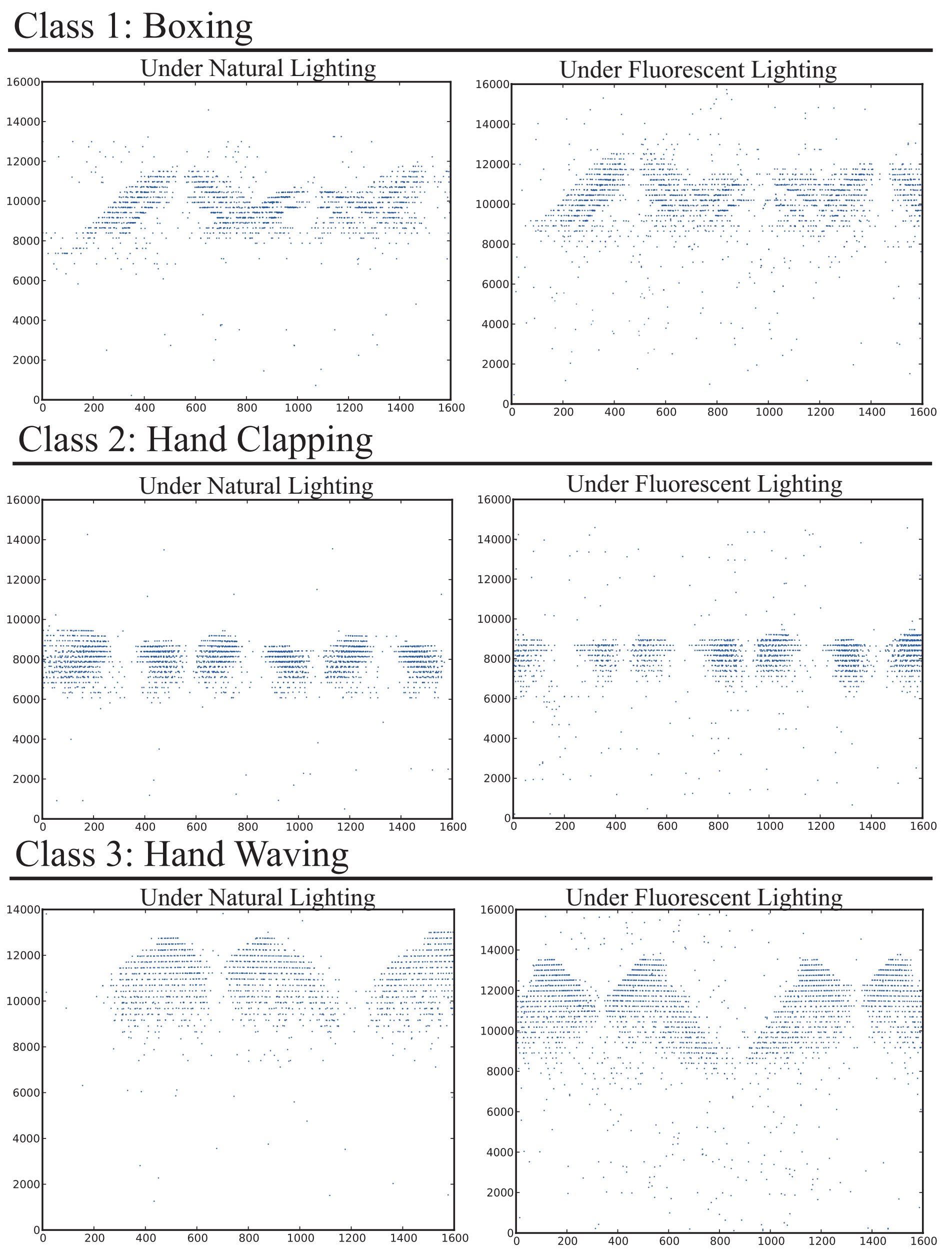


Figure 9.1.1: This figure shows the raster plots of two samples, for each of the three classes, where the samples in the first column were obtained under natural lighting conditions, and those in the second column were obtained under florescent lighting conditions. The higher level of noise present in the samples obtained under florescent lighting is apparent.

Table 9.2.1: Parameter settings for SNN classifiers

For neurons and synapses	
Excitatory synapse time constant	2 ms
Inhibitory synapse time constant	5 ms
Neuron time constant (tau mem)	20 ms
Membrane leak	20 mV
Spike threshold (Vthr)	800 mV
Reset value	0 mV
Fixed inhibitory weight	0.20 volt
Fixed excitatory weight	0.40 volt
Thermal voltage	25 mV
Refractory period	4 ms
For learning related parameters (Fusi)	
Up/Down weight jumps (Vthm)	5 x (Vthr/8)
Calcium variable time constant (tau ca)	5 x (tau mem)
Steady-state asymptode for Calcium variable (*wca*)	50 mV
Stop-learning threshold 1 (stop if $Vca < thk1$)	1.7 x wca
Stop-learning threshold 2 (stop LTD if $Vca > thk2$)	2.2 x wca
Stop-learning threshold 2 (stop LTP if $Vca > thk3$)	(8 x wca) - wca
Plastic synapse (NMDA) time constant	9 ms
Plastic synapse high value (wp hi)	6 mvolt
Plastic synapse low value (wp lo)	0 mvolt
Bistability drift	0.25
Delta Weight	0.12 x wp hi
Other miscellaneous parameters / values	
Input Size	16,000 spike train
Simulation time	1,600 ms
mod (for rank order)	0.8

learning. Compared to the previous experiment (see Table 7.2.2), when we use the SDSP SNN algorithm, the Hand Waving and Boxing classes are misclassified as Hand Clapping class.

However, from the two experiments, where deSNNm came out as the winning algorithm, it can be assumed that deSNNm is a suitable spiking neural network algorithm for spatio-temporal learning or more specifically, for AER-based

Table 9.2.2: Classification accuracy for four SNN-based classifiers. The parameter C for the eSNNm and deSNNm has been optimized between 0 and 1 (with 0.1 step). NL represents natural lighting conditions and FL stands for under fluorescent lighting conditions.

SNN Classifiers				
Accuracy	eSNNs	eSNNm	deSNNs	deSNNm
NL (%)	50.93±11.1895	33.33±0.0000	43.87±6.1879	70.20±5.0011
FL (%)	47.22±14.4323	35.76±2.4784	43.87±12.2248	54.56±3.3667

human action/motion recognition.

9.2.1 Hardware feasibility

The artificial silicon retina (DVS-128) was developed in the Institute of Neuroinformatics (INI), University of Zurich (Delbruck, 2007). They have also developed neuromorphic hardware which can be potentially used to implement the deSNN method on the hardware. As a part of research collaboration (Kasabov, Dhoble, Nuntalid, Mohhemed, et al., 2012), a pilot experiment has been carried out as a feasibility study. Since the neuromorphic hardware has inherent (thermal) noise (as do any other electronic devices), in the simulated hardware feasibility study, we have added 25 percent noise to all parameters. More details on the parameter setting with simulated thermal noise and the results are given below.

Table 9.2.3: Parameter settings with added 25% noise to test hardware feasibility. This is the amount of thermal/electrical noise usually found in the neuromorphic hardware.

For neurons and synapses	
Excitatory synapse time constant	2.5 ms
Inhibitory synapse time constant	6.25 ms
Neuron time constant (tau mem)	25 ms
Membrane leak	25 mV
Spike threshold (Vthr)	1000 mV
Reset value	0 mV
Fixed inhibitory weight	0.25 volt
Fixed excitatory weight	0.5 volt
Thermal voltage	31.25 mV
Refractory period	5 ms
For learning related parameters (Fusi)	
Up/Down weight jumps (Vthm)	6.25 x (Vthr/8)
Calcium variable time constant (tau ca)	6.25 x (tau mem)
Steady-state asymptode for Calcium variable (wca)	62.5 mV
Stop-learning threshold 1 (stop if $Vca < thk1$)	2.125 x wca
Stop-learning threshold 2 (stop LTD if $Vca > thk2$)	2.75 x wca
Stop-learning threshold 2 (stop LTP if $Vca > thk3$)	(10 x wca) - wca
Plastic synapse (NMDA) time constant	11.25 ms
Plastic synapse high value (wp hi)	7.5 mvolt
Plastic synapse low value (wp lo)	0 mvolt
Bistability drift	0.3125
Delta Weight	0.15 x wp hi
Other miscellaneous parameters / values	
Input Size	16,000 spike train
Simulation time	1,600 ms
mod (for rank order)	0.8

Experimental settings and results for deSNN with 25% simulated thermal noise

On comparing the results of deSNN with 25 percent thermal noise (see Table 9.2.4) with deSNN without simulated thermal noise (see Table 9.2.2), it can be seen that the deSNN method works well without significant loss in classification

Table 9.2.4: Classification accuracy for two SNN-based classifiers with 25% noise added to all parameter settings. Data recorded under natural lighting conditions is used for this hardware feasibility test.

SNN Classifiers		
	deSNNs	deSNNm
Accuracy (%)	42.99±28.8689	70.83±21.3822

performance. It also shows us the robustness of the deSNN method against noise. However, as future works, more experiments would be needed with various datasets and optimized parameter settings in order to conclude the feasibility of deSNN methods on neuromorphic hardware.

9.3 Experimental settings and results for deSNNr and learning deSNNr

The parameters for the deSNNm and deSNNs have been kept the same in order to have an unbiased classification performance comparison (see table 9.2.1). The LSM network topology is constructed using 16000 neurons (N) arranged in a three-dimensional topology of 20 x 20 x 40 neurons. Other LSM and neuron parameters are given in table 9.3.1.

Table 9.3.2 shows the learning deSNNr's classification performance under different stochastic neuronal models (Nuntalid, Dhoble, & Kasabov, 2011) such as Leaky-Integrate and Fire (LIF), Noisy Reset Model (NR), Noisy Threshold Model (NT) and Step-wise Noisy Threshold Model (SNT).

Table 9.3.1: The table provides the parameter settings that are used in our experimental setup for the neuron models and LSM

Parameters	*Value/s*
For Neuron	
Time Constance	10 ms
Reset Potential	0 mV
Initial Firing Threshold	10 mV
Standard Deviation of reset fluctuation	3 mV
Standard Deviation of Step-wise Firing Threshold	2 mV
Standard Deviation of Continuous Firing Threshold	1 mV
For LSM	
Simulation Time	1600 ms
Number of Neurons	16000
Excitatory to Inhibitory Neuron Ratio	4:1
Input Neurons Connection Probability	0.2
Input Neurons Connection Weight	0.8 mV

Table 9.3.2: Classification accuracy for deSNNr and learning deSNNr with standard LIF and stochastic neural models in reservoir.

SNN Classifiers (deSNNr)				
Accuracy	LIF	NR	NT	SNT
deSNNm (%)	73.32±3.2636	75.98±15.1028	67.48±09.6182	74.79±16.3201
deSNNs (%)	69.96±1.2780	69.54±12.4604	77.69±16.8379	68.16±14.7022
deSNNr classifiers with STDP learning rule (learning deSNNr)				
deSNNm (%)	76.00±08.2715	87.53±9.4280	75.33±18.4580	60.00±02.7002
deSNNs (%)	73.83±09.8732	78.30±4.4263	82.33±22.4810	76.00±10.1756

As established in one of our previous studies (Kasabov, Dhoble, Nuntalid, & Indiveri, 2012), the deSNN models resulted in a better classification performance when compared with other SNN models that use either rank-order or STDP learning rule. The reason is that the deSNN makes use of both the information contained in the order of the first input spikes present in input data streams (this is crucial to consider in some tasks) and the information contained in the timing of the following spikes that is learned by the dynamic synapses as a whole spatio-temporal pattern.

The proposed learning deSNNr, which incorporates STDP learning rule in the LSM reservoir, was also shown to have far superior performance in terms of classification accuracy when compared to the deSNNr approach. Previous studies (Maass & Sontag, 2000; Natschlager & Maass, 2002; Brader et al., 2007) that have utilized STDP learning have also shown that such networks can approximate a very rich class of non-linear filters and have been successfully utilized in practical applications. Hence, due to the STDP's ability to function as memory buffers and to account for the precise time of spikes, the reservoir with STDP learning rule has a higher classification performance in comparison to the results from reservoir without STDP learning rule (deSNNr).

9.4 Summary

In this chapter we have used AER-based human action recognition as a case study for evaluating the earlier proposed generic framework epSNNA-v. We have compared the proposed architecture with standalone SNN-based classifiers such as eSNN and deSNN. On comparison, the classification performance was shown to be superior when using hybrid algorithms such as deSNNr and learning deSNNr. It is to be noted that in our study we do not focus on optimizing any parameters or feature selection methods, but on the feasibility and applicability of SNN-based approach for solving real world problems. In the next chapter we focus on the epSNNA-s architecture for spectro-temporal pattern recognition.

10

A Novel eSNN Architecture for Spectro-Temporal Data (epSNNA-s)

A lot of importance has been placed on the development of autonomous machine learning systems with various practical applications. Recognition of patterns in spectro-temporal data is one of the many challenging tasks. In this chapter we propose an evolving spiking neural network architecture for spectro-temporal pattern recognition (epSNNA-s).

10.1 A Novel Evolving Spiking Neural Network Architecture for Spectro-Temporal Data (epSNNA-s)

Figure 10.1.1 shows the generic spectro-temporal pattern recognition framework we have used in our study. The proposed epSNNA-s architecture is quite similar to the earlier introduced framework (epSNNA-v) for spatio-temporal pattern recognitions. However, it is also different in terms of the use of sound to spike encoding scheme.

Many of its modules are similar to those in the epSNNA-v architecture and the specifics and characteristics of these modules have been explained in Chapter 8. Therefore, in this chapter those modules are only briefly reintroduced.

The framework can be split into the following modules:

- Data Acquisition Module;
- Transformation Module;
- Learning Module.

10.2 Data Acquisition Module

The data acquisition module acquires the spectro-temporal data and represents the data as spikes that in turn can be fed to the transformation module. In the

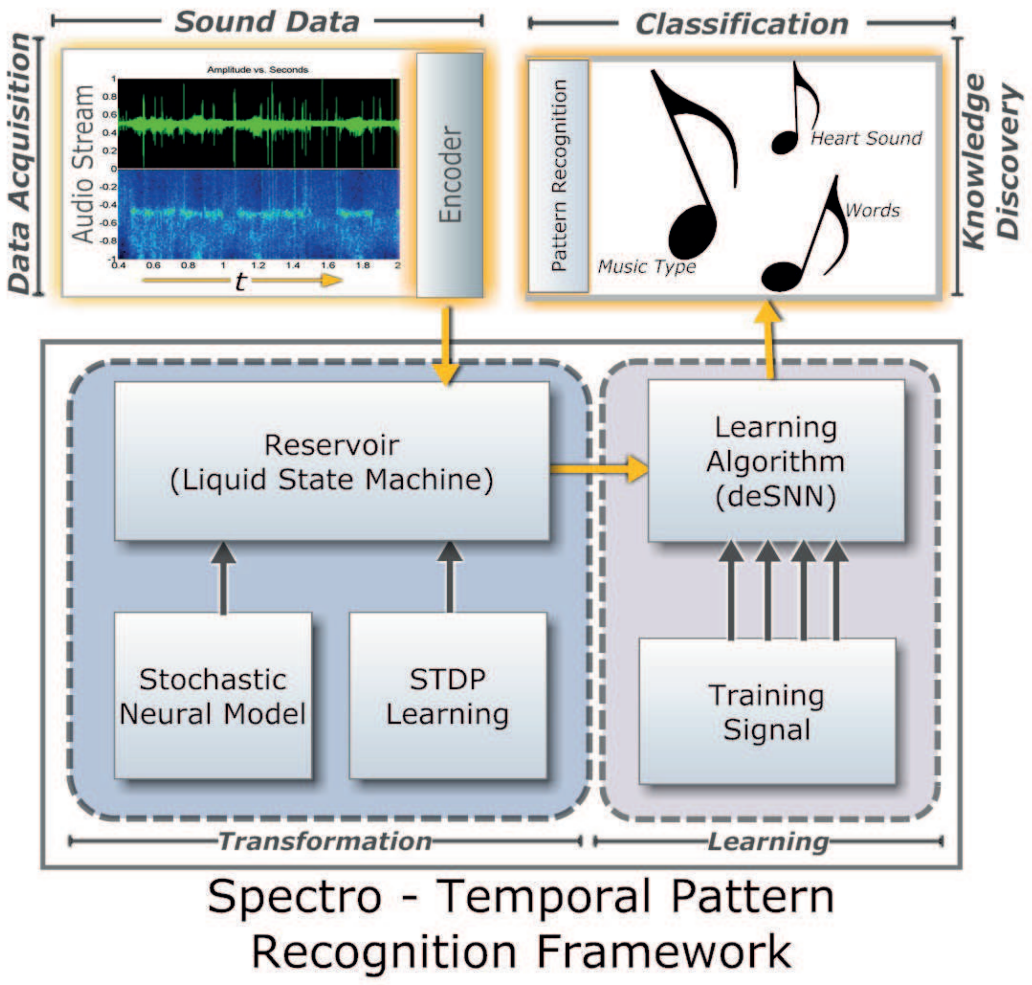


Figure 10.1.1: A generic spectro-temporal pattern recognition framework (epSNNA-s).

following section we discuss the spectro-temporal data representation and the spike encoding methods incorporated into the epSNNA-s framework.

