Evolving Probabilistic Spiking Neural Networks for Modelling and Pattern Recognition of Spatio-temporal Data on the Case Study of Electroencephalography (EEG) Brain Data

Nuttapod Nuntalid

A thesis submitted to

Auckland University of Technology

in fulfilment of the requirements for the degree of

Doctor of Philosophy (PhD)

2012

Knowledge Engineering and Discovery Research Institute

Faculty of Design and Creative Technologies

Primary Supervisor: Prof. Nikola Kasabov

Secondary Supervisor: Assoc. Prof. Petia Goergieva

Evolving Probabilistic Spiking Neural Networks for Modelling and Pattern Recognition of Spatio-temporal Data on the Case Study of Electroencephalography (EEG) Brain Data

Nuttapod Nuntalid

A thesis submitted to

Auckland University of Technology

in fulfilment of the requirements for the degree of

Doctor of Philosophy (PhD)

2012

Knowledge Engineering and Discovery Research Institute

Table of Contents

List of Figures

List of Tables

List of Abbreviations

μ	Gaussian Distribution Mean
ANN	Artificial Neural Network
BCI	Brain Computer Interface
BSA	Ben Spike Encoder Algorithm
CNGM	Computation Neuro-genetic Models
CSN	Chaotic Spiking Neuron
CT	Continuous Stochastic Threshold Model
DepSNN	Dynamic Evolving Probabilistic Spiking Neural Network
DepSNNm	Dynamic Evolving Probabilistic Spiking Neural Network (membrane potentials)
DepSNNs	Dynamic Evolving Probabilistic Spiking Neural Network (synapses)
E	Error
EEG	Electroencephalography
EPD	Evoked Potential Duration
epSNNr	Evolving Probabilistic Spiking Neural Network Reservoir
eSNN	Evolving Spiking Neural Network
Ex	Excitatory
FIR	Finite Impulse Response Filter
GRN	Gene Regulatory Network
HMM	Hidden Markov Model
HAS	Hough Spiker Algorithm
IF	Integrate-and-Fire Model
Inh	Inhibitory
LIF	Leaky Integrate and Fire Model

LSM	Liquid State Machine
LTD	Long-term Depression
LTP	Long-term Potentiation
MLP	Multi-layer Perceptron
NR	Stochastic Noisy Reset Model
POC	Population Coding
pSNM	Probabilistic Spiking Neural Model
pSNN	Probabilistic Spiking Neural Network
PSTH	Peristimulus Time Histograms
ReSuMe	Remote Supervised Method
RO	Rank Order
ROC	Rank Order Coding
SDSP	Spike Driven Synaptic Plasticity
SNM	Spiking Neural Model
SNN	Spiking Neural Network
SPAN	Precise-time Spike Pattern Association Neuron
SSPAN	Stochastic Precise-time Spike Pattern Association Neuron
ST	Stochastic Step-wise Noisy Threshold Model
STD	Spatio-temporal Data
STDP	Spike-time Dependent Plasticity
Σ	Standard Deviation

Declaration

"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 stated 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."

Auckland, 2012

Nuttapod Nuntalid

Acknowledgements

First of all, I am really grateful to God Almighty who has given me inspiration to complete this study.

I am deeply grateful to my primary supervisor, Prof. Nikola Kasabov, for his sincere guidance, support and patience. Without his support, this thesis would not have been accomplished.

Many thanks to my secondary supervisor, Assoc.Prof. Petia Goergieva, for supplying fundamental information on EEG and BCI.

I also extend thanks to the past and present members of KEDRI (Knowledge Engineering and Discovery Research Institute) at the Auckland University of Technology, who directly or indirectly provided me with the conditions required for completion. From the valuable information provided by them, I have accumulated important knowledge related to the research.

I am very grateful to Joyce D'Mello, the manager of KEDRI, for all her kindness and support which she has offered me since before the time I joined KEDRI.

I am delighted to thank my parents for their support, all the things I've learnt, what you have taught me remains by far the most important.

I would also like to thank the many researchers, whom I directly contacted regarding methods related to this study and their technical suggestion and support such as Prof. Benjamin Schrauwen, Prof. Hiroyuki Torikai, Dr. Stefan Schilibs and Dr. Ammar Mohammed.

I also thanks to my reviewers for their time and effort including Prof. Bogdan Gabrys, Prof. Zeng-Guang Hou and Dr. George Coghill.

Finally, I wish to pass on my sincere thanks to many researchers and scholars worldwide whose works have been consulted and drawn upon during the research.

Abstract

The use of Electroencephalography (EEG) in Brain Computer Interface (BCI) domain presents a challenging problem due to presence of spatial and temporal aspects inherent in the EEG data. Many studies either transform the data into a temporal or spatial problem for analysis. This approach results in loss of significant information since these methods fail to consider the correlation present within the spatial and temporal aspect of the EEG data. However, Spiking Neural Network (SNN) naturally takes into consideration the correlation present within the spatiotemporal data. Hence by applying the proposed SNN based novel methods on EEG, the thesis provide improved analytic on EEG data. This thesis introduces novel methods and architectures for spatio-temporal data modelling and classification using SNN. More specifically, SNN is used for analysis and classification of spatiotemporal EEG data.

In this thesis, for the first time, Ben Spiker Encoder Algorithm (BSA) is applied on EEG data and its applicability is demonstrated successfully. Moreover, three new stochastic neural models are introduced; namely, Stochastic Noisy Reset (NR), Stochastic step-wise noisy threshold model (ST) and Continuous stochastic threshold (CT). The stochastic models mimic activity of biological neurons while retaining low implementation cost. Also, a modification of precise-time spike pattern association neuron (SPAN) called stochastic precise-time spike pattern association neuron (SSPAN) is introduced. The SSPAN demonstrates superior performance than SPAN, especially when used with stochastic neural models.

A novel Dynamic Evolving Probabilistic Spiking Neural Network (DepSNN) is introduced as an extension of the eSNN model. Five novel variants of DepSNN (DepSNNm, DepSNNs, NR-DepSNNs, ST-DepSNNs, and CT-DepSNNs) are presented and the results show that it requires high density of input spikes if SDSP learning is to be made efficient.

The thesis then offers a critical analysis of Electroencephalography (EEG) data analysis and classification methods used to date. The developed SNN methods have been adopted in EEG analysis and classification investigated on two datasets (real-world audio-visual stimuli perception EEG dataset and P300 based BCI dataset), with promising results relative to other methods.

Furthermore, the proposed novel SNN architecture for spatio-temporal data termed evolving probabilistic SNN reservoir (epSNNr) shows enhanced performance when integrated with stochastic neural models. The utilization of 3D Localization mapping along with DepSNN as a readout unit, showed very outstanding results especially on P300 based BCI application.

Chapter 1
Introduction

Spiking Neural Networks (SNN) is the third generation of Artificial Neural Networks (ANN). Artificial Neural Networks have been successfully applied to problems in classification and medical decision support, among others. Throughout the past decade, neural network research has tended towards more biologically realistic models and the need to better comprehend the significant information processing competencies of the mammalian brain. This current focus has culminated in more complicated and biologically probable connectionist models, including Spiking Neural Networks (SNN). A defining characteristic of SNNs is that they prototype the transmission of the biological network in time and hence by themselves constitute a continual dynamic system. This makes them particularly applicable to sequential patterns and, as such, they are suitable alternatives to the HMM method.

SNN, which simulates information-processing by biological neurons, possesses two major characteristics. First, the weights in SNN are of different strength and secondly, a spike is released when the signal threshold is exceeded. A detailed discussion of SNN models (SNM), including LIF and Hodgkin Huxley models, is available in Gerstner and Kistler (2002). Nevertheless, SNN theory makes no provision for the processing of temporal prototypes. The currently accessible learning algorithms for Spiking Neural Networks can process non-temporal vector data only. The learning rule requires that free parameters in the Spiking Neural Networks models are alterable, enabling the SNN to learn a suitable output for a given collection of spatio-temporal inputs.

This research aims to develop modules and a simulator for Probabilistic Spiking Neuron Model (pSNM). This simulator will be used to model the evolving connectivity of pSNN on a larger scale, dealing with spatiotemporal data. The approach will be tested on a case study of EEG, which is a type of spatio-temporal brain data.

1.1 Statement of the Problem

In previous research, network structures have been evolved using standard SNN, which limits the functionality of the network during a learning process. New types of SNN, termed probabilistic SNN (pSNN), were recently proposed in by Kasabov, offering greater flexibility in both neural functions and connectivity (Kasabov, 2010). Spatiotemporal pattern recognition is crucial in improving the accuracy of data mining approaches. However, neither SNN nor pSNN have been applied to spatiotemporal data processing. In particular, application to human EEG data would contribute to the development of a brain computer interface (BCI). Therefore, this study aims to develop modules and a simulator for pSNM and to use this simulator to model the evolving connectivity of pSNN on a large scale, dealing with spatiotemporal data. The modules will be tested on an EEG case study.

1.2 Objectives of the Study

The objectives of this study are as follows:

(1) Investigate and apply a suitable spike encoding method for EEG data analysis.

(2) Develop new models of SNN inspired by the Probabilistic Spiking Neural model.

(3) Develop a new model of SNN for spatio-temporal data (STD) processing, including methods for online learning.

(4) Develop a new model of SNN utilizing the reservoir approach for spatio-temporal data (STD) processing, including methods for adaptive learning and visualization.

(5) Develop an SNN system for processing brain EEG data as spatial-temporal inputs, and evaluate the efficiency of the system.

(6) Apply the developed SNN and systems to the design of a novel BCI.

1.3 Significance of the Study

Determining suitable spatiotemporal pattern recognition approaches are crucial for improving the accuracy and decreasing processing time of data mining approaches. So far the applicability of spatiotemporal data processing approaches such as SNN (which are capable of naturally processing spatiotemporal data) has not been investigated thoroughly. Therefore, this research aims to develop novel spatiotemporal pattern recognition techniques utilizing SNN. Also, the existing SNM has limited functionality in terms of learning process. Hence, new models of SNM, inspired by Kasabov’s pSNM, are introduced in this study, particularly for application on human EEG data.

Appropriate spike representation of data is also important since SNN processes spatiotemporal data as spikes. Therefore, besides the development of novel spatiotemporal processing techniques based on eSNN, suitable fast spike encoding methods for human EEG data is also essential. An encoder that appropriately represents the data into spikes along with novel SNN methods will greatly contribute to the further development of brain computer interfaces (BCI).

Furthermore, the software simulator used in this research will provide insights into the brain activities and its functions. This will result in more efficient and effective simulation of spiking neuron connectivity during spatiotemporal data processing.

1.4 Research Questions

The following research questions will be addressed:

(1) How can new models of SNN, inspired by the Probabilistic Spiking neural model, be developed?

(2) How should we proceed with adapting the new SNN for spatio-temporal data (STD) processing?

(3) How can the new SNN utilizing reservoir approach be developed for spatio-temporal data (STD) processing?

(4) How can the new SNN systems be developed for processing EEG data as spatial-temporal input? How will the efficiency of these systems be evaluated?

(5) How can the new SNN systems be used to design a novel BCI?

(6) How can the new SNN models be utilized for brain data modeling?

1.5 Methodology

Corresponding to the research objectives, the following methodology is derived.

Firstly, a literature review based on past and current SNN based approaches is carried out. This allows in better understanding of the advantages and limitations present in the existing approaches and its functions such as the dynamicity of a synapse, learning rules, spike encoding methods, characteristic of EEG, and its applications. This in turn results in the development of new SNN approaches for spatiotemporal pattern recognition such as the novel pSNM.

Second is the implementation phase, considered as one of the most important steps in this study. In this phase designs for new SNN architectures for spatio-temporal pattern recognition of EEG needs to be determined. This is achieved through initial pilot studies. The new models are developed in Python language because it is open source, platform independent and has large collection of SNN libraries.

The last phase is testing and validation. Each method is first evaluated on synthetic dataset so that the performance and characteristics of the proposed system can be investigated. Then partial real world EEG datasets were used for optimizing parameters of the proposed SNN methods. Corresponding to the main objective of this study, the proposed methods design focuses on online processing capability and its suitability for BCI based applications. In this study, each dataset was split into training set and testing set for evaluation. The performance of the SNN models is evaluated in terms of classification accuracy and compared to traditional methods.

Hence, the methodology of this study is concluded in a research framework demonstrated in figure 1.1, which consists of 4 main stages of information processing those need to be explored and developed: (a) datasets; (b) converting continuous inputs into spikes; (c) spatio-temporal pattern recognition; (d) quality evaluation.

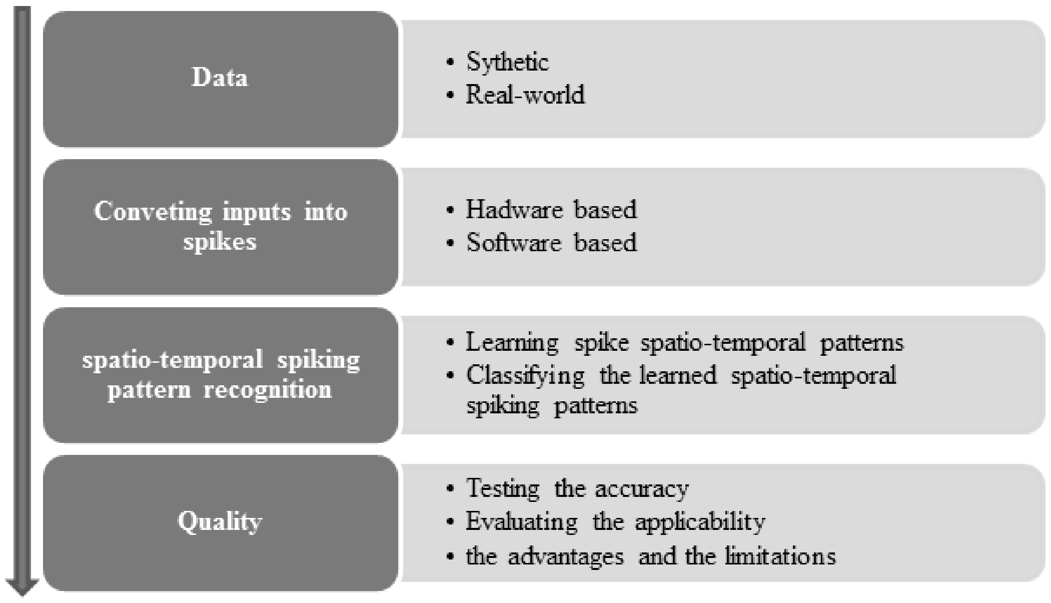

Data
• Sythetic
• Real-world
Conveting inputs into spikes
• Hadware based
• Software based
spatio-temporal spiking pattern recognition
• Learning spike spatio-temporal patterns
• Classifying the learned spatio-temporal spiking patterns
Quality
• Testing the accuracy
• Evaluating the applicability
• the advantages and the limitations

Figure 1.1: Research Framework

1.6 Outline of the Research

The research comprises the following twelve chapters:

Chapter 1 – Introduces the research, defines the problem, highlights the objectives and significance of the study, poses the research questions and outlines the methodology.

Chapter 2 - This chapter discusses the mechanisms of biological neurons which have inspired the development of SNN.

Chapter 3 - This chapter proposes the novel stochastic spiking neural models and describes an initial exploration of their behaviours.

Chapter 4 - In this chapter, the Stochastic Precise-time Spike Pattern Association Neuron (SSPAN) is introduced and its behaviour is investigated. This model is a modification of SPAN: Precise-time Spike Pattern Association Neuron, in which the deterministic LIF model has been replaced with the stochastic spiking neural models introduced in Chapter 3.

Chapter 5 - This chapter introduces Dynamic Evolving Probabilistic Spiking Neural Network (DepSNN), an extension of the eSNN model, in which both deterministic LIF and the stochastic models are utilized.

Chapter 6 - This chapter introduces Spiking Neural Network Reservoir (epSNNr), in which stochastic neural models have not only replaced deterministic LIF, but are used to introduce a non-deterministic component into a liquid state machine.

Chapter 7 - This chapter briefly introduces EEG and reviews the current situation regarding EEG and SNN-based EEG applications.

Chapter 8 - This chapter investigates the proposed spike encoding method, with emphasis on its suitability for EEG and the novel approach of transforming EEG data into spikes.

Chapter 9 - This chapter proposes the network architectures for EEG Spatio-temporal Pattern Recognition on Stochastic Precise-time Spike Pattern Association Neuron (SSPAN), and proposes the frameworks for online EEG Spatio-temporal Pattern Recognition on DepSNN and Online EEG Spatio-temporal Pattern Recognition on epSNNr.

Chapter 10 - This chapter discusses the implementation and investigates the feasibility and performance of Spatio-temporal Pattern Recognition (SSPAN, DepSNN, epSNNr and mixed methods) during processing of real-world audio-visual stimuli perception EEG data.

Chapter 11 - This chapter discusses the implementation and investigates the feasibility and performance of DepSNN on a P300 based BCI. The results are compared with those of an existing method applied to the same dataset.

Chapter 12 - This chapter concludes the research and discusses the contribution of this thesis to SNN research. This chapter also provides recommendations for future improvements.

1.7 Contribution of the Research Published as Peer-reviewed Journal and Conference Papers

Most of experiments in this study have been shared with the scientific community in six blind peer-review international academic papers and one poster as the following:

(1) Kasabov, N., Dhoble, K., **Nuntalid, N.**, & Indiveri, G. (2013). Dynamic Evolving Spiking Neural Networks for On-line Spatio- and Spectro-Temporal Pattern Recognition. *Neural Networks*(Autonomous Machine Learning).

Contribution involves:

- Application and performance evaluation of DepSNN on real-world EEG.
- Application of BSA spike encoding for EEG data.
- Application of Rank Order Code.
- Application of STDP/SDSP learning in DepSNN.

(2) Dhoble, K., **Nuntalid, N.**, Indiveri, G., & Kasabov, N. (2012, 10-15 June 2012). Online spatio-temporal pattern recognition with evolving spiking neural networks utilising address event representation, rank order, and temporal spike learning Symposium conducted at the meeting of the Neural Networks (IJCNN), The 2012 International Joint Conference on doi:10.1109/ijcnn.2012.6252439.

Contribution involves:

- Application of Rank Order Code.

- Application of STDP/SDSP learning in DepSNN.
- Performance evaluation of DepSNN and its variants.

(3) **Nuntalid, N.**, Dhoble, K., & Kasabov, N. (2011). *EEG Classification with BSA Spike Encoding Algorithm and Evolving Probabilistic Spiking Neural Network* [Proceedings of the 18th international conference on Neural information processing: theory and algorithms - Volume Part I]. Changhai, China: Springer.

Contribution involves:

- Integration and application of Stochastic Neural Models in epSNNr.
- Application of epSNNr on real-world Riken EEG dataset.
- Evaluation of epSNNr for online learning.

(4) 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* [Proceedings of the 18th international conference on Neural information processing: theory and algorithms - Volume Part 3]. Changhai, China: Springer.

Contribution involves:

- Evaluation of LSM based epSNNr on synthetic data.
- Characteristics and parameters tuning of stochastic reservoir.

- Performance comparison with non-SNN based approaches.

(5) Schliebs, S., **Nuntalid, N.**, & Kasabov, N. (2010). *Towards spatio-temporal pattern recognition using evolving spiking neural networks* [Proceedings of the 17th international conference on Neural information processing: theory and algorithms - Volume Part I]. Sydney, Australia: Springer-Verlag.

Contribution involves:

- Proposal of Stochastic Neural Models.
- Performance comparison of stochastic against non-deterministic neural models.
- Integration and performance evaluation of reservoir with and without Stochastic Neural Models.

(6) **Nuntalid, N**., Kasabov, N. (in progress). Dynamic Probabilistic Evolving Spiking Neural Networks for Spatial-temporal EEG Pattern Recognition. *Evolving Connectionist System (ECOS).*

Contribution involves:

- Integration and performance evaluation of reservoir with and without Stochastic Neural Models.
- Integration and performance evaluation of DepSNN with and without Stochastic Neural Models.
- Integrate and performance evaluation of reservoir utilizing 3D localization mapping and depSNN in the readout unit.

- Application of above contribution on real-world P300 EEG based BCI dataset.

The following diagram summarises these contributions in terms of the datasets, problems to be solved and proposed models:

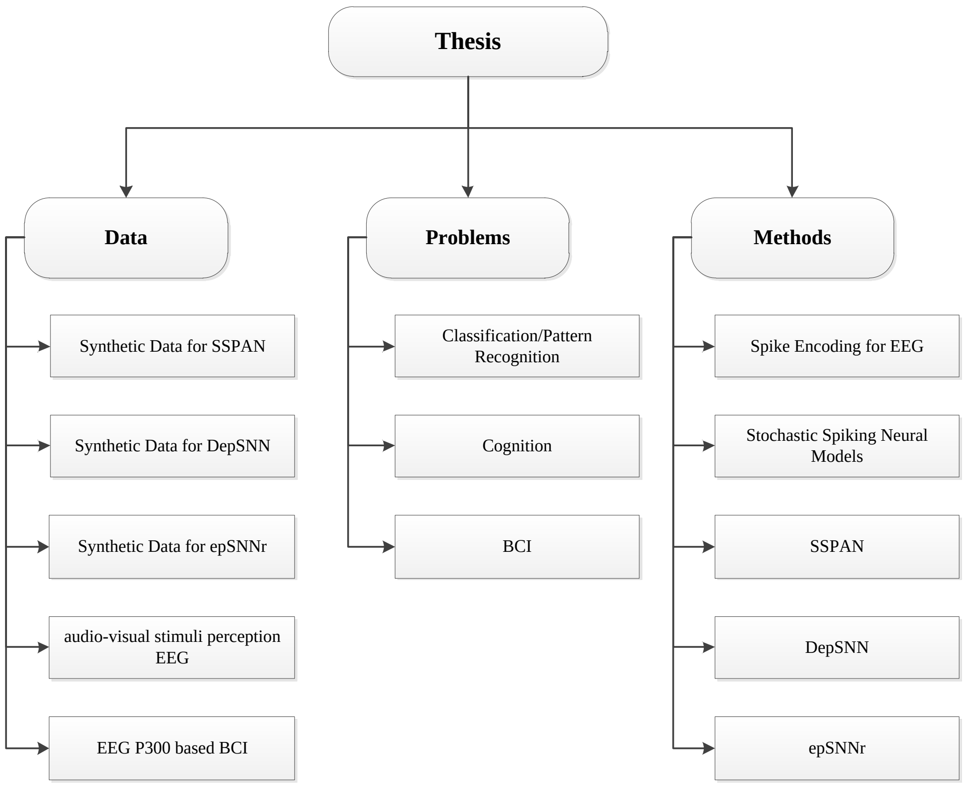

Thesis
Data
Problems
Methods
Synthetic Data for SSPAN
Synthetic Data for DepSNN
Synthetic Data for epSNNr
audio-visual stimuli perception EEG
EEG P300 based BCI
Classification/Pattern Recognition
Cognition
BCI
Spike Encoding for EEG
Stochastic Spiking Neural Models
SSPAN
DepSNN
epSNNr

Figure 1.2: Schematic illustrating the contributions of this thesis to SNN research, in terms of problems solved, methods developed and dataset.

Chapter 2

Spiking Neural Networks

Artificial Neural Networks (ANNs) models are mathematical models comprising an interconnected group of artificial neurons based on biological neural networks. Information is processed using a computational connectionist approach. In ANNs, the artificial nodes are known as neurons. ANNs have been successfully applied to data intensive problems such as classification, medical decision support, data mining, sales forecasting and target marketing, forecasting economic indicators and pattern recognition.

In this chapter, the relevant background information is introduced, including previous and current research on Spiking Neural Networks (SNN), the reservoir computing approach and SNN learning rules. Certain aspects of SNNs are highly relevant to this study and have inspired the development of probabilistic Spiking Neural Models (pSNM).

2.1 Spiking Neurons

The brain is a specially-adapted organ through which sentient organisms interact with their environment. This complex organ can intercept and process vast streams of input information in a highly efficient way. A machine modelled on the human brain, which can be utilized for multiple languages, is the ultimate goal of Artificial Intelligence (AI) research. On close examination, the brain comprises a large number of nerve cells known as neurons. These neurons establish connections with each other to form a live Neural Network. Artificial Neural Networks, which constitute a sub-field of Artificial Intelligence, attempt to prototype these biological neurons and to emulate the brain’s processing efficacy by establishing networks between these model neurons (Booij, 2004).

Mass and colleagues (2001) proposed that computational spiking neural networks are innately embedded in time (Maass, Natschläger, & Markram, 2002). According to Brader and his colleagues, if a neuron receives the same sequence of spikes, it will reside in the same final state with no interference from other parameters. Two types of neurons have been described; efferent (motor neurons, which stimulate movement) and afferent (sensory neurons, through which sensation is perceived). A causal relation may exist between these entities and the neuron of interest may lie beyond the transmission delay (in the past or future). Given that the past spiking activity of a neuron affects its membrane potential and influences its reaction to the next spike, the neuron is itself causally linked to transmission delay. This linkage provides a unified framework for the current neuron connected to all of its efferent and afferent neurons (Brader, Senn, & Fusi, 2007).

From biological observation, real SNN neurons are known to be sparsely and irregularly connected in space, and the variability of their spike flows implies that they communicate irregularly in time with a low average activity. The nodal spatial distribution and temporal activity describe the network topology and the network dynamics of the SNN, respectively. Moisy and Bohte stated that "It is important to note that the network topology becomes a simple underlying support to the neural dynamics, but that only active neurons are contributing to information processing. The novelty of the concept of SNNs means that many lines of research are still open and are actively being pursued" (Paugam-Moisy & Bohte, 2009). When performing a perceptual task, precision pooling occurs when an organism's decisions are based on the activities of a small set of highly informative neurons. The global pooling hypothesis states that the activities of all active neurons (or perhaps all active neurons in a particular brain region) contribute to an organism's perception and thus, to an organism's perceptual decision (Shadlen & Newsome, 1996).

Hence, the model studied in this research is focussed mainly on the Spiking Neural Model, which is currently attracting much attention in the ANN discipline. A formal definition of neural networks is presented in this chapter, together with descriptions on how they are modelled. Section 2.1.1 explains the working mechanism of biological neurons. Section 2.1.2 outlines the different approaches to modelling these neurons. Section 2.1.3 introduces the Spiking Neural Models and their applications. Section 2.1.4 presents the general architecture of the network, while 2.1.5 describes the likely coding patterns for altering an input signal so that it can be provided to a neural network. Section 2.1.6 provides an overview of SNN applications.

2.1.1 Biological neurons

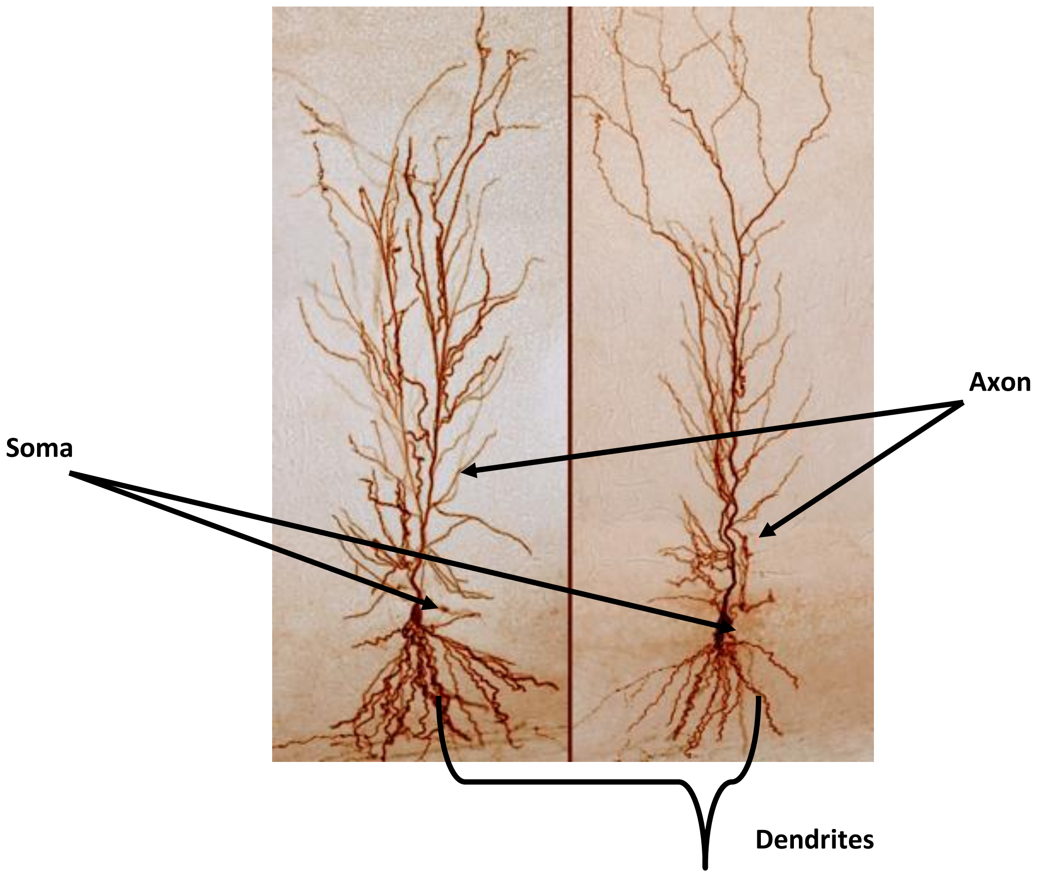


Figure 2.1: Images of Biological Neurons

To emulate complex biological neural networks such as occur in the brain, one must comprehend biological neurons, the languages of neural networks. Indeed, most of the ANN vocabulary derives from the biological equivalents of artificial components. In this section, biological neurons and their functioning are discussed. Neurons, which constitute the real processing units of the brain, perform simple calculations but compared to silicon chips their processing speed is very slow.

Nevertheless, large quantities of these simple units produce a powerful network. A silicon based computer typically has a single processing unit; within a neural network the entire set of neurons operates in parallel. To establish this network every neuron is connected to, on average, thousands of other neurons. Though various types of neurons exist in the brain, their basic structure is the same. The cell-body or the soma of the neuron radiates numerous refined networking fibres known as dendrites, together with one or more axons that branch from the base before extending towards other neurons, as shown in Figure 2.1. Towards the bottom of that figure, the triangular soma is indicated, with the dendrite tree surrounding it. The axon spreads up and branches towards the top of the image. The basic functioning of a neuron is described below.

Neurons hold a small negative electrical charge of -70 mV, known as their inactive capacity. This capacity is enhanced by incitements from other neurons. When the capacity reaches a limit, normally around -55 mV, the neuron shoots an electrical pulse along its axon, termed a spike. At the termination of the axon, the axon-branches link to dendrites of other neurons. This linkage between the neurons is known as a synapse. When a spike touches such a synapse it alters the electrical capacity in the dendrites of the recipient neuron. Because this procedure is comparatively time- consuming, the influence is deferred by a specific time which is typical for that synapse. The transmitter and receiver of the spike are termed the presynaptic and postsynaptic neuron, respectively. Depending on type of synapse, this alteration can cause the capacity of the postsynaptic neuron capacity to rise (positive effect), or drop (negative effect). A positive effect induces propulsion of the neuron; in this case the synapse is known as excitatory, whereas in the opposite case the synapse is known as inhibitory. The influence of the capacity-alteration is temporary; after some time it disappears as the default state of the neuron is its inactive capacity. Having fired a spike, a neuron requires some time to recuperate before it can spike again. This time interim is known as the refractory period. The type of synapse, whether inhibitory or excitatory, is fixed, but the intensity of the capacity-alteration it creates can vary. This effect, termed synaptic plasticity, allows the network to gain from previous practice.

Bio-neurological research has revealed the changes occurring in synapses over time (Maass et al., 2002; Maass & Zador, 1999). However, this knowledge is gleaned from remote neurons rather than the bigger network. How biological neural networks interact to learn like the human brain remains poorly understood. Among the numerous types of neurons and synapses, certain ones possess extremely long axons which can exert long-range effects. Other neurons are adapted to regional processing, possessing both a small axon and small dendrites. Yet others develop inhibitory or excitatory synapses, but not both. Axons do not establish synapses with dendrites at all times. Certain synapse with the cell body of another neuron, enabling them to exert a strong effect. Some neurons establish inhibitory synapses to other axons, which prohibits those axons from spreading their spikes. Biological neural networks always constitute a conglomerate of different neurons. No fragment in the human brain contains a homogenous pool of a single neuron type. In addition, biological neural networks are extremely recurrent; that is, numerous loops exist within the network assist positive and negative responses (Maass, Natschläger, & Markram, 2004). Obviously, a solitary biological neuron is part of a very complicated vibrant system. It is extremely difficult to copy the features and behaviour of a neuron in its entirety.

2.1.2 Neural Models

This section discusses some current models of biological neural activity. It focuses on the sigmoid and the spiking neural models and the differences between them. One model is not essentially superior to another, but models vary widely in their level of abstraction. Some models generate precise incitation of the neuron, incorporating all dissimilar biochemical information (Koch & Segev, 1989; MacGregor, 1987). These models are generally not aimed at neural network building, but at accurately simulating the behaviour of single neurons. Other models are more conceptual and depict the condition of a neuron by a real number, termed its activation, in the absence of any molecular consideration (Rosenblatt, 1962; Rumelhart & McClelland, 1986). From these types of models, it is simpler to create a network and to deduce a learning algorithm for it. The most famous neural-model is the sigmoid unit (Rumelhart & McClelland, 1986), depicted in Figure 2.2. In this model, the output or stimulation of a neuron is assigned a single variable, generally between 0 and 1. The synapse bridging the two neurons is modelled by a weight variable describing the magnitude of the influence on the postsynaptic neuron. These weights may be positive (denoting an excitatory synapse) or negative (inhibitory synapse). The capacity of the sigmoid neuron is obtained by summing the weighted ejecting rates of its presynaptic neurons. From this potential, the activation is calculated by an activation function.

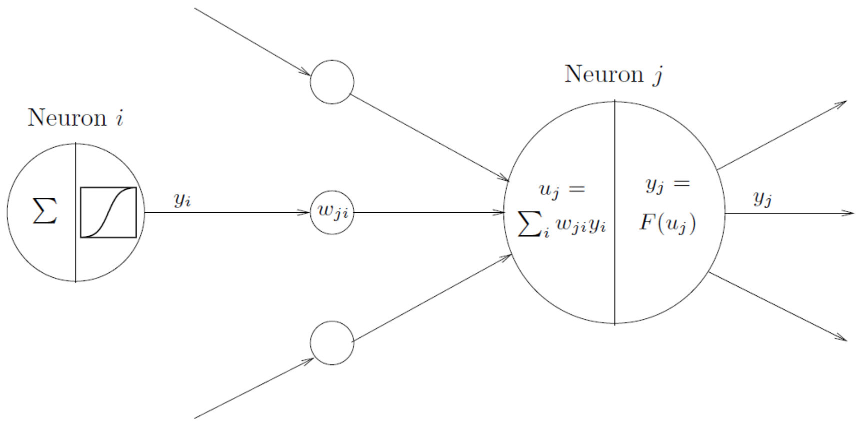


Figure 2.2: Sigmoid Unit (Neural Model)

The primary functioning steps of a sigmoid unit are addition of its input and its stimulation. The productivity of neuron i is weighted by the synapse linking neuron i to its adjoining neuron j, determined as the weighted input $w_{ji}y_i$. The capacity u_i is obtained by summing all j neuron inputs; then the activation y_i is computed from the sigmoid function F(·). The sigmoid neuron is named after the sigmoidal form of its stimulation function, as shown in Figure 2.2 for neuron i. The stimulation variable in this model is the rate at which the neuron ejects its spikes; i.e., the number of spikes in a specific time window. This has long been considered as the only information shared between two biological neurons (Fred Rieke, Warland, Rob, & Bialek, 1997). The neural code of the neural-network is known as the firing rate. Researchers argued that the firing rate cannot be the sole neural code. Psychological experiments have shown that some neural processing is too quick to be modelled by this type of calculation (Thorpe, Delorme, & Van Rullen, 2001).

Real neurons could not have computed the average number of spikes in such a short time period. Numerous neurobiological studies have determined another type of neural-code (Fred Rieke et al., 1997); accurately timed spikes. In this scenario, information passed from neuron to neuron is not encoded in the ejecting rate of the spikes, but in their accurate timing, leading to high speed neural processing. Spiking neural networks (SNN) (also pulse-coupled or integrate-and-fire networks) are comprehensive models which utilize this neural-code of accurately timed spikes (W. Gerstner & Kistler, 2002). The input and output of a spiking neuron is exhibited by a sequence of ejecting times known as a spike-train. A spiking train is illustrated diagrammatically in Figure 2.3. The ejecting time is depicted by the vertical bars.

A single ejecting time is the time at which a neuron has released a pulse. Additional pulses of the same form are ignored, since the pulses emitted by a specific type of neuron appear similar. The capacity of a spiking neuron is expressed as an active variable and operates as a leaky integrator of the receiving spikes: newer spikes exert stronger influence over the potential than the older spikes. If this addition exceeds a predefined threshold, the neuron ejects a spike. SNN also accounts for the refractory interlude and synaptic interval. Consequently, an SNN is a dynamic system, unlike sigmoid neural networks, enabling time-dependent calculations in a very natural manner.

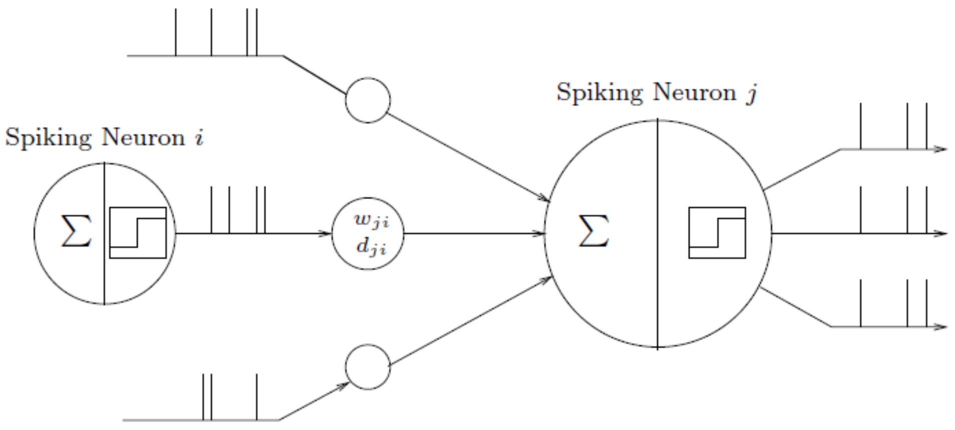


Figure 2.3: Schematic of Spiking Neurons

2.1.3 Hodgkin Huxley model

Most of the SNN models have been well explained by Gerstner and Kistler (2002). The Hodgkin Huxley model, introduced by Hodgkin and Huxley in 1952, is based on their experiment on squid giant axons. They discovered the presence of three ion channels in a neuron; sodium, potassium and a leakage channel. The Hodgkin Huxley model is a complex model which simulates the role of ionic mechanisms in calculation and propagation of the potential in the neurons.

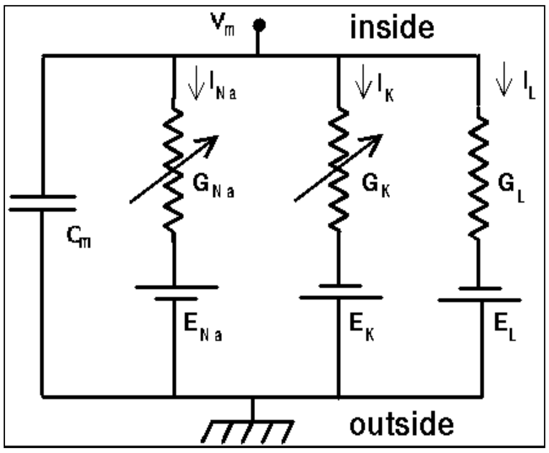


Figure 2.4: Electrical Circuit of the Hodgkin- Huxley model (Gerstner & Kistler, 2002)

The membrane potential I_{ion} is calculated in the standard Hodgkin - Huxley model as:

$$\sum I_{ion} = G_{NA} \times m^3 \times h \times (V_M - E_{NA}) + G_K \times n^4 \times (V_M - E_K) + G_L \times (V_M - E_L) \quad (2.1)$$

$$\frac{dm}{dt} = \alpha_m(v) \times (1-m) - \beta_m(V) \times m \quad (2.2)$$

$$\frac{dh}{dt} = \alpha_h(v) \times (1-h) - \beta_h(V) \times h \quad (2.3)$$

$$\frac{dn}{dt} = \alpha_n(v) \times (1-n) - \beta_n(V) \times n \quad (2.4)$$

where G_{NA}, G_K and G_L denote sodium, potassium and a leakage channel respectively, while E_{NA}, E_K and E_L are constants called reversed potentials. Blockage of a channel is controlled by the additional variables m and n for the N_a channel and h for the K channel. α and β are empirical functions of V_m chosen to fit the data of the huge squid axon. Although this model is commonly used to estimate the parameters of a neural ionic channel, it also has some disadvantages resulting from the approximations required.

2.1.4 Izhikevich model

In 2003, Izhikevich proposed a simple spiking model which combines the biological credibility of the Hodgkin-Huxley model with the computational competency of integrate and fire models (Izhikevich, 2003). The Izhikevich model defines four parameters (*a*, *b*, *c*, *d*) which reproduce spike bursting behaviour. A further two variables (v and u) represent respectively the membrane capacity (post synaptic potential: PSP) and reset potential, which accounts for the potassium ion activation and sodium ion deactivation. This model encapsulates many biophysically accurate Hodgkin–Huxley-type neural models in the following formula:

$$\frac{dv}{dt} = 0.04 \times v^2 + 5 \times v + 140 - u + I \text{ where } \frac{du}{dt} = a \times (b \times v - u) \qquad (2.5)$$

Reset potential after-spike:

$$if \quad v \geq 30mV \quad then \quad \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{2.6}$$

After the spike reaches the threshold (30 mV), the membrane voltage (v) and the recovery variable (u) are rearranged according to equation (2.6).

