Design and Creative Technologies
School of Engineering, Computer and Mathematical Sciences
Knowledge Engineering and Discovery Research Institute (KEDRI)

Neuromorphic Computational Models for Machine Learning and Pattern Recognition from Multi-modal Time-series Data

Neelava Sengupta

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

Supervisors Nikola Kasabov, Denise Taylor and Grace Wang

Neelava Sengupta

Neuromorphic Computational Models for Machine Learning and Pattern Recognition from Multi-modal Time-series Data

Computer and Information Sciences, July 4, 2018

Supervisors: Nikola Kasabov, Denise Taylor and Grace Wang

Auckland University of Technology

Knowledge Engineering and Discovery Research Institute (KEDRI)

School of Engineering, Computer and Mathematical Sciences

Design and Creative Technologies

55 Wellesley St. E

1010 Auckland

Abstract

The fields of neuroscience and artificial intelligence have a long and entwined history. In recent times, however, communication and collaboration between the two fields has become a rarity as they have evolved. Written in the era when artificial intelligence and deep learning is revolutionising the world, this thesis revisits and searches for inspiration from biological intelligence. The efficiency and accuracy with which the human brain processes incoming stimulus (data) in millisecond resolution using remarkably low power is unprecedented. Motivated by this very capability in the generic sense, this thesis has focused on developing neurobiologically inspired computational models known as spiking neural networks to tackle multi-modal time-series data. In a more definitive formalisation, this work has aimed to answer three research questions:

1. How to optimally design an implementation of neuromorphic architecture which is capable of processing large volumes of spatio-temporal data? To answer this research question, the unsupervised SNNc algorithm (as part of NeuCube architecture) were studied and numerous designs of the SNNc graph was analysed in regards to storage and execution time complexities. Further, the study was extended to include an analysis of the software design principles for achieving modularity and heterogeneity. The design principles formalised here are implemented in the NeuCube software publicly available from www.kedri.aut.ac.nz/neucube. The design principles proposed in the study are also utilised in other parts of this work.

2. How to perform neural encoding on real-world data to represent information as spike-timings? This topic has been analysed from the viewpoint of data compression and information theory. To answer this research ques-

tion, the focus was on using temporal encoding as a framework for concise representation of large volumes of data. Apart from comparing the state of the art temporal encoding algorithms from literature, a novel *a priori* knowledge driven temporal encoding framework was formalised and an algorithmic realisation of it for fMRI data, called GAGamma, was proposed. The temporal encoding algorithms on benchmark fMRI data was experimentally evaluated to demonstrate its superiority to succinctly represent the discriminatory information in the data without appreciable information loss. It was also demonstrated that the proposed encoding framework provides enhanced flexibility to include *a priori* knowledge of the data source and thus, provide the compression/encoding algorithms sufficient redundancy to compress large datasets in an optimally concise manner.

3. How to recognise patterns from multi-modal data with spatial, temporal and orientation information using neuromorphic architectures? To answer this research question, a novel, unsupervised learning algorithm, namely oiSTDP learning algorithm, was proposed for fusing temporal, spatial and orientation information in a spiking neural network architecture. Furthermore, a case study is presented on building a computational model that discriminates between people with schizophrenia who respond or do not respond to mono-therapy with the anti-psychotic clozapine. The performance of the proposed algorithm as part of the modified NeuCube framework has demonstrated superiority against the state of the art deep learning algorithm not only in prediction performance but also degree of interpretability.

Overall, through this work, the researcher has presented several novel approaches of recognising patterns in time series data using spike-timings as the unit of data representation in the neuromorphic computation framework. The several software design challenges that arose due to the nature of neuromorphic computation framework has also been addressed in this thesis.

Contents

List of Figures

List of Tables

List of Algorithms

List of code snippets

Declaration

I, Neelava Sengupta, declare that this thesis titled, Neuromorphic Computational Models for Machine Learning and Pattern Recognition from Multi-modal Time-series Data and the work presented in it are my own. I confirm that:

- This work was done wholly while in candidature for a research degree at this University.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Auckland, July 4, 2018

Neelava Sengupta

Acknowledgement

I would like to take this opportunity to thank all the people without whom this study would never have been possible. Although it is just my name on the cover, it has been a collaborative effort as many have in their own special way helped and for that I want to give them special congratulations.

First, my supervisors Prof. Nikola Kasabov, Prof. Denise Taylor and Dr. Grace Wang. Nik, you have created the invaluable space for me to do this research and develop myself as a researcher in the best possible way. I greatly appreciate the freedom you have given me to find my own path, and the guidance and support you offered when needed. I highly appreciate you continuously pushing me to make this piece of work better. Denise and Grace, thank you for being an incredibly valuable part of this journey of mine and providing the guidance when required. I truly hope that you will give me the opportunity to work even closer together with you in the future. A special thanks also goes out to my examiners, Prof. Chrisina Jayne and Prof. Giacomo Indiveri for taking time to read and examine my thesis.

I would also want to extend my gratitude towards Mrs. Joyce D'mello, the administrative manager of KEDRI. Joyce, you are the lifeline of KEDRI and I, among many, am indebted to you for all that you have done to make this journey worth. Thank you for helping me keep it all together in difficult times.

Next, some people of outstanding importance for my research. Carolyn, Nathan and Anne, thank you for accepting to work alongside me, and for having numerous valuable conversations, all of which have shaken parts of my grey matter and resulted in this work. Our discussions were always lively, and although generally longer than planned, have always ended too soon. I hope to continue our collaborations in the future. I would also like to extend my acknowledgement to all the KEDRI members, especially Vivienne, Israel, Reggio, Fahad, Muhaini, Kaushalya, Clarence, Elisa, Maryam, Zohreh, Akshay, and everyone else I have had an opportunity to interact with. I also wish all of you all the very best for your future.

I would not be sitting here today writing an acknowledgement for my thesis without the selfless support and encouragement of my family. Shumi, my dear wife, thank you for supporting me day in and day out throughout the ups and downs of this journey. This work is more yours than mine. Also, I thank you for your innumerable iterations of reading and proofreading this text. I am sure your PhD journey will much more enjoyable and you will always have my support on yours. Maa and baba, no words can ever describe my gratefulness to you and all the contributions you have made, quietly, to make this happen. Thank you for being my parents. I hope this work makes you proud. And of course to my precious little dogs, Burfi and Golu, who have always kept me going with all the love and cuddles. This writing is incomplete without a mention of all my family and friends in New Zealand, India and around the world who have continuously influenced me directly or indirectly over the years.

And finally, to my late maa and baba. This is for you. This is a gift that you deserve. May god bless you and may you rest in peace together. You are always and forever will be in my thoughts and prayers.

1 Introduction

In the recent past, significant progress has been made in neuroscience and artificial intelligence (AI). With the inception of computers in the early nineteenth century, progress in AI was inextricably intertwined with neuroscience and psychology, and many of the early pioneers straddled both fields, with collaborations between these disciplines proving highly productive (Turing, 1950; Hinton, Mcclelland & Rumelhart, 1986; McCulloch & Pitts, 1943; Hebb, 1949; Churchland & Sejnowski, 1988). Nevertheless, progress in recent times has seen interactions and collaborations becoming less commonplace, as both fields have evolved massively in complexity, and disciplinary boundaries have solidified.

The development of AI began with the premise that creating human like general purpose artificial intelligence (or Turing-powerful intelligent systems) is an intimidating task, due to the massive search space of sparsely populated solutions. This underscored the utility of examining the principles of the human brain which is the only existing proof that such an intelligence is even possible. As described by Hassabis, Kumaran, Summerfield and Botvinick (2017), the benefits of developing AI by close scrutinisation of biological intelligence are two-fold. First, neuroscience provides a rich source of inspiration for new algorithms and architectures, independent of and complementary to the mathematical and logic-based methods, and ideas that have largely dominated traditional approaches to AI. For example, where a new facet of biological computation found to be critical to supporting a cognitive function, then it has been considered as an excellent candidate for incorporation into artificial systems. Second, neuroscience can provide validation of artificial intelligence techniques that already exist. If a known algorithm is subsequently found to be implemented in the brain, then that is strong support for its plausibility as an integral component of an overall general intelligence system.

1.1 Rationale and Motivation

It is without a doubt that in the era of ‘smart’ everything, availability and utility of massive volumes of digital data plays a substantial role. Historically, data was used as a part of business and gathered to serve specific needs. For example, retailers recorded sales for accounting, manufacturers recorded raw materials for quality management and the number of mouse clicks on advertising banners was collected for calculating advertisement revenue. But as the demand for big data analytics emerged, data no longer served only its initial purpose. Organisations were able to access huge amounts of data and possess a valuable asset that when combined with the ability to analyse it, has given rise to a whole new industry.

This thesis began its journey in the middle of a technological revolution in the year 2014 when the industry was booming with the following technological revolutions:

- Big data: The total volume of digital data has grown into Zetabytes and is projected to be reaching 180 Zetabytes by the year 2025. This enormous growth in digital data has brought on an era of "Big data". Big data is a term applied to data sets whose size or type is beyond the capabilities of traditional relational databases to capture, manage, and process the data with low-latency. Also, it has one or more of the following characteristics: high volume, high velocity, or high variety. Big data traditionally originates from Internet of Things (IoT) enabled sensors, devices, video/audio, networks, log files, transactional applications, and social media - much of it generated in real time and on a very large scale.

- Deep learning: Deep learning is a specialised stream of AI which has come into prominence in the past decade. Deep learning is considered to be an extension of machine learning, which consists of a set of technologies around neural networks that empowers computers to learn, evolve and improve upon their own learning by reiterating and consistently updating the data bank through recursive experiments and human intervention. Deep learning has already revolutionised AI and pushed the boundaries of intelligent systems through technologies such as autonomous cars, fully automated language translation and speech transcription systems.

- High performance computing (HPC): Experts have described deep learning as analogous to a rocketship that needs a really big engine (a model) and a lot of fuel (the data) in order to go anywhere interesting. A big model that learns from big data, of course, requires large scale computational capabilities, or, in other words, high performance computing for it to be efficient and effective. The need for HPC has also meant a massive progress in the form of cloud computing and infrastructure as a service providing capability to scale infrastructure requirements (computation, storage, network speed) on a demand basis.

The state-of-the-art of AI is composed of big data, big models and big infrastructure where solutions are traditionally developed by stacking up resources in the form of multiple instances of computers (physical and/or logical). The main motivation and/or question that drives the present research in this domain is the following: "Is stacking up computers an ideal solution for the future direction of AI?" This question led to the revisiting of the field of neuroscience seeking inspiration from biological intelligence for efficient and accurate computing.

An important distinction of the human brain from AI is its ability to operate at full capacity using orders of magnitude lower power. This is a rather subtle hint that the human brain does not scale by stacking up computational units. Rather, the efficiency is due to the ability of the human brain to represent sensory information as electrical impulses. The idea of information representation as electrical impulses is at the base of this thesis. The present work concentrates on developing algorithms for pattern recognition, not by stacking layers of computational units and infrastructure, rather by developing algorithms for representing, processing and recognising patterns in big data, especially time-series data, efficiently and accurately.

1.2 Aims of this Thesis and Research Questions

On the basis of the rationale and motivations discussed previously, this work constrains its focus on the development of neuromorphic computational models for recognising patterns using multi-modal time-series data.

1. Neuromorphic models: This thesis concentrates on developing neuro-biology inspired computational models known as spiking neural networks for recognising patterns in data. The most important characteristic ensuring neuromorphic behaviour is the ability to represent, transform and process spike-time data.

2. Multi-modal time-series data: As it has been briefly described earlier, the concept of big data is characterised not only by volume and velocity, but also very importantly, by variety. The data sources in the era of IoT are diverse, leading to multi-modal data with a variety of spatio-temporal properties. The aim is to develop technologies that has the capacities to deal with such multi-modal data sources.

3. Time-series, especially brain data: The studies in this thesis are also limited to experiments performed on time-series data, especially, non-invasive brain data, such as functional Magnetic Resonance Imaging (fMRI), Electroencephalography (EEG), and Diffusion Tensor Imaging (DTI). The brain activity data is particularly useful as data sources of multi-modal nature with prominent impact on healthcare for several brain related diseases. This will be discussed in detail later.

Keeping the above mentioned constraints in mind, this research aims to answer the following more in-depth research questions:

1. **Research question 1**: How to design architectures of spiking neural networks that are capable of digesting and processing large volumes of spatio-temporal data? This research question focuses on software design principles, including the data structure design, modularity and scalability aspects of spiking neural network architectures.

2. **Research question 2**: How to perform neural encoding on real-world data to represent information as timings of spikes? This research question can be observed in the light of information theory and theories of lossy data compression.

3. **Research question 3**: How to integrate spatial, temporal and orientation information present in multi-modal brain data using spiking neural network architecture?

1.3 Thesis Structure, Navigation and List of Peer-reviewed Publications

This thesis is presented in eight Chapters. Figure 1.1 depicts a self-explanatory bird's eye view of the relationships between the different components of interest of this thesis. It is highly recommended to review this figure before proceeding further. The components are categorised as Chapters, topics, research questions, tangible outcomes (software implementations or algorithms) and publications (journals, conferences or book chapters). There are three categories of relationships (arrows in the diagram) drawn in the figure: (1) Contains: These are one-to-many relationships. A relationship of this kind can be observed between a chapter and multiple topics. (2) Dependence: These are directed one-to-one relationships. The one-to-one directed relationships are drawn between chap-

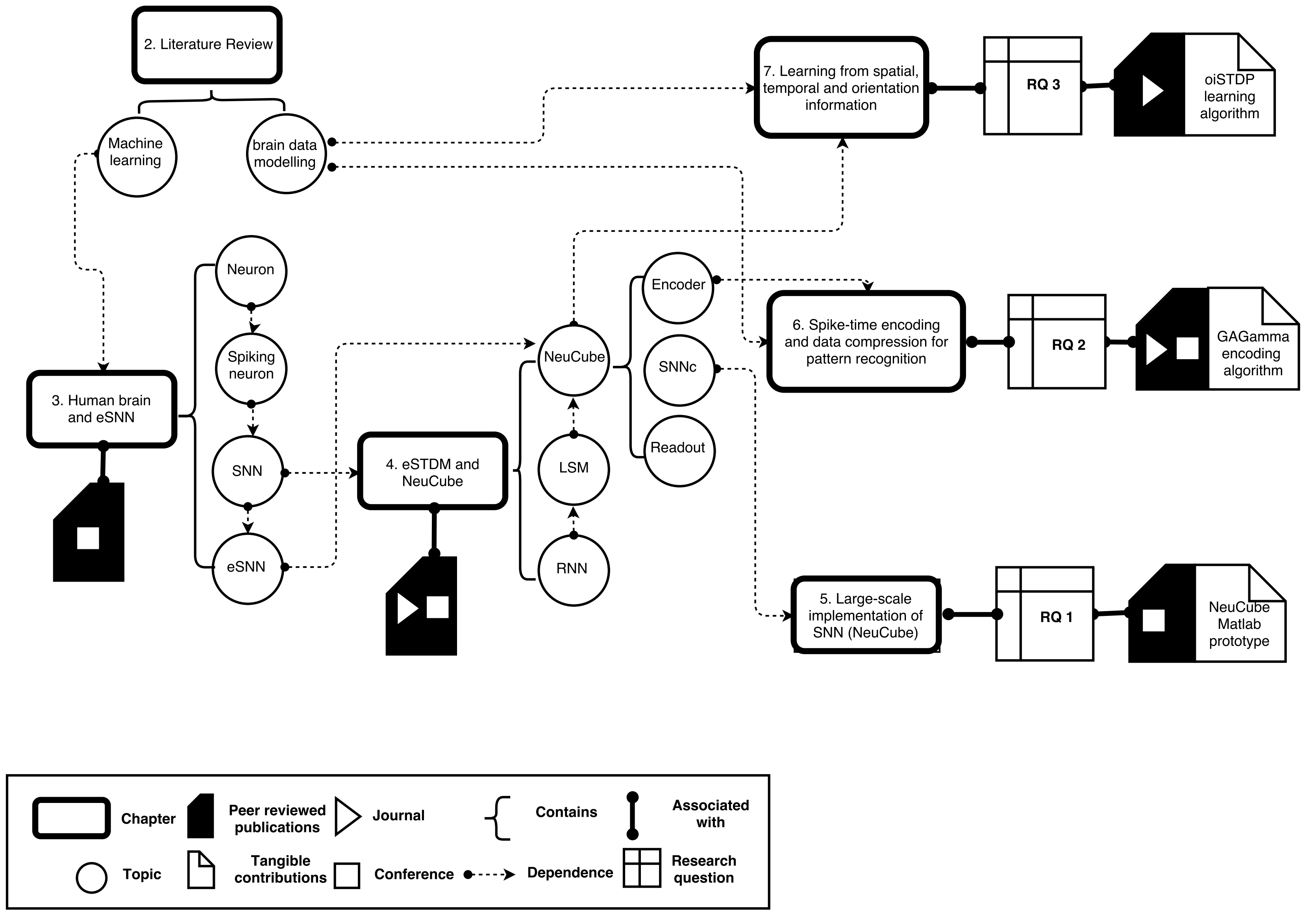


Figure 1.1.: Bird's eye view of thesis components and their inter-dependence.

ters and/or topics. Following the dependence graph, the reader can navigate through the prerequisites and/or flow of topics. (3) Associated with: These are undirected one-to-one relationships associating a pair of components (such as a Chapter and a research question, or a Chapter and a publication). An example of a possible navigation through the diagram is as follows: To read Chapter 7 for example (which is associated with research question 3 and resulted in one journal publication and a tangible outcome in the form of oiSTDP learning algorithm), a reader is recommended to read Chapter 2, especially the topics on brain data modelling and Chapter 4, especially the topics related to NeuCube (which also has certain prerequisites).

This thesis generally follows a linear and continuous flow. This means that efforts have been made throughout the thesis to cross-reference across chapters and sections in order to minimise repetitions. Chapter 5, 6 and 7 are presented as the main contributions of this thesis and directly relates to the research questions discussed in section 1.2. Chapters 3 and 4 on the other hand forms the basis on which the main chapters are written and many of the concepts are introduced in these chapters and reused thereafter in the main contributions. The following table lists the contributions that have been made in the form of peer reviewed publications through the work of this thesis.

Table 1.1.: List of peer-reviewed publications

Type	Publication	Year
Journal	**Sengupta, N.**, McNabb, C. B., Kasabov, N. & Russell, B. (2018), integrating space, time and orientation in spiking Neural Networks: A case study on multi-modal brain data modelling, IEEE Transactions on Neural Networks and Learning System. DOI: https://doi.org/10.1109/TNNLS.2018.2796023.	2018
Journal	**Sengupta, N.**, & Kasabov, N. (2017). Spike-time encoding as a data compression technique for pattern recognition of temporal data. Information Sciences, 406, 133-145.	2017
Journal	Kasabov, N., Scott, N. M., Tu, E., Marks, S., **Sengupta, N.**, Capecci, E., ... & Espinosa-Ramos, J. I. (2016). Evolving spatio-temporal data machines based on the NeuCube neuromorphic framework: design methodology and selected applications. Neural Networks, 78, 1-14.	2016
Conference	**Sengupta, N.**, Scott, N., & Kasabov, N. (2015). Framework for knowledge driven optimisation based data encoding for brain data modelling using spiking neural network architecture. In Proceedings of the Fifth International Conference on Fuzzy and Neuro Computing (FANCCO-2015) (pp. 109-118). Springer, Cham.	
Conference	Abbott, A., **Sengupta, N.** & Kasabov, N. (2016). Which method to use for optimal structure and function representation of large spiking neural networks: A case study on the Neucube architecture. In 2016 international joint conference on neural networks(ijcnn)(pp. 1367-1372). IEEE	2016
Conference	Kasabov, N., **Sengupta, N.**, & Scott, N. (2016, September). From von neumann, John Atanasoff and ABC to Neuromorphic computation and the NeuCube spatio-temporal data machine. In IEEE 8th International Conference on Intelligent Systems (IS), 2016 (pp. 15-21). IEEE.	2016
Conference	Arya, A. S., Ravi, V., Tejasviram, V., **Sengupta, N.**, and Kasabov, N. (2018, January) Cyber fraud detection using evolving spiking neural network, in IEEE International conference on industrial and information systems (ICIIS), 2016 (pp. 263-268). IEEE.	2016
Chapter	**Sengupta, N.**, Ramos, J.I.E., Tu, E., Marks, S., Scott, N.,... & Abbott, A. (2018). From von Neumann architecture and Atanasoffs ABC to Neuromorphic Computation and Kasabovs NeuCube: Principles and Implementations, Learning Sytems: From Theory to Practice. Springer.	2018

2 Literature Review on Brain Data Modelling and Machine Learning

2.1 A Review on Modelling Brain Data

The human curiosity to understand the neural activity associated with cognition and perception is as old as the field of neuroscience itself. During its inception, neuroscience studies were focused on invasive techniques for modelling the brain. This strand of research suffered from two major disadvantages:

1. The studies were non-functional studies due to their invasive nature (Geschwind, 1974; Shallice, 1988).

2. These studies needed to be implemented on non-human subjects (Felleman & Van Essen, 1991; Pandya & Yeterian, 1985) due to the risk associated with invasive procedures.

Due to these limitations, therefore, for many years, neuroscientists faced great difficulties in measuring the neural activity in the mammalian brain, while they performed tasks or simply rested. The advent of non-invasive techniques, such as electrical activity (Electroencephalography, Electromyography, Electronystagmography), haemodynamic activity (fMRI), water diffusion (DTI), and others, allowed for indirect measurements of neural activity and connectivity, leading to the affordance of a new departure of computational neuroscience.

- Functional Magnetic Resonance Imaging (fMRI): Magnetic resonance imaging (MRI) applies an external magnetic field to align the magnetic moments of hydrogen ions in the body and then applies a radio frequency pulse, which flips the ions in the opposite direction. The recovery of these hydrogen ions back to the direction of the external magnetic field produces a current, which generates the black and white image in an MRI photo. Resting state functional magnetic resonance imaging (rs-fMRI) measures changes in blood oxygenation within the brain during rest. When neurons are activated in the brain, they expend energy, which needs to be replaced. This is achieved via an increase in blood flow to the active area (occurring between 6 and 10 seconds after activity is initiated). This increase in blood flow produces a net decrease in deoxygenated haemoglobin levels, changing the magnetic properties of the blood. Areas of high activity are consequently associated with higher signal in the MRI and this can be measured across time.

- Electroencephalography (EEG): EEG is a technique for measuring the electrical activity of the brain. Silver chloride electrodes placed on the scalp record the integrated activity of a large number of neurons within the brain. Only neural structures with a specific spatial organisation (*i.e.*, layers of neurons with all cell bodies/dendrites facing in the same direction, such as the cortex, thalamus and cerebellum) can generate scalp potentials. Consequently, non-invasive EEG can only measure a subset of the overall activity of the brain. Despite this limitation, however, EEG is a valuable tool for investigating abnormalities in brain function and provides a useful data source for classifying patient populations with known changes in their EEG output.

- Diffusion Tensor Imaging (DTI): DTI measures the net movement of water within the brain. Water molecules outside of the cell have equal probability of movement in every direction. However, when water is trapped within a neuronal cell (within an axon or dendrite), its diffusion is restricted to movement along the direction of the axon or dendrite. The diffusion direction of these trapped water molecules can be measured using DTI, providing a map of neuronal tracts (white matter) within the brain.

Numerous studies have focused on analysing and modelling neural activities related to cognitive or non-cognitive tasks, using single modalities of non-invasive techniques, such as EEG, fMRI or DTI. Functional Magnetic Resonance (fMRI) was used by Boksman et al. (2005) to study brain connectivity related to word fluency during the first episode of schizophrenia. Supekar, Menon, Rubin, Musen and Greicius (2008) used fMRI to study brain connectivity in Alzheimers disease. Diffusion Tensor imaging (DTI) was also employed in the works of Eluvathingal et al. (2006), Price et al. (2007) for cognitive studies. Ravan, Hasey, Reilly, MacCrimmon and Khodayari-Rostamabad (2015) used EEG data for studying the effect of Clozapine therapy in schizophrenia. Cabeza and Nyberg (2000) studied fMRI and PET, two modalities of neuroimaging to explore the functional anatomy of different cognitive functions, such as, attention, perception, language and so on. McClure et al. (2007) studied a predictive model for treatment of anxiety disorder in young children. DTI is used by Hamstra, Rehemtulla and Ross (2007), as a biomarker for treatment response in oncology. Gordon (2007) discussed using neuromarkers like fMRI and DTI for brain-related personalised medicine and treatment. Apart from these, applications, such as, Brain Computer Interfaces (BCI), neuroprosthetics, neuro-rehabilitation are a few of many other areas, which can potentially gain from brain data modelling and analysis. All of the research work discussed above show the trend of modelling a single modality of brain data. This is primarily due to the technological constraints associated with simultaneous acquisition of multiple modalities of brain data. Concurrent recording suffers from several technical difficulties, such as, ballistocardiographic artefact, MRI pulse artefact and others, leading to high noise to signal ratio. Current technical advances in MRI and EEG recordings, though, have overcome the aforementioned difficulties and allow for simultaneous recording of mixed modalities of brain data (Menon & Crottaz-Herbette, 2005; Horovitz, Skudlarski & Gore, 2002). This is ground breaking, as fused analysis of multiple data types is potentially more informative about the complex brain activity during the measured task. Until recently, the most commonly used method of integrated data analysis for this kind of problem was by converging evidence (Horwitz & Poeppel, 2002). Typically, each data type is analysed separately, and the results from other analyses that support one's finding are discussed in the discussion Section. A standard sentence looks like "the activation that was found in region X is consistent with studies in patients with focal lesion in region X and in ERP studies, where a latency in the Y component on task Z has been observed during.." (Horwitz & Poeppel, 2002). Horwitz and Poeppel (2002) also discussed an alternative data fusion analysis called computational neural modelling. This is done by creating biologically realistic neural network models where each network simulates data of a certain type and is compared with observed data. One major setback of this paradigm of data analysis is that the hypothesis-driven neural network model is built under several assumptions for simulated data generation. Hence, it is difficult to know, whether any lack of agreement between observed and simulated data is due to the assumptions in the model, or whether it is simply wrong.

There is a third alternative for multi-modal data integration, known as direct data fusion (George et al., 1995). Direct data fusion can be loosely defined as, the technique for directly fusing multiple datasets using statistical and machine learning algorithms. BieSSmann, Plis, Meinecke, Eichele and Muller (2011) presented a detailed description and analysis of this approach. In this approach, if one is interested in the neural potentials induced by a certain stimulus event, one averages epochs of EEG time series aligned to the presentation of that stimulus. If one is not interested in the exact phase of the neural response, one often extracts the amplitude modulations of neural oscillations in a certain frequency band (e.g. (Laufs et al., 2006)). For fMRI data, one typically extracts patterns of activity that are correlated with the time course of the experimental stimulus. Several exploratory unsupervised methods are also proposed for fused data analysis. In (Eichele et al., 2008), temporal ICA is applied to EEG, and spatial ICA is applied to fMRI data. Other unsupervised feature extraction techniques include microstate analyses (Brandeis & Lehmann, 1989; Patil, Carmena, Nicolelis & Turner, 2004), which is based on clustering to find quasi-stable topographical EEG scalp maps whose time courses can be compared with fMRI signals (Muthukumaraswamy & Singh, 2008). This thesis intends to emphasize the broad technique of direct data fusion. Next, a brief review of the literature on statistical and machine learning treatment of brain data will be presented.

From a methodological perspective, there exists several statistical techniques for brain data analysis. Conceptually, they can be divided into two different groups according to Horwitz, Tagamets and McIntosh (1999):

- Subtraction paradigm: This paradigm hypothesizes the functional specialisation, that different brain regions are associated with different brain functions (Fristen, 1997; Posner, Petersen, Fox & Raichle, 1988). It compares the signals between sets of scans, where each set represents a different experimental condition. The locations of large differences in signals between the two presumably delineated brain regions differentially involved in the two conditions. This paradigm is implemented using different forms of univariate (Woolrich, Ripley, Brady & Smith, 2001) feature by feature approaches, and does not take into account the influence of the spatial neighbourhood, considering each feature activity as statistically independent.

- Covariance paradigm: The assumption in covariance paradigm is that any experimental condition is mediated by a network of interacting regions of interest (ROI), and different functional tasks relate to different functional networks (Horwitz, Soncrant & Haxby, 1992; Mesulam et al., 1990). This paradigm focuses on correlation (Horwitz, Grady et al., 1992), covariance and regression (Friston et al., 1997) to analyse the relationships between brain regions and producing activity maps of the ROIs. Until recently, the General Linear Model (GLM), a form of statistical linear model was used to perform multivariate statistical modelling of neuroimaging data (Beckmann, Jenkinson & Smith, 2003; Calhoun, Stevens, Pearlson & Kiehl, 2004), and it was integrated in dedicated neuroimage analysis tools like SPM (Friston, 1994) and AFNI (Cox, 1996).

Recently, the machine learning community collaborated with computational neuroscience, which led to the application of several machine learning techniques in cognitive pattern recognition. The ability of pattern recognition algorithms to map neural activity to mental states led to the practical realisation of mind reading (Norman, Polyn, Detre & Haxby, 2006). A study by Haxby et al. (2001) illustrated how multi voxel pattern analysis can be used to distinguish cognitive states. Similar methods were also applied to discriminate the viewing of different orientation of stripes (Haynes & Rees, 2005; Kamitani & Tong, 2005), movement direction of a field of dots (Kamitani & Tong, 2006), determine whether a subject is viewing a picture or a sentence, and whether a subject was reading an ambiguous or an unambiguous word, and the semantic category of a viewed object (Mitchell et al., 2004). The University of Pittsburg organised brain competitions in the years 2006 and 2007 that aimed to predict dynamic experience in a virtual reality environment. In 2009, a brain connectivity challenge was organised by them that aimed to map the cables of human brain mapping approximately $300,000$ fibres streamline into $20-50$ cables or tracts. In all of the above mentioned studies, advanced pattern classification algorithms like Gaussian Nave Bayes (Mitchell et al., 2004), Multilayer Perceptron (MLP) and Kernel Ridge Regression (Chu, Ni, Tan, Saunders & Ashburner, 2011) were used as discriminative models and achieved significant classification accuracy.

2.2 A Review of Machine Learning

The problem of searching and recognising patterns in data is a fundamental form of science which has a long and fruitful existence in the history of the human race. One of the earliest instances of the successful pattern recognition endeavour was the discovery of empirical laws of planetary motion by Johannes Kepler in the 16th century AD. The area of pattern recognition aims at automated discovery of regularities in data through the usage of computational algorithms, and with the use of these regularities, to take action on the task in hand (Bishop, 2006). Over the years, several approaches of pattern recognition have come into prominence, the most primitive of which is hand crafted heuristics based on known knowledge, as used by Kepler during the discovery of empirical laws of planetary motion using radio astronomy data.

In this work, however, the interest lies in a more sophisticated and automated approach called machine learning. A machine in the form of a computer program is formally said to "learn from experience E with respect to some task T and performance measure P if its performance at task in T, as measured by P, improves with experience E" (Mitchell et al., 1997). The machine learning way of pattern recognition approaches the problem as a three step process. Each step is accompanied by a subset of data. In the first step, known as the training phase, a learning algorithm trains a mathematical model $y(x)$ using a set of training data $\mathbf{x_{train}} := \{x_1, x_2, \cdots x_n\}$ (experience E). It must be noted, that in the majority of cases, the data lies in a multidimensional feature space. The training data $\mathbf{x_{train}}$ can be labelled or unlabelled according to the class of the problem. In the second phase, known as the validation phase, the model is validated on its performance P on the set $\mathbf{x_{valid}}$. The validation step is performed to estimate the best model parameters and properties. At the end of the training and validation phase, the best model $y(x)$ is applied on new unseen data sample x_{new} to recognise the pattern.

If one considers the pattern recognition programs as intelligent agents, based on the type of feedback from which it learns the patterns, the area of machine learning can be broadly divided into three main categories (Russell, 2003):

- Unsupervised learning: This class of learning tasks involve no explicit feedback from the environment. The most common unsupervised learning task is clustering, which aims to create distinct groups of data in multidimensional space, where there is significant homogeneity within a group and heterogeneity between groups.

- Supervised learning: The supervised learning task is to learn a mapping between input and output pairs. In this class of machine learning, the agent receives explicit feedback from the environment, which acts as the supervisor.

- Reinforcement learning: This type of learning is concerned with how an agent ought to take actions in an environment with an aim to maximise certain short and long term rewards. In this form of learning, the agent receives indirect weak feedback from the environment in the form of the reward (Sutton & Barto, 1998).

The focus in this research is on supervised and unsupervised learning paradigms. Traditionally, the supervised learning paradigm deals with static data, where a dataset is defined by $D := \{\mathbf{X}, \mathbf{y} | \mathbf{X} \in \mathbb{R}^{n \times m}\}$. The input data $\mathbf{X}$ is multisample (n) and multivariate (m) in nature. The output label is a vector $\mathbf{y}$. In a classification task, $\mathbf{y}$ is sampled from a nominal set $C := \{c_1, c_2, \cdots, c_k\}^n$ and in a regression task, $\mathbf{y}$ is continuous $\mathbb{R}^n$. A simple example of a classification task is that it is an intelligent agent being able to discriminate between a 'spam' and a 'non-spam' email given an email. In this task, the output $\mathbf{y} \in \{spam, no\ spam\}$ is nominal in nature. On the contrary, prediction of temperature from multivariate weather information is a good example of a regression/prediction task, where the output $\mathbf{y} \in \mathbb{R}$ is a real continuous number.

A brief review of the classical supervised ML, as presented in Kotsiantis, Zaharakis and Pintelas (2007), classified the research directions in the following categories:

1. **Logic based algorithms**: These type of algorithms are generally divided into the decision trees and rule based classifiers.