10.2.1 Spectro-temporal data representation

The initial step in biological auditory systems is the conversion of sound waves into spikes that can be used for processing the brain. The task of changing sound vibration into spike information is carried out at the basilar membrane (BM), which is a structure in our inner ear (see Figure 10.2.1). It is also known as cochlea, and its function can be considered similar to that of a vibrating string (Andringa, Niessen, & Nillesen, 2004; Muthusamy, Cole, & Slaney, 1990). Andringa, Niessen, and Nillesen (2004) explain that the basilar membrane consists of hair cells called Stereocillia and has spiking neurons, and it behaves in a non-linear manner. This provides the cochlea with an ability to process frequency within a working range of over 100 decibel (dB).

The two most important features of the basilar membrane is firstly, its ability to maintain the continuity in time, place and frequency and secondly, it is sensitive to different frequencies at different positions (Andringa et al., 2004; Wikipedia, 2012). This ability of maintaining the continuity in time, place and frequency is a very important feature that is not commonly found in many speech signal preprocessing techniques such as the commonly used Fast Fourier Transform (FFT). Hence, apart from the biological plausibility, this continuity preserving feature is one of the reasons we have utilized cochleagram method. Abdollahi, Valavi, and Noubari (2009) state that the spectro-temporal representation of cochleagram is structurally the same as mel-spectrogram. However, in the cochleagram representation, the Equivalent Rectangular Bandwidth (ERB) scale is used instead of Mel-scale and the filters are gammatone filter shaped. The cochlear filtering is often carried out by gammatone filterbank (or other cochlear filtering models), followed by nonlinear rectification; the latter corresponds to hair cell transduction by either a hair cell model or simple compression operations (Goodman & Brette, 2010).

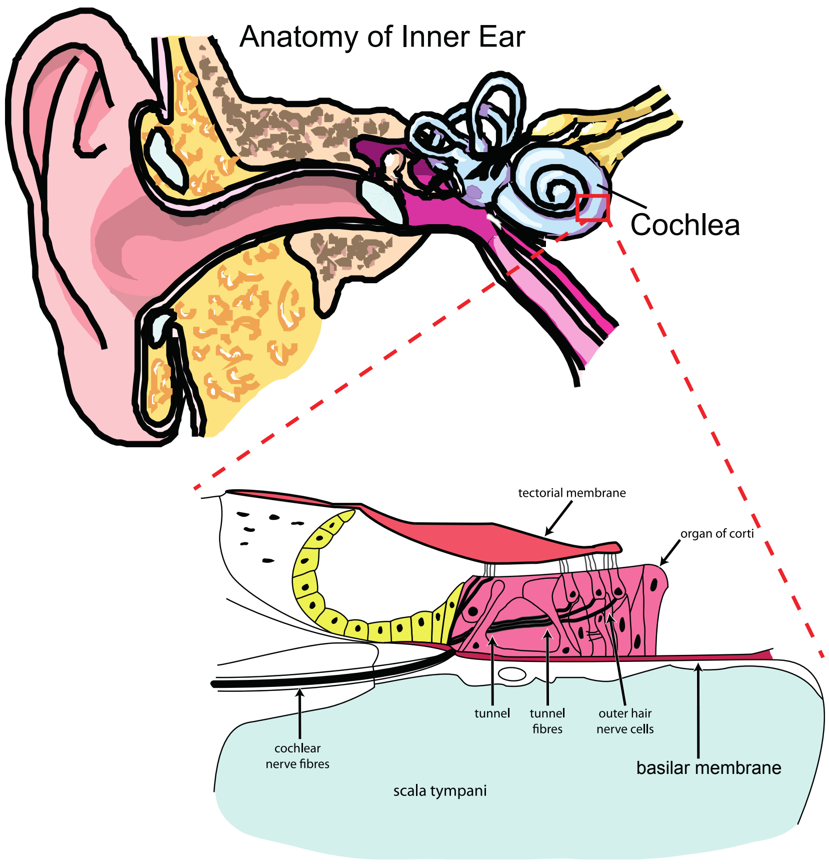


Figure 10.2.1: This figure shows the anatomy of human inner ear. The anatomy of the basilar membrane (BM) and its location in the cochlea is also depicted in this figure. The cochleagram's function is based on the working of cochlea. Adapted from Wikipedia (2012).

This task has been carried out using Brian neural network simulator, since it provides the built-in functions and auditory models to carry out the above mentioned tasks (Goodman & Brette, 2010). For more detailed information on cochleagram, refer to Andringa, Niessen, and Nillesen (2004); Goodman and Brette (2010).

Also, the study by Muthusamy, Cole, and Slaney (1990), where only the coefficient of cochleagram (discrete Fourier transform) was used, the classification performance of cochleagram was shown to be better than spectrogram. In another study by Abdollahi, Valavi, and Noubari (2009), different spectro-temporal representations methods such as mel-spectrogram, cochleagram and auditory spectrogram were used and similarly to the previous study (Muthusamy et al., 1990), the performance was found to be superior for spectro-temporal representations by cochleagram. There are some spectro-temporal representation (or extended) methods (Gao, Woo, & Dlay, 2012; Narayanan & Wang, 2012) that perform better than the standard cochleagram. The feature extraction step in machine learning is essential for knowledge discovery. Therefore an appropriate spike based feature extraction methods should be as per the dataset characteristics. However, in our study we have used cochleagram method on all the spectro-temporal data for its simplicity, biological plausibility and fast implementation. This will allow us to evaluate the proposed framework in comparison to other spiking neural network based approaches, such as eSNN and deSNN, without any bias.

In the following section, we explain the cochleagram-based spike encoding method in more details.

10.2.2 Cochleagram-based spike encoding

After acquiring an appropriate spectro-temporal representation, which in our case is done by the cochleagram method, the next step is to convert the analogue signal obtained from cochleagram into biologically realistic spike trains. Various approaches are presented in relevant literature (De Garis, Nawa, Hough, & Korkin, 1999; Schrauwen & Van Campenhout, 2003) that allow conversion of analogue data to digital spike trains. These are referred to as spiker algorithm since they convert continuous data into discrete spike timing. Two such commonly used algorithms (Verstraeten et al., 2007; Wall, McGinnity, & Maguire, 2011; Verstraeten, Schrauwen, & Stroobandt, 2005; De Garis, Korkin, Gers, Nawa, & Hough, 2000; Nuntalid et al., 2011) are namely Hough Spiker Algorithm (HSA) by De Garis, Nawa, Hough, and Korkin (1999), and Bens Spiker Algorithm (BSA) developed by (Schrauwen & Van Campenhout, 2003). Glackin et al. (2011) explain that both algorithms function by utilizing a convolution/deconvolution filter that is optimized for encoding and decoding by means of genetic algorithm in order to reduce the error in the encoding and decoding process.

In our study, we take a straightforward approach, where we use the continuous values obtained from cochleagram to modify the current injected into a leaky integrate-and-fire (LIF) neuron (see Figure 10.2.2) (Glackin et al., 2011). Injecting the data from cochleagram into the leaky integrate-and-fire (LIF) neuron results in the firing of neuron(s). The rate of firing depends on the data, various neurons parameters and the network parameters. According to Glackin et al. (2011), since communication between some neurons is dependent on spike-timing, the spike trains obtained from cochleagram may be thought of as an analogue to digital spike conversion. For more details on the spectro-temporal spike encoding, refer to (Goodman & Brette, 2010; Glackin et al., 2011).

The spike trains are fed into the transformation module for further processing.

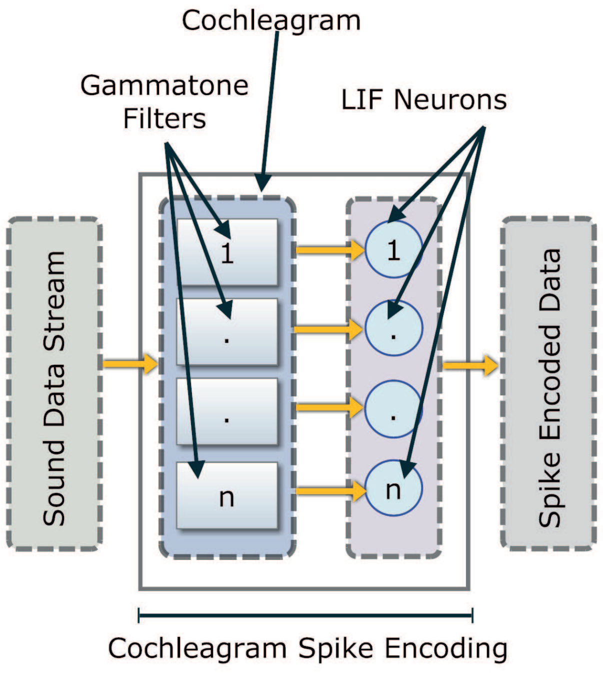


Figure 10.2.2: The above figure represents the cochleagram based spike encoding method. From the above figure, it can be seen that we use the same number of LIF neurons as the number of gammatone filters in cochleagram.

10.3 Transformation Module

It can be seen in Figure 10.1.1 that three sub-modules collectively make up the transformation module. These sub modules are:

- Reservoir;
- Stochastic Neural Model/s;
- The Spike-Timing dependent plasticity (STDP) learning mechanism.

The neuron model consists of the standard Leaky Integrate and Fire (LIF) neurons and three stochastic neural models called Noisy Reset (NR), (Continuous) Noisy Threshold (NR) and Step-wise Noisy Threshold (SNT). This transformation module can be regarded as an extension to the epSNNr architecture. The transformation module also includes the STDP learning mechanism that can be incorporated in the LSM reservoir. The integration of the Liquid State Machine (LSM) reservoir with Stochastic Neural Models and STDP learning rule produces a hybrid algorithm which we refer to as the deSNNr and learning deSNNr. The spikes obtained from the transformation module are fed into the learning module.

10.4 Learning Module

In this experiment, we have used the earlier proposed methods in this thesis deSNNm and deSNNs as the learning algorithms in this module. The function and performance of these methods are discussed in Chapter 6. The outcome from the learning module results in knowledge discovery.

In a nutshell, the sound data is first encoded into spikes. The spikes are then fed into the Liquid State Machine (LSM) reservoir. The LSM can be constructed using several neuronal models. Apart from the standard leaky integrate-and-fire (LIF) neural model, in our study we have used three other stochastic neural models, namely Noisy Reset Model (NR), Noisy Threshold Model (NT) and Step-wise Noisy Threshold Model (SNT) for constructing the reservoir. Also, when the reservoir makes use of STDP learning rule, in our study we refer to this approach as 'learning deSNNr'; when STDP learning is not used, we refer to the model as 'deSNNr'. We then feed the reservoir output to the learning algorithms.

10.5 Summary

In this chapter we present a new generic framework epSNNA-s for spectro-temporal pattern recognition. The framework structure is similar to the epSNNA-v architecture. We have also provided and explained a simple cochleagram-based spike encoding method that is incorporated into this framework. It is to be noted that the cochleagram-based spike encoding method is not suitable for all spectro-temporal datasets. The spike encoding module can be replaced with other suitable spike encoding schemes as per the dataset characteristics and knowledge discovery requirements. In the next chapter we present two case studies that evaluate the performance of the proposed epSNNA-s framework against standalone SNN-based classifiers such as eSNN and deSNN.

11 Spectro-Temporal Pattern Recognition using the epSNNA-s

Previous studies by Verstraeten, Schrauwen, Stroobandt, and Campenhout (2005), that have utilized Liquid State Machine (LSM) reservoir for spectro-temporal pattern recognition have shown to produce very high classification performance on various benchmark datasets. Other studies (Uysal, Sathyendra, & Harris, 2007; Abdollahi & Liu, 2011), have focused on spike feature extraction methods and have yielded great results using traditional machine learning algorithms. In our study, we use a simple straightforward cochleagram-based spike encoding method in order to demonstrate the performance of epSNNA-s architecture against standalone spiking neural network (SNN) classifiers such as eSNN and deSNN.

Many spiking neural networks (SNN) based models have shown the ability of capturing spatial and temporal data. In our study we have used one such SNN algorithm called evolving SNN (eSNN) and its extension dynamic eSNN (deSNN) (Kasabov, Dhoble, Nuntalid, & Indiveri, 2012; Dhoble et al., 2012). These algorithms utilize a one-pass rank-order learning along with a mechanism to evolve a new spiking neuron and new connections to learn new patterns from incoming data on the basis of SpikeTiming Dependent Plasticity (STDP) and its variant Spike Driven Synaptic Plasticity (SDSP). Also, we have compared the eSNN and deSNN methods with our epSNNA-s architecture that consists of deSNNr and learning deSNNr methods which incorporate STDP learning rule in the LSM reservoir.

In the initial experiment we use a standalone Evolving Spiking Neural Network (eSNN) and Dynamic Evolving Spiking Neural Network (deSNN) algorithms (Kasabov, Dhoble, Nuntalid, & Indiveri, 2012) for classification performance evaluation. Further details about the dataset, parameter configurations, experiment setup and results are provided in the following sections.

11.1 Spectro-Temporal Dataset

11.1.1 Dataset 1: Heart Sound

In this experiment, for the spectro-temporal pattern recognition we have used 'The PASCAL Classifying Heart Sounds Challenge' dataset (Bentley, Nordehn, Coimbra, & Mannor, 2011).

According to the 2012 World Health Organization report (WHO, 2012), cardiovascular diseases (CVDs) results in more deaths annually than from any other cause. In 2008, approximately 17.3 million people died from cardiovascular diseases, which represents 30% of all deaths globally. From this, 6.2 million deaths were due to stroke and 7.3 million were due to coronary heart disease. By 2030, the predicted number of deaths due to cardiovascular diseases (mainly from heart disease and stroke) is estimated to be almost 25 million. Evidently, methods that can help in the early detection of heart disease signs can have a major impact on human health. According to Bentley, Nordehn, Coimbra, and Mannor (2011), the Heart Sounds challenge is to be used for producing methods to help detect signs of heart diseases. The first level of screening of cardiac pathologies can be performed both in a hospital environment by a doctor (using a digital stethoscope) and at home by the patient (using a mobile device). The data used in our case study has been collected in real-world conditions and therefore has background noise of every conceivable type.