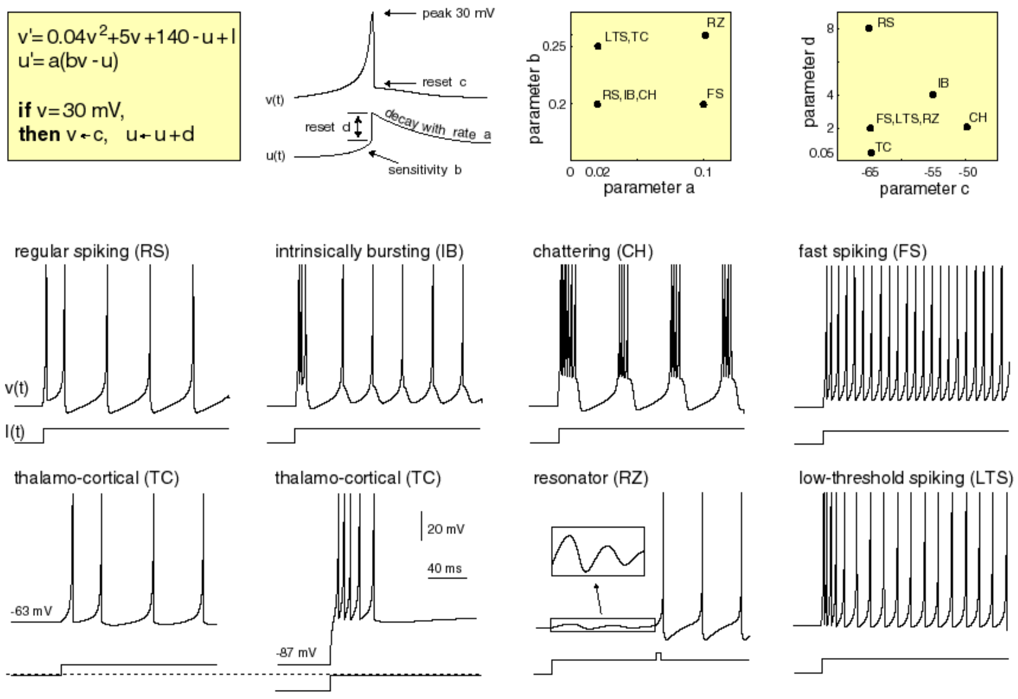


Figure 2.5: The Dynamics of the Izhikevich Model (Izhikevich, 2003)

The equation describing membrane potential dynamics, $\frac{dv}{dt} = 0.04 \times v^2 + 5 \times v + 140 - u + I$, is sometimes regarded as a quadratic integrate. It governs the firing of a neuron and is activated by installing the spike, thus commencing the dynamics of a cortical neuron. The reset u is between -60 and -70 mV depending on the value of *b*. In imitation of biological neurons, the threshold is not fixed, but is allowed to vary between -55 and -40 mV. The parameter a (typically 0.02) sets the time scale of u, b (also 0.02) describes the sensitivity of u to v, while c and d influence the value of v after spiking by altering the settings of the factors *a*, *b*, *c* and *d*, different neural characteristics can be modelled.

2.1.5 Leaky Integrate and Fire Model (LIF)

The Leaky Integrate and Fire (LIF) model is also comprehensively described in Gerstner and Kistler (2002). In this model, a neuron is considered as an electrical circuit and the current potential is calculated by an appropriate equation. Conceptually, the LIF model is typified by an electrical circuit as shown in Figure 2.6, comprising a capacitor *C* in parallel with a resistor *R* through which a current *I* (*t*) flows. The current *I* (*t*) splits into two components, I_R *and* I_C. The I_R flows through the linear resistor *R* while the current I_C charges the capacitor *C*. I_R is computed from Ohm's law as $I_R = u/R$. $I_C = q/u$ (where *q* is the charge and *u* the voltage), The current across the capacitor increases with time as $I_C = C\frac{du}{dt}$.

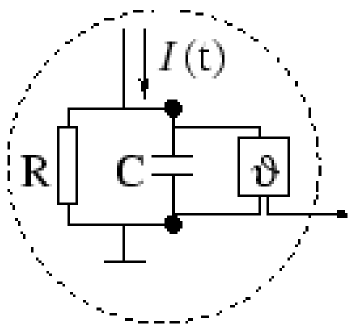


Figure 2.6: Leaky Integrate and Fire Model (Gerstner & Kistler, 2002)

The current $I(t)$ can be divided into two components, $I(t) = I_C + I_R$. I_C charges the capacitor C and I_R passes through the resistor R. Using the Ohm's law, $C = q/u$ where q is the charge, u is the voltage and $I_R = u/R$, the capacitive current $I_C = C\frac{du}{dt}$, therefore:

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

Multiplying equation (2.7) by R and defining the time constant of the leaky integrator as $T_m = RC$, we obtain the formulation of the LIF model:

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

u refers to the membrane capacity and T_m is the time constant of the neural membrane. When the membrane capacity reaches the firing threshold, the neuron spikes and the membrane potential is reset to its resting potential. The LIF and Izhikevich models differ in their treatment of the threshold; in the former it is fixed whereas in the latter it fluctuates. LIF is more computationally efficient than Izhikevich, however, because it contains fewer biological parameters. Since LIF is simple as well as computationally efficient, it can be applied to large networks.

2.1.6 Probabilistic Spiking Neural Model (pSNM)

This model, suggested by Kasabov (Kasabov, 2010), is diagrammatically presented in Figure 2.7. In pSNM, a neuron (n_j) receives input spikes from a pre-synaptic neuron $n_i(i=1,2,\ldots,m)$. The state of the neuron n_j is the sum of the inputs received from each of the m synapses – the postsynaptic potential, PSP$j(t)$. When PSP$j(t)$ reaches the firing threshold $\vartheta j(t)$, neuron n_j fires, i.e. releases a spike.

The linked synapses are associated with connection weights ($w_{i,j}$, $i=1,2,...,m$), formed during learning from Thorpe's rule: $\Delta w_{i,j} = \mathrm{mod}^{order(i)}$, where *mod* is a modulation aspect (a constraint value between 0 and 1) and *order(i)* is the sequence in which the spike from neuron n_i reaches the synapse $s_{i,j}$ relative to the spike entrance from other neurons, after neuron n_j has ejected a spike. Thorpe's rule is a quick learning rule that requires data to be broadcasted once only.

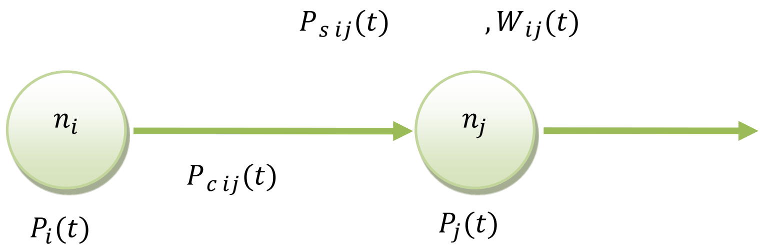


Figure 2.7: Probabilistic Spiking Neural Model

Supplementary to the connection weights $w_{i,j}$ (t), the pSNM has three new probabilistic constraints. First is a probability parameter $p_{ci,j}(t)$ denoting the likelihood that a spike ejected from neuron n_i will reach neuron n_j at a time t at the synapse $s_{i,j}$ through connecting the two neurons. The probability constraint models the structural and functional integrity of each neural connection. If $p_{ci,j}(t)$=0, no connection exists and no spike is broadcast.

Secondly, a probability parameter $p_{si,j}(t)$ is added to the PSP$j(t)$ of synapse $s_{i,j}$ once the synapse has obtained a spike from neuron n_i. In future investigations, we will assume that the default state of $p_{si,j}$ =1 (i=1,..,m). The final probability parameter, $p_j(t)$, denotes the probability that neuron n_j t outputs a spike at time t , once the total PSP$_j(t)$ has attained a value close to the PSP threshold. The PSP$j(t)$ is now calculated as:

$$PSP_j(t) = \sum_{p=t_0}^{t} \sum_{i=1}^{m} e_i g\left(p_{s_{i,j}}(t-p)\right) f\left(p_{s_{i,j}}(t-p)\right) w_{i,j}(t) + \eta(t-t_0) \quad (2.9)$$

where e_i is 1 if a spike has been ejected from neuron $n_{i,}$ and 0 otherwise; $g(p_{ci,j}(t))$ is 1 with probability $p_{ci,j}(t)$, and 0 otherwise; $f(p_{si.j}(t))$ is 1 with probability $p_{si,j}(t)$, and 0 otherwise; t_0 is the time of the last spike ejected by n_j; $\eta(t-t_0)$ is a supplementary expression representing degeneration in the PSP. The pSNM is simplified when all or few of the probability parameters are set to "1".

Other PSNM models proposed in terms of probabilistic dynamic synaptic weighting include the Maass Model (Maass et al., 2002), which incorporates a parameter equivalent to $p_{si,j}(t)$ in Kasabov's model.

2.2 Coding of information for neuro-computation

This section addresses a fundamental question in Neuroscience, the code used by neurons to transfer information. Is it possible for an external observer to read and understand neural activity? Traditionally, there are two main theories of neural encoding – pulse codes and rate codes. Both theories are discussed below.

2.2.1 Pulse codes

The first type of neural encoding is referred to as a spike or pulse code. These codes assume the precise spike time as the carrier of information between neurons. Evidence for temporal correlations between spikes has been shown through computer simulations (see, for example, (Legenstein, Naeger, & Maass, 2005) using integrate-and-fire models, as well as through biological experiments, such as electrophysiological recordings and staining procedures (Nawrot, Schnepel, Aertsen, & Boucsein, 2009), and the *in vivo* measurements described in (Villa, Tetko, Hyland, & Najem, 1999), in which spatio-temporal patterns of neural activity are used to predict the behavioural responses of rats.

A pulse code based on the timing of the first spike following a reference signal is discussed (Simon Thorpe & Jacques Gautrais, 1998). This encoding is known as time-to-first-spike. It was argued that each neuron has time to emit only a few spikes that can contribute to the overall processing of a stimulus. It was further shown (Thorpe et al., 2001) that a new stimulus is processed within 20 to 50ms of its onset. Thus, earlier spikes carry most of the information contained in the stimulus. The Thorpe model, which emphasises the importance of early spikes, has been discussed in section 2.3.4.1.

Other pulse codes consider correlation and synchrony to be important. Neurons that represent a similar concept, object or label are “labelled” by firing synchronously (von der Malsburg, 1983). More generally, any accurate spatiotemporal pulse prototype is potentially significant and may encode particular information. Neurons that fire with a certain relative time delay may signify a certain stimulus. This concept is central to the so-called rank order population encoding presented in section 2.6.1. Additional information on neural encoding can be found in the book (Fred Rieke et al., 1997).

2.2.2 Rate codes

The second theory assumes that the mean ejecting rate of a neuron carries most, if not all, of the transmitted information. These codes are referred to as rate codes and have inspired the classical perceptron approaches. The mean firing rate v is usually taken as the ratio of the average number of spikes n_{sp} observed over a specific time interval T and T itself:

$$v = \frac{n_{sp}}{T} \tag{2.10}$$

This concept has been especially successful in the context of sensory or motor neural systems. In a pioneering study, Adrian found a direct connection between the ejecting rate of stretch receptor neurons and the applied force in the muscles of frog legs (Adrian, 1926). Nevertheless, the idea of a mean firing rate has been repeatedly criticised (F. Rieke, Warland, van Steveninck, & Bialek, 1999). The main argument is the comparably slow transmission of information from one neuron to another, since each neuron must integrate the spike activity of pre-synaptic neurons at least over time T. Especially, the extremely short response times of the brain for certain stimuli cannot be explained by the temporal averaging of spikes. For example, Thorpe et al. (1996) reported that the human brain can recognise a visual stimulus in approximately 150ms. Since a moderate number of neural layers are involved in the processing of visual stimuli, if every layer had to wait a period T to receive the information from the previous layer, the recognition time would be much extended.

An alternative interpretation defines the mean firing rate as the average spike activity over a population of neurons. The principle of this interpretation is explained in Figure 2.8. A post-synaptic neuron receives stimulating inputs in the form of spikes emitted by a population of pre-synaptic neurons. This population produces a spike activity A, defined as the fraction of neurons being active within a short interval $[t, t + \Delta t]$ divided by the population size N and the time period Δt (W. Gerstner & Kistler, 2002).

$$A = \frac{1}{\Delta t} \times \frac{n_{act}(t, t+\Delta t)}{N} \tag{2.11}$$

A neuron attains input spikes from a population of pre-synaptic neurons producing a certain activity A. The activity is defined as the fraction of neurons being active within a short interval$[t, t + \Delta t]$, divided by the population size N and the time period Δt (W. Gerstner & Kistler, 2002).

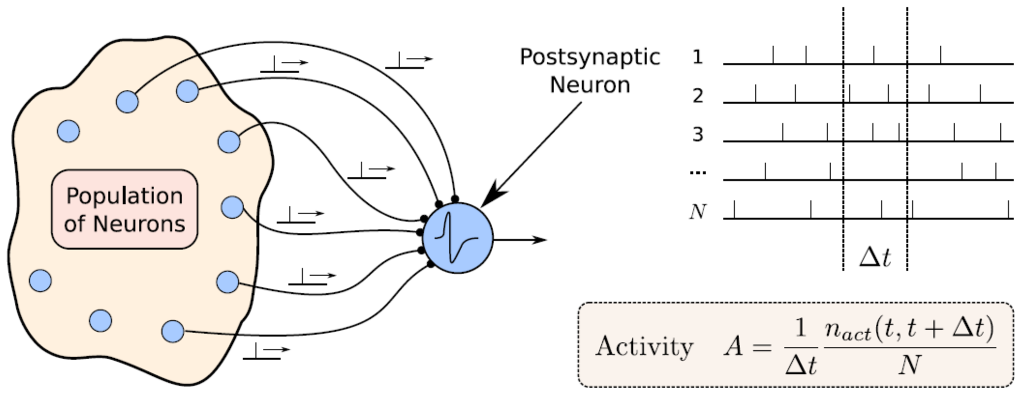

Population of Neurons
Postsynaptic Neuron
1
2
3
...
N
Δt
Activity $A = \frac{1}{\Delta t} \frac{n_{act}(t, t+\Delta t)}{N}$

Figure 2.8: Pre-synaptic Neurons (Gerstner &Kistler, 2002)

Here, $n_{act}(t, t + \Delta t)$ denotes the number of active neurons in interval $[t, t + \Delta t]$ and N is the total number of neurons in the population. The activity of a population may vary rapidly, enabling fast responses of the neurons to changing stimuli (Wulfram Gerstner, 2000) and (Brunel, Chance, Fourcaud, & Abbott, 2001).

2.3 Learning Rules

This section presents some typical learning methods including spiking neural network architectures, which are related to particular learning rules, in the context of spiking neurons. Diverse problems weaken the development of learning procedures for SNN. The precise time reliance causes asynchronous information inputs that often require complicated software and/or hardware applications before the neural network can function. The repeated network topologies typically used in SNN preclude the creation of an uncomplicated learning method such as back-propagation using MLP. Similar to traditional neural networks, three different learning paradigms can be distinguished in SNN, namely, unsupervised, reinforcement and supervised learning. Reinforcement learning in SNN is probably the least common of the three. Some algorithms have been successfully applied to robotics (R. V. Florian, 2005), as well as being theoretically analysed (Z. V. Florian, 2007), (Seung & Hughes, 2003) and (Xie & Seung, 2003). Unsupervised learning in the form of Hebbian learning is the most biologically realistic learning scenario (Cooper, 2005). The so-called Spike-Timing Dependent Plasticity (STDP) belongs to this category and is discussed in the next section.

Supervised techniques impose a certain input-output mapping on the network which is essential in SNN practical applications. Two supervised learning methods are discussed in greater detail in following sections. The learning algorithm employed in the eSNN architecture is discussed separately in section 2.6.2. An excellent comparison between supervised learning methods developed for SNN can be found in (Ponulak & Kasiski, 2010).

2.3.1 Spike-Timing Dependent Plasticity (STDP)

The concept of spike-timing dependent plasticity was inspired by the experiments of Donald O. Hebb, published in his famous book “The Organisation of Behaviour” (Hebb, 1949). His essential postulate is often referred to as Hebb’s Law:

“When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A’s efficiency, as one of the cells firing B, is increased.”

The first experimental evidence of Hebb’s postulate was published twenty years later in (T. V. Bliss & Lomo, 1970) and (T. Bliss & Lomo, 1973). Today, the change of synaptic efficacy in the brain is known to be correlated with the timing of pre- and post-synaptic activity of a neuron (Bell, Han, Sugawara, & Grant, 1999; Bi & Poo, 2001; Markram, Lübke, Frotscher, & Sakmann, 1997).

Reinforcement or decline in synaptic efficacy is known as long-term potentiation (LTP) or long-term depression (LTD), respectively. STDP is described by a function $W(t_{pre} - t_{post})$ that determines the fractional change of the synaptic weight in terms of the difference between the arrival time $t_{pre}$of a pre-synaptic spike and the time t_{post} of an activity emitted by the neuron. Function W is also known as the STDP window. W is typically expressed as:

$$W(t_{pre} - t_{post}) = \begin{cases} \mathrm{A}_{+} \exp\left(\frac{\mathrm{t_{pre}} - \mathrm{t_{post}}}{\tau_{+}}\right) & if\ \ t_{pre} < t_{post} \\ \mathrm{A}_{-} \exp\left(-\frac{\mathrm{t_{pre}} - \mathrm{t_{post}}}{\tau_{-}}\right) & if\ \ t_{pre} > t_{post} \end{cases} \tag{2.12}$$

where parameters τ_{+} and τ_{-}delineate the temporal range of the pre- and postsynaptic time interval, while A_{+} and A_{-}denote the maximum fractions of synaptic modification, at $t_{pre} - t_{post}$close to zero. Figure 2.9 presents the STDP window W generated by Equation 2.12.

The parameters A_{+}, A_{-}, τ_{+} and τ_{-} can be adjusted to suit the neuron of interest. The window W is usually temporally asymmetric ($A_{+} \neq A_{-}$and $\tau_{+} \neq \tau_{-}$), but exceptions exist. For instance, the synapses of layer 4 spiny-stellate neurons in the rat barrel cortex appear to have a symmetric window (Egger et al., 1999).

The dynamics of synaptic pruning consequential to the STDP learning rule have been investigated (J. Iglesias, Eriksson, Grize, Tomassini, & Villa, 2006).

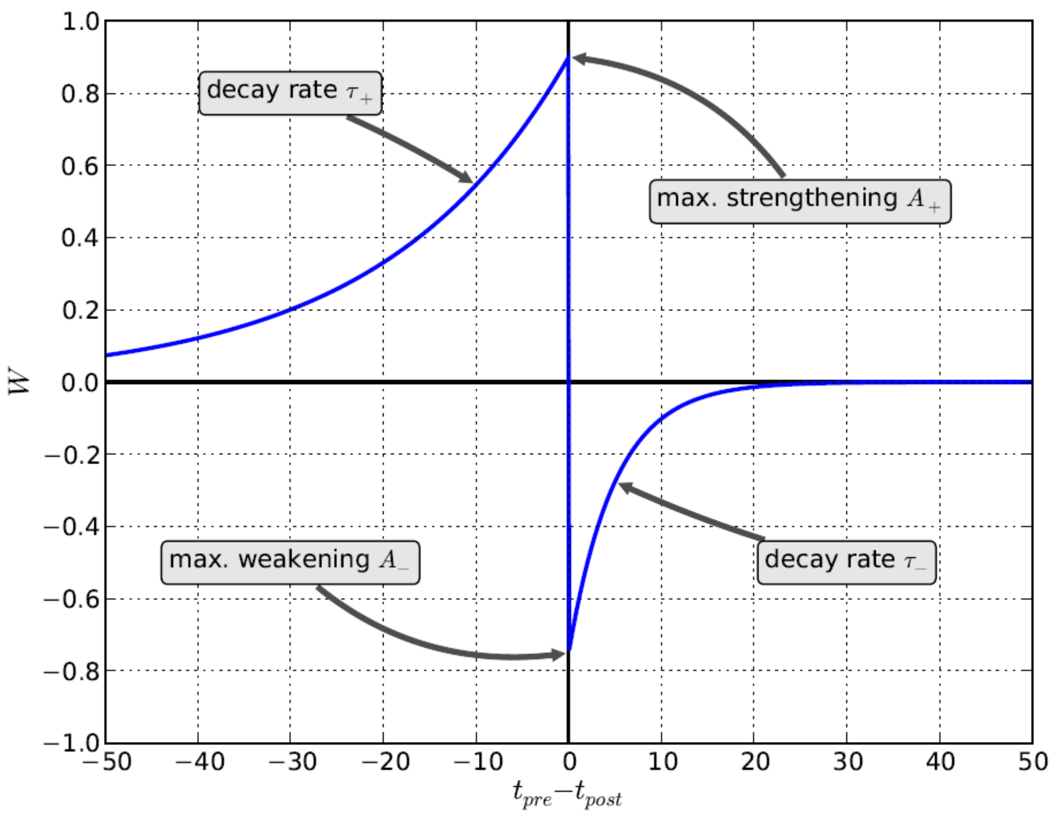


Figure 2.9: STDP Learning Window W (Egger, Feldmeyer, & Sakmann, 1999)

Figure 2.9 was generated by setting $A_+ = 0.9$, $A_- = 0.75$, $\tau_+ = 20$, and $\tau_- = 5$ in Equation 2.12.

Synaptic pruning is a generic feature of mammalian brain maturation, in which the embryonic nervous system is refined by removal of inappropriate synaptic connections between neurons, while preserving appropriate ones. Later studies extended this work by bringing apoptosis (genetically programmed cell death) into the analysis. The emergence of cell assemblies has been verified by identification of spatio-temporal patterns in the pruned network (Javier Iglesias &

Villa, 2006). More information on STDP can be found in the excellent review by (Bi & Poo, 2001) and in (Wulfram Gerstner, 2000; W. Gerstner & Kistler, 2002; Kempter, Gerstner, & van Hemmen, 1999).

2.3.2 Spike Back-Propagation (Spike-Prop)

Traditional neural networks, like the multi-layer perceptron, usually adopt some form of gradient based descent, namely error back-propagation, to modify synaptic weights. This action results in a particular input-output representation of the network. However, the topological recurrence of SNNs and their explicit time dependence allow no straightforward evaluation of the gradient in the network. Special assumptions are required before back-propagation can be applied to spiking neurons.

In (Bohte et al., 2002; Bohtem, Poutre, & Kok, 2000) a back-propagation algorithm called Spike-Prop is proposed, which is suitable for training SNN. Under this method, SNN learns a set of desired firing times t_j^d of all output neurons j for a given input pattern. Spike-Prop minimises the error E, defined as the squared difference between all network output times tout$j$ and the desired output time t_j^d.

$$E = \frac{1}{2}\sum_j \left(t_j^{out} - t_j^d\right)^2 \tag{2.13}$$

with respect to the weights w_{ij}^{k} of each synaptic input:

$$\Delta w_{ij}^{k} = -\eta \frac{dE}{dw_{ij}^{k}} \qquad (2.14)$$

With η defining the learning rate of the update step.

A limitation of the algorithm is that, like the Thorpe neural model presented in section 2.3.5.1, each neuron is allowed to fire once only. Consequently, the error function defined in Equation 2.13 depends entirely on the difference between actual and desired spike times, so that Spike-Prop is suitable only for time-to-first-spike encoding.

2.3.3 Remote Supervised Method (ReSuMe)

Here we discuss the Remote Supervised Method (ReSuMe) introduced in 2010 by Ponulak & Kasiski. ReSuMe aims to enforce a required input-output spike pattern on a SNN, *i.e.* to produce target spike trains in reaction to a given input stimulus. This approach is based on the STDP learning windows presented in section 2.3.1. The synaptic weights are balanced by two opposite update rules. Additional teacher neurons, which remotely supervise the evolution of the synaptic weight, are assigned to each synapse. The teacher neuron is not explicitly connected to the network, but generates a reference spike signal by which connection weight is updated in a STDP-like fashion. The post-synaptic neuron, whose activity is influenced by the weight update, is termed the learning neuron.

The synaptic change depends on the correlation of spike activities between input, teaching and learning neurons. Figure 2.10 illustrates the ReSuMe principle of organized learning in spiking neural networks (Ponulak & Kasiski, 2010),which regards spiking neural networks as a readout function of LSM. Let n_i^ldenote the learning neuron receiving spike sequences from a pre-synaptic neuron $n_k^{in}(i)$ with corresponding synaptic weight w_{ki}, and let neuron $n_d(i)$ be the teacher for weight w_{ki}. If input neuron $n_k^{in}(i)$ emits a spike which is followed by a spike of the teacher neuron $n_d(i)$, the synaptic weight w_{ki} is increased. On the other hand, if $n_k^{in}(i)$ spikes before the learning neuron n_i^l is activated, the synaptic weight is decreased. The amplitude of the synaptic change is determined by two functions $w^d(s^d)$ and $w^l(s^l)$, where $s^d$is the temporal variance between the spike times of teacher neuron and input neuron, while s^ldescribes the difference between the spike times of learning neuron and input neuron. Thus, the precise time difference of spiking activity defines the strength of the synaptic change.

Figure 2.10 illustrates the principle of organized learning in spiking neural networks, namely ReSuMe (Ponulak & Kasiski, 2010) which is an example of spiking neural networks as a readout function of LSM. Let n_i^ldenote the learning neuron which receives spike sequences from pre-synaptic neuron $n_k^{in}(i)$, the correspondingsynaptic weight being $w_{ki}$and neuron $n_d(i)$ being the teacher for weight w_{ki}. If input neuron $n_k^{in}(i)$ emits a spike which is followed by a spike of the teacher neuron $n_d(i)$, the synaptic weight $w_{ki}$is increased. On the other hand, if $n_k^{in}(i)$spikes before the learning neuron n_i^l is activated, the synaptic weight is decreased. The amplitude of the synaptic change is determined by two functions $w^d(s^d)$ and $w^l(s^l)$, where $s^d$is the temporal variance between the spike times of teacher neuron and input neuron, while s^ldescribes the difference between the spike times of learning neuron and input neuron. Thus, the precise time difference of spiking activity defines the strength of the synaptic change.

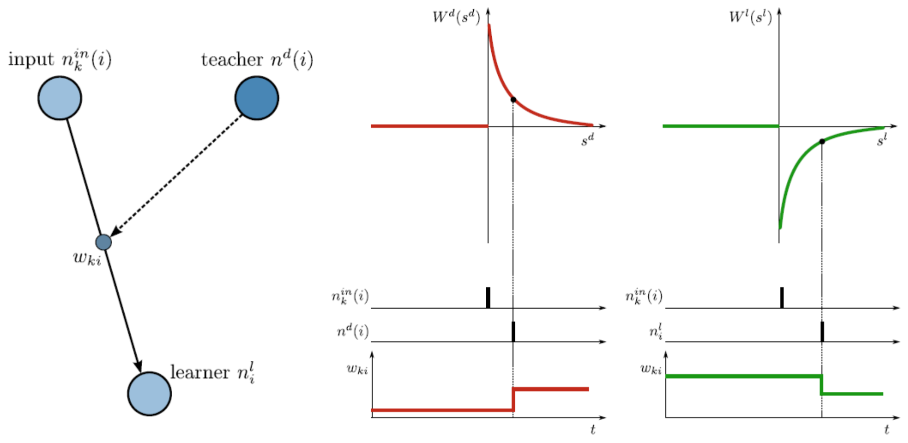


Figure 2.10: The Remote Supervised Method (ReSuMe) Approach (Ponulak & Kasiski, 2010)

2.3.4 The Fusi's spike driven synaptic plasticity (SDSP) learning rule

The SDSP, a modification of STDP (Song, Miller, Abbott, & others, 2000) which has been described as an unsupervised learning method (Fusi, Annunziato, Badoni, Salamon, & Amit, 2000). SDSP models the synaptic plasticity V_{w0} of a synapse w_0 contingent at the time of spiking of the pre-synaptic and post-synaptic neurons. V_{w0} rises or reduces depending on the comparative timing of the pre and post synaptic spikes.

When a pre-synaptic spike reaches the synaptic station before a postsynaptic spike within a specified time window, the synaptic efficiency is enhanced. If the post-synaptic spike is ejected immediately after the pre-synaptic spike, synaptic efficiency is reduced. This plastic synaptic efficiency is embodied in the following equations:

$$\Delta V_{w0} = \frac{I_{pot}(t_{post})}{C_p} \Delta t_{spk} \text{ if } t_{pre} < t_{post} \tag{2.15}$$

$$\Delta V_{w0} = -\frac{I_{dep}(t_{post})}{C_d} \Delta t_{spk} \text{ if } t_{post} < t_{pre} \tag{2.16}$$

where Δt_{spk} is the pre and post synaptic spike time window.

The SDSP rule can be applied to a supervised learning algorithm, when a trainer signal imitating the required output spiking order is inserted together with the training spike pattern, but without any alteration of the trainer input weights. In a study by Brader and colleagues (2007), a SDSP-driven SNN learned to recognise

293 characters (classes). Each character (a static image) was encoded as a 2000 bit feature vector, and each bit was transformed into a spike rate, with binary values 0 and 1 corresponding to 0 Hz and 50 Hz, respectively (Brader et al., 2007). For each class, 20 distinct training designs were utilized and 20 neurons were assigned, one for each pattern, and trained for hundreds of reiterations. The SDSP model is implemented in the INI analogue SNN silicon chip (Indiveri et al., 2011). The silicon synapses are composed of bi-stability circuits that assign a synaptic weight to one of two likely analogue values (either increased or decreased). These circuits push the synaptic weight voltage by a positive or negative current superimposed on that produced by the STDP. Within a short time frame, 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 activity. In the absence of a current (and also in the learning phase) this background current will drive the weight toward its potentiated state. If the STDP decreases the synaptic weight below the threshold, the bi-stability circuits generate a negative current that inhibits spiking activity and vigorously drives the weight toward its analogue value. The synapse is now in a depressed state. The STDP and bi-stability circuits simplify the simulation of both long term and short term memory.

Although SDSP is effective at identifying generally static patterns, the potential of the SDSP SNN model and its accompanying hardware has not been completely investigated in spatiotemporal pattern recognition, particularly in fast on-line learning.

2.3.5 Evolving Spiking Neural Networks

In this section, the encoding principle used in the eSNN and Thope models (Thorpe et al., 2001) are presented, followed by a description of the one-pass learning method and the overall functioning of the eSNN method.

2.3.5.1 Thorpe model

The Thorpe model is a simplified Integrate and Fire spiking neural model (Thorpe et al., 2001). The model adopts the LIF concept but simplifies the leaky operation of the computational neuron. The potential of a given neuron is either disabled or reset to the level at which it fires a spike. In this model, each neuron is allowed to fire once before being disabled. Because of its low computational cost and efficiency, this model has become the most widely used (Kasabov 2007; Schliebs, Defoin-Platel, Worner, & Kasabov, 2009; Wysoski, Benuskova, & Kasabov, 2010).

2.3.5.2 Population encoding

Population Coding (POC) is a well-known -spike encoding technique initially proposed by Thorpe and Gautrais (S. Thorpe & J. Gautrais, 1998), and fully proposed by Bohtem (Bohtem et al., 2000). A single input is distributed to multiple input neurons. In POC, each input neuron is associated with a particular spike time. The firing time of input neuron *i* is computed by engaging conjoint Gaussian functions in which the centre and the width are calculated from Equations 2.17 and 2.18 respectively, within the variable interlude [$I_{\min}$, $I_{\max}$]. The width of each Gaussian receptive field is controlled by the parameter β. The POC process, proposed in 2009 (Schliebs et al., 2009), is illustrated in Figure 2.11.

$$\mu = I_{\min} + (2 * i - 3)/2 * (I_{\max} - I_{\min})/(M - 2) \tag{2.17}$$

$$\sigma = 1/\beta(I_{\max} - I_{\min})/(M - 2)\ where\ 1 \le \beta \le 2 \tag{2.18}$$

For an input value of 0.75, (thick vertical line in top figure) the intersections with each Gaussian are calculated (triangles), which are then interpreted into spike time intervals.(Schliebs et al., 2009).

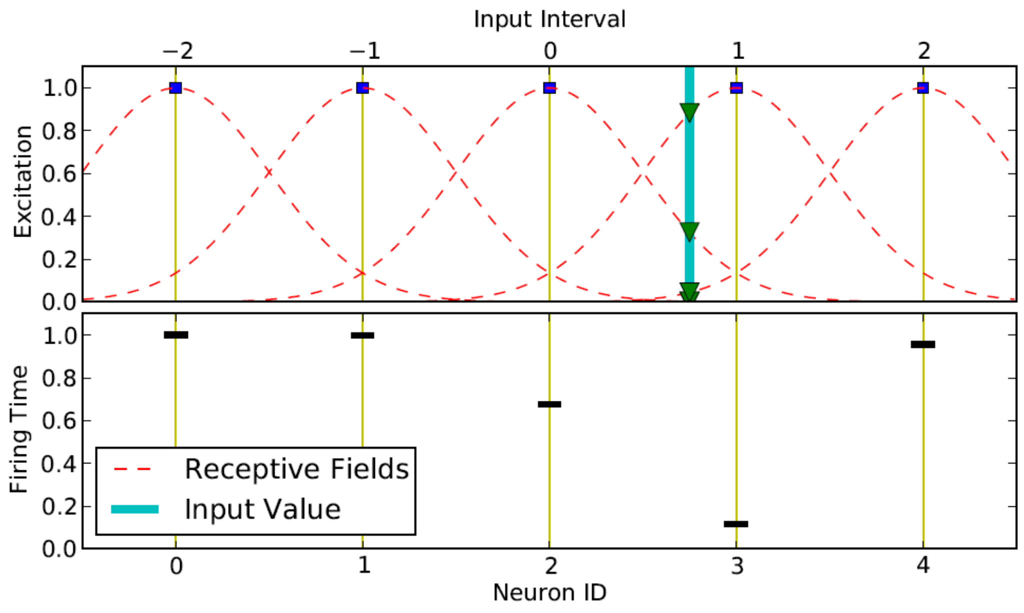


Figure 2.11: Population Encoding Based on Gaussian Receptive Fields (Schliebs et al., 2009)

2.3.5.3 One-Pass Learning

The aim of this learning method is to create output neurons, each labelled with a certain class label $l \in L$. The number and value of class labels depends on the classification problem of interest. L denotes the set of class labels of the given data set. Following presentation of a specified input sample to the network, the corresponding spike train is propagated through the SNN, triggering a subset of output neurons to fire. It is possible that no output neuron is activated and the network remains silent. In this case, the classification result is undetermined. If one or more output neurons have emitted a spike, the neuron with the shortest response time (earliest spike time) determines the classification. The label of this neuron is then the classification result for the presented input sample.

During training, the learning algorithm consecutively forms a repository of proficient output neurons. For every class label $l \in L$ an individual repository is evolved. For each training sample i with class label $l \in L$, a new output neuron is created and is fully linked to the previous layer of neurons, resulting in a real-valued weight vector $w^{(i)}$, with $\mathrm{w}_{\mathrm{j}}^{(\mathrm{i})} \in \mathrm{R}$ denoting the connection between the pre-synaptic neuron j and the created neuron i. In the next step, the input spikes are propagated through the network and the value of weight $\mathrm{w}_{\mathrm{j}}^{(\mathrm{i})}$ is computed according to the *order* of spike transmission through a synapse j:

$$w_j^{(i)} = (m_l)^{order(j)} \; \forall j \mid j \; pre-synaptic \; neuron \; of \; i \tag{2.19}$$

The parameter m_l is the modulation factor of the Thorpe neural model. Differently labelled output neurons may have different modulation factors m_l. The function $order(j)$ reflects the position of the spike produced by neuron j. For instance, $order(j)$ is assigned a rank of zero if neuron j is the foremost arrival of all spiking pre-synaptic neurons of i. In the same way, the spikes of all pre-synaptic neurons are positioned prior to weighting.

The firing threshold $\vartheta^{(i)}$ of the created neuron i is defined as some fraction $c_l \in R, 0 < c_l < 1$, of the maximal possible potential $u_{max}^{(i)}$:

$$\vartheta^{(i)} = c_l \mathrm{u}_{\mathrm{max}}^{(\mathrm{i})} \tag{2.20}$$

$$u_{max}^{(i)} = \sum_j w_j^{(i)} (m_l)^{order(j)} \tag{2.21}$$

The fraction c_l is a parameter of the model and can be separately specified for each class label $l \in L$. The weight vector of the proficient neuron is then compared to those of neurons that are already in the repository. If the minimal Euclidean distance between the weight vectors of neuron i and an existing neuron k is smaller than a specified similarity threshold s_l, the two neurons are considered too similar to be treated individually, and their firing thresholds and weight vectors are merged according to:

$$w_j^{(k)} \leftarrow \frac{w_j^{(i)}+Nw_j^{(k)}}{\frac{\vartheta^{(i)}+N\vartheta^{(k)}}{1+N}} \text{ , } \forall j, j\ presynaptic\ of\ neuron\ i \tag{2.22}$$

$$\vartheta^{(k)} = \frac{\vartheta^{(i)}+N\vartheta^{(k)}}{1+N} \tag{2.23}$$

Integer N denotes the number of samples previously used to update neuron k. The merging process involves the (continuing) average of the connection weights, and the (continuing) average of the two ejecting thresholds. Following merging, the trained neuron i is discarded and the next sample processed. If no other neuron in the repository is similar to the trained neuron i, that neuron is added to the repository as the latest output neuron.

Figure 2.12 illustrates the architecture of the evolving Spiking Neural Network architecture (eSNN). Real-valued vector elements are transformed into the time domain via rank order population encoding founded on Gaussian receptive fields. As a result of this transformation, input neurons eject spikes at pre-defined firing times, inciting the one-pass learning algorithm of the Spiking Neural Network architecture. The learning iteratively generates a repository of output neurons for each separate class. The developing nature of the network enables accretion of knowledge as it becomes available, without the need to re-train with previously learnt samples.

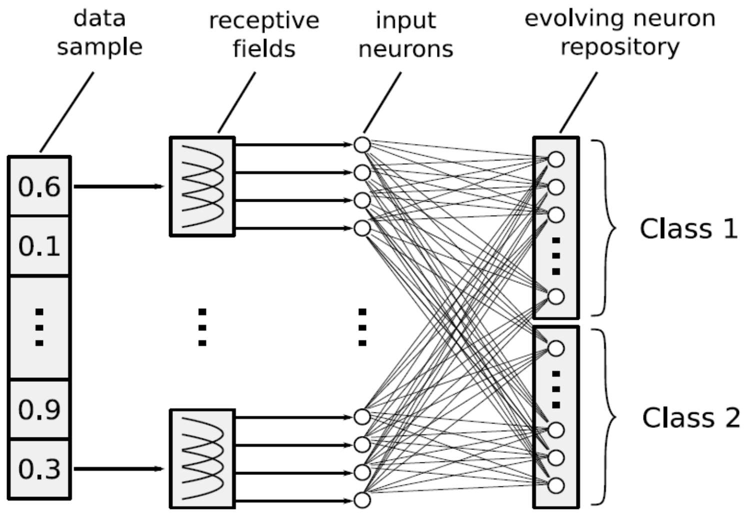


Figure 2.12: Architecture of Evolving Spiking Neural Network architecture (eSNN) (Schliebs et al., 2009)

2.4 Conclusion

The eSNN architecture exhibits a rapid one-pass learning mechanism; however, some further require further consideration.

Firstly, eSNN commonly employs the Thorpe Model (Thorpe et al., 2001), which includes the least number of biological activities and which behaves similarly to a summation function. Hence, satisfactory classification is achieved only when a number of neural and learning parameters have been selected appropriately. Configuring these parameters can quickly become a challenging task, since the influence of each parameter must usually be precisely known. Parameters that are linked to other parameters should not be chosen independently. For example, modifying the modulation factor of the Thorpe neural model should also involve careful choice of the firing threshold. A small modulation factor significantly increases the sensitivity of the neuron to the input, thus the threshold has to be adapted accordingly to prevent the neuron from becoming over-specialised for a certain input. The situation is complicated further when many class labels exist, since the number of parameters increases linearly with number of classes. All of the above–mentioned examples require a careful manual tuning of the eSNN parameters (Schliebs et al., 2009; Wysoski et al., 2010).

Secondly, the importance of rank order coding in eSNN cannot be over-emphasised. Population encoding, in which information is basically encoded vector by vector, increases the number of inputs to eSNN, to the extent that eSNN must process more data than are originally input. Thus, population encoding is not suitable for massive spatio-temporal datasets.

Furthermore, the above–cited studies used standard eSNN, which has limited structure-evolving efficacy and is therefore of reduced functionality during a learning process. New types of SNN have been recently proposed in (Kasabov, 2010), which offer more flexible neural functioning and connectivity. These new algorithms are termed probabilistic SNN (pSNN).

The Spiking Neural Network architecture classifier maps a single data vector to a particular class label. This behaviour is appropriate for the categorizing of time-invariant data. However, most current data volumes are updated continuously, imposing an additional time dimension on the data sets. The categorizing of spatio-temporal patterns remains a major challenge in data mining approaches. Most of the data vectors are successively input to an algorithm which determines the mapping of this succession to a particular class label. In its present form, Spiking Neural Network architecture does not permit categorization of spatial time-based data.