   Decision trees These trees perform instance classification by data partitioning based on feature values (Murthy, 1998). Figure 2.1 shows an example of decision tree. Each node in the tree represents a feature. A new sample is classified based on the corresponding feature value beginning from the root node. Numerous techniques are found in the

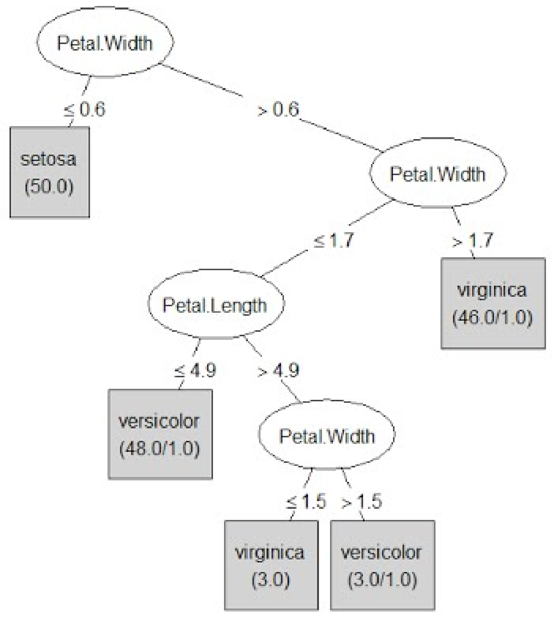
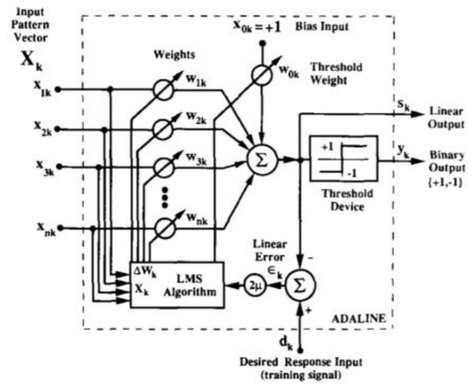


Figure 2.1.: A decision tree model built on the Iris dataset (Fisher, 1936) (source Girones (n.d.)).

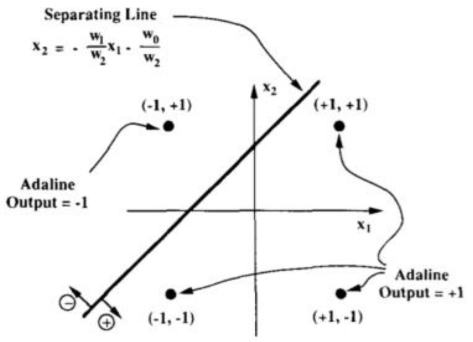
literature for data partitioning including information gain (Quinlan, 1986), gini index and others. The best known algorithms for creating decision trees are the C4.5 (Quinlan, 2014) and ID3 (Quinlan, 1986). The major stand out aspect of the decision tree is the representation of the model as a knowledge extraction system which makes it comprehensible for the user.

Rule induction Apart from decision trees, various algorithms for inducing rules from training data have been proposed in the literature. The rule induction algorithms aim to minimise the set of rules consisting of training data. Minimising the rule-set ensures generalisation and avoids overfitting (Kotsiantis et al., 2007). RIPPER (Cohen, 1995), AQ family (Michalski, 1980) and CN2 (Clark & Niblett, 1989) are some of the more popular rule based learning algorithms.

2. **Perceptron based**: Perceptron oriented algorithms are one of the most researched and powerful class of machine learning algorithms. These ML algorithms, as described in the seminal article (Rosenblatt, 1958), are inspired by the working principles of the brain, and can be categorised



(a) Adaptive Linear Elements (ADALINE)



(b) Example of a linearly separable problem.

Figure 2.2.: The ADALINE single layer perceptron model and an example of linearly separable problem (source Widrow and Lehr (1990)).

loosely as neuromorphic in nature. Historically, the perceptron based algorithms are divided into two sub categories:

Single layer perceptron A single layer perceptron model consists of a set of computational neurons (perceptrons) arranged in a single layer and fully connected to the input data $\mathbf{x}$ by connectors with weight $\mathbf{w}$. These networks were equipped to solve a binary classification problem ($\mathbf{y} := -1, 1$). The neurons compute weighed sum $\sum_i w_i \cdot x_i$ of the input data, and is passed through an adjustable threshold gate to output -1 or 1. In the ADALINE model the weighed sum is further passed through an activation function (Widrow & Lehr, 1990). Figure 2.2a shows the architecture of the ADALINE network. The most prevalent method of learning the patterns is by running multiple iterations of the training data and adjusting the connection weights $\mathbf{w}$ until the output matches ground truth. Some of the well known learning algorithms for single layer perceptron are described in (Littlestone & Warmuth, 1994; Freund & Schapire, 1999). Despite its computational ability to solve linearly separable binary output problems as shown in Figure 2.2b, these ML algorithms cannot solve problems of non-linearly separable variety, such as, the XOR problem (Minsky & Papert, 1969).

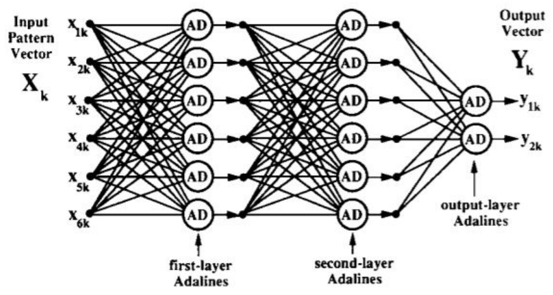


Figure 2.3.: A fully connected feed-forward multi layer perceptron model (source Widrow and Lehr (1990)).

Multi layer perceptron MLP models are extended from the earlier perceptrons. It is defined as a fully connected feed-forward ANN having multiple layers of nodes arranged as a directed graph. The layers are segregated into three classes: input units, hidden and output units. The hidden elements of MLP possess non-linear activation functions and can distinguish non-linearly separable data. Figure 2.3 shows an example of an MLP model. The feed-forward MLP networks are trained usually using some variant of the gradient based backpropagation algorithm (Rumelhart, Hinton & Williams, 1988), which optimise the output by minimising the network prediction error by backpropagation of the error. It must be noted that due to the gradient descent-based approach of the learning algorithm, these networks require the activation functions to be fully differentiable and the input data to be in a continuous space. Genetic algorithms have also been used to train the weights of neural networks (Siddique & Tokhi, 2001) and to find the architecture of neural networks (Yen & Lu, 2000). There are also Bayesian methods in existence which attempt to train neural networks (Vivarelli & Williams, 2001).

3. **Support vector machines**: SVM is one of the most recent and widely used supervised ML algorithm introduced by Cortes and Vapnik (1995). It was developed to solve binary classification problems using linear hyperplanes. SVM revolves around the idea of a margin that maximally separates two

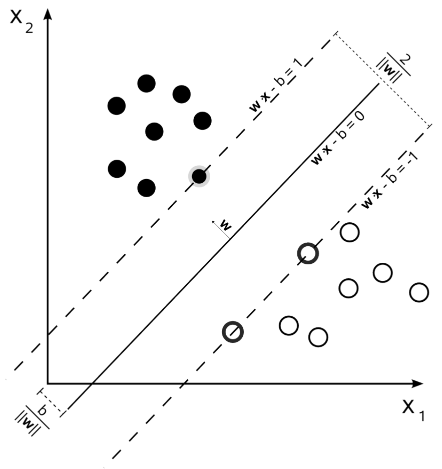


Figure 2.4.: Maximum-margin hyperplane and margins for an SVM trained with samples having two features from two classes (source Cortes and Vapnik (1995)).

data classes around a hyperplane (see Figure 2.4). For linearly separable training data, the tuple $(\mathbf{w}, b)$ exists such that:

$$\mathbf{w^T x_i} + b \geq 1 \; \forall x_i \in P \tag{2.1}$$

$$\mathbf{w^T x_i + b \leq -1 \; \forall x_i \in N} \tag{2.2}$$

The black and white dots represent the positive class P and negative class N, respectively. The decision rule is given by:

$$f_{w,b}(x) := sgn(\mathbf{w^T x_i} + b) \tag{2.3}$$

An optimum separating hyperplane can be found by minimising the squared norm of the separating hyperplane. The minimisation can be set up as a convex quadratic programming (QP) problem:

$$\begin{array}{ll} \min_{w,b} & \mathbf{\Phi(w) := \frac{1}{2}||w||^2} \\ \text{s.t.} & \mathbf{y_i(w^T x_i + b) \geq 1}, i = 1, \cdots, l \end{array} \tag{2.4}$$

For linearly separable data, at the end of the optimally separating hyperplane search, data points lying on its margin are known as support vector points. This hard margin solution is later modified (Veropoulos, Campbell, Cristianini et al., 1999) to accommodate misclassification of the training samples.

To overcome the issue of non-linearly separable problems, non-linear SVM is introduced by applying the kernel approach to find maximum margin hyperplanes. The kernel function transforms the given data space in a higher dimensional space in the hope of achieving linear hyperplane in a higher dimension that can separate the classes. There are numerous transformation kernels existing in the literature including:

- linear: $K(x_i, x_j) = x_i x_j$
- polynomial: $K(x_i, x_j) = (\gamma x_i x_j + c)^d$
- RBF: $K(x_i, x_j) = \exp(-\gamma |x_i x_j|^2)$
- sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i x_j + c)$

The kernel function represents the dot product of the input data points mapped into higher dimensional feature space. Some of the known disadvantages of SVM are the high computational complexity of the quadratic programming based training phase (Horvath, 2003) and sensitivity to overfitting the model selection criteria in kernel models (Cawley & Talbot, 2010).

The brief review presented in this chapter are very general and touches upon the two broad strands of research the studies in this thesis belongs to. Chapters 3 and 4 further reviews and delves into the nitty-gritties of computational neuron and network of such neurons as a basis of the main studies presented in Chapters 5, 6 and 7. Further, literature reviews are also performed locally in the context of the topics.

3 From the Human Brain to Artificial Evolving Spiking Neural Networks

3.1 Introduction

This Chapter introduces the biophysiology of a human brain cell known as a neuron and proceeds through the historical evolution of artificial neurons and network of such neurons as computational models. Further, a brief review and the work on the evolving connectionist systems (ECOS) will be presented. This is an evolving neural network framework developed for recognising patterns from data.

3.2 The Human Brain and the Morphology of a Neuron

The human brain is the central processing unit of the human nervous system. It is responsible for processing, integrating, and coordinating the information it receives from the sensory organs, and making decisions as to the instructions sent to the rest of the body. The human brain is made up of a network of approximately ten billion interconnected neurons. A neuron is the basic information processing unit which receives, processes and transmits information using biochemical reactions. A typical neuron, as shown in Figure 3.1, comprises three functional units, namely, dendrites, soma and axon. Dendrites are tree-like structural fibres which emanate from the cell body and provide the receptive zones for receiving the signals from other neurons. The morphology of the dendritic

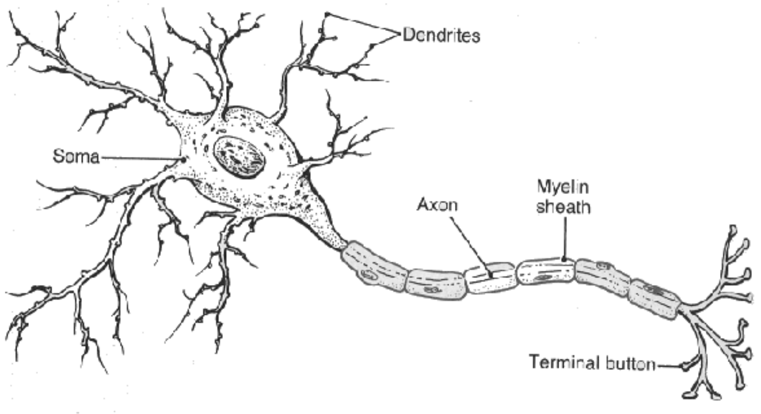


Figure 3.1.: Morphology of a biological neuron (source Carlson (1967)).

tree plays an important role in the integration of the synaptic inputs and influences the way the neuron processes information (Mel, 1994). The integration process is performed through a non-linear process of spatio-temporal summation (Koch, 1999). The resulting activation flows to the soma and suffers voltage attenuation, so that about half of the total charge injected into the distal dendritic site reaches the soma. The primary function of the soma is to perform the continuous maintenance required to keep the neuron functional (Kandel, Schwartz, Jessell, Siegelbaum, Hudspeth et al., 2000). The part of the soma that performs the important nonlinear processing of the input is known as axon hillock. If the total input produces a depolarisation up to the neural threshold, the axon hillock fires an action potential. The output signal is transmitted down the axon, which delivers it to other neurons. Some of the important characteristics of of neuron are enumerated below:

- Synapse: The contact point between a pre-synaptic axon and a post-synaptic dendrite or soma is known as a synapse. In a chemical synapse, arrival of action potential triggers a complex chain of biochemical reactions leading to the release of neurotransmitters. This process generates a post-synaptic potential (PSP) in the post-synaptic neuron. The internal state of a post-synaptic neuron is driven by complex integration of PSP arising from multiple such synapses. When the state of a post-synaptic neuron is sufficiently stimulated, it gives rise to an action potential or a spike which is then transmitted through the axon. Synapses are believed to be the locus of learning and memory in the brain (Squire & Kandel, 1999). This is where the brain is the most flexible and the most vulnerable (Marian, 2002).

- Action potential and neuron dynamics: The output generated by a biological neuron is referred to as the action potential. The action potential is an all or none electrical voltage based impulse response (Aur & Jog, 2010) also known as a spike. A neuronal action potential is generated when the membrane potential of the neuron reaches a threshold limit. Figure 3.2 plots a typical example of membrane potential (in mv) with time (in ms). The membrane potential starts with a resting potential of $-70mV$. On application of a stimulus at $t = 1ms$, the membrane potential is raised above $-55mV$ which is the threshold limit. After the stimulus is applied, the membrane potential rises to $+40$ mV. This phase is known as the depolarisation phase. The potential then drops down to $-90mV$ at time = $3ms$ (the repolarisation phase), and finally rises to the resting potential of $-70mV$ is at time = $5ms$. The period between repolarisation and resting state is known as the refractory period during which the neuron goes into a hyperpolarisation state. During the hyperpolarised state, the neuron is incapable of firing any action potential. This mechanism stops a neuron firing spikes immediately after firing a spike.

3.2.1 Neural Encoding and Information Communication

Neurons are thought to convey signals mainly if not exclusively through the information content of their spike-trains. A spike-train consists of a series of times at which the neuron has fired. It is possible to record spike trains from individual neurons using various electro-physiological methods in-vivo and in-vitro. Such methods have generated a good number of datasets, which, in turn, have revealed many properties of the neural computation. These properties constitute the main body of results in the rapidly growing neuroscience literature.

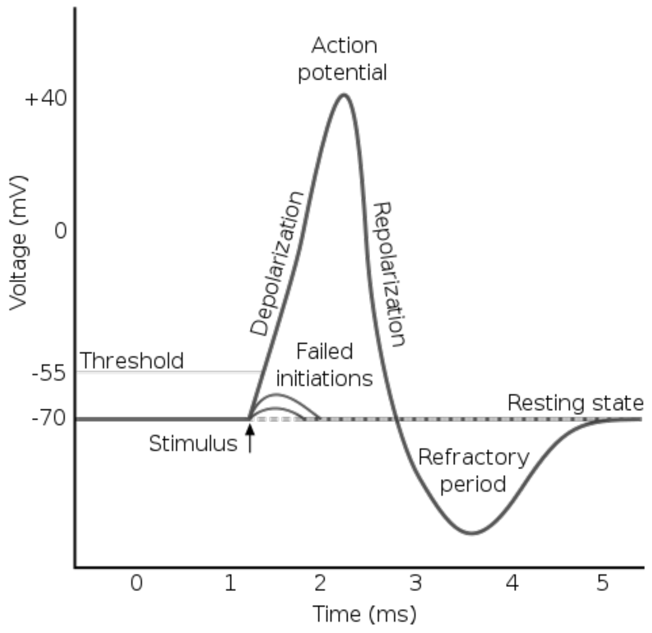


Figure 3.2.: Dynamics of the membrane potential of a typical neuron (source "Afterhyperpolarization" (2007)).

Neural encoding refers to the study of how neurons perceive and process information received from other neurons or through incoming stimuli. As mentioned earlier, information in a biological neuron is represented by action potentials. Even though minor variations exist in the morphology of an action potential, they are generally treated as identical stereotyped events in time. The study of neural encoding and decoding has taken massive leaps over the past decade with the availability of sophisticated invasive and non-invasive recording mechanisms from the brain cells. The curiosity regarding the nature of the neural code has been around from the beginning of neuroscience and till now, there has been no single answer towards it. Several large scale projects, such as the brain decoding project, (Tsien et al., 2013) have emerged due to progress in this area. The scientific fraternity has presented evidence over the years pointing to the existence of two major categories of neural encoding schemes:

1. Rate coding: The most classical view of the neural encoding is the rate coding and is originally shown in Adrian (1926) through experiments conducted by hanging different weights from a muscle. They found the increase of stimulus weight increases the number of spikes recorded from the nerve. The frequency based approach of rate coding states that the fir-

ing rate of the neuron changes monotonically with the intensity or salience of the input. This also means that the randomness in the inter spike interval (ISI) in a spike train is considered as noise. This coding hypothesis can be explained by Poisson's model of random process (Pachitariu, Brody, Jun & Holmes, n.d.). Out of the abundant examples of this coding scheme, the experiments conducted by (Hubel & Wiesel, 1959, 1962) in their ground breaking articles are the most interesting. During the experiments, light stimuli was presented to a cat and the action potentials of the primary visual cortex receptive fields were recorded over time. Figure 3.3 shows how different orientation of light bars generates response of varying spike rates in the receptive fields of cat eyes. Such observations formed the basis of the rate coding scheme.

Even today, neuro-scientists often assume that useful theories can be learned about neural coding by summarising the post-stimulus time histogram (PSTH) that plots the rate of firing as a function of time. There is no doubt that rate coding seems to be the most obvious explanation of how neurons encode information. However, it is also quite evident that the firing rate as a coding scheme is highly inefficient for information processing (S. Thorpe, Delorme & Van Rullen, 2001; Gautrais & Thorpe, 1998; Van Rullen & Thorpe, 2001). S. J. Thorpe and Imbert (1989) argued that the primate visual cortex brain cells need to operate in less than $10\ ms$ resolution. On the other hand, the upper bound of the firing rate is $\approx 100\ Hz$, which implies that such processing can be accomplished if the individual neurons only get to fire none or one spike. The inefficiency in rate coding is caused by the requirement of precise calculation of the firing rate. Gautrais and Thorpe (1998) showed within a mathematical framework that during the first $10\ ms$ of computation, n neurons can transmit $\log_2(n+1)$ bits of information in a rate coding scheme and $log_2(n!)$ bits of information with the alternate temporal coding scheme. The necessity of approximating firing rate from a large volume of spikes is a significant hindrance in the acceptance of this theory, especially considering the extreme efficiency of the human brain. Additionally, the refractory period of the neuron dynam-

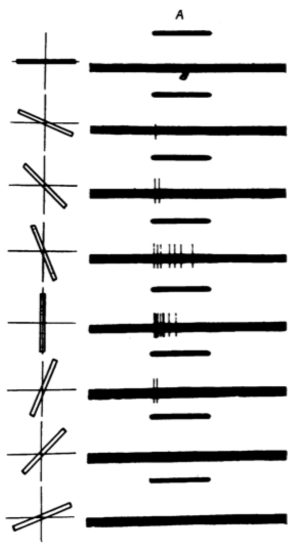


Figure 3.3.: Experimental results obtained by Hubel and Wiesel (1962). The orientation of light bars presented at various orientations are shown on the left and the spike responses are shown on the right (source Hubel and Wiesel (1959)).

ics also prevents neurons from generating large numbers of spiking events within a short period. Olshausen and Field (2006) empirically demonstrated from the electro-physiological data that rate coding V1 consumes ten times more energy to operate.

2. Temporal coding: Temporal encoding hypothesizes that the information is encoded in the relative timings of the spikes. This is in complete contrast to the rate coding scheme where irregularities in ISI do not arise from stochastic forces and thus are not a random process. Temporal codes employ the features of the spiking activity that cannot be described by the firing rate. For example, the time to the first spike after the stimulus onset, characteristics based on the second and higher statistical moments of the ISI probability distribution, spike randomness, or precisely timed groups of spikes (temporal patterns) are candidates of temporal coding schemes (Kostal, Lansky & Rospars, 2007). Several studies have also suggested the existence of the temporal coding scheme across the animal kingdom. For fast encoding of visual stimuli in the retina cells, latency time between the stimulus onset and time to first spike is used for encoding (Gollisch & Meister, 2008). This is also known as the rank order coding (Van Rullen & Thorpe, 2001). As with the visual system, in mitral/tufted cells in the

olfactory bulb of mice, first-spike latency relative to the start of a sniffing action seem to encode much of the information about an odour. This strategy of using spike latency allows for rapid identification of and reaction to an odourant. In addition, some mitral/tufted cells have specific firing patterns for given odourants. This type of extra information could help in recognising a certain odour, but is not completely necessary, as the average spike count over the course of the animal's sniffing was also a good identifier (R. I. Wilson, 2008). Along the same lines, experiments done with the olfactory system of rabbits showed distinct patterns which correlated with different subsets of odourants, and a similar result was obtained in experiments with the locust olfactory system (Theunissen & Miller, 1995).

3.3 Generations of Artificial Neuron

It is a common understanding that primate intelligence has formed the base of artificial intelligence. The idea of mimicking or translating human intelligence into machines has been a significant ambition for the scientific community in the past centuries. The evolution of artificial neural networks has massively contributed towards realisation of that ambition. ANN is a mathematical realisation of a network of neurons or processing units that can perform complex functional mappings. Although ANNs have progressed through various stages of evolution, attempts were made only very recently to classify them into generations of neural network (Maass, 1997). Due to the multiple branching of ANN research and development, this has been a challenging task. Additionally, such categorisation is subjective and dependent on what is considered as achievement. However, one such identifiable conceptual progress has been the development of mathematically-defined activation function as the information processing mechanism of an artificial neuron (Maass, 1997).

An artificial neuron has been conceived as a mathematical model of a biological neuron. Figure 3.4 shows a block diagram of the components of an artificial neuron. Starting from the left, an ANN consists of multiple input channels. The

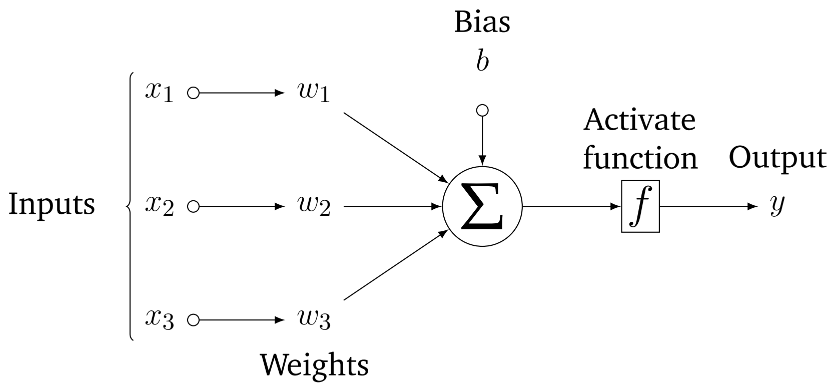


Figure 3.4.: Block diagram showing components of an artificial neuron.

input channels resemble the properties of dendrites in a biological neuron. These channels feed input signals (originating from pre-synaptic neurons) $\{x_1, x_2, \cdots\}$ to the processing unit of the neuron. The synaptic weights $\{w_1, w_2, \cdots\}$ define the strengths of the synapses and allows for strengthening or weakening of synapses through synaptic plasticity. The input along with the synaptic strengths are fed into the processing units which consists of the summation and activation units. These units together are responsible for mapping of the inputs x to the output y. Next, The evolution and properties of an artificial neuron is well summarised by Ghosh-Dastidar and Adeli (2009), however, for the sake of continuity, I present below, a brief overview of the three generations of computational neuron:

3.3.1 First Generation Neurons:

The perceptron model described by Rosenblatt (1958) was the beginning of the first generation of artificial neuron. The variations of this model used the perceptrons as integrate and fire units, which integrate the inputs and fired if the internal state (synaptic weighed sum of inputs) reached a threshold. Mathematically, the activation can be described by a Heavyside step function (range $\{0, 1\}$).

$$f(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

Equation 3.1 describes the activation of a simple perceptron. It produces binary output $y = 1$ or $y = 0$, if the weighed sum $\mathbf{w} \cdot \mathbf{x}$ goes beyond b or otherwise.

Unlike the biological neuron, the inputs of the perceptron were real and continuous, *i.e.* the magnitude of inputs contribute to the activation of the neuron. This is a primitive realisation of the rate coding scheme, *i.e.* higher input causes the neuron to fire. However, the outputs were binary spikes and did not cater to the rate coding scheme. The perceptrons were also time agnostic. The inputs are always considered to be synchronous in time and it did not matter when the threshold is reached, *i.e.*, as they contribute to the internal state at the same time, and hence can be directly integrated. Any events of the past do not affect the activation at the present time.

3.3.2 Second Generation Neurons:

The second generation of neurons were developed in the 1960s as an extension of the first generation neurons. In the second generation, non linear smooth activation functions were introduced. Hence, instead of a fixed threshold value for the output determination, the outputs were proportionate within range to input signals. Tanh (Equation 3.3 range $(-1, 1)$) or Sigmoid (Equation 3.2 range $(0, 1)$) functions were used most often as activation functions. With this development, the outputs became real and continuous, and contrary to the first generation, the post-synaptic neurons could generate rate coded information. These neurons became extremely popular in the AI community with the introduction of feed-forward neural networks and back-propagation (BP) algorithm (Rumelhart et al., 1988), which enabled supervised learning. Since the BP algorithm was constrained by its requirement of a continuous and differentiable activation function, a significant portion of the ensuing research became focused on finding more appropriate continuous and differentiable activation functions. This model was significantly more powerful than the one based on first generation neurons and could solve complex pattern recognition problems (the most notable early example was the XOR problem). However, the computational power of the neuron still did not reach its full potential because the temporal information about individual spikes was not represented.

$$f(\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}\cdot\mathbf{x}+\mathbf{b})}} \tag{3.2}$$

$$f(\mathbf{x}) = \tanh(\mathbf{w} \cdot \mathbf{x} + \mathbf{b}) \tag{3.3}$$

3.3.3 Third Generation Neurons

In the past decade, to overcome the shortcomings of the first and second generation neurons, neurons that can communicate via the precise timing of spikes or a sequence of spikes have been developed and adapted for ANNs. These neurons are known as spiking neurons. In the literature, these spiking neurons have been referred to as third generation neurons (Maass & Bishop, 2001). Similar to the first generation neurons, spiking neurons act as integrate-and-fire units and have an all or none response. The spiking neuron, however, has an inherent dynamic nature characterised by an internal state which changes with time. Each post-synaptic neuron fires an action potential or spike at the time instance that its internal state exceeds the neuron threshold. Similar to biological neurons, the magnitude of the spikes (input or output) contains no information. Rather, all information is encoded in the timing of the spikes.

3.4 Spiking Neuron Models

The first and second generation neurons discussed in the previous Section use rather simplistic activation of the artificial neurons. The biophysical neuron models, however, are developed to quantitatively characterise neuronal behaviour based on neuron membrane potential and ion channel conductance models (Ghosh-Dastidar & Adeli, 2009). Marian (2002) distinguished between the two spiking neuron models by the extent of detail included in the neuron model.

1. Detailed neuron models: In this approach, researchers aimed at creating detailed and complex models of the neurons. De Schutter and Bower (1994), Segev, Burke and Hines (1998) described the chemical process at the sub-cellular level. This included ion channel and dendritic tree biophysics, synaptic interplay between excitation and inhibition, and voltage dependent events in the active dendrites (Mainen & Sejnowski, 1998). Among all the detailed neuron models, the most prominent is the conductance based Hodgkin-Huxley model (Hodgkin & Huxley, 1952). The Hodgkin-Huxley model describes the generation of action potentials at the level of ion channels and ion current flow. It is the starting point for detailed biophysical neuron models which in general include more than the three types of currents considered by Hodgkin and Huxley. Electrophysiologists have described an overwhelming richness of different ion channels. The set of ion channels is different from one neuron to the next. The precise channel configuration in each individual neuron determines a good deal of its overall electrical properties. The model provides a detailed description of the biophysics of ionic mechanisms underlying the initiation and propagation of the neural spike. By doing this, it offers an accurate quantitative model of the physiological data. But, complex frameworks, such as this, which account for numerous ions-channels and different types of synapses are difficult to construct and analyse. An important conceptual drawback of this family of models is that their numerical complexity (e.g., solve a large number of non-linear differential equations) can prevent one from understanding which features are responsible for a particular phenomenon and which are irrelevant (Koch, 1999).

   The Hodgkin-Huxley like models are useful in modelling point neurons. For the purpose of building detailed neuron models that can take into consideration extreme cell complexities, such as branched cable structure, the standard way is to divide a neuron into a fixed number of compartments where each compartment acts like a resistance-capacitance (RC) circuit. The RC circuits are modelled by a system of differential equations. Numerous biological neuron simulators, such as Genesis (M. A. Wilson, Bhalla, Uhley & Bower, 1989) and Neuron (M. L. Hines & Carnevale, 2006), implemented the compartmental models. Nonetheless, there is always a trade-off between computational cost and biological realism.

2. Formal spiking neuron models: The second direction focuses on the spiking nature of the neurons and retaining the essential elements of the behaviour being modelled, while trying to simplify the complexity of the resulting description. The main motivation for the creation of simplified models is that they allow studying more easily the computational and functional principles of neural systems.

   The reduction of the detailed neuron models to formal models requires simplifications in at least two respects. First, the nonlinear dynamics of spike generation must be reduced to a single ordinary differential equation and second, the spatial structure of the neuron (*i.e.*, the dendritic tree) is neglected and reduced to an input (Gerstner & Kistler, 2002). To support the validity of the former simplification, Kistler, Gerstner and van Hemmen (1997) demonstrated that spike generation in the Hodgkin-Huxley model can be reproduced to a high degree of accuracy (up to 90%) by a single variable model. Several simplified neural models have been proposed in the last decades. The leaky-integrate-and-fire (LIF) neuron is probably the best-known example of a formal neural model. It simulates the dynamics of the neuron membrane potential in response to a synaptic current by implementing an equivalent electrical circuit. The function of the integrate-and-fire circuit is to accumulate the input currents, and, when the membrane potential reaches the threshold value, to generate a spike. Immediately after emitting a pulse, the potential is reset and maintained there for an absolute refractory period.

   The simplified mathematical models for spiking neurons cannot account for the entire range of computational functions of the biological neuron. Rather, they try to abstract a number of essential computational aspects of the real cell function. The essential features implemented can differ between models, as a function of what the modeller considers to be relevant and crucial for its domain study. Thus, the integrate-and-fire model focuses upon the temporal summation function of the neuron (Bugmann, Christodoulou & Taylor, 1997). The spike response model proposed by Gerstner (1998) simplifies the action potential generation to a threshold process. The resonate-and-fire model (Izhikevich, 2001) focuses upon the operation of the neuron in a resonating regime. By contrast with the detailed neural models, the computational strength of the spiking neurons arises from the way they interact with each other, when they work cooperatively in large networks.

Next, I will outline some of the most prominent spiking neural network models present in the literature:

3.4.1 Integrate and Fire

The description of the Integrate and Fire (IF) and Leaky Integrate and Fire (LIF) model is adapted from (Gerstner, Kistler, Naud & Paninski, 2014) which serves as a comprehensive review on this topic. An IF model is described by a single variable known as membrane potential. The effect of an incoming spike on a post-synaptic neuron can be recorded using intra-cellular electrodes which measure the potential difference $v(t)$ at time t between the inner and outer wall of a cell membrane. This is known as the membrane potential. In absence of any input, the neuron is said to be at rest with a constant resting membrane potential v_{rest}. The arrival of a spike changes the membrane potential and finally decays back to v_{rest}. For excitatory synapses the change is positive and for inhibitory, it is negative. The neuronal dynamics of an IF neuron can be conceived as an integration process combined with a mechanism that triggers action potential above a threshold voltage V_{thr}. From hereon, post-synaptic and pre-synaptic neurons will be indexed by symbols i and j respectively. A simple IF model can be described using: (i) A linear differential equation to describe the evolution of the membrane potential $v_i(t)$. (ii) A threshold for spike firing.

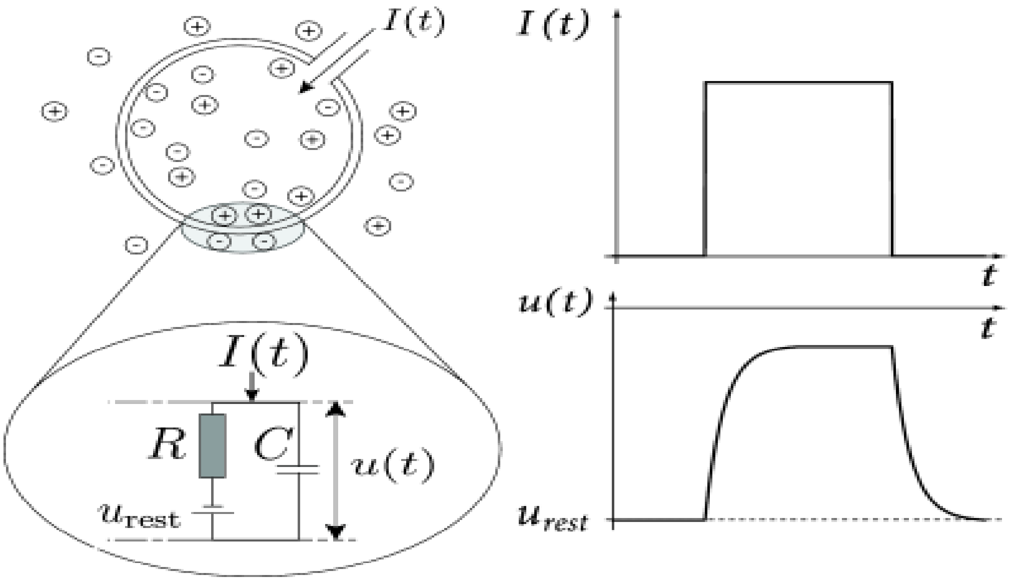


Figure 3.5.: Electrical properties of the passive membrane of an IF neuron. (A) A cell membrane enclosed neuron receives a positive input current $I(t)$ resulting in increase of electrical charge inside the cell. The cell membrane acts as a capacitor in parallel with a resistor which is in line with a battery of potential v_{rest}. (B) The reaction of the cell membrane to a step current (top) with a smooth voltage trace (source Gerstner, Kistler, Naud and Paninski (2014)).