Bentley, Nordehn, Coimbra, and Mannor (2011) state that the differences between heart sounds corresponding to different heart symptoms can also be

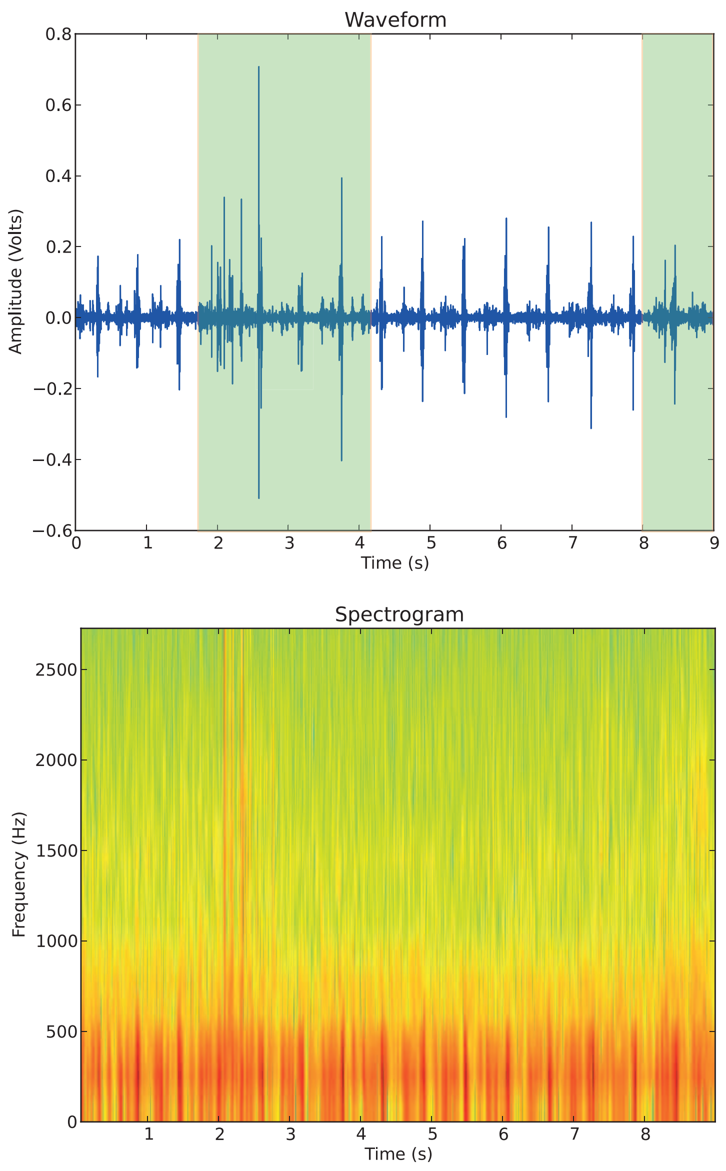


Figure 11.1.1: This figure shows the waveform and spectrogram of heart murmur sound sample. The region in the waveform graph that is highlighted in light green shows significant presence of noise especially within the 2 - 2.5 seconds range. This noise can also be seen in the spectrogram.

extremely subtle and challenging to separate. Success in classifying this form of data requires extremely robust classifiers. Despite its medical significance, to date this remains a relatively unexplored application for machine learning.

Motivated by this, we have decided to use the 'The PASCAL Classifying Heart Sounds Challenge' dataset. This will not only allow us to test the robustness of our methods but also test the real world application feasibility of our methods.

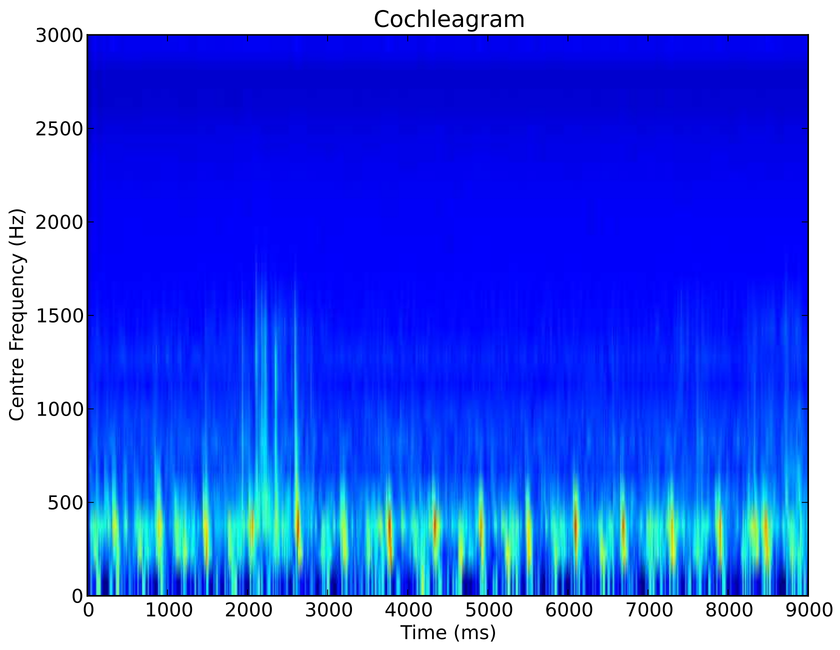


Figure 11.1.2: This figure shows the cochleagram for the heart murmur sample whose waveform and spectrogram are shown in figure 11.1.1. This figure shows the difference in sound processing between standard cochleagram and spectrogram (from figure 11.1.1), where the cochleagram-processed sound has significantly less noise.

In this experiment, we use Dataset A. The data belonging to Dataset A has been gathered from the general public via the iStethoscope Pro iPhone app (Bentley et al., 2011). The classification task is to check if our proposed method can classify real heart audio (also known as beat classification) into one of the two categories (i.e. 1 - Heart Normal Sound and 2 - Heart Murmur Sound). Here we consider a subset of the original dataset that has a total of four classes.

For the cochleagram spike encoding, in our case, we have used 20 gammatone filters covering the human auditory range of 20 Hz to 20 kHz followed by half-wave rectification, cube root compression (Goodman & Brette, 2010). Instead of the 3000 gammatone filters which is the equivalent to human cochlea we have used 20 filters, which fits the requirement of our heart sound audio files. The need to use only 20 gammatone filters for the heart sound dataset can be verified from Figure 11.2.1 (in the upper plot labeled cochleagram encoded spikes).

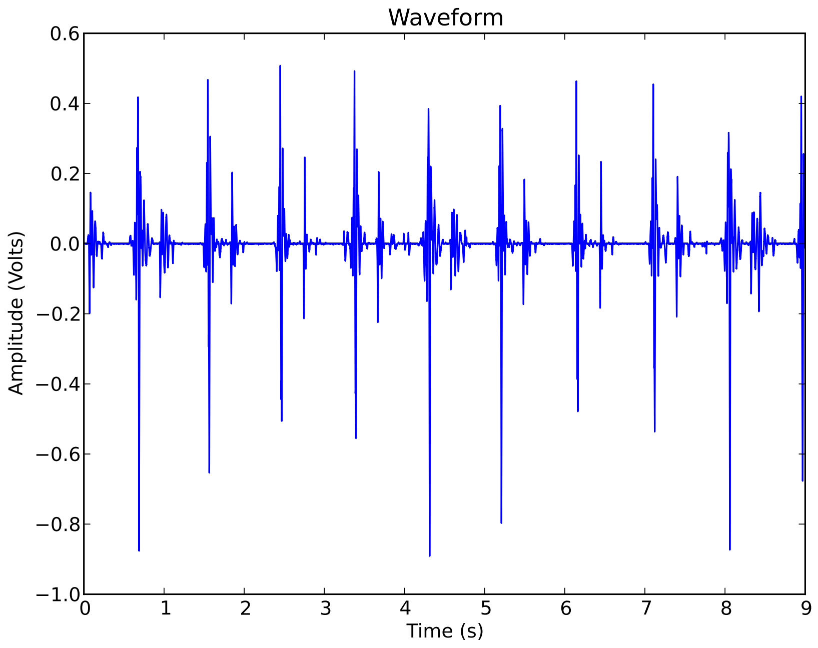


Figure 11.1.3: This figure shows the waveform of normal heart sound sample. It can be seen that the normal heart's waveform is much cleaner than the heart murmur sound. However, there is noise present from external sources which can be seen at the end of the waveform.

11.1.2 Dataset 2: Isolated Spoken Words Dataset

The 'Isolated Spoken Words' dataset contains pronunciation of the words 'alpha', 'beta', and 'charlie'. For each of the 3 classes/words, there are 20 samples, so in total there are 60 samples in the dataset. The recorded isolated words are spoken by 3 males and 2 females in American spoken English and United Kingdom spoken English. The original sound files are in MP3 file format (.mp3). These were converted to Wav file format (.wav) to make it compatible for processing with the Brian neural network simulator libraries. The encoding attributes of the Wav file are: sampling rate (standard audio CD) of 44100 Hz, bit depth of 16 Bits, Mono. The temporal length of the recorded isolated words are between 500 ms to 1000 ms.

For the cochleagram spike encoding, in our case, we have used 70 gammatone filters using the Brian's libraries (Goodman & Brette, 2010). We have used 70 gammatone filters as per the requirement of our vowels recognition audio files.

11.2 Case Study: Heart Sound Dataset

11.2.1 Experimental settings and results for deSNN

In this pilot spectro-temporal pattern recognition experiment, we use two of our deSNN classifiers, namely deSNNs and deSNNm, on 'The PASCAL Classifying Heart Sounds Challenge' dataset (Bentley et al., 2011), along with two other Spiking Neural Network (SNN) based classifiers called eSNNs and eSNNm (Dhoble et al., 2012). The parameter settings for all the four SNN based classifiers (eSNNs, eSNNm, deSNNs and deSNNm) are given in Table 11.2.1,

Table 11.2.1: Parameter settings for SNN-based classifiers (i.e. eSNNs, eSNNm, deSNNs and deSNNm)

For neurons and synapses	
Excitatory synapse time constant	2 ms
Inhibitory synapse time constant	5 ms
Neuron time constant (tau mem)	20 ms
Membrane leak	20 mV
Spike threshold (Vthr)	800 mV
Reset value	0 mV
Fixed inhibitory weight	0.20 volt
Fixed excitatory weight	0.40 volt
Thermal voltage	25 mV
Refractory period	4 ms
For learning related parameters (Fusi)	
Up/Down weight jumps (Vthm)	5 x (Vthr/8)
Calcium variable time constant (tau ca)	5 x (tau mem)
Steady-state asymptode for Calcium variable (wca)	50 mV
Stop-learning threshold 1 (stop if $Vca < thk1$)	1.7 x wca
Stop-learning threshold 2 (stop LTD if $Vca > thk2$)	2.2 x wca
Stop-learning threshold 2 (stop LTP if $Vca > thk3$)	(8 x wca) - wca
Plastic synapse (NMDA) time constant	9 ms
Plastic synapse high value (wp hi)	6 mvolt
Plastic synapse low value (wp lo)	0 mvolt
Bistability drift	0.25
Delta Weight	0.12 x wp hi
Other miscellaneous parameters / values	
Input Size	64 spike train
Simulation time	8 seconds
mod (for rank order)	0.8

along with other miscellaneous parameter values such as input size and simulation time. All the experiments are performed using 6-fold cross-validation method.

Table 11.2.2 shows the classification performance of the four deSNN classifiers (Dhoble et al., 2012) on the 'The PASCAL Classifying Heart Sounds Challenge'

Table 11.2.2: Classification accuracy for Spiking Neural Network based classifiers.

SNN Classifiers				
Accuracy	eSNNs	eSNNm	deSNNs	deSNNm
Without LSM (%)	53.85±9.4211	41.81±4.4032	48.08±3.3309	69.23±6.2807

dataset (Bentley et al., 2011). deSNNm classifier gives us the highest accuracy of 69.23%±6.2807. Compared to the other three classifiers, the classification performance of deSNNm is found to be consistent as shown in the study by Dhoble, Nuntalid, Indiveri, and Kasabov (2012). This particular experiment serves as a benchmark for experiments performed with deSNNr (consisting of deSNN with LSM reservoir) and learning deSNNr (consisting of deSNNr with STDP learning rule in the LSM reservoir).

11.2.2 Experimental settings and results for deSNNr

As explained in the previous section, deSNNr approach consists of the deSNN classifier along with the liquid state machine (LSM) reservoir. The LSM network topology is constructed using 64 neurons (N) and in a three-dimensional topology of $4 \times 4 \times 4$ neurons (Kasabov, Dhoble, et al., 2011). The neurons (N) are either excitatory (N_{ex}) or inhibitory (N_{inh}) neurons with $4:1$ ratio respectively. The simulation time for the reservoir has been set to 8000 ms, since the average time scale for the 'The PASCAL Classifying Heart Sounds Challenge' dataset is 8 seconds. These 8 seconds audio timescale is internally represented by the LSM reservoir as 8000 ms.

Table 11.2.3: The table provides the parameter settings that are used in our experimental setup for the neuron models and LSM

Parameters	*Value/s*
For Cochleagram Spike Encoder	
Min-Max Frequency Range	20 Hz - 20 kHz
Number of Channels/Neurons	20
Neuronal Model	LIF
Duration	8 sec
For Neuron	
Time Constance	10 ms
Reset Potential	0 mV
Initial Firing Threshold	10 mV
Standard Deviation of reset fluctuation	3 mV
Standard Deviation of Step-wise Firing Threshold	2 mV
Standard Deviation of Continuous Firing Threshold	1 mV
For LSM	
Simulation Time	8000 ms
Number of Neurons	64
Excitatory to Inhibitory Neuron Ratio	4:1
Input Neurons Connection Probability	0.2
Input Neurons Connection Weight	0.8 mV
Time-bin for Liquid Responses	10 ms

Table 11.2.4 shows the classification performance for deSNNr that utilizes different stochastic neuronal models in the liquid state machine reservoir. Several studies (Kasabov, Dhoble, et al., 2011; Nuntalid et al., 2011) have shown that stochastic neuronal models often results in better classification performance. This increase in classification performance due to stochastic neuronal models is also apparent in our experiment where the Noisy Reset neuron model gives us an accuracy of 74.54%±23.4674.

Table 11.2.4: Classification accuracy for Spiking Neural Network based classifiers.

SNN Classifiers (deSNNr)				
Accuracy	LIF	NR	NT	SNT
deSNNm (%)	49.82±3.2516	45.98±0.3028	50.78±9.6122	71.33±17.3491
deSNNs (%)	51.36±12.2790	74.54±23.4674	67.69±22.8449	65.56±15.7332

11.2.3 Experimental settings and results for learning deSNNr

Figure 11.2.1 shows the cochleagram encoded spikes for a heart murmur sample. This cochleagram encoded spikes are fed to the LSM reservoir with STDP learning. The lower plot in Figure 11.2.1 shows the raster plot for LSM with STDP learning. It can be seen that the spikes are highly synchronous due to the STDP learning that has been introduced into the LSM reservoir. We infer that because of this synchronization, the learning is reinforced. Therefore, the LSM reservoir with STDP learning produces better results compared to the LSM reservoir with no STDP learning.