Hence, this study addresses the following issues:

- Exploration and development of new probabilistic spiking neural models (Chapter 3 and research question 1)
- An extension of the new Spiking Neural Network architecture is suggested that permits the technique to acquire spatiotemporal knowledge. A supplementary layer is inserted into the network architecture that converts the spatio-temporal input prototype into a single high-dimensional network condition using the latest reservoir computing paradigm, termed Liquid State Machine (LSM). This intermediate condition is then assigned a required class label by the original one-pass learning algorithm of Spiking Neural Network architecture (Chapters 5 and 6, research question 3).
- Exploration and development of new learning rules, including a novel one-pass learning algorithm for spatio-temporal patterns recognition processing (Chapter 6, research question 2).

- Exploration of a suitable encoding method for spatio-temporal patterns recognition processing, especially human EEG data (Chapter 8, partial of research question 4).
- Evaluation of proposed methods for further development of EEG spatio-temporal patterns recognition processing and brain-computer interface (Chapter 10 and 11, research question 4, 5 and 6).

Chapter 3

Novel Stochastic Spiking Neural Models

This chapter introduces three new Models of Spiking Neural Networks inspired by the probabilistic spiking neural model proposed in 2010 (Kasabov, 2010). Section 3.1 includes relevant background and motivation for developing the models.

Corresponding to the first research question, the contributions involve three novel Stochastic Spiking Neural Models, namely; Noisy Reset (NR), Step-Wise Noisy Threshold (ST) and Continuous Stochastic Threshold (CT), the stochastic models of this chapter also has been shared to international neural network community in 2010 (Nuntalid, Schliebs, & Kasabov, 2010). The details of each stochastic model are presented in sections 3.1.1- 3.1.3. Section 3.2 concludes the chapter.

3.1 Stochastic Spiking Neural Models

Spiking neural models (SNM) differ in their biological parameters and implementation cost. Figure 3.1 illustrates the recreated figure of the quality-cost tradeoff between various SNMs that inspired by Izhikevich in 2003 (Izhikevich, 2003). After 2003 most development in this area has been mainly focused on as the following:

- dynamicity of synapses (Bi & Poo, 2001; Brunel, Chance, Fourcaud, & Abbott, 2001; Gerstner & Kistler, 2002; Indiveri & Horiuchi, 2011; Indiveri et al., 2011; Izhikevich & Edelman, 2008; Kasabov, 2010; Maass, Natschläger, & Markram, 2002; Maass et al., 2004; Maass & Zador, 1999; Markram, Lübke, Frotscher, & Sakmann, 1997; Pecevski, Natschläger, & Schuch, 2009; Seung & Hughes, 2003).

- hardware implementation (Delbruck, 2007; Faiña, Bellas, Souto, & Duro, 2011; Indiveri & Horiuchi, 2011; Indiveri et al., 2011; Lalor et al., 2005; Misra & Saha, 2010; Schrauwen, D'Haene, Verstraeten, & Campenhout, 2008; Van Schaik & Liu, 2005).

- Learning rules/architecture and optimization related learning mechanism (Brader, Senn, & Fusi, 2007; Cooper, 2005; Dhoble, Nuntalid, Indiveri, & Kasabov, 2012; Florian, 2005; Ghosh-Dastidar & Adeli, 2009; Hamed, Kasabov, & Shamsuddin, 2012; Iglesias, Eriksson, Grize, Tomassini, & Villa, 2006; J. Iglesias & A. E. P. Villa, 2006; J. Iglesias & A. P. Villa, 2006; Izhikevich & Edelman, 2008; Kasabov, 2007; Kasabov, 2012a, 2012b; Kasabov, Dhoble, Nuntalid, & Indiveri, 2012; Kasabov, Dhoble, Nuntalid, & Mohemmed, 2011; Kasabov & Hu, 2010; Kasabov, Schliebs, & Kojima, 2011; Legenstein, Naeger, & Maass, 2005; Maass, Natschläger, & Markram, 2004; Meng, Jin, & Yin, 2011; Mohemmed, Schliebs, & Kasabov, 2011; Mohemmed, Schliebs, Matsuda, & Kasabov, 2011; Nawrot, Schnepel, Aertsen, & Boucsein, 2009; Norton & Ventura, 2009; Nuntalid, Dhoble, & Kasabov, 2011; Ponulak & Kasiski, 2010; Schliebs, Defoin-Platel, Worner, & Kasabov, 2009; Schliebs, Nuntalid, & Kasabov, 2010; Schrauwen, Verstraeten, & Campenhout, 2007; Seung & Hughes, 2003; Toups, Fellous, Thomas, & Sejnowski, 2012; Xie & Seung, 2003; Yamazaki & Tanaka, 2007).

From Figure 3.1, the *y* axis indicates the biological possibility (features and parameters) of the models, while the *x* axis indicates implementation cost. Models with fewer biological features (such as the Integrate-and-Fire model) demand much less processing time. By contrast, the Hodgkin-Huxley model mimics biological neurons with high accuracy but at prohibitive implementation cost.

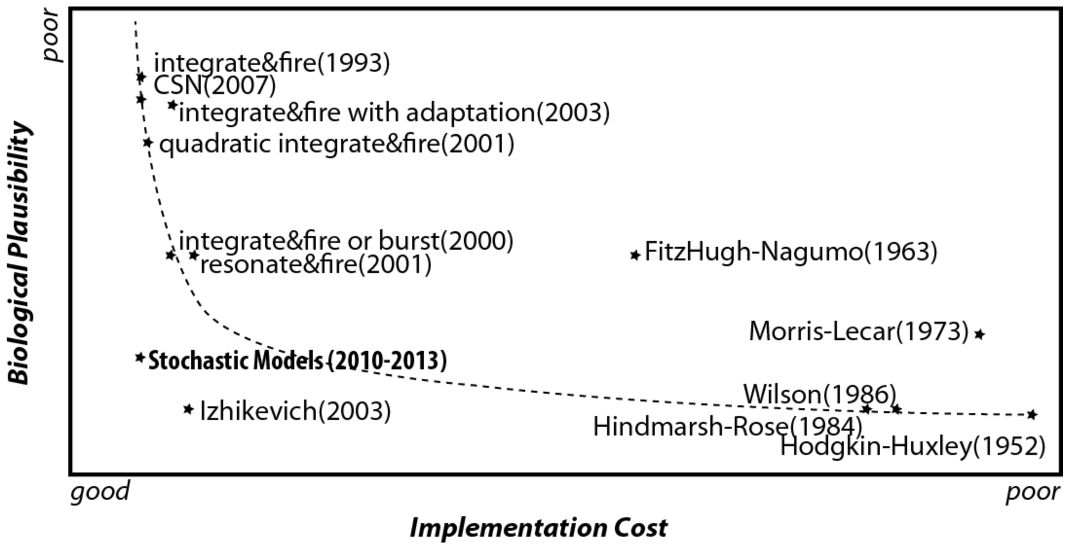


Figure 3.1: The flexibility of use of different SNM.

This thesis focuses on spatio-temporal pattern recognition on EEG data with potential applications to brain computer interface (BCI). BCI requires both fast processing and low implementation cost. To this end, we consider whether models such as Integrate-and-Fire model (IF) and Leaky Integrate-and-Fire model (LIF) can mimic the more realistic behaviors of the Hodgkin-Huxley model without compromising their implementation cost. An output of a Hodgkin-Huxley model, showing the evolution of simulated post synaptic potential, is depicted in Figure 3.2. The figure demonstrates that when a neuron receives an incoming spike, its membrane potential increases up to a threshold. When the threshold is reached, a neuron is fired and the potential drops to a (flexible) reset value. The flexibility in the reset potential of neurons leads to the visible difference between the LIF and Hodgkin-Huxley models.

The activity shown in Figure 3.2 is also linked to the Kasabov model (Kasabov, 2010), outlined in figure 2.7. In this model, $P_i(t)$ and $P_j(t)$ define the probability of release associated with the threshold and reset potential of neurons *i* and *j*, respectively.

Theoretical values of the probability parameters, determined by adding slow noise to the network, were proposed in 2002 by Gerstner and Kistler (W. Gerstner & Kistler, 2002). They explained that a completely different noisy model results if the value of a constraint is altered after every spike. This implies that constraints such as threshold, reset potential or the length of the refractory interlude can provide this type of noise. Figure 3.3 shows membrane potential activity when noise is added to the threshold and reset potentials of a neural model. The noise forces these parameters to change every time a neuron fires.

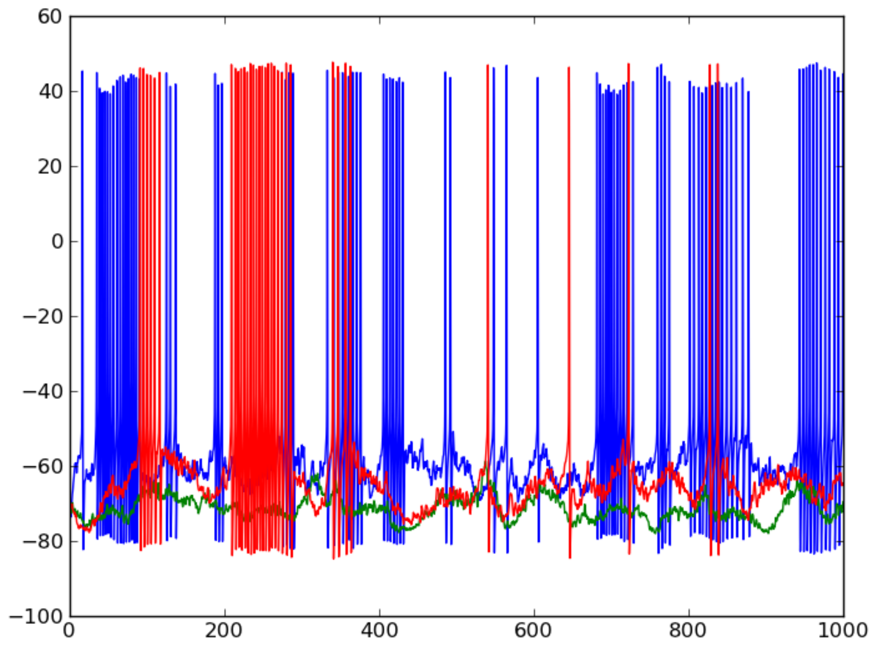


Figure 3.2: Evolution of membrane potentials in the Hodgkin-Huxley Model. The horizontal axis represents the number of time steps, while the vertical axis displays the membrane potential in mV.

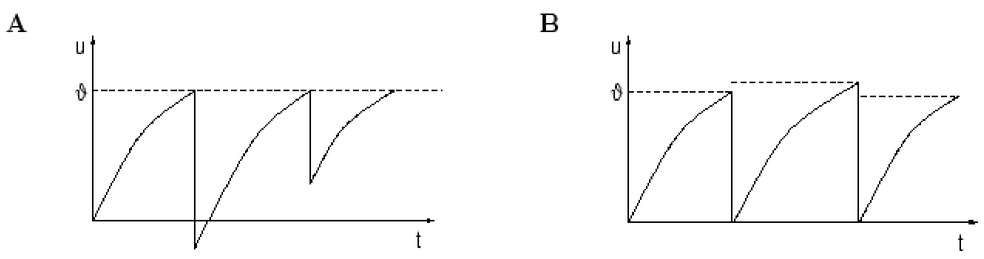


Figure 3.3: Slow noise in the probability parameters (W. Gerstner & Kistler, 2002)

With each neuron ejection, either the reset value (**A**) or the ejecting threshold (**B**) is set to a new randomly selected value. Hence, stochastic spiking neural models are a modification of standard LIF, which evolves according to equation 3.1, in which the threshold and reset potentials are modulated by a noise parameter. Stochastic LIF mimics the activity of biologically plausible SNM while maintaining low implementation cost.

$$\tau_m \frac{du}{dt} = -u(t) + RI(t) \tag{3.1}$$

In equation 3.1, u refers to the membrane potential and τ_m represents the membrane time constant of the neuron. Whenever the membrane capacity reaches the specified threshold, the neuron spikes and the membrane capacity is reset. This equation underpins the three stochastic models in sections 3.1.1, 3.1.2 and 3.1.3. The outputs of these models are compared in Figure 3.4.

Development of the post-synaptic potential $u(t)$ and the ejecting threshold $\vartheta(t)$ over time (blue and red curves in figure 3.4, respectively) are recorded from a single neuron of each neural model. The input stimulus to the neurons is depicted at the top of the figure. The thick black vertical lines above the related threshold curve indicate the output spikes of each neuron.

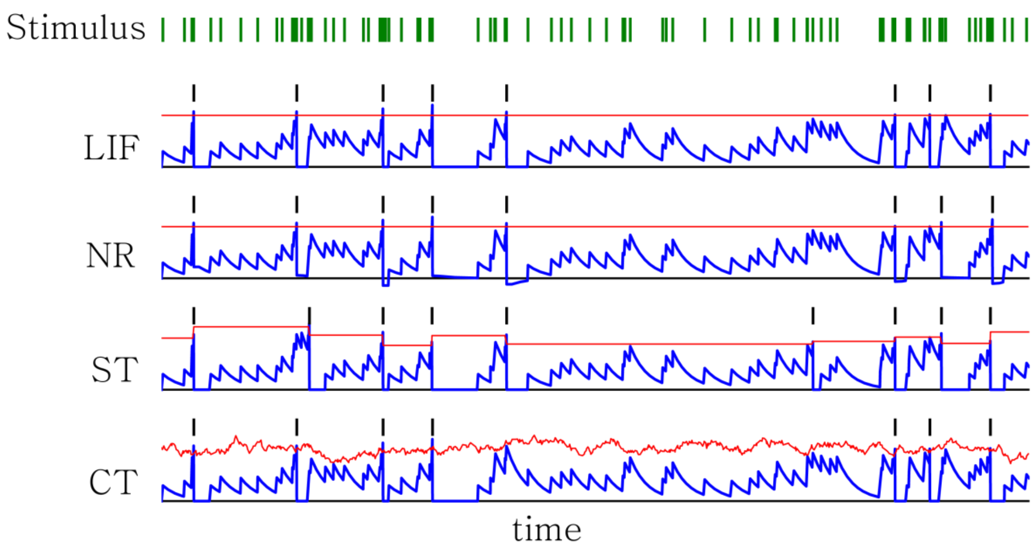


Figure 3.4: Comparisons between stochastic versions of models based on equation 3.1 (see text for details).

3.1.1 Stochastic Noisy Reset Model (NR)

The stochastic noisy reset model modified from deterministic LIF inspired the concept of slow noise model (W. Gerstner & Kistler, 2002). An entirely different concept of noise models is possible whereby the value of a parameter is altered after each spike. In standard LIF, membrane potential always reverts to its reset potential u_{reset} (commonly equal to zero). In noisy reset mode, the reset potentials u_{reset} are Gaussian distributed mean μ and standard deviation σ, as shown in the upper panel of Figure 3.4.

The NR model replaces the deterministic reset potential of LIF with a stochastic equivalent. Let $t(f) : u(t(f)) = \vartheta$ be the firing time of a LIF neuron, after which the post-synaptic capacity is rearranged. N $(\mu, \ \sigma)$ is a Gaussian dispersed random variable with mean μ and standard deviation σ. Variable σ_{NR} denotes a parameter of the model (equation 3.2).

$$\lim_{t\to t^{(f)}, t>t^{(f)}} u(t) = N(u_r, \sigma_{NR}) \tag{3.2}$$

3.1.2 Stochastic Step-wise Noisy Threshold Model (ST)

The stochastic step-wise noisy threshold model (ST) is similar to the NR model described in Section 3.1.1, but u_{reset} is now fixed, while a new threshold value is chosen at each firing time step.

This model replaces the constant ejecting threshold ϑ of the LIF model with a stochastic one. Again, let *t*(*f*) be the ejecting time of a LIF neuron. The dynamics of the threshold update are governed by equation 3.3.

$$\lim_{t\to t^{(f)}, t>t^{(f)}} \vartheta(t) = N(\vartheta_r, \sigma_{ST}) \tag{3.3}$$

Variable σ_{ST} is the standard deviation of $\vartheta(t)$ and is a constraint of the model. In equation 3.3, the threshold is a ϑ_0-centered Gaussian random variable which is sampled whenever the neuron fires. This model does not permit immediate spike movement. Particularly, the neuron can only eject at time *t*(*f*) while simultaneously receiving a pre-synaptic input spike at *t*(*f*); without such a spur, a spike output is impossible.

3.1.3 Continuous Stochastic Threshold Model (CT)

The Continuous Stochastic threshold (CT) model, inspired by the stochastic spike arrival model of the hazard model, was described by Gerstner and Kistler (W. Gerstner & Kistler, 2002). In the CT model, the threshold $\vartheta(t)$ is revised continuously over time. Hence, this model permits immediate spike movement, that is; a neuron may eject at time *t*(*f*) even in the absence of a pre-synaptic input spike at *t*(*f*). The threshold is given by equation 3.4.

$$\tau_\vartheta \frac{d\vartheta}{dt} = \vartheta_0 - \vartheta(t) + \sigma_{CT}\sqrt{2\tau_\vartheta}\,\xi\,(t) \tag{3.4}$$

The noise term ξ follows a Gaussian White Noise distribution with zero mean and unit standard deviation. Variable σ_{CT} denotes the standard deviation of the fluctuations of $\vartheta(t)$ and is a constraint of the model. $\vartheta(t)$ tends exponentially towards a mean value ϑ_0 at rate τ_ϑ, while the magnitude of $\vartheta(t)$ is always directly proportional to the distance $\vartheta_0 - \vartheta(t)$.

As is evident from equation 3.5, the CT model is of similar form to the stochastic LIF model.

$$\tau_m \frac{du}{dt} = -u(t) + \sigma_{CT}\sqrt{\tau_m}\,\xi\,(t) \tag{3.5}$$

u refers to the membrane potential and τ_m is the membrane time constant of the neuron.

3.2 Conclusion

Although sophisticated spiking neural models (SNMs) such as the Hodgkin-Huxley model can simulate realistic biological behaviours, they incur high implementation cost and processing time in spatiotemporal patterns recognition, especially on EEG data. This chapter introduced three stochastic modifications of the standard LIF model, in which threshold and reset potential are perturbed by a noise parameter. The stochastic LIF models can imitate the biologically plausible activity of SNMs while preserving the low implementation cost of standard LIF. In Chapters 6-11, the feasibilities and performances of the three stochastic models will be compared to those of the deterministic LIF model.

Chapter 4

SSPAN: Stochastic Precise-Time Spike Pattern Association Neuron

In this chapter, the stochastic neural model for the precise-time Spike Pattern Association Neuron (SSPAN) is introduced. SSPAN is a modification of SPAN, designed for the precise-time spike pattern association neuron (Mohemmed, Schliebs, & Kasabov, 2011; Mohemmed, Schliebs, Matsuda, & Kasabov, 2011),

The deterministic LIF model is replaced with the stochastic spiking neural models described in Chapter 3; namely, the NR, ST and CT models, and the amendment of architecture contributing to the decrease of training iterations. The above mentioned corresponds with the second research question.

Section 4.1 introduces the original SPAN concept. SSPAN is detailed in Section 4.2. Section 4.3 compares the feasibility and performance of SSPAN and original SPAN, in an experiment using a synthetic dataset. The chapter concludes with Section 4.4.

4.1 SPAN: Precise-Time Spike Pattern Association Neuron

SPAN, precise-time spike pattern association neuron, was proposed in 2011 (Mohemmed, Schliebs, & Kasabov, 2011; Mohemmed, Schliebs, Matsuda, et al., 2011). SPAN aims to explore how a LIF neuron can memorize patterns. The spike trains are converted into analogue signals which are convolved with a kernel function into a continuous-value signal. This processing step enables standard subtraction and multiplication operations which can easily employ existing methods for developing supervised-learning rules in spiking neurons.

The dynamics of SPAN are governed by Equation 4.1. Note the similarity of this equation to that of the LIF model (see Chapters 2 and 3 for a more detailed description of the LIF model).

$$\tau_m \frac{du_i}{dt} = -u_i(t) + RI_i^{syn}(t) \tag{4.1}$$

The ejecting times must be calculated. The synaptic current I_i^{syn} of neuron i is modelled using an α-kernel as shown in Equations 4.2 and 4.3.

$$I_i^{syn}(t) = \sum_j w_{ij} \sum_f \alpha\left(t - t_j^{(f)}\right) \tag{4.2}$$

$$\alpha(t) = e\tau_s^{-1} t e^{-t/\tau_s} \Theta(t) \tag{4.3}$$

w_{ij} is the synaptic weight denoting the strength of the connection between neuron i and its pre-synaptic neuron j, $\Theta(t)$ in Equation 4.3 is the Heaviside function and τ_s is the synaptic time constant. Although spike trains are converted to continuous values, the learning rule of SPAN can be defined as for other artificial neural networks using established training algorithms. The synaptic weights of SPAN are modified to create a required spike design. To modify the weight of a synapse I, the common Widrow-Hoff rule (Equation 4.4) is employed.

$$\Delta w_i^{WH} = \lambda x_i (y_d - y_{out}) \tag{4.4}$$

where λ is a learning rate whose value cannot exceed 1, x_i is the input transmitted via synapse i, and $y_d$and y_{out}denote the desired and actual output spike outputs, respectively. In defining the distance between spike trains, each spike train is convolved with the α-kernel function (Equation 4.3). The convolved input spike train $t_i^{(f)}$ is described by Equation 4.5.

$$x_i(t) = \sum_f \alpha(t - t_i^{(f)}) \tag{4.5}$$

The transformation from spikes to function permits the algorithm to subtract and multiply or calculate the difference between spike sequences. The desired spike output and actual spike output are determined by Equations 4.6 and 4.7, respectively.

$$y_d(t) = \sum_f \alpha(t - t_d^{(f)}) \tag{4.6}$$

$$y_{out}(t) = \sum_f \alpha(t - t_{out}^{(f)}) \tag{4.7}$$

Δw_i updates the weight of synapse i, and is obtained by integrating Δw_i^{WH}.

$$\Delta w_i = \lambda \int x_i\,(t)(y_d(t) - y_{out}(t))dt \tag{4.8}$$

In Figure 4.1, plots (D) and (E) are graphical illustrations of Equation 4.9. Three presented stimuli induce the three output spikes t_0, t_1 and t_2. t_0 equals the required spike time $t_0^{\,d}$ in Figure 4.1 (C). An anomalous spike time will generate errors, as evidenced in Figure 4.1(D). This error will be incorporated into Δw_i via equation 4.8, as shown in Figure 4.1(E).

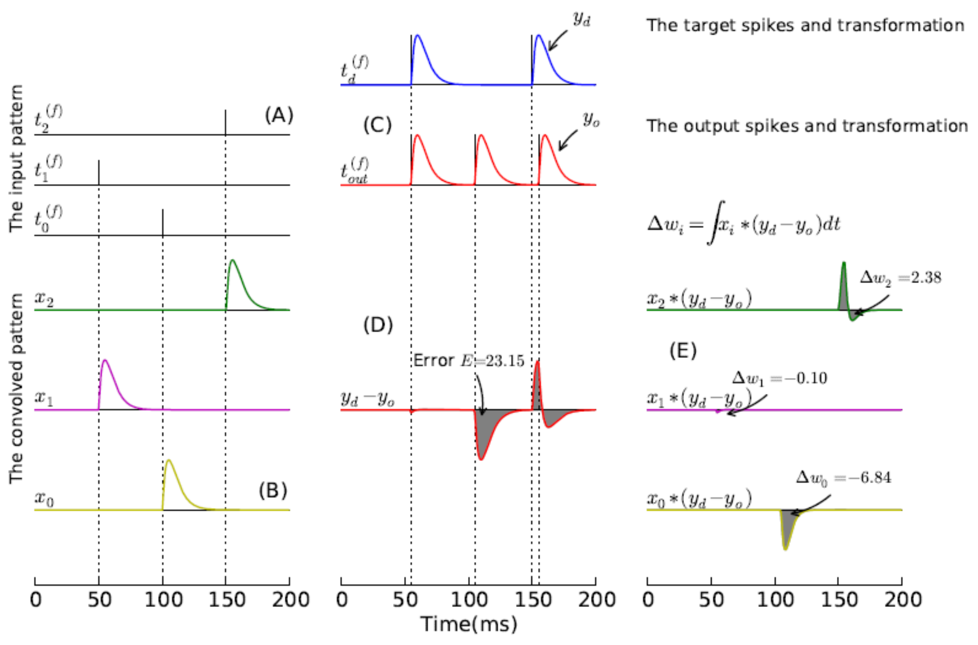


Figure 4.1: Demonstration of learning rule in SPAN(Mohemmed, Schliebs, & Kasabov, 2011)

The error E between the desired and actual spike time, which equals the area under the horizontal axis in figure 4.1, is calculated from equation 4.9.

$$E = \int |y_d(t) - y_{out}(t)| dt \tag{4.9}$$

As the number of synapses increases, the SPAN can acquire more memory in the training process. Increasing the number of training iterations can also enhance the result, but at the cost of increased processing time.

4.2 Stochastic Precise-time Spike Pattern Association Neuron (SSPAN)

The deterministic LIF model is probably not be able to ill-equipped to deal with noisy and stochastic data such as EEG very well. Moreover, in the original SPAN, many synapses are required for multiple pattern recognition; especially when the patterns are spatiotemporal. SPAN adating a single neuron for patterns recognition may become problematic when few input streams arrive, activating a low number of synapses. Therefore, the main purposes of SSPAN are to:

(1) Replace the deterministic LIF model by a stochastic neural model (NR, ST or CT), whose details are provided in Chapter 5. This may improve the performance of SPAN in handling noisy stochastic natural data, especially EEG.

(2) Employ one output spiking neuron for each class instead of using a single neuron for all classes. This ensures that the more output neurons, the more synaptic weights. For instance, assume that 4 classes of EEG data exist

and that 19 channels are output. In this instance, 76 synapses (19 × 4) will be obtained in SSPAN because the algorithm uses one neuron per class.

SSPAN employs the three stochastic neural models described in Chapter 5; namely, stochastic noisy reset (NR) model, stochastic step-wise noisy threshold model (ST) and continuous stochastic threshold (CT). Equations 4.4, 4.6, 4.8, and 4.9 must be adapted to handle multiple output neurons; otherwise, the learning rules of SSPAN are very similar to those of original SPAN.

To modify the weight of a synapse I in SSPAN, Equation 4.4 is replaced by equation 4.10 below.

$$\Delta w_{i,k}^{WH} = \lambda x_i (y_{d,k} - y_{out,k}) \qquad (4.10)$$

where λ is a learning rate whose value does not exceed 1. k =1, 2 ...n where n is the number of classes in a particular dataset.$y_{d,k}$ and $y_{out,k}$ denote the required and true spike outputs of a neuron in class k, respectively, determined by Equation 4.11 and 4.12. Here $t_{d,k}^{(f)}$ and $t_{out,k}^{(f)}$ are the desired and true spike times of a class k output neuron, respectively.

$$y_{d,k}(t) = \sum_f \alpha(t - t_{d,k}^{(f)}) \qquad (4.11)$$

$$y_{out,k}(t) = \sum_f \alpha(t - t_{out,k}^{(f)}) \qquad (4.12)$$

The weights are updated according to Equation 4.13, where $\Delta w_{i,k}$ is the weight change of synaptic connection i of a class k output neuron, obtained by integrating $\Delta w_{i,k}^{WH}$.

$$\Delta w_{i,k} = \lambda \int x_i\,(t)(y_{d,k}(t) - y_{out,k}(t))dt \tag{4.13}$$

The error *E* between the desired and actual spike times of neuron in class *k* is determined as (Equation 4.14).

$$E = \int \left| y_{d,k}(t) - y_{out,k}(t) \right| dt \tag{4.14}$$

4.3 A Comparison between SPAN and SSPAN

To investigate the feasibility and performance of SPAN (Section 4.1) and SSPAN (Section 4.2), both methods are tested on a synthetic dataset comprising three classes. The classification accuracy of the methods is compared in table 4.2.

Figure 4.2 demonstrated an input sample of a class in term of spike activity of the overall different 3 classes. The data, comprising 140 input spike trains with spikes scattered over the time period, were generated over 500 ms. In order to generate extra data samples for testing and training each class sample was then perturbed by 35% noise, to yield 4 and 16 samples respectively.

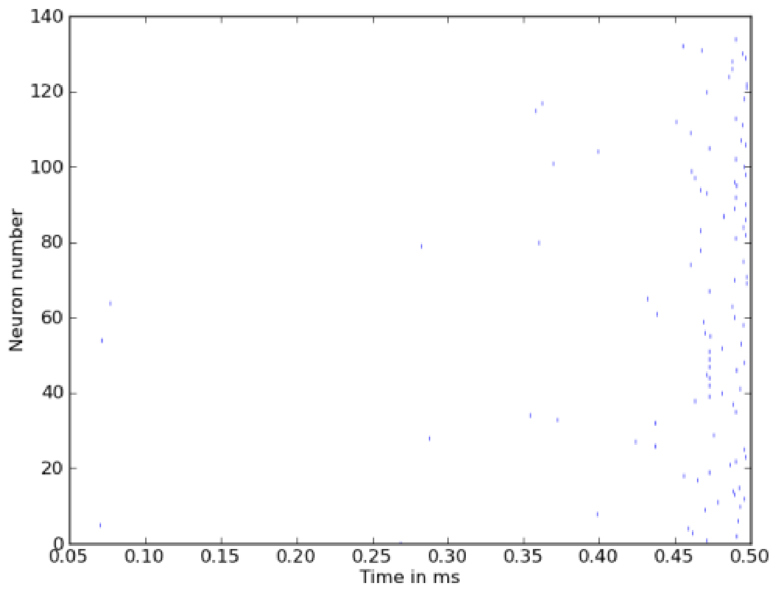


Figure 4.2: Graphical illustration of one sample of a class from the synthetic dataset.

The jitter mechanism employed to create this synthetic dataset was different to that of the original SPAN experiment (Mohemmed, Schliebs, & Kasabov, 2011; Mohemmed, Schliebs, Matsuda, et al., 2011), because the recognition of patterns which change by spike time jitter alone may not fully test the algorithm performance.

The parameter setup of this experiment is summarised in Table 4.1. In this experiment, deterministic LIF with a single output neuron was used in SPAN. The target spike times (desired spike times) were 510 ms, 540 ms, and 560 ms for classes 1, 2 and 3 respectively. The spike time of the output neuron was considered to be correctly classified when it was within 10 ms of the desired spike time.

Table 4.1: Parameter setup for a performance test of SPAN and SSPAN

Methods/Parameters	**SPAN**	**SSPAN**
Neural model	LIF	LIF,NR, ST, CT
Output neuron	1 neuron	3 neurons (of each model experiment)
Input spike train	140 spike trains	140 spike trains
Membrane time constant	10 ms	10 ms
Reset potential	0 mV	0 mV
Firing threshold	30 mV	30 mV
Input weight	1.62 mV	1.62 mV
Reset μ		0 mV
Reset σ		3.0 mV
Threshold σ		2.0 mV
Noisy time constant		10 ms
Synaptic time constant	10 ms	10 ms
Simulation time	600 ms	600 ms
Simulation time step	0.1 ms	0.1 ms
Training iteration	60	60

The three stochastic models (NR, ST, and CT), plus deterministic LIF, were then used in SSPAN. Three output neurons, each with target spike time 515 ms, were employed. In the testing process, every output spike may be able to fire but not necessarily at the same time. Consequently, the spike output of the output neuron that fires closest to the desired spike time was assigned to that class. For instance, if the spike of output neuron 2 is closer to 515 ms than that of the other neurons, this testing sample will be classified as Class 2.

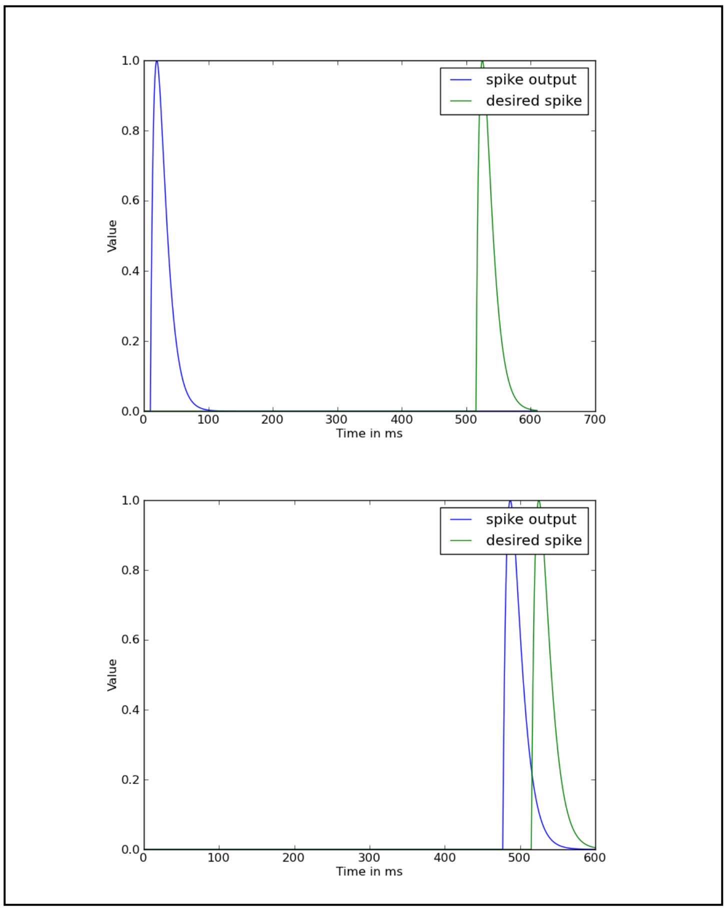


Figure 4.3: True spike output (blue) and desired spike output (green) of SSPAN using the CT model

In Table 4.1, reset μ and reset σ are parameters of the NR model. Threshold σ is a parameter of the ST and CT models, while noisy time constant is a parameter of the CT model only. The simulation time exceeded the stimulus time of 100 ms to allow all spikes in the stimulus to be presented to SPAN and SSPAN before the desired spike time. This ensures that all stimulating information is processed.

The upper panel of Figure 4.3 plots the α curves at the first training iteration, and the lower demonstrates the curves at the last iteration of training (number of iterations = 60).

Table 4.2: Classification accuracy of stochastic LIF-based models

SPAN	**SSPAN(LIF)**	**SSPAN(NR)**	**SSPAN(ST)**	**SSPAN(CT)**
33.33%	50%	66.67%	66.67%	83.33%

The upper panel of Figure 4.4 shows the error curve at the first iteration of training, while the lower panel depicts the error curve at the 60th training iteration.

The percentage classification accuracies of each method are presented in Table 4.2. After 60 training iterations, all of the stochastic SSPAN models exceeded SPAN in terms of accuracy. CT-modelled SSPAN excelled for this synthetic dataset, with a percentage accuracy of 83.33%. Besides, the classification accuracy improved when the deterministic model (LIF model) was replaced with stochastic neural models in SSPAN. SPAN may require more training iterations, possibly as many as 500 (Mohemmed, Schliebs, & Kasabov, 2011; Mohemmed, Schliebs, Matsuda, et al., 2011), which may take around 26 hours (200-260 msec/sample) in system training process and may take longer about 5-7 times in real world application such as EEG because of the higher density of spikes.

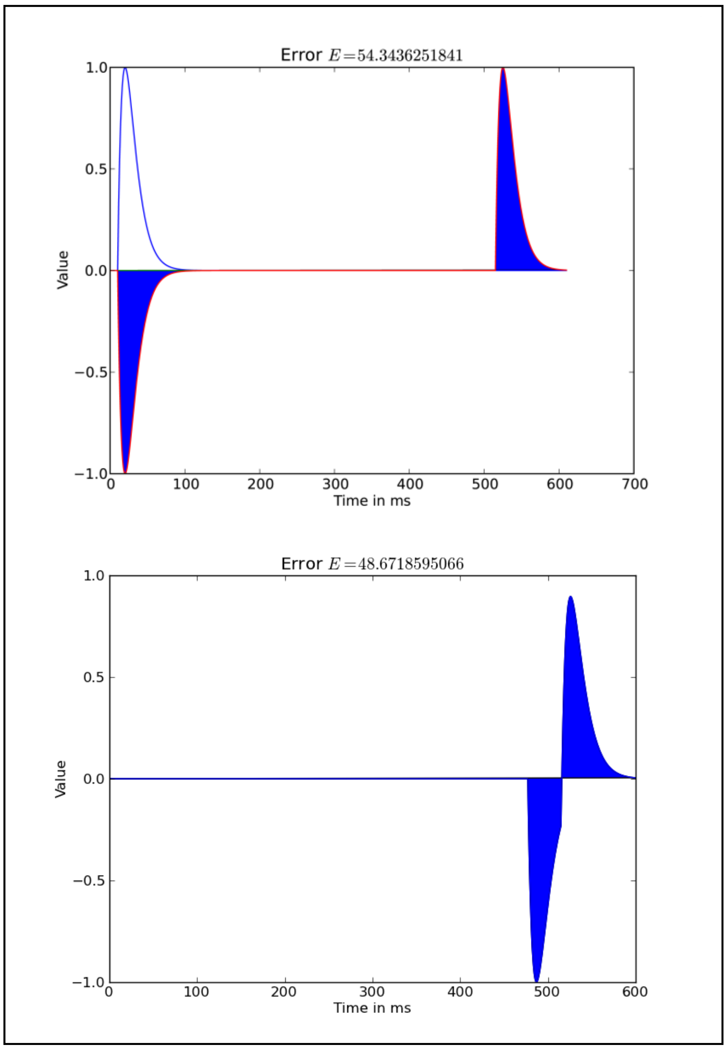


Figure 4.4: Error (E) dynamics of SSPAN using the CT model

Figure 4.3 illustrates the α-transformed true spike output (blue) and desired spike output (green), the spikes were transformed via equation 4.3 of SSPAN using CT model. The upper panel shows the α curves at the first training iteration, where the actual and desired spikes occur at 40.0 ms and 515 ms, respectively. The lower panel illustrates the α curves at the last training iteration (the 60th iteration). Note that the true spike has shifted towards the desired spike (final spike time is 477.2 ms).

The error (E) dynamics of SSPAN using CT model are displayed in Figure 4.4. The curves have converged and the error is decreased after multiple training sessions. The upper panel illustrates the error curve at the first training iteration, where E = 53.34 .After the final training iteration (60th iteration; bottom panel of Figure 4.4), E = 48.67. The error is determined from Equation 4.14.

4.4 Conclusion

This chapter describes the stochastic precise-time spike pattern association neuron (SSPAN), designed to enhance the performance of SPAN: precise-time spike pattern association neuron proposed in 2011 (Mohemmed, Schliebs, & Kasabov, 2011; Mohemmed, Schliebs, Matsuda, et al., 2011).

The performances of both model types were investigated and compared on a synthetic dataset. SSPAN exhibited improved performance over the original SPAN, especially when incorporating the CT model. Moreover, the classification accuracy improved when the deterministic model (LIF model) was replaced with stochastic neural models in SSPAN. However, SPAN may require more training iterations, possibly as many as 500 (Mohemmed, Schliebs, & Kasabov, 2011;

Mohemmed, Schliebs, Matsuda, et al., 2011), before exemplary results are obtained. Hence, we conclude this chapter with the following points:

(1) More synapses can memorize more patterns, leading to increased performance; SSPAN has 420 synapses versus 140 synapses in SPAN.

(2) Stochastic neural models can improve the performance of SPAN

(3) Stochastic neural models are better equipped than deterministic models to deal with noisy stochastic data.

Chapter 5

DepSNN: Dynamic Evolving Probabilistic Spiking Neural Networks

Aside from being computationally inexpensive, the eSNN enhances the significance of the order in which input spikes reach the output neuron; hence eSNN may be prepared using on-line learning. The disadvantages of eSNN are as follows;

(1) No method is known to handle multiple spikes reaching the same synapse at different times and demonstrating the same spatiotemporal design. Such a method is required for spatiotemporal patterns recognition. Although the synapses seize long term memory at the learning stage, their short-term memory is acquired solely through post-synaptic potential growth. Unrestricted short-term memory acquisition is vital for complicated spatiotemporal patterns recognition tasks.

(2) eSNN is appropriate and employs population encoding alone (see Chapter 2 for details) to transform original data into spike trains. Under eSNN, a value is encoded into at least 3 spikes; that is, population encoding may introduce more information to the system than originally exists. For example, at three spikes per datum, population encoding will produce 300 spikes for a data vector containing 100 features.

(3) This thesis focuses heavily on EEG spatio-temporal pattern recognition, which is not suitable for eSNN with population encoding at least 19 channels are present in clinical EEG, which would introduce at least 57 input spike trains to an eSNN, rather than the desired 19 spike trains.

This chapter introduces a new method, namely DepSNN: Dynamic Evolving Probabilistic Spiking Neural Network, which overcomes these limitations. DepSNN is an extended eSNN model which utilises the Rank order learning (RO) (see Section 5.1) and the Fusi's spike driven synaptic plasticity (SDSP) learning rule (see Section 5.2). Section 5.3 provides details of DepSNN using the deterministic LIF model and the three stochastic models NR, ST and CT, introduced in Chapter 3. Rank order learning fixes the original connection weights for a given spatiotemporal pattern using the existing event order information. The SDSP rule then regulates these connection weights as further spikes (occurrences) enter as segments of the same spatiotemporal pattern. The BSA encoding method is employed for transforming EEG data to spikes (as explained in Chapter 8). The chapter concludes with Section 5.4.
This proposed solution addresses the contribution stated in the second research question.

5.1 Rank Order Learning (RO)

The Rank order learning rule enables the neuron to recognize a pattern of neuron links as a positive model. The neurons form centralised clusters in the region of the synaptic weights. In some applications, alike neurons are combined (Kasabov, 2007; Wysoski et al., 2010). In this way, very rapid learning is possible in an eSNN (a single pass may be sufficient), both in an organized and an unorganized mode. In an unsupervised mode, the evolved neurons represent a learned pattern (or a prototype of patterns). The neurons can be labelled and assigned to the same class if the model performs a classification task in a supervised mode of learning.