Gerstner et al. (2014) modelled an IF neuron as a single RC circuit (see Figure 3.5) following the laws of electricity. The cell membrane (insulation to the cell body) acts as a capacity of a capacitor C. Also, because the insulation is imperfect, the charge over time, slowly leaks through the cell membrane. The cell membrane hence can be characterised by a finite leak resistance R. Due to the leaky nature of the resistance, the model is also known as leaky integrate-and-fire (LIF) model.

$$I(t) = I_R + I_C \tag{3.4}$$

$$I(t) = \frac{v(t) - v_{rest}}{R} + C\frac{dv}{dt} \tag{3.5}$$

$$\tau_m \frac{dv}{dt} = RI(t) - [v(t) - v_{rest}] \tag{3.6}$$

$$v(t) = v_{rest} + RI_0[1 - \exp(-\frac{t}{\tau_m})] \tag{3.7}$$

The circuit shown in Figure 3.5 can be analysed from the law of current conservation as per Equation 3.4. This can be further rewritten as Equation 3.5 by calculating $I_R = v_R/R$ as per Ohm's law, where $v_R = v - v_{rest}$. The I_C charges capacitor C. The capacitive current can be written as $I_C = dq/dt = Cdv/dt$. Eq. Equation 3.6 is the linear differential equation describing a LIF neuron's passive membrane. The trajectory of membrane potential (Equation 3.7) for a constant input current $I(t) = I_0$ starting and ending at $t = 0$ and $t = \Delta$ can be derived by solving the differential Equation 3.6. The LIF neuron emits a spike denoted as $t_i^{(f)}$ generates a spike when the membrane potential $v(t)$ reaches threshold v_{thr}, *i.e.*,

$$t^{(f)} : v(t^{(f)}) = v_{thr} \tag{3.8}$$

3.4.2 Hodgkin-Huxley

One of the earliest proposed (1952) models for spiking neuron is the Hodgkin-Huxley (HH) model. This model describes the influence of ion channel conductance on the spike responses of axon and was empirically studied on the axons of a Giant Squid. The choice of squid was due to its non-microscopic size. This large size was necessary as electrodes had to be inserted into the axon, in order to record the changes in electrical state experienced when neurons are active.

The HH model is described by a RC circuit connected in a parallel scheme resembling the IF model. Following Equation 3.4, the HH model rewrites it as:

$$C\frac{dv}{dt} = I(t) - I_R \tag{3.9}$$

HH model describes the resistive current as the sum of three different ion currents due to the presence of movement of sodium, potassium and leakage. The resistive current is described by Equation 3.10.

$$I_R = G_{Na}m^3h(u - V_{Na}) + G_K n^4(u - V_K) + G_{Leak}(u - V_{Leak}) \tag{3.10}$$

Where V_{Na}, V_K and V_{Leak} are known as reversible potential constants. G_{Na} and G_k are the maximum conductance of the sodium and potassium channels, while the voltage independent leak channel is represented by G_{Leak}. The variables m, n and h are gating variables described by Equations 3.11, 3.12 and 3.13.

$$\frac{m}{dt} = \alpha_m(u)(1 - m) - \beta_m(u)m \tag{3.11}$$

$$\frac{n}{dt} = \alpha_n(u)(1 - n) - \beta_n(u)n \tag{3.12}$$

$$\frac{h}{dt} = \alpha_h(u)(1 - h) - \beta_h(u)h \tag{3.13}$$

where m and h control the sodium channel and n controls the potassium channels. The function $\alpha_x(.)$ and $\beta_x(.)$ represent empirically determined voltage dynamics across capacitor v, are adjusted to simulate different neuron types.

3.4.3 Izhikevich

The Izhikevich model claims to combine biological plausibility of the HH model with the lower computational complexity of the IF and LIF models (Izhikevich, 2004). Based on the theory of dynamic systems, the dynamics of this model are governed by two equations:

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I \tag{3.14}$$

where u is a membrane recovery variable providing negative feedback for v; variables u and v, and parameters a, b, c and d are dimensionless. The input stimuli in the form of input current is represented by I. When the membrane potential reaches the (fixed) threshold $v_{thr} = 30mV$, the neuron spikes and resets u and v as:

$$if v \geq 30mV \begin{cases} v \leftarrow c, \\ u \leftarrow u + d, \end{cases} \tag{3.15}$$

Dependent on parameters a (decay rate of membrane potential), b (sensitivity of membrane recovery), c and d (reset values of v and u respectively), a huge variety of neuronal types can be modelled with relative ease.

It can be observed that the mathematical representation of the biological neurons are extremely complex and focus on mimicking the biological properties accurately. This direction of work focuses on discoveries of biophysical properties of neurons and the human brain through simulations. The work of this thesis is, however, not intended to follow that direction. This work focuses on using the spiking properties of the biological neurons, interwoven together in a network for solving pattern recognition problems. Therefore, it is necessary to compare the capabilities of the spiking neuron models in regards to the biological plausibility and complexity vs computational cost. Izhikevich (2004) presented an insightful comparison of over twenty spiking neuron models in this regard. Izhikevich, in this article, compared the presence or absence of 22 categories of biological properties, such as tonic spiking, phasic spiking, spike latency and chaos in the spiking neuron models. Figure 3.6 shows the comparison of spiking neuron models presented in (Izhikevich, 2004). The biological plausibility on the Y axis is measured by the total number of biological properties (out of 22 different categories) present in the neuron model. The X axis plots the implementation cost of the neuron model by the number of floating

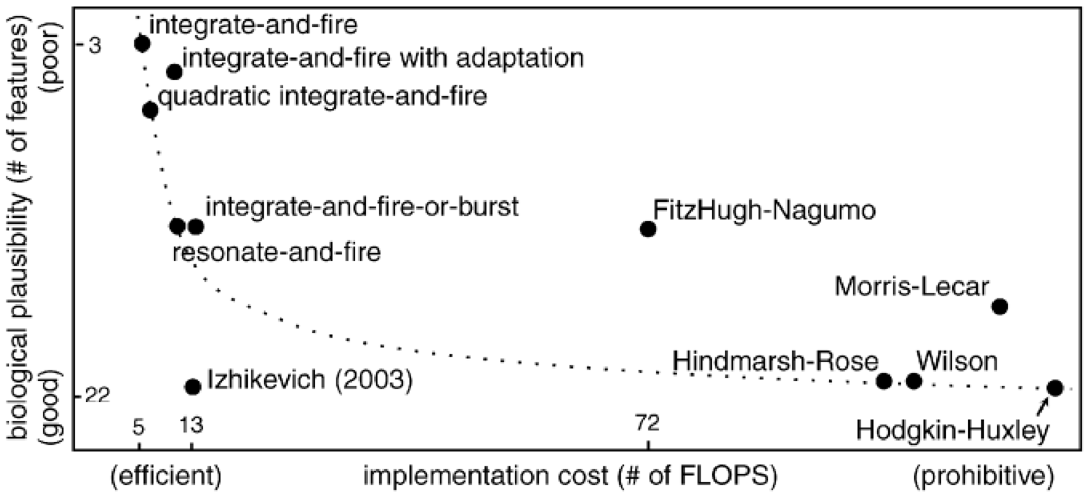


Figure 3.6.: Comparison of spiking neuron models in the evaluation landscape of biological plausibility and implementation cost (source Izhikevich (2004)).

point operations (FLOPS). It can be observed that out of many models proposed in the literature, Hodgkin-Huxley and integrate-and-fire neurons, reside at the extremity of the evaluation landscape. The choice of neuron models, as argued in (Izhikevich, 2004) really depends on the goal. If the goal is to simulate a large number of neurons interacting within a network, efficiency plays an important role. Variations of IF models are suitable for this purpose. For the rest of this thesis, variations of the IF model are used and discussed.

3.5 Spiking Neural Networks

The phenomenological models of spiking neurons described earlier can be simplified to lesser realistic models where spikes are not modelled which leads to the first two generations of the neurons. Of course, the computational load reduces drastically along with biological plausibility. For decades, the modelling of the neurons was limited by the available computing power because the hardware was unable to support large ANNs based on detailed neuronal models. This limitation dictated the design of the learning algorithms. Subsequently, even when advances were made in computing power, proportionate advances were not made in the complexity of the neuronal models because the existing learning algorithms were not compatible with the detailed models.

Accordingly, two distinct research areas emerged. The field of Artificial Neural Networks concentrated on the behaviour of large networks of neuron-like processing units (*i.e.*, the second generation neurons), which were primitive and oversimplified formulations of biological neurons. However, it was demonstrated that even such networks were capable of learning using pseudo-realistic learning algorithms, such as backpropagation. ANNs were applied with great success to pattern recognition, classification, and pattern completion tasks in a wide variety of areas. The other field became known as Computational Neuroscience. Within this broad interdisciplinary field, the detailed biophysical and phenomenological models were primarily used in relatively smaller networks to study electro-physiological processes, pattern generation, and the dynamic behaviour of small groups of neurons. There have also been studies involving very large numbers of interconnected biophysical neuron models. However, it has not been possible to use such networks of detailed neurons in a manner similar to ANNs for large real-world pattern recognition and classification tasks.

Modern advances and the accessibility of computing resources have increased the overlap between the two fields. On one hand, the processing units, networks, and learning algorithms for ANNs have become biologically more realistic. On the other hand, networks of biophysical neurons have become increasingly larger in size and the biophysical models, more detailed. The available computing power still limits the use of the detailed models in large biophysical neural networks for pattern recognition and classification tasks. As the computing power becomes more readily available, suitable learning algorithms are also being developed for such models. The development of Spiking Neural Networks (SNN) was the next logical step towards achieving this goal.

Simply stated, SNNs are networks of spiking neurons. The SNN is architecturally similar to that of a traditional ANN. The processing units, however, are spiking neurons, typically modelled by a phenomenological model, such as a Spike Response Model. As discussed earlier, the use of the biophysical models in certain applications of SNNs is less common due to the computational burden.

3.6 Neuromorphic Computing Beyond von Neumann Hardware Architecture

This Section is an adaptation of the article (N. Kasabov, Sengupta & Scott, 2016) published in 2016. The tremendous push of AI towards emulation of real intelligence has been sustained by the realisation of Moore's law (Schaller, 1997), which states that the processing power of central processing units (CPU) will double every couple of years. The scalable computer architecture proposed by John von Neumann in 1945 as part of the draft of EDVAC computer (Randell, 2013) had to play a substantial role in accomplishing the continuous miniaturisation of the CPU chips. In more recent years, CPU chip manufacturing companies have spent billions of dollars in CMOS technology to shrink the transistor size to a minuscule (≈ 14 nanometres) and thus keep Moore's law alive. It is evident that this is non-sustainable and as per well-supported predictions will reach its boundary in the next three to five years (Toumey, 2016). The literature suggests that the ongoing effort to enhance the traditional von Neumann architectures are unlikely to lower simulation runtimes significantly, as single and multi-core architectures are reaching a state of saturation in terms of transistor size (Thompson & Parthasarathy, 2006), energy consumption (Esmaeilzadeh, Blem, St Amant, Sankaralingam & Burger, 2011) and communication (Perrin, 2011).

The Von Neumann architecture is a multi-modular design based on rigid physically separate functional units. It specifically consists of three different entities:

- Processing unit: The processing unit can be broken down into several subunits, the arithmetic and logical unit (ALU), the processing control unit and the program counter. The ALU computes the arithmetic logic needed to run programs. The control unit is used to control the flow of data through the processor.

Table 3.1.: A comparison of the key contrasts between von Neumann and neuromorphic computing paradigm.

Properties	**von Neumann**	**Neuromorphic**
Representation of the data	Sequence of binary numbers	Spike (event) timings
Memory	Volatile and non-volatile	Long and short term memory
Plasticity (Learning)	No	Long, short term potentiation and depression
Processing	Deterministic, centralised and sequential	Stochastic, decentralised and parallel

- I/O unit: The i/o unit essentially encompasses all I/O the computer could possibly do (printing to a monitor, to paper, inputs from a mouse or keyboard, and others.).

- Storage unit: The storage unit stores anything the computer would need to store and retrieve. This includes both volatile and non volatile memory.

These units are connected over different buses like data bus, address bus and control bus. The bus allows for the communication between the various logical units. Though very robust, this architecture inherently suffers from the bottleneck created due to the constant shuffling of the data between the memory unit and the central processing unit. This bottleneck leads to rigidity in the architecture as the data needs to pass through the bottleneck in a sequential order. An alternate solution of parallelising the computers has been proposed where millions of processors are interconnected. Despite the increase in processing power, this solution is still limited by the bottleneck in its core elements (Schuler, 2015).

The saturation in the scalability of the von Neumann architecture led to developments in computer and computing architectures. Neuromorphic computing was coined by Carver Mead (Mead, 1990) in the 1980s and further developed in a new paradigm of computing. As the name 'neuromorphic' suggests, this is inspired by the human brain. Moreover, as the existence of AI is complimented by computing architectures, having a real neuromorphic computer architecture oriented processing unit is a step towards the development of highly neuromorphic AI. The focus of neuromorphic hardware systems are to model the behaviour of the biological neurons in digital or analog circuits. It also draws great inspiration from our brain's ability to manage tens of billions of processing units connected by the hundreds of trillions of synapses using tens of watts of power

Table 3.2.: Comparison of the key features of the popular neuromorphic systems and human brain. The details are adapted from (Scott, 2015; S. Furber, 2016).

Feature	Human brain	SpiNNaker (S. B. Furber, Galluppi, Temple & Plana, 2014)	Zheijang FPGA (J.-l. Li, Wang, Li, Han & Zhang, 2005)	TrueNorth (Hsu, 2014)
Type	Biological	Programmable digital	FPGA	Fixed Digital
Neuron model	Diverse	Programmable	LIF	LIF
Synapse model	Diverse	Programmable	Programmable	Binary
Max # neurons	$100B$	$16K$	2048	$1M$
Max # synapses	10^{15}	$14M$	$4.2M$	$256M$
Runtime plasticity	yes	Programmable	No	No
Energy per connection	10fJ	10nJ	Unknown	25pJ
Biological speed up	$1x$	$1x$	$1x$	$1x$

on average. The large network of the processing units (neurons) in the brain are in a true sense a mesh. The data is transmitted over the network via the mesh of synapses seamlessly. Architecturally, the presence of the memory and the processing unit as a single abstraction is uniquely advantageous leading to dynamic, self-programmable behaviour in complex environments (Schuler, 2015). The highly stochastic nature of computation in our brain is a very significant divergence from the bit-precise processing of the traditional CPU. The neuromorphic computing hence aspires to move away from the bit-precise computing paradigm towards the probabilistic models of simple, reliable, power and data efficient computing (Calimera, Macii & Poncino, 2013) by implementing neuromorphic principles, such as spiking, plasticity, dynamic learning and adaptability. This architecture morphs the biological neurons, where the memory and the processing units are present as part of the cell body leading to the decentralised presence of memory and processing power over the network. There is a significant interest in such hardware implementations of SNN due to many factors. The primary factor being the massive power efficiency -somewhere in the order of 20 watts- for its ability to learn and operate in a stochastic environment. Such systems can be broadly categorised in one of the families of Application-Specific Integrated Circuit (ASIC), Field Programmable Gate Array (FPGA), or digital systems. Table 3.1 lists some fundamental characteristics of the von Neumann and neuromorphic architecture.

A number of large-scale neuromorphic systems have emerged over recent years taking advantage of the enormous transistor resources now available on a single microchip. The enhanced capabilities of the neuromorphic systems enable modellers to contemplate building models of complete brains of animals from insects up to smaller mammals, or substantial sub-areas of the human brain, and the same systems also offer platforms capable of supporting new scales of cognitive architecture (S. Furber, 2016). A tabular comparison of the neuromorphic hardware platforms are adapted from (Scott, 2015) and shown in Table 3.2.

SpiNNaker: The SpiNNaker hardware, developed as part of the Human Brain Project is a massively parallel digital computer whose communication infrastructure is motivated by the the objective of modelling massively scalable SNN with the brain like connectivity profile in biological real time. In many respects, SpiNNaker resembles conventional supercomputers, with the following differences:

- The processors in SpiNNaker are mobile IC chips.
- The communication protocol in SpiNNaker is brain inspired and is optimised for broadcasting large quantities of small data packets to destinations in a stochastic manner.

The SpiNNaker system is designed around a plastic ball grid array package which incorporates a custom processing chip and a 128 Mbyte SDRAM memory chip. The processing chip contains 18 ARM968 processing cores, each with 23 Kbytes of instruction memory and 64 Kbytes of data memory, a multicast packet router and sundry support components (S. Furber, 2016). The SpiNNaker communication fabric is based on a 2D triangular mesh with each node formed from a processor layer and a memory layer. The routing is based upon packet-switched Address Event Representation and relies on the fact that the connections from a particular neuron are static, or at most slowly changing. Each neuron can route through a unique tree, though in practice routing is based on populations of neurons rather than individual neurons, and the restricted size of each routing table makes this optimisation necessary in most cases. In addition to the hardware system, the project also developed numerous high level neural description language, such as PyNN (Davison et al., 2008) and Nengo (Bekolay et al., 2013) for application development on SpiNNaker.

IBM TrueNorth: The IBM TrueNorth chip is the hardware developed under the DARPA SYNAPSE programme aimed at developing dense, power-efficient hardware for cognitive applications. This hardware consists of a 5.4 million transistor 28 nm CMOS chip with 4096 cores, where each core is made up of 256 neurons each having 256 synaptic inputs (Cassidy et al., 2013).

The design of the TrueNorth core is a 256×256 cross-bar which selectively connects incoming neural spike events to outgoing neurons. The cross-bar inputs are coupled via buffers that can insert axonal delays. The outputs from the cross-bar couple into the digital neuron model, which implements a form of IF algorithm with 23 configurable parameters that can be adjusted to yield a range of different behaviours, and digital pseudo-random sources are used to generate stochastic behaviours through modulating the synaptic connections, the neuron threshold and the neuron leakage. Neuron spike event outputs from each core follow individually-configurable point-to-point routes to the input to another core, which can be on the same or another TrueNorth chip. Where a neuron output is required to connect to two or more neurosynaptic cores, the neuron is simply replicated within the same core. The TrueNorth hardware is supported by a software emulator, which, exploiting the deterministic nature of the hardware, can be relied upon to predict the performance of the hardware exactly.

Apart from these two systems that rose to prominence, numerous other neuromorphic hardware systems are being constantly developed. Some of the systems worth a mention include Neurogrid (Stanford University) (Boahen, 2006), BrainScaleS (University of Heidelberg) (Markram et al., 2011), and multi-PWM pulse generator (Zheijiang University) (J.-l. Li, Wang, Li, Han & Zhang, 2005).

3.7 Brief Review of SNN Software Implementations

The number of software implementations that have appeared, as a result of on-going research in the area of ANN and SNN, is ever growing. These software packages are developed for two main purposes:

- Data analysis: They are aimed at analysing real-world data derived from practical applications. These software use a relatively simple static architecture, hence are easily configurable and easy to use. A few examples of such software are: multi-layer perceptron (MLP) (Baum, 1988), RBF network (Park & Sandberg, 1991), Probabilistic network (PNN) (Specht, 1990), Self organising maps (SOM) (Kohonen, 1998), Evolving connectionist systems, such as DENFIS and EFuNN (N. Kasabov, 2007). These software are either available as independent open source APIs, such as NeuCom ("NeuCom", n.d.), PyBrain (python) (Schaul et al., 2010), Fast Artificial Neural Network (C++) (Nissen & Nemerson, 2000), or as part of a data analytics suite like Weka (Hall et al., 2009), Knime (Berthold et al., 2008), Orange (Demar, Zupan, Leban & Curk, 2004) and so on.

- Research and development systems: As opposed to data analysis software, they are complex in behaviour, and require expert knowledge for usage and configuration. The majority of the existing SNN software belong to this class of ANN.

NEURON (M. L. Hines & Carnevale, 1997): Neuron is aimed at simulating a network of detailed neurological models. Its ability to simulate biophysical properties, such as multiple channel types, channel distributions, ionic accumulation and so on, renders it well suited for biological modelling (Brette et al., 2007). It also supports parallel simulation environment through: (1) distributing multiple simulations over multiple processors, and (2) distributing models of individual cells over multiple processors.

PyNEST (Eppler, Helias, Muller, Diesmann & Gewaltig, 2008): The neural simulation tool (NEST) is primarily developed in C++ to simulate a heterogeneous network of spiking neurons. NEST is implemented to ideally model neurons in the order of 10^4 and synapses in the order of 10^7 to 10^9 on a range of devices from single core architectures to supercomputers. NEST interfaces with python via implementation of PyNEST. PyNEST allows for greater flexibility in simulation setup, stimuli generation and simulation result analysis. A 'node' and a 'connection' comprise the core elements of the heterogeneous architecture. The flexibility to simulate 'a neuron', 'a device' or 'a subnetwork (which can be arranged hierarchically)' as a node, provides a major improvement over (Pecevski, Natschläger & Schuch, 2009). Due to the bottom-up approach of network simulation, the software allows for individually configurable neuron states and connection setup.

Circuit Simulator (Natschläger, Markram & Maass, 2003): The circuit simulator is a software developed in C++ for simulation of heterogeneous networks with major emphasis on high-level network modelling and analysis, as opposed to (M. L. Hines & Carnevale, 1997). The C++ core of the software is integrated with Matlab based GUI, for ease of use and analysis. CSIM enables the user to operate both spiking and analog neuron models along with mechanisms of spike and analog signal transmission through its synapse. It also performs dynamic synaptic behaviour by using short and long-term plasticity. In 2009, circuit simulator was further extended to parallel circuit simulator (PCSIM) software with the major extension being implementation on a distributed simulation engine in C++, interfacing with Python based GUI.

Neocortical Simulator (Drewes, 2005): NCS or Neocortical Simulator is a SNN simulation software developed for simulating mammalian neocortex (Brette et al., 2007). During its initial development, NCS was a serial implementation in Matlab but later rewritten in C++ to integrate distributed modelling capability (E. C. Wilson, 2001). As reported in (Brette et al., 2007), NCS could simulate in the order of 106 single compartment neurons and 1012 synapses using STP, LTP and STDP dynamics. Due to the considerable setup overhead of the ASCIIbased files used for the I/O, a Python-based GUI scripting tool called BRAINLAB (Drewes, 2005) was later developed to process I/O specifications for large scale modelling.

Oger Toolbox (Pecevski, n.d.): Oger toolbox is a Python-based toolbox, which implements modular learning architecture on large datasets. Apart from traditional machine learning methods such as PCA and ICA, it also implements SNN based reservoir computing paradigm for learning from sequential data. This software uses a single neuron as its building block, similar to the implementation in (Eppler et al., 2008). A major highlight of this software includes the ability to customise the network with several non-linear functions and weight topologies, and a GPU optimised reservoir using CUDA.

BRIAN (D. F. Goodman & Brette, 2009; D. F. Goodman, 2010): Brian is a SNN simulator application programming interface written in Python. The purpose of developing this API is to provide users with the ability to write quick and easy simulation code (D. F. Goodman, 2010), including custom neuron models and architecture. The model definition equations are separated from the implementation for better readability and reproducibility. D. F. Goodman and Brette (2009) also emphasise the use of this software in teaching a neuroinformatics course (Diesmann, Gewaltig & Aertsen, 1999). A major limitation of BRIAN is, however, the requirement of Python knowledge to run the simulation, and the lack of GUI for the non-technical user community.

3.8 Evolving Connectionist System

It is beyond any speculation that AI and neuromorphic hardware systems belong to a unified symbiotic ecosystem, where a neuromorphic hardware is truly neuromorphic if and only if it has dynamic learning, adaptation and creativity incorporated in the framework. On the other hand neuromorphic learning systems are paradigm shifting in machine learning where it is not only about accurate performance, but also about performing it economically using limited resources.

Subsequently, I will concentrate on the principle and evolution of the Evolving Connectionist System (ECOS) with emphasis on our work in evolving spiking neural network (eSNN). The rest of the Chapter is an adaptation of our work presented in (Arya, Vadlamani, Valadi, Sengupta & Kasabov, 2016) along with an article under review.

3.8.1 Principles of ECOS

The human brain's unique ability to combine low level learning within an interconnected framework and high level rule abstraction leads to the learning of abstract concepts. ECOS is inspired by this very concept and aims at training neural networks for deriving abstract knowledge representations that explains data and can be used as an interpretable knowledge-based system. The ECOS was developed as a trend in neural networks and computational intelligence by N. Kasabov (2001) and incorporated in many novel computational methods over the last few decade across different application domains. Some classical example of ECOS systems are EFuNN (N. Kasabov, 1998) and DENFIS (N. Kasabov & Song, 2002), where classical McCulloch and Pitts neuron models are used to perform pattern recognition in single dimensional scalar data. ECOS principle was propagated further to evolving spiking neural networks (eSNN) that used spiking neuron models and temporal spike sequences as data representation. The eSNN was architected primarily as a visual pattern recognition system. The first eSNNs were based on the Thorpes 'time to first spike' rule (S. Thorpe et al., 2001), in which the importance of early spikes (after the onset of a certain stimulus) is boosted, called rank-order encoding and learning. N. Kasabov, Dhoble, Nuntalid and Indiveri, 2013 further proposed dynamic eSNN (deSNN), which combines rank order encoding of eSNN with dynamic Hebbian spike-time based learning. The main advantage of the eSNN when compared with other supervised or unsupervised SNN models is that it is computationally inexpensive due to the one-pass nature of the learning algorithm and boosts the importance of the order in which input spikes arrive, thus making the eSNN based algorithms suitable for on-line learning with a range of applications. For a comprehensive study of ECOS and eSNN see (Watts, 2009; Wysoski, Benuskova & Kasabov, 2010). I, in collaboration with The Institute of Development and Research in Banking Technology, India, have used eSNN in applications of automatic cyber fraud detection and stock price movement prediction case studies. In the following two Sections, the results that have been obtained will be summarised from these two studies. The detail of each of the studies can be found in Appendix A and B

3.8.2 Application of eSNN in Forecasting Stock Price Movement

Prediction of the stock price index and its trend are considered challenging tasks due to their complexity, non-linearity, dynamic and chaotic nature. In addition, the stock market behaviour is also influenced by socio-political movements, investor psychology, and so on. Although many researchers have explored several forecasting techniques, it is for the first instance that eSNN has been employed to develop a computational model for the stock market movement trend prediction. In this study, two computational models are proposed, namely, CUDA-SI-eSNN, which is a parallel CUDA implementation of the standard eSNN architecture. The second algorithm, known as SW-eSNN, is the incremental learning of eSNN using a sliding window of data. Appendix B describes the motivation behind this study along with descriptions of the proposed SI-eSNN, CUDA-SI-eSNN and SW-eSNN architectures along with the formal description of the algorithms.

Dataset Description and Experiments with the SI-eSNN and the CUDA-eSNN Models

The datasets used in this study are obtained from QUANDL ("Quandl Financial, Economic and Alternative Data." n.d.), ("Historical - Indices." n.d.), and ("NSE - National Stock Exchange of India Ltd." n.d.). These datasets cover stock market indices of different countries: BSE, Nikkie-225, NIFTY-50, S&P-500, Dow-Jones, NYSE-Amex, DAX, NASDAQ and Shanghai stock exchange. In this study, thirteen technical indicators have been selected as input variables based on earlier research done in (K.-j. Kim, 2003; J. Patel, Shah, Thakkar & Kotecha, 2015; Kara, Boyacioglu & Baykan, 2011). The direction of daily stock price index is categorised as 'UP' or 'DOWN'. If the stock price index at time t is higher than that at time $t + 1$, then the trend is 'DOWN'. If the stock price index at time t is lower than that at time $t + 1$, then the trend is 'UP'. The number of instances of each of the stock index data is given in Table B.3. The details about selected indicators is given in Table B.2 and the summary statistics of selected technical indicators for each stock indices are provided in the supplementary information.

In the experiments, the first 70% of the temporal stock data represented by the 13 indicators on a daily basis are used as input variables for training the SI-eSNN and the CUDA-SI-eSNN models and the future 30% of the time series stock data is used to test the model accuracy. Experiments in the present study were carried on systems with 32 GB RAM and eight cores. The GPU device used for all of our experiment is GeForce GT730. The details of the GPU device is given in Table B.4.

Classification accuracy and AUC scores were used to evaluate the performance of both eSNN and CUDA-SI-eSNN models. Sensitivity and specificity are used to assess the AUC score. In addition to the sensitivity, specificity and accuracy equations described earlier, the AUC score can be defined as:

$$AUC\ score = \frac{Sensitivity + Specificity}{2} \tag{3.16}$$

Table 3.3.: Accuracy and AUC of CUDA-eSNN model with Gaussian Distribution for different stock indices.

Datasets	BSE			Nikkie-225			NASDAQ			NIFTY-50			S&P-500		
No. of Gaussian Receptive Fields	Acc. (%)	T.T (Sec)	AUC	Acc. (%)	T.T (Sec)	AUC	Acc. (%)	T.T (Sec)	AUC	Acc. (%)	T.T (Sec)	AUC	Acc. (%)	T.T (Sec)	AUC
3	82.17	0.84	0.82	81.57	3.3	0.81	84.46	0.89	0.84	76.85	0.61	0.76	82.96	8.51	0.82
4	85.71	1.06	0.86	82.53	4.67	0.82	85.43	1.1	0.85	79.96	0.7	0.79	83.74	9.77	0.83
5	83.88	1.25	0.83	80.33	5.96	0.8	84.22	1.23	0.84	80.13	0.69	0.8	81.16	11.59	0.81
6	85.59	1.36	0.85	84.41	7.15	0.84	84.22	1.24	0.83	77.37	0.79	0.77	82.72	13.78	0.82
7	87.05	1.39	0.87	80.88	8.06	0.80	**85.92**	**1.53**	**0.86**	82.55	0.81	0.82	82.53	18.95	0.82
8	85.47	1.55	0.85	86.15	8.91	0.85	81.79	1.55	0.81	82.03	0.84	0.81	84.03	22.05	0.84
9	**88.76**	**1.65**	**0.88**	82.12	9.68	0.82	84.7	1.67	0.84	81.69	0.93	0.81	82.16	29.05	0.82
10	86.08	1.75	0.86	**87.12**	**10.6**	**0.87**	85.92	1.84	0.85	83.93	0.99	0.83	84.13	35.7	0.84
11	88.64	1.78	0.88	83.73	11.58	0.83	84.7	1.91	0.84	80.31	1.05	0.8	84.5	39.36	0.84
12	87.91	1.88	0.88	86.11	12.41	0.86	83.49	2.04	0.83	82.38	1.08	0.82	81.26	46.94	0.81
13	88.4	2	0.88	83.68	13.15	0.83	84.22	2.19	0.84	**84.45**	**1.18**	**0.84**	**84.74**	**51.51**	**0.84**
14	87.91	2.25	0.88	86.57	14.04	0.86	83.73	2.19	0.83	81	1.22	0.8	84.13	51.88	0.84

∗Acc.=Accuracy, ∗T.T = Training Time

Table 3.4.: Accuracy and AUC of CUDA-eSNN model with Gaussian Distribution for different stock indices (Continued).

	Sanghai Stock Exchange			Dow-Jones			NYSE-Amex			DAX-Index		
No. of Gaussian Receptive Fields	Acc. (%)	T.T (Sec)	AUC	Acc. (%)	T.T (Sec)	AUC	Acc. (%)	T.T (Sec)	AUC	Acc. (%)	T.T (Sec)	AUC
3	77.48	1.68	0.77	75.31	0.83	0.74	78.37	2.01	0.77	79.23	2.67	0.79
4	78.14	2.08	0.78	76.59	0.93	0.76	84.04	2.5	0.83	77.64	4	0.77
5	80.74	3.13	0.8	75.31	0.96	0.74	82.75	3.67	0.81	81.55	5.12	0.81
6	80.37	3.3	0.8	78.38	1.05	0.78	80.56	3.96	0.8	82.07	5.79	0.81
7	79.92	3.7	0.79	79.79	1.23	0.79	82.62	4.4	0.82	78.41	6.43	0.78
8	79.18	3.73	0.79	79.28	1.46	0.79	77.79	4.7	0.77	82.22	7.22	0.81
9	79.48	3.76	0.79	78.51	1.49	0.78	82.78	5.1	0.83	82.53	7.76	0.82
10	80.37	4.8	0.8	77.74	1.59	0.77	83.65	5.51	0.82	82.07	8.48	0.81
11	**82.66**	**5.2**	**0.82**	77.62	1.69	0.77	81.78	6.05	0.81	83.41	9.11	0.83
12	79.77	5.33	0.79	79.28	1.77	0.79	86.29	6.45	0.85	**84.9**	**9.66**	**0.84**
13	79.7	5.9	0.79	**81.2**	**1.84**	**0.8**	83.78	6.87	0.83	82.22	10.68	0.81
14	80.44	5.44	0.8	80.17	2.03	0.79	**86.42**	**7.42**	**0.86**	84.75	11.23	0.84

∗Acc.=Accuracy, ∗T.T = Training Time

Tables 3.3 and 3.4 reports the accuracy and AUC of CUDA-SI-eSNN model for a different number of Gaussian receptive fields on various stock indices.

In Table 3.5, the performance of both SI-eSNN and CUDA-SI-eSNN models have been included for the best value of some Gaussian receptive fields. The accuracy of both these models is very high across all stock indexes (between 80% and 90%) and only slightly different between the two models. Table 3.5 also reports the time for training performed by both the SI-eSNN and the CUDA-SI-eSNN on the same data. While the latter is 2 to 5 times faster, the SI-eSNN training was very fast as well. For example, it took only 40 seconds to train the SI-eSNN on 5000 samples for the S&P-500, while it took 20 seconds on the CUDA-SI-eSNN.