The learning deSNNr consists of deSNN and LSM reservoir with STDP learning rule. The learning deSNNr is an extension of the epSNNr approach proposed by Kasabov, Dhoble, Nuntalid, and Mohemmed (2011). It is extended by implementing the STDP learning rule inside the LSM reservoir. The parameter settings for LSM reservoir for this learning deSNNr experiment can be seen in Table 11.2.3. The parameters for the LSM reservoir are kept the same in order to have an unbiased classification performance comparison. Table 11.2.5 shows the learning deSNNr's classification performance under different stochastic neuronal models (Nuntalid et al., 2011) such as Leaky-Integrate and

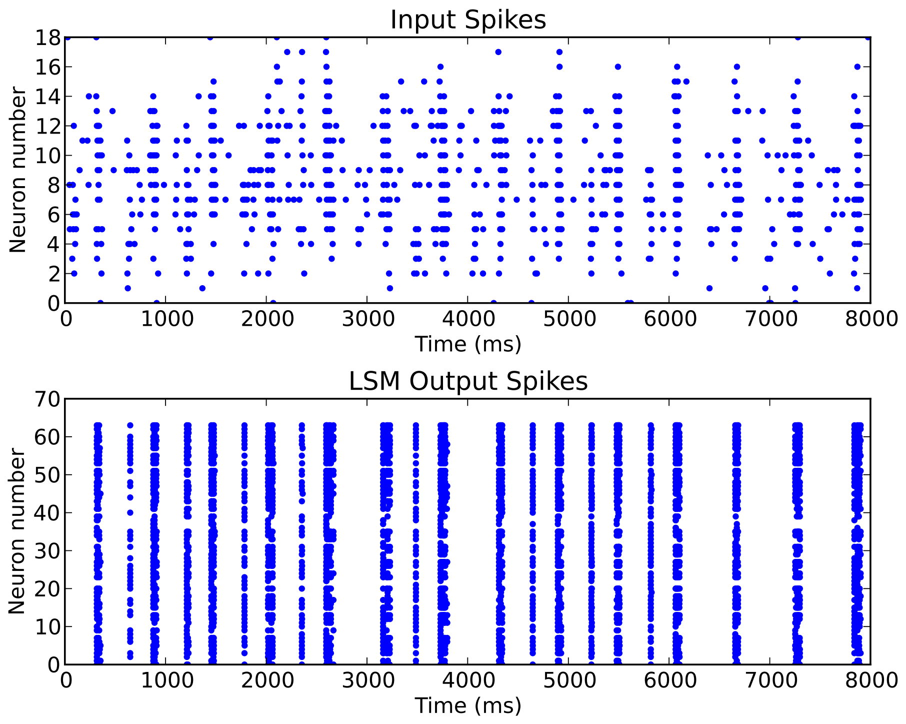


Figure 11.2.1: The above figure shows the cochleagram encoded spikes for a heart murmur sample. This cochleagram encoded spikes are fed to the LSM reservoir having STDP learning. The plot below shows the raster plot for LSM with STDP learning. It can be seen that the spikes are highly synchronous.

Fire (LIF), Noisy Reset Model (NR), Noisy Threshold Model (NT) and Step-wise Noisy Threshold Model (SNT).

All previous classification results show that the stochastic neuronal models performance was superior. The same is true for this experiment, where the Noisy Threshold neuronal model gives the highest classification accuracy of 83.33%±11.0550 using deSNNm classifier. The second highest classification accuracy of 77.00%±08.2915 is obtained with Leaky-Integrate and Fire neuronal model, also using the deSNNm classifier. It is interesting to note that

Table 11.2.5: Classification accuracy for Spiking Neural Network based classifiers using deSNNr with STDP learning in the reservoir. (Neuronal models: LIF- Leaky-Integrate and Fire, NR- Noisy Reset Model, NT- Noisy Threshold Model and SNT- Step-wise Noisy Threshold Model.)

deSNNr classifiers with STDP learning rule (learning deSNNr)				
Accuracy	LIF	NR	NT	SNT
deSNNm (%)	77.00±08.2915	53.33±9.4280	83.33±11.0550	70.00±16.7332
deSNNs (%)	60.83±16.8931	50.00±10.000	55.66±11.0550	56.00±10.1986

in the previous experiment (see Table 11.2.4), where STDP learning rule is absent in the LSM reservoir, the highest classification accuracy obtained with LIF neuronal model is 51.36%±12.2790, which is approximately 26 percent lower on comparison to the learning deSNNr's result.

11.3 Case Study: Isolated Spoken Words Dataset

Similarly to the previous experiment with heart sound dataset, here we use a dataset consisting of isolated spoken word. Apart from the parameter values given in Table 11.3.1 for deSNN, and for neuron models and LSM provided in Table 11.3.3, all the other parameter values remain the same. The experiments are performed using 6-fold cross-validation method. Since the total number of neurons used by the reservoir is 125 (which corresponds to the reservoir output size), 125 neurons were used as an input size for the deSNN learning algorithms. Also, since the temporal lengths of the recorded isolated words are between 500 ms to 1000 ms, the simulation time is set to 1000 ms (see Table 11.3.1).

Table 11.3.1: Parameter settings for deSNN. Apart from the parameters shown here, all other parameters are the same as shown in Table 11.2.1.

For neurons and synapses	
Spike threshold (Vthr)	8000 mV
Other miscellaneous parameters / values	
Input Size (eSNN)	70 spike train
Input Size (deSNN)	125 spike train
Simulation time	1000 ms

Table 11.3.2: Classification accuracy for Spiking Neural Network based classifiers.

SNN Classifiers				
Accuracy	eSNNs	eSNNm	deSNNs	deSNNm
Without LSM (%)	42.50±10.8972	37.50±14.7009	54.00±8.0000	50.00±14.1421

11.3.1 Experimental settings and results - deSNN

The classification performance of eSNN and deSNN variants for this particular dataset is given in Table 11.3.2. It can be seen that deSNNm and deSNNs approaches, are extensions of eSNN (Kasabov, Dhoble, Nuntalid, & Indiveri, 2012), perform significantly better in terms of classification task. Therefore, in the next experiments we only use the deSNN approach.

11.3.2 Experimental settings and results - deSNNr

The input neurons connection weights for the LSM reservoir vary between 1.3 mV and 1.7 mV. This is because, for some samples, a small weight (1.3 mV) does not initiate the network and otherwise, for some samples the input neurons connection weight of 1.7 mV results in over excitation of the LSM network due to chaotic dynamics (Bertschinger & Natschläger, 2004). For this reason, dependent on the sample, varying input neurons connection weight are used in the LSM reservoir (see Table 11.3.3). The input neurons connection weight for each of the samples were determined through a trial and error.

Table 11.3.3: The table provides the parameter settings used in our experimental setup for the neuron models and LSM. Apart from the shown parameters, all other parameters are the same as shown in Table 11.2.3.

Parameters	*Value/s*
For Cochleagram Spike Encoder	
Min-Max Frequency	20 Hz - 20 kHz
Number of Channels/Neurons	70
Neuronal Model	LIF
Duration	1000 ms
For LSM	
Number of Neurons	125
LSM Topology	$5 \times 5 \times 5$
Input Neurons Connection Weight	1.3 mV - 1.7 mV
Simulation Time	1000 ms

In the following experiment (see Table 11.3.4), we use the deSNN algorithm with LSM reservoir (deSNNr). Also, various stochastic neural models are used in the LSM reservoir. The classification performance corresponding to each of the stochastic neural models along with the LIF neural model for deSNN is given in Table 11.3.4. It can be seen that Noisy Reset (NR) neuronal model with deSNNm gives the highest accuracy of 65.00%±9.5742 for this particular dataset.

Table 11.3.4: Classification accuracy for Spiking Neural Network based classifiers (deSNN) with LSM reservoir. The parameter C for deSNNm has been optimized between 0 and 1 (with 0.05 step).

SNN Classifiers (deSNNr)				
Accuracy	LIF	NR	NT	SNT
deSNNm (%)	50.00±8.9442	65.00±9.5742	50.00±8.1649	58.33±10.6718
deSNNs (%)	34.00±10.1980	38.33±18.6338	25.00±12.5830	43.33±16.9967

11.3.3 Experimental settings and results for learning deSNNr

This experiment is similar to the previous experiment (using deSNNr). However, in this experiment we incorporate the standard STDP learning rule in the LSM reservoir along with the deSNN approach (hence this method is called learning deSNNr). The classification performance is given in Table 11.3.5. It can be seen that the Noisy Reset (NR) neuronal model with deSNNs gives the highest accuracy of 80.00%±20.9761 followed by NR neuronal model with deSNNm. The performance of deSNN variants is dependent on various parameters such as the spike density and temporal length of the spike train along with the dataset. Hence, with the heart sound dataset, the classification performance of deSNNm was better than the others and vice-versa, deSNNs performed better on this particular dataset. However, the difference in the classification performance between deSNNm and deSNNs is often very small (Kasabov, Dhoble, Nuntalid, & Indiveri, 2012).

As established in the previous study by Kasabov, Dhoble, Nuntalid, and Indiveri (2012), the deSNN models resulted in a better classification performance when compared with other SNN models that use either rank-order

Table 11.3.5: Classification accuracy for Spiking Neural Network based classifiers using deSNNr with STDP learning in the reservoir. (Neuronal models: LIF- Leaky-Integrate and Fire, NR- Noisy Reset Model, NT- Noisy Threshold Model and SNT- Step-wise Noisy Threshold Model.)

deSNNr classifiers with STDP learning rule (learning deSNNr)				
Accuracy	LIF	NR	NT	SNT
deSNNm (%)	70.00±11.5470	78.00±07.4833	70.00±11.5470	66.66±4.7140
deSNNs (%)	61.66±18.6338	80.00±20.9761	78.33±14.6249	66.66±11.0554

or STDP learning rule. The reason is that the deSNN makes use of both the information contained in the order of the first input spikes present in input data streams (this is crucial to consider in some tasks) and the information contained in the timing of the following spikes that is learned by the dynamic synapses as a whole spatio-temporal pattern.

The proposed learning deSNNr, which incorporates the STDP learning rule in the LSM reservoir, is shown to have far superior performance in terms of classification accuracy when compared to deSNNr approach. Previous studies (Maass & Sontag, 2000; Natschlager & Maass, 2002; Brader et al., 2007) that have utilized STDP learning have also shown that such networks can approximate a very rich class of non-linear filters and have been successfully utilized in practical applications. Hence, due to the STDP's ability to function as memory buffers and its ability to account for the precise time of spikes, deSNNr with STDP learning (i.e. learning deSNNr) has a higher classification performance on comparison to the result obtained from the reservoir without STDP learning rule (deSNNr). The method is illustrated on two different case studies using the heart sound dataset and the isolated spoken words dataset respectively for spectro-temporal pattern recognition. The cochleagram-based spike encoding approach worked well for the above spectro-temporal datasets due to the datasets simplicity; however, an alternative spike information encoding and preprocessing method is suggested for more complex spectro-temporal datasets.

For future works, various parameter settings would need to be optimized for the cochleagram-based spike encoding, the reservoir and the deSNN algorithms. Also, the sound to spike encoding with the reservoir is computationally expensive, therefore a hardware implementation would be recommended for real time computation for real-time practical applications. Since STDP learning is now implementable on hardware (Thorpe, 2012), it makes it feasible to attempt implementation of the deSNN model on such hardware for real time spectro-temporal pattern recognition tasks (Kasabov, Dhoble, Nuntalid, & Indiveri, 2012).

11.4 Summary

The classification performance of the epSNNA-s architecture is evaluated against standalone online one-pass spiking neural network based classifiers such as eSNN and deSNN. It is found that for the two presented case studies, epSNNA-s performs significantly better than eSNN and deSNN alone. Also, the hypothesis that the introduction of STDP learning into the probabilistic reservoir would result in better classification performance has been demonstrated. We conclude the thesis with the next chapter where the achievement of research objectives, the scientific contributions of this work are presented along with the future directions.

"The more you know, the more you realize you know nothing."

Socrates (469 BC 399 BC)

12 Conclusion and future directions

We conclude the thesis by presenting in this chapter a summary of the research main achievements and contributions.

12.1 Introduction

This thesis proposes novel frameworks and classification methods employing a class of evolving spiking neural networks (eSNN) called dynamic evolving spiking neural networks (deSNN) along with reservoir computing. In our study, we have shown that using the proposed frameworks results in better classification performance when compared to standalone spiking neural network classifiers such as eSNN and deSNN.

All the frameworks and methods proposed in this thesis have been evaluated on synthetic and real world problems. In order to evaluate the efficacy of the new methodology, initially a pilot experiment has been performed as a benchmark test using the synthetic video dataset. This is followed by applying the methods on real world problems relating to motion and sound such as human action recognition, heart sound recognition and isolated spoken words recognition.

This chapter concludes the thesis by summarizing the achievements of the presented research and indicates several directions for future work.

12.2 Summary of achievements

- **Developed new generic SNN methods for spatio and spectro-temporal data processing.**

  As per the research objectives we have introduced two new eSNN-based classifiers called deSNNm and deSNNs and two deSNN based hybrid algorithms called deSNNr and learning deSNNr. The characteristics and specifics of deSNNm and deSNNs are explained in Chapter 6. Also, both methods have been independently evaluated on several datasets such as synthetic video dataset, simple moving object dataset, AER-based human action recognition dataset, heart sound dataset and isolated spoken words dataset. It was found that the deSNNm classifier performed better than deSNNs. However, when the density of the spikes is high, the deSNNs performance increases significantly. Also, deSNNm and deSNNs performance on classification tasks is significantly higher than online one-pass spiking neural network (SNN) based classifiers such as eSNN and multi-layered feed-forward SNN with SDSP learning (SDSP-SNN). The results of the initial study is presented in Chapter 7.

- **Developed a generic architecture and methods for spatio-temporal pattern recognition.**

  Two new generic architectures for spatio-temporal pattern recognition have been introduced in this thesis. The construction and function of the proposed evolving probabilistic SNN reservoir (epSNNr) and evolving SNN architecture for spatio-temporal data (epSNNA-v) are explained in Chapter 4 and Chapter 8 respectively. Apart from being a generic framework for spatio-temporal pattern recognition, the spike-based pattern recognition approach also complements the artificial silicon retina, thereby allowing the architecture to directly utilize the spikes produced by the artificial silicon retina. The proposed epSNNA-v is made up of two new hybrid algorithms called deSNNr and learning deSNNr. The core of the deSNNr algorithm consists of the Liquid State Machine (LSM) with stochastic neural models and the deSNN classifiers. The learning deSNNr is the same as deSNNr, but it has unsupervised STDP learning rule in the LSM reservoir. The proposed architecture (epSNNA-v and epSNNA-s) is characterized by: one-pass propagation of SSTD during learning; utilizing STDP/SDSP and dynamic synapses in learning deSNNr that are modifiable during both learning and recall; it is an entirely spike-based spatio-temporal pattern recognition approach. Also the hypothesis put forward in Chapter 8, that introduction of STDP learning into the probabilistic reservoir will result in better classification performance has been proved.

- **Developed of a software simulator for artificial silicon retina.**

  This simulator allows converting stored video data into AER-based spikes. If required, it also allows real time acquisition of spikes from camera for real time spike processing. The simulator uses Address Event Representation approach employed in the artificial silicon retina. The characteristics and specifics of the simulator are presented in Chapter 5.

- **Developed a generic architectures for spectro-temporal pattern recognition**

  The proposed epSNNA-s architecture is introduced in Chapter 10. Some modules of the architecture are the same as epSNNA-v modules. However, they differ in their spike-encoding approach as in the epSNNA-s architecture we use basic cochleagram-based spike encoding scheme.

- **Evaluated the classification performance of the generic framework and methods on real world data.**

  Through comprehensive experimental analysis, the classification performance of the generic framework and methods are evaluated on several datasets. These include a synthetic video dataset and real world datasets.

  From the initial study using synthetic video dataset presented in chapter 4, we found that the proposed epSNNr architectures classification was significantly superior to other approaches due to the stochastic neuron models. This study demonstrated the applicability of reservoir computing approach on whole spatio-temporal data.

  In a pilot study using real word data, we evaluated the classification performance of deSNNm and deSNNs methods against other SNN methods such as SDSP-SNN and eSNN using the simple moving object dataset (presented in Chapter 7). The performance of deSNNm was found to be significantly higher than those of eSNNs, eSNNm and SDSP SNN. This was due to the presence of the SDSP mechanism and dynamic synapses in the deSNN. This study demonstrated the applicability of SNN-based classifiers (deSNN) on real word data.