The postsynaptic potential of a neuron *i* at time *t* is computed as follows:

$$PSP(i,t) = \sum \mathrm{mod}^{order(j)} w_{j,i} \tag{5.1}$$

where: *mod* is a modulation factor; *j* is the index of the received spike at synapse *j,i* and $w_{j,i}$ is the related synaptic weight; *order(j)* denotes the order (the position) in which the spike occurs at synapse *j,i*, relative to all spikes input from all *m* synapses to neuron *i*. The *order(j)* is 0 for the first spike and grows concurring to the input spike sequence. An output spike is produced by neuron *i* if the *PSP (i,t)* exceeds a threshold PSP_{Th}.

Throughout the preparation process, for every training input design a new output neuron is generated and the linked weights are computed based on the sequence of the arriving spikes. In eSNN, the weights of on-line links generated between a neuron n_i and its connections form an input pattern of a recognized class, from which an activated input (aspect) neuron n_j is created utilizing the Rank order learning rule:

$$\Delta W_{j,i} = \mathrm{mod}^{order(j,i(t))} \qquad (5.2)$$

When the overall input pattern is given, the threshold of the neuron n_i is set such that the neuron ejects when the same ST pattern is presented again in the recollection form. The threshold is computed as a fraction (C) of the total PSP:

$$PSP_{\max} = \sum_{j=1}^{m}\sum_{t=1}^{T}(\mathrm{mod}^{order(j,i(t))}\,W_{j,i(t)}) \quad ,for\ j=1,2,...m;\ \ t=1,2,...,T \qquad (5.3)$$

$$PSP_{Th} = C * PSP_{\max} \qquad (5.4)$$

If the linked weight vector of the prepared neuron is akin to that of the already prepared neuron in a repository of outcome neurons characterising the similar class, the novel neuron will combine with the most similar neuron, averaging the linked weights and the threshold of the two neurons (Kasabov, 2007; Wysoski et al., 2010). Otherwise, the novel neuron supplements the established set of neurons (or the related class repository of neurons when a supervised learning for categorization is undertaken). The resemblance between the newly generated neuron and a trained neuron is calculated as the inverse of the Euclidean distance between the weight matrices of the two neurons.

Two algorithms are commonly used in the recall process:

(1) The foremost is used when Rank order learning is applied to a new input pattern (either for recalling or testing; see Equation 5.2). The linked weight vector for this input is matched with the patterns of prevailing neurons for which the output class is generated throughout training. The neighboring neuron is the 'winner' that determines the class of the new input design. This algorithm utilizes transductive interpretation codes and closest neighbor categorization. It matches the synaptic weight vectors of a novel neuron acquiring a new input pattern with prevailing weight vectors. This model will be designated eSNNs.

(2) An alteration of the above algorithm is implemented when a new input pattern of spikes is broadcasted as the spikes reach all of the trained neurons. The foremost ejecting neuron (whose PSP exceeds its threshold) determines the class. This algorithm supposes that the fastest ejecting neuron best categorizes the input ST pattern. This eSNN is designated eSNNm.

5.2 The Fusi's spike driven synaptic plasticity (SDSP) learning rule

The SDSP, a modification of STDP (Song, Miller, Abbott, & others, 2000) which has been described as an unsupervised learning method (Fusi, Annunziato, Badoni, Salamon, & Amit, 2000). SDSP models the synaptic plasticity V_{w0} of a synapse w_0 contingent at the time of spiking of the pre-synaptic and post-synaptic neurons. V_{w0} rises or reduces depending on the comparative timing of the pre and post synaptic spikes.

When a pre-synaptic spike reaches the synaptic station before a postsynaptic spike within a specified time window, the synaptic efficiency is enhanced. If the post-synaptic spike is ejected immediately after the pre-synaptic spike, synaptic efficiency is reduced. This plastic synaptic efficiency is embodied in the following equations:

$$\Delta V_{w0} = \frac{I_{pot}(t_{post})}{C_p} \Delta t_{spk} \text{ if } t_{pre} < t_{post} \tag{5.5}$$

$$\Delta V_{w0} = -\frac{I_{dep}(t_{post})}{C_d} \Delta t_{spk} \text{ if } t_{post} < t_{pre} \tag{5.6}$$

where Δt_{spk} is the pre and post synaptic spike time window.

The SDSP rule can be applied to a supervised learning algorithm, when a trainer signal imitating the required output spiking order is inserted together with the training spike pattern, but without any alteration of the trainer input weights. In a study by Brader and colleagues (2007), a SDSP-driven SNN learned to recognise 293 characters (classes). Each character (a static image) was encoded as a 2000 bit feature vector, and each bit was transformed into a spike rate, with binary values 0 and 1 corresponding to 0 Hz and 50 Hz, respectively (Brader et al., 2007). For each class, 20 distinct training designs were utilized and 20 neurons were assigned, one for each pattern, and trained for hundreds of reiterations. The SDSP model is implemented in the INI analogue SNN silicon chip (Indiveri et al., 2011). The silicon synapses are composed of bi-stability circuits that assign a synaptic weight to one of two likely analogue values (either increased or decreased). These circuits push the synaptic weight voltage by a positive or negative current superimposed on that produced by the STDP. Within a short time frame, 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 activity. In the absence of a current (and also in the learning phase) this background current will drive the weight toward its potentiated state. If the STDP decreases the synaptic weight below the threshold, the bi-stability circuits generate a negative current that inhibits spiking activity and vigorously drives the weight toward its analogue value. The synapse is now in a depressed state. The STDP and bi-stability circuits simplify the simulation of both long term and short term memory.

Although SDSP is effective at identifying generally static patterns, the potential of the SDSP SNN model and its accompanying hardware has not been completely investigated in spatiotemporal pattern recognition, particularly in fast on-line learning.

5.3 Dynamic Evolving Probabilistic Spiking Neural Network (DepSNN)

Non-stochastic DepSNN is also known as DeSNN (Dynamic Evolving Spiking Neural Network) (Dhoble, Nuntalid, Indiveri, & Kasabov, 2012).

Five types of DepSNN (namely DepSNNm, DepSNNs, NR-DepSNNs, ST-DepSNNs, and CT-DepSNNs) are proposed in this study. The models, based on the ideas presented in Section 5.1, are the eSNN equivalents of the stochastic models introduced in Chapter 5. They differ in their recall algorithms and type of eSNN (eSNNs or eSNNm). The essences of the five models are outlined below:

(1) DepSNNm

In this model, every new spatiotemporal pattern to be recognized or associated with a previously learned spatiotemporal pattern is propagated to all neurons created during the training session. The first neuron to spike indicates the desired association (or class, in classification tasks). The neurons in the DepSNNm can be designed to inhibit each other (the so-called 'winner takes all' – WTA connection), so that a firing neuron will prevent other neurons from firing (both during recall and training) (Tymoshchuk & Kaszkurewicz, 2005). In this case the firing neuron represents the recognized spatiotemporal pattern (a concept neuron).

(2) DepSNNs

This model, the dynamic equivalent of eSNNs, compares the connection weights of a newly created neuron (representing a new spatiotemporal pattern to be recognized) with the connection weights of the neurons created during training. The new spatiotemporal pattern becomes associated with the closest neuron. This model demonstrates superior performance over DepSNNm in preliminary tests of EEG spatiotemporal pattern classification.

(3) NR-DepSNNs

In this model, the deterministic LIF model in conventional DepSNNs is replaced by the stochastic noisy reset model (see Section 3.1.1), in which the reset potential is refreshed after every spike.

(4) ST-DepSNNs

Like NR-DepSNNs, this model is conceptually analogous to standard DepSNNs; however, the deterministic LIF model is replaced by the stochastic step-wise noisy threshold model (see Section 3.1.2), in which the reset potential value (u_{reset}) is fixed while a new threshold is selected at each firing time step.

(5) CT-DepSNNs

In this model, the deterministic LIF model in standard DepSNNs is replaced by the continuous stochastic threshold (see Section 3.1.3), in which the threshold $\vartheta(t)$ is revised continuously over time. This model allows spontaneous spike activity, *i.e.* a neuron may fire at time $t(f)$ even if no pre-synaptic input spike presents at that time (Schliebs, Nuntalid, & Kasabov, 2010).

Figures 5.1 and 5.2 show how DepSNN responds to different input spike trains. The spike raster is plotted in the top panel of each figure. The central panels illustrate the change in weights over time (blue curves) for the DepSNNm. The initial weights are defined by rank order. The green curves display the weights modified by SDSP dynamics over the simulation time (note that, in Figure 5.1, the green and blue curves almost overlap, so the green curve is scarcely visible). The bottom panels illustrate the evolution of post synaptic potential of the first neuron.

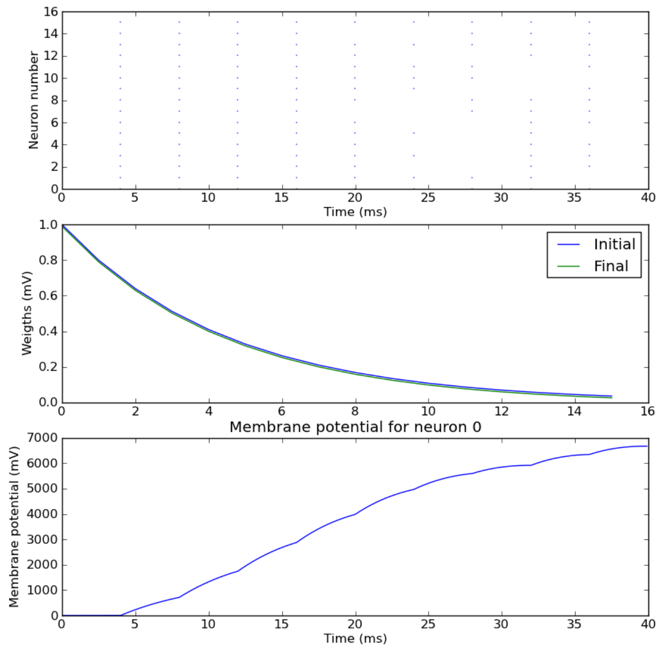


Figure 5.1: Spike Raster Plot (top panel), Weight and PSP evolution (centre panel) and changes in post-synaptic potential of the first firing neuron (bottom panel), for a slow rate of input spikes to a DepSNNm model

DepSNN requires high firing activity in the spike trains to activate a SDSP synapse. This is clearly illustrated in Figure 5.1, where spiking activity is low. In contrast to the case of high spiking activity (Figure 5.2), the initial and final synaptic weights remain similar throughout the evolution time.

The DepSNN algorithm is provided in Table 5.1.

Table 5.1: DepSNN Algorithm

```
1:    Set DepSNN constraints (comprising of: Mod, C, Sim and the SDSP constraints)
2:    For every input STP $i$ demonstrated as BSA Do
        2a. Generate a new output neuron $j$ for this pattern and compute the starting values of connection weights utilizing the RO learning rule: $w_j = (Mod)^{order(j)}$
        2b. Modify the connection weights $\mathbf{w}_j$ for successive spikes on the related synapses utilizing the SDSP learning rule.
        2c. Compute $PSP_{max}$
        2d. Compute the threshold value $\chi_i = PSP_{\max(i)} * C$
        2e. If the new neuron $j$ weight vector $\mathbf{w}_j$ is alike to the weight vector of previously trained output neuron utilizing the Euclidean distance and a threshold Sim, then combine the two neurons (elective):
        $w = \frac{w_{new} + w * N}{N+1}$,   $\chi = \frac{\chi_{new} + \chi * N}{N+1}$
          where $N$ is the number of all previous merges of the merged neuron
          Else
          Add the new neuron to the output neuron repository for the same class (if a classification task is considered).
          End If
3.      End For (Repeat to all input STP)
```

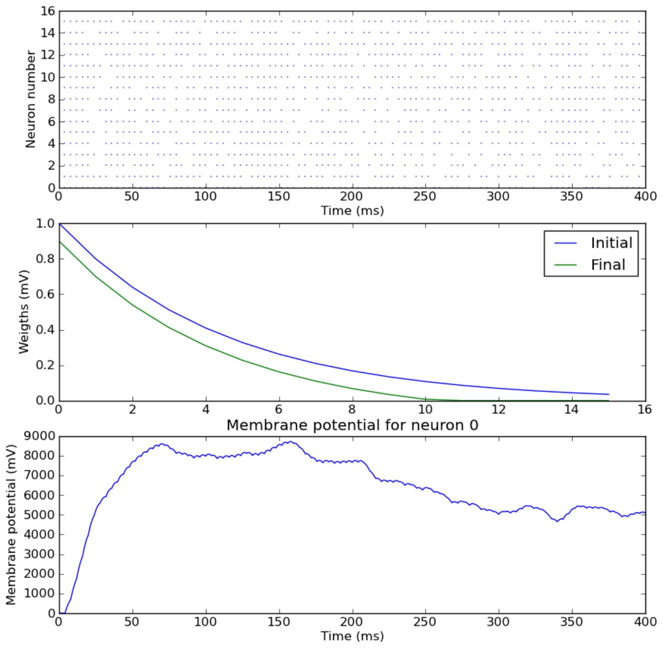


Figure 5.2: Spike Raster Plot (top panel), Weight and PSP evolution (centre panel) and changes in post-synaptic potential of the first firing neuron (bottom panel), for a fast rate of input spikes to a DepSNNm model

5.4 Conclusion

This chapter has introduced five types of Dynamic evolving probabilistic spiking neural network (DepSNN), namely, DepSNNm, DepSNNs, NR-DepSNNs, ST-DepSNNs, and CT-DepSNNs. As an initial investigation of DepSNN behaviour, the model was tested on two sets of synthetic input spike trains. The results (Figures 5.1 and 5.2) show that high levels of spiking activity are required in the spike trains before a SDSP synapse is activated in DepSNN. This is revealed in the curves showing modification of synaptic weights. Thus, DepSNN may be suitable for processing BSA-encoded EEG data, provided that the encoded data spike frequently. In Chapters 10 and 11, the performance and feasibility of DepSNN is further explored on two real-world EEG datasets.

Chapter 6

epSNNr: Evolving Probabilistic Spiking Neural Network Reservoir

The latest reservoir computing paradigm, Liquid State Machine (LSM), was developed to process pattern recognition tasks on spatio-temporal information. LSM is a recurring network of ejecting neurons that converts a spatio-temporal input pattern into a single intermediary high-dimensional network state which first proposed by Maass (Maass et al., 2002).

As a result, an evolving probabilistic spiking neural network reservoir (epSNNr) emerges. Replacing the deterministic LIF model with stochastic neural models introduces a non-deterministic component into the LSM.

Partial information in this chapter has been shared and published to international neural network community in 2010 (Schliebs, Nuntalid, & Kasabov, 2010). The proposed epSNNr is a contribution that answers to the third research question.

The principle of LSM is presented in Section 6.1, followed by a proof of concept in Section 6.2. An improved separation capability of the epSNNr is demonstrated in section 6.3. Section 6.4 concludes the chapter.

6.1 Introduction to Reservoir Computing

The general architecture of reservoir computing is illustrated in Figure 6.1. The structure is analogous to that of a neural network. Essentially, a signal is input to a static dynamic system (the reservoir) and the dynamics of the reservoir transform the input to a greater state. A straightforward readout function is then trained to convert the response state (higher dimension) into the required output. The major benefit is that the learning procedure is conducted only at the readout step. Two major types of reservoir computing are liquid-state machines and echo state networks. Backpropagation-Decorrelation and Temporal Recurrent Networks (Benjamin Schrauwen et al., 2007) also belong to reservoir computing.

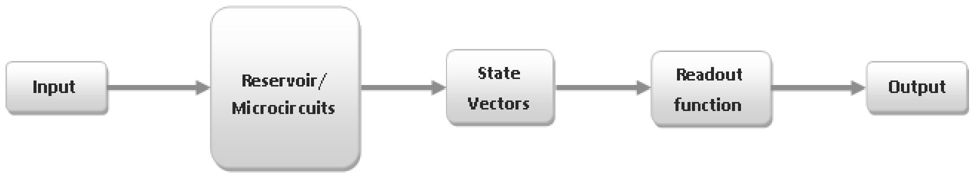


Figure 6.1: Block diagram of Reservoir Computing

The LSM was first proposed by Maass (Maass et al., 2002) and has been widely investigated both conceptually and practically as a new framework for neural computation in machine learning (Benuskova & Kasabov, 2007; Brader et al., 2007; Buteneers et al., 2009; Maass et al., 2002; Maass & Zador, 1999; Norton & Ventura, 2009; Benjamin Schrauwen et al., 2007; Verstraten, Schrauwen, Stroobandt, & Van Campenhout, 2005; Yamazaki & Tanaka, 2007) . LSM has also been proposed for solving time-series problems, since it is free of the complications which persist in recurrent neural networks and cause problems with learning methods. LSM is typically applied to nonlinear problems and is frequently used in conjunction with LIF models. The readout is normally linear. By default, the batch training methods use linear regression to determine the output weights. A least mean squares algorithm is adopted for online training of the traditional reservoir.

A LSM uses an excitable medium (recurrent networks of spiking neurons), to transform multi-dimensional inputs to a single linear dimension. Simple readout units can then extract detailed temporal information from the translated data. Furthermore, LSM mimics the mammalian brain process, lending a certain biological plausibility to the LSM approach. Some areas of the brain may perform as a liquid generator while others learn how the liquid responds to external sensory incentives.

To visualise the LSM concept, imagine a pool of water into which various objects are dropped. The resulting splashes and ripples that fade away over time can be transformed into a spatiotemporal pattern of liquid (liquid state). In other words, the water can retain information about recent events. Real-time events, therefore, should be tractable by reading the water surface of the pool.

LSM comprises two main parts, a liquid unit (a reservoir for transforming the input time series into liquid states or state vectors) and a readout unit (simple function(s) which map(s) the liquid state at time *t* onto the output). The LSM mechanism proposed by Maass (Maass et al., 2002) is shown schematically in Figure 6.2. Here, $u(.)$ is a continuous input stream (spatiotemporal data) of disturbances which is injected into medium L^M that acts as a spatiotemporal filter (liquid filter). The “liquid” constitutes anything that generates a readable liquid state $x^M(t)$ at each time step t. The liquid state is mapped to the desired output function $y(t)$. The readout functions f^M (multiple readouts are permitted) extract different task specifications (e.g. classification, clustering, prediction) in parallel from the current output of L^M.

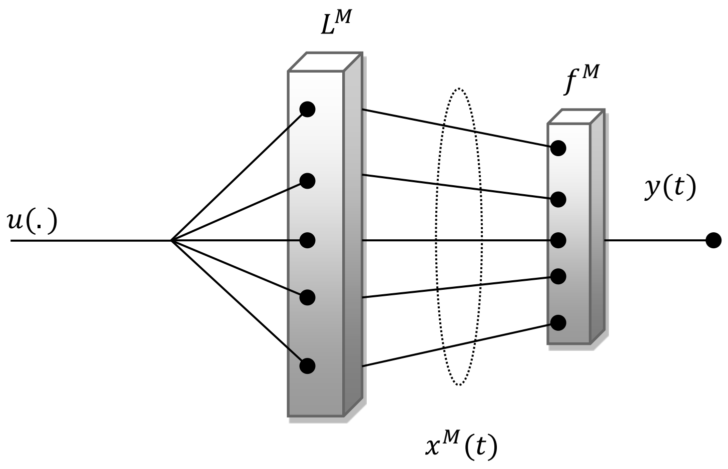


Figure 6.2: Mechanism of Liquid State Machine (LSM)

The liquid unit is typically implemented by a recurrent SNN. Any spiking neural model can be used as a liquid, with generation of different network states resulting in different readouts (Grzyb, Chinellato, Wojcik, & Kaminski, 2009). The liquid unit is a non-linear stochastic system composed of a pool of spiking neurons which receives temporal input and transforms it into significantly different liquid states. The readout unit is a task-dependent portion that can be trained to extract information from liquid states. The pool of spiking neurons is set up on a regular 3D grid space ($N = n_x \times n_y \times n_z$ neurons), where, n_x, n_y and n_z signify the number of neurons assigned to the *x*, *y*, and *z* axes respectively (Burgsteiner, Kroll, Leopold, & Steinbauer, 2007).

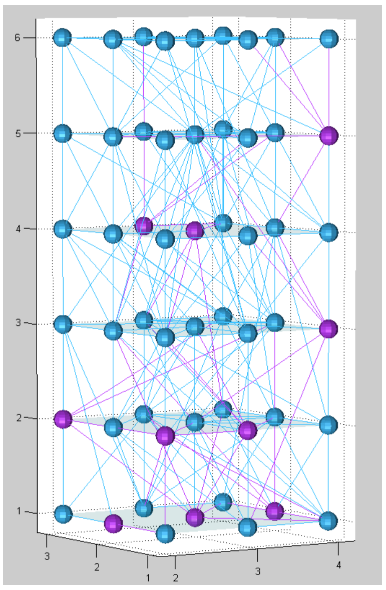


Figure 6.3: Example of a liquid unit of a LSM containing $3 \times 3 \times 6$ **neurons**

Figure 6.3 illustrates the configuration of a pool of spikes in a hypothetical LSM. This figure was created by PCSIM interfaced to the Python programming language (Neural Microcircuit library for Python) (Pecevski, Natschläger, & Schuch, 2009).

The readout unit must be able to detect features from a set of patterns. This unit receives a single linear input from the liquid unit. By its capacity to interpret numerous linear algorithms, the unit can be instructed to execute a particular task. When the spatiotemporal input $u(.)$ is fed into the reservoir, the spiking neurons in the pool are activated. The pool then acts as a filter that transforms the data into the liquid state. Samples of the liquid state are combined into a state vector, which is input to the trained readout unit to perform a specific task.

6.2 Frameworks of the Evolving Probabilistic Spiking Neural Network Reservoir (epSNNr)

The Evolving Probabilistic Spiking Neural Network Reservoir (epSNNr) categorises spatiotemporal data founded on a probabilistic reservoir computing pattern. The framework of epSNNr is presented in Figure 6.4. Noise diminution or feature withdrawal is possibly applied at the pre-processing stage.

In the first step, every spatial/spectral-temporal data channel is converted to a spike train by the BSA spike encoding method described in Chapter 4. Spike encoding may process whole temporal data or chunks of data. Next, the spike trains are dispersed through a LSM-based spatiotemporal filter. In this study, the conventional LIF model as liquid generator has been replaced by the stochastic neural models proposed in chapter 5. At each time step, the liquid state generated by the filter is collected into a linear state vector. The Classifier will use this state vector to perform classification or pattern recognition tasks, possibly in both batch and real-time.

The data in the state vectors is fed into the readout unit as a time-dependent vector or as a whole liquid state pattern. The Classification can comprise any linear classifier or a second trainable neural network.

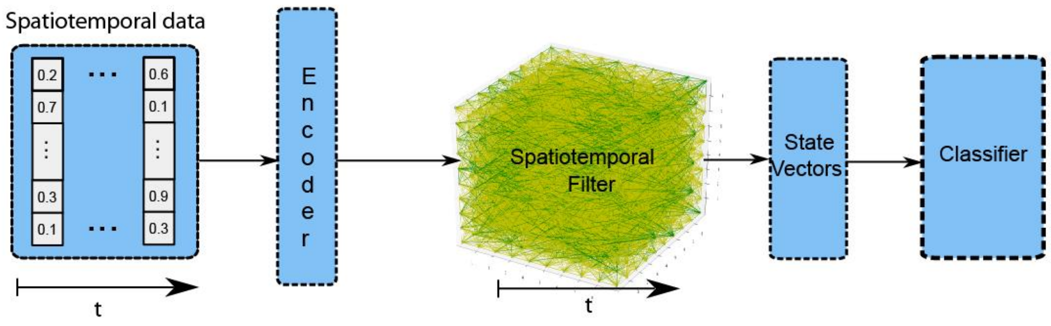


Figure 6.4: epSNNr: Framework of evolving probabilistic spiking neural network reservoir

6.2.1 Design of the encoder

This study focuses on EEG spatio-temporal pattern recognition using the Ben Spiker Algorithm (BSA) introduced in Chapter 4. Because BSA generates spike trains by mimicking input waveforms, it is inherently suitable for EEG data processing, and it should retain information during data transformation to spikes.

6.2.2 Design of the Spatiotemporal Filter

At this stage, the LIF model is replaced by the stochastic models (see Chapter 5 for details); namely the NR, ST, and CT models inspired by probabilistic models

(Kasabov, 2010). A schematic of the probabilistic SNN is shown in Figure 2.7 (Chapter 2). As mentioned in Section 3.1 (Chapter 3), the $P_j(t)$ and $P_i(t)$ parameters define the probabilities of release that are directly associated with neural threshold and reset potential value, respectively. The probability of connection $P_{cij}(t)$ between two neurons *i* and *j* is given by:

$$P(i,j) = C \times e^{\frac{-d(i,j)}{\lambda^2}} \tag{6.1}$$

where $d(i,j)$ is the Euclidean distance between neuron *i* and *j* and λ represents the density of connections (defaulted to λ=2 for LSM). The probability of connection increases with decreasing distance between neurons. C is a constant whose value depends on neural type(excitatory (ex) or inhibitory (inh).

6.2.3 Design of State Vector

Once the spatiotemporal filter has transformed the data into the liquid state, the liquid state is sampled and input to a time-dependent sequence or vector, namely the state vector. The responses of active neurons are used to train the readout unit (Classifier) to perform a classification or pattern recognition task. In this study, the output spikes of the LSM (state vector) are transformed into a binary data representation of *N* parallel spike trains covering a data stretch of *T* bins, each of time-width *h* (Grun, Diesmann, & Aertsen, 2010), where *h* is chosen to suit the dataset under consideration. Next, the binary data are mapped, either by a simple linear mapping or by distribution of coincidence, onto a single binary vector which is an input to a readout function (classifiers). When spiking neural networks are used as readout functions e.g. DepSNN or stochastic SPAN (SSPAN) (see Chapters 6 and 7 for details), the entire dynamic sample of spike responses from the LSM can be fed directly to classifiers.

6.2.4 Design of the classifier

The classifier in Figure 6.4, constituting the readout functions of LSM, can extract different task specifications (such as classification, clustering and prediction) in parallel from the state vector. At a given time *t*, the state vector linearizes the high-dimensional inputs via a simple linear mapping, or uses distribution of coincidence to map binary vectors into a single binary vector within a specific time bin. This unit can implement diverse linear pattern recognition methods such as Naïve Bays, MLP, or another trainable SNN.

6.3 Deterministic LSM Versus epSNNr

The appropriateness of the suggested epSNNr can be assessed by demonstrating the separation ability of the model. Experiments in this section were inspired by a 2009 study of various neural models in a LSM context (Grzyb et al., 2009). Grzyb and colleagues demonstrated that the departing capability of the liquid depends upon the selected neural model. The following constraints were placed on the neural models: the membrane time constant (τ) = 10ms, the reposing potential u_{reset} = 0 mV, the ejecting threshold ϑ_0 = 10mV, the after-spike refractory period

Δabs = 5ms, the standard deviation of reset variations σ_{NR} = 3mV, the standard deviation of step-wise ejecting threshold σ_{ST} = 2mV and the standard deviation of continuous ejecting threshold σ_{CT} = 1mV.

To investigate the differences in neural reaction between the stochastic models, a random spike train produced by a Poisson process with mean rate 150Hz was provided to each model as neural input. After 1000 iterations of each model, the related Peristimulus Time Histograms (PSTH) were computed. The PSTH generates a histogram of spikes arising in a raster plot. A frequency vector is calculated which is standardized by dividing each vector element by the number of repetitions and by the size of the time bin (here 1ms). Maass et al. (2002) used a window width of 10ms to Gaussian-smooth the raw PSTH. In a second experiment, they constructed a liquid possessing a small-world inter-connectivity pattern.

In this study, a repetitive SNN is created by associating 1000 neurons in a three dimensional grid of size 10 × 10 × 10 neurons. Links between any two neurons in this grid are established with association probability given by Equation 6.1, with λ =2 in all simulations. Parameter C is a constant whose value relies on proportion of links between excitatory (ex) and inhibitory (inh) neurons as defined below:

C_{ex-ex} =0.3, C_{ex-inh} =0.2, C_{inh-ex} =0.5, and $C_{inh-inh}$ 0.1. A network comprising 80% and 20% excitatory and inhibitory neurons respectively typifies the biological neurons in the mammalian brain (Schliebs et al., 2010).

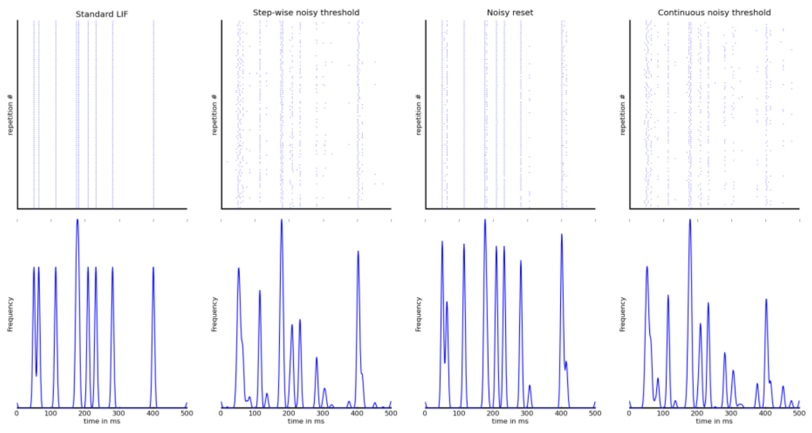


Figure 6.5: a) Raster plot of the neural response. b) PSTH for each raster plot

Figure 6.5 a) shows the Raster plot of the neural response of deterministic LIF neurons and stochastic neurons documented over 1000 repetitions. Figure 6.6 b) shows the related smoothed PSTH for each raster plot. Each column relates to the neural model specified in the plot title. As evidenced in Figure 6.5, the non-deterministic neural dynamics exerts a strong effect on the output response. Some of the spikes appear in every repetition, causing sharp peaks in the PSTH.

Four repetitive SNN are created, each utilizing a different neural model. All networks possess similar network topology and a similar link weight matrix. The created networks are incited by two input spike trains separately created by a Poisson procedure with a mean rate of 150 Hz for the first stimulus (Stimulus A) and 200Hz for the second (Stimulus B). The reaction of each network was recorded over 25 repetitions. The averaged reaction of the networks is illustrated in Figure 6.6. The upper and central panels of the figure depict the average raster plots of the spike movement under stimuli A and B, respectively. The darker the shade in these plots, the more likely the related neuron ejected within the permitted time bin during the 25 runs. The size of a single time bin is 1 ms. White areas indicate time bins in which no neural movement was detected in any run. Similar to the raster plots of Figure 6.5, some reliable spikes are observed in the response, corresponding to the very dark shades in the plots.

Figure 6.6 illustrates the average spike response of the reservoirs using different neural models. Responses to input stimuli A and B recorded over 25 independent runs are displayed in the upper and central panels, respectively. The bottom panel shows the averaged normalized Euclidean distances between two reactions for each time bin (bin size 1 ms). Similar distance calculations were used by Grzyb (Grzyb et al., 2009) to evaluate differences in response patterns. We note the comparably low separation ability of the deterministic LIF model, which confirms the findings of Grzyb and colleagues (Grzyb et al., 2009). The results indicate that stochastic models can potentially enhance the separation ability of the reservoir. However, further experimental analysis is needed to provide strong statistical proof of this claim.

Figures 6.7 and 6.8 show the configuration of the 1000 neurons in the 3D grid of size$10\times10\times10$. The yellow and green lines represent links to excitatory and inhibitory synapses respectively. A liquid possessing a small-world inter-connectivity pattern, in which most neurons are connected locally, was constructed according to equation 6.1. The red lines show the connections between input stimuli and liquid.

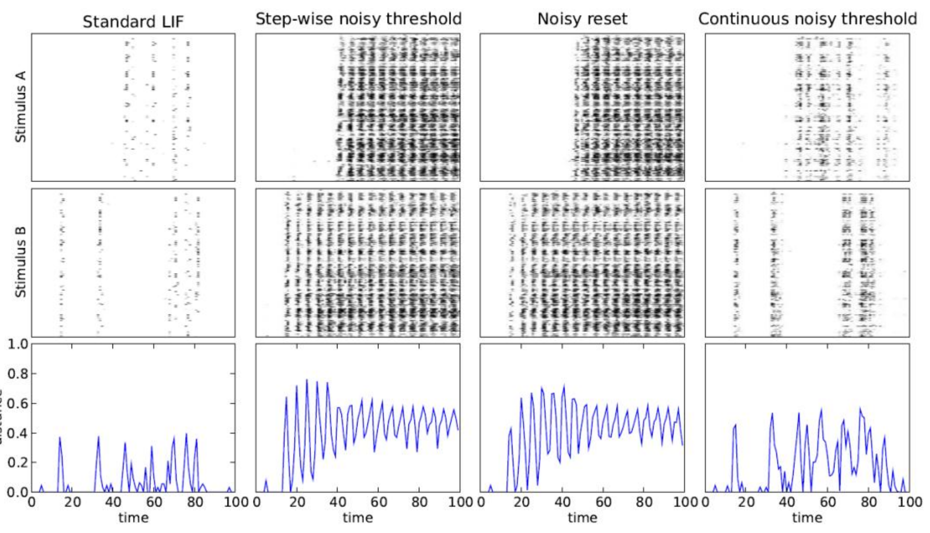


Figure 6.6: Averaged spike response of reservoirs using different neural models. Upper and central panels show the dynamic responses to Stimuli A and B, respectively, while the bottom panel displays the average Euclidean distance between paired reactions (see text for details).

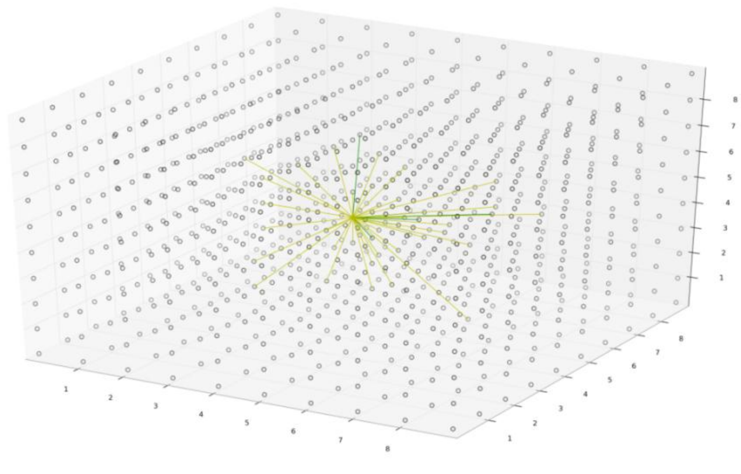


Figure 6.7: Illustration of a liquid possessing a small-world inter-connectivity pattern (see text for details).

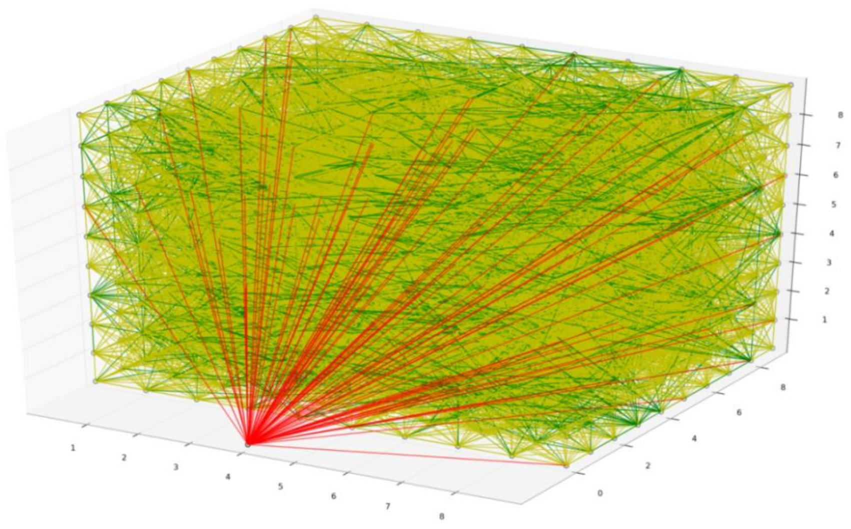


Figure 6.8: Evolving probabilistic spiking neural network (epSNNr) visualization

Figures 6.7 and 6.8 were produced by code written in the pure Python programming language. The synaptic connection information was obtained from the Brian library for SNN simulation (Goodman & Brette, 2008).

6.4 Conclusion

An epSNNr framework was proposed that allows the method to develop spatiotemporal data. epSNNr protrudes a spatiotemporal signal onto a single higher-dimensional network condition that can be learned by a linear readout function or another SNN network. A preliminary feasibility investigation was undertaken on the proposed epSNNr approach. Probabilistic neural models (stochastic models) proved to be primarily appropriate reservoirs with enhanced capacity to increase the separation capability of the system. Further studies will explore the features of the epSNNr on general benchmark functions. Moreover, the approach is anticipated to be extendable to real world EEG data sets, as suggested in successive chapter 9, 10 and 11.

Chapter 7

Electroencephalography

This chapter explains the principles of Electroencephalography (EEG), the data of which are used throughout the remaining of this thesis. An overview of EEG and a broad outline of its applications are presented in Section 7.1. Section 7.2 introduces several applications of EEG to the computer science field. Section 7.3 presents the application of Spiking Neural Networks on EEG. The chapter concludes with Section 7.4.

7.1 Electroencephalography (EEG)

Electroencephalography (EEG) involves the recording of neural-generated electrical brain signals as they move along the scalp. The essential EEG apparatus is shown in Figure 7.1. EEG has been utilized in clinical recordings of cerebral electrical movement over specified periods of time. It collects data from nineteen electrodes positioned strategically across the head. In neurology, EEG is widely adopted to investigate epilepsy, since epileptic movement can generate distinct spike movement on standard EEG equipment (Niedermeyer & Da Silva, 2005; Tatum, 2007). EEG is commonly used to determine the type and position of brain activity movement during a spasm. It is also used to examine individuals with brain functioning problems such as coma, tumours, short term memory, or weakening of particular parts of the body such as occurs in stroke (Niedermeyer & Da Silva, 2005; Tatum, 2007).

This image has been removed by the author of this thesis for copyright reasons.

Figure 7.1: Electroencephalography (EEG) Equipment (Berber, 2011; Murph, 2007)

The left image of Figure 7.1 shows the standard EEG equipment with 256 and 64 electrodes (channels) (Berber, 2011). The wireless EEG equipment is photographed in the right image. The EEG signals are sourced from billions of neurons (approximately $10^{10}$of them) (Murph, 2007). Neurons are electrically charged (or "polarized") by sodium and potassium ion pumps that operate across their membranes. When a neuron receives an action potential signal from other neurons, many neurons emit ions simultaneously and inhibit neighbour neurons in a wave-like fashion. Therefore, when a wave of ions intercepts the electrodes on the scalp, it repels or attracts electrons on the metal of the electrodes. As metals are strong conductors of electrons, the repulsion or attraction induces a voltage between two electrodes, which can be detected by a voltmeter. The EEG is a record of these changing voltages over a specified time (Yamazaki & Tanaka, 2007).

EEG was pioneered in 1842, when a physician practicing in Liverpool attempted to measure the electrical activity of cerebral hemispheres of rabbits and monkeys. At that time, electrodes needed to be directly inserted into specific brain regions. A prototype of modern EEG was tested on dogs in 1912, and applied to humans too later to investigate seizures (Swartz & Goldensohn, 1998).

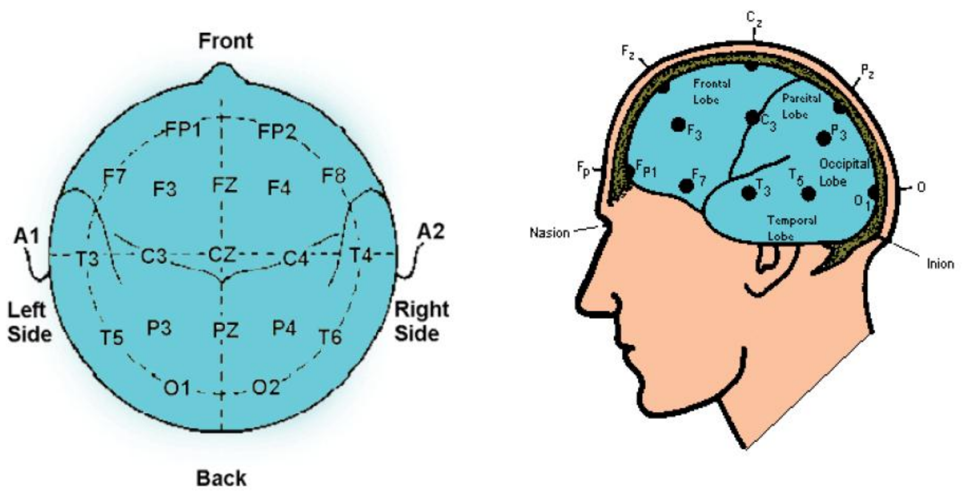


Figure 7.2: Spatial Positioning of the EEG Electrodes over the Frontal, Central and Parietal Lopes (Ferreira, Almeida, Georgieva, Tomé, & Silva, 2010)

Figure 7.2 illustrates a sample (21 channels) of EEG location over the frontal, central and parietal lobes of the brain. EEG is generally explained in terms of rhythmic movement and is divided into five frequency bands (Lotze et al., 1999). The characteristics of these bands are described below:

(1) Delta Waves

Delta waves are brainwaves of 4Hz or less. These waves, generated by the thalamus, convey cross-referenced information from several sensory systems and transmit it to the cerebral cortex. In adults, Delta waves are presented only during sleep, whereas in infants they are present during sleeping and waking periods.