Table 3.5.: Comparisons of SI-eSNN and CUDA eSNN for the best value of number of Gaussian receptive fields.

			eSNN			CUDA-eSNN		
Dataset	Size of training data	No. of Gaussian Receptive Fields (Best)	Accuracy (%)	Training Time (Sec)	AUC	Accuracy (%)	Training Time (Sec)	AUC
BSE	1912	9	88.15	**5.91**	0.88	**88.76**	1.65	0.88
Nikkie-225	5092	10	87.12	23.67	0.87	**87.12**	10.6	0.87
NASDAQ	1923	7	**86.04**	5.22	0.86	85.92	1.53	0.86
NIFTY-50	1353	13	83.76	5.98	0.84	**84.45**	1.18	0.83
S&P-500	9591	13	84.71	82.56	0.84	**84.74**	51.51	0.84
Shanghai Stock Exchange	3149	11	**83.18**	12.87	0.82	82.66	5.2	0.83
Dow-Jones	1824	13	81.2	8.53	0.8	**81.2**	1.84	0.8
NYSE-Amex	3625	14	86.16	20.09	0.86	**86.42**	7.42	0.86
DAX-Index	4527	12	84.16	24.05	0.84	**84.9**	9.66	0.84

Table 3.6.: Accuracy and AUC of CUDA-SI-eSNN model using Logistic Distribution for different stock indices.

Datasets	BSE			Nikkie-225			NASDAQ			NIFTY-50			S&P-500		
No. of Logistic Receptive Fields	Acc. (%)	T.T (Sec)	AUC	Acc. (%)	T.T (Sec)	AUC	Acc. (%)	T.T (Sec)	AUC	Acc. (%)	T.T (Sec)	AUC	Acc. (%)	T.T (Sec)	AUC
3	78.14	0.85	0.78	84.87	3.05	0.84	70.87	0.87	0.71	72.02	0.57	0.72	70.85	8.24	0.7
4	77.53	1.01	0.75	86.98	3.32	0.86	83.98	1.12	0.83	78.23	0.64	0.78	79.68	9.16	0.79
5	79.36	1.12	0.78	81.57	3.7	0.81	83.85	1.11	0.83	82.21	0.68	0.82	84.47	11.01	0.84
6	84.37	1.32	0.84	86.29	4.29	0.86	85.19	1.27	0.84	80.65	0.72	0.8	80.58	12.23	0.8
7	86.81	1.38	0.86	81.57	5.07	0.81	87.37	1.81	0.87	83.24	0.76	0.83	83.09	15.37	0.82
8	85.34	1.55	0.85	86.34	6.45	0.86	86.89	1.52	0.86	81.69	0.79	0.81	**85.86**	**19.97**	**0.85**
9	**90.47**	**1.63**	**0.9**	84.55	7.67	0.84	86.77	1.65	0.86	82.03	0.89	0.82	83.81	24.53	0.83
10	86.81	1.74	0.86	**86.66**	**9.73**	**0.86**	85.8	1.74	0.85	83.93	0.96	0.83	84.28	30.96	0.84
11	89.01	1.78	0.89	85.24	10.59	0.85	**87.5**	**1.85**	**0.87**	81.69	1.02	0.81	85.71	35.78	0.85
12	88.03	1.89	0.88	86.38	11.58	0.86	87.01	1.99	0.86	84.8	1.09	0.84	83.5	41.02	0.83
13	89.13	2.03	0.89	86.52	12.87	0.86	86.16	2.05	0.86	**85.14**	**1.15**	**0.85**	84.54	45.49	0.84
14	87.05	2.08	0.87	86.61	13.87	0.86	86.28	2.19	0.86	84.8	1.18	0.84	85.76	49.69	0.85

Table 3.7.: Accuracy and AUC of CUDA-SI-eSNN model using Logistic Distribution for different stock indices (Continued).

	Sanghai Stock Exchange			Dow-Jones			NYSE-Amex			DAX-Index		
No. of Logistic Receptive Fields	Acc. (%)	T.T (Sec)	AUC	Acc. (%)	T.T (Sec)	AUC	Acc. (%)	T.T (Sec)	AUC	Acc. (%)	T.T (Sec)	AUC
3	78.74	1.6	0.78	73.91	0.79	0.73	69.75	2.07	0.69	75.32	2.65	0.74
4	78.88	1.95	0.78	74.8	0.81	0.74	81.59	2.43	0.8	77.94	3.63	0.77
5	77.85	2.54	0.77	78.9	0.86	0.78	81.14	3.14	0.8	85	4.78	0.84
6	81.11	3.03	0.81	78	0.92	0.78	81.85	3.63	0.81	81.04	5.59	0.8
7	80	3.23	0.79	79.66	1.05	0.79	86.16	4.15	0.85	80.47	6.24	0.8
8	79.7	3.49	0.79	81.45	1.27	0.81	82.11	4.59	0.81	82.27	6.82	0.81
9	80.29	3.78	0.8	79.79	1.34	0.79	84.42	4.91	0.84	83.51	7.57	0.83
10	81.92	4.15	0.81	79.79	1.46	0.79	84.23	5.34	0.83	82.63	8.11	0.82
11	**83.03**	**4.53**	**0.83**	82.09	1.63	0.81	84.1	5.79	0.83	84.49	8.83	0.84
12	81.55	4.73	0.81	80.56	1.74	0.8	**86.62**	**6.23**	**0.85**	**85.67**	**9.57**	**0.85**
13	81.03	5.1	0.81	**82.99**	**1.81**	**0.83**	83.91	6.87	0.83	84.23	10.25	0.83
14	82.48	5.41	0.81	81.84	1.97	0.81	85.97	7.14	0.85	84.75	11.07	0.84

Table 3.8.: Comparisons of SI- eSNN and CUDA-SI-eSNN for the best value of number of Logistic receptive fields.

			eSNN			CUDA-SI-eSNN		
Dataset	Size of training data	No. of Logistic Receptive Fields (Best)	Accuracy (%)	Training Time (Sec)	AUC	Accuracy (%)	Training Time (Sec)	AUC
BSE	1912	9	90.04	5.78	0.89	**90.47**	1.63	0.9
Nikkie-225	5092	10	86.66	22.17	0.86	**86.66**	9.73	**0.86**
NASDAQ	1923	11	**88.14**	6.14	0.87	87.5	1.85	0.87
NIFTY-50	1353	13	84.74	5.65	0.84	**85.14**	1.15	0.85
S&P-500	9591	8	85.23	40.83	0.85	**85.86**	19.97	0.85
Sanghai Stock Exchange	3149	11	**83.85**	11.83	0.83	83.03	4.53	0.83
Dow-Jones	1824	13	82.87	5.43	0.83	**82.99**	1.81	0.83
NYSE-Amex	3625	12	85.94	19.23	0.85	**86.62**	6.23	0.85
DAX-Index	4527	12	85.16	23.02	0.85	**85.67**	9.57	0.85

Table 3.9.: *p* values of McNemar test for the pairwise comparison of performance of CUDA-SI-eSNN with Logistic and Gaussian distribution.

Stock Indices	p Values
BSE	0.648
Nikkie-225	0.804
NASDAQ	0.794
NIFTY-50	0.470
S&P-500	0.269
SSE	0.044
DJUS	0.007
NYSE-Amex	0.001
DAX-Index	0.125

The results of CUDA-SI-eSNN with Logistic Distribution is presented in Tables 3.6 and 3.7. Logistic distribution was employed in place of Gaussian in (Farquad, Ravi & Raju, 2012) too, where better results were obtained with the former. The experimental results showed that CUDA-SI-eSNN with Logistic distribution outperforms the CUDA-SI-eSNN model with Gaussian distribution on all stock indices except on Nikkie-225 stock index. The accuracy of CUDA-SI-eSNN with Logistic distribution varies between 82.99% and 90.47%. The result of the CUDA-SI-eSNN with the best value of a number of logistic receptive fields presented in Table 3.8. The training time of the CUDA-SI-eSNN with logistic receptive fields is different from the training time of the CUDA-SI-eSNN with Gaussian receptive fields for the same number of receptive fields is due to running of other background processes. The CUDA-SI-eSNN model with logistic receptive fields outperformed the CUDA-SI-eSNN model with Gaussian receptive fields in all stock indices since logistic distribution has higher kurtosis than the Gaussian distribution.

To test whether CUDA-SI-eSNN with Logistic Distribution significantly outperforms another model, the McNemar test was used. The result of McNemar test is given in Table 3.9. The McNemar test showed that CUDA-SI-eSNN with Logistic Distribution significantly outperforms other model on DJUS, NYSE-Amex, and SSE stock indices at 5% statistical significance level.

Table 3.10.: Average AUC score of SW-eSNN Incremental approach using Logistic and Gaussian distributions.

Dataset	eSNN+Logistic	eSNN+Gaussian
BSE	**0.77**	0.71
Nikkie-225	**0.72**	0.69
NASDAQ	0.76	**0.77**
NIFTY-50	**0.69**	0.67
S&P-500	**0.75**	0.73
Sanghai Stock Exchange	**0.67**	0.65
Dow-Jones	**0.73**	0.7
NYSE-Amex	**0.73**	0.69
DAX-Index	**0.72**	0.7

Appendix B also presented the SW-eSNN model. In this approach, a model is first trained with one-year data and then tested for the prediction on the subsequent month. Subsequently, the window was slid by one-month for next one-month prediction. The experimental results show that the average AUC score of the models over all the benchmark data sets vary from 65% to 80% and for some months reaches 100%. The manifested fluctuation across monthly prediction accuracy is expected with this model due to the short window length used and external factors affecting the stock price movement.

The average AUC score of SW-eSNN incremental approach for both logistic distribution and Gaussian distribution on all stock indices is presented in Table 3.10. This result shows that SW-eSNN incremental approach with logistic distribution outperformed SW-eSNN incremental approach with Gaussian distribution on all stock indices except on NASDAQ stock index.

3.8.3 Application of eSNN in Cyber Fraud Detection

In the present work (Arya et al., 2016) published in 2016, an application of eSNN based system for detecting cyber frauds in phishing websites was presented. This study has applied the eSNN algorithm to learn a model for detecting phishing websites from URL and web page source information. The detail of the eSNN learning algorithm used is described in Appendix A.

Table 3.11.: Comparison of performance of eSNN and other iterative and non iterative machine learning algorithms on the phishing website data.

Classifiers	Type	Sensitivity (%)	Specificity (%)	Accuracy (%)
Gaussian process	Iterative	99	100	99.5
Logistic Regression		91	88	89.5
MLP		89	80	84.5
CART		94	90	92
Gaussian Process+CART		94	87	90.5
Probabilistic NN	One pass	93	92	92.5
eSNN		89	90	89.5

Table 3.12.: t-test based model comparison.

		t-statistic value (accuracy)
eSNN vs.	Gaussian Process	4.33
	Logistic Regression	1.12
	PNN	1.17
	MLP	3.04
	CART	0.2

Dataset Description

For the purpose of experiments on phishing website detection, a web phishing dataset was used for phishing website detection. 200 phishing websites were chosen for analysis, where 50% URL's were phishing website and the rest were legitimate websites URLs. The URLs are collected from PhishTank (www.phishtank.com). The dataset was created by extracting 17 features based on t-statistics values from web pages source code and URLs.

Results

The result of the experiments performed on the web phishing data sets are presented in Tables 3.11 and 3.12. All the reported results are 10-fold cross validated. The details of eSNN model testing method and hyperparameter selection protocol are described further in Appendix A. The performances of the various models were compared with respect to sensitivity, specificity and overall accuracy. These measures are defined as the following:

$$Specificity = \frac{TN}{TN + FP} \tag{3.17}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{3.18}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.19}$$

Where TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative, where a phishing website is considered to be a positive class and non-phishing as negative class.

In Table 3.11, the proposed eSNN algorithm for cyber fraud detection is compared with the iterative and one-pass learning algorithms. The result shows that the proposed eSNN model achieves better performance with respect to overall accuracy and sensitivity/specificity in comparison to the one pass PNN algorithm. Table 3.12 compares the eSNN model performance with the benchmark models based on the t-statistic value. The t-test value is found to be less than 2.83 (t-table value concerning 18 degrees of freedom, *i.e.*, $10 + 10 - 2 = 18$) in all the classifier except GP and MLP. For all the cases, eSNN performance is statistically equivalent.

3.9 Contributions and Publications

Contributions

1. Historical evolution of artificial neuron and neural networks.
2. Review and motivation of the hardware and software implementation of spiking neural networks.
3. Application of evolving spiking neural network on cyber fraud detection.
   - Implementation of a novel sequential eSNN classification algorithm.
   - Comparison of the proposed algorithm with iterative and one-pass learning algorithms on cyber fraud detection problem.
4. Application of evolving spiking neural network on stock movement forecast.
   - GPU implementation of the evolving spiking neural network algorithm.
   - Implementation of the sliding window approach for evolving spiking neural network.
   - Evaluation of one day ahead stock index forecast across 9 different stock indices.

Publications

1. Kasabov, N., **Sengupta, N.**, & Scott, N. (2016, September). From von neumann, John Atanasoff and ABC to Neuromorphic computation and the NeuCube spatio-temporal data machine. In IEEE 8th International Conference on Intelligent Systems (IS), 2016 (pp. 15-21). IEEE.

2. Arya, A. S., Ravi, V., Tejasviram, V., **Sengupta, N.**, and Kasabov, N. (2018, January) Cyber fraud detection using evolving spiking neural network, in IEEE International conference on industrial and information systems (ICIIS), 2016 (pp. 263-268). IEEE.

4 A Review of NeuCube: An Evolving Spatio-Temporal Data Machine Framework

In the last decade or so, with the advent of deep neural network, a significant proportion of research in artificial intelligence and machine learning has been conducted in the area of neural networks. This has led to a massive diversification in the architecture and functionality of the neural networks. This diversity across the domain of neural networks makes it extremely difficult to review the area exhaustively or even perform systematic taxonomy as such. van Veen (2016) has visually demonstrated (see Figure 4.1) the node level block architectures of several popular neural networks. This is a very useful comparative diagram to provide broad understanding of different neural networks. However, care must be taken in interpreting the networks from the figure as the node level diagram hides several degrees of abstractions and similar looking architectures (e.g. VAE and DAE), which may be greatly different with respect to training protocol and application areas.

In this Chapter, the focus will be shifted towards discussing the novel evolving spatio-temporal data machine, NeuCube, proposed by Nikola Kasabov in (N. Kasabov, 2012). This framework is especially designed to take advantage of heterogeneous properties present in the data, especially in the form of spatial and temporal information. NeuCube draws its inspiration from the of recurrent neural network based reservoir computing paradigm and this topic will be discussed further.

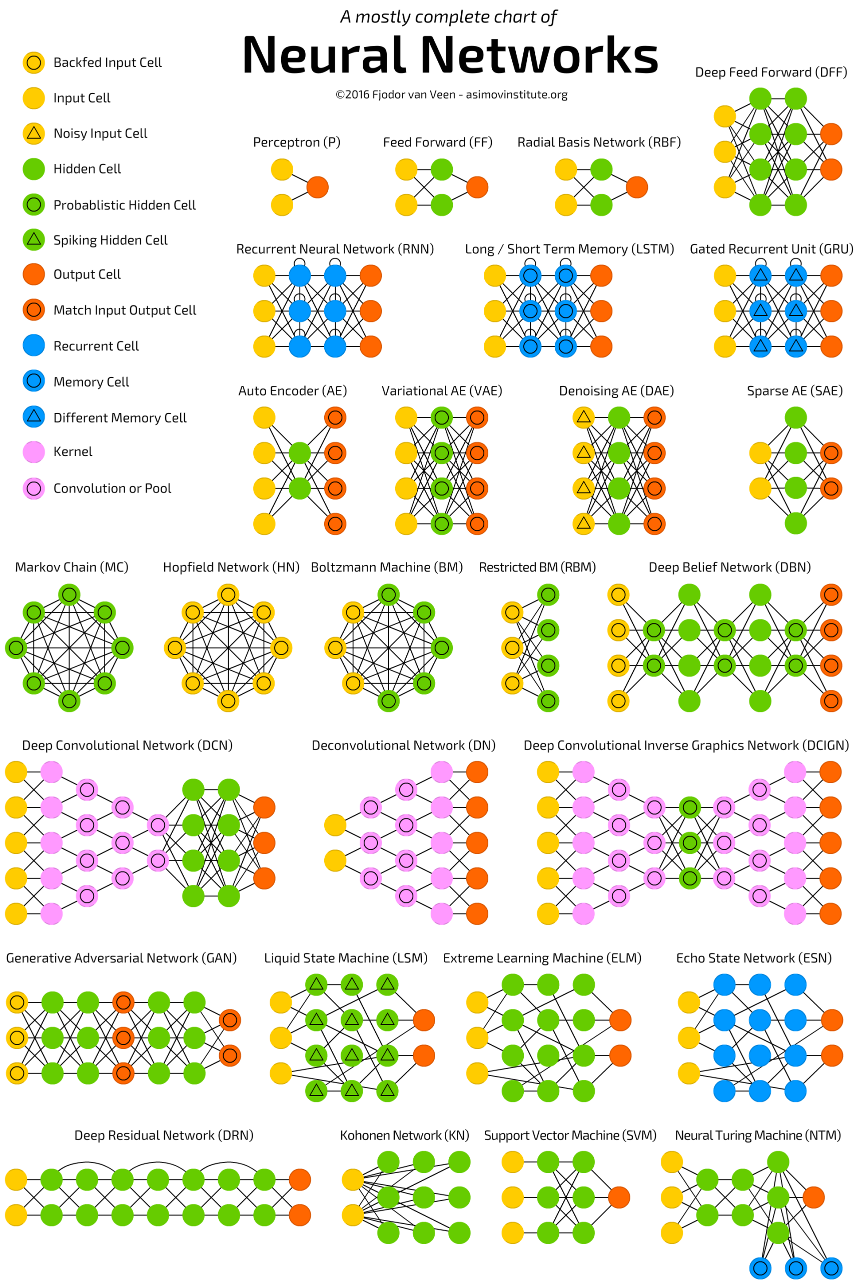

A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

Figure 4.1.: Node level architectures of popular neural networks (source van Veen (2016)).

4.1 Recurrent Neural Networks

Rather than diverting into describing all the networks (see (van Veen, 2016) for a brief description of all the architectures shown in Figure 4.1), I will concentrate on the recurrent neural networks (RNN). RNNs can be described simply as feed forward neural networks (FFNN) with a time twist. RNNs consist of a set of processing elements (neurons) which are interconnected by abstractions of synaptic connections. The interconnected mesh enables activations to be propagated through the network. The characteristic feature that differentiates RNN from FFNN is the connection topology. RNNs possess cyclic connections forming intra and inter-layer loops. The existence of cyclic connection has profound impact:

1. Even in the absence of input signals, RNNs possess the ability to develop a self-sustained activation dynamics along the cyclic connections. This property allows RNNs to act as a dynamic system as opposed to the function like behaviour of FFNN.

2. RNNs possess dynamic memory and are able to make use of temporal contextual information.

The philosophy of the recurrent neural network is rooted in the fact that humans do not start thinking from scratch every second. When an individual reads or speaks, their understanding of a concept at any point in time is contextually dependent on what they understood earlier. Thoughts have persistence. Traditional neural networks are not capable of using information in a contextual manner, which is a major shortcoming. For example, to classify events in a video over time, it is unclear how a traditional neural network could use its reasoning about previous events in the video to inform future events.

RNNs are built to address this exact issue. The network retains information by including loops in them. On the left hand side of Figure 4.2, a RNN block is

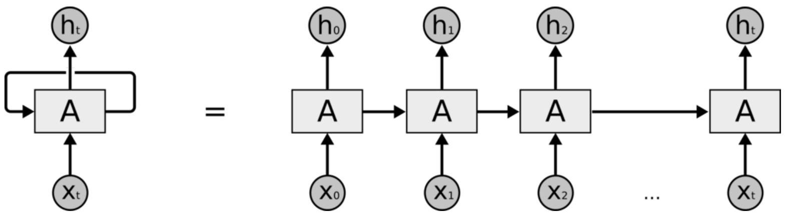


Figure 4.2.: Unrolled RNN (source Olah (2015)).

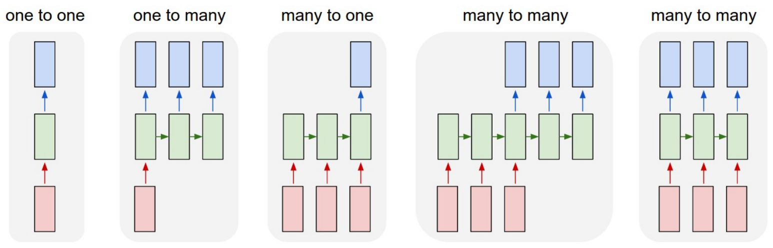


Figure 4.3.: Mapping capabilities of RNN (source Karpathy (2015)).

shown. Block A, receives some input x_t and outputs a value h_t. A loop passes the information from one step of the network to the next step (Olah, 2015). An RNN can be intuitively described as multiple copies of the same network unrolled across the time dimension (see Figure 4.2). This chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists. They are the natural architecture of neural networks to use for such data.

4.1.1 What Makes RNN Effective?

Andrej Karpathy in his blog post (Karpathy, 2015) addresses this question with great detail. In his post, he argued that the ability of the RNNs to operate over sequences of vectors, input, output or both makes it more powerful. Figure 4.3 taken from (Karpathy, 2015) describes the effectiveness with respect to mapping capabilities of RNN. Each rectangle in the figure represents a vector and the arrows are functions (such as matrix multiplication). The input vectors are coloured in red, output vectors in blue and RNN hidden states in green. From left to right, it shows: (1) The typical FFNN style vanilla processing with fixed size input and output; (2) Sequence of outputs (Image captioning. One image as input and a sequence of words as output); (3) Sequence of inputs (Sentiment analysis. Sequence of word as input and sentiment as output); (4)Sequence input and delayed sequence output (Machine translation); and, (5) Synced sequence input and output (Video frame labelling).

From a dynamic system viewpoint, two major classes of RNNs exist. The first class of models are characterised by symmetric connections and energy minimising stochastic dynamics (Lukoeviius & Jaeger, 2009). The best known architectures of this category are Hopfield Network (Hopfield, 1982), Boltzmann Machine (Hinton & Sejnowski, 1986), Deep Belief Network (Bengio, Lamblin, Popovici & Larochelle, 2007) and Long Short Term memory (Hochreiter & Schmidhuber, 1997a). These networks are mostly trained in some unsupervised learning scheme. Typical targeted network functionalities in this field are associative memories, data compression, the unsupervised modelling of data distributions, and static pattern classification, where the model is run for multiple time steps per single input instance to reach some type of convergence. In contrast, the second big class of RNN models typically features a deterministic update dynamics and directed connections. Systems from this class implement non-linear filters, which transform an input time series into an output time series. The mathematical background here is non-linear dynamical systems (Lukoeviius & Jaeger, 2009). I will focus on the second category of RNNs as part of my work.

4.2 Formalisation of the Temporal Learning Problem

A traditional machine learning task can be expressed mathematically as the learning of functional dependence given an input $x(n) \in \mathbb{R}^{N_x}$ and ground truth $y(n) \in \mathbb{R}^{\mathbb{N}}$, where $n = 1, \cdots, T$, and T is the number of samples in the training dataset $\{x(n), y(n)\}$. In a static data scenario, there is no temporal dependence between the samples, and the objective is to learn a function $\hat{y} = f(x)$, such that the error or loss function $E(\hat{y}, y)$ is minimised. On the contrary, in a temporal task, x and y are signals in a discrete time domain $n = 1, \cdots, T$, and the goal is to learn $\hat{y} = f(x(1), \cdots, x(n-1), x(n))$, such that that $E(\hat{y}, y)$ is minimised.

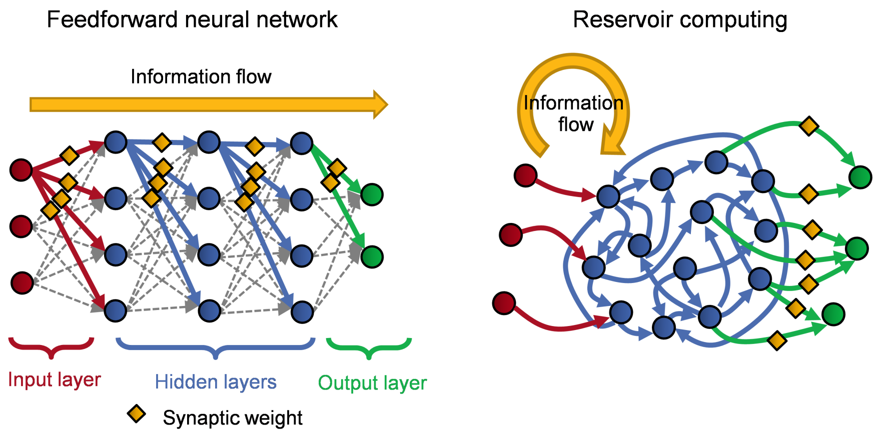


Figure 4.4.: Schematic diagram of a feed forward neural network vs. a reservoir computing system (source Abel and Fompeyrin (n.d.)).

This formalism clearly shows that in the temporal learning scenario, the function learned is stateful or memory driven as opposed to the stateless non-temporal task. In a dynamic filter approach of RNN, it typically implements the non-linear expansion of memory as a state vector of the form described in the equation below:

$$h(n) = f(W_{in}x(n) + Wx(n-1), \cdots), \; n = 1, \cdots, T \tag{4.1}$$

where h is the activation function of a computational unit.

4.3 Reservoir Computing and Liquid State Machines

Figure 4.4 compares the node level diagram of a typical FFNN and the reservoir computing approach. Reservoir computing as described by Schrauwen, Verstraeten and Van Campenhout (2007) is a dynamic filter, where an input signal is fed into a fixed and random dynamic system called reservoir, and the dynamics of the reservoir performs non-linear expansion of the input into a higher dimensional space. A recurrence-free readout mechanism then takes the data and maps it to the desired output. Figure 4.5 shows the intuition of reservoir comput-

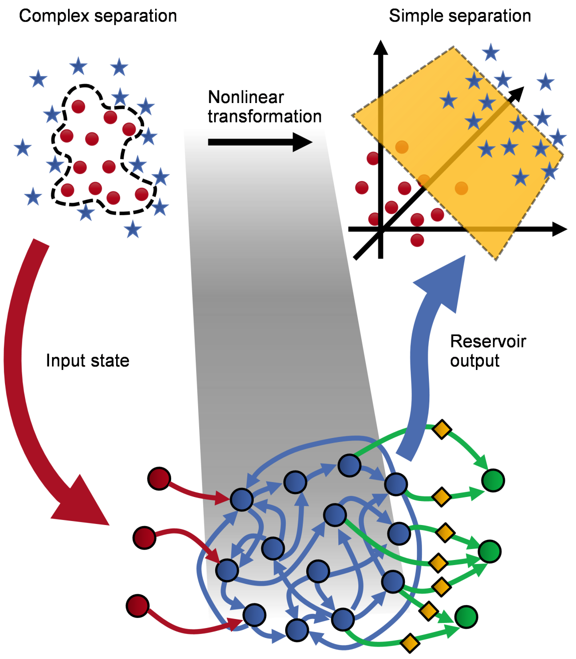


Figure 4.5.: Principle of reservoir computing: The input states are transformed into a high-dimensional feature space where classification can be performed with linear operation (source Abel and Fompeyrin (n.d.)).

ing paradigm in a schematic diagram. Since the expansion and the readout serve different purposes, training/generating them separately and even with different goal functions makes sense. The blue nodes in FFNN acts as non-dynamic and non-linear transfer functions. On the other hand, the blue nodes in a reservoir computing paradigm acts as a reservoir of recurrently connected neurons which possess the ability to act stateful and transform data into a higher dimensional space for the readout layer (green) to map into output with relative ease. The "traditional" RNN training methods do not make the conceptual separation of a reservoir vs. a readout, and train both reservoir-internal and output weights in technically the same fashion. For a detailed review of reservoir computing, see (Lukoeviius & Jaeger, 2009).

4.3.1 Liquid State Machines

Liquid State Machines (LSM) are a special kind of RNN architecture within the reservoir computing paradigm proposed by Maass, Natschläger and Markram (2002). LSMs were primarily intended for elucidating computational properties of microcircuits from a computational neuroscience perspective. Therefore, LSMs possess great sophistication and biological realism with respect to the behaviour of the neurons within the reservoir. The reservoir of LSM is often described as a liquid state. Input data feeds to the reservoir resembles "throwing pebbles into a pond" creating ripples in the fluid in the reservoir. The interaction of the ripples creates useful patterns over space and time. The neurons of LSM are spiking in nature and due to the biological plausibility are extremely complex and parameter-heavy in nature. For this reason, the LSMs have proven to be not only computationally expensive but also difficult to tune. However, a major advantage of using LSM is its ability to perform complex information processing through temporal encoding in a highly efficient manner. I will provide discussion on this topic in Chapter 6. The main theoretical contributions of the LSM brand to reservoir computing consist in analytical characterisations of the computational power of such systems.

4.4 NeuCube Evolving Spatio-temporal Data Machine

The brain is a complex integrated spatio-temporal information processing machine. The mammalian brain is made up of spatially distributed structural and functional areas constrained within a three dimensional space. External stimuli and/or inner processes are of a varying nature, such as visual, auditory, somatosensory, olfactory and so on, from which emanate a complex spatio-temporal activity path within the brain leading to highly efficient and accurate recognition of patterns.

For example Benuskova and Kasabov (2010) provide the following example:

"· · · the language task involves transfer of information from the inner ear through the auditory nucleus in thalamus to the primary auditory cortex (Brodmanns area 41), then to the higher-order auditory cortex (area 42), before it is relayed to the angular gyrus (area 39). Angular gyrus is a specific region of the parietal-

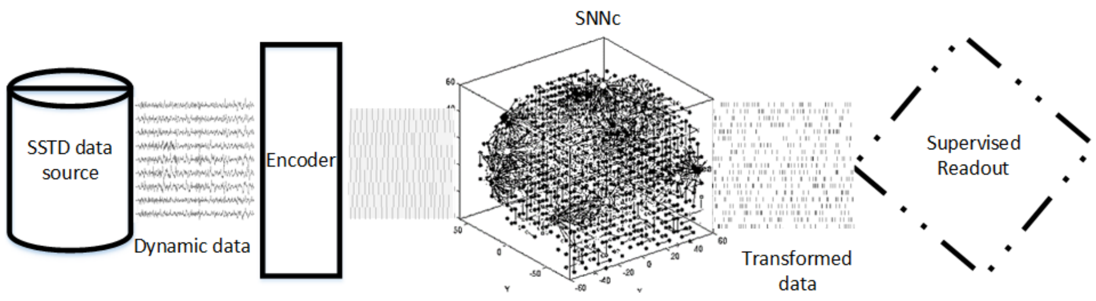


Figure 4.6.: Schematic architecture diagram of NeuCube.

temporal-occipital association cortex, which is thought to be concerned with the association of incoming auditory, visual and tactile information. From here, the information is projected to Wernickes area (area 22) and then, by means of the arcuate fasciculus, to Brocas area (44, 45), where the perception of language is translated into the grammatical structure of a phrase and where the memory for word articulation is stored. This information about the sound pattern of the phrase is then relayed to the facial area of the motor cortex that controls articulation so that the word can be spoken. It turns out that a similar pathway is involved in naming an object that has been visually recognised. This time, the input proceeds from the retina and LGN (lateral geniculate nucleus) to the primary visual cortex, then to area 18, before it arrives at the angular gyrus, from where it is relayed by a particular component of arcuate fasciculus directly to Brocas area, bypassing Wernickes area."

The example above shows that the brain processes information through the activation of complex spatiotemporal trajectories involving multiple functional areas. The NeuCube evolving Spatio-Temporal Data Machines are derived exactly from this very philosophy.

The generic principles of the NeuCube architecture was introduced in (N. Kasabov, 2012) and was further elaborated in (N. Kasabov et al., 2014; N. K. Kasabov, Doborjeh & Doborjeh, 2017; Sengupta, Ramos et al., 2018). Figure 4.6 shows a schematic diagram of the NeuCube evolving Spatio-Temporal Data Machine (eSTDM) architecture. This framework is designed to use the spatial and temporal relationships in the data and therefore expects a spatio-temporal data source to feed spatio or spectro temporal data (SSTD) to the system. The NeuCube framework is a tiered architecture of three layers.

4.4.1 Encoding

The encoding layer is the first layer of NeuCube. This layer is responsible for converting real world continuous SSTD into a sequence of events or binary spikes. Formally, the encoding layer transforms real continuous data $\mathbb{R}^{N \times T}$ (N is the feature count or the spatial component and T is the temporal component) into spike trains $\{0,1\}^{N \times T}$. Numerous temporal encoding algorithms like BSA (Schrauwen & Van Campenhout, 2003), temporal contrast, GAGamma (Sengupta, Scott & Kasabov, 2015) are proposed and used in an application specific manner. The data encoding layer in NeuCube can also be observed as a data compression layer which has the unique property of compressing data in temporal domain by representing important events by spike-timings. In the temporal encoding scheme, the timings of the spikes are considered to be useful rather than the quantity of the spike. This is in contrast to the traditional data compression algorithms like auto-encoder and PCA, as the compression in the data is performed in a temporal dimension of the data. Sengupta and Kasabov (2017) discusses the temporal encoding by spike-time representation in the light of data compression and information theory, and compares the capabilities of different temporal encoding algorithms.

4.4.2 SNNc

The SNNc layer is claimed to be the most complex and novel component of this architecture. It is an unsupervised learning layer composed of a 3D grid of spatially arranged spiking and input neurons. Each neuron inside the grid has a spatial location and resides within a neighbourhood of other neurons. This grid is known as the spiking neural network cube (SNNc) in the NeuCube architecture. The purpose of the SNNc layer is to transform the compressed spike representation of input data into a higher-dimensional space through unsupervised learning ($g : \{0,1\}^{N \times T} \rightarrow \{0,1\}^{M \times T} | M >> N$) inside an SNNc, using a modified Hebbian spike-time dependent plasticity (STDP) learning (Song, Miller & Abbott, 2000). The purpose of learning in the SNNc is to dynamically update the synaptic strengths within the network to mimic spatio-temporal synchronicity in the data.