  After carrying out the above mentioned study, we tested the performance of the deSNN methods on more complex real world data. The AER-based human action recognition dataset was used for evaluating the classification performance of deSNN. The classification results confirmed the previously shown performance of deSNN (presented in Chapter 9). The noise robustness of the methods was also evaluated, suggesting 50 percent noise did not significantly affect the deSNN methods classification performance. As a hardware feasibility study for deSNN, we also added 25 percent noise (to mimic the average level of thermal noise found in neuromorphic hardware) to the neural and network parameters; this again confirmed the deSNN methods robustness against noise.

deSNNm utilizes threshold and deSNNs does not utilizes threshold, deSNNs only uses connection weights, where new neuron is created in the repository for every sample and its connection weights are compared with the existing ones, where the closer match is the winner. So deSNNs requires the whole sample to be processed, therefore deSNNm is more desirable option than deSNNs because (for deSNNm) you can tune the threshold with coefficient c. This means that we do not need to propagate the entire pattern maybe around 70%, this will result in spike. But the real reason deSNNm performs better is because of its emphasis on first spikes. In AER the first spikes are more important rather than the whole pattern and deSNNm takes advantage of this since deSNNm also places more importance on initial spikes by means of ROC. After ROC, the spikes are regulating via SDSP in an unsupervised manner.

The new generic architecture for spatio-temporal pattern recognition (epSNNA-v) was also evaluated using the human action recognition dataset. The two hybrid approaches, deSNNr and learning deSNNr, born from the epSNNA-v architecture, showed improved classification performance against standalone eSNN and deSNN classifiers (study is presented in Chapter 9). The learning deSNNr approach introduces the unsupervised STDP learning mechanism in Liquid State Machine, similar to the deSNN method. Due to the presence of STDP mechanism in learning deSNNr, the classification performance improved significantly.

The third generic architecture for spectro-temporal pattern recognition (epSNNA-s) is evaluated using two case studies - heart sound dataset and isolated spoken words dataset. The learning deSNNr approach confirmed the improved classification performance shown using spatio-temporal data (study presented in Chapter 11). Also, we found that the methods and architectures are able to process the entire spatio- and spectro-temporal data having small (in milliseconds) and large (in minutes) temporal length.

It was also found that in the presence of high-density spikes, the deSNNs methods perform better than deSNNm and vice versa, when data with low-spike density is presented deSNNm performs better than deSNNs. We note that the connection weights of the neural network are not subject to any optimization. Instead, the weights are obtained through the use of an efficient one-pass learning algorithm that was developed as part of the epSNNA-v/s architecture. Therefore, through initial study on real-world problems, we have demonstrated the feasibility and applicability of the developed generic framework and methods.

12.3 Future directions

As future works, I would like to test the scalability of the system in terms of network size, number of classes and complex classes. Also, the proposed methods need to be tested on datasets having both spatio and spectro-temporal components such as the Electroencephalography (EEG) and functional Magnetic Resonance Imaging (fMRI) datasets.

In all experiments in this study, the neural and learning parameters of the applied method were manually fine-tuned using trial and error method in order to achieve satisfying classification results. The generic frameworks and methods have numerous parameters and finding an appropriate value for parameter setting becomes a challenging task. Hence, an optimization mechanism needs to be incorporated into the framework that will provide optimal parameter settings for a particular dataset.

Optimization has been identified as one of the major issues for the future development of deSNN models and architecture. One way is to combine the local learning of synaptic weights with global optimization of SNN parameters. Three optimization approaches are suggested, namely evolutionary optimization, neurogenetic optimization and reservoir optimization. These approaches are discussed in some details below.

12.3.1 Optimization

Evolutionary optimization

The inspiration for evolutionary optimization comes from natural evolution. One of the natural evolution principles that has been borrowed by the evolutionary optimization approaches such as genetic algorithms is the survival of the fittest. Evolutionary computing techniques do not require the use of domain knowledge, so they can be incorporated depending on the requirements. Some of the commonly used evolutionary computation methods are: genetic algorithms, particle swarm optimization, quantum inspired evolutionary computation methods (Schliebs, Kasabov, & Defoin-Platel, 2010; Schliebs et al., 2009; Kasabov & Hu, 2010; Hamed et al., 2010). These methods can be used to explore the performance of many deSNN in a population, each having different parameter settings until a close to optimum performing model can be found.

Neurogenetic optimization

Gene regulatory network (GRN) models from the neurogenetics domains can also be considered as one of the optimization methods for future applications. In biology, the dynamics of genes often have an influence on the neural network. Inspired by this gene interaction, many neurogenetic SNN models have been developed (Kasabov, 2012; Kasabov, Benuskova, & Wysoski, 2004; Kasabov, Schliebs, & Kojima, 2011; Beňušková & Kasabov, 2007). These models function at two levels: at GRN level of slow changes of the gene parameter values and at SNN level of fast information processing that is affected by the gene parameter changes. Genes control SNN parameters, but how are gene values optimized? Nature has been continuously optimizing genes for millions of years now through evolution. Applying evolutionary algorithms to optimize genes in GRN that control SNN parameters for a specific problem represented as spatio- or spectro-temporal data can be investigated as future works.

Reservoir optimization

Reservoir such as Liquid State Machine (LSM) has been the core component in many of our methods and architecture. In reservoir computing, there are various other approaches such as Echo State Networks that are incorporated in the architecture. Also, the reservoirs separability can be optimized using evolutionary computing methods (Hourdakis & Trahanias, 2011). Furthermore, the model capacity can be optimized by means of regularization or by changing the size of the reservoir. Regularization is usually carried out by introducing noise during training (Jaeger & Haas, 2004) or by means of ridge regression (Ilies et al., 2007). Studies have shown that the noise insertion approach is computationally inexpensive and is more suitable than ridge regression approach in terms of the dynamical stability of the trained system (Jaeger et al., 2007).

12.3.2 Hardware Implementation

The spiking neural networks (SNN) based architectures and methods proposed in this thesis have been designed while taking into consideration the computational constraints present in neuromorphic hardware. Implementing deSNN on hardware SNN chips would enable the development of some practical engineering applications. The feasibility of implementing the deSNN model on some particular SNN chips is discussed in the following (Kasabov, Dhoble, Nuntalid, & Indiveri, 2012).

The SDSP learning rule applied on the LIF model of a neuron is implemented in the Very Large Scale Integration (VLSI) SNN silicon chip (Indiveri, Chicca, & Douglas, 2009) making it possible for a deSNN model to be implemented on this chip. The silicon synapses of the chip comprise bi-stability circuits for driving a synaptic weight to one of two possible analogue values (either potentiated or depressed). These circuits drive the synaptic-weight voltage with a current that is superimposed on that generated by the STDP and which can be either positive or negative. If, on short time scales, the synaptic weight is increased above a set threshold by the network activity via the STDP learning mechanism, the bi-stability circuits generate a constant weak positive current. In the absence of activity (and hence learning) this current will drive the weight toward its potentiated state. If the STDP decreases the synaptic weight below the threshold, the bi-stability circuits will generate a negative current that, in the absence of spiking activity, will actively drive the weight toward the analogue value, encoding its depressed state. The chip allows for different types of dynamic synapses to be implemented, including the Tsodyks model (Tsodyks, Skaggs, Sejnowski, & McNaughton, 1996).

Another SNN chip that implements LIF model of a neuron is the VLSI SRAM SNN chip (Moradi & Indiveri, 2011). It is characterized by the following: 32x32 SRAM matrix of weights, each 5 bits (values between 0 and 31); 32 neurons of the adaptive, exponential IF model of a neuron; each neuron has 2 excitatory and 2 inhibitory inputs to which any of the 32 input dendrites (rows of weights) can be connected; AER for input data, for changing the connection weights and for output data streams; since it does not have any learning rule hardware implemented, it allows to experiment with different supervised and unsupervised learning rules; learning (changing of the synaptic weights) is calculated outside the chip (in a computer, connected to the chip) in an asynchronous manner (only synaptic weights that need to change at the current time moment are changed (calculated) and then loaded into the SRAM) applying suitable learning rule and parameter settings. The fact that modifying connection weights is done asynchronously outside the chip and then the weights are loaded in the SRAM allows for the deSNN learning algorithm to be implemented on this chip. After an input is entered in the SRAM (as AER), the output from the neurons is produced and then used (outside the chip) to change connection weights according to the deSNN learning rules. The new values of the weights are entered into the SRAM also asynchronously (Kasabov, Dhoble, Nuntalid, & Indiveri, 2012). deSNN is also implementable on the digital IBM SNN chip. Despite the fast, one-pass learning in the deSNN models, in terms of large scale modeling of millions and billions of neurons using the SpiNNaker SNN supercomputer system (Jin et al., 2010) for simulation purposes would be appropriate, especially at the level of parameter optimization. Potentially, the deSNN can be used to implement neuromorphic architectures for complex engineering applications.

References

Abbott, L., & Nelson, S. (2000). Synaptic plasticity: taming the beast. *Nature neuroscience*, *3*, 1178–1183.

Abdollahi, M., & Liu, S. (2011). Speaker-independent isolated digit recognition using an AER silicon cochlea. In *Biomedical circuits and systems conference (biocas), 2011 ieee* (pp. 269–272).

Abdollahi, M., Valavi, E., & Noubari, H. (2009). Voice-based gender identification via multiresolution frame classification of spectro-temporal maps. In *IJCNN 2009. International Joint Conference on Neural Networks, 2009.* (pp. 1–4).

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1987). A learning algorithm for boltzmann machines. , 522–533.

Allen, J. (2009). *The lives of the brain: human evolution and the organ of mind*. Belknap Press.

Amari, S., & Kasabov, N. (1998). *Brain-like Computing and Intelligent Information Systems* (1st ed.). Singapore: Springer-Verlag.

Andringa, T., Niessen, M., & Nillesen, M. (2004). *Continuity Preserving Signal Processing.* Retrieved from `{http://www.ai.rug.nl/acg/cpsp/index.html}`

Arbib, M. (2003). *The Handbook of Brain Theory and Neural Networks* (2nd ed.). Singapore: MIT Press.

Arel, I., Rose, D., & Karnowski, T. (2010). Deep Machine Learning: A New Frontier in Artificial Intelligence Research [Research Frontier]. *IEEE Computational Intelligence Magazine*, *5*(4), 13–18.

Atiya, A., & Parlos, A. (2000). New results on recurrent network training: Unifying the algorithms and accelerating convergence. *Neural Networks, IEEE Transactions on*, *11*(3), 697–709.

Azevedo, F. A., Carvalho, L. R., Grinberg, L. T., Farfel, J. M. M., Ferretti, R. E., Leite, R. E., … Herculano-Houzel, S. (2009, April). Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *The Journal of comparative neurology*, *513*(5), 532–541. doi: 10.1002/cne.21974

Bentley, P., Nordehn, G., Coimbra, M., & Mannor, S. (2011). *The PASCAL Classifying Heart Sounds Challenge 2011 (CHSC2011) Results.* Retrieved from `http://www.peterjbentley.com/heartchallenge/index.html`

Beňušková, L., & Kasabov, N. (2007). *Computational neurogenetic modeling.* Springer Verlag.

Bertschinger, N., & Natschläger, T. (2004). Real-time computation at the edge of chaos in recurrent neural networks. *Neural Computation*, *16*(7), 1413–1436.

Boahen, K. (2007, June). The brain and the computer. In *Device research conference, 2007 65th annual* (p. 235-235).

Bohte, S., & Kok, J. (2005). Applications of spiking neural networks. *Information Processing Letters*, *95*(6), 519–520.

Bower, J., Beeman, D., & Hucka, M. (2002). The genesis simulation system. *The Handbook of Brain Theory and Neural Networks*, 475–478.

Brader, J., Senn, W., & Fusi, S. (2007). Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural computation*, *19*(11), 2881–2912.

Bradski, G. (2000). The OpenCV library. *Dr. Dobbs Journal: Software Tools for the Professional Programmer*, *25*(11), 120–124.

Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J., ... others (2007). Simulation of networks of spiking neurons: A review of tools and strategies. *Journal of Computational Neuroscience*, *23*(3), 349–398.

Brillinger, D. R. (1992). Nerve cell spike train data analysis: a progression of technique. *Journal of the American Statistical Association*, *87*(418), 260–271.

Bryson, A. E., & Ho, Y. C. (1975). *Applied Optimal Control: Optimization, Estimation, and Control.* Waltham, MA, USA: Blaisdell Publishing Company.

Buonomano, D., & Maass, W. (2009). State-dependent computations: spatiotemporal processing in cortical networks. *Nature Reviews Neuroscience*, *10*(2), 113–125.

Burgsteiner, H., Kröll, M., Leopold, A., & Steinbauer, G. (2005). Movement prediction from real-world images using a liquid state machine. *Innovations in Applied Artificial Intelligence*, 92–99.

Chan, V., Liu, S., & van Schaik, A. (2007). AER EAR: A matched silicon cochlea pair with address event representation interface. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *54*(1), 48–59.

Corning, W., & Balaban, M. (1968). *The mind; biological approaches to its functions*. Interscience Publishers.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*(1), 21–27.

De Garis, H., Korkin, M., Gers, F., Nawa, E., & Hough, M. (2000). Building an artificial brain using an FPGA based CAM-Brain Machine. *Applied Mathematics and Computation*, *111*(2), 163–192.

De Garis, H., Nawa, N., Hough, M., & Korkin, M. (1999). Evolving an optimal de/convolution function for the neural net modules of ATR's artificial brain project. In *IJCNN'99. International Joint Conference on Neural Networks, 1999* (Vol. 1, pp. 438–443).

Delbruck, T. (2007). *jAER open source project.* Retrieved from `{http://jaer.wiki.sourceforge.net}`

Delorme, A., Perrinet, L., & Thorpe, S. J. (2001). Networks of integrate-and-fire neurons using rank order coding b: spike timing dependent plasticity and emergence of orientation selectivity. *Neurocomputing*, *38*, 539–545.

Delorme, A., & Thorpe, S. (2003). SpikeNET: an event-driven simulation package for modelling large networks of spiking neurons. *Network: Computation in Neural Systems*, *14*(4), 613–627.

Dhoble, K., Nuntalid, N., Indiveri, G., & Kasabov, N. (2012). Online spatio-temporal pattern recognition with evolving spiking neural networks utilising address event representation, rank order, and temporal spike learning. In *The 2012 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–7).

Dominey, P. (1995). Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. *Biological Cybernetics*, *73*(3), 265–274.

Dominey, P. (2005). From sensorimotor sequence to grammatical construction:

Evidence from simulation and neurophysiology. *Adaptive Behavior*, *13*(4), 347–361.

Dominey, P., Hoen, M., Blanc, J., & Lelekov-Boissard, T. (2003). Neurological basis of language and sequential cognition: evidence from simulation, aphasia, and erp studies. *Brain and language*, *86*(2), 207–225.

Dominey, P., Hoen, M., & Inui, T. (2006). A neurolinguistic model of grammatical construction processing. *Journal of Cognitive Neuroscience*, *18*(12), 2088–2107.

Dominey, P., & Ramus, F. (2000). Neural network processing of natural language: I. Sensitivity to serial, temporal and abstract structure of language in the infant. *Language and Cognitive Processes*, *15*(1), 87–127.

Elman, J. L. (1991). Distributed Representations, Simple Recurrent Networks, And Grammatical Structure. *Machine Learning*, *7*(2-3), 195–225. doi: http://dx.doi.org/10.1007/BF00114844

Fusi, S., Annunziato, M., Badoni, D., Salamon, A., & Amit, D. (2000). Spike-driven synaptic plasticity: theory, simulation, vlsi implementation. *Neural Computation*, *12*(10), 2227–2258.

Gao, B., Woo, W., & Dlay, S. (2012). Unsupervised Single-Channel Separation of Nonstationary Signals Using Gammatone Filterbank and Itakura–Saito Nonnegative Matrix Two-Dimensional Factorizations.

Gerlach, J. (1858). *Microscopische Studien aus dem Gebiet der menschlichen Morphologie* (2nd ed.). E Enke, Erlangen.

Gerstner, W. (1995, Jan). Time structure of the activity in neural network models. *Phys. Rev. E*, *51*(1), 738–758. doi: 10.1103/PhysRevE.51.738

Gerstner, W., & Kistler, W. (2002b). *Spiking neuron models: Single neurons, populations, plasticity.* Cambridge Univ Pr.

Gerstner, W., & Kistler, W. M. (2002a). *Spiking neuron models.* Cambridge University Press.