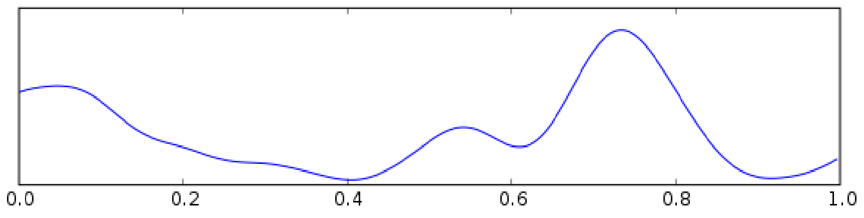


Figure 7.3: An illustration of EEG in Delta oscillation(Gamboa, 2005)

(2) Theta Waves

The frequency of theta waves ranges from 4 to 7 Hz. Theta waves are generated by the hippocampus, which is vital for memory formation and spatial movement. Theta waves appear during dreaming and REM (Rapid Eye Movement) sleep, and also during deep meditation. Together with delta waves, theta waves characterize the EEG of sleeping adults.

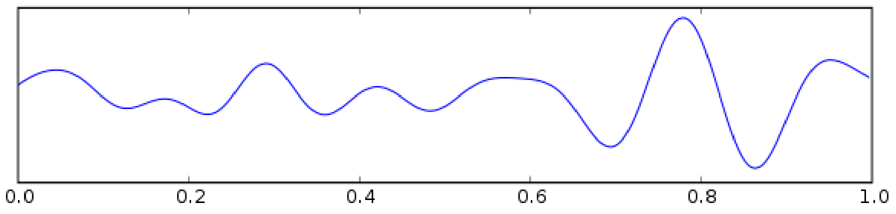


Figure 7.4: An illustration of EEG in Theta oscillation (Gamboa, 2005)

(3) Alpha Waves

Alpha waves possess frequencies ranging from 8 to 12Hz. Thoughts, reasoning, judgment, associative thinking and fantasizing are all associated with alpha movement. Alpha waves appear during relaxation and alpha movement is immediately provoked by closing the eyes while one is awake.

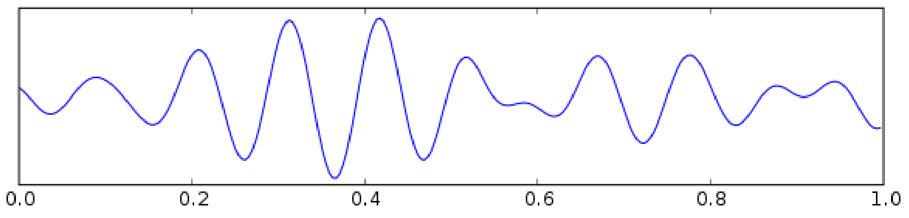


Figure 7.5: An illustration of EEG in Alpha oscillation (Gamboa, 2005)

(4) Beta Waves

The frequency of beta waves ranges from12to 30Hz. Alpha and beta waves collectively characterize the waking EEG of adults. While alpha waves signify a relaxed state, beta waves are associated with readiness and attentiveness. They are also linked to general movement and logical reasoning, and also to motor behaviour such as active movement.

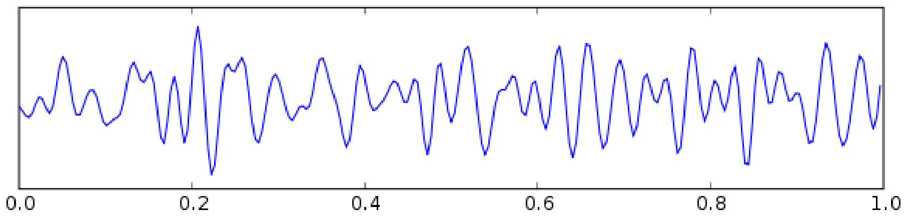


Figure 7.6: An Illustration of EEG in Beta Oscillation (Gamboa, 2005)

(5) Gamma Waves

Gamma waves possess frequencies of 40Hz or higher. These appear during sophisticated mental activities such as insight seeking and problem solving. Spontaneous eruptions of gamma movement accompany quick sparks of vision or perception (so-called “a-ha” moments). Children with higher than average attention spans, reasoning abilities and language capabilities exhibit strong gamma wave movement, whereas adults with high gamma wave activity possess above-average intelligence. High gamma wave activity can also arise from meditation.

Dominant band frequency has been used as a diagnostic tool; in particular, alpha band activity in the temporal or frontal lobes.

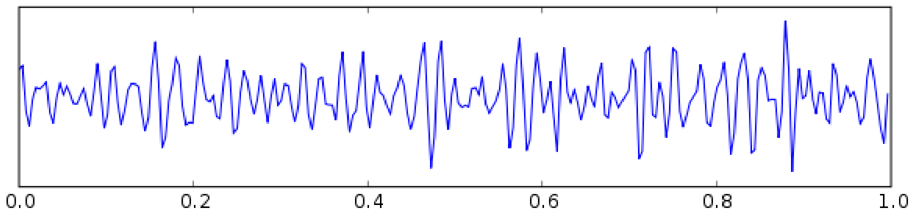


Figure 7.7: An Illustration of EEG in Gamma Oscillation (Gamboa, 2005)

7.2 EEG Application

During the past decade, neurological advances have clarified that a direct interface exists between the human brain and an artificial system, known as the Brain Computer Interface (BCI). Although the BCI is a feasible concept, considerable research and development is required before these technologies can be put to everyday practical use (Berger et al., 2008).

There have been various data analysis and machine learning techniques successfully used with EEG and BCI application such as independent component analysis (ICA) for P300 detection (Xu et al., 2004), 3D game based BCI utilizing the steady-state visual evoked potential (SSVEP) and power-spectrum estimation methods for feature extraction in a series of offline classification tests (Lalor et al., 2005), EEG-based Brain-Computer Interfaces for Control and biometry utilizing Principle Component Analysis (PCA) for noise reduction and Support Vector Machine (SVM) feature space mapping functions with Radial Basis Function (RBF) for the nonlinear SVM version (Ferreira et al., 2010; Marcel & Millán, 2007; Palaniappan & Mandic, 2007).

Figures 7.8 and 7.9 respectively illustrate the use of BCI in controlling a Honda robot (Binns, 2009) and a mind-controlled wheelchair developed by Toyota and RIKEN Brain Science Institute in Japan (Abolfathi, 2009).

The principle of BCI is prototyping of brain activity fluctuations and recording them into some form of actuation or command at an aimed output, such as a computer interface or a robotic system. BCI research is presently motivated chiefly by the potential advantages to those with severe motor disabilities, such as “brainstem stroke, amyotrophic lateral sclerosis or severe cerebral palsy.” (P.

Goel, Liu, Brown, & Datta, 2008; Marcel & Millán, 2007; Yamazaki & Tanaka, 2007)

Figure 7.8: BCI -controlled Robot by Honda, Japan (Binns, 2009)

The most effective means of analysing the physiological activity of the brain is recording the EEG signals from the cortex, whose sources are the action potentials of the cerebral nerve cells. This is because cortex EEG signals comprise waves spanning the 0-60 Hz frequency band and distinct brain activity movements can be recognized based on the recorded fluctuations. For example, signals within the delta band (below 4 Hz) correspond to deep sleep, theta band (4-8 Hz) signals typify a dreamlike state, alpha frequencies (8-13 Hz) correspond to relaxed states with closed eyes, beta frequencies (13-20 Hz) are associated with waking activity, while gamma frequencies (40 Hz and higher) characterise mental activities such as perception and problem solving (Niedermeyer & Da Silva, 2005).

Figure 7.9: Mind Controlled Wheelchair Developed by Toyata and RIKEN Brian Science Institute in Japan (Abolfathi, 2009)

Present research interest lies in acquiring knowledge hidden in the EEG signals as well as developing EEG-based implementations. BCIs for motor control and biometry are the latest EEG-based applications in computational neuro-engineering. The BCI concept initiated from observations of alpha band activity in a subject performing real and imaginary movement, for which EEG data are not significantly different. Since this discovery, EEG has been extensively applied in BCI competitions (Piyush Goel, Liu, Brown, & Datta, 2006; Xu et al., 2004) and mind-controlled machines, such as robot, wheelchair and mind games (Abolfathi, 2009; Binns, 2009). Of equal interest is EEG data analysis, because the efficiency and accuracy of classification and pattern recognition methods are necessary not only in BCI but also for improved understanding of the remarkable information processing abilities of mammalian brains.

EEG-based biometry has become a new paradigm by which to investigate biometric systems (Ferreira et al., 2010; Marcel & Millán, 2007; Palaniappan & Mandic, 2007). Because individual human brain wave patterns are unique, EEG can be used as a personal identification tool or as an alternative verification system. A few studies have investigated the efficacy of brain signals in identifying individuals (Ferreira et al., 2010). EEG identification aims to extract the identity of a given individual from a restricted list of persons (one from many), whereas verification attempts to validate or reject an individual’s claimed identity (one to one matching; (Marcel & Millán, 2007). The identified person is subjected to a incitement (generally visual or auditory) for a certain period of time and the EEG signals originating from a number of electrodes arranged around the subject’s head are gathered and entered into the biometry system. Initial tests have shown that the “type of stimulus (for example mental task, motor task, image presentation or a combination of these) is crucial for reliable extraction of personal characteristics. It seems that some mental tasks are more appropriate than others. At the same time, experiments with combinations of stimuli appear to be more advantageous for the personal uniqueness of the EEG patterns” (Ferreira et al., 2010).

7.3 Spiking Neural Networks Applications on EEG

Studies in which spiking neural networks (SNN) have been used for EEG analysis, have achieved remarkable success in categorization tasks, compared to other alternative methodologies. Recently, Goel and colleagues (P. Goel et al., 2008; Piyush Goel et al., 2006) presented a classification scheme of continuous EEG data using SNN, which engages pulse and rate encoding to transform data to spike trains.

The data set of this study was provided by EEG signals recorded during spell checking, downloaded from the BCI Competition website. The data comprise two classes: the target group (P300: slow positive potential which builds up over approximately 250-500 ms), and a non-target group (potential build up over 250-500 ms). Feature selection and extraction employed a Wavelet Transform to eliminate the noise and to obtain the low frequency signal. This system utilises the LIF model as nodes in a multi-layered structure (2 layers) to form a weak classifier; each node contributes to its own classification in generating its overall classification. Layer One includes 2 networks; the input network (LIF neurons receiving EEG input), and bias network (a single network with strong connection weight). The secondly, termed the γ-neuron, receives as input the output from the first layer, and spikes whenever synchrony exists within the two networks. The higher the number of γ-spikes, the more similarity between the input signal and the bias current. Goel and colleagues compared the performance of this approach with that of two others; Support Vector Machine (SVM), and log sampling, low-pass filtering and Continuous Wavelet Transform (CWT) for pre-processing and filtering EEG input data, followed by Linear Discriminate Analysis. The SNN-based approach improved (incomparison to SVM and CWT) the classification accuracy to 94.7% and 83.68% for target and non-target groups, respectively.

Goel et al. (2007) commented that their approach could be made more accurate and flexible if a biological stimulus response was built into the network, more parameters were added or the existing probability parameters in SNN adjusted. A couple of years later, Ghosh-Dastidar and Adeli (Ghosh-Dastidar & Adeli, 2009) suggested a method for improving SNNs performance in categorising EEG data and detecting epilepsy and epileptic seizures. Three SNN training algorithms were investigated and compared; Spike Propagation (SpikeProp), Quick Propagation (QuickProp) and Resilient Propagation (Rprop). The data sets for assessing the performance of the training algorithms were XOR, Fisher Iris benchmark and EEG data. Three aspects of computational efficiency and classification accuracy were investigated. Epilepsy and epileptic seizure detection data were divided into 3 classes; namely, healthy individuals, relapsed epileptic individuals and epileptic individuals during an attack. During processing of XOR and Fisher Iris data, RProp yielded the highest classification accuracy (92.5%), especially for training datasets. RProp was thus selected as the SNN training algorithm and for generating EEG data classification networks. The experiments of Ghosh-Dastidar and Adeli (2009) used a feed forward architecture (input layer, hidden layer and output layer) to achieve average classification accuracy around 90.7%. However, as mentioned by the authors, introducing spike time dependent and plasticity (STDP) into the SNN training algorithm would enhance the biological plausibility of the algorithm, similarly to Hebbian rules.

A study aimed at processing rat EEG data using a reservoir approach was proposed in 2008 and published in 2009 (Buteneers, Schrauwen, Verstraeten, & Stroobandt, 2009). This study showed that introducing a reservoir for real-time epileptic seizure detection in 4 channels of rat EEG markedly improved model performance. In this experiment, 200 Leaky-Integrator neurons were deposited in the reservoir. From 20%-80% of data, two classes, detection seizure and tonic seizure, were extracted by training and analysis respectively. The reservoir was found to enhance the detection time performance to approximately 85% for 0.5 seconds attack time and 85% for 3 seconds tonic seizure time. Because this study utilised EEG rat data which engaged only 4 channels, limited frequency information was available.

The above studies paved the way for the development of the Liquid State Machine, a reservoir apparatus that was applied to EEG data for the first time in 2011 (Nuntalid, Dhoble, & Kasabov, 2011).

Figure 7.10 shows the six time steps of the Evoked Potential Duration (EPD) in a single subject. EPD transforms the original EEG signal into a power spectrum that changes over time in response to varying stimulus. The EPD in the same subject undergoing similar stimuli over nine time steps is shown in Figure 7.11. Comparing Figures 7.10 and 7.11, we observe that increasing the number of time steps, enables more information to be extracted from EEG data. Therefore, processing information in vector format may not be efficient for complex EEG pattern recognition.

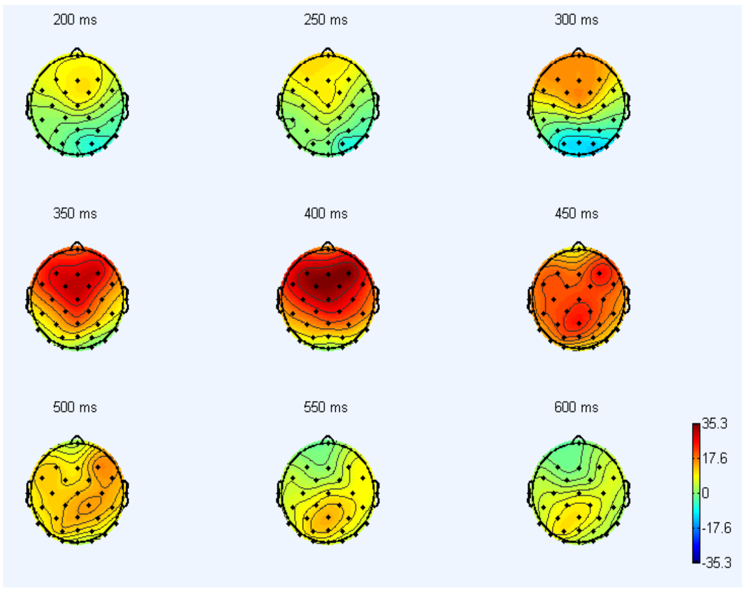


Figure 7.10: Evoked Potential Duration (EPD) of a subject responding to a stimulus

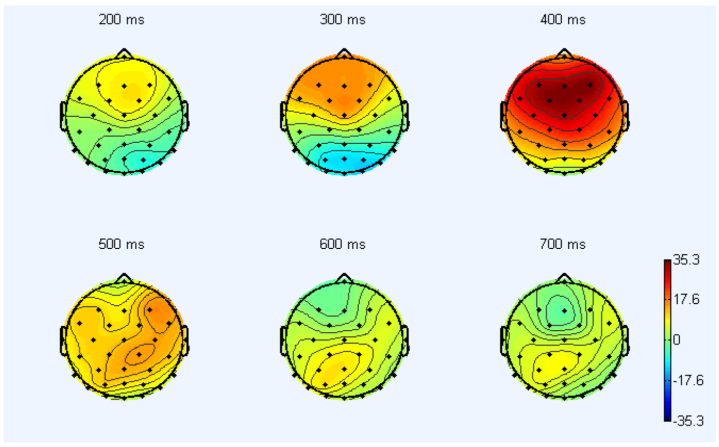


Figure 7.11: Illustration of the EPD of a subject responding to a stimulus in nine time steps

Figure 7.10 and 7.11 were produced utilizing MATLAB (R2011) with EEGlab library (Arnaud Delorme & Scott Makeig, 2004).

7.4 Conclusion

From previous SNN studies, it appears that the development of probabilistic spiking neural models and spatiotemporal pattern recognition of human EEG data other than that arising from epileptic seizure is complex and challenging. Further investigation into these aspects is the aim of this study. In particular, EEG-based spatiotemporal pattern recognition may improve the performance of SNN, lead to a better understanding of the human brain and make significant contributions to neuroscience.

Chapter 8

Novel Algorithm for EEG Transformation to Spikes

This chapter discusses the possible encoding methods applicable to EEG data and to hardware development of a BCI with a fast processing mechanism. Section 8.1 focusses on the Paralleled Spike Encoding Function, which is well designed and implemented in hardware but is less practical for software implementation. A more software-friendly approach, the Finite Impulse Response filter (FIR), is discussed in Section 8.2. FIR underlies the Ben Spike Encoder Algorithm (BSA) presented in Section 8.3. The last section concludes the chapter.

In this study, BSA is selected as an encoding method for EEG because it can be practically implemented in both hardware and software. The contribution involves novel application of the existing BSA encoding scheme on EEG data, and partially corresponds to the forth research question.

8.1 Paralleled Spike Encoding Function

Torikai and Nishigami proposed a spiking neural model and a corresponding encoding function, termed Chaotic Spiking Neuron (CSN) and Paralleled Spiking Encoding respectively (Torikai & Nishigami, 2009). CSN was inspired by the reliable encrypting mechanism of the mammalian spiral ganglion cell. Essentially, CSN is a small network of LIF neurons, whose mechanism is depicted in Figure 8.1. CSN includes two units, the Base Unit and the Neural Unit, both operating under an electrical circuit mechanism similar to the LIF model. The current $I(t)$, input to both units, can vary in waveform. The capacitor voltage of base unit B is increased by an amount depending on the integration of $I(t)$ and internal current I_0. When B exceeds the firing threshold V_β , it is reset to zero. The neural unit works similarly to the base unit, where V_i (*i*=1, 2, 3..., *N*) denotes the membrane potential. The V_α is the neural threshold of neuron increased by an amount dependent on $I(t)$ + integration of $I(t)+I_0$ + output of base unit. If $V_i(t)$ fires, its potential is reset to $-B(t)$.

The CSN encoding function imitates the mechanism of ganglion cells by the spike density of summed trains of spikes $Y(t)=\sum_i Y_i(t)$ (*i*=1, 2, 3..., *N*), which is equivalent to the periodic sinusoidal waveforms received by inner hair cells. In this research, we show that CSNs can encode not only periodic sinusoidal input but a wide variety of inputs such as constant, random, and the non-periodic waveforms characteristic of human EEG data.

CSN is designed primarily for hardware implementation, which is very simple and convenient. In contrast, it is unsuitably inflexible for software implementation because swapping the potential of the base unit is outside of the LIF rules which the model need to be changed for software development. Therefore, to adapt CSN to software implementation, a new neural model for the base unit must be constructed.

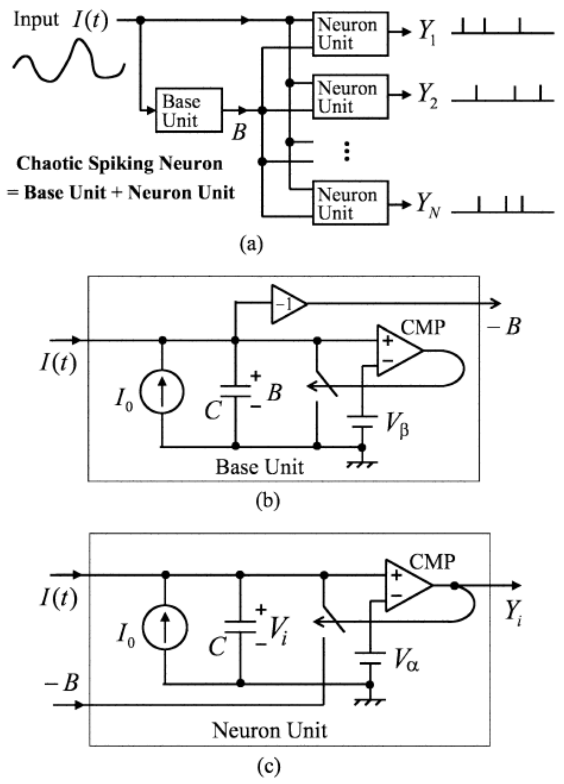


Figure 8.1: The Chaotic Spiking Neuron: CSN (Torikai & Nishigami, 2009)

8.2 Finite Impulse Response filter (FIR)

A causal filter with a finite impulse response calculates the output signal $y[n]$ as a weighted addition of the real and M earlier input samples $x[n]$, where M is the order of the filter. Consequently, the link between input and output can be modelled by the following variance equation 8.1:

$$y[n] = \sum_{k=1}^{M} b_k x[n-k] \tag{8.1}$$

The result $y[n]$ is the discrete convolution of $x[n]$with a (finite) impulse response.

$$h[n] = \begin{cases} sb_k & k = 0, 1, \dots., M \\ 0 & otherwise \end{cases} \tag{8.2}$$

If the filter order is M, then the impulse response has $M + 1$ coefficients.

The z-transform $H[z]$ of the impulse response $h[n]$ of a fundamental FIR filter has M zeros, whose positions are found by the coefficients b_k and an M-fold pole at the origin (i.e., $z = 0$). Thus, an FIR filter is inherently stable. Nevertheless, the constancy condition, related to the location of the poles, must also be satisfied i.e. $\sum_{k=0}^{M} |b_k| < \infty$. This condition is automatically fulfilled for FIR. Because the transfer operation of the filter is entirely calculated by the location of the zeros, FIR filters are known as 'all-zero filters'.

Linear-Phase Filter property is another property unique to FIR filters. It can be demonstrated that if the impulse response $h[n]$ of the filter is symmetric, i.e. $h[M-n] = h[n] \quad n = 0, 1, \dots\dots, M$ (even symmetry, cosine terms only) or $h[M-n] = -h[n] \quad n = 0, 1, \dots\dots, M$ (odd symmetry, sine terms only), the phase function $\varphi(\theta)$ of the system's frequency response

$H\left(e^{j\theta}\right) = \left|H\left(e^{j\theta}\right)\right| e^{j\angle H\left(e^{j\theta}\right)}$ is linear. If the symmetry condition is fulfilled, the following holds:

$$\varphi(\theta) = \angle H\left(e^{j\theta}\right) = \pm\frac{k\pi}{2} - \frac{M}{2}\theta \tag{8.3}$$

In this instance, the filter has a continuous group delay of

$$\tau_g(\theta) = -\frac{d\varphi(\theta)}{d\theta} = M/2 \text{ Samples}$$

or

$$\tau_g(\omega) = -\frac{d\varphi(\omega)}{d\omega} = M/(2f_s) \text{ Seconds}$$

8.3 Ben Spike Encoder Algorithm (BSA)

Almost all real-world data signals, and especially EEG signals, are analogue. To use spiking neural networks to process analogue values, we need to transform the signals to spike-trains, in such a way that errors and information loss induced by the transformation process are minimised. In this study, we incorporate the BSA spike encoding scheme to transform EEG data into trains of spikes. BSA is a very fast and stable encoder. Moreover, because it is based on FIR, the encoded spike trains can be back-transformed into their original waveforms. In this way, one can check the extent to which the encoded spike trains mimic the original wave. A hardware implementation of BSA has been shown to produce superior results to other encoding methods (such as population encoding) in speech recognition (Benjamin Schrauwen, D'Haene, Verstraeten, & Campenhout, 2008) .

To date, this encoding scheme has been applied to complete data only. Because EEG signals exist in the frequency as well as the time domain, it has been conjectured that BSA encoding can convert EEG signals into spikes. The main advantage of BSA is that the frequency and amplitude characteristics are more tractable than those of the HSA (Hough Spiker Algorithm) spike encoding scheme, which has been employed in a Robot (Garis, Korkin, & Fehr, 2001). Due to the smoother threshold optimization curve, BSA is also less susceptible to changes in the filter and the threshold. Studies have shown that BSA offers an improvement of 10dB-15dB over the HSA spike encoding scheme (B. Schrauwen & Van Campenhout, 2003).

Indicated below are samples that show how FIR filters transform a waveform into spikes. The original waveform is subtracted iteratively by FIR filters with a 1 bit shift per iteration, until all results are zeros. When the first FIR bit exceeds the first bit of the original wave, a spike is emitted (1). In contrast, if the first FIR bit is equal to or less then the first bit of the original wave, no spike is emitted (0).
-Assume analogue waveform values of {1 5 13 15 7 7 6 2 9 5 -2}, and
FIR filter values of {1 4 9 5 -2}.

Iteration 1:   Spikes is {1}.

1	5	13	15	7	7	6	2	9	5	-2
1	4	9	5	-2						

Iteration 2: Spikes is {1 1}.

0	1	4	10	9	7	6	2	9	5	-2
	1	4	9	5	-2					

Iteration 3: Spikes is {1 1 0}.

0	0	0	1	4	9	6	2	9	5	-2
			1	4	9	5	-2			

Iteration 4: Spikes is {1 1 0 1}.

0	0	0	0	0	0	1	4	9	5	-2
				1	4	9	5	-2		

Iteration 5: Spikes is {1 1 0 1 0}.

0	0	0	0	0	0	1	4	9	5	-2
					1	4	9	5	-2	

Iteration 6: Spikes is {1 1 0 1 0 0}.

0	0	0	0	0	0	1	4	9	5	-2
						1	4	9	5	-2

Iteration 7: Spikes is {1 1 0 1 0 0 1} (the original spiking pattern).

0	0	0	0	0	0	1	4	9	5	-2
						1	4	9	5	-2
0	0	0	0	0	0	0	0	0	0	0

Therefore, the result {1 1 0 1 0 0 1} is obtained from original waveform.

Table 8.1: Convolution function for transforming train of spikes into original waveform

Spikes	**Iterations**										
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	1	4	9	5	-2						
1		1	4	9	5	-2					
0			0	0	0	0	0				
1				1	4	9	5	-2			
0					0	0	0	0	0		
0						0	0	0	0	0	
1							1	4	9	5	-2
Result	1	5	13	15	7	7	6	2	9	5	-2

The original waveform can be recovered from the spikes using the same FIR filter and employing the convolution function, as shown in Table 8.1. Given that the spike train is {1 0 0 1 0 1 1} and the FIR filter is {1 4 9 5 -2}, the mask filter, which must be the same length as the FIR filter, is {0 0 0 0 0}.

The spike train is first bit-reversed aligned into the first column. The FIR filter is inserted into the first row and shunted down the rows, shifting by 1 bit from left to right each time, until the end of the spike train is reached. If the value of Spikes for a given row is 1, apply the FIR filter; otherwise, apply the mask filter. Finally, sum the values in each column to retrieve the original wave {1 5 13 15 7 7 6 2 9 5 -2}, as shown in the last row of Table 8.1.

The above analysis highlights the importance of a stable filter. In BSA the incitement is calculated from the spike train by equation 8.4, similarly to HSA.

$$s_{est} = (h \times x)(t) = \int_{+\infty}^{-\infty} x(t-\tau)h(\tau)d\tau = \sum_{k=1}^{N} h\left(t - t_k\right) \tag{8.4}$$

In table 8.1, the spike train is {1 0 0 1 0 1 1} and the FIR filter is {1 4 9 5 -2}. If t_k represents the neuron firing time, *h(t)* denotes the linear filter impulse response, t_k is the set of firing times of the neuron, then the neuron spike *x(t)* is determined as

$$x(t) = \sum_{k=1}^{N} \delta(t - t_k)$$

(8.5)

BSA additionally calculates two error metrics for each time τ, as revealed in Table 8.2. The first error metric is

$$\sum_{k=0}^{M} abs\big(s(k+\tau) - h(k)\big) \tag{8.6}$$

while the second is

$$\sum_{k=0}^{M} abs\big(s(k+\tau)\big) \tag{8.7}$$

If the first error metric is smaller than the second minus threshold, then eject a spike and subtract the FIR filter from the input wave (if not, do nothing). The above error metrics calculation smoothes the frequency and amplitude in BSA, rendering those features less vulnerable to changes in the filter and threshold, in contrast to the HSA algorithm (Benjamin Schrauwen, Verstraeten, & Campenhout, 2007).

The pseudo-code of BSA is displayed in Table 8.2. The binary spikes output is readily transformed to spike time of EEG data by dividing output(i)=1 by (sample rate of EEG/real time of recoding) , ignoring output(i)=0 because, in general, only the times of spikes are required in forming the spike trains.

Table 8.2: A pseudo-code of BSA algorithm on EEG data

BSA Algorithm

```
for (i=1 to size (input_signal))
    error_matrix_1=0
    error_matrix_2=0
for (j=1 to size(FIR_filter))
if ((i+j-1) <= size(input_signal))
            error_matrix_1+= abs(input_signal(i+j-1) – FIR_filter(j))
            error_matrix_2+= abs(input_signal(i+j-1)
end if
 end for
if (error_matrix_1 <= (error_matrix_2-BSA threshold))
        output(i)=1;
for (j=1 to size(FIR_filter))
if ((i+j-1) <= size (input_signal))
input_signal(i+j-1) - = FIR_filter
end if
end for
else
        output(i)=0
end if
end for
```

In this specific synthetic dataset, the Finite Impulse Response (FIR) filter size is set to 20 and the BSA threshold to 0.86. These values generate the lowest

Euclidian distance (14.3) between the original EEG signal and the re-transformed spikes recovered from the convolution function using the same FIR filter. The dependence of Euclidean distance on threshold is shown in Figure 8.2. In the original BSA, the optimal threshold was found by minimising the SNR (signal noise ratio), to enable direct comparison with the HSA algorithm.

Convolving the spike train *x(t)* with a discrete FIR filter, Eq.8.5 becomes

$$o(t) = (h \times x)(t) = \sum_{k=0}^{M} x(t-k)h(k)$$

(8.8)

where *M* represents the number of filter taps. (Benjamin Schrauwen et al., 2007).

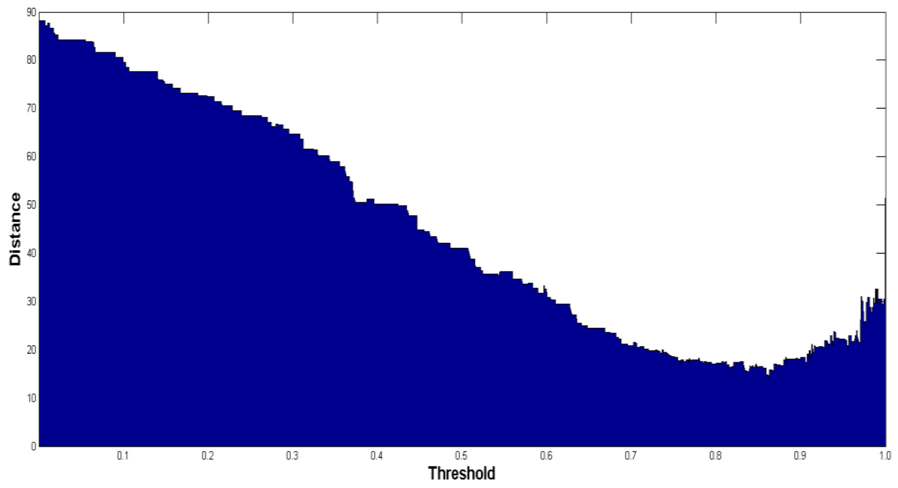


Figure 8.2: Ben Spike Encoder Algorithm (BSA)

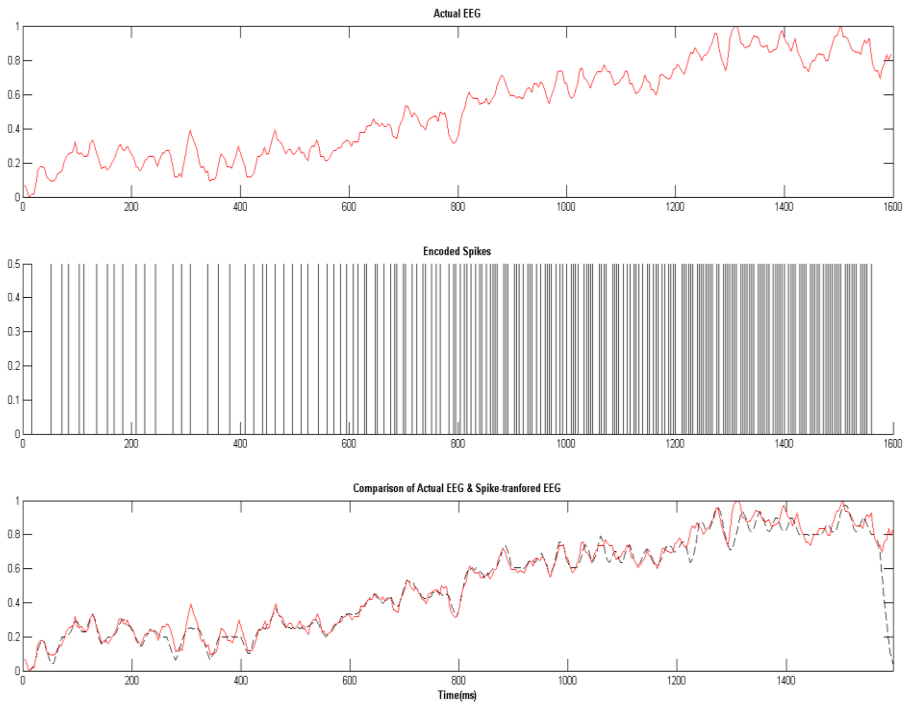


Figure 8.3: EEG signal, spike representation and actual one channel EEG signal

The *y* axis of Figure 8.2 is the Euclidian distance between the original EEG signal and the signal recovered from the spikes. The *x* axis is threshold value of BSA algorithm, which ranges from 0.0 to 1.0.

In Figure 8.3 a synthetic EEG signal, created by adding noise to a sine wave (duration 1600 ms), is compared with a true single-channel EEG signal (duration 1.6 s). Synthetic and true signals are displayed in the top and bottom panels of 8.3, respectively. The centre panel is the spike representation of the top figure attained from BSA. In the bottom panel, the true single-channel EEG signal has been overlaid with another signal (dashed lines) showing the EEG signal rebuilt from the BSA-encrypted spikes. The clear resemblance between the two signals indicates that the modified BSA is suitable for EEG spike encryption.

8.4 Conclusion

BSA may be suitable for transforming EEG into spike trains due to the following advantages of this method over other existing methods:

(1) It presents as a fast, stable and practical encoder because it has hardware support yet is readily implemented in software.

(2) Because BSA is based on FIR, the encoded spike trains can be back-transformed to their original form to determine how closely the encoded spike trains match the original wave.

(3) It requires fewer data vector inputs than other approaches (c.f. population encoding, detailed in Chapter 2), enabling much faster and more efficient processing; this is a primary advantage of SNNs in general.

(4) The only parameters that require optimizing are FIR filter size and BSA threshold. In addition, these parameters are relatively robust, ensuring a stable system.

Chapter 9

Proposed Architectures for EEG Spatio-temporal Pattern Recognition

Section 9.1 of this chapter describes a proposed architecture for EEG Spatio-temporal Pattern Recognition on Stochastic Precise-time Spike Pattern Association Neuron (SSPAN). Section 9.2 explains the architecture for online EEG Spatio-temporal Pattern Recognition Utilizing the DepSNN and Section 9.3 describes the architecture for online EEG Spatio-temporal Pattern Recognition Utilizing epSNNr. In each of these architectures, the encoder is BSA (described in Chapter 8) and the neural models are LIF and stochastic (described in Chapter 3). The chapter concludes with Section 9.4.

This chapter consist of the above mention three main contributions specifically designed spatiotemporal EEG processing. These contributions are in line with the fourth and last research question.

The architectures proposed in this chapter will be investigated on two real world EEG datasets; the Audio–Visual Stimuli Perception EEG dataset, whose details are presented in Chapter 10, and the P300 EEG dataset, which is discussed in Chapter 11.

In some states, a portion of each dataset may be required for parameter adjustment before the feasibility of each method can be assessed on the full datasets.

9.1 Proposed Architecture for EEG Spatio-temporal Pattern Recognition on Stochastic Precise-time Spike Pattern Association Neuron (SSPAN)

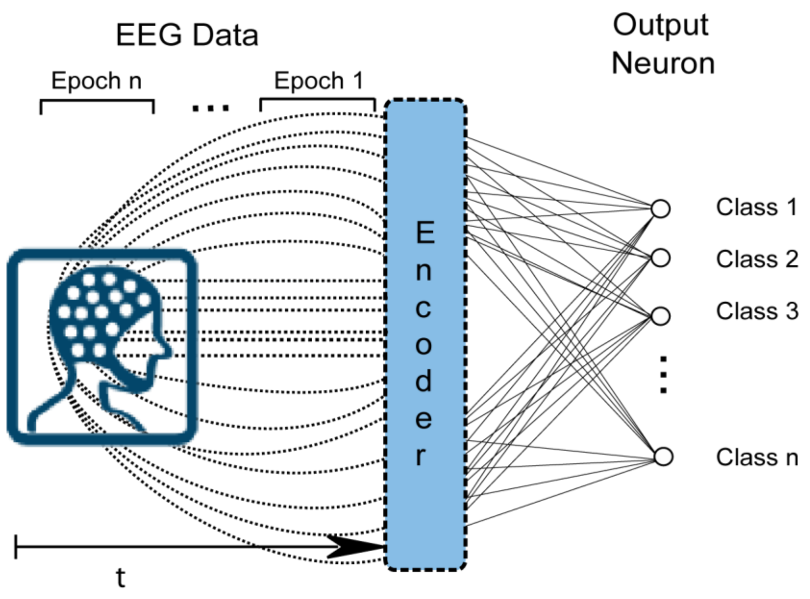


Figure 9.1: SSPAN: Stochastic Precise-time Spike Pattern Association Neuron for EEG

A recent study mentioned that neurons respond at precise times to fluctuating current injections, leading to peaks in ensemble firing rate. The structure of spike events provides a compact description of the neural response (Toups, Fellous, Thomas, & Sejnowski, 2012).

Hence, in this architecture, SSPAN is employed to investigate EEG datasets, as explained in Chapter 4. This simple architecture is mainly focussed on the spatiotemporal learning process and spike timing.

In the first step, each EEG data channel is transformed into trains of spikes by the spike encoding methods (BSA) described in Chapter 8.

The spike input signals are then injected to output neurons. The number of output neurons is based on the number of available classes in the dataset. An output neuron is connected to every incoming input signal from a particular class.

Because EEG data are both noisy and stochastic, the stochastic SNN models are beneficial to in the investigation and might produce some interesting results and knowledge discovery.

Four SNN models are utilized for output neurons; the deterministic LIF model, Stochastic Noisy Reset model (NR), Stochastic Step-wise Noisy Threshold model (ST) and Continuous Stochastic Threshold Model (CT).

9.2 A Novel Architecture for Online EEG Spatio-temporal Pattern Recognition Utilizing the Dynamic Evolving Spiking Neural Network (DepSNN)

The major advantage of Evolving Spiking Neural Networks (eSNN) is that the trained network responds to newly introduced samples without the need for retraining (Hamed, Kasabov, & Shamsuddin, 2012).

Currently, however, ESNN cannot handle multiple spikes reaching the same synapse at different times. The need to process such data has inspired the development of DepSNN, an extension of eSNN that can handle spatiotemporal input.

The architecture presented in this section enables DepSNN to be investigated on a real world EEG dataset. The results provide new insights into spatiotemporal learning. This method has been recently published in the journal *Neural Networks* (Kasabov, Dhoble, Nuntalid, & Indiveri, 2012).

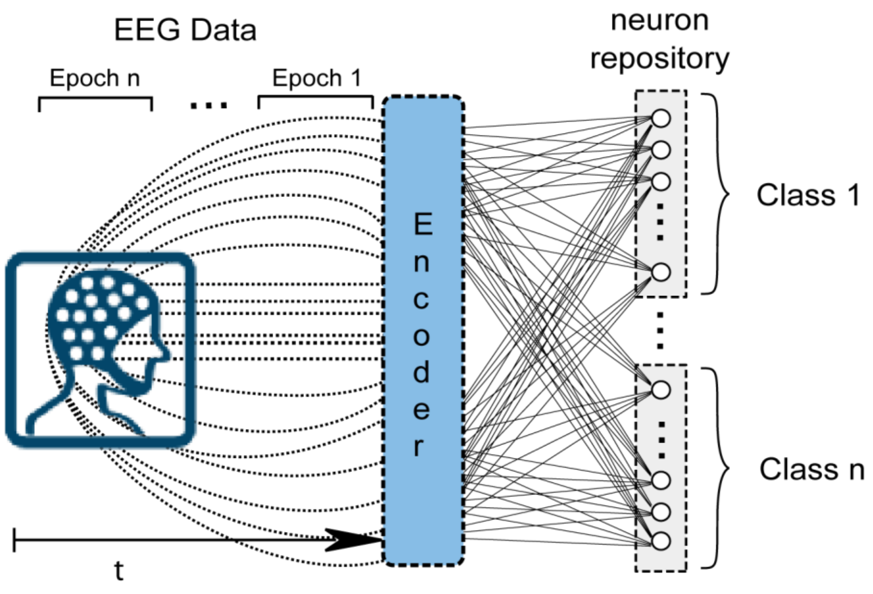


Figure 9.2: DepSNN: Dynamic Evolving Probabilistic Spiking Neural Network for EEG

Figure 9.2 illustrates the architecture of DepSNN for EEG data processing. EEG data is first encoded into spike trains by the BSA spike encoding algorithms. To define the initial synaptic weights, the spike input signals are ranked from the time of first spike of each EEG channel Details of the calculation are presented in Chapter 5.

The SDSP rule then adjusts the synaptic weights based on incoming spikes. The mapping of the output spikes onto the desired class label is learned using a one-pass learning mechanism. A repository of trained output neurons is also created in the learning process. A new neuron is trained and compared to one stored in the neuron repository. If the new neuron displays similar synaptic weights patterns over time to an existing neuron (within an adjustable similarity threshold), the two neurons will be merged. Otherwise, the new neuron is added to the neural repository.