STDP Learning

STDP is a temporally asymmetric form of Hebbian learning induced by temporal correlations between the spikes generated by the pre and post-synaptic neurons. With STDP, repeated pre-synaptic spike arrival earlier than post-synaptic spike release leads to strengthening of the synapse known as long-term potentiation (LTP), and, in contrast, repeated spike arrival after post-synaptic spikes leads to weakening of the synapses known as long-term depression (LTD). The change of the synapse plotted as a function of the relative timing of pre- and post-synaptic action potentials is called the STDP function or learning window and varies between synapse types. I will formalise and discuss on the STDP learning rule and its variations as part of Chapter 5.

4.4.3 Readout

The readout layer is the last layer of the sequence in the NeuCube framework. This layer digests the spike-time data generated by the SNNc and maps it to relevant pattern labels (classification) or values (regression). KNN-based models (N. Kasabov et al., 2013) are the typical choice of supervised learning in almost all of the work done until now. However, the architecture is flexible enough to include other supervised learning methods that can perform spike pattern associations, such as SPAN (Mohemmed, Schliebs, Matsuda & Kasabov, 2012), SpikeProp (Schrauwen & Van Campenhout, 2004), ReSuMe (Ponulak & Kasiski, 2010) etc.

4.5 Departure of NeuCube from LSM

The NeuCube framework described in Section 4.4, in many respects, is inspired from the LSM architecture. In fact, the NeuCube framework can be considered as the next generation evolution of LSM. From Figure 4.6, it is quite clear that the second and the third layer of NeuCube architecture is the same as that of the LSM shown in Figure 4.4.

Prior to discussing the novelty that NeuCube provided in the last couple of layers, it must be realised that the encoding layer of NeuCube is unique to the framework. One major disadvantage of the applicability of LSM is its inability to perform computation in continuous valued data. The encoding layer of NeuCube addresses this disadvantage of LSM by adding the encoding layer. Useful encoding of high density data is of paramount importance for efficient and effective recognition of patterns in the spatio-temporal domain. The encoding layer typically aims to design algorithms to encode data following the temporal encoding paradigm discussed earlier. The objective of these encoding methods is to minimise the spike density while maximising the information retention in the encoded data. Realisation of such an objective ensures that the data encoder acts as a data compression machine. The data compression by encoding allows for: (1) Reduction of systematic noise in the data; and (2) Improvement in variability between patterns, and thus makes it much more recognisable.

The NeuCube SNNc shares numerous conceptual similarities to the LSM, primarily among which is the fading memory of the neurons, ability to transform otherwise non-linearly separable data to higher-dimensional space. Some of the limitations of LSM architecture are:

- In contrast to the contentions in (Natschläger, Maass & Markram, 2002), LSM alone is not sufficient to model human brain functionality.

- The effectiveness of the reservoir network in LSM is heavily dependent on a random selection of parameters.

- Due to the random connections within the reservoir, LSMs are spatially irrelevant and non-analysable. This makes the LSM behave very much like a black box.

The NeuCube SNNc seeks to address these issues through creating a meaningful structure of the network. The random connectome of LSM is replaced with a meaningful spatially organised connectome structure in NeuCube. This connectome is designed to physically encode a priori knowledge of the data being processed. This allows inspection of the evolution of clusters of the model over space and time. The spatio-temporal knowledge discovery and analysis through inspection resembles the visualisation and analysis capability of Kohonen's self organising maps (Kohonen, 1990). However, the information represented in the SNNc model is distinctly different from the Kohonen's map. Contrary to the static information representation in the synaptic strengths of the SOM, the SNNc can capture the spatio-temporal dynamics. This feature in conjunction with the common neural network analysis techniques allows knowledge extraction from the structure of the network. This means that general patterns, aberrant behaviours, and insights not otherwise comprehensible can be surfaced by analysis.

4.6 Software Design Architecture of the NeuCube Framework

Any theoretical and conceptual design of pattern recognition system is invariably followed by implementation and design considerations. The design considerations necessary for applying the generalised NeuCube architecture in an application specific manner is well elaborated in (Scott, 2015) and provides an useful insight on how to use *a priori* knowledge in designing the NeuCube architecture. The multi-modular and rather minimally-constrained description of the

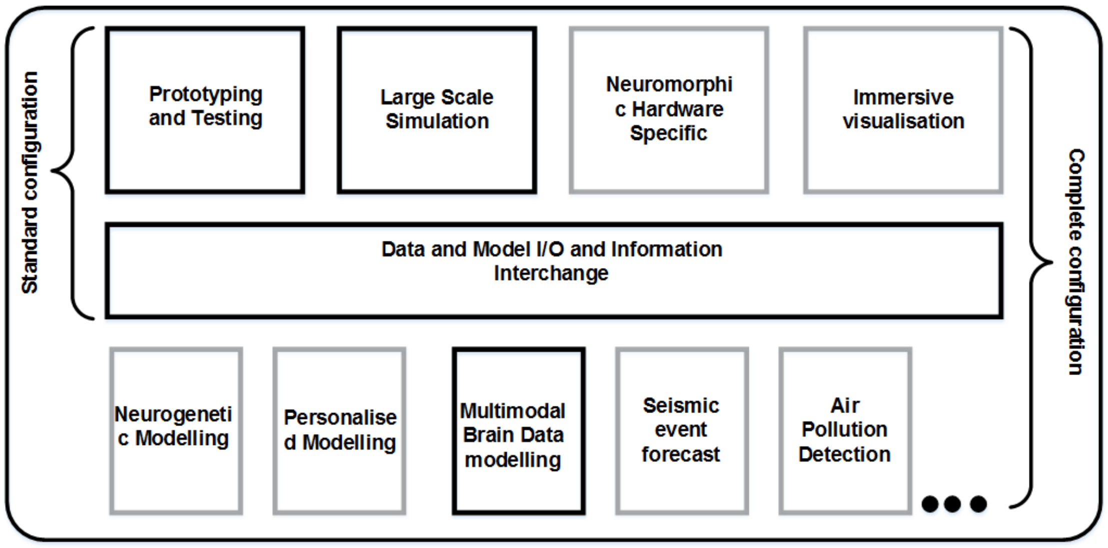


Figure 4.7.: Block diagram of the multi-modular NeuCube software architecture.

NeuCube framework allows high flexibility in plugging in variety of algorithms and innovations as part of the framework. This flexibility also poses a great challenge in defining a clear ground truth of singularity in NeuCube. The evolution of NeuCube from a tool to perform neuro-informatic data analysis to a general purpose pattern recognition machine has also constrained the software development process to be unified in a single direction. Figure 4.7 introduces a block level diagram of the NeuCube software implementation strategy. This thesis directly contributes towards the development of the blocks highlighted in black. This architecture uses the core pattern recognition framework described earlier as the core component, and wraps a set of functionalities around the framework. There are three configuration abstractions present in the implementation strategy:

- Basic configuration: The basic configuration includes a prototyping and testing module integrated with the I/O module. The main intention of this configuration is to use the NeuCube framework for general purpose pattern recognition, prototype model development and experimentation. Apart from model development capabilities, this module possesses additional integrated capabilities, such as dynamic visualisation, knowledge discovery and model analysis, and parameter optimisation modules. Due to the requirement of general purpose usage, this implementation is focused more towards usability and hence graphical user interface driven.

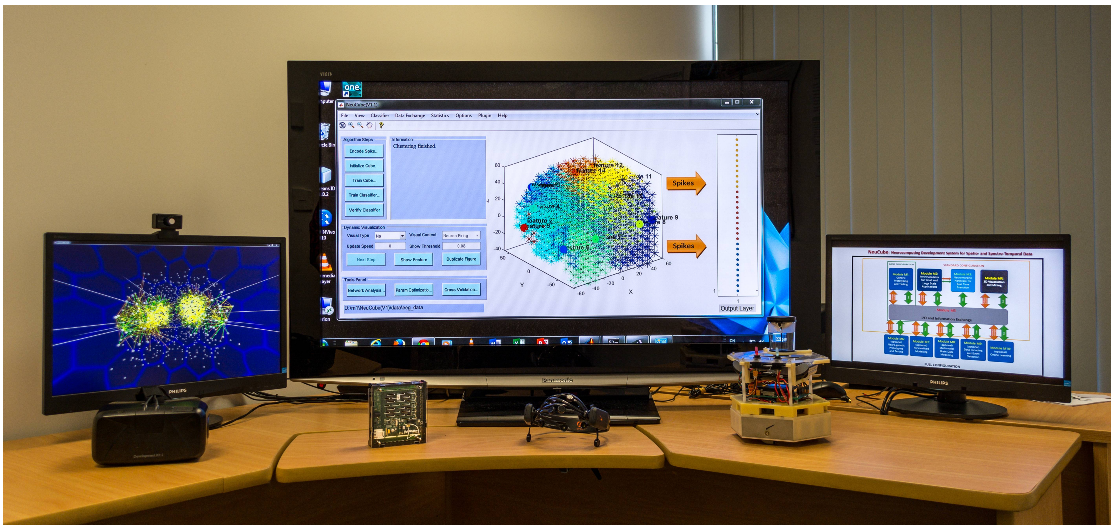
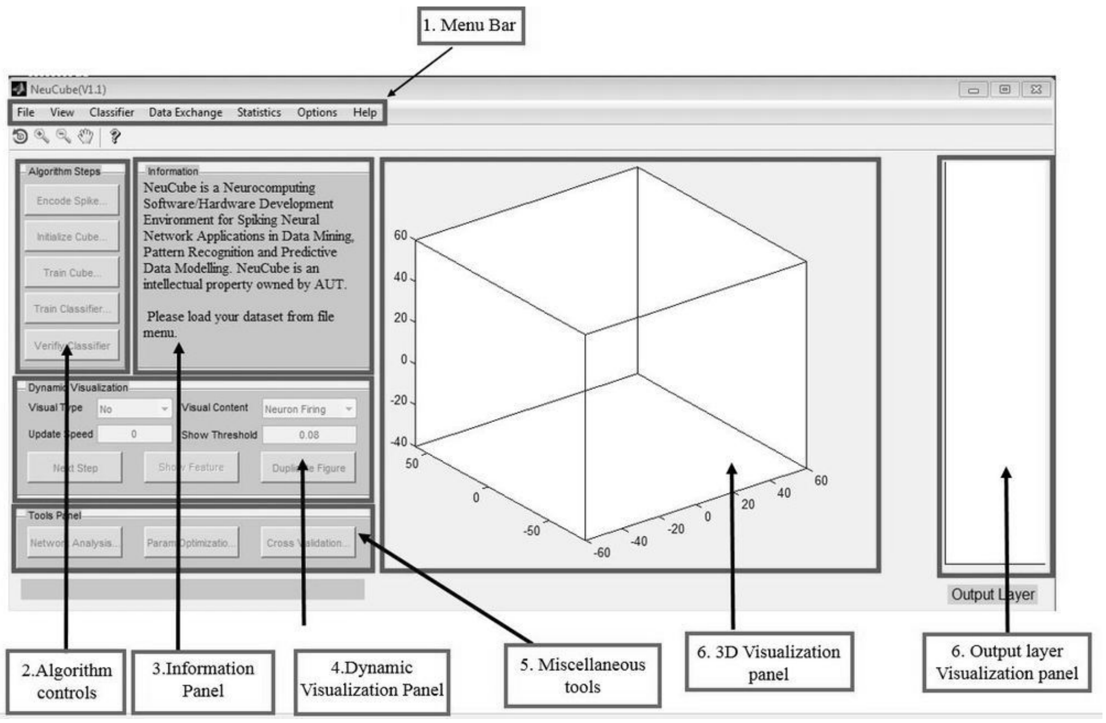


Figure 4.8.: Standard configuration of the NeuCube software architecture consisting of the prototyping and testing module, immersive visualisation module and the SpiNNaker neuromorphic hardware chip.

The basic configuration is developed using Matlab and is freely available (licensed) for public usage. I will describe the design of the prototyping and testing software later in this Chapter.

- Standard configuration: The standard configuration is an extension of the basic configuration and includes development of scalable software for large scale experiments, and general purpose and neuromorphic hardware implementations. Since the prime focus is on scalability, the software implementations for large scale simulation are API driven and much more consideration is given to efficiency of implementation rather than usability. Additionally, the standard configuration also includes immersive visualisation of the NeuCube model for deeper analysis and knowledge discovery. These implementations generally follow strict software patterns and uses higher level languages like Java and Python. The contribution of this thesis towards the large scale implementation and simulation is presented in Chapter 5.

- Full configuration: In addition to standard configuration, the full configuration includes all the application specific implementations of the NeuCube, such as personalised modelling, multi-modal brain data modelling, seismic event forecasting and so on. These application specific modules


NeuCube is a Neurocomputing Software/Hardware Development Environment for Spiking Neural Network Applications in Data Mining, Pattern Recognition and Predictive Data Modelling. NeuCube is an intellectual property owned by AUT.

Please load your dataset from file menu.

Figure 4.9.: NeuCube generic prototyping and testing software user interface and panel descriptions.

are developed for specific application purpose and extends the NeuCube architecture by adding application specific algorithms or functionality. This thesis contributes towards the development of methods and functionalities for multi-modal brain data. Chapters 6 and 7 are dedicated to this topic.

4.6.1 NeuCube for Generic Prototyping and Testing of Applications

The generic prototyping and testing tool is a GUI based Matlab implementation for rapid development of SNN application prototype systems for temporal or SSTD. The software is implemented as a set of continuous signal processing steps as shown in Figure 4.6. The user interface of NeuCube, as shown in Figure 4.9, is built as a set of functional components. An exhaustive description of the components can be found in the manual distributed along with the software. Here, I will briefly describe some of them:

Table 4.1.: Supported file format for descriptors.

Descriptor type	Mat	JSON	CSV
Dataset	✓	✓	✓
SNNc	✓	✓	✗
Parameter	✓	✓	✗
Result	✓	✗	✓

I/O and Information Exchange

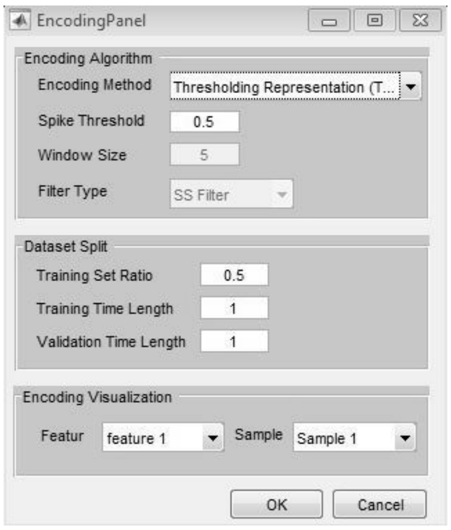
The information exchange component is used to import or export user defined information to and from the software, which includes temporal or SSTD data, NeuCube models, parameters and results. This module interacts with the external environment using four data descriptors. They are the following:

- Dataset descriptor: The Dataset descriptor consists of the data (and the meta data), that is to be learned and analysed. In the majority of cases, a dataset consists of a set of time series samples and the output label/value for the sample set. It is also possible to add miscellaneous information like feature name, encoding method and other meta information in the dataset.

- SNNc descriptor: The SNNc descriptor is essentially the model descriptor containing all information related to the structure and learning of the SNN. Some of the most important information stored in this descriptor are the spatial information of the input and reservoir neurons, structural information of the SNNc and the state of the SNNc during learning.

- Parameter descriptor: The parameter descriptor stores all the user defined parameters including hyperparameters of data encoding algorithms, the unsupervised learning algorithm and the supervised learning algorithm.

- Result descriptor: The result descriptor stores information about the experimental results produced by NeuCube.


EncodingPanel

Encoding Algorithm

Encoding Method: Thresholding Representation (T...

Spike Threshold: 0.5

Window Size: 5

Filter Type: SS Filter

Dataset Split

Training Set Ratio: 0.5

Training Time Length: 1

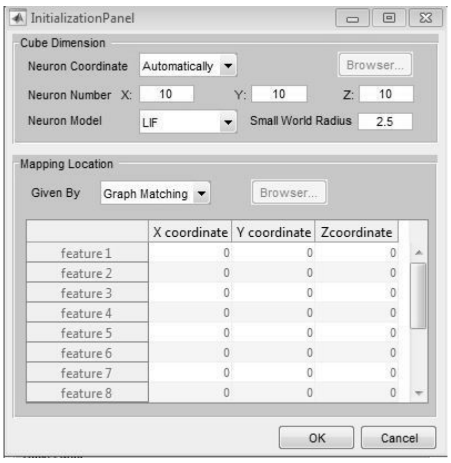
Validation Time Length: 1

Encoding Visualization

Featur: feature 1 Sample: Sample 1

OK Cancel

(a) data encoding.


InitializationPanel

Cube Dimension

Neuron Coordinate: Automatically Browser...

Neuron Number X: 10 Y: 10 Z: 10

Neuron Model: LIF Small World Radius: 2.5

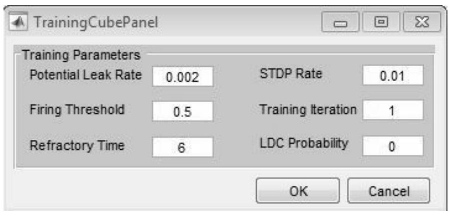
Mapping Location

Given By: Graph Matching Browser...

	X coordinate	Y coordinate	Zcoordinate
feature 1	0	0	0
feature 2	0	0	0
feature 3	0	0	0
feature 4	0	0	0
feature 5	0	0	0
feature 6	0	0	0
feature 7	0	0	0
feature 8	0	0	0

OK Cancel

(b) SNNcube initialisation.


TrainingCubePanel

Training Parameters

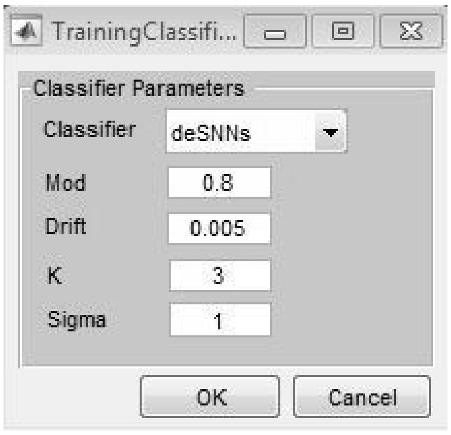
Potential Leak Rate: 0.002 STDP Rate: 0.01

Firing Threshold: 0.5 Training Iteration: 1

Refractory Time: 6 LDC Probability: 0

OK Cancel

(c) Unsupervised learning.


TrainingClassifi...

Classifier Parameters

Classifier: deSNNs

Mod: 0.8

Drift: 0.005

K: 3

Sigma: 1

OK Cancel

(d) Supervised learning.

Figure 4.10.: Algorithmic control UI panels.

This module supports three different file formats, Mat (binary), JSON (structured text) and CSV (comma separated plain text). Table 4.1 describes the supported file formats for each of the descriptor type. As a heuristic, the Mat format is recommended for achieving faster I/O. The CSV files are the recommended choice for import/export of dataset and results for later analysis. The JSON format is recommended for inter modular communication.

Algorithm Interactions

This module allow users to interact with the pattern recognition and signal processing algorithms via the algorithm controls panel, shown in Figure 4.9. The algorithm control panel includes a set of buttons for configuring and running the step by step process of data encoding, network initialisation, unsupervised learning (by training the SNNc) and supervised learning. The software uses a guided approach for performing the algorithmic steps by enabling or disabling buttons after every operation. Figures 4.10a, 4.10b, 4.10c and 4.10d shows the individual user interaction panels for encoding, initialisation, unsupervised and supervised learning respectively. Each panel allows users to choose from a set of algorithms and corresponding hyperparameters. For example, the data encoding panel (Figure 4.10a) that encodes the real-valued signal to spike trains also provides the option of choosing from a set of encoding algorithms and its hyperparameters from the drop down menu. The algorithmic control interacts with the visualisation panels, for visualisation and analysis of the data and the models.

Integrated Visualisation and Network Analysis

Visualisation and model analysis is an integral feature of this module. Due to NeuCube's ability to create an interpretable and analysable model, the visualisation and analysis features plays an important part. Visualisation capabilities includes: comparative display of real and encoded data; online dynamic visualisation of the SNNc learning; Visualisation of the SNNc model and the output readout layer.

Analysis of an SNNc network can be performed using the network analysis toolbox. This toolbox includes a bunch of graph analysis capabilities for measuring and visualising interactions between nodes and edges the SNNc at different levels. The network analysis consists of two major functionalities: (1) Neuron cluster analysis; and, (2) Information route analysis. An example of neuron cluster analysis is shown in Figure 4.11

Parameter optimisation Parameter optimisation is developed to allow users to search for the optimal set of hyperparameters for the NeuCube prototype system (model). The computational time for parameter optimisation depends on

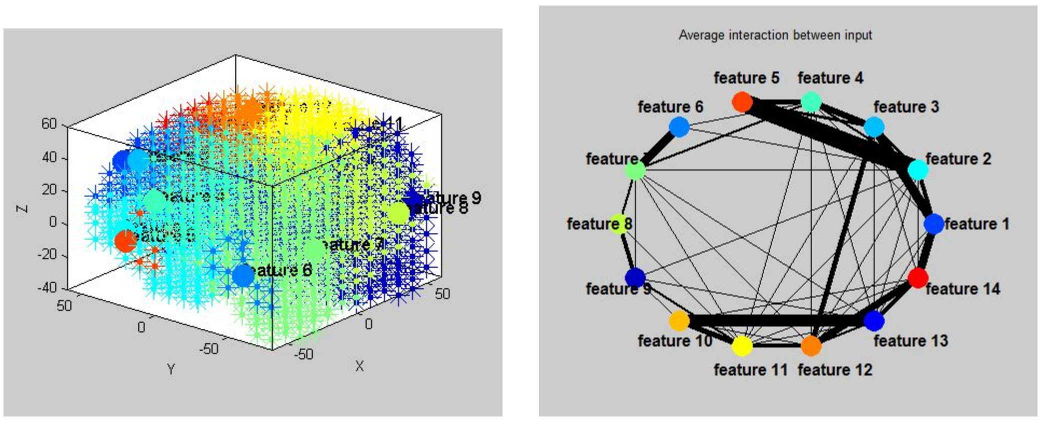


Figure 4.11.: Neuron cluster analysis by network analysis toolbox (A)Neuron clustering based on connection weights of the network (B)average one to one interaction between the input nodes. The thicker lines signifies more interaction.

the number of parameters to be optimised and the size of the NeuCube model. Parameter optimisation, can be performed using various methods, such as: Grid search; Genetic Algorithm; Differential Evolution; Quantum Inspired Evolutionary Algorithms, and PSO. The current release of NeuCube includes two methods (Grid search and Genetic algorithm).

4.6.2 NeuCube Based Spatio-temporal Data Research and Applications

Application of NeuCube in Brain Data Modelling

A number of recent studies have applied NeuCube architecture to brain data modelling, especially using ElectroEncephalography (EEG) data and functional Magnetic Resonance Images (fMRI) brain data. The spatio-temporal information contained in EEG and fMRI data poses challenges to standard statistical or machine learning techniques. Though, often standard machine learning techniques are used to process spatio-temporal brain data, they lack the ability: (1) to recognise differences in neurological dynamics that occur over the time; (2) identify the functional brain area involved; and, (3) to quantify the information involved. The RNN architecture of NeuCube, however, improves such capabilities (Taylor et al., 2014; Hu, Hou, Chen, Kasabov & Scott, 2014; N. Kasabov et al., 2013). In (Capecci, Kasabov & Wang, 2015) for example, NeuCube was used to study mental tasks using data collected by a 19-channel EEG recorder. This research showed the ability of NeuCube to classify and analyse changes in functional brain activities. This study was quite significant for identification of the appearance of mild cognitive impairment (MCI) and the staging of its degeneration toward Alzheimers Disease (AD). Recently, there has been a huge interest in using functional magnetic resonance imaging (fMRI) to understand, analyse and predict behaviour and cognition. The ability of fMRI to sample high resolution spatial information over time has been successfully used in correlating high resolution neural activity with behaviour. NeuCube has been used in several studies (Doborjeh, Capecci & Kasabov, 2014; N. K. Kasabov et al., 2017) involving fMRI data.

NeuCube for BCI

The feasibility of using NeuCube with EEG data to develop a functional electrical stimulation BCI/BMI system that is able to assist in the rehabilitation of complex upper limb movements was shown in (Taylor et al., 2014). In order to provide an effective tool for this purpose, a NeuCube model was trained on EEG data for a series of relatively complex muscle movements. The preliminary experiments suggest that NeuCube is much more efficient for this task than standard machine learning techniques, resulting in high recognition accuracy, a better adaptability to new data, and a better interpretation of the models, leading to a better understanding of the brain data and the processes that generated it.

NeuCube for Neuro-rehabilitation

Neuro-rehabilitation is another area of research NeuCube was applied as feasibility analysis study. Biomimetic learning and information processing time-scales of NeuCube provides appropriate technology for integration with mental tasks.

In addition, the fast and incremental learning offered by the framework is capable of adapting to the users' changing abilities as their rehabilitation progresses. Repetitive activities of daily living (ADL) and robotic active training are commonly practised in the rehabilitation of paralysed patients, both of which have been proven effective in the recovery of locomotor function in impaired limbs. Classification of ADL from EEG is of interest for the active robotic rehabilitation of patients with spinal cord injuries (SCI). This classification is a significant challenge with classical techniques, as these cannot manage the high noise, variability, and gradual change (due to the subject learning or rehabilitating the task) in the EEG signals effectively. Hu et al. (2014) performed an experiment using the NeuCube eSTDM to identify the upper-limb ADL of three classes with 14-channel EEG data. Classification accuracy using this technique is shown to be promising despite the highly noisy, low resolution EEG data (Hu et al., 2014). This experiment indicates strong potential for further exploration of the NeuCube for neuro-rehabilitation tasks.

NeuCube for SSTD Modelling in Other Applications

Several applications of NeuCube in the above areas are described in (N. Kasabov, Scott et al., 2016; Doborjeh et al., 2018), including: individual risk of stroke prediction based on personal, static data and temporal climate data; early experimental results on earthquake prediction; predicting establishment of harmful species in ecology; and stock index prediction.

4.7 Contributions and Publications

Contributions

1. *Introduction to recurrent neural networks and evolution of reservoir computing and liquid state machines.*
2. *Overview of the NeuCube evolving spatio-temporal data machine architecture.*
3. *Description of the Software design framework of NeuCube and the NeuCube prototyping and testing environment.*

Publications

1. Kasabov, N., Scott, N. M., Tu, E., Marks, S., **Sengupta, N.**, Capecci, E., ... & Espinosa-Ramos, J. I. (2016). Evolving spatio-temporal data machines based on the NeuCube neuromorphic framework: design methodology and selected applications. Neural Networks, 78, 1-14.
2. **Sengupta, N.**, Ramos, J.I.E., Tu, E., Marks, S., Scott, N.,... & Abbott, A. (2018). From von Neumann architecture and Atanasoffs ABC to Neuromorphic Computation and Kasabovs NeuCube: Principles and Implementations, Learning Sytems: From Theory to Practice Springer.

5 Data Structure Optimisation and Software Architecture Design for the Implementation of Large-Scale Spiking Neural Networks

5.1 Introduction

Chapters 3 and 4 have, on numerous occasions, discussed the importance of finding a trade-off between the biological plausibility and the computational complexity of computational models. The discussions have, especially, been focused on the spiking neuron. In this Chapter, the concentration will be on implementation level design considerations for simulating scalable spiking neural network architectures. The push towards big data analytics is driving the development of fast, scalable and real-time pattern recognition systems all over the world. Scalability of the machine learning framework plays a crucial role in realising the big data dream. Interestingly, this issue of scalability was seldom solved using actual scaling in machine learning, at least not in the big data sense. Part of the reason is certainly that multi-core processors did not yet exist on the scale they do today and that the idea of just scaling out was not as pervasive as it is today. Instead, 'scalable' machine learning is almost always based on finding more efficient algorithms, and most often, approximations to the original algorithm which can be computed much more efficiently. According to Vowpal Wabbit (Langford, Li & Strehl, 2007):

"There are two ways to have a fast learning algorithm: (a) start with a slow algorithm and speed it up, or (b) build an intrinsically fast learning algorithm. This project is about approach (b), and its reached a state where it may be useful to others as a platform for research and experimentation."

This Chapter will be taking the first approach and will use the second layer or the SNNc component of the NeuCube framework (Read Chapter 4 for overview) to drive the discussion. Before moving into the crux of this Chapter, a broad picture of the contrasting properties of a digital computer will be painted. There will also be discussions of a biological brain which poses significant challenges in integrating both in a single framework.

5.2 Brain vs. Digital Computers

Although, the brain-computer metaphor has served cognitive psychology well, research in cognitive neuroscience has revealed many important differences between brains and computers.

1. Memory and storage: In computers, data stored in memory is accessed by looking at a precise memory location. This is known as byte-address memory. On the contrary, the brain uses content-address memory, such that data can be addressed in memory through activation spread from closely related concepts.

2. Parallelism: The brain is naturally a massively parallel machine. Communication and synchronisations are automatically dealt with by the neuronal mesh of the brain. On the other hand, von Neumann architecture driven computer systems are modular and serial. Parallelism cannot be integrated into such an architecture intuitively.

3. Processing speed and system clock: The speed of neural information processing is subject to a variety of constraints, including the time for electro-

chemical signals to traverse axons and dendrites, axonal myelination, the diffusion time of neurotransmitters across the synaptic cleft, differences in synaptic efficacy, the coherence of neural firing, the current availability of neurotransmitters, and the prior history of neuronal firing. Although there are individual differences in processing speed, this does not reflect a monolithic or unitary construct, and certainly nothing as concrete as the speed of a microprocessor. Instead, psychometric processing speed probably indexes a heterogeneous combination of all the speed constraints mentioned above. Similarly, there does not appear to be any central clock in the brain, and there is debate as to how clock-like the brains time-keeping devices actually are.

4. Processing unit and memory interactions: Computers process information from memory using CPUs, and then write the results of that processing back to memory. No such distinction exists in the brain. As neurons process information they are also modifying their synapses which are themselves the substrate of memory.

5. Heterogeneity and size: The brain in its full maturity is a gigantic network of over 86 billion brain cells (or neurons) with approximately 1750 synapses per neuron. It is estimated that a truly biological model of the brain would have to include nearly 225 million billion interactions between cell types, neuro-transmitters, neuro-modulators, axonal branches and dendritic spines. This means layers of hierarchy and heterogeneity is present in the brain with complex interactions among them.

5.3 NeuCube SNNc

The second layer or the SNNc layer of the NeuCube (see Figure 4.6) is considered to be the most complex layer of the NeuCube architecture and it is computationally most expensive. For example, the NeuCube implementation distributed as part of the prototyping and testing module requires gigabyte order memory for running the unsupervised learning for an SNNc of size less than 1000 neurons. Hence, it is necessary to concentrate on this layer to improve scalability of the NeuCube system.

5.3.1 SNNc Architecture, Mapping and Initialisation Scheme

The SNNc architecture of NeuCube consists of a spatially arranged (in three dimensions) set of neurons (computational units), denoted by set M (with $|M|$ defining the cardinality of the set). The neurons are partially connected by recurrent synapses forming a directed incomplete and acyclic graph. The neurons and synapses form the vertices and the edges of the graph. The SNNc architecture is thus formalised as a directed graph $G = \{M, C, W\}$, consisting of the set of neurons M, the set of directed synaptic connections, C and the corresponding strengths or weights of the connections W. The network consists of two types of neuron:

- Input neurons: The input neurons, denoted by set $N \subset M$ (with $|N|$ defining the cardinality of the set), feed the input spike data (generated by the encoding layer) to the SNNc. These neurons do not have any activations and do not perform any computations. The input neurons form a similar level of abstraction as does the input layer in traditional neural networks. It is apparent that the input neurons do not have pre-synaptic connections, *i.e.*, a synapse can only originate from such a neuron.

- Spiking neurons: The spiking neurons, denoted by set $Q \subset M$ (with $|Q|$ defining the cardinality of the set), are the computational units, and are leaky integrate and fire (LIF) in nature. These neurons are responsible for performing computations on input data. The details of the neuron model is described later. These neurons can act both as post and pre-synaptic (connection) neurons, *i.e.* if one considers a pair of neurons connected by a directed synapse, the synapse can both originate and end at a spiking neuron.

The neurons in the SNNc are arranged spatially following the natural spatial arrangements in the data or by using automated mapping algorithms. Tu, Kasabov and Yang (2017) maps feature covariance in the data to Euclidean distance in the SNNc. The spatial arrangement of the neurons are typically done in two or three dimensional space. The synaptic connectivity of the SNNc graph is created using the small world connectivity (SWC) algorithm (N. Kasabov et al., 2014; Tu et al., 2014). The SWC algorithm connects a neuron to its spatial neighbourhood (controlled by the hyperparameter radial distance r_{swc}) of neurons.

5.3.2 Neuron Dynamics Model

The activation of the spiking neurons present in the SNNc is modelled by the spike response model (SRM), which is a kernel based simplified realisation of the LIF model. The SRM model generalises the differential equation based dynamics of the LIF model (see Equation 3.7) by replacing them with arbitrary kernels. SRM is a powerful computational framework for temporal integration with elegant mathematical formulation.

Figure 5.1 shows a typical example of a spiking neuron's configuration. A spiking neuron has a multi-input, multi-output configuration. A pair of neurons are connected by synapses. The synaptic strengths are represented by $\mathbf{w_i}$. A spiking neuron receives spike data at different time instances from the pre-synaptic neurons and emit spike data when sufficiently stimulated. The activation state of a spiking neuron i is described by the membrane potential v_i. In a non-stimulated state, the membrane potential is said to be in a resting state $v_{rest} = 0$. The SRM model in the present setup consists of multiple components:

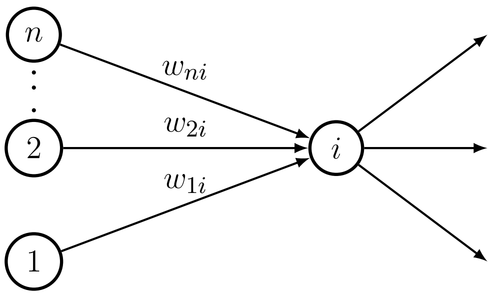


Figure 5.1.: A typical connectivity configuration of a spiking neuron i.