Glackin, C., Maguire, L., McDaid, L., & Wade, J. (2011). Lateral inhibitory networks: Synchrony, edge enhancement, and noise reduction. In *The 2011 International Joint Conference on Neural Networks (IJCNN)* (pp. 1003–1009).

Goodman, D., & Brette, R. (2009). The Brian simulator. *Frontiers in*

Neuroscience, *3*(2), 192.

Goodman, D., & Brette, R. (2010). Spike-timing-based computation in sound localization. *PLoS computational biology*, *6*(11), e1000993.

Grzyb, B., Chinellato, E., Wojcik, G., & Kaminski, W. (2009). Facial expression recognition based on liquid state machines built of alternative neuron models. In *IJCNN 2009: International Joint Conference on Neural Networks.* (pp. 1011–1017).

Hamed, H. N. A., Kasabov, N., & Shamsuddin, S. (2010). Probabilistic evolving spiking neural network optimization using dynamic quantum-inspired particle swarm optimization. *Australian Journal of Intelligent Information Processing Systems*, *11*(1).

Hamed, H. N. A., Kasabov, N., Shamsuddin, S. M., Widiputra, H., & Dhoble, K. (2011). An extended evolving spiking neural network model for spatio-temporal pattern classification. In *The 2011 International Joint Conference on Neural Networks (IJCNN)* (pp. 2653–2656).

Harmon, L. D. (1962). Neural analogs. In *AIEE-IRE '62 (Spring): Proceedings of the May 1-3, 1962, spring joint computer conference* (pp. 153–158). New York, NY, USA: ACM. doi: http://doi.acm.org/10.1145/1460833.1460852

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation.* New York: Macmillan.

Hebb, D. O. (1949). *The Organization of Behavior: A Neuropsychological Theory* (New edition ed.). New York: Wiley.

Hill, A. (1936). Excitation and accommodation in nerve. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, *119*(814), 305–355.

Hines, M. (1994). The NEURON simulation program. *Kluwer International Series in Engineering and Computer Science*, 147–147.

Hinton, G. E., & Sejnowski, T. J. (1983a). Analyzing cooperative computation. In *5th annual congress of the cognitive science society.* Rochester, NY.

Hinton, G. E., & Sejnowski, T. J. (1983b). Optimal perceptual inference. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 448–453). Washington DC: IEEE Computer Society.

Hinton, G. E., & Sejnowski, T. J. (1986). Learning and relearning in boltzmann machines. , *1*, 282–317.

Hodgkin, A. L., & Huxley, A. F. (1952a). Currents carried by sodium and potassium ions through the membrane of the giant axon of Loligo. *Journal of Physiology*, *116*(4), 449-472.

Hodgkin, A. L., & Huxley, A. F. (1952b). The dual effect of membrane potential on sodium conductance in the giant axon of Loligo. *Journal of Physiology*, *116*(4), 497-506.

Hodgkin, A. L., & Huxley, A. F. (1952c). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, *117*, 500–544.

Hopfield, J. J. (1982, April). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, *79*(8), 2554–2558.

Hopfield, J. J. (1988). Neurons with graded response have collective computational properties like those of two-state neurons. , 82–86.

Hourdakis, E., & Trahanias, P. (2011). Improving the classification performance of liquid state machines based on the separation property. *Engineering Applications of Neural Networks*, 52–62.

Huguenard, J. R. (2000, August 15). Reliability of axonal propagation: the spike doesn't stop here. *Proceedings of the National Academy of Sciences of the United States of America*, *97*(17), 9349–9350. Retrieved from `http://view.ncbi.nlm.nih.gov/pubmed/10944204`

Ilies, I., Jaeger, H., Kosuchinas, O., Rincon, M., Sakenas, V., & Vaskevicius, N. (2007). Stepping forward through echoes of the past: forecasting with echo state networks. *Avaible: http://www. neural-forecastingcompetition. com/downloads/methods/27-NN3_Herbert_Jaeger_report. pdf*.

Indiveri, G. (2008). Neuromorphic VLSI models of selective attention: from single chip vision sensors to multi-chip systems. *Sensors*, *8*(9), 5352–5375.

Indiveri, G., Chicca, E., & Douglas, R. (2009). Artificial cognitive systems: From VLSI networks of spiking neurons to neuromorphic cognition. *Cognitive Computation*, *1*(2), 119–127.

Indiveri, G., Linares-Barranco, B., Hamilton, T., Van Schaik, A., Etienne-Cummings, R., Delbruck, T., ... others (2011). Neuromorphic silicon neuron circuits. *Frontiers in neuroscience*, *5*.

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE transactions on neural networks*, *14*(6), 1569–1572.

Izhikevich, E. M. (2004, Sept.). Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, *15*(5), 1063-1070. doi: 10.1109/TNN.2004.832719

Izhikevich, E. M. (2006). Polychronization: Computation with Spikes. *Neural Comput.*, *18*(2), 245–282. doi: http://dx.doi.org/10.1162/089976606775093882

Izhikevich, E. M. (2007). *Dynamical systems in neuroscience : the geometry of excitability and bursting.* MIT Press.

Izhikevich, E. M., & Edelman, G. M. (2008, March). Large-scale model of mammalian thalamocortical systems. *Proceedings of the National Academy of Sciences*, *105*(9), 3593–3598. doi: 10.1073/pnas.0712231105

Jaeger, H. (2001a). *Short term memory in echo state networks.* GMD-Forschungszentrum Informationstechnik.

Jaeger, H. (2001b). The echo state approach to analysing and training recurrent neural networks-with an erratum note. *Tecnical report GMD report*, *148*.

Jaeger, H. (2002). *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach.* GMD-Forschungszentrum Informationstechnik.

Jaeger, H. (2007). Echo state network. *Scholarpedia*, *2*(9), 2330.

Jaeger, H., & Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, *304*(5667), 78–80.

Jaeger, H., Lukoševičius, M., Popovici, D., & Siewert, U. (2007). Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, *20*(3), 335–352.

Jaeger, H., et al. (2003). Adaptive nonlinear system identification with echo state networks. *networks*, *8*, 9.

Jin, X., Luján, M., Plana, L., Davies, S., Temple, S., & Furber, S. (2010).

Modeling spiking neural networks on SpiNNaker. *Computing in Science & Engineering*, *12*(5), 91–97.

Jolivet, R., Lewis, T. J., & Gerstner, W. (2004). Generalized integrate-and-fire models of neuronal activity approximate spike trains of a detailed model to a high degree of accuracy. *Journal of Neurophysiology*, *92*(2), 959–976.

Ju, H., Xu, J., & VanDongen, A. (2010). Classification of musical styles using liquid state machines. In *The 2010 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–7).

Kandel, E., Schwartz, J., & SzJessell, T. (1991). Principles of neural science. *London: Appleton and Lange*.

Kasabov, N. (2001). Evolving fuzzy neural networks for supervised / unsupervised online knowledge-based learning. *Part B: Cybernetics, IEEE Transactions on Systems, Man, and Cybernetics*, *31*(6), 902–918.

Kasabov, N. (2002). Evolving connectionist systems for adaptive learning and knowledge discovery: methods, tools, applications. In *Proceedings of first international IEEE symposium on intelligent systems* (Vol. 1, pp. 24–28).

Kasabov, N. (2003). *Evolving Connectionist Systems: Methods and Applications in Bioinformatics, Brain Study and Intelligent Machines*. Springer. Retrieved from `http://books.google.co.nz/books?id=MyBND1AbGokC`

Kasabov, N. (2007). *Evolving connectionist systems: the knowledge engineering approach*. Springer-Verlag, London.

Kasabov, N. (2009). Integrative probabilistic spiking neural networks utilizing quantum inspired evolutionary algorithm: A Computational framework. *ICONIP '08: 15th International Conference on Neural Information Processing of the Asia-Pacific Neural Network Assembly*, *5506*, 3–13.

Kasabov, N. (2010a). Integrative Probabilistic Evolving Spiking Neural Networks Utilising Quantum Inspired Evolutionary Algorithm: A Computational Framework. In *Advances in Machine Learning II* (Vol. 263, p. 415-425). Springer Berlin / Heidelberg.

Kasabov, N. (2010b). To spike or not to spike: A probabilistic spiking neuron model. *Neural Networks*, *23*(1), 16–19.

Kasabov, N. (2012). NeuCube EvoSpike Architecture for Spatio-temporal Modelling and Pattern Recognition of Brain Signals. *Artificial Neural*

Networks in Pattern Recognition, 225–243.

Kasabov, N., Benuskova, L., & Wysoski, S. (2004). Computational neurogenetic modeling: integration of spiking neural networks, gene networks, and signal processing techniques. In *IEEE International Workshop on Biomedical Circuits and Systems* (pp. S2–7).

Kasabov, N., Dhoble, K., Nuntalid, N., & Indiveri, G. (2012). Dynamic evolving spiking neural networks for on-line spatio- and spectro-temporal pattern recognition. *Neural Networks*. Retrieved from `http://www.sciencedirect.com/science/article/pii/S0893608012003139` doi: 10.1016/j.neunet.2012.11.014

Kasabov, N., Dhoble, K., Nuntalid, N., & Mohemmed, A. (2011). Evolving probabilistic spiking neural networks for spatio-temporal pattern recognition: A preliminary study on moving object recognition. In *Neural information processing* (pp. 230–239).

Kasabov, N., Dhoble, K., Nuntalid, N., Mohhemed, A., Schliebs, S., Indivery, G., ... Sheikh, S. (2012). *EvoSpike: Evolving Probabilistic Spiking Neural Networks for Spatio-Temporal Pattern Recognition.* Retrieved from `{http://ncs.ethz.ch/projects/evospike}`

Kasabov, N., & Hu, Y. (2010). Integrated optimisation method for personalised modelling and case studies for medical decision support. *International Journal of Functional Informatics and Personalised Medicine*, *3*(3), 236–256.

Kasabov, N., et al. (1998). ECOS: Evolving connectionist systems and the ECO learning paradigm. In *International Conference on Neural Information Processing, Kitakyushu, Japan* (pp. 1222–1235).

Kasabov, N., Schliebs, R., & Kojima, H. (2011). Probabilistic Computational Neurogenetic Modelling: From Cognitive Systems to Alzheimer's Disease. *IEEE Transactions on Autonomous Mental Development*, *3*(4), 1–12.

Kasabov, N., & Song, Q. (2002). DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE Transactions on Fuzzy Systems*, *10*(2), 144–154.

Keat, J., Reinagel, P., Reid, R. C., & Meister, M. (2001). Predicting every spike: a model for the responses of visual neurons. *Neuron*, *30*(3), 803–817.

Kempter, R., Gerstner, W., & Hemmen, J. (2001). Intrinsic stabilization of output rates by spike-based Hebbian learning. *Neural Computation*, *13*(12), 2709–2741.

Kempter, R., Gerstner, W., & Van Hemmen, J. (1999). Hebbian learning and spiking neurons. *Physical Review E*, *59*(4), 4498.

Kistler, W. M., Gerstner, W., & van Hemmen, J. L. (1997). Reduction of the Hodgkin-Huxley Equations to a Single-Variable Threshold Model. *Neural Computation*, *9*(5), 1015–1045.

Kohonen, T. (1989). *Self-organization and associative memory.* New York, NY, USA: Springer-Verlag Berlin and New York.

Kohonen, T., & Honkela, T. (2007). Kohonen network. *Scholarpedia*, *2*(1), 1568.

Lapicque, L. (1907). Recherches quantitatives sur lexcitation électrique des nerfs traitée comme une polarisation. *J. Physiol. Pathol. Gen*, *9*(1), 620–635.

LeCun, Y. (1985). Une procédure d'apprentissage pour réseau à seuil asymétrique. *Proceedings of Cognitiva 85*, 599–604.

LeCun, Y. (1986). Learning processes in an asymmetric threshold network. In *Disordered systems and biological organization* (p. 233-240). Berlin: Springer.

Legenstein, R., Naeger, C., & Maass, W. (2005). What can a neuron learn with spike-timing-dependent plasticity? *Neural Computation*, *17*(11), 2337–2382.

Lichtsteiner, P., & Delbrück, T. (2005). A 64x64 AER logarithmic temporal derivative silicon retina. *Research in Microelectronics and Electronics*, *2*, 202–205.

Lukoševičius, M., & Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, *3*(3), 127–149.

Maass, W., & Bishop, C. M. (Eds.). (1999). *Pulsed neural networks.* Cambridge, MA, USA: MIT Press.

Maass, W., Joshi, P., & Sontag, E. (2006). Principles of real-time computing with feedback applied to cortical microcircuit models.

Maass, W., & Markram, H. (2002). Synapses as dynamic memory buffers. *Neural Networks*, *15*(2), 155–161.

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, *14*(11), 2531–2560.

Maass, W., Natschlager, T., & Markram, H. (2003). A model for real-time computation in generic neural microcircuits. *Advances in neural information processing systems*, 229–236.

Maass, W., Natschläger, T., & Markram, H. (2004). Computational models for generic cortical microcircuits. *Computational neuroscience: A comprehensive approach*, *18*, 575.

Maass, W., & Sontag, E. (2000). Neural systems as nonlinear filters. *Neural Computation*, *12*(8), 1743-1772.

Maass, W., & Zador, A. (1999). Computing and learning with dynamic synapses. *Pulsed neural networks*, 157–178.

Masters, T. (1993). *Practical neural network recipes in c++*. San Diego, CA, USA: Academic Press Professional, Inc.

McCarthy, J. (1963). *Programs with common sense*. Defense Technical Information Center.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysic*, *5*, 115–133.

Minsky, M. (1954). *Theory of Neural-Analog Reinforcement Systems and its Application to the Brain-Model Problem.* Doctoral dissertation, Princeton University, University Microfilms, Ann Arbor.

Minsky, M. (1961). Steps toward artificial intelligence. In *Computers and Thought* (pp. 406–450). McGraw-Hill.

Mohemmed, A., Schlibes, S., Matsuda, S., & Kasabov, N. (2012). Span: Spike pattern association neuron for learning spatio-temporal spike patterns. *International Journal of Neural Systems*, *22*(04).

Moradi, S., & Indiveri, G. (2011). A VLSI network of spiking neurons with an asynchronous static random access memory. In *IEEE Biomedical Circuits and Systems Conference (BioCAS)* (pp. 277–280).

Morrison, A., Diesmann, M., & Gerstner, W. (2008). Phenomenological models of synaptic plasticity based on spike timing. *Biological Cybernetics*, *98*(6), 459–478.

Muthusamy, Y., Cole, R., & Slaney, M. (1990). Speaker-independent vowel recognition: Spectrograms versus cochleagrams. In *ICASSP-90., 1990 International Conference on Acoustics, Speech, and Signal Processing* (pp. 533–536).

Narayanan, A., & Wang, D. (2012). A CASA-Based System for Long-Term SNR Estimation. *IEEE Transactions on Audio, Speech, and Language Processing*, *20*(9), 2518–2527.

Natschlager, T., & Maass, W. (2002). Spiking neurons and the induction of finite state machines* 1. *Theoretical computer science*, *287*(1), 251–265.

Natschläger, T., Maass, W., & Markram, H. (2002). The liquid computer: A novel strategy for real-time computing on time series. *Special issue on Foundations of Information Processing of TELEMATIK*, *8*(1), 39–43.

Natschläger, T., Markram, H., & Maass, W. (2003). Computer models and analysis tools for neural microcircuits. *Neuroscience databases. A practical guide*, *9*, 123–138.

Nissl, E. (1894). Ober die soganannten Granula der Nervenzellen. *Neurol*, *13*, 676–685.

Norton, D., & Ventura, D. (2009). Improving the separability of a reservoir facilitates learning transfer.

Nuntalid, N., Dhoble, K., & Kasabov, N. (2011). EEG classification with BSA spike encoding algorithm and evolving probabilistic spiking neural network. In *Neural information processing* (pp. 451–460).