The feasibility and performance of DepSNN on EEG data is assessed for the five DepSNN models described in Chapter 5; namely, DepSNNm, DepSNNs, NR-DepSNNs, ST-DepSNNs, and CT-DepSNNs.

9.3 A Novel Architecture for Online EEG Spatio-temporal Pattern Recognition Utilizing the Evolving Probabilistic Spiking Neural Network Reservoir (epSNNr)

The structure of the Evolving Probabilistic Spiking Neural Network Reservoir (epSNNr) for categorization of spatiotemporal data founded on probabilistic reservoir computing pattern is explained in Chapter 6. In this chapter, epSNNr is investigated for its efficacy on real world EEG data. The architecture of the system is shown in Figure 9.3.

The state vector in Figure 9.3 is specified only for non-SNN classifiers (readout functions), because the SNN classifier of DepSNN can receive time-dependent spike information directly from the spatiotemporal filter.

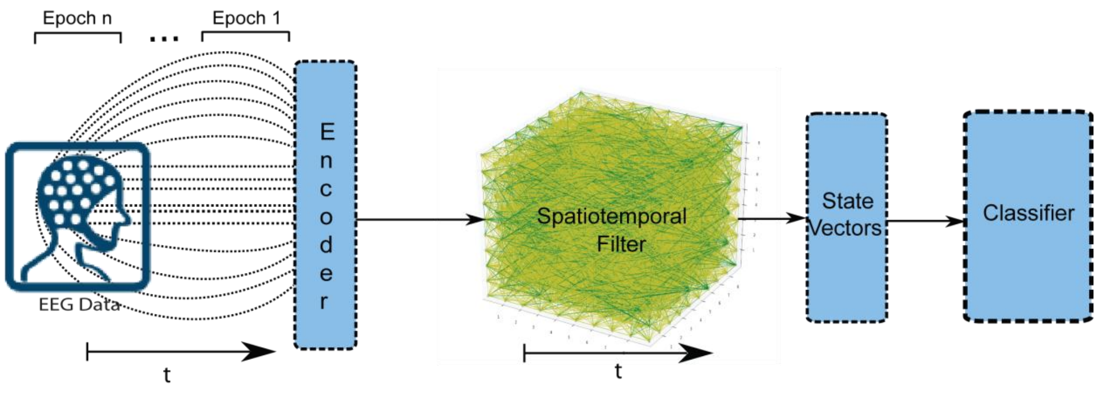


Figure 9.3: epSNNr: Evolving Probabilistic Spiking Neural Network Reservoir for EEG

In the first processing step, each EEG data channel is transformed into trains of spikes by the BSA spike encoding methods. The spike trains are then dispersed into the spatiotemporal filter, which employs the Liquid State Machine (LSM).

In this architecture, the deterministic SNN model (LIF) and stochastic models (Noisy Reset model, Step-wise Noisy Threshold model and Continuous Stochastic Threshold) generate the liquid for the spatiotemporal filter.

The data in the state vectors are fed into a readout unit (the processing of state vectors is detailed in Chapter 6). In this study, Naïve Bay and Multi-Layer Perception (MLP) are employed for non-SNN classifier exploration. DepSNN or SSPAN might improve the performance of this architecture, because the spatiotemporal filter can inject its output directly into DepSNN or SSPAN, unlike the situation for non-SNN classifiers (Nuntalid et al., 2011) or eSNN (Schliebs, Nuzly Abdull Hamed, & Kasabov, 2011).

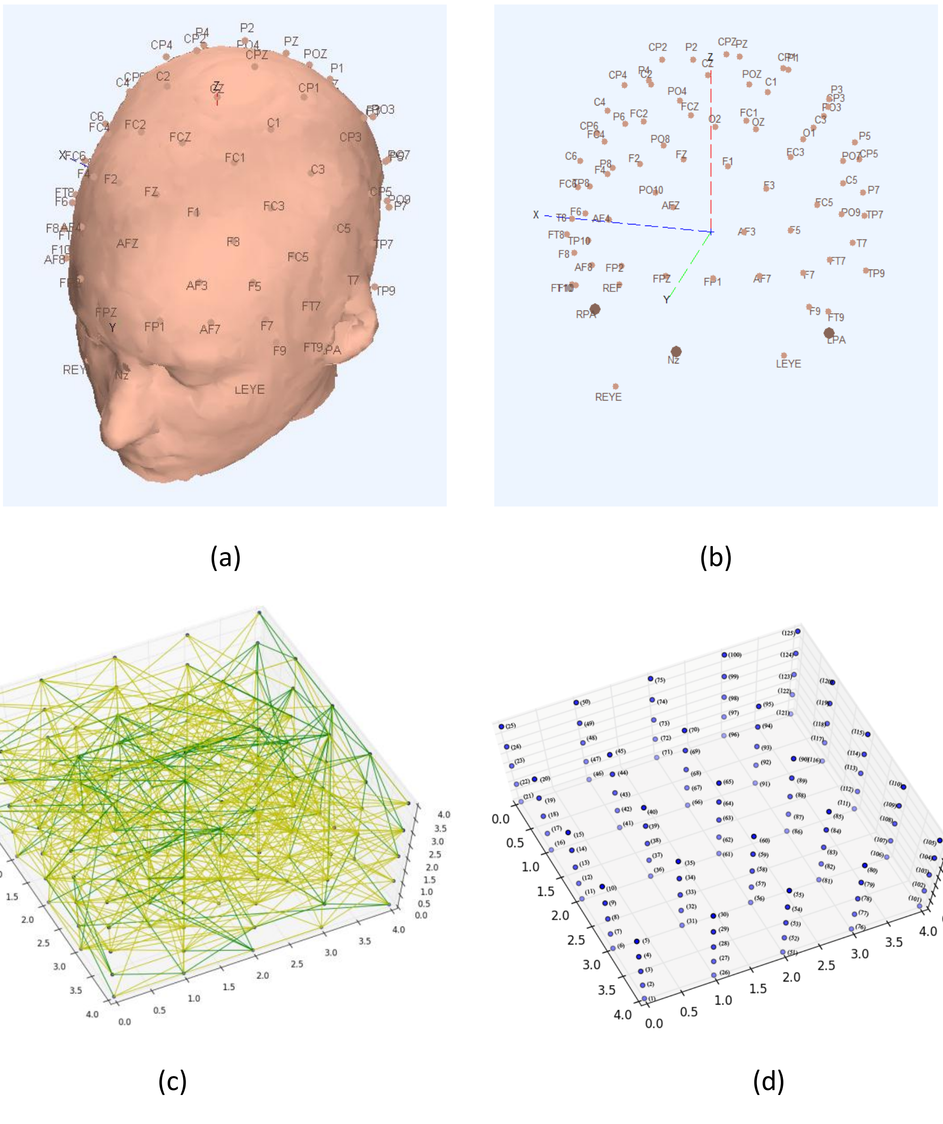


Figure 9.4: Demonstration of 3D EEG head map localization and 3D LSM visualization.

During processing of real EEG data, the way of in which input spikes into are fed into spatiotemporal filter may vary. This study uses two approaches of input signal projection; standard LSM in which data are normally randomly distributed with a certain probability of connection, and EEG localization mapping to LSM (as shown in figure 9.4). Before the SNN is trained on data, the spatiality distributed EEG channels are mapped onto allocated spatially distributed neurons in a 3D SNN reservoir, in which the idea is first published in (Kasabov, 2012b).

Figure 9.4 (a) and (b) show standard EEG channel names on a human head and the location of each EEG channel in 3D space, respectively. Figure 9.4 (c) is a 3D view of LSM comprising 125 neurons (5×5×5), the likely size of an EEG system containing 64 channels or less. The green and yellow lines show connections to inhibitory and excitatory synapses, respectively. Figure 9.4 (d) illustrates the alignment of neuron numbers on the LSM.

Figure 9.4 (a) and (b) were created using the EEGLAB library (A Delorme & S Makeig, 2004), while Figure 9.4 (c) and (d) were generated from pure Python programming code using synaptic connection information obtained from the Brian library for SNN simulation (Goodman & Brette, 2008).

Table 9.1 shows the LSM mapping of 64 standard EEG channels arranged in real-time according to Fig. 9.4 (a). The centres of EEG and LSM channels Cz and neuron 65 (the central neuron of the top layer of LSM) respectively, are the reference points for mapping. The centre channels (FPz, AFz, Fz, FCz, Cz, CPz, Pz, POz, Oz and Iz) are firstly mapped to LSM neurons (3, 4, 5, 35, 65, 95, 125, 124, 123 and 122), whose numbers denote the electrodes placed along the midline of the skull.

Table 9.1: Mapping table of standard 64 EEG channels matched to neuron number in epSNNr (size 5×5×5 neurons)

EEG channel	Neuron number in LSM	EEG channel	Neuron number in LSM	EEG channel	Neuron number in LSM	EEG channel	Neuron number in LSM
FPz	3	C4	25	CP2	70	P2	75
AFz	4	C6	24	CP4	50	P4	74
Fz	5	T8	23	CP6	49	P6	73
FCz	35	T10	22	TP8	48	P8	72
Cz	65	FTz	78	F7	52	AF7	29
CPz	95	FC5	79	F5	53	AF3	30
Pz	125	FC3	80	F3	54	AF4	10
POz	124	FC1	60	F1	55	AF8	9
Oz	123	FC2	40	F2	15	PO7	119
Iz	122	FC4	20	F4	14	PO3	120
T9	102	FC6	19	F6	13	PO4	100
T7	103	FT8	18	F8	12	PO8	99
C5	104	TP7	108	P7	112	FP1	28
C3	105	CP5	109	P5	113	FP2	8
C1	85	CP3	110	P3	114	O1	118
C2	45	CP1	90	P1	115	O2	98

From table 9.1 and Figure 9.4 (a), we observe that the letter identifies the brain functional area of each EEG channel while the number identifies the brain hemisphere (left or right). Letters F, T, C, P and O represent frontal, temporal, central, parietal, and occipital lobes, respectively. The C region is introduced for identification purposes only; no central lobe exists in the human brain. Lower-case z refers to an electrode placed on the midline. Even numbers (2, 4, 6, 8) refer to the right hemisphere of the brain, while odd numbers (1, 3, 5, 7, 9) refer to the left hemisphere (Niedermeyer & Da Silva, 2005).

The EEG electrodes are positioned relative to two anatomical landmarks; the point between the forehead and the nose (nasion) and the lowest point of the skull at the back of the head (inion). A prominent bump is commonly utilized as a reference point (Niedermeyer & Da Silva, 2005).

This architecture of input projection also imitates the human brain, in that some neurons receive information directly from a stimulus and neighbouring neurons may respond to the transmitted information (Arbib, 2002).

9.4 Conclusion

In this chapter, three novel network architectures were proposed for EEG spatiotemporal pattern recognition, ranging in design from simple to more complex. BSA is utilized for transformation of EEG data into spikes. The architectures are implemented and assessed on two real-world EEG datasets, as discussed in the following chapters.

Chapter 10

Spatio-temporal Pattern Recognition of Audio–Visual Stimuli Perception EEG

To investigate and evaluate the performance of a spike encoding method for EEG and SNN for spatiotemporal pattern recognition (whose mechanisms are described in Chapters 3-8), applications to real-world EEG datasets are required. The EEG dataset and the method by which it is converted to spikes are described in Section 10.1. Sections 10.2, 10.3, 10.4 and 10.5 evaluate the response of SSPAN, DepSNN, epSNNr (using DepSNN as readout unit) and mixed models respectively, to EEG inputs. The chapter concludes with Section 10.6.

This chapter's contribution consists of performance evaluation of the proposed methods and architectures on real-world EEG data and corresponds to the fourth research question.

10.1 Audio-Visual Stimuli Perception EEG Dataset

The dataset employed in this series of experiments is the audio-visual stimuli perception EEG Dataset, archived in the RIKEN Brain Science Institute in Japan. It comprises four stimulus states (in classification terminology, it has 4 classes). Class 1 is Auditory stimulus, Class 2 is Visual stimulus, Class 3 is Mixed auditory and visual stimuli and Class 4 lacks any stimulus. The EEG data were obtained from a 64 electrode EEG system and were filtered by a 0.05Hz to 500 Hz band pass filter and tested at 2 KHz. In this initial verification of the hypothesis, a small subset of the data points was used. More precisely, for each class, 11 epochs with similar sampling rate were selected from 50 epochs (1988 to 2153 samples/epoch/50ms, 4 classes making 44 epochs in total). It should also be noted that the sample rate of diverse auditory and visual stimuli is not steady within a class. 80% and 20% of data were used for training and testing, respectively.

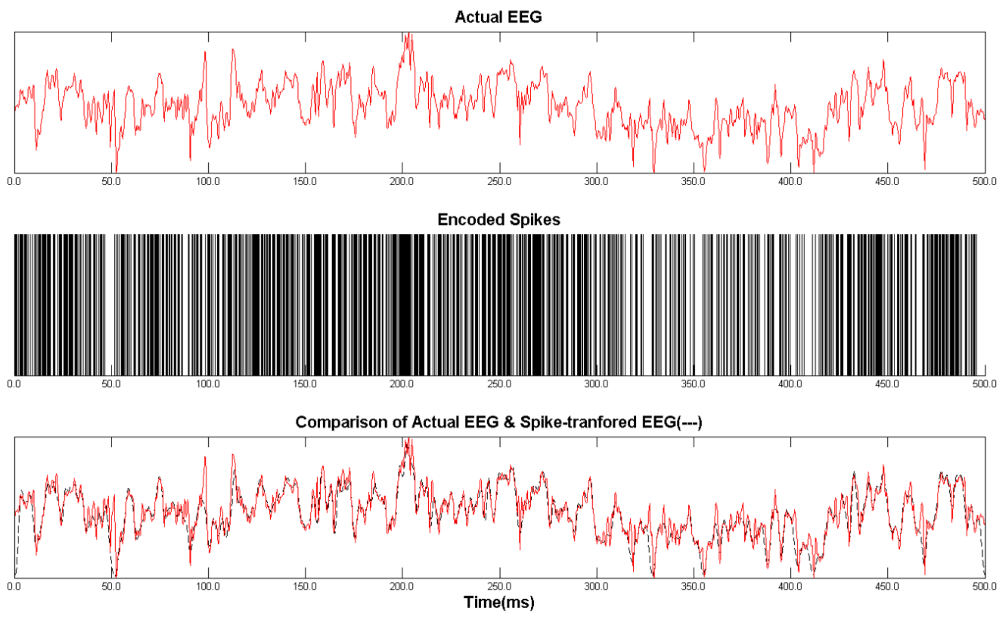


Figure 10.1: Single Channel EEG Signal (top panel), Spike Representation (central panel) and Actual EEG Signal superimposed on a Back-Transformed Signal (bottom panel)

The top panel of Figure 10.1 displays a single-channel EEG signal recorded over 500ms imitation time (50ms in real time). The central panel is the spike train of the EEG signal acquired by BSA encoding. The bottom panel illustrates the true EEG signal overlaid with the remodelled EEG signal, back-transformed from the BSA encrypted spikes. The resemblance between the two signals demonstrates that BSA conversion can accurately reconstruct EEG data.

To obtain a benchmark comparison, the above EEG dataset was applied to two traditional methods, non-adaptive Naïve Bayes and Multi-Layered Perceptron (MLP), with the following parameters: 139 sigmoid nodes in the hidden layer; learning rate 0.3; momentum 0.2; 500 training repetitions and authenticate threshold 20. The traditional methods performed poorly on this EEG dataset; 66.9% and 64.87% reconstruction accuracy was obtained for Naïve Bays and MLP respectively.

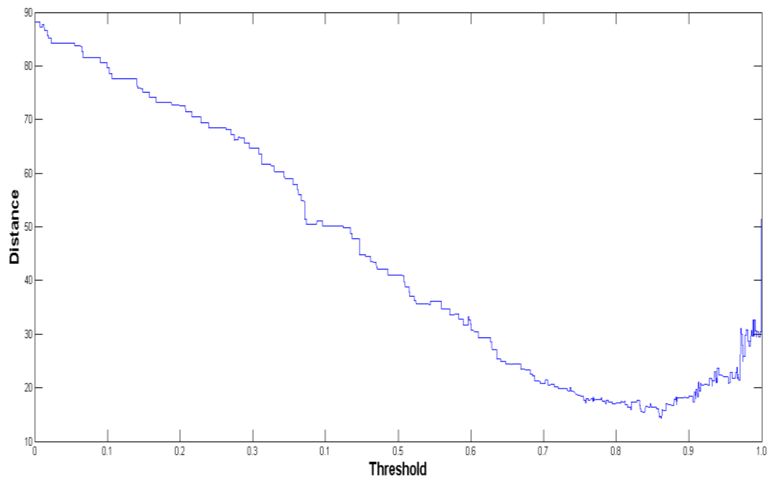


Figure 10.2: Optimal BSA Threshold on EEG

Figure 10.2 shows how distance varies with threshold when spikes are back-transformed into EEG data using the BSA algorithms. The *y* axis is the Euclidian distance between the original and back-transformed EEG signals. Spikes are back-transformed into signal via a convolution function with the same FIR filter as used for the original transformation. The *x* axis gives the threshold values of BSA algorithm (ranging from zero to one). The optimum threshold is that yielding the lowest Euclidian distance between the original and re-transformed signal. From the graph, this threshold appears around 0.85; a more precise estimate is 0.862 (for which the Euclidean distance is 14.288).

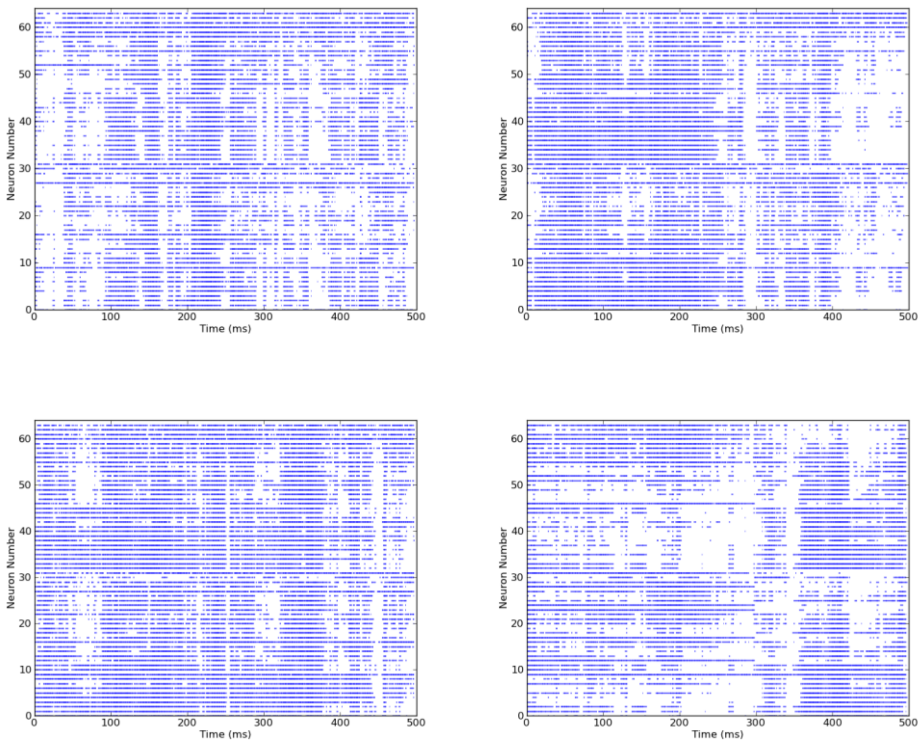


Figure 10.3: Samples of Input Stimuli Transformed to Spike Using BSA Algorithm. Graphs show how number of active neurons varies with time.

For the EEG dataset used in this study, the finite impulse response (FIR) filter size is set to 20, the wavelength to 0.05, and the BSA threshold to 0.862.

Several samples of input stimuli transformed to spikes using the BSA algorithm with the above parameter setup are shown in Figure 10.3. The left and the right of the first row display audio and visual stimuli respectively. On the bottom row, the left plot illustrates a mixed stimulus while the right represents no stimulus.

10.2 SSPAN on Audio-Visual Stimuli Perception EEG Dataset

This section explores the use of SSPAN (see Chapter 4 for details) on the audio-visual stimuli perception EEG dataset. Parameter setup is listed in table 10.1. The error in Figure 10.4 (1263.23) is the error at the first training iteration. The error at the third training iteration (27.16), during which no spikes were emitted from any SSPAN neuron, is depicted in Figure 10.5.

The above results show that when SSPAN neurons receive many spikes in a short time (high density of spikes), a high error results. To reduce the error rate, SSPAN updates most of the synaptic weights into inhibitory (negative weight values), which prohibits neurons from emitting spikes in subsequent training iterations (figure 10.5). At least 10-20 iterations of training are required for the neurons to become excitatory. Because the error rate in the absence of spiking is exceedingly low, SSPAN increases the synaptic weights rather slowly.

This phenomenon was observed recursively in this experiment. The performances are shown in table 10.2.

Table 10.1: SSPAN Parameter Setup

Methods/Parameters	**SSPAN**
Neural model	LIF,NR, ST, CT
Output neuron	4 neurons (of each model experiment)
Input spike train	64 spike trains
Membrane time constant	10 ms
Reset potential	0 mV
Firing threshold	10 mV
Input weight to SSPAN	0.01 mV
Reset μ	0 mV
Reset σ	3.0 mV
Threshold σ	2.0 mV
Noisy time constant	10 ms
Synaptic time constant	10 ms
Simulation time	600 ms
Simulation time step	0.1 ms
Training iteration	50
SSPAN learning rate	0.1

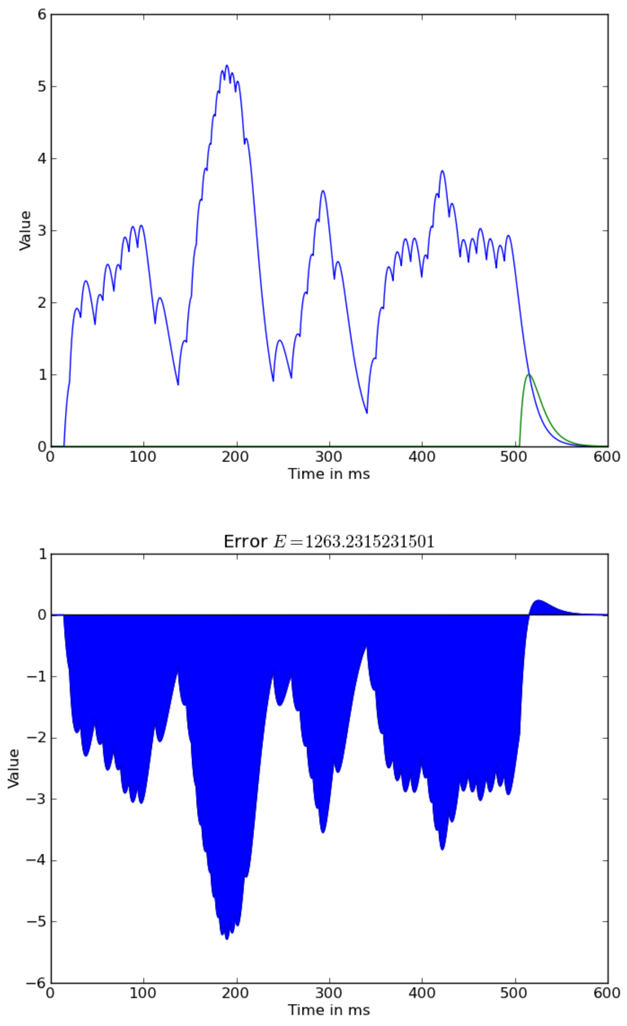
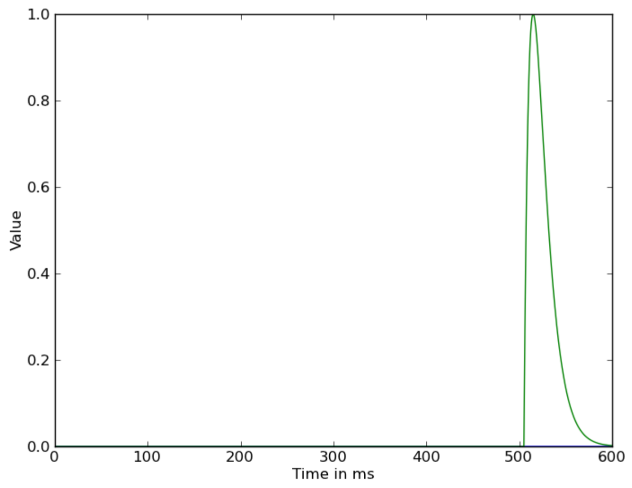


Figure 10.4: (top) Kernel Graphs of Actual (blue curve) and Desired (green curve) Spike of a SSPAN neuron after the first training iteration; (bottom) Plot showing the error between the desired and the obtained signal.

The upper plot of Figure 10.4 shows the desired spike (green curve) and the output spikes of a SSPAN neuron (blue curve). The lower plot displays the error graph which is used for updating the weight (computing Δw) in SSPAN.



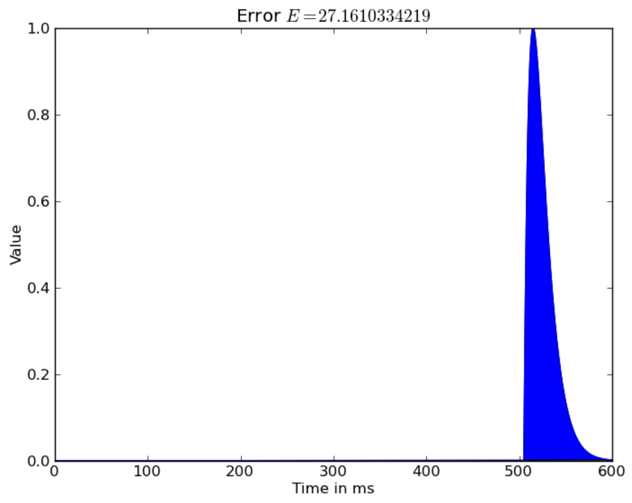


Figure 10.5: (top) Desired (green curve) spike of a SSPAN neuron with no firing activity; (bottom). The desired signal determines the magnitude of the error.

In Figure 10.5, the upper plot shows the kernel graph of the desired spike (green curve). Note that, because the SSPAN neuron emits no spikes, the blue curve is absent. The lower plot is the error graph which is used for updating the weight (computing Δw) in SSPAN.

Table 10.2: SSPAN performance, Percentage of classification accuracies are between actual and desired EEG spike after 50 iterations of training.

SSPAN(LIF)	SSPAN(NR)	SSPAN(ST)	SSPAN(CT)
12.5%	12.5%	25.0%	37.5%

From table 10.2, it is seen that SSPAN performs no better than traditional methods (Naïve Bay and MLP) on this EEG dataset, because SSPAN may not be suitable for high-density spike inputs arriving in a short time frame, such activity is characteristic of EEG data. When SSPAN neurons receive many spikes in a short period, a high error is incurred which most of its synaptic weights (Δw) into the inhibitory state to lower the error rate, corresponding the equation 4.13 (Chapter 4). This results in non-spikes propagation from output neurons. The issue may take at least 10-20 iterations of training are required before the neurons are again excited.

Increasing the number of training iterations (e.g. 500 iterations) may improve the performance, but is vary time consuming and is not suitable especially for real-time BCI application.

On a more positive note, Stochastic Neural models, especially Step-Wise Noisy Threshold (ST) and Continuous Noisy Threshold (CT) models, yield better results than the deterministic model. This trend mirrors the results of Chapter 6, when SSPAN was investigated on a synthetic dataset.

10.3 DepSNN for Audio-visual Stimuli Perception EEG Dataset

In this section, the performance of five types of DepSNN, namely DepSNNm, DepSNNs, NR-DepSNNs, ST-DepSNNs and NT DepSNN (see Chapter 5 for details), is assessed on the audio-visual stimuli perception EEG dataset.

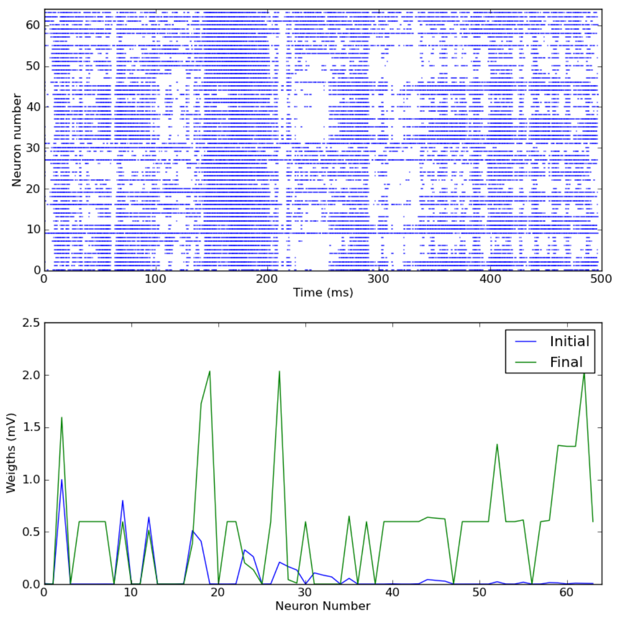


Figure 10.6: Input EEG (top) Spike Form Raster Plot and (bottom) evolution of synaptic weights

The upper panel of Figure 10.6 shows the input EEG as a spike form raster plot, while the lower panel plots the weight changes of the DepSNNs. Recall that DepSNNs ranks weights by order, and defines initial weights based on the SDSP dynamic synapse. EEG data are transformed into spikes using the BSA spike encoding method, with the parameter setup described in Sections 10.1 and 10.2. The parameters of the DepSNN trained on the EEG data are shown in Table 10.3.

In DepSNNm the spike threshold (V_{thr}) is adaptable to allow acquisition of PSP_{max}, which is important for calculating the optimal threshold of DepSNNm.

Table 10.3: Parameters for Neurons, Synapses and Learning Related Parameters

Parameters for neurons and synapses	
Excitatory synapse time constant	2 ms
Inhibitory synapse time constant	5 ms
Neural time constant (tau mem)	20 ms
Ejecting threshed (Vthr)	800 mV
Rearranged potential	0 mV
Inhibitory weight	0.20 V
Excitatory weight	0.40 V
Thermal voltage	25 mV
Refractory period	4 ms
Learning Related Parameters (Fusi's Synapse, Section 7.2)	
Up/Down weight jumps(Vthm)	5× (Vthr/8)
Calcium variable time constant (tau ca)	5× tau mem
Steady-state asymptote for calcium variable (wca)	50 mV
Stop-learning threshold 1(stop if Vca < threshold 1)	1.7 × wca
Stop-learning threshold 2 (stop LTD if Vca > threshold 2)	2.2 × wca
Stop-learning threshold 2 (stop LTP if Vca >threshold 3)	(8 × wca) –wca
Plastic synapse (NMDA) time constant	9 ms
Plastic synapse highest value (wp hi)	6 mV
Plastic synapse lowest value (wp lo)	0 mV
Bistability drift	0.25
Delta weight	0.12 × wp hi
Other parameters	
Input size (64 electrode EEG)	64 spiketrains
Simulation time	500 ms
Mod (for rank order)	0.8
Sim (DepSNN)	1

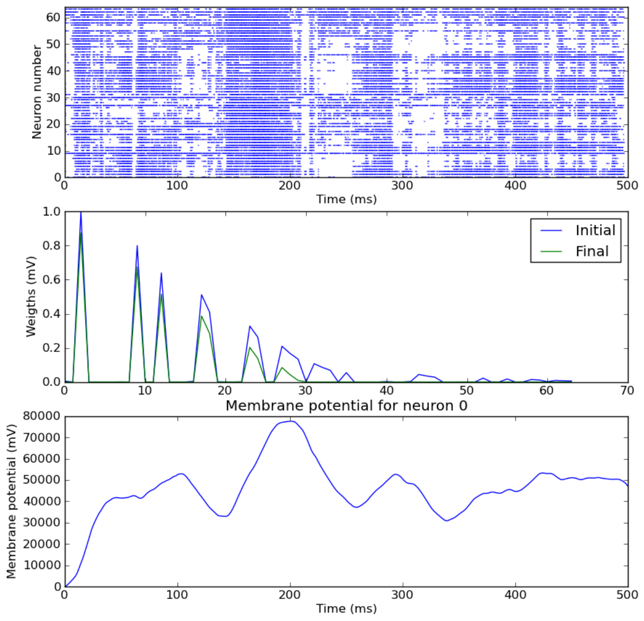


Figure 10.7: (top) Spike raster plot, (centre) synaptic weights and (bottom) post synaptic potential evolutions

Figures 10.6 and 10.7 illustrate the one-pass learning method on EEG data using DepSNNs and DepSNNm respectively. Figure 10.6 shows the synaptic weights of DepSNNs with the spike threshold set at 800 mV. In Figure 10.7, the highest membrane potential is seen to occur at around 200ms; this time point is taken as PSP_{max}. The top panels in both figures display the spike raster plot from which the weights changes are derived.

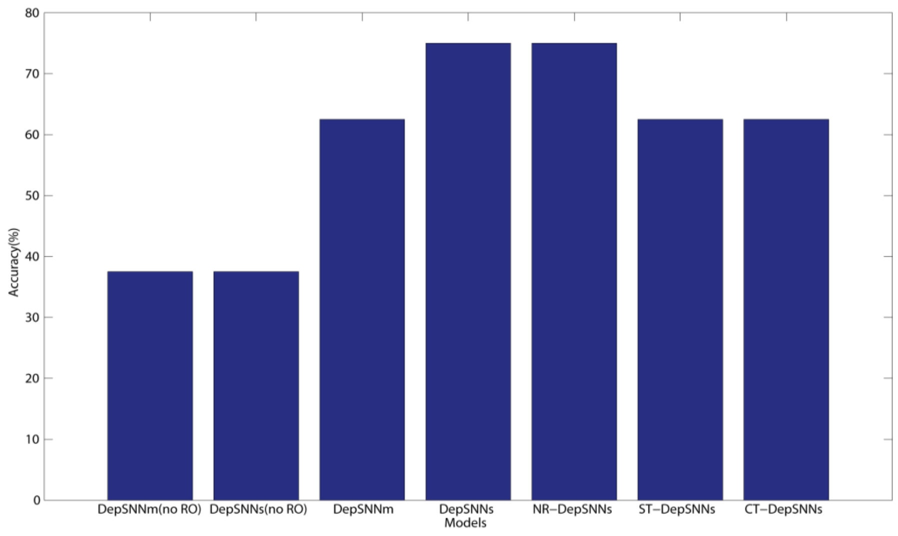


Figure 10.8: Classification Accuracy of Different DepSNN Models

From Figure10.8, it can be concluded that Rank Order Code (RO) is critical for DepSNN. With RO not implemented in DepSNNm and DepSNNs, poor classification accuracy (37.5%) is obtained for both methods. DepSNNs and NR-DepSNNs yield the highest classification accuracy (75%), but DepSNNs misclassified all no-stimulus samples. NR-DepSNNs recognised one no-stimulus sample but misclassified an auditory-stimulus sample. In addition, DepSNN exhibited superior performance in both computational efficiency and classification accuracy than the SSPAN analysed in Section 10.2. DepSNNs and NR-DepSNNs also displayed higher classification accuracy than the traditional methods (Naïve Bay and MLP) presented in Section 10.1.

10.4 epSNNr for Audio-Visual Stimuli Perception EEG Dataset

The experimental setup of this investigation is presented in Table 10.4. A reservoir possessing a small world interconnectivity pattern was constructed as described previously (Maass et al., 2002). To ensure a valid comparison, the recurring SNN is produced using Brian (Goodman & Brette, 2008) library with Python programming language whose grid alignment resembles that of CSIM (Pecevski et al., 2009) (A neural Circuit Simulation). LSM comprises 135 neurons in a three-dimensional grid of size 9 × 5 × 3, with default positioning. In this grid, two neurons A and B are linked with a probability given by

$$P(A,B) = C \times e^{\frac{-d(A,B)}{\lambda^2}} \tag{10.1}$$

where $d(A,B)$ is the Euclidean distance between the neurons and λ represents the density of connections (set to the default value for LSM i.e. λ=2). We expect that the closer the two neurons, the higher the probability of connection. C is a constant whose value depends on neural type (whether excitatory (ex) or inhibitory (inh). We assign the following values to C: C_{ex-ex} = 0.3, C_{ex-inh} = 0.2, C_{inh-ex} = 0.5, and $C_{inh-inh}$ = 0.1.

The following table shows the parameter setting that has been utilized in the experimental setup for the epSNNr.

Table 10.4: lists the parameter settings used in the epSNNr experiments

Parameters	**Value**
For Neuron	
Time Constance	10 ms
Reset Potential	0 mV
Firing Threshold	10 mV
Standard Deviation of NR model	3 mV
Standard Deviation of ST Model	2 mV
Standard Deviation of NT Model	1 mV
For LSM	
Simulation Time	500 ms
Number of Neuron	135
Excitatory and Inhibitory Ratio	4:1
Input Neurons Connection Probability	0.02
Input Neurons Connection Weight	1.62 mV
Time-bin for Liquid Responses	25 ms

Figure 10.9 illustrates the reservoir reaction of one epoch (50 ms) of auditory stimulus. The reservoir constitutes 135 LIF neurons, the *x* axis is the simulation time (maximum 500 ms) and the *y* axis shows the activation of the neurons (spikes are represented by dots). The liquid response of the network was mapped into 25 ms time bins (25 time bins/ epoch). The details of the mapping procedure are explained in Chapter 6 (Section 6.1.3). Figure 10.9 shows the most precise response obtained under this experimental framework.

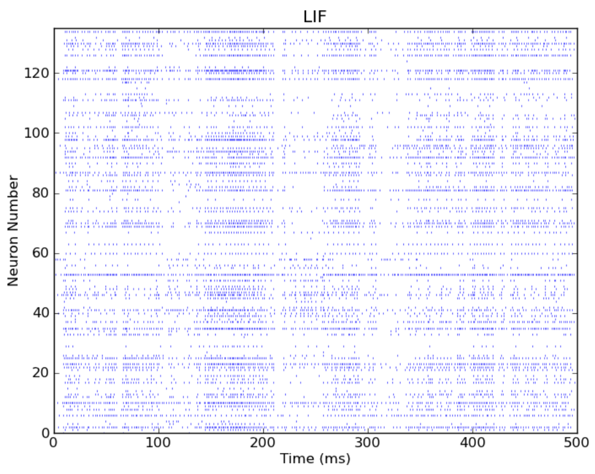


Figure 10.9: Liquid Response of the Network

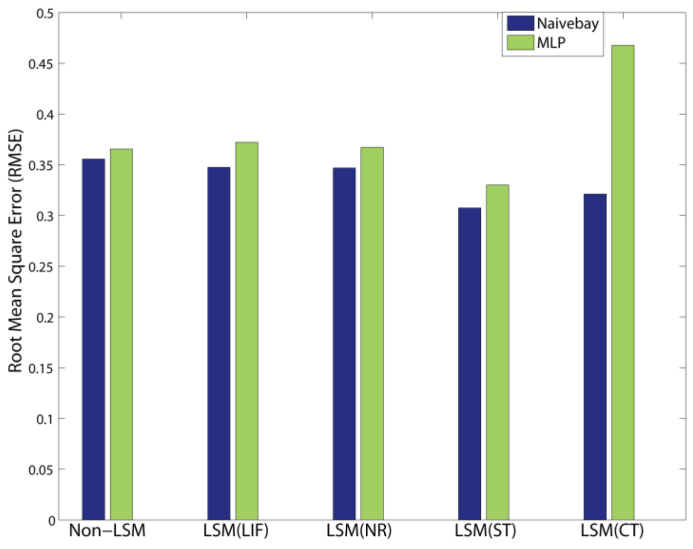


Figure 10.10: Root Mean Squared Error (RMSE) in epSNNr Models with non-adaptive Naïve Bayes and Multi-layered Perceptron Readout Functions

Two readout functions were analysed and tested in this experimentation; non-adaptive Naïve Bayes and Multi-Layered Perceptron (MLP). The MLP parameters were set to the following values: 139 sigmoid nodes in the hidden layer; learning rate = 0.3, momentum = 0.2, number of training repetitions = 500, authentic threshold = 20.

The performance of the epSNNr models using two types of classifier were compared with those of the two classifiers in the absence of both reservoir and spike representation (raw data).

The percentage classification accuracy (%) of the different methods is listed in Table 10.5. The classifiers alone do not accurately process raw EEG data. However, when the raw EEG is converted using BSA spike encryption and is passed into the classifier via epSNNr with different stochastic models, the performance of both classifiers improves markedly. Other classifiers were tested in the research but were deemed unsuitable for processing complicated spatiotemporal EEG data; hence they are not discussed here.

Table 10.5: Classification Accuracy

Readout Methods	**Without epSNNr**	**epSNNr Models**			
	Accuracy	**LIF**	**NR**	**ST**	**CT**
Naïve Bays	66.9%	75%	75%	75%	75%
MLP	64.87%	50%	50%	75%	50%

The precision achieved from epSNNr with Naïve Bayes as readout function is similar for all of the neural models (75%, misclassified no-stimulus class), but the root mean squared error (RMSE) values (as shown in Figure 10.10) vary significantly. In particular, the ST model with Naïve Bayes yields the lowest RMSE, demonstrating the maximum operation and steadiness of this model compared with the deterministic and other probabilistic models for this experiment.

The key results confirm that converting EEG signals into spike trains via the BSA spike encoding scheme considerably enhances the classification precision. Using a stochastic neural model in the epSNNr (for example, the ST model) may further enhance the precision, as seen in Table 10.5, epSNNr plus ST attained 75% classification accuracy and misclassified one sample in the no-stimulus class. By contrast, other models misclassified all samples in the no-stimulus class and some in the visual stimulus class (see also the root mean square errors in Figure 10.10).