Post-synaptic Potential Kernel

Firing of pre-synaptic neuron j at time t_j^f, evokes a post-synaptic potential (PSP) in neuron i and is modelled by the kernel response function ϵ_0.

$$\epsilon_0 = \exp(-\frac{t - t_j^f}{\tau_m})\mathcal{H}(t - t_j^f) \tag{5.1}$$

where

$$\mathcal{H}(t - t_j^f) = \begin{cases} 1, & \text{if } t - t_j^f \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

The PSP kernel is a function of $t - t_j^f$, representing the PSP trace over time generated by the firing of neuron j over time. Figure 5.2 plots the PSP kernel as the function of $t - t^f$. τ_m (Equation 5.1) is the membrane constant which controls the decay rate of the PSP.

Temporal Integration of PSP Kernels and Conditions for Spike Emission

Under the SRM model, the PSPs evoked by the pre-synaptic neurons are temporally integrated to activate the spiking neuron. The overall contribution of the pre-synaptic spikes elicited by the pre-synaptic neurons j at any time t is given as Equation 5.3 describing the SRM model:

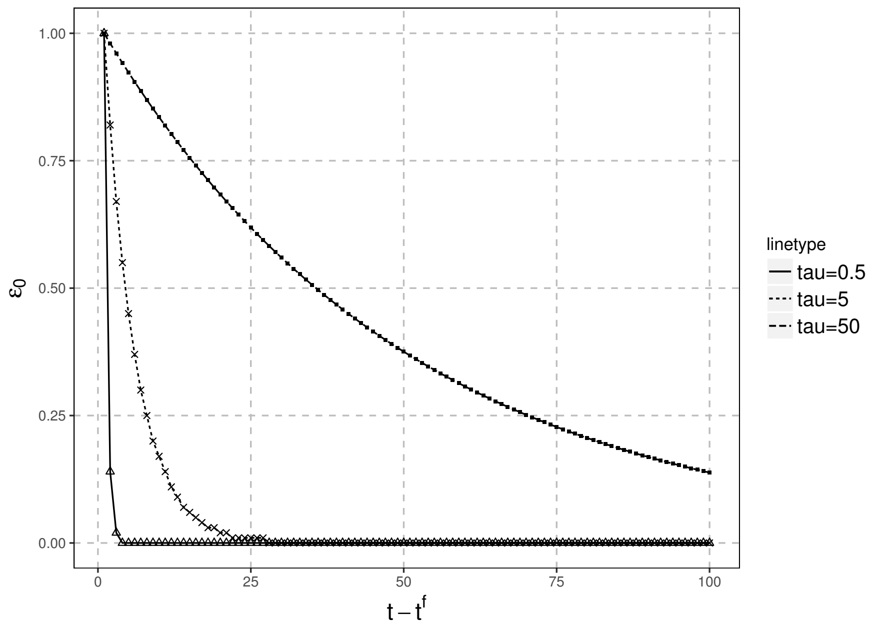


Figure 5.2.: Plot of the PSP trace ϵ_0 as a function of $t - t^f$. This figure plots the PSP simulation for different τ_m.

$$v_i(t) = v_{rest} + \sum_{j \in \mathrm{T}_i} w_{ji} \sum_{t_j^f \in F_j} \epsilon_0(t - t_j^f) \tag{5.3}$$

The temporal summation is a double summation operation. The inner sum adds up the PSP contributions due to the firings $t_j^f \in F_j$ of one pre-synaptic neuron. The outer sum adds up the PSP contributions of all the pre-synaptic neurons $j \in \mathcal{T}_i$ connected to neuron i. Equation 5.3 describes the membrane potential (activation state) v_i of a spiking neuron i can be calculated by adding the resting potential term and the temporal PSP sum. Each incoming spike perturbs the value of v_i and if, after the summation of the inputs, the membrane potential reaches the threshold v_{thr} then an output spike is generated. The firing time is given by the condition $v_i(t_i^f) >= v_{thr}$. After a neuron fires the neurons' membrane potential is reset to v_{rest}. In standard NeuCube implementations, the inner sum is generally set to 1. By setting the inner sum to 1, NeuCube only uses the information of present instance and forgets the influence of historical spike-trains, thus demonstrating minimal memory in the neuron model. However, any instance of the usage of NeuCube based architecture in this thesis uses the historical information with a preset hyperparameter controlling the memory horizon.

Refractory Period

After emitting the spike, a spiking neuron enters a period of quietness known as the refractory period. During this period, the membrane potential remains unaffected by incoming spikes. The refractory behaviour can be mathematically achieved by setting the membrane potential to a infinitely low value. In the SRM model the neuron behaviour under the influence of refractoriness depends only on the last firing moment leading to a short-term memory in the neuron. In the literature, the refractory period is described by absolute and relative refractory period. During the absolute refractory period, the neurons do not accumulate membrane potential and hence cannot fire. During the relative refractory period, it can be relatively difficult but not impossible to fire the neuron. In this current implementation, an absolute refractory period for the sake of simplicity has been used here. The absolute refractory period of a neuron is specified by the hyperparameter η_{thr}.

The modified SRM neuronal dynamics of NeuCube is formalised by Equation 5.4. The modifications of the canonical SRM model can be observed in: (1) the implementation of the PSP kernel which outputs a unit pulse at the time neuron i receives a spike from neuron j; (2) the implementation of refractoriness, where the membrane potential is set to negative infinity during the period η_{thr} after neuron i fires a spike at time t_i^f.

$$\begin{aligned} v_i(t) &= v_{rest} + \sum_{j \in \mathrm{T}_i} w_{ji} \sum_{t_j^f \in F_j} \epsilon_0(t - t_j^f) + \eta(t - t_i^f) \\ \epsilon_0(t - t_j^f) &= \begin{cases} 1, & \text{if } t - t_j^f = 0 \\ 0, & \text{otherwise} \end{cases} \\ \eta(t - t_i) &= \begin{cases} -\infty, & \text{if } t - t_i^f < \eta_{thr} \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{5.4}$$

Figure 5.3 shows a plot of three simulations of a spiking neuron for 200 discrete times with random spike inputs. Each simulation uses a preset v_{thr}. At the beginning of the simulation, the neuron is in a resting state $v_{t=0} = v_{rest}$. With the

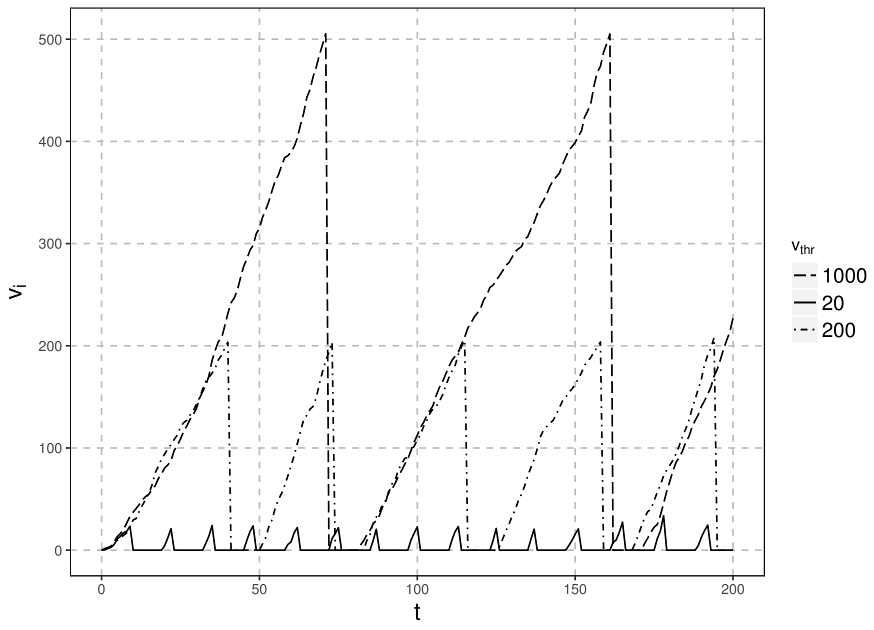


Figure 5.3.: Plot of the membrane potential traces(v_i) of a neuron i simulated over $T = 200$ time points using the SRM model. For the simulations, 3 predecessor neurons were connected to a spiking neuron. The spike data from the predecessor neurons are sampled randomly from uniform random distribution. The η_{thr} for the spiking neuron was set to 10. Each of the three v_i traces correspond to a preset v_{thr} mentioned in the label.

arrival of spikes, the membrane potential increases in a linear fashion and when sufficiently stimulated (sufficiency is determined by v_{thr}), the neuron spikes, and then goes back to the resting state. At this point, the neuron is said to be in a refractory state. The neuron stays in this state for a predetermined period η_{thr} and then goes back to a non-refractory state.

5.3.3 Unsupervised Weight Adaptation in SNNc

The unsupervised weight adaptation mechanism in the SNNc is an extremely important aspect of the dynamics. In a neural network paradigm, learning or plasticity is achieved through the synaptic strength updates of the network. The learning behaviour of the SNNc can be explained using the learning model of a single spiking neuron. Considering the single neuron architecture in Figure 5.1, the unsupervised learning problem is to formalise a scheme of updating the weights W of the network by $\Delta W(t)$ over the simulation time T. The NeuCube

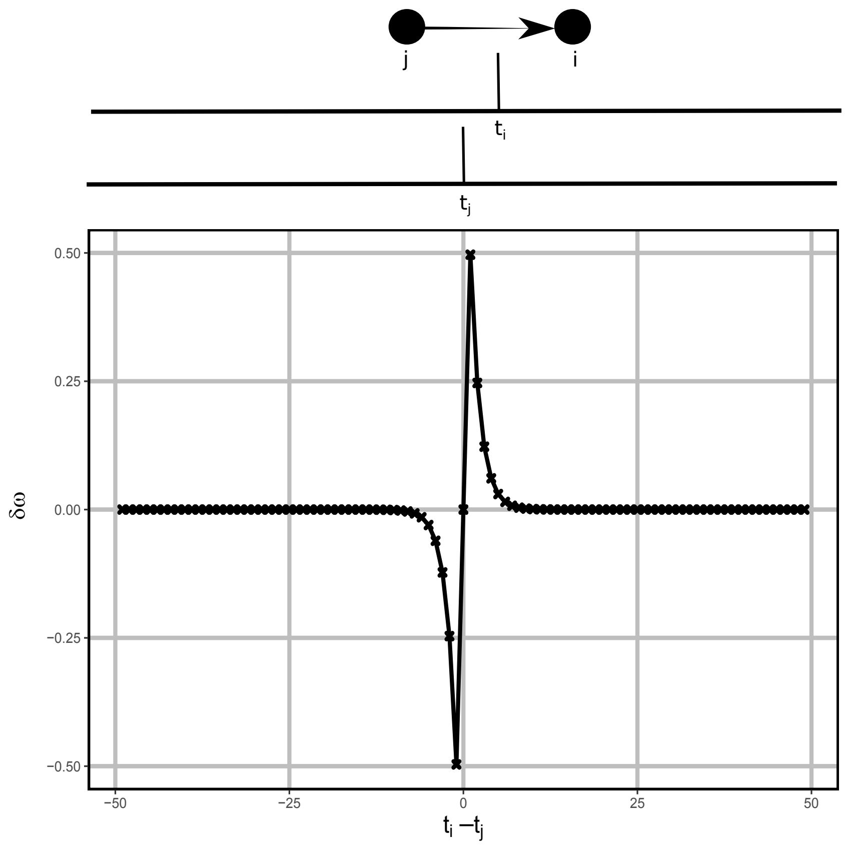


Figure 5.4.: Plot showing the functional dependence of the spike-time dependent plasticity learning rule. The STDP function shows the change of synaptic weights Δw as a function of difference in post and pre-synaptic spike-time difference.

SNNc has employed numerous variations of temporally asynchronous forms of Hebbian learning in different implementations.

STDP

STDP is a temporally asymmetric form of Hebbian learning induced by temporal correlations between the spike-timings of pre- and post-synaptic neurons (Song et al., 2000). As with other forms of synaptic plasticity, it is thought to underlie learning and memory in the brain, as well as the development and refinement of neuronal circuits during the development of the brain. With STDP, repeated pre-synaptic spike arrival, a few discrete times earlier to post-synaptic firing leads to Long Term Potentiation (LTP) of a synapse, whereas repeated spike arrival after post-synaptic spike generation leads to Long Term Depression (LTD) of the same synapse. The synaptic weight changes as a function the relative firing-time of the pre- and post-synaptic neurons, also known as the STDP learning window (Gerstner & Kistler, 2002). The overall significance of STDP lies in the ability of a spiking neuron to discriminate between, and then integrate, temporally significant inputs and transforming that to meaningful output, even though the actual meaning is not strictly known by the neuron (Markram et al., 2011). Networks that employ STDP operates as palimpsests, *i.e.* older stimuli are forgotten gradually to make room for new ones.

Gerstner, Kempter, van Hemmen and Wagner (1996), Song et al. (2000) formalised the mathematical model of STDP learning as per Equations 5.5 and 5.6. Symbols j and i are used to indicate pre- and post-synaptic neurons. In STDP learning, the dynamic change of weight Δw is estimated using a learning window function $W(\cdot)$. The learning window takes historical pre-synaptic firing times $\{t_j^1 \cdots t_j^f\}$ and post-synaptic firing times $\{t_i^1 \cdots t_i^g\}$ as input and calculates the LTP and LTD traces. These historical firing times are nothing but indices of a historical spike sequence. For example, a historical spike sequence $[01001011]$ can be rewritten as sequence of spike-time indices $t^f := \{1, 4, 6, 7\}$. Exponential decay functions are a popular choice for the learning window and Equation 5.6 is a good choice of the learning window function. The κ_+ and κ_- parameters control the maximum LTP and LTD update respectively and $\kappa_- = \kappa_+ = 1$ is a good choice to keep the bounds of weight update between $[-1, 1]$. From Equation 5.6, it can be observed that the polarity of $(t_i^g - t_j^f)$ defines the polarity of Δw_{ji}. This is Hebbian model of causal relationship where synapses are rewarded positively (strengthened) for causal firing (i fires later than j *i.e.* firing of i is caused by firing of j) and penalised (weakened) for non-causal firing. However, Equations 5.5 and 5.6 describes a batch update scheme and requires modification for on-line learning in the SNNc. Sjöström and Gerstner (2010a) proposed a modified on-line STDP update rule. In the on-line setting, Δw_{ji} is calculated every time neuron i fires a spike or receives a spike from neuron j. Equation 5.7 formalises the weight update rule for on-line mode. The first term in the right hand side of Equation 5.7 corresponds to the LTP update and is calculated when neuron i fires a spike at time t. The second term is the LTD update and is calculated when neuron i receives a spike from neuron j at time t. Both the batch and on-line formalisations of STDP learning are extended from (Sjöström & Gerstner, 2010b) which discusses the properties of the STDP learning model extensively. Figure 5.4 shows the plot of the STDP learning function where the Δw in quadrants I and III correspond to LTP and LTD respectively.

$$\Delta w_{ji} := \sum_f \sum_g W(t_i^g - t_j^f) \tag{5.5}$$

$$W(s) := \begin{cases} \kappa_+ \exp(-s) & \text{if } s > 0 \\ -\kappa_- exp(-s) & \text{if } s < 0 \end{cases} \tag{5.6}$$

$$\Delta w_{ji}(t) := \sum_f \kappa_+ \exp(-(t - t_j^f)) - \sum_g \kappa_- \exp(-(t - t_i^g)) \tag{5.7}$$

It is evident from the discussion above that the STDP learning rule enhances or depletes the synaptic strength of the connections, based on the relative coincidence of the spikes. This behaviour mimics the ability of the biological neurons to encode information by detecting the occurrence of temporally close but spatially distributed input signals and thus incorporating spatio-temporal information in the model.

Modified STDP in NeuCube

The standard NeuCube implementation uses a modified form of the STDP learning algorithm. The modification mainly relates to when and which neurons are updated. The contrast between STDP and modified STDP are:

1. As opposed to the STDP learning, in modified STDP learning, synaptic updates are only performed when a neuron i, fires a spike, and not when it receives a spike.

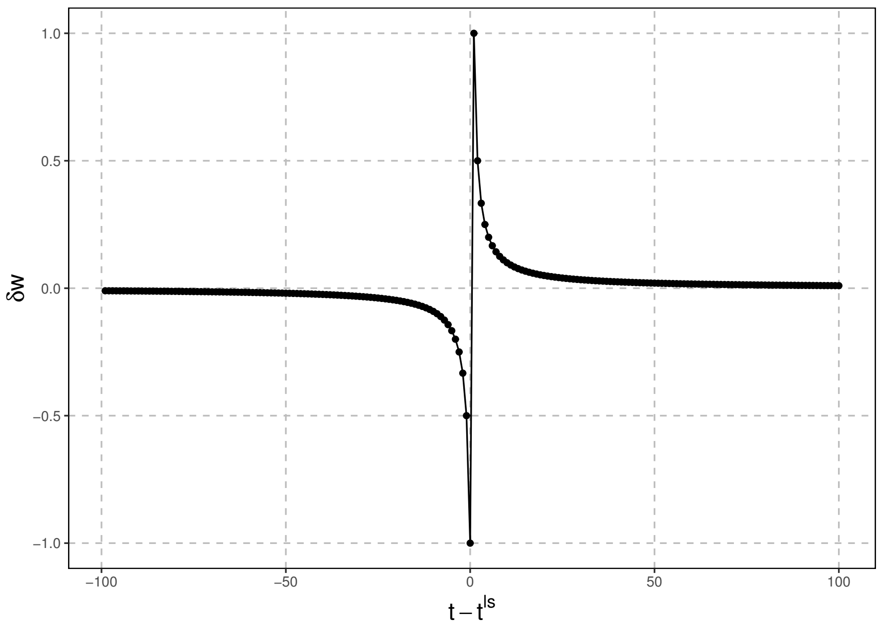


Figure 5.5.: Plot of modified spike-time dependent plasticity learning rule implemented in SNNc learning algorithm. The STDP function shows the change of synaptic weights Δw as a function of difference in last and present spike-time difference.

2. When a neuron i fires, both pre- and post-synaptic strengths are updated. The pre-synaptic connections are strengthened and post-synaptic connections are depleted.

The weight update rule can be formalised by Equations 5.8 and 5.9, where t^{ls} is the last spike time of a neuron and t is the current time instance. It is quite evident that the smaller the difference between t and t^{ls} the more significant the weight update is. This weight update rule also implies that enhanced importance is given to faster firing rate. κ_+ and κ_-, referred to as learning rate in the NeuCube literature, control the upper and lower bound of Δw similar to STDP. Figure 5.5 shows the functional dependence plot of the modified STDP learning. It is quite evident from the plot that the weight update curve is very similar to the STDP weight update rule. The difference, however, lies in the functional dependence as discussed.

$$\Delta w_{ik}(t) = \frac{\kappa_+}{t - t_j^{ls} + 1} \tag{5.8}$$

$$\Delta w_{ji}(t) = -\frac{\kappa_-}{t - t_k^{ls} + 1} \tag{5.9}$$

Algorithm 5.1 STDP based SNNc unsupervised learning algorithm

```
 1: input: $G = \{M, C, W\}$, $D_{seq} \in \{0,1\}^{|N|\times|T|}$, $\{hyperparameters :=$
    $v_{thr}, \eta_{thr}, \kappa\}$
 2: output: $O_{seq} \in \{0,1\}^{|M|\times|T|}$
 3: for all $t \in T$ do
 4:   initialise $C_{learn} \leftarrow \{\}$
 5:   for all $i \in Q$ do
 6:     find firing (at $t-1$) pre-synaptic neurons, $J_i^{spk(t-1)}$
 7:     set $C_i^{ltd} \leftarrow (J_i^{spk}, i)$
 8:     set $C_{learn}+ \leftarrow C_i^{ltd}$
 9:     simulate neuron $i$ as per Equation 5.4
10:     if $i$ fires then
11:       set $O_{seq}[i, t+1] \leftarrow 1$
12:       find pre-synaptic neurons, $J_i^{spk(t)}$
13:       set $C_i^{ltp} \leftarrow (J_i, i)$
14:       set $C_{learn}+ \leftarrow C_i^{ltp}$
15:     end if
16:   end for
17:   for $c_{ji} \in C_{learn}$ do
18:     calculate $\Delta w_{ji}(t) \leftarrow \sum_f \kappa_+ \exp(-(t - t_j^f)) - \sum_g \kappa_- \exp(-(t - t_i^g))$
19:     update $w_{ji}+ \leftarrow \Delta w_{ji}$
20:   end for
21: end for
```

5.3.4 Formal Description of SNNc Unsupervised Learning Algorithm

A classical implementation of the SNNc unsupervised learning algorithm is formally presented in Algorithm 5.2. This algorithm uses the modified STDP learning described in Section 5.3.3. The goal of the learning algorithm is to continuously input data sequence D_{seq} in the form of spikes and simulate the network G in a way that the synaptic strengths W of the network are updated over time T to learn poly-synchronous relationships across space and time. The hyperparameters for neuron simulation (v_{thr}, η_{thr}) and learning (κ) are also input into the learning algorithm. At the end of the simulation, the algorithm outputs the spike sequence O_{seq}. The simulation of G is performed at every time instance

Algorithm 5.2 Modified STDP based SNNc unsupervised learning algorithm

```
1: input: $G = \{M, C, W\}$, $D_{seq} \in \{0,1\}^{|N|\times|T|}$, $\{hyperparameters := v_{thr}, \eta_{thr}, \kappa\}$
2: output: $O_{seq} \in \{0,1\}^{|M|\times|T|}$
3: for all $t \in T$ do
4:    initialise $C_{learn} \leftarrow \{\}$
5:    for all $i \in Q$ do
6:       find firing (at $t-1$) pre-synaptic neurons, $J_i^{spk(t-1)}$
7:       simulate neuron $i$ as per Equation 5.4
8:       if $i$ fires then
9:          set $O_{seq}[i, t+1] \leftarrow 1$
10:         find pre- and post-synaptic neurons $J_i^{spk(t)}$ and $K_i^{spk(t)}$
11:         set $C_i^{ltp} \leftarrow (J_i^{spk(t)}, i)$ and $C_i^{ltd} \leftarrow (i, K_i^{spk})$
12:         set $C_{learn}+ \leftarrow C_i^{ltp}$ and $C_{learn}+ \leftarrow C_i^{ltd}$
13:      end if
14:   end for
15:   for $\{c_{ji}, c_{ik}\} \in C_{learn}$ do
16:      calculate $\Delta w_{ji} \leftarrow \frac{\kappa}{t-t_j^{ls}+1}$ and $\Delta w_{ik} \leftarrow -\frac{\kappa}{t-t_k^{ls}+1}$
17:      update $w_{ji}+ \leftarrow \Delta w_{ji}$ and $w_{ik}+ \leftarrow \Delta w_{ik}$
18:   end for
19: end for
```

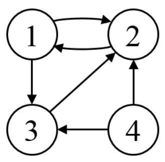
$t \in T$ and is described within the loop between line 3 and 19 in Algorithm 5.2. At every time instance, an empty variable C_{learn} is initialised, which stores over the subsequent steps, a subset of connection identities ($C_{learn} \subset C$) for the synaptic strength updates. The simulation is then performed in two subsequent phases. In the first phase (line 5 to line 14), all the spiking neurons Q are simulated based on the spiking neuron model. In NeuCube, the neuron simulations are done following Equation 5.4. In NeuCube, pre- ($J_i^{spk(t)}, i$) and post-synaptic ($i, K_i^{spk(t)}$) connections of the spiking neurons that fire at time instance t are candidates for weight evolution. These connections are stored in C_{learn} for update. The second step (line 15 to line 18) is the learning stage. During the learning stage, the connections C_{learn} are updated according to the learning rule, which in case of Algorithm 5.2 is the modified STDP learning rule. In addition, Algorithm 5.1 also presented the SNNc unsupervised learning algorithm using the canonical STDP learning rule described in Section 5.3.3. A careful comparison between the two algorithms reveal that the synaptic strength update in modified STDP is drastically different from canonical STDP in regards to when and what synapses are updated.

Considerations for Parallelisation

Neural networks are generally considered as a massively parallel problem, *i.e.*, the computations are simultaneous rather than sequential in layers of a typical neural network. Therefore, the divide and conquer type of parallelisation construct can be very easily achieved in neural networks by using the map and reduce paradigm of functional programming. However, by observing the characteristics of the SNNc learning algorithm, there does not seem to be a clear parallelisation approach. This is caused by the recurrence present as the output of the neurons in the SNNc layer, in the form of spike sequences which are recurrently fed back to other neurons creating a neuron level dependency over space and time. Therefore, although it is very tempting to merge the learning step and the neuron simulation step across the spiking neurons together, the asynchronous nature of the updates deems it an extremely hard parallelisation problem. The system clock driven computational simulation at the moment is clearly different from the clock precise parallelised scheme in the brain.

5.4 Analysis of the Data Structure Representations of SNNc

In this Section, focus of attention will be on the SNNc network structure representation G in light of Algorithm 5.2. The overall objective of the network structure representation analysis is guided by the objective to improve the computation and storage load of the algorithm as the unsupervised learning mechanism evolves the network over time. This work specifically looks into the storage and time complexity of the algorithm with increasing numbers of neurons in the network. During the iterative simulation process, the connections and the weights of the network are accessed very frequently. Parts of the SNNc network are accessed specifically in lines $6, 10-12$ and $15-17$. The access to the network is theoretically a search operation within the search space C for information on specific neuron identities. In particular, the algorithm needs to access the im-

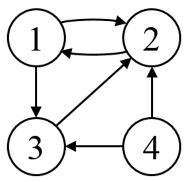


	1	2	3	4
1	0	1	1	0
2	1	0	0	0
3	0	1	0	0
4	0	1	1	0

Figure 5.6.: Example of the adjacency matrix representation.

mediate neighbours of a given neuron i. These are the pre- and post-synaptic neurons J_i^{spk} and K_i^{spk}. A data structure that is used for representing the network must, therefore, provide fast accessibility of neighbouring nodes. As an additional constraint, the present work focuses on the algorithmic implementation of a general purpose von-Neumann architecture computer (as opposed to the implementation on a neuromorphic hardware (Scott, 2015) setup), designed for commodity consumption. Therefore, storage space and computing capacity are of course constrained and an optimum data structure representation should be storage- and time-efficient. For the current experiments, a general purpose PC running a 64 bit Windows 7 enterprise operating system was used; one that had 16GB RAM, Intel Core i5 processor with 3.20 GHz clock speed. The implementations of Algorithm 5.2 was done in Matlab version R2014b.

In the Matlab based prototype and testing version of NeuCube, an adjacency matrix was used to store the connection structure C. According to graph theory, an adjacency matrix is defined as a square matrix C of order M (number of neurons) where 1 represents an existing edge between the two vertex indices. The edges correspond to the connections, and the vertices are the neuronal unique identifiers. In the present implementation, a second adjacency matrix is needed to store the weights W in order to avoid confusion between the existence of an edge and the weight values themselves. Figure 5.6 shows an example of an adjacency matrix. Since all values in an adjacency matrix can be directly accessed by using the corresponding neuron IDs as indices, this data structure is extremely fast. However, due to the nature of storing all relations between vertices, the storage had grown at a squared rate, and it also showed to be the most storage demanding option when compared to other data structures.



1	2	w_1
1	3	w_2
2	1	w_3
3	2	w_4
4	2	w_5
4	3	w_6

Figure 5.7.: Example of the edge-weight table representation.

The second option that was investigated was an edge-weight table. An edge-weight table stores the connection and their weights in a simple look-up table with each row containing a pair of neuron indices (identities) i and j, and the weight of the connection between them, as shown in Figure 5.7. This edge-weight table used far less storage than the adjacency matrix; however, it was also considerably slower, because in order to access a connection between a pair of neurons, the whole table had to be searched, which meant an average time complexity of $O(\frac{1}{2}c)$. Ordering the table by neuron i to use it as an index could to some extent alleviate the problem for finding post-synaptic neurons, but not for pre-synaptic neurons, since in that case only neuron j was given. Therefore, this data structure is sub-optimal in regards to computation time especially in case of larger networks.

The third data structure was an adjacency forward list. In graph theory, this is a list of vertices in which each entry contains a sub-list of neighbouring vertices. An example for an adjacency forward list is depicted in Figure 5.8. In terms of storage space, the adjacency list, like the edge-weight table, is considerably smaller than the adjacency matrix as it only stores the connections. Compared to the edge-weight table, the adjacency forward list saves space by listing all neurons connected to neuron i in an indexed list (the index being neuron i) instead of repeating the index for every connection. However, for Algorithm 5.2, a second adjacency forward list was needed to store the weight values of the connections, which is why this data structure uses slightly more storage space than the edge-weight table in the experiments. In terms of temporal performance, the adjacency forward list was also very similar to the edge-weight table in that it was faster to access post-synaptic connections due to indexing than looking up

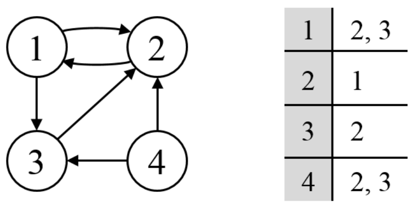
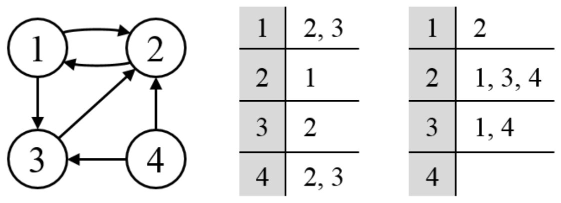


Figure 5.8.: Example of the adjacency forward list representation.

pre-synaptic connections where all sub-lists had to be searched. However, the indexing mechanism of the adjacency forward list provides a significantly faster look-up of post-synaptic connections than the edge-weight table. For these reasons, the adjacency forward list is overall expected to scale up best for a larger number of neurons, compared to the edge-weight table and the adjacency matrix.

Taking into consideration that the adjacency forward list is a very storage-efficient data structure and that its main bottleneck for temporal performance is the look-up of pre-synaptic indices, a decision was made to add a second adjacency list called an adjacency backward list to the structure to represent the connections from the opposite perspective, thus making the neuron at the end of the connection (neuron j) the index of this second list. A schema of this approach is shown in Figure 5.9. This backwards indexing mechanism caused a significant decrease of the algorithm's execution time, because it effectively reduced the time complexity of the data structure to $O(1)$ for finding the right pre-synaptic and post-synaptic indices. In comparison with the other data structures, the adjacency forward-backward list was now the best alternative for representing the SNNc network. Table 5.1 gives a comparative overview of the theoretical complexities in regards to time and storage with the different data structures discussed here. The complexities are measured by n and c referring to the neuron count and connection count respectively. The relationship between n and c can be represented by $c = \alpha \times n^2$, where $\alpha \in [0, 1]$ is the degree of sparseness. All of the present experiments have used $\alpha = 0.02$.

The findings from the theoretical analysis of the structures' complexity could be verified through the present experimental results. For the experiments, a bench-



1	2, 3
2	1
3	2
4	2, 3

1	2
2	1, 3, 4
3	1, 4
4	

Figure 5.9.: Example of the adjacency forward-backward list representation.

Table 5.1.: Comparison of time and storage complexity for different data structures.

data structure	connection type	time			storage
		worst	average	best	
adjacency matrix	all	$O(1)$			$S(n^2)$
edge-weight table	all	$O(c)$	$O(\frac{1}{2}c)$	$O(1)$	$S(c)$
adjacency forward list	pre-synaptic	$O(c)$	$O(\frac{1}{2}c)$	$O(1)$	$S(2 \times c)$
	post-synaptic	$O(1)$			
adjacency forward-backward list	all	$O(1)$			$S(3 \times c)$

mark EEG dataset was used. The dataset consisted of EEG data measuring brain signals during a task of wrist movement. The wrist movements were categorised into upward, downward, and central directions. This task was performed on a single subject and EEG data was sampled from 14 channels at a sampling rate of 128 Hz while the subject performed the task. 20 independent trials of one second duration were collected while the subject performed each movement task. 14 of the 1485 neurons in the reservoir were randomly chosen as input neurons for the EEG channels. A network with a highly sparse connectivity ($\approx 2\%$ of all possible connections) was initialised randomly for every experiment. The value of two per cent was chosen because this was the average amount of connections used typically during experiments.

Figure 5.10 shows the comparison of storage required in megabytes by each of the data structures. The graph itself is dependent on the number of neurons in the SNNc (between 0 and $70,000$ for the storage, and between 500 and $3,500$ for the execution time). When it was decided that the number of neurons should be increased further, all values above $80,000$ neurons for the adjacency matrix and above $120,000$ neurons for the sparse matrix had to be excluded, due to the above discussed technical restrictions in the experimental setup. In addition,

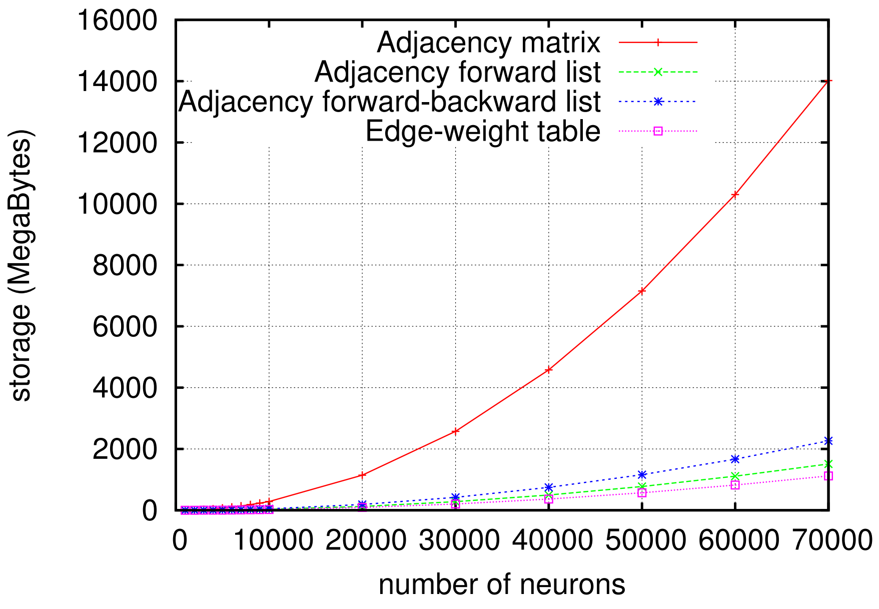


Figure 5.10.: Storage space comparison of different data structures with increasing number of neurons.

it became difficult to distinguish between the curves, especially in the lower regions of the graph.