Panchev, C., & Wermter, S. (2004). Spike-timing-dependent synaptic plasticity: from single spikes to spike trains. *Neurocomputing*, *58-60*, 365 - 371. (Computational Neuroscience: Trends in Research 2004) doi: DOI:10.1016/j.neucom.2004.01.068

Pape, L., de Gruijl, J., & Wiering, M. (2008). Democratic liquid state machines for music recognition. *Speech, Audio, Image and Biomedical Signal Processing using Neural Networks*, 191–215.

Parker, D. B. (1985). *Learning-logic* (Tech. Rep. Nos. TR–47).

Poppe, R. (2010). A survey on vision-based human action recognition. *Image* *and Vision Computing*, *28*(6), 976–990.

Powell, M. J. D. (1987). Radial basis functions for multivariable interpolation: a review. , 143–167.

Purves, D., Augustine, G. J., Fitzpatrick, D., Hall, W. C., LaMantia, A., McNamara, J. O., & White, L. E. (2008). *Neuroscience* (4th ed.). Sinauer Associates.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (Vol. 1). Cambridge, MA: MIT Press.

Russell, S. J., & Norvig, P. (1995). *Artificial intelligence: a modern approach.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Schiller, U., & Steil, J. (2005). Analyzing the weight dynamics of recurrent learning algorithms. *Neurocomputing*, *63*, 5–23.

Schliebs, S., Defoin-Platel, M., & Kasabov, N. (2009). Integrated feature and parameter optimization for an evolving spiking neural network: Exploring heterogeneous probabilistic models. *Neural Networks*, *22*(5-6), 623–632.

Schliebs, S., Hamed, H. N. A., & Kasabov, N. (2011). Reservoir-based evolving spiking neural network for spatio-temporal pattern recognition. In *Neural Information Processing* (pp. 160–168).

Schliebs, S., Kasabov, N., & Defoin-Platel, M. (2010). On the Probabilistic Optimization of Spiking Neural Networks. *International Journal of Neural Systems*, *20*(6), 481–500.

Schliebs, S., Nuntalid, N., & Kasabov, N. (2010). Towards spatio-temporal pattern recognition using evolving spiking neural networks. In *Proceedings of the 17th international conference on Neural information processing: theory and algorithms-Volume Part I* (pp. 163–170).

Schrauwen, B., & Van Campenhout, J. (2003). BSA, a fast and accurate spike train encoding scheme. In *Proceedings of the International Joint Conference on Neural Networks, 2003* (Vol. 4, pp. 2825–2830).

Schuldt, C., Laptev, I., & Caputo, B. (2004). Recognizing human actions: A local SVM approach. In *ICPR 2004: Proceedings of the 17th International Conference on Pattern Recognition.* (Vol. 3, pp. 32–36).

Soltic, S., & Kasabov, N. (2010a). A Biologically Inspired Evolving Spiking Neural Model with Rank-Order Population Coding and a Taste Recognition System Case Study. *System and Circuit Design for Biologically-Inspired Intelligent Learning*, 136.

Soltic, S., & Kasabov, N. (2010b). Knowledge extraction from evolving spiking neural networks with a rank order population coding. *International Journal of Neural Systems*, *20*(6), 437–445.

Song, Q., & Kasabov, N. (2005). NFI: A neuro-fuzzy inference method for transductive reasoning. *IEEE Transactions on Fuzzy Systems*, *13*(6), 799–808.

Song, S., Miller, K., Abbott, L., et al. (2000). Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, *3*, 919–926.

Steil, J. (2004). Backpropagation-decorrelation: Online recurrent learning with o (n) complexity. In *Neural networks, 2004. proceedings. 2004 ieee international joint conference on* (Vol. 2, pp. 843–848).

Stein, R. B. (1967). Some models of neuronal variability. *Biophysical journal*, *7*(1), 37–68.

Sulehria, H. K., & Zhang, Y. (2007). Hopfield neural networks: a survey. In *AIKED'07: Proceedings of the 6th Conference on 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases* (pp. 125–130). Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS).

Swanson, L. (2000). A history of neuroanatomical mapping. In *Brain Mapping: The Applications* (pp. 77–109). San Diego, USA: Academic Press.

Thorpe, S. (2012). Spike-based image processing: Can we reproduce biological vision in hardware? *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7583 LNCS*(PART 1), 516-521.

Thorpe, S., & Gautrais, J. (1998). Rank order coding. In *CNS '97: Proceedings* *of the sixth annual conference on Computational neuroscience : trends in research* (pp. 113–118). New York, NY, USA: Plenum Press.

Thorpe, S., Guyonneau, R., Guilbaud, N., Allegraud, J., & VanRullen, R. (2004). SpikeNet: Real-time visual processing with one spike per neuron. *Neurocomputing*, *58*, 857–864.

Tou, J., & Gonzalez, R. (1974). *Pattern recognition principles*. Reading, MA: Addison-Wesley.

Trentin, E., & Gori, M. (2001). A survey of hybrid ANN/HMM models for automatic speech recognition. *Neurocomputing*, *37*(1-4), 91–126.

Tsodyks, M. V., Skaggs, W. E., Sejnowski, T. J., & McNaughton, B. L. (1996). Population dynamics and theta rhythm phase precession of hippocampal place cell firing: a spiking neuron model. *Hippocampus*, *6*(3), 271–280.

Turaga, P., Chellappa, R., Subrahmanian, V., & Udrea, O. (2008). Machine recognition of human activities: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, *18*(11), 1473–1488.

Uysal, I., Sathyendra, H., & Harris, J. (2007). Spike-based feature extraction for noise robust speech recognition using phase synchrony coding. In *Circuits and systems, 2007. iscas 2007. ieee international symposium on* (pp. 1529–1532).

Van Schaik, A., & Liu, S. (2005). AER EAR: A matched silicon cochlea pair with address event representation interface. In *ISCAS-IEEE International Symposium on Circuits and Systems* (pp. 4213–4216).

Verstraeten, D., Schrauwen, B., D'Haene, M., & Stroobandt, D. (2007). An experimental unification of reservoir computing methods. *Neural Networks*, *20*(3), 391–403.

Verstraeten, D., Schrauwen, B., & Stroobandt, D. (2005). Isolated word recognition using a liquid state machine. In *Proceedings of the 13th European Symposium on Artificial Neural Networks (ESANN)* (pp. 435–440).

Verstraeten, D., Schrauwen, B., Stroobandt, D., & Campenhout, J. V. (2005). Isolated word recognition with the liquid state machine: a case study. *Information Processing Letters*, *95*(6), 521 - 528. Retrieved from `http://www.sciencedirect.com/science/article/pii/S0020019005001523` doi: 10.1016/j.ipl.2005.05.019

Vreeken, J. (2004). *On real-world temporal pattern recognition using liquid state machines.* Unpublished doctoral dissertation, Masters Thesis, University Utrecht (NL).

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., & Lang, K. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, *37*(3), 328–339.

Wall, J., McGinnity, T., & Maguire, L. (2011). A comparison of sound localisation techniques using cross-correlation and spiking neural networks for mobile robotics. In *The 2011 international joint conference on neural networks (IJCNN)* (pp. 1981–1987).

Watts, M. (2009). A decade of kasabov's evolving connectionist systems: a review. *Part C: Applications and Reviews, IEEE Transactions on Systems, Man, and Cybernetics*, *39*(3), 253–269.

Werbos, P. J. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences.* Doctoral dissertation, Harvard University, Cambridge, MA, USA.

Werbos, P. J. (1994). *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting.* New York, NY, USA: Wiley-Interscience.

WHO. (2012, September). *WHO Fact sheet N317: Cardiovascular diseases (CVDs).* Retrieved from {`http://www.who.int/mediacentre/factsheets/fs317/en/index.html`}

Widrow, B. (2005, Jan.). Thinking about thinking: the discovery of the LMS algorithm. *Signal Processing Magazine, IEEE*, *22*(1), 100-106. doi: 10.1109/MSP.2005.1407720

Widrow, B., & Hoff, M. E. (1988). Adaptive switching circuits. In (pp. 123–134). Cambridge, MA, USA: MIT Press.

Widrow, B., & Lehr, M. (1990, Sep). 30 years of adaptive neural networks: perceptron, Madaline, and backpropagation. *Proceedings of the IEEE*, *78*(9), 1415-1442. doi: 10.1109/5.58323

Wikipedia. (2010). *Neuron — Wikipedia, the free encyclopedia.* `http://en.wikipedia.org/wiki/Neuron`. (Online; accessed

19-January-2010)

Wikipedia. (2012). *Cochlea — Wikipedia, the free encyclopedia.* `http://en.wikipedia.org/wiki/Cochlea`. (Online; accessed 21-December-2012)

Wilkes, A. L., & Wade, N. J. (1997). Bain on neural networks,. *Brain and Cognition*, *33*(3), 295 - 305. doi: DOI:10.1006/brcg.1997.0869

Wysoski, S., Benuskova, L., & Kasabov, N. (2008). Fast and adaptive network of spiking neurons for multi-view visual pattern recognition. *Neurocomputing*, *71*(13-15), 2563–2575.

Wysoski, S., Benuskova, L., & Kasabov, N. (2010). Evolving spiking neural networks for audiovisual information processing. *Neural Networks*, *23*(7), 819–835.

Wysoski, S. G., & Benuskova, L. (2006). Biologically realistic neural networks and adaptive visual information processing. *Bulletin of Applied Computing and Information Technology*, *4*(2).

A
Appendix: Algorithms

In this section we have provided the demonstration python code of the novel methods implemented in our study. Python v2.7 was used for the implementation of our algorithms. Some of the python libraries that you will require in order to run the following codes are: Brian, Numpy, Scipy, Pylab and OpenCV.

A.1 KEDRI's AER Software Simulator

```
'''
KEDRI's:
AER: Address Event Representation Simulator
Author: Kshitij Dhoble
Institute: KEDRI, AUT University, Auckland
Implemented using python OpenCV library (http://opencv.org/)
'''
#---------------------------------------------------------
import cv2         # OpenCV library
import numpy as np # Numpy library
#---------------------------------------------------------
def diffImg(t0, t1, thresh):
  # Calculate difference in log intensity
  d = cv2.absdiff(cv2.log(t1), cv2.log(t0))
  # Thresholding for AER Events
  aer = np.zeros(d.shape)
  aer[np.logical_and(d < -thresh, d < 0)] = -1
  aer[np.logical_and(d > thresh, d > 0)] = 1
  #aer = cv2.threshold(d, thresh, 1, cv2.THRESH_BINARY)[1]
  return aer
#--------------------------------------------------------
#Capture from Camera Live
#capture = cv2.VideoCapture(0)

#Capture from Video File
capture = cv2.VideoCapture('Great_Cars_GTO.mov')
#--------------------------------------------------------
#Create window to display the AER Represenation
cv2.namedWindow("AER Representation", cv2.CV_WINDOW_AUTOSIZE)

# Read first three frames & convert to grayscale:
t_minus_g = cv2.cvtColor(capture.read()[1], cv2.COLOR_RGB2GRAY)
t_g = cv2.cvtColor(capture.read()[1], cv2.COLOR_RGB2GRAY)
t_plus_g = cv2.cvtColor(capture.read()[1], cv2.COLOR_RGB2GRAY)

# Normalize above frame values between 0 and 1
t_minus = cv2.normalize(t_minus_g, t_minus_g, 0, 1, cv2.NORM_MINMAX, cv2.CV_32F)
t = cv2.normalize(t_g, t_g, 0, 1, cv2.NORM_MINMAX, cv2.CV_32F)
t_plus = cv2.normalize(t_plus_g, t_plus_g, 0, 1, cv2.NORM_MINMAX, cv2.CV_32F)

# AER threshold to generate spike event
thresh = 0.22
#--------------------------------------------------------
while True:
# Calculate difference in log intensity
diff = diffImg(t_minus, t, thresh)
# Display AER Representation
cv2.imshow("AER Representation", diff)

# Read next image
t_minus = t
t_plus_g = cv2.cvtColor(capture.read()[1], cv2.COLOR_RGB2GRAY)
t_plus = cv2.normalize(t_plus_g, t_plus_g, 0, 1, cv2.NORM_MINMAX, cv2.CV_32F)
t = t_plus

# Stop when 'Esc' key is pressed
key = cv2.waitKey(10)
if key != -1:
cv2.destroyAllWindows()
break
#--------------------------------------------------------
```

A.2 Cochleagram-based Spike Encoding

```
 '''
Cochleagram based Spike Encoding
Author: Kshitij Dhoble
Institute: KEDRI, AUT University, Auckland
Implemented using Brian Simulator (http://briansimulator.org/)
'''
# Program to convert sound (wav) file to spike trains

from pylab import *
from brian import *
from brian.hears import *
import os
import operator

seed(1)
```

```
out = []

#--- Read all files from defined directory path ----#
#path = 'xyz/'  # directory path

listing = os.listdir(path)

for infile in listing:
    print "Reading from file: " + infile,"\n"
    sound = loadsound(path+infile)
    #sound.level = 60*dB
    #sound = sound.ramp()
    cf = erbspace(1*Hz, 20*kHz, 125)
    cochlea = Gammatone(sound, cf)

    # Half-wave rectification and compression [x]^(1/3)
    ihc = FunctionFilterbank(cochlea,
          lambda x: 3*clip(x, 0, Inf)**(1.0/3.0))
    lowpass = LowPass(ihc, 4*Hz)
    output = lowpass.process()

    # Leaky integrate-and-fire model with noise and refractoriness
    eqs = '''
  dv/dt = (I-v)/(1*ms)+0.2*xi*(2/(1*ms))**.5 : 1
  I : 1
  '''

    # Apply filter
    anf = FilterbankGroup(lowpass, 'I', eqs, reset=0,
                        threshold=1, refractory=2*ms)

    # Monitor spikes
    M = SpikeMonitor(anf)

    run(sound.duration) # Run simulation
    out = M.spikes

    # Save the spike trains in a file
    filename = "dir_path_xyz/"+infile
    np.save(filename, out)
    print "current saved file is:"+filename

    # Plot the spike trains obtained from sound file
    raster_plot(M, title='Cochleagram Encoding Spikes')
    savefig(filename+".eps", format='eps')
    show()
```