10.5 epSNNr utilizing DepSNN as a Readout Unit on Audio-Visual Stimuli Perception EEG Dataset

In this section, mixed models, i.e. different epSNNr models utilizing DepSNN as a readout unit, are investigated. The experimental design comprised two main setups.

In the first setup, the parameters of DepSNN and epSNNr were set as listed in Tables 10.3 and 10.4 respectively, but the number of input neurons in DepSNN was made equal to that of epSNNr (i.e. 135 neurons).

The Time-bin for Liquid Responses parameter is not required for DepSNN because the spikes output of epSNNr is the input of DepSNN. This may increase the processing time because spike trains are input to DepSNN directly, by passing the binary transformation linear mapping requirements of conventional methods (Naive Bay and MLP) used as epSNNr readout functions. In addition, DepSNN has a rapid one pass learning mechanism which precludes the need for multiple iterations in the readout training process.

The second experimental setup investigates different ways of injecting input into the epSNNr spatiotemporal filter. The input injection mapping of 64 EEG channels (see Table 9.1) implies that the number of neurons in epSNNr is slightly smaller than in the first setup (125 neurons, 5×5× 5).

However, in the second setup, the input connection weights must be increased to 8.1 mV so that information can transmit to other neurons which are not directly connected to the input signal.

The classification accuracy for this dataset depends critically on the connection weights of the input neurons. An optimum input weight exists for which the classification accuracy is highest.

Table 10.6: EEG Classification Accuracy (%), for Two types of DepSNN with Various epSNNr Neural Models.

Readout Methods	**Without epSNNr**	**epSNNr Neural Models**			
	Accuracy	**LIF**	**NR**	**ST**	**CT**
DepSNNm	62.5%	37.5%	37.5%	37.5%	50%
DepSNNs	**75%**	62.5%	62.5%	62.5%	62.5%

From section 10.4, it is seen that epSNNr together with traditional methods (Naive Bays, MLP) as a readout function yields better classification accuracy than traditional methods alone. Remarkably, although the processing time of mixed models (epSNNr with DepSNN as readouts) is reduced, the classification accuracy is worse than when DepSNN alone is used (see Table 10.6). This behaviour results because DepSNN requires the complete spike patterns, which are transformations of the original signals, throughout the sampling period. Thus, temporal information is crucial for DepSNN. The mechanism of epSNNr not only transforms input information to higher dimension but combines spatial and temporal dimensions, changing the original patterns to several shorter temporal patterns of closely matched signals.

Table 10.7: Classification accuracy (%) of epSNNr with DepSNN as a readout and EEG localization input mapping.

SNN model in epSNNr	**Classifier**	
	DepSNNs	**DepSNNm**
LIF	25%	**75% (c=1)**
NR	37.50%	62.5%(c=1)
ST	25%	50% (c=9)
CT	37.50%	50% (c=3)

In contrast, when input is projected into the spatiotemporal filter unit as a localization mapping method in epSNNr, a similar accuracy to that obtained in Table 10.5 is obtained (75%; see Table 10.7). Recall that in Section 10.3,

DepSNNs alone and epSNNr with traditional methods (Section 10.4) were applied to the same dataset. However, the method presented here tended to misclassify the mixed-stimulus instead of the no-stimulus class.
This emphasises that DepSNN on EEG requires the whole spike patterns across the entirety of the sampling period. The suggested EEG input projection approach must be performed on other EEG datasets before it can be declared valid.

SSPAN was not adopted as the readout function in this section because when SSPAN neurons receive many spikes in a short period, a high error is incurred (as explained in Section 10.2). SSPAN updates most of its synaptic weights into the inhibitory state (negative synaptic weights) to lower the error rate, resulting in many idle neurons during subsequent training iterations (see Figure 10.5). At least 10-20 iterations of training are required before the neurons are again excited. Due to the low error rate incurred by no spiking activity, SSPAN increases its synaptic weights slowly. In this experiment the same phenomenon occurred, resulting in the performances shown in Table 10.2.

The above-mentioned problem will be exacerbated when the input signals inject to epSNNr, since the output of epSNNr is now the input of SSPAN, which is larger than the original dataset but has similar spike density.

10.6 Conclusion

The BSA spike encrypting scheme was found to be appropriate for encrypting EEG data stream into spike trains. In this chapter, we also addressed the question of whether probabilistic neural models are suited to LSM reservoir computing. From the results of the experiments, the following conclusions are drawn:

(1) The classification accuracy of epSNNr is enhanced when the deterministic LIF model is replaced with a probabilistic neural model.

(2) The classification operation of the epSNNr depends on the type of underlying probabilistic neural model. This suggests that an epSNNr can be optimized by considering the neural models used and by choosing parameters that better suit the noise and the dynamics of particular EEG data in a changing environment.

(3) Given the same EEG data, the results of DepSNNs are similar to those of epSNNr. However, DepSNN has a simpler structure and faster performance. It requires no more than single-pass learning and relatively few neurons (in this experiment a mere 33 output neurons were assigned; 32 neurons in the training process and one in the recall process).

(4) The DepSNN model is easier to realize as a specialised SNN hardware than the epSNNr.

(5) The performance of DepSNNs differs from that of DepSNNm. The critical parameters for for DepSNNs and DepSNNm are the threshold and the parameter C, respectively.

(6) The processing time of mixed models (epSNNr with DepSNN as readouts) is faster than epSNNr with traditional method as readout, but the

performance declines relative to DepSNN alone. However, epSNNr may shorten the extended temporal dimension of spatiotemporal EEG samples (which may exceed 1 minute). Combining epSNNr with DepSNN may also prove useful in other applications involving pattern recognition and prediction, such as video or the evolution of biological cells.

(7) The injection of input into the spatiotemporal filter in epSNNr utilizing EEG localization mapping (DepSNN as readouts) enables higher classification accuracy than the traditional means of projection input into epSNNr, but depends critically on the input connection weights.

While DepSNN performed more accurately than epSNNr and the mixed models (adopting epSNNr with DepSNN as readout functions), the latter may prove superior when dealing with extended STP (1 second or longer). Although the DepSNN models employ more sophisticated learning algorithms, their simple scalar synaptic weights are limited in their ability to capture long and complex temporal patterns.

The new models, especially DepSNN, are superior to conventional methods in terms of learning time and accuracy, because they utilize information that is ordered relative to the first input spike, as well as information related to the time of arrival of consecutive spikes. These two information formats constitute a spatiotemporal input pattern.

Injection into the spatiotemporal filter of epSNNr utilizing EEG localization mapping (DepSNN as readouts) improves the classification accuracy relative to the conventional approach. Further investigation on additional EEG datasets may produce different and interesting results, and forms the basis of the next chapter.

Since EEG signals can vary when repeating a given task, depending on the overall state of the brain and its local environment, the proposed methods could be used to construct adaptive and robust Brain-Computer Interfaces (BCI). They might also find use in other applications requiring fast encoding methods and one-pass learning. Also in the next chapter, these approaches are investigated in DepSNN and EEG localization mapping to epSNNr.

Chapter 11

Spatio-temporal Pattern Recognition on P300 EEG application

In Chapter 10, it was shown that using DepSNNs and NR-DeSNNs on the same EEG data produces results similar to those of epSNNr. However, DepSNN has a simple structure and rapid processing capacity. It requires fewer neurons (33 output neurons) and learning is achieved in a single iteration. Moreover, the DepSNN model is easier to realize as a specialized SNN hardware than the epSNNr. We determined that the threshold is the critical parameter for DepSNNs, whereas the parameter C is important for DepSNNm

When epSNNr utilizing DepSNN was used as a readout unit and the EEG localization data were mapped to the spatiotemporal filter in epSNNr, interesting results were obtained which are worthy of further investigation.

In this chapter, the above-mentioned configuration is applied to BCI; specifically, to the P300. The P300 is a real time EEG-based verbal communication BCI which assists people with speaking disabilities.

The proposed methods and architecture presented in this case study reasserts the algorithms capability and applicability in BCI domain by providing better performance and analysis of the EEG derived brain activities. This is carried out by integrating various SNN approaches such as 3D Mapping, spike encoding schemes and stochastic reservoirs. This SNN based systems provided a unique design and novel direction in BCI domain. This contribution is one of the potentially BCI design approaches and answers to the fifth and last research question.

An overview of BCI and the P300 is provided in Section 11.1, and the dataset is described. Section 11.2 discusses the application of DepSNN to P300 EEG data, using BSA as a spike encoding scheme. In Section 11.3, epSNNr utilizing DepSNN as a readout unit is presented, and the mapping of EEG localization data to the spatiotemporal filter in epSNNr is proposed. The last section (11.4) discusses and closes the chapter.

11.1 What is a Brain Computer Interface (BCI)?

BCI provides a direct link between the computer and the human brain. It is essential to the development of Human Computer Interfaces (HCI). Current BCI advances include Brain Machine Interfaces (BMI), in which scientists detect ejecting neurons with the aid of hundreds of pins inserted into the cerebral cortex. The computer translator executes algorithms that decrypt the neural language into computer language. The many applications of BCI include unmanned vehicles, robotics and non-verbal communication. In EEG, the electrodes are positioned around the scalp and signals are intercepted from the brain. In Magnetoencephalography (MEG), the room is adorned with helium tanks and super conducting magnets. MEG is very precise but requires formidable technological advances.

11.1.1 P300 EEG application

P300 (also known as P3) is a neural-evoked potential component of EEG. It presents as a positive voltage deflection (2-5µV) 300-600 ms following the start of a stimulus (Ekanayake, 2010).

Major applications of P300 include the P300 BCI Spelling tool. Mak et al. (2012) identified EEG characteristics associated with P300 reliant Brain Computer Interface (P300 BCI) performances in people suffering from Amyotrophic Lateral Sclerosis (ALS) (Mak et al., 2012).

11.1.2 The P300 Dataset Description and Former Investigations

A six-choice P300 paradigm was tested using six different images flashed in random order, as shown in Figure 11.1. Electrode configurations comprised 32 electrodes (datasets and algorithms are available from the website of the Ecole Polytechnique F´ed´erale de Lausanne, Signal Processing Institute, EPFL BCI group http://bci.epfl.ch/p300).

Dataset acquisition involves the extraction of single trials (duration 1000 ms) from the data. Single trials began at stimulus onset, i.e. at the initiation of the strengthening of an image, and were completed 1000 ms later. To account for the

ISI of 400 ms, the final 600 ms of each trial were coincided with the first 600 ms of the succeeding trial.

Data were strained through a sixth-order forward-backward Butterworth band pass filter. Frequencies between1.0 Hz and 12.0 Hz were selected.

Subjects were requested to soundlessly count the instances of a presented image. The six images were exhibited on the screen and a warning tone was released. Four seconds following the warning tone, a random sequence of flashes was initiated and the EEG was recorded. The order of flickers was block randomized.

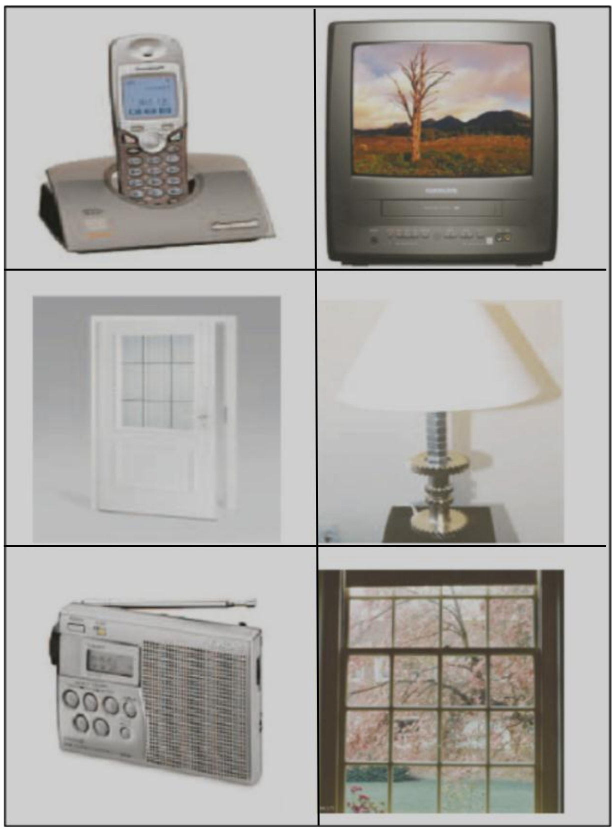


Figure 11.1: Images Used for Evoking the P300 (Ulrich Hoffmann, Vesin, Ebrahimi, & Diserens, 2008)

Hence, after every six flashes each image was flashed once and after twelve flashes each image was flashed twice. This sequence cycled continuously.

Images were flashed consecutively by altering the overall brightness of the images. This study employed the dataset of Hoffmann et al. (2005), which differs from that of their 2008 study (Ulrich Hoffmann et al., 2008). This detail is inconsequential because the method of data acquisition was identical in the 2005 and 2008 studies.

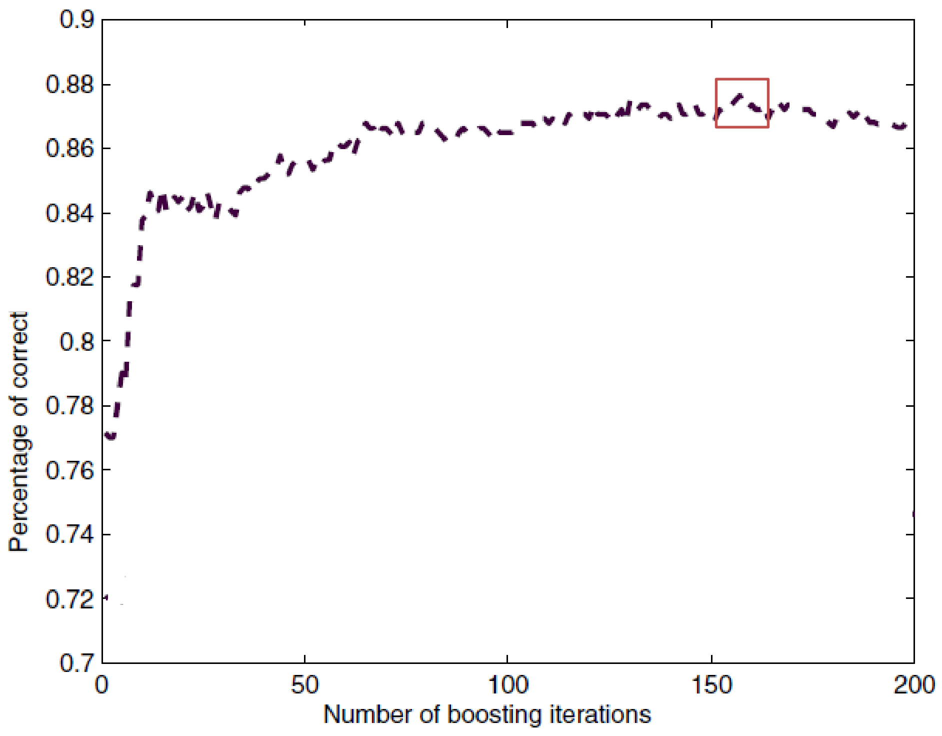


Fig. 11.2: Percentage of Correctly (×100) Categorized Tests for Varying *Number of Boosting iterations* (U. Hoffmann, Garcia, Vesin, Diserens, & Ebrahimi, 2005)

Hoffmann and colleagues (2005) analysed the dataset used in this chapter by gradient boosting, a machine learning method that creates one strong classifier from many weak classifiers. Hoffman et al. (2005) proposed a gradient-boosting inspired algorithm that finds event-associated potentials in single EEG samples. The algorithm identifies the P300 in the human EEG, from which a Brain Computer Interface (BCI), namely a spelling device, is constructed. Significant advantages of this method are its high classification precision and easy presentation of concept.

The application of gradient boosting to this dataset is illustrated in Figure 11.2. Here, the maximum number of repetitions of the boosting algorithm ($M_{max)}$ was set to 200 and the optimum numbers were computed in a 10 fold cross-validation.

11.2 DepSNN on P300 EEG Application

This section assesses the feasibility of DepSNN as a P300 BCI application. The dataset for this experiment is similar to that mentioned in Section 11.1. Data was recorded from 32 channels; Fp1, Fp2, AF3, AF4, F7, F3, Fz, F4, F8, FC1, FC5, FC6, FC2, T7, C3, Cz, C4, T8, CP1, CP5, CP6, CP2, P7, P3, Pz, P4, P8, PO3, PO4, O1, Oz, O2 at 2048Hz (U. Hoffmann et al., 2005).

Epochs initiating from the commencement of a flash and remaining for 1s were filtered from the data. Time consuming flows in the data were eradicated by least squares fitting of a linear function applied to each channel and deduction from the data. The data were then re-referenced to the average of channels O1, Oz and O2, low- pass filtered between 0 and 9 Hz with a 7th order Butterworth filter and down- sampled to 128Hz. The dataset is downloadable from the EPFL BCI group website (http://bci.epfl.ch/p300).

The methodology is tested on a reduced dataset of 942 spatiotemporal samples from a single subject, comprising 476 target samples and 466 non-target samples. The dataset was divided into a training set (60%) and a testing set (40%).

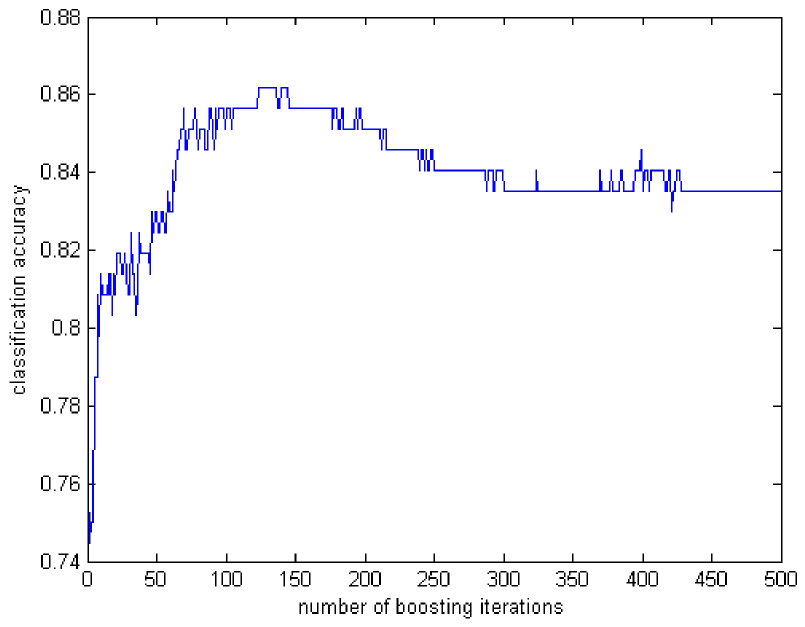


Figure 11.3: Percentage of Accurately Categorized Tests for Varying Values of Number of Boosting Iterations

Figure 11.3 graphs the percentage of accurately categorized tests for different numbers of boosting iterations (M), where M was set to 500 in this experiment. The highest accuracy (86%) was attained at around 140-150 iterations.

In this experiment, data were converted to spikes via BSA with parameters set as follows:

1. Finite Impulse Response filter size: 20
2. Finite Impulse Response filter wave length: 0.05
3. Optimal Threshold: 0.88 (at Euclidean distance 16.1061).

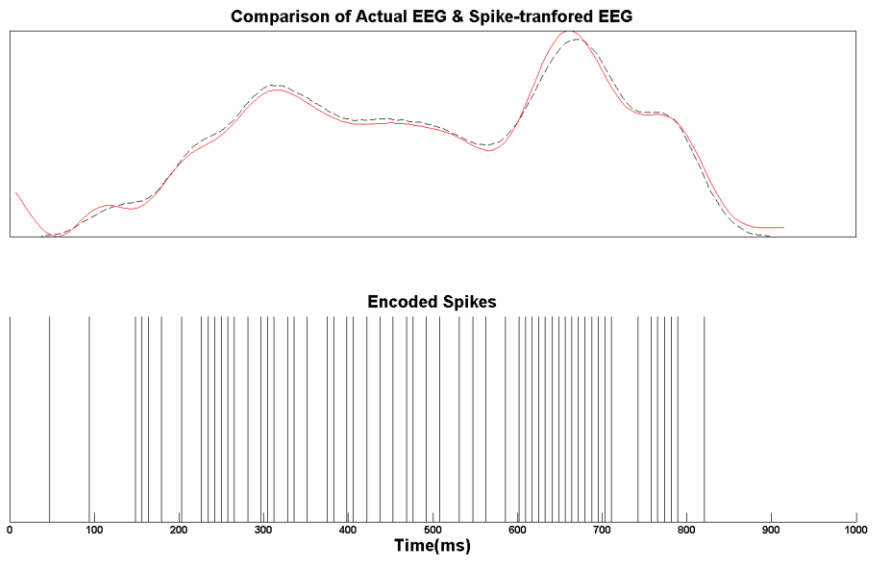


Figure 11.4: (top) EEG Signal for a Duration of 1000 ms (Simulation Time and Real Time) Solid and dashed curves represent the true and reconstructed signal, respectively. (Bottom) BSA-transformed signal

The top panel of Figure 11.4 represents a single-channel EEG signal sampled over 1000 ms (in both simulation time and real-time). The true EEG signal is overlaid with the EEG signal back-transformed from the BSA encoded spikes (dashed lines). The bottom panel is the spike transform of the true signal acquired from BSA.

Table 11.1: Setup of DepSNN Parameters

For Neurons and Synapses	
Excitatory synapse time constant	2 ms
Inhibitory synapse time constant	5 ms
Neural time constant (tau mem)	20 ms
Firing threshed (Vthr)	800 mV
Reset potential	0 mV
Inhibitory weight	0.20 V
Excitatory weight	0.40 V
Thermal voltage	25 mV
Refractory period	4 ms
For learning related parameters	
Up/Down weight jumps(Vthm)	5× (Vthr/8)
Calcium variable time constant (tau ca)	5× tau mem
Steady-state asymptote for calcium variable (wca)	50 mV
Stop-learning threshold 1(stop if Vca< threshold 1)	1.7 × wca
Stop-learning threshold 2 (stop LTD if Vca> threshold 2)	2.2 × wca
Stop-learning threshold 2 (stop LTP if Vca> threshold 3)	(8 × wca) –wca
Plastic synapse (NMDA) time constant	9 ms
Plastic synapse highest value (wp hi)	6 mV
Plastic synapse lowest value (wp lo)	0 m
Bistability drift	0.25
Delta weight	0.12 × wp hi
Other parameters	
Input size (32 electrode EEG)	32 spiketrains
Simulation time	1000 ms
Mod (for rank order)	0.8
Sim (DepSNN)	1

The clear resemblance between the original and back-transformed signals highlights the applicability of the BSA conversion to the P300 EEG data. As also evident from Figure 11.4, just 58 spikes are encrypted from 128 vectors describing the original wave, implying that BSA-encoding reduces the number of data vectors to be processed, while retaining sufficient information to reproduce raw EEG signals.

The parameter settings of the DepSNN trained on the EEG data are listed in Table 11.1. However, as mentioned in Chapter 10 (Section 10.3), the spike threshold (V_{thr}) is flexible so that DepSNNm can acquire the PSP_{max} which optimises the threshold. The parameter settings of the Stochastic Spiking Neural models used in conjunction with DepSNN are listed in Table 11.2.

Table 11.2: Parameter settings for the Stochastic Spiking Neural Models

Parameters	**Value**
Time Constant	10 ms
Reset Potential	0 mV
Firing Threshold	10 mV
Standard Deviation of NR model	3 mV
Standard Deviation of ST Model	2 mV
Standard Deviation of CT Model	1 mV

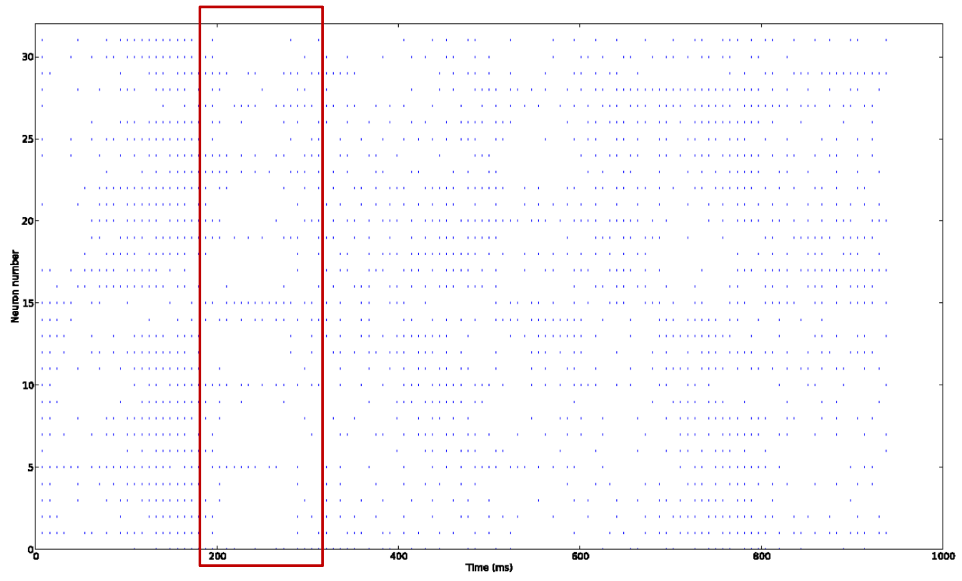


Figure 11.5: Encoded Spike Trains of the BSA-encoded P300 Dataset for One Sample of the Non-Target Class

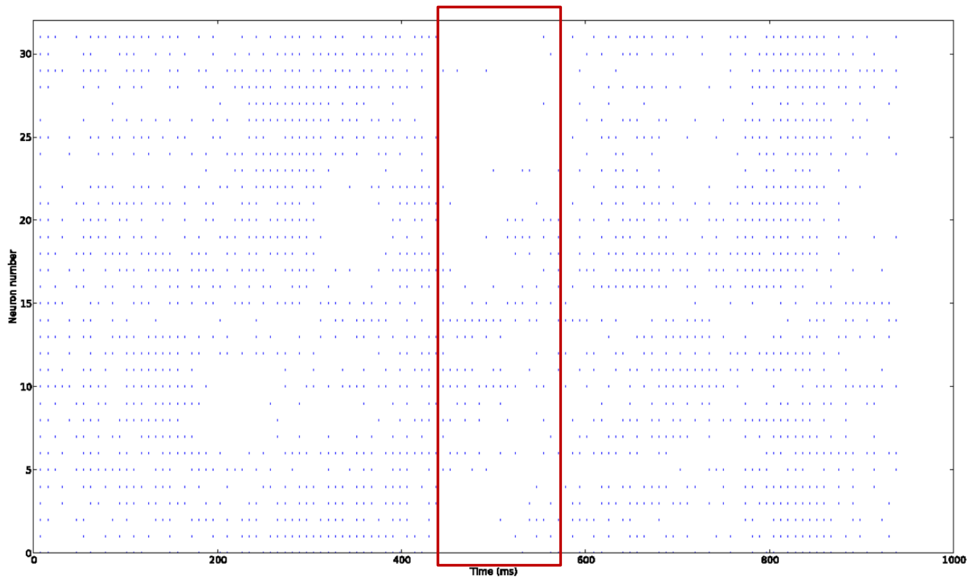


Figure 11.6: Illustration of the Encoded Spike Trains of the BSA-encoded P300 Dataset for One Sample of the Target Class

Figures 11.5 and 11.6 display the BSA-encoded spike trains of the P300 dataset for a single sample of non-target and target data class, respectively. The top panel of Figure 11.7 shows the weight changes in the DepSNNm (where initial weights are ranked by order), together with the SDSP dynamic synapse. The bottom panel plots the evolution of the post synaptic potential of DepSNNm responding to a target class sample.

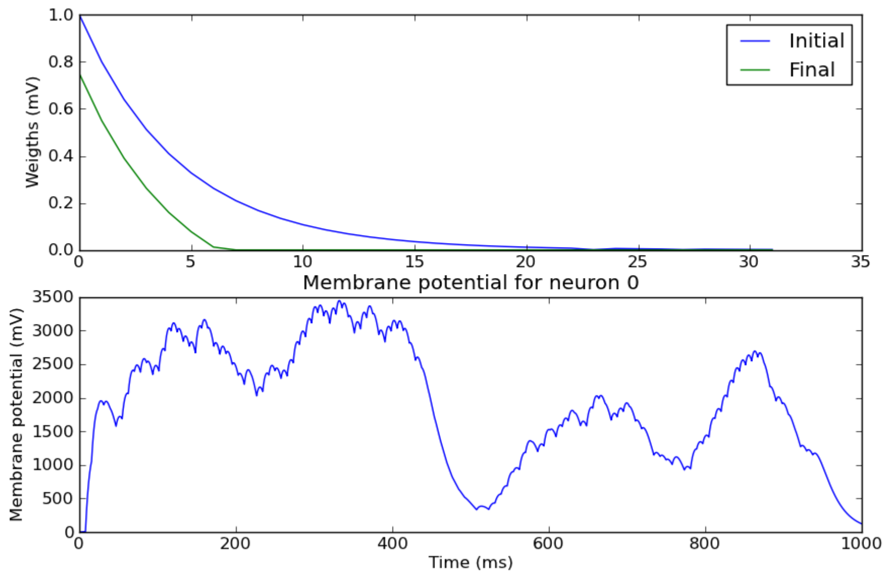


Figure 11.7: (Top panel) Weights Changes for a single neuron of the DepSNNm and (bottom panel) Evolution of Post Synaptic Potential of DepSNNm

The top figure shows the weights change for the DepSNNm that utilizes rank order defining initial weights along with the SDSP dynamic synapse. The bottom figure illustrated the evolution of post synaptic potential of DepSNNm responded to a sample of Target class.

Figure 11.8 plots of weights change for the various models of DepSNNs responding to a sample of the target class, where the left of the first row is represented the response of standard DepSNNs, the right of the first row is NR-DepSNNs, the left of the button row is ST-DepSNNs, and the right column of the bottom row is CT-DepSNNs.

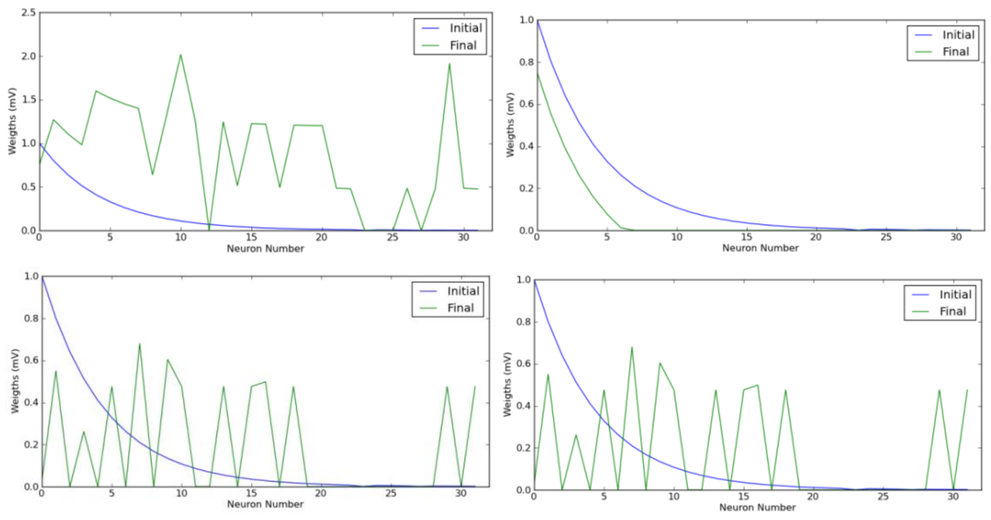


Figure 11.8: Weights Change for the DepSNNs that Utilizes Rank Order

In this section, two main experiments were conducted. The first was undertaken on a small dataset (32 samples) of which 60% were reserved for training and 40% for testing. The aim was to assess DepSNN feasibility, and the results are summarised in Table 11.3. The second experiment used the full dataset (942 samples from one subject, comprising 476 target class and 466 non-target class). 60 and 40% of the data were assigned to training and testing respectively, as shown in Table 11.4.

As evidenced from Tables 11.3 and 11.4, the different DepSNN models exhibit interesting behaviours. DepSNNm performed superbly in the small dataset (with 100% classification accuracy). In the full dataset investigation, DepSNNs performed better than other DepSNN models (accuracy 84.41%). Similar to the EEG data from audio-visual stimuli (Chapter 10), replacing the deterministic LIF model with stochastic models in DepSNNs made no improvement to the performance of the system.

We noted in Chapter 10 that Rank Order Code (RO) is critical for DepSNN. In the absence of RO, DepSNNm and DepSNNs demonstrated poor classification accuracy, whereas the DepSNNs and NR-DepSNNs exhibit highest classification accuracy.

In this chapter, when DepSNN is tested on a portion of the P300 based BCI dataset (see Table 11.3), DepSNNm exhibits higher classification accuracy than DepSNNs.

On the full P300 based BCI dataset, however, DepSNNs out-performed DepSNNm, in contrast to the results of the audio-visual stimuli perception EEG dataset. This demonstrates that rank or code can perform properly with DepSNNs when the dataset can itself perform the ranking. We note that the timing of the first spike of each spike train differs between the two datasets. The dynamic timing of spikes generated from the audio-visual data enables DepSNNm to perform better on this dataset. On the full BCI dataset, DepSNNs performs better, probably because information is acquired from more samples in the training phase.

Therefore, it appears that the classification accuracy of a given DepSNN model depends on the nature and size of the EEG dataset. However, it remains unclear whether stochastic spiking neural models in DepSNNs offer any sizeable advantages over traditional models.

Table 11.3: Classification Accuracy of Different Models applied to a subset of the P300 based BCI Data

Methods	**Accuracy**
DepSNNs	58.33%
NR-DepSNNs	50%
ST-DepSNNs	50%
CT-DepSNNs	50%
DepSNNm	**100% (C=0.5)**

Table 11.4: Classification Accuracy of Different Models applied to the full P300 Based BCI Dataset

Methods	**Accuracy**
Gradient Boosting	**86%**
DepSNNs	**84.41%**
NR-DepSNNs	48.39%
ST-DepSNNs	50%
CT-DepSNNs	50%
DepSNNm	74.19% (C=0.4)

Table 11.4 compares the classification accuracy between Gradient Boosting and DepSNN applied to the P300 based BCI dataset. DepSNNs and Gradient Boosting exhibit similar performance (84.41% and 86% respectively). However DepSNNs requires single-pass learning where Gradient Boosting needs 150 boosting iterations to achieve maximum accuracy. Moreover, according to Hoffmann et al. (2005), the minimum acceptable classification accuracy of P300 based BCI data is 80%. At 84.14% accuracy, DepSNNs is a promising candidate for further BCI application and research. The rapid processing time of DepSNNs is an additional advantage.

11.3 epSNNr with EEG localization mapping on P300 EEG Application

In Chapter 10, we established that EEG localization mapping injected into the spatiotemporal filter of epSNNr with DepSNN as readout provides similar classification accuracy to DepSNN. In this section, in order to confirm the benefit of the injection into epSNNr approach, EEG localization mapping into epSNNr with DepSNN is applied to the P300 dataset.

The parameter setup of DepSNN remains unchanged from Table 11.1. The epSNNr parameters are set up as shown in Table 11.5.

The epSNNr comprises 125 spiking neurons (5×5×5). 1000 ms simulation time is assigned to each spatiotemporal sample. 32 EEG channels (input signals) were projected into the epSNNr (see Table11.6 for details).

In this experiment, the input neurons connection weight parameter (Table 11.5) must be increased to 22.68 mV to enable transmission of information to neurons not directly connected to the input signal. The value of this parameter was selected to yield optimal accuracy for this dataset. Higher or lower input weights would not improve the classification accuracy of the system.

Table 11.5: Parameter settings in the experimental setup for the epSNNr

Parameters	**Value**
For Neuron	
Time Constance	10 ms
Reset Potential	0 mV
Firing Threshold	10 mV
Standard Deviation of NR model	3 mV
Standard Deviation of ST Model	2 mV
Standard Deviation of NT Model	1 mV
For LSM	
Simulation Time	1000 ms
Number of Neurons	125 (5×5×5)
Excitatory and Inhibitory Ratio	4:1
Input Neurons Connection	EEG Localisation mapping to LSM
Input Neurons Connection Weight	22.68 mV

Table 11.6: Mapping table of 32 EEG channels (from the BCI based P300 dataset); name is mapped to neuron number in epSNNr (5×5×5)

EEG channel	Neuron number in LSM	EEG channel	Neuron number in LSM
Fz	5	F7	52
Cz	65	F3	54
Pz	125	F4	14
Oz	123	F8	12
T7	103	P7	112
C3	105	P3	114
C4	25	P4	74
T8	23	P8	72
FC5	79	AF3	30
FC1	60	AF4	10
FC2	40	PO3	120
FC6	19	PO4	100
CP5	109	FP1	28
CP1	90	FP2	8
CP2	70	O1	118
CP6	49	O2	98

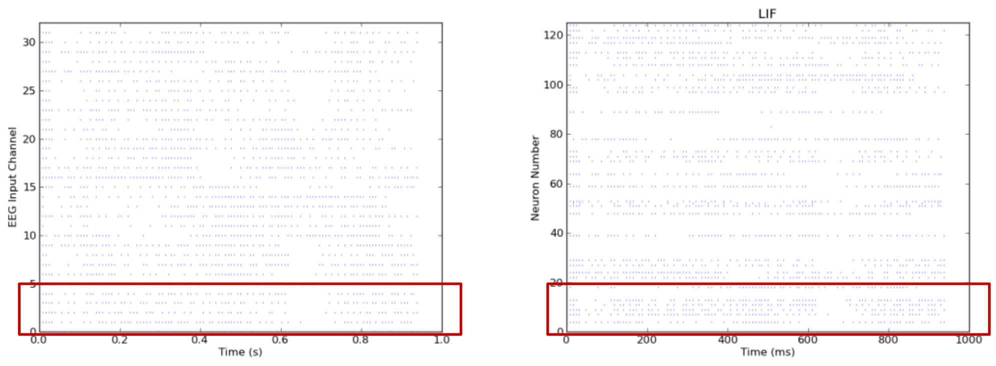


Figure 11.9: Raster plot of EEG input signel before (left) and after (right) injection into epSNNr

The raster plot of the EEG input before and after feeding into LSM is displayed in Figure 11.9. To the left is the raster plot of EEG signal spikes (32channels); on the right is the response of epSNNr to the spike information under the EEG localisation mapping of Chapter 9 (Figure 9.4). The spiking trends are quite similar in specific neuron groups; for example, neurons 1 to 5 in the left plot and neurons 0 to 20 in the right plot. However, the right plot contains extra information referenced indirectly from the input spikes (tranmitted from other neurons as a small-world connection in epSNNr).

Table 11.7 summarises the results of this experiment on EEG-based P300. The classification efficiency is compared between various models utilized in epSNNr. The input signals were projected into epSNNr by EEG localisation mapping utilizing DepSNN as a readout function.

Table 11.7: Experimental results

epSNNr neural model	**Classifier**	
	DepSNNs	**DepSNNm**
LIF	72.58%	**98.92%** (C=0.3)
NR	22.04%	53.22%(C=0.6)
ST	73.66%	61.83% (C=0.5)
CT	33.87%	**98.92%** (C=0.3)

In Section 11.2, the classification accuracy of DepSNN was established as 84.41%, similar to that of Gradient Boosting (86%).

Interestingly, EEG localisation mapping into the spatiotemporal filter in epSNNr with DepSNN as a readout unit yields a vastly superior result (98.92%) when LIF and CT models are used in DepSNNr. The highest accuracy is achieved using DepSNNm as readout with the C (fraction) set to 0.3.

DepSNNm as a classifier obtains better results than DepSNNs, because the repressed spiking activity of neurons in the spatiotemporal filter leads to ambiguity in synaptic weight patterns between two classes.

From the results of this section, we conclude that the mode of input injection into epSNNr is crucial in deciding the efficacy of the spatiotemporal filter unit. This projection method also imitates human brain activity, in that some neurons receive stimulus information directly while their neighbours receive it by transmission. Hence, DepSNN as a readout acquires more information from epSNNr than is available from direct input. Biological brains also attempt to fill in, match and complete partial information from stimuli (Arbib, 2002). As a practical example, glasses designed for 3D movies trick the watchers into believing they are enveloped by a scene which actually appears on a flat screen.

11.4 Conclusion

DepSNN is superior to conventional methods in terms of learning time and accuracy. DepSNN also out- performed the epSNNr models, though the latter may be better equipped to handle longer STP signals, of duration 1 second or more. This is attributable to the simple scalar synaptic weights in DepSNN models, which limit the extent to which DepSNN can capture long and complex temporal patterns, despite the sophisticated learning algorithms.

Equally important is the method of input injection into epSNNr, as revealed in Section 11.3.

DepSNN and epSNNr with EEG localization mapping are strong candidates for building adaptive and robust BCIs and other systems requiring fast encoding methods and single-pass learning processes.

To summarise, a feasibility study was conducted on DepSNN and epSNNr with EEG localization mapping using P300 based Brain Computer Interface data. From the results presented in this chapter, the following conclusions can be drawn:

(1) The DepSNN is superior to conventional methods (including Gradient Boosting) in terms of learning speed and accuracy.

(2) The classification performance of each DepSNN model depends on the nature of the dataset. Especially, if the first spike of each input spike train arrives at a different time, the rank ordering mechanism of DepSNNs is compromised because DepSNNs needs the correct order of initial synaptic weight information. (This fact was confirmed by studies on the audio-visual stimuli perception EEG dataset discussed in Chapter 10).