It is clearly visible that the adjacency matrix has the steepest increase of storage space needed. The other three data structures are relatively similar in their development, with the edge-weight table growing slowest. It was interesting to see that the curve of the sparse matrix was very close to the one of the adjacency list, which indicates that the internal representation of a sparse matrix in Matlab is similar to an adjacency list. Figure 5.11 shows the results of comparing the execution time of the data structures. For the comparison of execution times, the curve of the edge-weight table showed its disadvantage as it increases near exponentially. Thus, another graph representation without the edge-weight table was created. This showed that the adjacency list was considerably slower than the matrices and the adjacency backwards list.

These experimental results verify the previous theoretical analysis. The most promising data structure for representing an SNNc network having a large number of neurons in the NeuCube SNN is the adjacency list with backwards indexing due to its constant complexity in terms of execution time of the algorithm, and linear storage growth with increasing connection density.

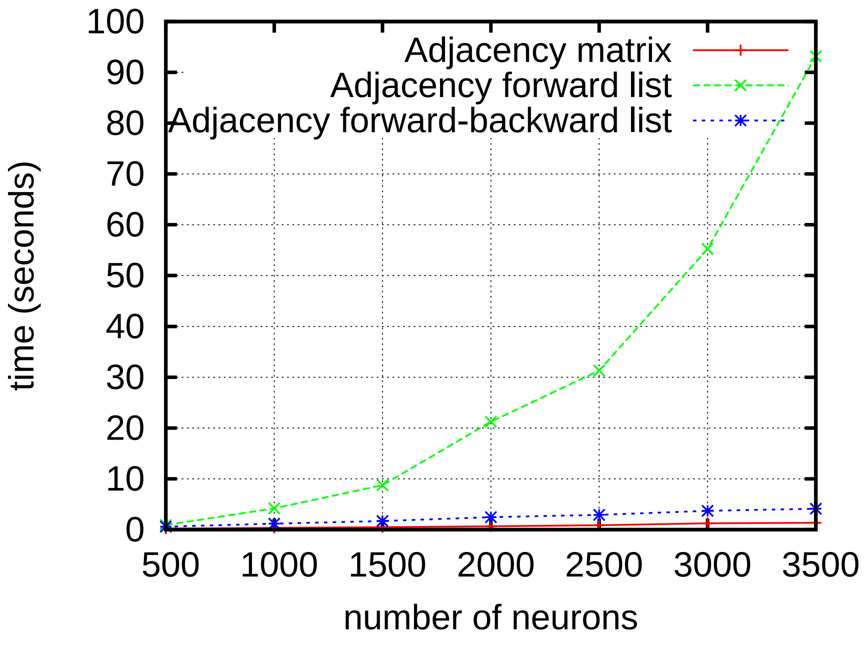


Figure 5.11.: Execution time comparison of different data structures with increasing number of neurons.

5.4.1 Large-scale Unsupervised Learning of SNNc Using the Adjacency Forward-backward List

As a conclusion of the current experimental results, it seemed evident that the adjacency forward-backward list data structure representation is the most economic in regards to storage and temporal complexity. To further demonstrate this, the NeuCube SNNc learning algorithm in Matlab capable of using the ad-

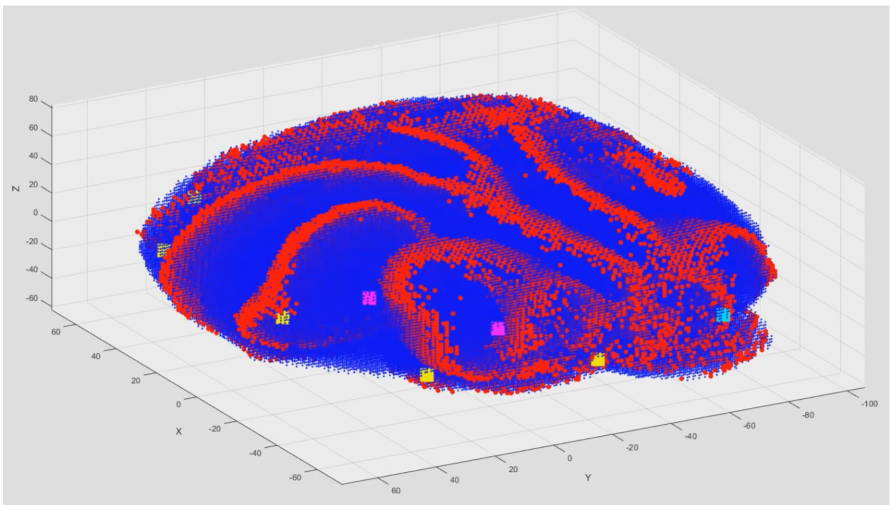


Figure 5.12.: Snapshot of the spike activity at a time instant inside the SNNc.

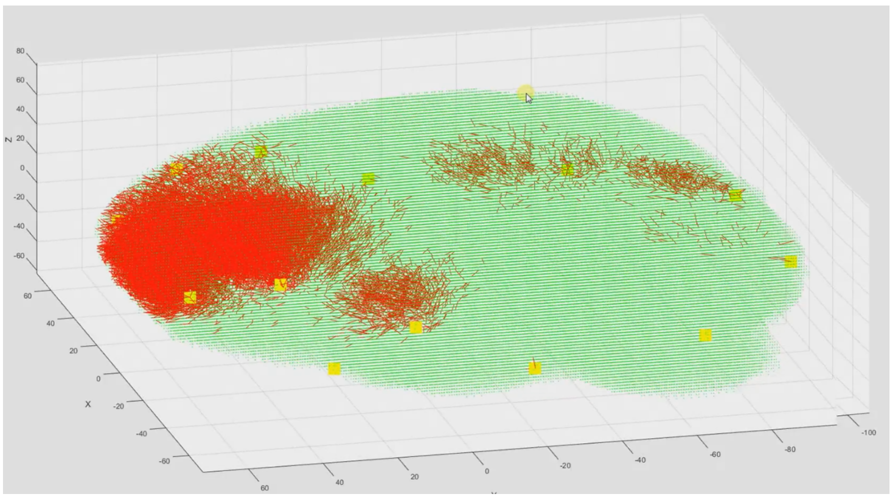


Figure 5.13.: Strongest connections learned by the SNNc at the end of SNNc simulation.

jacency forward-backward list data structure was implemented (see Appendix C for the Matlab programme). In order to test the efficiency of the implementation on a much larger scale, Algorithm 5.2 was run on a SNNc consisting of $241,606$ neurons taking the same EEG input data described earlier. The SNNc was simulated to mimic a brain with neural cells in the order of 10^6 and connections in the order of 10^{10}. The spatial coordinates of the neurons were obtained from the xjView (Cui, Li & Song, 2011) software and resembled the spatial distribution of the human brain based on the MNI coordinate system. Figure 5.12 shows a snapshot of spiking patterns at a certain time instance during the unsupervised learning simulation. The ripple like behaviour of the liquid state within the SNNc can be clearly observed in the network. The ripple effect in the SNNc is of course clearly visible in a dynamic simulation environment. Figure 5.13 shows the strongest connections formed in the brain-like network as a result of the unsupervised learning.

5.5 Considerations for Modularity and Heterogeneity: Towards Graph Based Software Design of SNNc

In the last Section, the considerations of scalability in regards to the size of the SNNc was discussed. At this point, it is critical to mention that apart from the large size, biological brains are considerably modular and heterogeneous. The theory of modularity suggests that there are functionally specialised regions in the brain that are domain specific for different cognitive processes. The brain is often represented as a network of interconnected, dynamically interacting elements. Cognitive processes are thought to result from the integration of neuronal processing distributed across these complex networks at different temporal and spatial scales. Several graph theory based methods (Sporns & Betzel, 2016; Nicolini & Bifone, 2016) have been proposed in analysing the modularity and heterogeneity in the brain.

The idea of modularity and heterogeneity is of major importance for the neuromorphic inspirations of the SNNc architecture design. Heterogeneity and modularity is observed not only in the brain but is quite prevalent in a broad range of networks, such as groups in social networks, ensembles of interacting proteins or coregulated genes in cellular network. These clusters or groups of items have homogeneous property behaviours within the group, and vary considerably between the groups across the network.

The objective of this Section was to design algorithmic or implementation improvements to facilitate heterogeneity and modularity in the SNNc. Let us revisit the implementation of unsupervised learning algorithms (see Appendix C). The functional style of implementation constrains the ability to inject heterogeneity into the network, especially if one considers designing networks with varieties of neurons, synapses, learning behaviour etc.

The basic concept behind the template method design pattern is relatively simple. Generally abstract classes are created representing necessary steps for a general algorithm operation. An instantiation of the template (class) then implements these steps with necessary extension. In Algorithm 5.2, the network G is represented by the tuple $< M, C, W >$. M is a list of neuron identifiers and serves no purpose other than storing the identifiers. The code for neuron simulation is integrated within the network simulation programme and is independently treated compared to M. Additionally, weights are modelled independently of the connections in Algorithm 5.2.

In order to overcome these shortcomings, the SNNc network in the graph based design is constructed as a directed graph data structure. The graph based architecture that was designed is summarised in the UML class diagram shown in Figure 5.14. The graph $G =< V, E >$ is made of vertices $V = \{v_1, v_2, \cdots, v_m\}$ and edges $E = \{e_1, e_2, \cdots, e_c\}$. The class SNNc is designed at the network level of abstraction, where behaviours and operations are performed on the whole or part of the network. Some operations on the network are $initialiseNetwork()$: used to initialise the SNNc network. Several algorithms can be implemented as part of the SNNc class; $learnNetwork()$: Method to perform unsupervised learning on the SNNc network. This method should be used to handle the synchronisation and broadcasting of the data in the form of spikes across the network; $visualiseNetwork()$: is used to visualise the dynamic and static states of the network. The output of this method are similar to the ones shown in Figures 5.13 and 5.12. The vertices and edges forming the SNNc graph are designed to be modelled individually. Personalised models of vertices and edges ensures the flexibility of implementing varying degrees of heterogeneity in the network through encapsulation and polymorphism of the objects.

A vertex in graph G is modelled as an object with the following properties:

1. ID: Stores the unique identification of a vertex.

2. Location: Stores the spatial location of the vertex in the 2D/3D space.

3. Neuron: Refers to an instance of a suitable neuron model. The Neuron object is modelled as an instantiation of the $GenericNeuron$ class which can morph into either input or spiking neurons. A couple of implementations of spiking neuron models that inherits from the $GenericNeuron$ are shown as $LIFNeuron$ and $IzhikevichNeuron$. Each of the spiking neurons are initialised by the $init()$ method and simulated over time using the $simulate()$ method. It is evident that the realisation of any neuron models or even newer behaviours in the existing neuron models can be achieved without much effort through this modular design.

An edge in graph G consists of the following properties:

1. ID: Stores the unique identification of an edge.

2. fromVertexID: Stores the source vertex ID of the edge.

3. toVertexID: Stores the destination vertex ID of the edge.

4. synapse: Refers to an instance of a suitable synapse model implementation. An example synapse model is described in class $Synapse$. The primary behaviour of the synapse is controlled $updateSynapse()$ method. This method modifies the synaptic strength (weight) using learning rules implemented as static methods.

Overall to inject modularity and variety in the SNNc, the architecture has been designed in hierarchical layers of abstraction in a top-down manner. At the highest layer of abstraction, the SNNc network has been designed only at network level, keeping the vertex and edge level implementations abstract. In the next layer of hierarchy, the individual vertices and edges are modelled and drills down further into individual neuron models, learning behaviour and so on. Implementing the architecture this way also allows a user to configure the SNNc at varying degrees of generality.

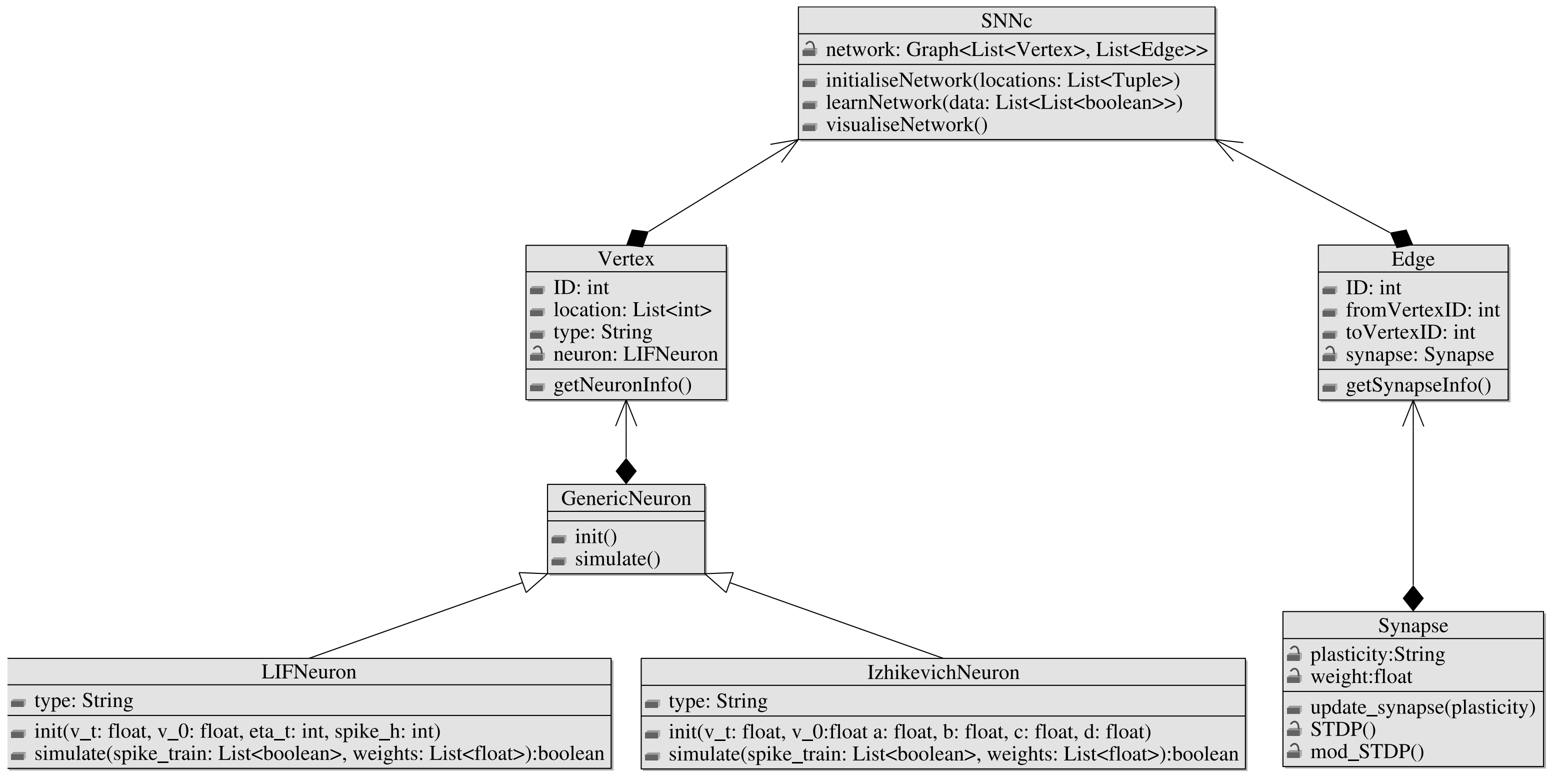

SNNc
network: Graph<List<Vertex>, List<Edge>>
initialiseNetwork(locations: List<Tuple>)
learnNetwork(data: List<List<boolean>>)
visualiseNetwork()
Vertex
ID: int
location: List<int>
type: String
neuron: LIFNeuron
getNeuronInfo()
Edge
ID: int
fromVertexID: int
toVertexID: int
synapse: Synapse
getSynapseInfo()
GenericNeuron
init()
simulate()
Synapse
plasticity:String
weight:float
update_synapse(plasticity)
STDP()
mod_STDP()
LIFNeuron
type: String
init(v_t: float, v_0: float, eta_t: int, spike_h: int)
simulate(spike_train: List<boolean>, weights: List<float>):boolean
IzhikevichNeuron
type: String
init(v_t: float, v_0:float a: float, b: float, c: float, d: float)
simulate(spike_train: List<boolean>, weights: List<float>):boolean

Figure 5.14.: UML Class diagram of the template based object-oriented architecture of the SNNc. This architecture should be considered an example architecture for incorporating modularity. A python implementation of the SNNc following this architecture is presented in Appendix C.

5.6 Summary and Conclusion

In this Chapter, the discussions have primarily revolved around the implementation aspects of the SNNc. This Chapter began by contemplating the discrepancy between a biological brain and a computer. The discrepancies formed the basis of the challenges that arises in developing human brain-like computation algorithms in computers. Then, there was a presentation of an in-depth formalisation, discussion and analysis of the several components of the SNNc architecture and unsupervised learning algorithms in NeuCube. This discussion paves the pathway for analysing the data structure representations of the SNNc network with respect to scalability and heterogeneity. During this discussion, it was shown how the network representation plays a decisive role in large scale implementations, especially balancing the storage and execution time. The importance of modularity and heterogeneity in SNNc was further discussed as well as a proposal put forward for a hierarchical template pattern-based software architecture for realising such a goal.

5.7 Contributions and Publications

Contributions

1. *There were qualitative comparisons of the contrasting characteristics of a biological brain and a computer.*
2. *In-depth descriptions about the SNNc layer of the NeuCube architecture were made. This included SNNc network initialisation, simulation and learning with formalisation of the unsupervised learning algorithm.*
3. *An analyses of the SNNc network storage architecture and data structure was presented in the light of large scale simulation of the SNNc.*
4. *A template pattern based software design architecture of SNNc to incorporate modularity and heterogeneity in the SNNc was proposed and implemented.*

Publications

1. Abbott, A., **Sengupta, N.**, & Kasabov, N. (2016, July). Which method to use for optimal structure and function representation of large spiking neural networks: A case study on the NeuCube architecture. In Neural Networks (IJCNN), 2016 International Joint Conference on (pp. 1367-1372). IEEE.

6

A Novel *a priori* Knowledge Driven Temporal Encoding Framework for Compressing and Recognising Pattern on Temporal Data

6.1 Introduction

Traditionally, data analytics, especially, predictive analytics and machine learning have focused on computer intensive approaches that are intended to be applied directly to the raw data present in the continuous space (Bishop, 2006). These methods aim to take advantage of the flexibility entailed by continuous mathematics (uncountable sets) to build complex learning theories that can perform pattern recognition in data. However, with the rise of big data, the machine learning technologies are dealing with new challenges with respect to real time processing of massive volumes of data. Although the machine learning community has continuously striven to learn from massive volumes of data, the development has been grounded on the assumption that the computation fits into the memory seamlessly. In contrast, the current data size has grown to such a scale that the data are becoming harder to store.

6.2 Data Compression and Inspirations from Neural Coding

In this Chapter, the focus will be on the inspirations from the human brain that allows us to see the problem of data processing and predictive analytics problem under the big data environment from an alternate perspective. It can be observed that the continuous incoming stimuli in various forms and frequencies processed by the human brain can indeed be characterised by the properties of big data, *i.e.*, volume, variety and velocity. The human brain is considered to be the most resourceful and efficient system which can recognise distinct patterns in the streaming continuous stimuli (volume) captured by multiple sensory organs (variety) in millisecond resolution (velocity). It is also observed that human brain cells, when presented with external stimuli, propagates the signal economically, over long distances using electrical impulses or spikes via the synaptic action potentials. Hence, it is imperative that there exists an efficient system, which converts the massive volume of continuous signal to discrete events or spikes. In neurobiology, the process of such analog to digital signal transformation is known as neural encoding (Brown, Kass & Mitra, 2004). It is intriguing that the process of neural encoding not only converts the big streaming continuous data space into a compressed space of spikes, but the brain cells also recognise patterns in such a compressed space. The biological organisation of the brain tends to create signals with a very specific class of distributions, and it is from the perspective of evolutionary understanding that these distributions are optimised for fast analysis. The most popular hypothesis states that the signal strengths are encoded by the mean firing rate, *i.e.* stronger input signal produces larger volumes of neuronal firing on an average in the brain. A range of studies (Mainen & Sejnowski, 1995; Maunsell & Gibson, 1992) across multiple species in the sensory and motor-neuronal system supports the validity of the mean firing rate hypothesis. A major drawback of this theory, however, lies in the association of information density with spike density. Determining the spike density in millisecond resolution from a large volume of spikes leads to a level of computational inefficiency. As per an alternate theory on neural encoding, neurons carry information in the precise timing of the spikes. This is known as the temporal encoding or spike-time encoding. Numerous research (Gollisch & Meister, 2008; Hallock & Di Lorenzo, 2006) has shown the presence of temporal encoding in different parts of the human brain. Temporal encoding supports the efficient representation of information that is required for very fast processing (in millisecond scale) of the stimulus presented to the human brain. As opposed to the rate coding scheme, high fluctuations in mean firing rate, also known as inter-spike interval (ISI) probability distribution is considered to be informative rather than noise in this scheme. The temporal spike-time representation of the data acts as a lossy compression of information. Most forms of learning, though, could be seen as forms of data compression. In fact, one can, in terms of pattern recognition, only learn something from data when there is redundancy in the data. In many data analysis tasks, the data is preprocessed or recoded in a way that could be seen as a form of data compression. If such preprocessing does not destroy the patterns of interest, it results in a comparative performance of the learning algorithms. The motivation of the temporal encoding, thus, in this context is to reduce large volumes of data into a compressed state with minimal loss and the maximal presence of discriminable information. Examples of data sources where such encoding is useful are high-frequency streaming data, such as the pulsar data in radio astronomy, and seismic activity data.

6.2.1 Information Theory and Data Compression

From the viewpoint of computational theory, the data encoding problem relates to the concepts of information theory. In the seminal work on information theory, Shannon (1948) proposed a mathematically complete theory to quantify transmission of information in a communication channel. A conclusive finding that the amount of information in any object can be estimated as the description length of the object continues to set the stage for the development of communications and data processing. Shannon's information theory is built on a presupposition that the computable information in an object is the characteristic of a random source with known probability distribution of which the object is a part. To realise this idea, Shannon derived the ‘entropy’ from the first principle of the theory, which is the measure of average information emitted by an object when observed. It can be described as the functional mapping of the random variable to a real number. Kolmogorov (1965), later proposed an alternate and more generalisable notion of information measurement known as algorithmic information theory. Contrary to Shannon’s theory, Kolmogorov’s theory of complexity (Kolmogorov, 1965; Chaitin, 1966) considers information as the property of an object in isolation, irrespective of the way the object came into existence (Grunwald & Vitányi, 2004). It describes information as the minimum number of bits from which a message or a file can effectively be reconstructed, *i.e.* the minimum number of bits suffice to store a reproducible file. A computational neuron responsible for emitting spikes from sensory data can be regarded as a logical transmission medium responsible for broadcasting continuous information received from the data source. The two neural coding hypotheses hence can be seen and described in the light of information theory. It can be observed that the rate coding scheme adheres to Shannon’s interpretation of encoding. The inherent assumption of the presence of a random source with a known probability distribution in Shannon’s theory is much apposite to the mean firing rate as it relates to the frequency of spikes over time. However, the interest in efficient compression of a large volume of data by a sequence of spike-timings and using it for the purpose of pattern recognition is much more in line with Kolmogorov’s notion of object representation by minimal description length using computer programs.

6.3 Literature Review on Analog to Digital Data Transformation Algorithms

A significant amount of research has focused on using the biological realism of the SNN for information processing applications akin to traditional neural networks (Maass, 1997). Under the broad umbrella of SNN, the area of data encoding has been relatively unexplored compared to neuronal dynamics, network learning behaviours and so on. Human Information Processing Research Laboratory's (Advanced Telecommunication Research Institute) artificial brain (Cellular Automata Machine Brain) project (De Garis, 1994) used data encoding as a part of its large-scale brain-like neural architecture. Hardware accelerated implementation of spike encoding for image and video processing was performed by Iakymchuk et al. (2014). The literature on the application of spike encoding on the information processing task in data science is restricted to a few algorithms, such as temporal-contrast (TC) (N. Kasabov, Scott et al., 2016), Hough spiker algorithm (HSA) (Hough, De Garis, Korkin, Gers & Nawa, 1999) and Bens spiker algorithm (BSA) (Schrauwen & Van Campenhout, 2003). HSA and BSA algorithms are event-driven in nature and can be classified under the temporal encoding paradigm where the time of occurrence of an event (spike) is considered as a unit of information. The TC algorithm, also known as AER encoding, is inspired from the human visual cochlea. The TC algorithm uses a threshold-based method to detect signal contrasts or changes (N. Kasabov, Scott et al., 2016). A user-defined or auto-generated contrast threshold determines the spike events in the TC algorithm. The HSA and BSA algorithm, however, determine a spike event using a deconvolution operation between the observed signal and a predefined filter. The HSA method which is based on convolution function produces a biased converted signal where it always stays below the original waveform and this would yield an error. The BSA method on the other hand uses the Finite Impulse Response (FIR) reconstruction filter. Even though BSA reduces the error in the HSA method and has less optimal threshold sensitivity, this method like HSA, needed a suitable filter for every type of input. Finding this filter automatically for each image would require a tremendous amount of work and time. There are some LIF neural network modelling approaches that for analog to digital transformation applied to computer vision (Van Rullen, Gautrais, Delorme & Thorpe, 1998). These methods, however, require a large number of input neurons.

6.4 A General Framework of Spike-time Encoding and Compression for Temporal Data Sequences

The temporal encoding problem for pattern recognition can be formalised as a data compression problem. An encoder is hence defined as the map $E : \mathbb{R}^T \to \mathbf{t}^f$, where the encoder $E(\cdot)$ release spikes at firing times $\mathbf{t}^f := \{t_1^f, t_2^f, \cdots t_n^f | t_i \in \mathbb{I}^+\}$. The temporal encoding algorithm primarily assumes that the discriminatory information is encoded by the sequence of discrete spike-timings rather than the magnitude and/or the spike density. As a consequence of this assumption, the temporal encoding aims at joint maximisation of information representation and minimisation of the spike density. Thus, it is in sharp contrast to the rate coding hypothesis. Next, the proposed *a priori* knowledge driven generalised framework for temporal encoding will be presented. This framework will be further extended to formalise a temporal encoding algorithm for fMRI data.

6.4.1 Formalisation of the *a priori* Knowledge Driven Optimisation Problem for Data Encoding

If one assumes a continuous source signal is represented by $\mathbf{s} \in \mathbb{R}^T$ representing a vector of continuous values, an encoded data or a spike-train is represented by $\mathbf{b} \in \{0, 1\}^T$ as a fixed-length binary sequence. This formalisation is slightly modified from the variable length sequence formalisation of spike-timings $\{t_1^f, t_2^f, \cdots t_n^f | t_i \in \mathbb{I}^+\}$ defined earlier without any loss of generality. For example, a historical spike sequence $[01001011]$ can be rewritten as a sequence of spike-time indices $t^f := \{1, 4, 6, 7\}$. T denotes the length of the temporal signal to be encoded into spikes. The *a priori* knowledge driven optimisation based encoding framework is built on the premise that

- The universal data encoder is non-existent.

- *a priori* knowledge about the data generation process or in other words, prior knowledge of the properties of the data generation source can be injected into a predictive system that can generate a predicted signal $\hat{\mathbf{s}}$.

For example, the fMRI data generation process behaves like a linear time invariant system, where a spike in the brain cell gives rise to a signal mimicking the gamma distribution function (Ashby, 2011), whereas the process of EEG data generation can be modelled as a phase varying mixture model of sinusoidal waves or multi-source Gaussian noise model (Nunez, Nunez & Srinivasan, 2016). The notion of knowledge injection is further elaborated in Section 6.5 using fMRI as an example. If it is possible to formalise a decompression function $\hat{\mathbf{s}}$ from the spike sequence $\mathbf{b}$, the optimal encoding of data can be formulated as an optimisation problem that minimises the discrepancy between the predicted and the original signal. One way of realising such a discrepancy is by minimising the root mean squared error (RMSE) of decompression between the observed signal s and the predicted signal $\hat{\mathbf{s}} := f(\mathbf{b}, \mathbf{\Theta})$, $\mathbf{\Theta}$ being the set of additional parameters required along with $\mathbf{b}$ to describe the prediction function. The optimisation problem can be formulated as:

$$\begin{aligned}
\min_{\mathbf{b},\mathbf{\Theta}} \quad & \sqrt{\frac{\sum_t (s_t - \hat{s}(b_t, \mathbf{\Theta}))^2}{t}} \\
\text{s.t.} \quad & b_t := \mathbb{I}^+ \\
& 0 \le b_t \le 1 \\
& \sum_t b_t \le \alpha \\
& \boldsymbol{\beta} \le \mathbf{\Theta} < \boldsymbol{\gamma}
\end{aligned} \tag{6.1}$$

The above optimiser solves for the RMSE, subject to the following constraints:

1. Binary constraints for spikes: The binary constraint for the spikes are implemented by forcing B_t to be an integer and within a range of $[0, 1]$.

2. Constraint on the number of spikes: The $\sum_t B_t \le \alpha$ constraint enforces the maximum number of spikes to be limited to a. This constraint is of major

importance from a biological plausibility perspective. Since the encoding scheme discussed here, aims to mimic the temporal coding behaviour of the human brain, it is always preferable to encode maximal information with the minimal number of spike.

3. Bounds can be set on the other parameters Θ to be optimised as part of the prediction model $f(\mathbf{B}, \mathbf{\Theta})$.

Mixed Integer Optimisation and Genetic Algorithms

The aforementioned optimisation problem is one related to mixed-integer programming optimisation. A mixed integer programming problem is an optimisation problem, linear or nonlinear, with or without constraints, in which some or all decision variables are restricted to have integer values. Such problems frequently arise in various application fields such as process industry, finance, engineering design, management science and others.

Several classical computational techniques (such as, branch and bound technique, cutting planes technique, and outer approximation technique), which are reasonably efficient, have been proposed in the literature for solving mixed integer programming problems (Cooper, 1981; Floudas, 1995; Grossmann, 2002). Over the last couple of decades, several stochastic algorithms have been developed and appropriately updated for problems related to mixed integer programming, such as simulated annealing, differential evolution and ant colony optimisation (Dorigo, Maniezzo & Colorni, 1996; Babu & Jehan, 2003; Yiqing, Xigang & Yongjian, 2007). However, algorithms of this class generally harbour the capacity to provide near global optimal solutions, although the quality of the obtained solution is unstable and requires large amount of computation time.

Genetic algorithms (GA) are general purpose population based stochastic search methods inspired by Charles Darwin's principles of natural selection and genetics. Holland (1992) introduced the concept of GA, and it was used by Jong

(1975) to solve the optimisation problem. Goldberg (1989) presents a detailed implementation of GA. To describe it in a simple manner, GA searches for sets of better solutions in the global search space. Potential solutions termed as chromosomes (individuals) are evolved iteratively over generations using a set of genetic operators such as selection, crossover and mutation. The quality of a solution or a population is measured by a fitness function, which is equivalent to a loss function in the field of machine learning and objective function in optimisation. The fitness function is responsible for evaluating how 'fit' a chromosome is for reproduction. The selection operator chooses the best 'fit' chromosomes for reproduction. In the reproduction process, new chromosomes are created by crossover and mutation operations. The Crossover operator blends the genetic information between chromosomes to explore the search space, whereas the mutation operator is used to maintain adequate diversity in the population of chromosomes to avoid premature convergence. The way the variables are coded into the chromosome is clearly essential for GAs' efficiency. Real coded genetic algorithms (RCGAs), which use real numbers for encoding, have fast convergence towards optima as compared to binary and Gray coded GAs (Deb, 2001). Also, 'RCGAs' overcome the difficulty of Hamming Cliff as in binary coded GAs. In the case of integer and mixed integer programming problems, many applications of GAs are available in the literature, some of which use binary coded representation (Cheung, Langevin & Delmaire, 1997; Luo, Guignard & Chen, 2001; Hua & Huang, 2006) and some use real coded representations (Y.-X. Li & Gen, 1996; Yokota, Gen & Li, 1996; Maiti, Bhunia & Maiti, 2006).

Integer programming with GA modifies the vanilla GA algorithm in several ways:

1. It requires custom creation, crossover and mutation function in order to enforce the variables to be integers (see (Deep, Singh, Kansal & Mohan, 2009) for detail).

2. The genetic algorithm attempts to minimise a penalty function, not the fitness function. The penalty function includes a term for infeasibility. This penalty function is combined with binary tournament selection to select

individuals for subsequent generations. The penalty function value of a member of a population is:

- If the member is feasible, the penalty function is the fitness function.
- If the member is infeasible, the penalty function is the maximum fitness function among feasible members of the population, plus a sum of the constraint violations of the (infeasible) point. For details of the penalty function, see (Deb, 2000).

3. GA does not enforce linear constraints when there are integer constraints. Instead, it incorporates linear constraint violations into the penalty function.

In the present implementation, the mixed integer genetic algorithm solver (Deb, 2000; Deep et al., 2009) was used. The constraints in Equation 6.1 are imposed on the parameters of $\hat{s}$. The first and second constraints are used to reduce the search space of the possible values of b_t to $\{0, 1\}$. The hyperparameter α is used to control the maximum number of spikes and hence the spike density in the optimal solution. The other sets of hyper-parameters $\{\boldsymbol{\beta}, \boldsymbol{\gamma}\}$ are used to control the upper and lower bounds of the model parameter $\boldsymbol{\Theta}$.