A.3 deSNN Algorithm

```
'''
deSNN Demo Code
Author: Kshitij Dhoble and Nuttapod Nuntali
Institute: KEDRI, AUT University, Auckland
Implemented using Brian Simulator (http://briansimulator.org/)
'''
from pylab import *
from brian import *
import numpy as np
from brian.utils.progressreporting import ProgressReporter
from time import time
from core import *
from core.learner import *
from core.utils import *
import os
import operator

#------ Parameters and constants --------#

defaultclock.dt= 0.2 * ms

### Basic neuron and synapse parameters ###
tau_exc = 2*ms          # Excitatory synapse time constant
tau_exc_inh = 0.2*ms    # Feedforward connection time constant
tau_inh = 5*ms          # Inhibitory synapse time constant
tau_mem = 20*ms         # Neuron time constant
El = 20*mV              # Membrane leak
Vthr = 800*mV           # Spike threshold
Vrst = 0*mV             # Reset value
winh = 0.20*volt        # Fixed inhibitory weight
wexc = 0.40*volt        # Fixed excitatory weight
#wexc_inh = 1 * volt    # Fixed feedforward excitatory weight
UT = 25*mV              # Thermal voltage
refr = 4*ms             # Refractory period

### Learning related parameters (Fusi’s SDSP) ###
Vthm =  5*Vthr/8 #0.75*Vthr # Up/Down weight jumps
tau_ca = 5*tau_mem          # Calcium variable time constant
wca = 50 * mV               # Steady-state asymptode for Calcium variable
th_low = 1.7*wca            # Stop-learning threshold 1 (stop if Vca<thk1)
th_down = 2.2*wca           # Stop-learning threshold 2 (stop LTD if Vca>thk2)
th_up = (8*wca) - wca       # Stop-learning threshold 2 (stop LTP if Vca>thk3)
tau_p = 9* ms               # Plastic synapse (NMDA) time constant
wp_hi = 0.6* volt           # Plastic synapse high value
wp_lo = 0 * mvolt           # Plastic synapse low value
wp_drift = .25              # Bistability drift
wp_thr= (wp_hi - wp_lo)/2.+wp_lo  # Drift direction threshold
wp_delta = 0.12*wp_hi             # Delta Weight


#------------ Equations: Neuron Model ------------#
eqs_neurons = Equations(’’’
dv/dt=(El-v+ge+ge_p+ge_inh-gi_out)*(1./tau_mem) : volt
dge_p/dt=-ge_p*(1./tau_p) : volt
dge/dt=-ge*(1./tau_exc) : volt
dgi/dt=-gi*(1./tau_inh) : volt
dge_inh/dt=-ge_inh*(1./tau_exc_inh) : volt
gi_out = gi*(1-exp(-v/UT)): volt  # shunting inhibtion
’’’)

eqs_reset = ’’’
v=Vrst
’’’
#--------- Architecture of the networks -----------#

input_size = 5    # Number of neurons, in the input layer.
neurons_class = 1 # Number of neurons in each class
number_class = 1  # Number of class in the output layer
output_size = number_class*neurons_class
out = []

SIM_TIME = 8*ms   # Simulation run time
seed(1)
mod=0.8           # ROC Modulation Factor

#--- Read all files from defined directory path ----#
path = ’sdsp_journalWeight/’  ## define directory path
listing = os.listdir(path)

# Get all data files from directory
for infile in listing:
    print "Reading from file: " + infile,"\n"
```

```
##---------Spiketrain stimulus from file-------##
spiketimes= np.load(path+infile)
s=sorted(spiketimes, key=operator.itemgetter(1)) # Rank Order Coding (ROC)
rankW=cal_weight_multi(s,mod,input_size)         # Calculate synaptic weight
wp0=rankW

#Random Initiation of Weights
#wp0 = ((wp_hi-wp_lo)*rand(input_size, output_size))+wp_lo

##---Convert imported/selected spike trains to brian (spike train) format--##
inputSpikeTrain = SpikeGeneratorGroup(input_size, spiketimes)
net = Network(inputSpikeTrain)
net.reinit()

#------------ Neurons ---------------#
# Create Output layer Neurons
neurons = NeuronGroup(N=output_size, model= eqs_neurons, threshold=Vthr, reset= Vrst)
# Create Inhibitory Neuron Group
inh_neurons = NeuronGroup(N=output_size, model = eqs_neurons, threshold = Vthr, reset = Vrst)

#------------ Connections -----------#
wexc_inh = (0.8+(rand(len(inputSpikeTrain), len(inh_neurons))*0.5)) *volt
c_inter = Connection(inputSpikeTrain, inh_neurons, 'ge_inh', structure = 'dense')
c_inter.connect(inputSpikeTrain, inh_neurons, wexc_inh)
c_inh = Connection(inh_neurons, neurons, 'gi')
c_inh.connect_full(inh_neurons, neurons, weight = winh)

# Connection between the input layer and the output layer
synapses = Connection(inputSpikeTrain, neurons, 'ge_p', structure = 'dynamic')
synapses.connect(inputSpikeTrain, neurons, wp0)

# Fusi's SDSP/STDP equation
eqs_stdp='''
x : 1                   # fictional presynaptic variable
dC/dt = -C/tau_ca : volt # your postsynaptic calcium variable
V : volt                # a copy of the postsynaptic v
'''
stdp=STDP(synapses, eqs=eqs_stdp,
pre='w += (V>Vthm)*(C<th_up)*(th_low<C)*wp_delta
          - (V<=Vthm)*(C<th_down)* (th_low<C)*wp_delta; x',
  post='C += wca; V', wmax=wp_hi)
stdp.post_group.V = linked_var(neurons,'v')

#-------------Record spike activities------------------#
spikes  = SpikeMonitor(inputSpikeTrain, record=True)
outspikes  = SpikeMonitor(neurons, record=True)
print "Inspikes", "\n", spikes
print "Outspikes", "\n", outspikes
M = StateMonitor(neurons,'v',record=0)
#------------------------------------------------------#

@network_operation
def drift_equation():
    synapses.W = DenseConnectionMatrix(
    bistable_drift(synapses.W.todense(), len(inputSpikeTrain), len(neurons))
    )

def bistable_drift(w, a, b):

    w = w.flatten()
    up_idx = w>wp_thr
    down_idx = w<=wp_thr
    w[up_idx] += wp_drift*defaultclock.dt
    w[w>wp_hi] = wp_hi
    w[down_idx] -= wp_drift*defaultclock.dt
    w[w<wp_lo] = wp_lo
```

```
        return w.reshape(a,b)

    run(SIM_TIME)  # Run simulation

# Plot input spikes, initial & final weights
figure(1)
subplot(211)
raster_plot(spikes)
subplot(212)
plot(wp0.flatten(), '-o',label="Initial")
plot(synapses.W.todense().flatten(), '-v', label="Final")
legend()
show()
# Print Rank Order Coding Weights
print 'ROC Weights: \n', wp0
# Print SDSP Weights
print 'Final SDSP Weights: \n', synapses.W.todense()
```

A.4 Reservoir

A.4.1 LSM Reservoir

```
from brian import *
from core import *
import numpy as np
import os

#------ Parameters and constants --------#

TAU = '10*ms'         # Neuron time constant
NOISY_TAU = '10*ms'   # Stochastic neuron time constant
RESET_POTENTIAL = 0*mV
FIRING_THRESHOLD = 10*mV
RESET_MU = 0*mV       # Reset value
RESET_SIGMA = 3.0     # Reset value
THRESHOLD_SIGMA = 2.0 # Stochastic neuron model threshold
NB_NEURONS = 64       # Total no. of neurons in LSM (4x4x4)
NB_INPUTS=64          # No. of input neurons
input_weight=0.8      # Initial Weights
SIM_TIME = 8*second   # Simulation Time

#--------------------------- Select neuron model ------------------------#

model = LIFModel(TAU, threshold=FIRING_THRESHOLD, reset=RESET_POTENTIAL)
#LIFModel(TAU, threshold=FIRING_THRESHOLD, reset=RESET_POTENTIAL)
        #NoisyResetModel(TAU, RESET_MU, RESET_SIGMA, threshold=FIRING_THRESHOLD)
        #StepNoisyThresholdModel(TAU, RESET_POTENTIAL, FIRING_THRESHOLD, THRESHOLD_SIGMA)
        #NoisyThresholdModel(TAU, NOISY_TAU, FIRING_THRESHOLD, RESET_POTENTIAL)

seed(1)
#--- Read all files from defined directory path ----#
path = 'xyz/'  # define directory path xyz

#--- Read all files from defined directory path ----#
listing = os.listdir(path)
for infile in listing:
    print "Reading from file: " + infile

    #------- Spike-train stimulus from file --------#
    spiketimes = np.load(path+infile)
    #Convert imported/selected spike trains to brian (spike train) format
    inputSpikeTrain = SpikeGeneratorGroup(NB_INPUTS, spiketimes)
    ################## Main Reservoir #################
    net = Network(inputSpikeTrain)
    net.reinit()
    ##--------Initiate Neuron group---------------
    neuronGroup =model.getNeuronGroup(NB_NEURONS)
    #Create input connection - #Dynamic Connections
    conn = Connection(inputSpikeTrain, neuronGroup, 'V', weight=input_weight *mV,
                      structure='dynamic', sparseness = 0.75)

    # Create LSM - Make Small World Connections
    c=SmallWorldConnection(neuronGroup, neuronGroup,'V')
    # Define/create LSM structure/topology and connect
    d,coord,w= c.connect(neuronGroup, neuronGroup,NB_NEURONS,(4,4,4))

    #-------------Record spike activities------------------#
    spikes  = SpikeMonitor(neuronGroup, record=True)

    net.add(neuronGroup, conn, c,  spikes)
    net.run(SIM_TIME)  # Run the simulation
    ################ End Main reservoir ###############

    filename= "xyz/"+infile # Define Save Directory Path xyz
    np.save(filename, spikes.spikes)
    print "Current file's liquid state/response saved as:"+filename
```

A.4.2 Learning LSM Reservoir

```
 '''
Learning Reservoir (LSM + STDP Learning rule)
Authors: Kshitij Dhoble
Institute: KEDRI, AUT University, Auckland
Implemented using Brian Simulator (http://briansimulator.org/)
'''
from brian import *
from core import *
import numpy as np
import os


#------ STDP Parameters and constants --------#
tau_pre= 0.01*ms     # Pre-synaptic time for STDP
tau_post=tau_pre     # Post-synaptic time for STDP
gmax=.05       # Max. weight limit - STDP
dA_pre=.01     # Initial pre-synaptic connections weights
dA_post=-dA_pre*tau_pre/tau_post*1.05


#------ Parameters and constants --------#

TAU = '10*ms'         # Neuron time constant
NOISY_TAU = '10*ms'   # Stochastic neuron time constant
RESET_POTENTIAL = 0*mV
FIRING_THRESHOLD = 10*mV
RESET_MU = 0*mV       # Reset value
RESET_SIGMA = 3.0     # Reset value
THRESHOLD_SIGMA = 2.0 # Stochastic neuron model threshold
NB_NEURONS = 64       # Total no. of neurons in LSM (4x4x4)
NB_INPUTS=64          # No. of input neurons
input_weight=0.8      # Initial Weights
SIM_TIME = 8*second   # Simulation Time

#--------------------------- Select neuron model ------------------------#

model = NoisyThresholdModel(TAU, NOISY_TAU, FIRING_THRESHOLD, RESET_POTENTIAL)
```

```
#LIFModel(TAU, threshold=FIRING_THRESHOLD, reset=RESET_POTENTIAL)
        #NoisyResetModel(TAU, RESET_MU, RESET_SIGMA, threshold=FIRING_THRESHOLD)
        #StepNoisyThresholdModel(TAU, RESET_POTENTIAL, FIRING_THRESHOLD, THRESHOLD_SIGMA)
        #NoisyThresholdModel(TAU, NOISY_TAU, FIRING_THRESHOLD, RESET_POTENTIAL)

seed(1)
#--- Read all files from defined directory path ----#
path = 'dataset/simpleMovingObj/'  # define directory path

#--- Read all files from defined directory path ----#
listing = os.listdir(path)
for infile in listing:
    print "Reading from file: " + infile

    #------- Spike-train stimulus from file --------#
    spiketimes = np.load(path+infile)
    #Convert imported/selected spike trains to brian (spike train) format
    inputSpikeTrain = SpikeGeneratorGroup(NB_INPUTS, spiketimes)

    ################## Main Reservoir #################
    net = Network(inputSpikeTrain)
    net.reinit()
    ##--------Initiate Neuron group---------------
    neuronGroup =model.getNeuronGroup(NB_NEURONS)
    #Create input connection - #Dynamic Connections
    conn = Connection(inputSpikeTrain,
      neuronGroup, 'V',
      weight=input_weight *mV,
      structure='dynamic',
      sparseness = 0.75)

    # Create LSM - Make Small World Connections
    c=SmallWorldConnection(neuronGroup, neuronGroup,'V')
    # Define/create LSM structure/topology and connect
    d,coord,w= c.connect(neuronGroup, neuronGroup,NB_NEURONS,(4,4,4))

    ###---------- Explicit STDP rule -------###
    eqs_stdp='''
    dA_pre/dt=-A_pre/tau_pre : 1
    dA_post/dt=-A_post/tau_post : 1
    '''
    dA_post*=gmax
    dA_pre*=gmax
    stdp=STDP(c,eqs=eqs_stdp,pre='A_pre+=dA_pre;w+=A_post',
      post='A_post+=dA_post;w+=A_pre',wmax=gmax)


    #-------------Record spike activities------------------#
    spikes  = SpikeMonitor(neuronGroup, record=True)

    net.add(neuronGroup, conn, c,  spikes, stdp)
    net.run(SIM_TIME)  # Run the simulation
    ################# End Main reservoir ###############

    filename= "HeartDataset_LSM_txt/LIF/"+infile # Define Save Directory Path
    np.save(filename, spikes.spikes)
    print "Current file's liquid state/response saved as:"+filename
```

LSM Small World Connection

```
'''
LSM Small World Connection
Authors: Nuttapod Nuntali and Stefan Schlibes
Institute: KEDRI, AUT University, Auckland
Implemented using Brian Simulator (http://briansimulator.org/)
```

```
'''

from brian import *
from numpy import *

class SmallWorldConnection(Connection):

    def connect(self, neuron_group1, neuron_group2, \
                    nb_neurons, \
                    grid_structure, \
                    lamda=2, \
                    Cex_ex=0.3, Cinh_inh=0.1, Cex_inh=0.2, Cinh_ex=0.4, \
                    ratio_ex=0.8):

        W = zeros((nb_neurons, nb_neurons))
        D = zeros((nb_neurons, nb_neurons))
        P = zeros((nb_neurons, nb_neurons))

        # Determine all excitatory neurons
        is_excitatory = rand(nb_neurons) <= ratio_ex

        # Unpack the grid structure
        x,y,z = grid_structure

        # Assign coordinates for each neuron in a 3D grid
        coordinates = []
        for i in xrange(x):
            for j in xrange(y):
                for k in xrange(z):
                    coordinates += [[i,j,k]]

        # Compute the distance between all neurons in the grid
        for i in xrange(nb_neurons):
             for j in xrange(nb_neurons):
                 # Calculate distance
                 dx = coordinates[i][0]-coordinates[j][0]#Ax-Bx
                 dy = coordinates[i][1]-coordinates[j][1]#Ay-By
                 dz = coordinates[i][2]-coordinates[j][2]#Az-Bz
                 distance = (dx**2 + dy**2 + dz**2)**0.5

                 # Store the distance in the matrix
                 D[i][j] = distance

                 # Compute the weight matrix

                 C = 0
                 # Assign C for each neuron type connection (inhibit and exhibit)
                 if is_excitatory[i] and is_excitatory[j]:
                     C = Cex_ex
                 elif is_excitatory[i] and not is_excitatory[j]:
                     C = Cex_inh
                 elif not is_excitatory[i] and is_excitatory[j]:
                     C = Cinh_ex
                 elif not is_excitatory[i] and not is_excitatory[j]:
                     C = Cinh_inh

                 # Compute probabilistic connection
                 p_conn = C * (exp(-(D[i][j]**2)/(lamda**2)))
                 P[i][j]=p_conn
                 #p_conn = (exp(-(D[i][j]**2)/(lamda**2)))
                 #print 'P_conn = ', p_conn
                 if rand() < p_conn:
                             # W[i][j]=C*p_conn *mV
                     if is_excitatory[i]:
                         W[i][j]= 1.62 *mV
                         #W[i][j]= (2.*p_conn)*mV
                     else:
                        W[i][j]= -9 *mV
                        #W[i][j]= -(10.*p_conn)*mV

# Connect the specified two neuron groups using the generated weight matrix
        Connection.connect(self, neuron_group1, neuron_group2, W)

        return D, coordinates, W
```

Rank Order Coding

```
'''
ROC Weight Calculation
Authors: Nuttapod Nuntali
Institute: KEDRI, AUT University, Auckland
Implemented using python
'''

 def cal_weight_multi(s,mod,input_size):
    firstspiketimes=spiketime_array(s,input_size)
    spiketimes=[]
    for j in xrange(len(firstspiketimes)):
        if(len(firstspiketimes[j])!=0):
           spiketimes +=[(j,float(firstspiketimes[j][0]))]
    spiketimes=sorted(spiketimes, key=operator.itemgetter(1))
    ss=zeros((input_size,1))
    for i in xrange(len(spiketimes)):
        ss[spiketimes[i][0]][0]=float(mod**i)
    return ss

    def spiketime_array(spiketime,NB_NEURONS):
    outspike=[]
    tmp=[]
    for j in xrange(NB_NEURONS):
        for i in xrange(len(spiketime)):
             if (spiketime[i][0]==j):
                 tmp.append(spiketime[i][1]/ms)
        outspike+=[tmp]
        tmp=[]
    return outspikes
```