(3) In BCI application, subjects are always trained prior to using a system. This causes most of the brain response to a given EEG electrode to arrive simultaneously. Ranking is now dictated by EEG channel rather than by time of first spike. This phenomenon is responsible for the poor performance of DepSNNs on the P300 dataset relative to DepSNNm. The

same trend results when DepSNN is applied to object-movement spatiotemporal pattern recognition (Dhoble et al., 2012).

(4) The performance of Stochastic Spiking Neural Models in DepSNNs is worse than that of standard DepSNNs on the BCI dataset. The BCI data require lucid information of each subject. The noise introduced by stochastic models may lead to confusion in the classifiers. However, when applied to audio-visual stimuli perception EEG data (see Chapter 9), DepSNNs and NR-DepSNNs rated first equal among the tested models. Hence, we conclude that highly complex EEG pattern recognition tasks will likely benefit from stochastic models, but further exploration is required to confirm this hypothesis.

(5) The injection of EEG localization mapping as input to the spatiotemporal filter using epSNNr with DepSNN as a classifier enhances the classification accuracy relative to DepSNN alone. This is due to the extra information provided by the spatiotemporal filter in epSNNr, which causes the system to mimic biological brain activity. The system’s performance depends crucially on the input connection weights to epSNNr.

Chapter 12

Conclusion and Future Work

This chapter summarizes the work undertaken to achieve the research objectives as specified in Chapter 1 and to answer the research questions mentioned there in. Suggestions for future are also presented in this chapter.

This thesis describes novel algorithms for spatiotemporal data modelling and classification using spiking neural networks (SNN). More specifically, SSN was used to analyse and classify EEG data. The algorithms belong to the class of evolving SNN (eSNN), the main principles of which were introduced by Kasabov in 2007. The following points outline the highlights of the research and contributions of the thesis:

(1) Different models of spiking neurons, learning algorithms and encoding schemes are reviewed in Chapter 2. The thesis then offers a critical analysis of Electroencephalography (EEG) data analysis and classification methods used to date.

(2) Next, methods of encoding EEG data into spikes are investigated. In this thesis, the Ben Spike Encoder Algorithm (BSA) is applied for the first time to EEG data. This is also a part of research question 4 mentioned in Chapter 1.

(3) Three new stochastic models are proposed as an extension of PSNM; namely, Stochastic Noisy Reset (NR), Stochastic step-wise noisy threshold model (ST) and Continuous stochastic threshold (CT). These models extend the LIF model by introducing noise parameters into the threshold and reset potentials, thereby mimicking activity of biological neurons while retaining low implementation cost. This contribution corresponds to the first research question.

(4) A stochastic precise-time spike pattern association neuron (SSPAN) is introduced. This is a modification of SPAN, a neuron for precise-time spike pattern association, partially answers the second research question.

(5) A new SNN model, termed Dynamic Evolving Probabilistic Spiking Neural Network (DepSNN), is introduced as an extension of the eSNN model. This contribution answers to the other half of the second research question.

(6) A novel SNN architecture is proposed for processing spatiotemporal data, termed evolving probabilistic SNN reservoir (epSNNr). The performance of epSNNr is enhanced when a probabilistic neural model is used, as demonstrated on EEG data in a changing environment. The developed epSNNr answers the third research question.

(7) New SNN systems for spatiotemporal pattern recognition including SSPAN, DepSNN and epSNNr have been developed investigated on the real world EEG datasets including BCI. The presented solutions answer to research questions 4, 5 and 6.

(8) As a future development, an epSNNr for EEG data is preliminarily explored. The reservoir structure is an approximate 3D map of the human brain and the data from the EEG channels are entered into corresponding neurons from the reservoir. The reservoir connections and parameter tuning are also crucial for exploration. These are promising approaches that are worthy of further investigation.

12.1 Conclusions

This thesis introduces two novel SSN models; Stochastic Precise-time Spike Pattern Association Neuron (SSPAN) and Dynamic Evolving Probabilistic Spiking Neural Network (DepSNN). SPAN convolves the spike train output with a kernel function to yield a continuously valued signal on which subtraction and multiplication operations can be carried out. SSPAN can readily accommodate existing methods for developing supervised-learning rules in spiking neurons. Method performances were evaluated and compared on a synthetic dataset created for that purpose. SSPAN was found to out-perform conventional SPAN, especially when incorporating the CT model. Moreover, classification accuracy was enhanced when the deterministic LIF model was replaced by stochastic models in SSPAN. More synapses can memorize more patterns leading to increased performance; SSPAN has 420 synapses while SPAN possesses a mere 140 synapses. Stochastic neural models are expected to process noisy stochastic data such as EEG more effectively than deterministic models.

Five types of Dynamic evolving probabilistic spiking neural network (DepSNN) were proposed; namely DepSNNm, DepSNNs, NR-DepSNNs, ST-DepSNNs, and NT-DepSNNs. DepSNN requires a high density of active spikes in the spike train before a SDSP synapse is sufficiently active to modify its synaptic weights. Hence, DepSNN may be suitable for BSA- encoded EEG data, which are automatically imbued with high spiking activity. epSNNr projects a spatiotemporal signal into a linearized high-dimensional network state that can be learned by a linear readout function or another SNN network. The epSNNr improves when the deterministic LIF model is replaced by a probabilistic model. The classification performance of the epSNNr depends on the type of probabilistic neural model adopted. This suggests that an epSNNr can be optimised in terms of neural models used and parameters that would better suit the noise and the dynamics of particular EEG data in a changing environment.

DepSNNs and epSNNr applied to the same EEG data yield similar accuracies. However, DepSNN has a simpler structure and performs faster. It requires fewer neurons (33 output neurons were assigned in the experiments of this thesis) and single-pass learning only is required. The DepSNN model is easier to realise as a specialised SNN hardware than the epSNNr. The performance of DepSNNs and DepSNNm differs in that the threshold is critical for DepSNNs, whereas for DepSNNm, the parameter C is more important.

The processing time of mixed models (epSNNr with DepSNN as readouts) is reduced but the classification accuracy is worse than that yielded by DepSNN alone. However, epSNNr may reduce long-duration spatiotemporal EEG samples (extending to1 minute or longer) to a more manageable time scale. Combined epSNNr and DepSNN may also find a use other pattern-decoding tasks such as video or the evolution of biological cells.

The projection of input signals to epSNNr as EEG localization mapping with DepSNN as readout results in superior classification accuracy and performance time. Here, the one-pass learning mechanism of DepSNN complements the spatiotemporal epSNNr filter, which linearly transforms the input signal without altering the trend of input signals over time. Similarly, the biological brain attempts to fill in, match and complete information that is missing from stimuli, but which is still decipherable. Choice of connection weights input to epSNNr is crucial to the success of this method.

Stochastic Spiking Neural Models in DepSNNs perform poorly on BCI data relative to standard DepSNNs because BCI data must accurately depict each subject. The noise introduced by stochastic models may tend to confuse the classifiers. Conversely, DepSNNs and NR-DepSNNs are equally competent at processing audio-visual stimuli perception EEG data. Hence, in highly complex EEG pattern recognition tasks, it is tentatively concluded that stochastic models will out-perform deterministic ones. Proof of this hypothesis would come with further testing.

Hence, projecting input signals to epSNNr as EEG localization mapping with DepSNN as readout presents as a promising means of building adaptive and robust Brain-Computer Interfaces (BCI), as well as other systems requiring fast BSA-encoding and one-pass learning in DepSNN.

12.2 Contributions

The major contributions of this study are as follows:

(1) The Ben Spiker Algorithm (BSA) for transforming EEG data into spike trains has been developed and modified (as explained in Chapter 8). This is the first application of BSA beyond that of speech recognition.

(2) Three novel Stochastic SNN models based on the proposed pSNM - Noisy Reset, Stepwise Noisy Threshold and Continuous Noisy Threshold - are introduced and described in Chapter 3.

(3) An extension of the SPAN architecture and neural model, namely Stochastic-SPAN (SSPAN: Chapter 4) has been proposed, which enables more efficient processing of spatiotemporal data. The deterministic neural model (LIF) in SPAN is replaced by the three stochastic SNM. One output neuron is assigned per class rather than one neuron for all classes. Consequently, more synapses are available in SSPAN, enabling more patterns to be recognized and reducing the time required for learning.

(4) DepSNN (see Chapter 5), an improvement of eSNN that can process spatiotemporal data, was proposed. DepSNN retains rapid single-pass learning ability but the population encoding method has been replaced by BSA. This modification not only reduces the number of spikes but improves the quality of original signal imitation. The Thorpe SNM has also been replaced by LIF model and Stochastic Models. The most important feature of DepSNN is the inclusion of SDSP, which allows the model to capture the temporal dimension and thereby to process spatiotemporal data.

(5) More complex SNN structures are proposed for spatiotemporal data processing, namely epSNNr (Chapter 6). epSNNr adopts LSM architecture for accumulating spatial and temporal data and transforming this non-linear information to a linear higher-dimension form. epSNNr also employs stochastic models whose performances are superior to that of deterministic SNM (traditional LSM).

(6) A method is introduced for injection of EEG to a spatiotemporal filter in epSNNr by mapping EEG localization on the human head to epSNNr. The transformed EEG data are then processed as described in Contribution (5). epSNNr imitates biological brain functions, and demonstrates superior performance to DepSNN or standard input injection of epSNNr.

(7) Two new SNN-based solutions to real world problems are developed; DepSNN and epSNNr. The latter has been combined with EEG localization mapping as described in Contribution (6). Applicability to

audio visual stimuli perception EEG data has been assessed in Chapter 10, and applicability to EEG-P300 based BCI has been explored in Chapter 11.

(8) In accordance with the study objectives, all experiments in this study have been shared with the scientific community in six blind peer-review international academic papers and one poster.

12.3 Recommendations and Future Prospects

epSNNr with EEG localization mapping plus DepSNN as readout is particularly effective on the EEG P300 dataset. The performance of this architecture may be further improved by optimizing parameters. However, both methods require many parameters to define the neural models, connections, and network topology. Optimization algorithms such as quantum-inspired genetic algorithm (Schliebs et al., 2009) or particle swarming (Hamed et al., 2012) may not therefore be suitable for the proposed architecture, due to the prohibitive time requirements incurred.

Given the large number of parameters involved, epSNNr and DepSNN might be optimized using Genetic Regulatory Networks,which was successfully used for recurrent Neural network (Cheng et al., 2011), or computational neuro-genetic models (CNGM) (Kasabov, 2012a) which designed especially for SNN. CNGM combines two dynamic models; (1) a low-level gene regulatory network (GRN) model and (2) a high-level SNN that models the dynamic interaction between genes and spiking patterns of activity under certain conditions (Kasabov, 2012a).

Prior to constructing the GRN, all important parameters of DepSNN and epSNNr must be defined.

The significant parameters which may require optimisation are:

(1) DepSNN parameters: Plastic synapse (NMDA) time constant, Plastic synapse highest value (w_p_hi), Plastic synapse lowest value (w_p_lo), Bistability drift, Delta weight, Mod, Sim and C.

(2) epSNNr parameters: Number of neurons and topology, Input Neurons Connection Weight.

(3) Parameters for both stochastic models: Standard Deviation of NR Model, Standard Deviation of ST Model, Standard Deviation of NT Model (SD-NT), Neural time constant, Firing threshed, Reset potential and Refractory Period.

Two types of GRN could be incorporated into GRN-DepSNN and GRN-epSNNr :

(1) Static-GRN, in which the GRN connection is fixed but the connection weights can be varied.

(2) Dynamic-GRN, in which the GRN connection weights are fixed but the connection is adaptive.

By optimising the connection weights in the spatiotemporal filter of epSNNr utilizing Spike-Time Dependent Plasticity (STDP) (Song et al., 2000), the classification accuracy of the system might be enhanced and increased understanding of human brain knowledge discovery may result.

Since the spatiotemporal filter in epSNNr employs the most recent reservoir approach (LSM) it shares some similarity with a study on a Poisson process conducted by Norton and Ventura (Norton & Ventura, 2009). According to this study, incorporating STDP into LSM resolves the two main LSM problems:

(1) The Pathological Synchrony problem, in which most of the neurons become caught in infinite excitatory firing, resulting in pattern loss.

(2) The Over-Stratification problem, in which neurons fail to propagate a series of spikes, resulting in temporal de-coherence.

Appendix A

The core programming source code of the three Stochastic Spiking Neural Models.

```
from brian import *


##########################################################################
# Base class for the spiking models
##########################################################################

class SNNModel:
    '''
    Base class for all spiking neural models.
    '''

    # slot for the equation string of the model
    eqs = ''

    # slot for a dictionary containing model parameters
    modelParams = None

    # slot for the name of the model
    modeName = 'Basic SNN model'

    def __init__(self):
        pass

    def getNeuronGroup(self, nbNeurons):
        '''
        Generates a group of "nbNeurons" neurons of the model represented by this class.
        '''
        print self.eqs, self.modelParams
        return NeuronGroup(nbNeurons, model=self.eqs, **self.modelParams)
    def has_adaptive_threshold(self):
        return False


#############################################################################
# Deterministic LIF neuron
#############################################################################

class LIFModel(SNNModel):
    '''
    Implements the traditional deterministic Leaky Integrate-and-Fire model.
    '''

    def __init__(self, tau, **kwargs):
        self.eqs = '''dV/dt = -V/(%s) : volt''' % tau
        self.modelParams = kwargs
        self.modelName = 'LIF'

    def getNeuronGroup(self, nbNeurons):
        group = SNNModel.getNeuronGroup(self, nbNeurons)
        return group


#############################################################################
#  Step-wise noisy threshold Model (ST)
#############################################################################

class AdaptiveThresholdStepnoise(object):
    '''
    Resets the firing threshold to a random reset value
    '''

    mu = None
    sigma = None
    reset_potential = None

    def __init__(self, reset_potential, mu, sigma):
        self.reset_potential = reset_potential
        self.mu = mu
        self.sigma = sigma

    def __call__(self,P):
        '''
        Sets the firing threshold to a random reset value.
        '''
        spikes=P.LS.lastspikes()
        P.V[spikes] = self.reset_potential
        for s in spikes:
            P.Vt[s] = self.mu + (self.sigma * randn(1)[0])*mV



class StepNoisyThresholdModel(SNNModel):
    '''
    Implements a Leaky Integrate-and-Fire model with a (slow) noisy step-wise threshold.
    '''

    reset_threshold = None
    sigma = None
    reset_potential = None

    def __init__(self, tau, reset_potential, reset_threshold, sigma=1., **kwargs):
        self.sigma = sigma
        self.reset_threshold = reset_threshold
        self.reset_potential = reset_potential
        self.eqs = '''
        dV/dt = -V/(%s) : volt
        dVt/dt = 0.0*mV/ms : volt ''' % tau
        self.modelParams = kwargs
        self.modelParams['threshold'] = lambda V,Vt:V>=Vt
        self.modelParams['reset'] = AdaptiveThresholdStepnoise(self.reset_potential,
self.reset_threshold, self.sigma)
        self.modelName = 'ST'

    def getNeuronGroup(self, nbNeurons):
        group = SNNModel.getNeuronGroup(self, nbNeurons)
        for n in xrange(nbNeurons):
            group.Vt[n] = self.reset_threshold + (self.sigma * randn(1)[0])*mV
        return group

    def has_adaptive_threshold(self):
        return True


##########################################################################
# Noisy Reset model (NR)
##########################################################################

class AdaptiveThresholdNoisyReset(object):
    '''
    Resets the potential to a random reset potential
    '''
    mu = None
    sigma = None

    def __init__(self, mu, sigma):
        self.mu = mu
        self.sigma = sigma

    def __call__(self,P):
        '''
        Sets the membrane potential to a random reset value.
        '''
        spikes=P.LS.lastspikes()
        for s in spikes:
            P.V[s] = self.mu + (self.sigma * randn(1)[0])*mV


class NoisyResetModel(SNNModel):
    '''
    Implements a Leaky Integrate-and-Fire model with a noisy reset after spike emittance
    '''

    sigma = None
    mu = None

    def __init__(self, tau, mu=0*mV, sigma=1., **kwargs):
        self.mu = mu
        self.sigma = sigma

        self.eqs = '''dV/dt = -V/(%s) : volt''' % tau
        self.modelParams = kwargs
        self.modelParams['reset'] = AdaptiveThresholdNoisyReset(mu, sigma)
        self.modelName = 'NR'

    def getNeuronGroup(self, nbNeurons):
        group = SNNModel.getNeuronGroup(self, nbNeurons)
        for n in xrange(nbNeurons):
            group.V[n] = self.mu + (self.sigma * randn(1)[0])*mV
        return group



##########################################################################
# Continuous Stochastic Treshold Model (CT)
##########################################################################

class AdaptiveThresholdNoisy(object):
    '''
    Resets the potential to resting potential
    '''

    reset_threshold = None
    reset_potential = None

    def __init__(self,reset_threshold, reset_potential):
        self.reset_threshold = reset_threshold
        self.reset_potential = reset_potential

    def __call__(self,P):
        '''
        Sets the membrane potential and the firing threshold to their reset values.
        '''
        spikes=P.LS.lastspikes()
        P.V[spikes] = self.reset_potential
        P.Vt[spikes] = self.reset_threshold

class NoisyThresholdModel(SNNModel):
    '''
    Implements a Leaky Integrate-and-Fire model with a (fast) noisy threshold
    '''

    reset_threshold = 10*mV
    reset_potential = 0*mV

    def __init__(self, tau, noisy_tau, reset_threshold, reset_potential=0*mV,  **kwargs):
        self.eqs = '''
        dV/dt = -V/(%s) : volt
        dVt/dt = (xi*mV)/(%s)**.5 : volt ''' % (tau, noisy_tau)
        self.reset_threshold = reset_threshold
        self.reset_potential = reset_potential

        self.modelParams = kwargs
        self.modelParams['threshold'] = lambda V,Vt:V>=Vt
        self.modelParams['reset'] = AdaptiveThresholdNoisy(self.reset_threshold,
self.reset_potential)
        self.modelName = 'CT'

    def getNeuronGroup(self, nbNeurons):
        group = SNNModel.getNeuronGroup(self, nbNeurons)
        group.Vt = self.reset_threshold
        return group

    def has_adaptive_threshold(self):
```

```
    return True
```

Appendix B

The core programming source code of kernel transformation, kernel operation and learning rule in SSPAN.

```
from brian import *
import numpy
import pylab
from scipy.integrate import simps

class Updater:

    def __init__(self,sim_time,dt,tau_s):

        self.dt = dt
        self.t = numpy.arange(0,sim_time, self.dt)
        self.tau_s=tau_s


    def alpha(self, tf):   # tf is firing time

        tau_s = self.tau_s

        So = numpy.e * 1. / tau_s

        v = numpy.zeros(len(self.t))

        for f in tf:

            s = (self.t-f)
            v +=  So * s * numpy.exp(-s / tau_s) * numpy.array(s>=0, dtype=float)
        return v


    def compute_deltas(self, stimulus, output, target):


        # transform the input spikes spike trains
        self.x = numpy.zeros((len(stimulus), len(self.t)))
```

```
for i,s in enumerate(stimulus):
   self.x[i] = self.alpha(s)


# transform the output spike train
self.y_out = self.alpha(output)

# transform the target spike train
self.y_d = self.alpha(target)

# compute the error
self.diff = self.y_d - self.y_out
self.error = simps(numpy.abs(self.diff), dx=self.dt)

# compute delta w
self.delta_w = simps(self.x * self.diff, dx=self.dt, axis = 1)
```

Appendix C

The core programming source code of SDSP and RO in DepSNN.

```
from pylab import *
from brian import *
from brian.utils.progressreporting import ProgressReporter
from time import time
from core.learner import *
from core.utils import *
import os
import operator

############################################
# Parameters and constants for the training set
############################################
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 ###
Vthm = 0.75*Vthr #5*Vthr/8. # Up/Down weight jumps
tau_ca = 5*tau_mem # Calcium variable time constant
wca = 50 * mV # Steady-state asymptote 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 # Bi-stability drift
wp_thr= (wp_hi - wp_lo)/2.+wp_lo # Drift direction threshold
wp_delta = 0.12*wp_hi # Delta Weight

###########Equations#########
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 inhibition
’’’)
eqs_reset = ’’’v=Vrst’’’

#############Parameters of the deSNN ##############
input_size = 19
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 = []
#Connection weights between the input layer and the output layer
SIM_TIME = 1000 *ms
seed(1)
mod=0.8

######## Read all files from defined directory path #######

path = ‘Test/’  ## directory path of the input patterns (stimuli)
listing = os.listdir(path)
# Get data Files
for infile in listing:
print ‘‘Reading from file: ‘‘ + infile,"\n##################’’

##———Spiketrain stimulus from file——-##
spiketimes=inputfile_to_spikes(path+infile)


## ———Rank Order Code (RO) ——-##
s=sorted(spiketimes, key=operator.itemgetter(1))
rankW=zeros((input_size,1))
for i in xrange(len(s)):
rankW[s[i][0]][0]=float(mod**i)
wp0=rankW
```

```
print ''Rank Order Weights:\n'',wp0

##—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)#, refractory=refr) # Output layer

# 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)
# 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)
M = StateMonitor(neurons,'v',record=0)

###########################################

#—————Bi-stable drift——————#
@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)
print ‘‘SDSP Weights: \n’’,synapses.W
run(SIM_TIME)
```

Appendix D

The core programming source code of epSNNr connection (spatiotemporal filter).

```
import brian
import pylab
import numpy

############################################################################


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))  # weight matrix
    D = zeros((nb_neurons, nb_neurons))   # distance matrix
    P = zeros((nb_neurons, nb_neurons))   # connection probabilistic matrix

    #### 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

            if rand() < p_conn:

                if is_excitatory[i]:
                    W[i][j]= 1.62 *mV
                else:
                    W[i][j]= -9 *mV

    #### connect the specified two neuron groups using the generated weight matrix

    Connection.connect(self, neuron_group1, neuron_group2, W)

##### Return Neural distance, coordinate and synaptic weight  matrix   ####
    return D, coordinates, W
```

Appendix E

The core programming source code of BSA for EEG.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%% BSA - BSA based spike encoder  for EEG %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function spikeTime = BSA(Twave, threshold)

%%%Twave is one EEG channel input  nomalized  to range[0,1]%%%

wave = Twave - min(Twave(:));
wave = (wave/range(wave(:)))*(1-0);
wave = wave + 0;

filter = fir1(10,0.05)
%%% for EGG need to be adjusted due to the sample rate %% %

N = length(wave);
P = length(filter);

spike = zeros(1, N);
spikeTime=[];

%%% add zeros at end so we don't have to check boundaries%%%
wave = horzcat(wave, zeros(1, P));
for i = 1:N
  segment = wave(i:i+P-1);            %% pre-cutout segment for speed %%

  if sum(abs(segment - filter)) <= sum(abs(segment)) - threshold    %% % BSA heuristic
    spike(i) = 1;   % get spike here                  %%% emmit spike
    wave(i:i+P-1) = segment - filter;  %%% substract filter from wave
  end
  if spike(i)==1

        spikeTime=[spikeTime i*0.261];

        %%% 0.261 is calculated by%%%
        %%% ---simulation time (in msec)/ number of vectors in one EEG sample --%%%

End
```

References

Abolfathi, P. P. (2009). *Toyota makes a wheelchair steered by brain waves*. Retrieved2011, from http://www.gizmag.com/toyota-wheelchair-powered-brain-waves/12121/

Arbib, M. A. (2002). *Handbook of brain theory and neural networks, Second Edition*: Cambridge, MA: MIT Press.

Bell, C., Han, V., Sugawara, Y., & Grant, K. (1999). Synaptic plasticity in the mormyrid electrosensory lobe. *The Journal of Experimental Biology, 202*, 1339–1347.

Benuskova, L., & Kasabov, N. (2007). *Computational neurogenetic modeling*: Springer Publishing Company, Incorporated.

Berber. (2011). *All nighter*. Retrieved May 1, 2011, from http://fuckingepilepsy.wordpress.com/2011/01/28/all-nighter

Berger, T. W., Chapin , J. K., Gerhardt , G. A., McFarland, D. J., Principe, J. C., Soussou, W. V., . Tresco, P. A. (2008). *Brain-Computer Interfaces: An international assessment of research and development trends*: Springer.

Bi, G., & Poo, M. (2001). Synaptic modification by correlated activity: Hebb's postulate revisited. *Annual review of neuroscience*, *24*(1), 139-166. doi:citeulike-article-id:785613 doi: 10.1146/annurev.neuro.24.1.139

Binns, C. (2009). *Control a Robot with Your Mind*. Retrieved 6th may, 2011, from http://www.popsci.com/scitech/article/2009-06/out-body-experience

Bliss, T., & Lomo, T. (1973). Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *J Physiol (Lond), 232*(2), 331-356.

Bliss, T. V., & Lomo, T. (1970). Plasticity in a monosynaptic cortical pathway. *The Journal of physiology, 207*(2), 61P.

Bohte, S. M., Poutre, H. A. L., Kok, J. N., La, H. A., Joost, P., & Kok, N. (2002). Error-Backpropagation in Temporally Encoded Networks of Spiking Neurons. *Neurocomputing, 48*, 17-37.

Bohtem, S. M., Poutre, J. A. L., & Kok, J. N. (2000). *Error-backpropagation in temporally encoded networks of spiking neurons*: CWI (Centre for Mathematics and Computer Science)

Booij, O. (2004). *Temporal Pattern Classification using Spiking Neural Networks*. University of Amsterdam, Amsterdam.

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

Brunel, N., Chance, F. S., Fourcaud, N., & Abbott, L. F. (2001). Effects of synaptic noise and filtering on the frequency response of spiking neurons. *Physical Review Letters, 86*(10), 2186-2189. doi:citeulike-article-id:584495

Burgsteiner, H., Kroll, M., Leopold, A., & Steinbauer, G. (2007). Movement prediction from real-world images using a liquid state machine. *Applied Intelligence, 26*(2), 99-109. doi:10.1007/s10489-006-0007-1

Buteneers, P., Schrauwen, B., Verstraeten, D., & Stroobandt, D. (2009). Real-time epileptic seizure detection on intra-cranial rat data using reservoir computing. *Advances in Neuro-Information Processing,* 56-63.

Cheng, L., Hou, Z.-G., Lin, Y., Tan, M., Zhang, W. C., & Wu, F.-X. (2011). Recurrent Neural Network for Non-Smooth Convex Optimization Problems With Application to the Identification of Genetic Regulatory Networks. *Trans. Neur. Netw., 22*(5), 714-726. doi:10.1109/tnn.2011.2109735

Cooper, S. J. (2005). Donald O. Hebb's synapse and learning rule: a history and commentary. *Neuroscience &amp; Biobehavioral Reviews, 28*(8), 851-874. doi:10.1016/j.neubiorev.2004.09.009

Delorme, A., & Makeig, S. (2004). EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics. *Journal of Neuroscience Methods, 134*, 9-21.

Delorme, A., & Makeig, S. (2004). EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. *Journal of Neuroscience Methods, 134*, 9-21.

Dhoble, K., Nuntalid, N., Indiveri, G., & Kasabov, N. (2012, 10-15 June 2012). Online spatio-temporal pattern recognition with evolving spiking neural networks utilising address event representation, rank order, and temporal spike learning Symposium conducted at the meeting of the Neural Networks (IJCNN), The 2012 International Joint Conference on doi:10.1109/ijcnn.2012.6252439

Egger, V., Feldmeyer, D., & Sakmann, B. (1999). Coincidence detection and changes of synaptic e±cacy in spiny stellate neurons in rat barrel cortex. *Nature Neuro-Science*, *2*, 1098-1105.

Ekanayake, H. (2010). P300 and Emotiv EPOC: Does Emotiv EPOC capture real EEG? Retrieved from http://neurofeedback.visaduma.info/emotivresearch.htm

Ferreira, A., Almeida, C., Georgieva, P., Tomé, A., & Silva, F. (2010). Advances in EEG-Based Biometry. Symposium conducted at the meeting of the ICIAR (2)'10

Florian, R. V. (2005). A reinforcement learning algorithm for spiking neural networks, Timisoara, Romania. Retrieved from http://dx.doi.org/10.1109/SYNASC.2005.13 doi:citeulike-article-id:8138417 doi: 10.1109/SYNASC.2005.13

Florian, Z. V. (2007). Reinforcement Learning Through Modulation of Spike-Timing-Dependent Synaptic Plasticity. *Neural Comput.*, *19*(6), 1468-1502. doi:10.1162/neco.2007.19.6.1468

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

Gamboa, H. (2005). Used under GNU Free Documentation and Creative Commons Licensing.

Garis, H. d., Korkin, M., & Fehr, G. (2001). The CAM-Brain Machine (CBM): An FPGA Based Tool for Evolving a 75 Million Neuron Artificial Brain to Control a Lifesized Kitten Robot. *Auton. Robots, 10*(3), 235-249. doi:10.1023/a:1011286308522

Gerstner, W. (2000). Population Dynamics of Spiking Neurons: Fast Transients, Asynchronous States, and Locking. *Neural Comput., 12*(1), 43-89. doi:10.1162/089976600300015899

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

Ghosh-Dastidar, S., & Adeli, H. (2009). A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection. *Neural Networks, 22*(10), 1419-1431.

Goel, P., Liu, H., Brown, D., & Datta, A. (2008). On the use of spiking neural network for EEG classification. *International Journal of Knowledge-Based and Intelligent Engineering Systems, 12*(4), 295-304.

Goel, P., Liu, H., Brown, D. J., & Datta, A. (2006). *Spiking neural network based classification of task-evoked EEG signals*. presented at the meeting of the Proceedings of the 10th international conference on Knowledge-Based

Intelligent Information and Engineering Systems - Volume Part I, Bournemouth, UK. doi:10.1007/11892960_99

Goodman, D., & Brette, R. (2008). Brian: a simulator for spiking neural networks in python. *Frontiers in neuroinformatics, 2*. doi:10.3389/neuro.11.005.2008

Grun, S., Diesmann, M., & Aertsen, A. (2010). *Analysis of Parallel Spike Trains* (1 ed., Vol. 7): Springer.

Grzyb, B. J., Chinellato, E., Wojcik, G. M., & Kaminski, W. A. (2009). *Which model to use for the liquid state machine?* presented at the meeting of the Proceedings of the 2009 international joint conference on Neural Networks, Atlanta, Georgia, USA.

Hamed, H. N. A., Kasabov, N., & Shamsuddin, S. M. (2012). *Dynamic Quantum-Inspired Particle Swarm Optimization as Feature and Parameter Optimizer for Evolving Spiking Neural Networks*. presented at the meeting of the International Journal of Modeling and Optimization, Retrieved from http://www.ijmo.org/papers/108-E007.pdf

Hebb, D. (1949). *The Organization of Behavior: A Neuropsychological Theory*: Wiley. Retrieved from http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&amp;path=ASIN/0805843000. doi:citeulike-article-id:500649

Hoffmann, U., Garcia, G., Vesin, J., Diserens, K., & Ebrahimi, T. (2005). A Boosting Approach to P300 Detection with Application to Brain-Computer InterfacesSPIE Symposium conducted at the meeting of the

IEEE EMBS Conference on Neural Engineering Retrieved from http://infoscience.epfl.ch/record/87218/files/Hoffmann2005_1207.pdf

Hoffmann, U., Vesin, J.-M., Ebrahimi, T., & Diserens, K. (2008). An efficient P300-based brain–computer interface for disabled subjects. *Journal of Neuroscience Methods, 167*(1), 115-125. doi:10.1016/j.jneumeth.2007.03.005

Iglesias, J., Eriksson, J., Grize, F., Tomassini, M., & Villa, A. E. P. (2006). Dynamics of pruning in simulated large-scale spiking neural networks. *Biosystems, 79*(1-3), 11-20.

Iglesias, J., & Villa, A. E. P. (2006). *Neuronal cell death and synaptic pruning driven by spike-timing dependent plasticity*. presented at the meeting of the Proceedings of the 16th international conference on Artificial Neural Networks - Volume Part I, Athens, Greece. doi:10.1007/11840817_99

Indiveri, G., Linares-Barranco, B., Hamilton, T. J., 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, 14*, 1569-1572.

Kasabov, N. (2007). *Evolving connectionist systems: The knowledge engineering approach*: Springer-Verlag New York Inc.

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

Kasabov, N. (2012a, January 2012). Evoving, Probabilistic Spiking Neural Networks and Neurogenetic Systems for Spatio- and Spectro-Temporal Data Modelling and Pattern Recognition. *Natural Intelligence: the INNS Magazine, 1*(2).

Kasabov, N. (2012b). NeuCube EvoSpike Architecture for Spatio-temporal Modelling and Pattern Recognition of Brain Signals. In N. Mana, F. Schwenker, & E. Trentin (Eds.), *Artificial Neural Networks in Pattern Recognition* (Vol. 7477, pp. 225-243): Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-33212-8_21. doi:10.1007/978-3-642-33212-8_21

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*(Autonomous Machine Learning).

Kempter, R., Gerstner, W., & van Hemmen, J. L. (1999). Hebbian learning and spiking neurons. *Phys. Rev. E, 59*(4), 4498--4514. doi:10.1103/PhysRevE.59.4498

Koch, C., & Segev, I. (1989). *Methods in Neuronal Modeling, from synapses to networks*. Cambridge: MIT Press.

Legenstein, R., Naeger, C., & Maass, W. (2005). What Can a Neuron Learn with Spike-Timing-Dependent Plasticity? *Neural Comput., 17*(11), 2337-2382. doi:10.1162/0899766054796888

Lotze, M., Montoya, P., Erb, M., H lsmann, E., Flor, H., Klose, U., . Grodd, W. (1999). Activation of Cortical and Cerebellar Motor Areas during

Executed and Imagined Hand Movements: An fMRI Study. *J. Cognitive Neuroscience, 11*(5), 491-501. doi:10.1162/089892999563553

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., Natschläger, T., & Markram, H. (2004). Computational Models for Generic Cortical Microcircuits. In *Computational Neuroscience: A Comprehensive Approach* (pp. 575-605): Chapman & Hall/CRC. Retrieved from http://infoscience.epfl.ch/record/117815

Maass, W., & Zador, A. (1999). Computing and learning with dynamic synapses. *Pulsed neural networks, 6*, 321-336.

MacGregor, R. J. (1987). *Neural and Brain Modeling* San Diego: Academic Press,.

Mak, J. N., McFarland, D. J., Vaughan, T. M., McCane, L. M., Tsui, P. Z., Zeitlin, D. J., . Wolpaw, J. R. (2012). Technology-aided programs for assisting communication and leisure engagement of persons with amyotrophic lateral sclerosis: Two single-case studies. *Research in Developmental Disabilities*, *33*(5), 1605.

Marcel, S., & Millán, J. R. (2007). Person authentication using brainwaves (EEG) and maximum a posteriori model adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 743-752.

Markram, H., Lübke, J., Frotscher, M., & Sakmann, B. (1997). Regulation of Synaptic Efficacy by Coincidence of Postsynaptic APs and EPSPs.

Science, 275(5297), 213-215. doi:citeulike-article-id:854200 doi: 10.1126/science.275.5297.213

Mohemmed, A., Schliebs, S., & Kasabov, N. (2011). *SPAN: A Neuron for Precise-Time Spike Pattern Association* [ICONIP (2)]. Springer.

Mohemmed, A., Schliebs, S., Matsuda, S., & Kasabov, N., &. (2011). *Method for Training a Spiking Neuron to Associate Input-Output Spike Trains* [EANN/AIAI (1)]. Springer.

Murph, D. (2007). *IMEC reveals wireless EEG headband, Geordi La Forge approves*. Retrieved May 1, 2011, from http://www.engadget.com/2007/11/01/imec-reveals-wireless-eeg-headband-geordi-la-forge-approves

Nawrot, M. P., Schnepel, P., Aertsen, A., & Boucsein, C. (2009). *Precisely timed signal transmission in neocortical networks with reliable intermediate-range projections* (Vol. 3). Retrieved from http://www.biomedsearch.com/nih/Precisely-timed-signal-transmission-in/19225575.html

Niedermeyer, E., & Da Silva, F. H. L. (2005). *Electroencephalography: basic principles, clinical applications, and related fields*: Lippincott Williams & Wilkins.

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

Nuntalid, N., Dhoble, K., & Kasabov, N. (conf/iconip/NuntalidDK11). (2011). *EEG Classification with BSA Spike Encoding Algorithm and Evolving Probabilistic Spiking Neural Network* [Proceedings of the 18th international conference on Neural information processing: theory and algorithms - Volume Part I]. Changhai, China: Springer.

Palaniappan, R., & Mandic, D. P. (2007). EEG Based Biometric Framework for Automatic Identity Verification. *J. VLSI Signal Process. Syst., 49,* 243-250.

Paugam-Moisy, H., & Bohte, S. M. (2009, September). Handbook of Natural Computing*Springer-Verlag.* Abstract retrieved from Liris-4305

Pecevski, D., Natschläger, T., & Schuch, K. (2009). PCSIM: a parallel simulation environment for neural circuits fully integrated with Python [Original Research]. *Frontiers in Neuroinformatics, 3*. doi:10.3389/neuro.11.011.2009

Ponulak, F., & Kasiski, A. (2010). Supervised learning in spiking neural networks with resume: Sequence learning, classification, and spike shifting. *Neural Comput., 22*(2), 467-510. doi:10.1162/neco.2009.11-08-901

Rieke, F., Warland, D., Rob, W., & Bialek, W. (1997). *Spikes: Exploring the Neural Code*. Cambridge: MIT Press.

Rieke, F., Warland, D., van Steveninck, R. R., & Bialek, W. (1999). *Spikes: exploring the neural code*: MIT press.

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Washington: Spartan Books.

Rumelhart, D. E., & McClelland, J. L. (1986). *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*. Cambridge: MIT Press.

Schliebs, S., Defoin-Platel, M., Worner, S. P., & 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., Nuntalid, N., & Kasabov, N. (2010). *Towards spatio-temporal pattern recognition using evolving spiking neural networks* [Proceedings of the 17th international conference on Neural information processing: theory and algorithms - Volume Part I]. Sydney, Australia: Springer-Verlag.

Schliebs, S., Nuzly Abdull Hamed, H., & Kasabov, N. (2011). *Reservoir-based evolving spiking neural network for spatio-temporal pattern recognition*. presented at the meeting of the Proceedings of the 18th international conference on Neural Information Processing Shanghai, China.

Schrauwen, B., D'Haene, M., Verstraeten, D., & Campenhout, J. V. (2008). 2008 Special Issue: Compact hardware liquid state machines on FPGA for real-time speech recognition. *Neural Netw., 21*(2-3), 511-523. doi:10.1016/j.neunet.2007.12.009

Schrauwen, B., & Van Campenhout, J. (2003). *BSA, a fast and accurate spike train encoding scheme* [Proceedings of the International Joint Conference on Neural Networks].

Schrauwen, B., Verstraeten, D., & Campenhout, J. V. (2007). *An overview of reservoir computing: theory, applications and implementations* [Proceedings of the 15th European Symposium on Artificial Neural Networks].

Seung, H. S., & Hughes, H. (2003). Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. *Neuron*.

Shadlen, M., & Newsome, W. (1996). Motion perception: seeing and deciding Symposium conducted at the meeting of the National Academy of Sciences (USA)

Song, S., Miller, K. D., Abbott, L. F., & others. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *nature neuroscience, 3*, 919-926.

Swartz, B. E., & Goldensohn, E. S. (1998). Timeline of the history of EEG and associated fields. *Electroencephalography and clinical Neurophysiology, 106*(2), 173-176.

Tatum, W. O. (2007). *Handbook of EEG interpretation*: Demos Medical Publishing.

Thorpe, S., Delorme, A., & Van Rullen, R. (2001). Spike-based strategies for rapid processing. *Neural Networks, 14(6-7)*, 715–725.

Thorpe, S., & Gautrais, J. (1998). Rank order coding. *Computational neuroscience: Trends in research, 13*, 113-119.

Thorpe, S., & Gautrais, J. (1998). *Rank order coding*. presented at the meeting of the Proceedings of the sixth annual conference on Computational neuroscience : trends in research, 1998: trends in research, 1998, Big Sky, Montana, United States.

Torikai, H., & Nishigami, T. (2009). *A novel chaotic spiking neuron and its paralleled spike encoding function* [Proceedings of the 2009 international joint conference on Neural Networks]. Atlanta, Georgia, USA: IEEE Press.

Toups, J., Fellous, J., Thomas, P., & Sejnowski, T. (2012). Multiple Spike Time Patterns Occur at Bifurcation Points of Membrane Potential Dynamics. *PLOS Computational Biology Organization, 8*(10). doi:10.1371/journal.pcbi.1002615

Tymoshchuk, P., & Kaszkurewicz, E. (2005). A winner-take-all circuit using neural networks as building blocks. *Neurocomput., 64*, 375-396. doi:10.1016/j.neucom.2004.08.002

Verstraten, D., Schrauwen, B., Stroobandt, D., & Van Campenhout, J. (2005). Isolated word recognition with the liquid state machine: a case study. *Inf. Process. Lett., 95*, 521-528.

Villa, A. E. P., Tetko, I. V., Hyland, B., & Najem, A. (1999). Spatiotemporal activity patterns of rat cortical neurons predict responses in a conditioned task. *Proceedings of the National Academy of Sciences, 96*(3), 1106-1111.

von der Malsburg, C. (1983). *The Correlation Theory of Brain Function*. Retrieved from http://books.google.co.nz/books?id=U0LDtgAACAAJ

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

Xie, X., & Seung, H. S. (2003). Equivalence of backpropagation and contrastive Hebbian learning in a layered network. *Neural Comput., 15*(2), 441-454. doi:10.1162/089976603762552988

Xu, N., Gao, X., Hong, B., Miao, X., Gao, S., & Yang, F. (2004). BCI Competition 2003-Data set IIb: enhancing P300 wave detection using ICA-based subspace projections for BCI applications. *IEEE transactions on bio-medical engineering, 51*(6), 1067-1072.

Yamazaki, T., & Tanaka, S. (2007). The cerebellum as a liquid state machine. *Neural Networks, 20*(3), 290-297.