The formulation above for the proposed framework for data encoding is generic, flexible and is driven by knowledge-injection from the data source. The knowledge-injection component allows the further inclusion of systematic noise as part of $\hat{\mathbf{s}}$, if present. Examples of the inclusion of noise models, such as acoustic noise as part of linear time invariant models of fMRI are treated in (Sierra, Versluis, Hoogduin & Duifhuis, 2008; Z. Cho et al., 1997). The hypothesis is that a sufficiently good choice of $\hat{\mathbf{s}}(\mathbf{b}, \boldsymbol{\Theta})$ preserves, in some cases, enhances the discriminative property of the data in a greatly compressed space. It must also be noted that this formulation adheres to the concept of the non-existence of a universal compression algorithm for all the data sources. The general framework described above can be used to derive specific methods for encoding of special types of data for which *a priori* knowledge is available. One such case is fMRI data based on blood-oxygen level dependent response (BOLD). This is further introduced and illustrated in Section 6.5.

6.5 GAGamma: A Spike-time Encoding and Compression Method for fMRI Data

This Section will formalise a sample prediction model $f(\mathbf{B}, \boldsymbol{\Theta})$ for functional Magnetic Resonance Imaging (fMRI) data, and will present experimental results and evaluation of data encoding by solving Equation 6.1.

6.5.1 fMRI As a Linear Time Invariant System

Functional magnetic resonance imaging (fMRI) is a form of magnetic resonance imaging that takes advantage of magnetic susceptibility artefacts caused by the deoxygenated haemoglobin in the brain. Magnetic susceptibility measures the magnetic properties of the interaction between a tissue or other substance and the in-scanner magnetic field strength. Magnetically susceptible materials distort the homogeneity of a magnetic field: materials with negative magnetic susceptibility are known as diamagnetic, and those with positive magnetic susceptibility are referred to as paramagnetic. Introduction of a paramagnetic substance such as deoxyhaemoglobin into the scanner magnetic field causes variability in field strength, spin dephasing, geometric distortion and loss of signal; fMRI exploits this property by measuring changes in the relative ratio of oxygenated (diamagnetic) to deoxygenated (paramagnetic) haemoglobin in the blood.

Functional Magnetic Resonance Imaging (fMRI) is most commonly acquired using Blood Oxygen Level Dependent (BOLD) response. The BOLD response is measured by the changes in deoxyhaemoglobin at time t, and is caused by neural activation in the brain. The neural activations are caused by some sequence of events driven by the task performed by the subject (Friston et al., 1995). The BOLD response is mathematically described as a time invariant system, *i.e.* a system whose output does not depend explicitly on time. Under the appropriate experimental protocol, BOLD response also pertain to the superposition principle and henceforth can be designed as a linear time invariant (LTI) (Vazquez & Noll, 1998). According to C.-T. Chen (1995), a LTI system is said to be completely characterised by convolution integral functions. The fMRI BOLD is described by the convolution of the spikes $\mathbf{b}$ and the haemodynamic response function (HRF), $h(\mathbf{\Theta})$. This operation is characterised by Equations 6.2 and 6.3.

$$\hat{\mathbf{s}} := \int_0^t \mathbf{b}(\tau)h(t-\tau)d\tau \tag{6.2}$$

$$\hat{\mathbf{s}}(\mathbf{b}, \mathbf{\Theta}) := \mathbf{b} * h(\mathbf{\Theta}) \tag{6.3}$$

$$h(\theta_1, \theta_2) := \frac{1}{\theta_2^{\theta_1}\mathcal{T}(\theta_1)} t^{\theta_1 - 1} e^{-\frac{t}{\theta_2}} \tag{6.4}$$

6.5.2 GAGamma Optimisation Problem for fMRI

Numerous models for HRF have been proposed in the literature (Boynton, Engel, Glover & Heeger, 1996; Friston, Josephs, Rees & Turner, 1998; Glover, 1999). The majority of mathematical models for the canonical HRF are found to be some variant of the gamma function. In all the experiments performed here, the gamma distribution function has been used as the HRF model (Equation 6.4). This function is characterised by the parameter set $\mathbf{\Theta} := \{\theta_1, \theta_2\}$, where $\theta_1 \in \mathbb{R}^+$ and $\theta_2 \in \mathbb{R}^+$ controls the shape and the scale of the gamma function respectively. By substituting Equations 6.3 and 6.4 in Equation 6.1, the encoding problem can be reduced to solving Equation 6.5 and is referred to as GAGamma encoding algorithm hereafter.

$$
\begin{aligned}
\min_{\mathbf{b},\theta_1,\theta_2} \quad & \sqrt{\frac{\sum_t (s_t - \hat{s}(b_t,\theta_1,\theta_2))^2}{t}} \\
\text{s.t.} \quad & b_t := \mathbb{I}^+ \\
& 0 \le b_t \le 1 \\
& \sum_t b_t \le \alpha \\
& \beta_1 \le \theta_1 \le \gamma_1 \\
& \beta_2 \le \theta_2 \le \gamma_2 \\
\text{where} \quad & \hat{s}_t(b_t,\theta_1,\theta_2) := b_t * \frac{1}{\theta_2^{\theta_1}\mathcal{T}(\theta_1)} t^{\theta_1 - 1} e^{-\frac{t}{\theta_2}}
\end{aligned}
\tag{6.5}
$$

6.5.3 Distinction of GAGamma from HSA and BSA

At this point, it is imperative to make the distinction between the GAGamma and the existing HSA and BSA algorithms. Apart from exhibiting similarities in convolution framework, HSA and BSA also resemble GAGamma as methods of stimulus estimation using FIR. Nevertheless, the data encoding approach in HSA and BSA use a deconvolution (of Equation 6.2) approach contrary to the optimisation approach in GAGamma. The knowledge-injection component of GAGamma, as part of formalisation of $\hat{\text{s}}$ and the optimisation approach, has two distinct benefits over the deconvolution-based methods:

- A generic Gamma function has been used as the knowledge-injection component to $\hat{\text{s}}$ in GAGamma, which is driven by the existing knowledge about the fMRI data as opposed to the generic sinusoidal function used as the FIR in BSA. It is also argued that this formulation allows the inclusion of additional knowledge about the data source (such as systematic noise) if present, providing greater flexibility in the formulation of the encoding algorithm.

- The optimisation problem formulation in GAGamma jointly optimises for the parameter set $\boldsymbol{\Theta}$ and $\mathbf{b}$. This formulation thus includes the parameter set $\boldsymbol{\Theta}$ of the prediction model $\hat{\text{s}}$ and the spike sequence $\mathbf{b}$ for each individual voxel or feature. In HSA and BSA, the equivalent filter parameters are predetermined for the whole set of voxels and are not learned from the data.

- The constraint $\sum_t b_t \leq \alpha$ in GAGamma ensures the flexibility in choosing the desired spike density, hence the ability to control the compression and quality of signal reconstruction. The BSA or HSA algorithm, on the contrary, accommodates no such control in the encoding framework.

6.6 Experiments and Evaluation

6.6.1 Description of Dataset

The experiments described in this Chapter were performed on the publicly available benchmark Starplus fMRI dataset (Mitchell & Wang, 2001) collected by The Centre for Cognitive Brain Imaging, Carnegie Mellon University. The Starplus experiment was conducted on a set of 7 subjects. Each subject had undergone multiple trials of exactly the same cognitive experiment. At every trial lasting for 27 seconds, a set of stimuli were presented to a subject in the following order:

1. The first stimulus (picture or sentence) was presented at the beginning for 4 seconds.

2. A blank screen was presented during the interval of $5-8$ seconds.

3. The second stimulus (sentence or picture) was presented during the interval of $9-12$ seconds.

4. A rest period of 15 seconds was added after the presentation of the second stimulus.

While the subject performed the cognitive tasks, fMRI images from specific regions of interest (ROI) of the brain were collected at every 500ms interval. The preprocessed fMRI dataset has been used in a number of pattern recognition studies (Mitchell et al., 2003; Mitchell et al., 2008; Shinkareva et al., 2008). In this study, this benchmark dataset was chosen to build pattern recognition systems that can predict and discriminate between the binary cognitive states of a subject 'seeing a picture' and 'reading a sentence'. Two subjects were chosen (id: 04847 and 07510) randomly and two spatial ROIs; Calcarine Sulcus ('CALC') and Left Intra-Parietal Sulcus ('LIPL') were used for the experiments. The choice of the ROI is based on previous work (Do & Yang, 2014) that found these ROIs to be amongst the most discriminatory in the raw continuous data space. The dataset for each subject is composed of 40 samples (trials) of each class, and each sample is made up of 452 and 483 voxels in subject 04847 and 07510 respectively. Each cognitive task lasted for a total of 8 seconds emitting 16 fMRI images for each class within a trial.

6.6.2 Evaluation Metrics

Three metrics have been used to evaluate and compare the performance of the encoding techniques and the traditional 'no-encoding' (raw data) approach. The evaluation criteria and the baseline encoding techniques are elaborated below:

Bit Compression Ratio

The Bit Compression Ratio (BCR) is defined as the ratio between the average number of bits required to store an encoded dataset and the number of bits required to store a raw dataset, respectively. BCR is directly associated with the relative description lengths (DL) and data type of the datasets. The DL of a dataset is described by the length of the dataset represented by the number of values in the dataset. If one assumes a dataset intended for pattern recognition is represented by $D_{raw} := \{x_1, x_2, \cdots x_n | type(x_i) = \mathbb{R}\}$ which is transformed by an encoding algorithm to $D_{encoded} := \{y_1, y_2, \cdots y_m | type(y_j) = \mathbb{I}^+\}$, where m and n are the DL of the raw and the encoded data respectively. The BCR is then estimated as:

$$BCR := \frac{m \times sizeof(\mathbb{I}^+)}{n \times sizeof(\mathbb{R})} \tag{6.6}$$

The notion of BCR (Equation 6.6) can be analysed from the viewpoint of the Kolmogorov complexity. As described earlier, Kolmogorov's descriptional complexity aims at a simpler object representation and simplicity is measured by the DL of the object. Here, the object being a pattern recognition dataset, the objective is to achieve simpler representation of the dataset by performing the encoding operation. This is achieved by minimising the numerator $m \times sizeof(\mathbb{I}^+)$. A compression is said to be achieved, if $0 < BCR < 1$ is satisfied. It is also quite evident from Equation 6.6 that the data type of the objects present in the dataset contributes significantly to the BCR metric. In this case, the encoded data being represented by positive integers (spike-timings) as opposed to the floating-point numbers in the raw data, already contribute significantly to BCR. Additionally, the temporal encoding algorithms aspire to minimise the DL of the object ($m << n$), thus achieving a lower BCR.

Decoding Error

The decoding error metric is the measure of the decompression reliability, *i.e.* the ability to recover the original signal from the compressed spike-timings reliably. RMSE of signal reconstruction between the original signal s, and the predicted signal $\hat{s}$ has been used as a measure of decompression reliability in this study. The RMSE is given by:

$$RMSE := \sqrt{\frac{\sum_t (s - \hat{s}(b_t, \Theta))^2}{t}} \tag{6.7}$$

A low RMSE of the signal reconstruction indicates higher preservation of the original data in the spike-timings. However, low RMSE is not necessarily indicative of a better encoding for pattern recognition. For example, an encoder producing better reconstruction error for noisy data may indicate inefficient noise filtering. It must also be noted that the prediction models are built on the spike-time data and have no knowledge of the mapping $\mathrm{s} \rightarrow \mathrm{b}$ being performed beforehand. Hence, although this metric plays an important role in evaluating the robustness of the encoding algorithm with respect to the reconstruction of the raw data, the effect on the quality of pattern recognition performance is unaffected.

Classification Performance

From the pattern recognition viewpoint, due to the balanced nature of the dataset, the classification accuracy is the most important and relevant measure of success and this has been used as a measure of classification performance. The mean accuracy is estimated from thirty independent runs of 50/50 train/test split of the binary classification data described previously.

As the data that was encoded was intended to be used for pattern recognition problems, conservation and possible enhancement of the discriminatory information in the spike-timings is as crucial as efficient compression of the data. This is a distinctly different approach from the existing ones in pattern recognition. In the traditional pattern recognition approach, the volume of the data plays a crucial role in the performance of the pattern recognition algorithms to produce accurate predictions. In the temporal encoding approach, by keeping both compressibility and classification performance as the criteria of evaluation, the aim is to benefit from the efficient representation of information in the data along with the classification performance. It is, thus, important to have a balance between compression and conservation of discriminatory information in the encoded data.

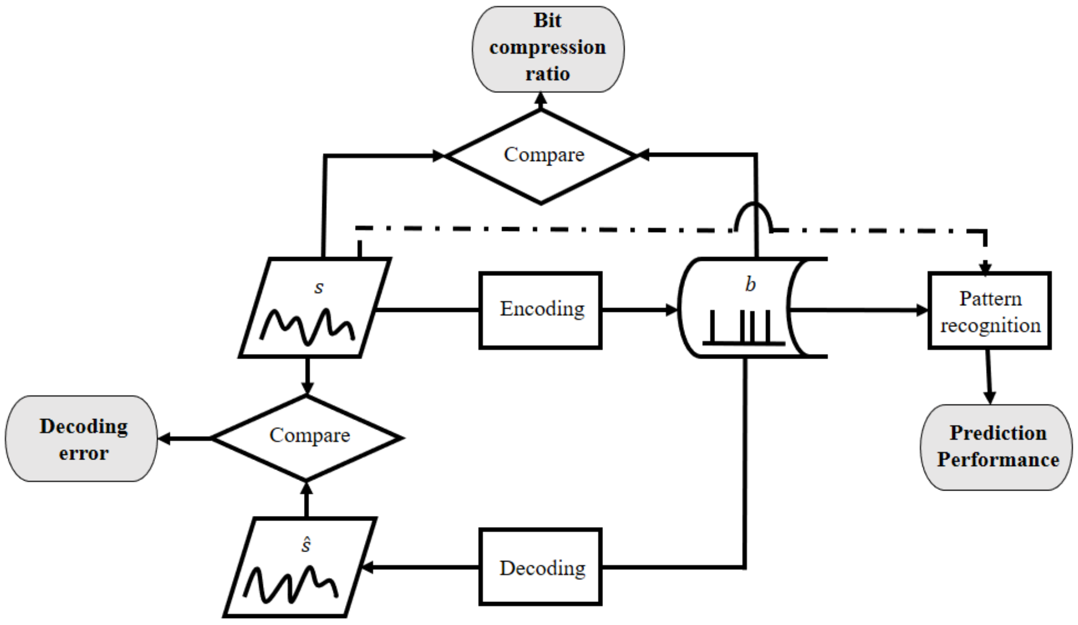


Figure 6.1.: Flowchart depicting the evaluation criteria and experimental protocol used in this research.

6.6.3 Design of Experiments

Figure 6.1 shows a flowchart of the experimental design used in this Chapter. The experimental protocol begins with the raw temporal data s. At the first step, the encoding operation is performed on s to generate the encoded spike-time data b. In the second step, a K-NN based prediction model is learned using a fraction of b. The rest of b is used to test the performance of the model emitting the prediction performance. The next evaluation criteria relate to the compressibility of the encoding algorithms. To evaluate the compressibility, BCR is calculated by comparing b and s. Finally, the lossiness of the encoding operation is evaluated by comparing predicted signal $\hat{s}$ produced by the decoding algorithm (corresponding to the encoding algorithm) and the ground truth s to emit the decoding error.

K-Nearest Neighbour (K-NN) Algorithm with Custom Distance Function

As discussed earlier, the non-parametric K-NN algorithm for building the classification model from the data is used. The class label prediction of a new sample (in this case a spike-train) in K-NN is a majority vote between the neighbours of the new sample, where the sample is assigned to the class label most common among its K nearest neighbours. To evaluate the neighbourhood of a sample, it is hence necessary to calculate pairwise distances between the sample to be predicted and the training samples. As the key interest is in learning about a K-NN model for both raw (in continuous space) and spike-time (discrete space) data, there have been two different distance functions used: raw and encoded data. For the raw data in the continuous space, the standard Euclidean distance or the L^2 norm as the distance function was used. On the other hand, for the spike-time dataset, the spike asynchronicity based distance function was used, and is described below:

Spike asynchronicity based distance function: Here, I propose a distance function that can capture the relative distance between a pair of the spike-train samples. Since the concern is with using spike-timings as a carrier of information, a useful way to capture similarity between a pair of spike-train samples is to record if the two samples have spiked at the same time instances. Therefore, mean absolute asynchronicity has been used as the distance function. The mean absolute asynchronicity based distance function between two spike-train samples $\mathbf{b_1} \in \{0,1\}^{T\times M}$ and $\mathbf{b_2} \in \{0,1\}^{T\times M}$ is formalised as the mean pairwise Hamming distance between all feature-wise pairs $\mathbf{b_1^m}$ and $\mathbf{b_2^m}$, where M is the feature count. As the spike-time data lies in the binary space, the mean pairwise Hamming distance is equivalent to the mean XOR distance between the pairs of spike sequences.

6.6.4 Baseline Encoding Algorithms

In this study, three different encoding methods have been compared and evaluated. It must be noted that for each encoding or compression algorithm, there is also a decoding algorithm which decompresses the spike-trains into the reconstructed signal $\hat{\mathbf{s}}$.

Algorithm 6.1 BSA encoding algorithm

```
input: $s, filter, threshold_{BSA}$
output: $b$
$b \Leftarrow 0$
$L = length(s)$
$F = length(filter)$
**for** $t = 1 : (L - F + 1)$ **do**
    $e_1 \leftarrow 0$
    $e_2 \leftarrow 0$
    **for** $k = 1 : F$ **do**
        $e_1 \mathrel{+}= |s(t + k) - filter(k)|$
        $e_2 \mathrel{+}= |s(t + k - 1)|$
    **end for**
    **if** $e_1 \le (e_2 - threshold_{BSA})$ **then**
        $b(t) \leftarrow 1$
        **for** $k = 1 : F$ **do**
            $s(i + j - 1) \mathrel{-}= filter(k)$
        **end for**
    **end if**
**end for**
```

Algorithm 6.2 BSA decoding algorithm

```
input: $b, filter$
output: $\hat{s}$
$L = length(b)$
$F = length(filter)$
**for** t=1:L-F+1 **do**
    **if** $b(t) == 1$ **then**
        **for** $k = 1 : F$ **do**
            $\hat{s}(t + k - 1) \mathrel{+}= filter(k)$
        **end for**
    **end if**
**end for**
```

- GAGamma: The GAGamma encoding method is outlined and described in Section 6.5. The encoding and decoding principles are given by Equations 6.3 and 6.5.

- BSA: The BSA encoding and decoding algorithms (Schrauwen & Van Campenhout, 2003) are formalised in Algorithms 6.1 and 6.2, respectively. The BSA encoding algorithm takes a filter function and a threshold value as input along with the signal s. The deconvolution approach of BSA begins with a FIR filter, and at every time instant τ calculates two error metrics: $\sum_{k=0}^{P} abs(s(k+\tau)-h(k))$ and $\sum_{k=0}^{P} abs(s(k+\tau)$, where P is the number of filter taps. If the first error is less than the second error minus the threshold, then the BSA encoder fires a spike and subtracts the filter from the input (Schrauwen & Van Campenhout, 2003).

- Temporal contrast: The temporal contrast algorithm captures the greater than average changes in the data as spikes. Algorithms 6.3 and 6.4 presents the temporal contrast encoding and decoding algorithms respectively. One major characteristic and deviation of temporal contrast algorithm from the temporal encoding framework is its ability to generate spikes with positive and negative polarity. Since the main interest is in the spike-timings, during the classification, the polarity of the spikes have been ignored. The algorithm takes the $factor \in \{0,1\}$ parameter as input. This parameter controls the estimate of the $threshold_{TC}$ variable, which is responsible for determining the spike-timings.

6.6.5 Results

For the comparative evaluation of the encoding methods with the classical 'no-encoding' (raw data) method, identical experiments were performed for the subjects 04847 and 07510. For the GAGamma encoding, two sets of hyperparameters have been used to demonstrate the tuning capability of the algorithm. In the GAGamma-16 method, the hyperparameter values $[\alpha = 16, \beta =$

Algorithm 6.3 Temporal contrast encoding algorithm

```
1: input: $s$, $factor$
2: output:$b, threshold_{TC}$
3: $L \leftarrow length(s)$
4: for $t = 1 : L - 1$ do
5:     $diff \leftarrow |s(t+1) - s(t)|$
6: end for
7: $threshold_{TC} \leftarrow mean(diff) + factor \cdot std(diff)$
8: $diff \Leftarrow [0, diff]$
9: for $t = 1 : L$ do
10:     if $diff(t) > threshold_{TC}$ then
11:         $b(t) \leftarrow 1$
12:     else if $diff(t) < -threshold_{TC}$ then
13:         $b(t) \leftarrow -1$
14:     else
15:         $b(t) \leftarrow 0$
16:     end if
17: end for
```

Algorithm 6.4 Temporal contrast decoding algorithm

```
1: input: $b, threshold_{TC}$
2: output: $\hat{s}$
3: $\hat{s} \leftarrow 0$
4: $L \leftarrow length(b)$
5: for $t = 2 : L$ do
6:     if $\hat{s}(t) > 0$ then
7:         $\hat{s}(t) \leftarrow \hat{s}(t-1) + threshold_{TC}$
8:     else if $\hat{s}(t) < 0$ then
9:         $\hat{s}(t) \leftarrow \hat{s}(t-1) - threshold_{TC}$
10:     else
11:         $\hat{s}(t) \leftarrow \hat{s}(t-1)$
12:     end if
13: end for
```

Table 6.1.: Comparative evaluation of the data encoding algorithms applied to subject 04847 and 07510 in the Starplus fMRI dataset.

subject id	method	data type [1]	BCR	decoding error	classification accuracy (K^2)
04847	GAGamma-16	int	0.15	**0.07**	$87.41 \pm 4.80(16)$
	GAGamma-3	int	**0.04**	0.29	$85.02 \pm 4.76(11)$
	BSA	int	0.08	0.15	$84.5 \pm 4.47(3)$
	TC	int	0.06	0.23	$54.16 \pm 5.47(1)$
	No-encoding	decimal	1	-	$\mathbf{89.55 \pm 4.60}(1)$
07510	GAGamma-16	int	0.15	**0.06**	$76.00 \pm 5.89(8)$
	GAGamma-3	int	**0.04**	0.27	$\mathbf{81.16 \pm 7.50}(2)$
	BSA	int	**0.04**	0.15	$74.08 \pm 6.71(8)$
	TC	int	0.05	0.26	$52.75 \pm 5.84(2)$
	No-encoding	decimal	1	-	$79.11 \pm 3.99(5)$
	Random	int	0.11	-	$52.58 \pm 4.79(1)$

$0, \gamma = 10]$ (see Equation 6.5) were used, and in the GAGamma-3 method, $[\alpha = 3, \beta = 0, \gamma = 10]$ were used. The BSA encoding algorithm takes a finite impulse response $filter$ and a $threshold_{BSA}$ as input. In the current experiments, the low pass FIR filter of size 10 and the $threshold_{BSA} = 0.95$ have been used. These values are guided by the existing literature on the application of BSA on brain data (Nuntalid, Dhoble & Kasabov, 2011). For the temporal contrast encoding, the hyper-parameter $factor = 0.6$ has been used. As a baseline, a randomly generated spike-train dataset has also been included. The random spike-time dataset was created using a Poisson's distribution function ($\lambda = 0.6$). Varying the λ parameter affects the BCR directly for random spike generation. It must be noted that the presented results are non-exhaustive in the hyperparameter space of different encoding methods. In the 'no-encoding' method, the raw dataset was created by transforming each multi-dimensional time series (set of images) within a trial into a single static observation by concatenating the feature values across the 16 time intervals similar to the approach employed by Mitchell et al. (2003).

Comparison Between 'Encoding' and 'No-encoding'

The advantages of performing encoding as opposed to using the raw data ('no-encoding') is well established in Table 6.1. This table shows the experiments replicated across subjects 04847 and 07510. For every subject, three temporal encoding methods were evaluated. They are the proposed GAGamma, BSA and the temporal contrast encoding. In the GAGamma-X method, X corresponds to the value of the maximum number of allowed spikes, α (Equation 6.5). The temporal encoding methods are compared against the raw data or the 'no-encoding' method along with a random spike-train as a baseline. As discussed earlier, the encoding methods are evaluated using BCR, decoding error and classification accuracy as the measures of success. The decoding error metric is not applicable for the 'random' and 'no-encoding' method. This is because in 'no-encoding' and 'random' method, the raw data and a random spike generator were used, respec-

[1] An integer is assumed to take 8 bits and floating-point number 32 bits.
[2] K: Number of nearest neighbours used in K-NN algorithm

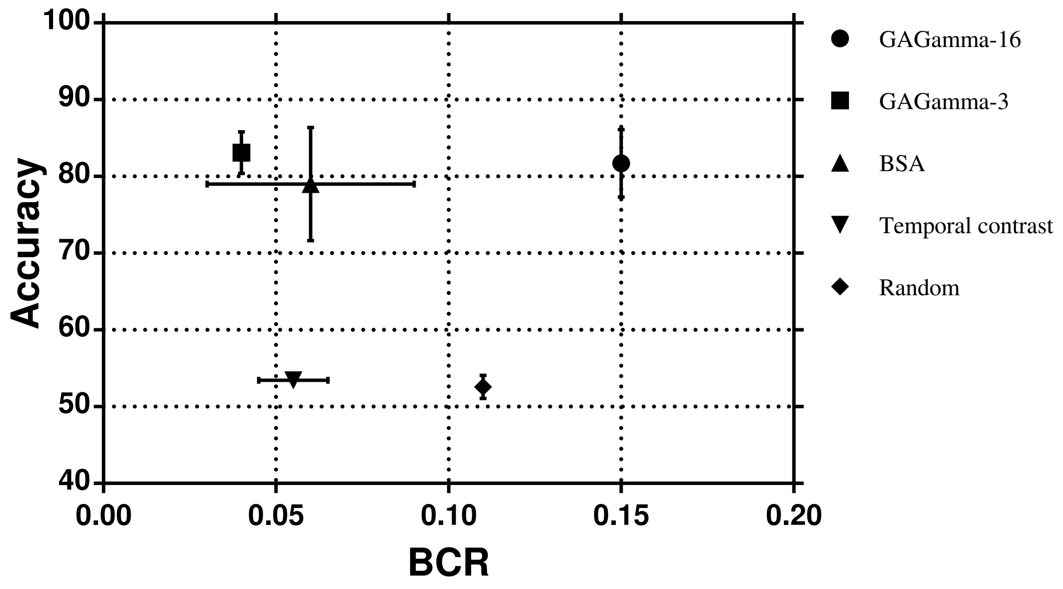


Figure 6.2.: Plot illustrating the comparison of the quality of the encoding methods with respect to the mean classification accuracy and bit compression ratio across the two subjects. The horizontal and the vertical error bars represent the standard deviation of accuracy and BCR across experiments.

tively, for pattern recognition and thus, encoding principles are not applied in these two cases.

It is observed that the encoding operations could compress the data dramatically and thus attaining BCR values between 0.04 and 0.15 *i.e.* an approximate compression of 6 to 25 times compared to the raw data. The most important metric of evaluation, the classification accuracy column in Table 6.1, shows that the GAGamma and BSA methods achieve comparable classification performances with respect to the 'no-encoding' method, and hence can capture the discriminatory information in the spike-time data well. It should also be noted that for subject 07510, the GAGamma-3 method achieved a classification performance of $81.16 \pm 7.50\%$ as opposed to the $79.11 \pm 3.99\%$ by the raw data and thus outperforming the 'no-encoding' method. This, it can be argued is due to the ability of the encoding algorithms to represent the information in the raw data into the spike-time and thus concisely present to the classifier.

Comparison of the Temporal Encoding Algorithms

Figure 6.2 graphically depicts the quality comparison of the different encoding techniques. The plot shows the mean BCR and the classification accuracy of the encoding techniques across the two subjects. The horizontal and vertical error bars are the standard deviations of the BCR and accuracy respectively. It can be initially observed that the GAGamma-16 and GAGamma-3 encodings show superior mean performance in the pattern recognition task compared to other methods. However, the aim is to simultaneously achieve high classification performance and a highly compressed information representation, and thus, the GAGamma-3 and BSA data points residing on the top left quadrant of the plot fare better overall in both respects. From the error bar representations, it can also be seen that the GAGamma method has a negligible deviation on BCR. This is due to the flexibility that the GAGamma method provides to control the spike density by the constraint $\sum_t b_t \leq \alpha$ (Equation 6.5) without sacrificing much pattern recognition performance. This can be of significant importance, especially, for the storage and transmission of large volumes of streaming data within limited resources, where the encoding operation can precisely tune the compression rate and hence the storage. It is also recognised from the plot that the temporal contrast encoding method fares poorly in this experiment and is no better than a random spike generator.

Table 6.1 also shows an inverse relationship between the BCR and the decoding error. This is because it requires significantly more effort to accurately represent the seasonal variations in the data using fewer spikes. However, if decoding robustness is of major importance, the GAGamma method can be tuned to maximise spike density and thus will have minimal loss in the signal reconstruction and thus sacrificing the compression. Figure 6.3 shows an example of the signal reconstruction done by the GAGamma-16, BSA and temporal contrast decoding algorithms. In Figure 6.4, comparisons were also performed on the RMSE of signal reconstruction across 100 different voxels using spikes encoded by the encoding methods. It can be clearly observed that the signal reconstruction by GAGamma (smaller RMSE means better reconstruction) is superior to the others

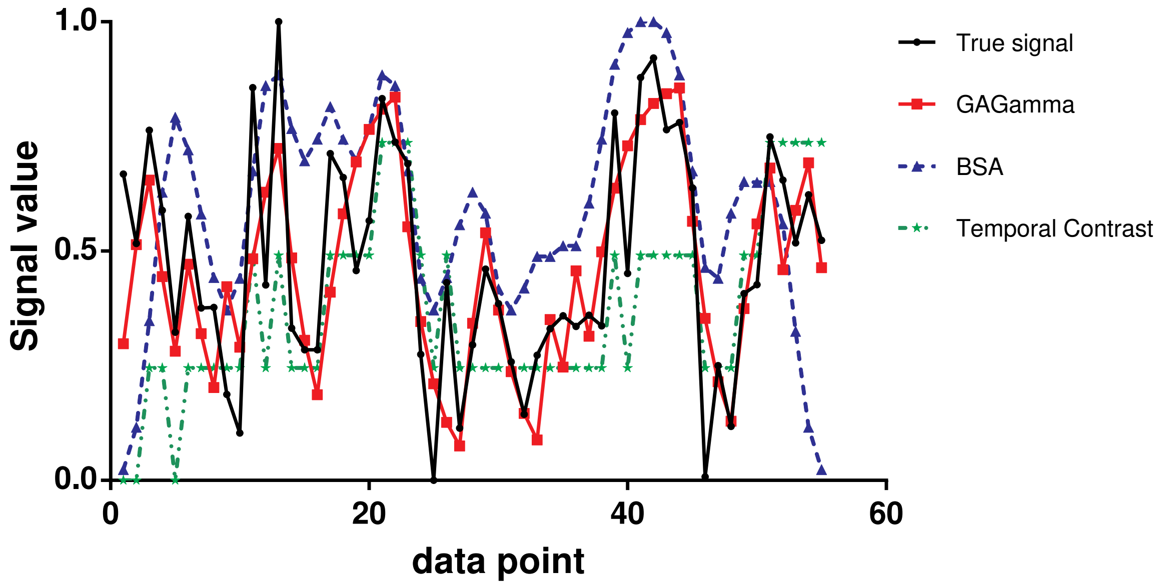


Figure 6.3.: Comparison of signal reconstruction ($\hat{s}$) from a spike sequence by GAGamma-16 decoding algorithm (Equation 6.3), BSA decoding algorithm (Algorithm 6.2) and Temporal contrast decoding algorithm (Algorithm 6.4). The true signal is randomly selected from subject 04847 (10^{th} trial and 23^{rd} voxel).

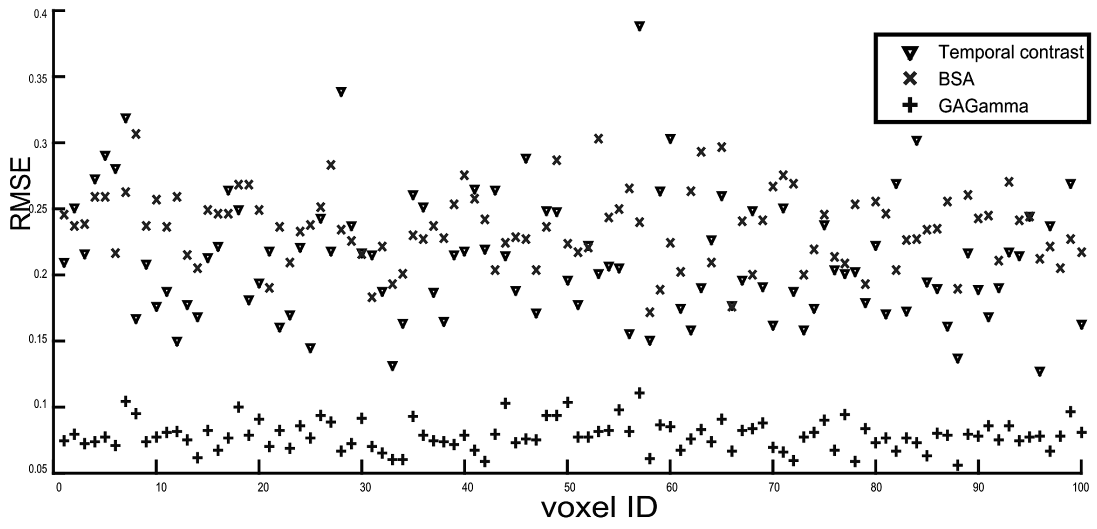


Figure 6.4.: Comparison of RMSE of reconstruction between GAGamma, BSA and Temporal Contrast across 100 voxels of second trial in subject 04847.

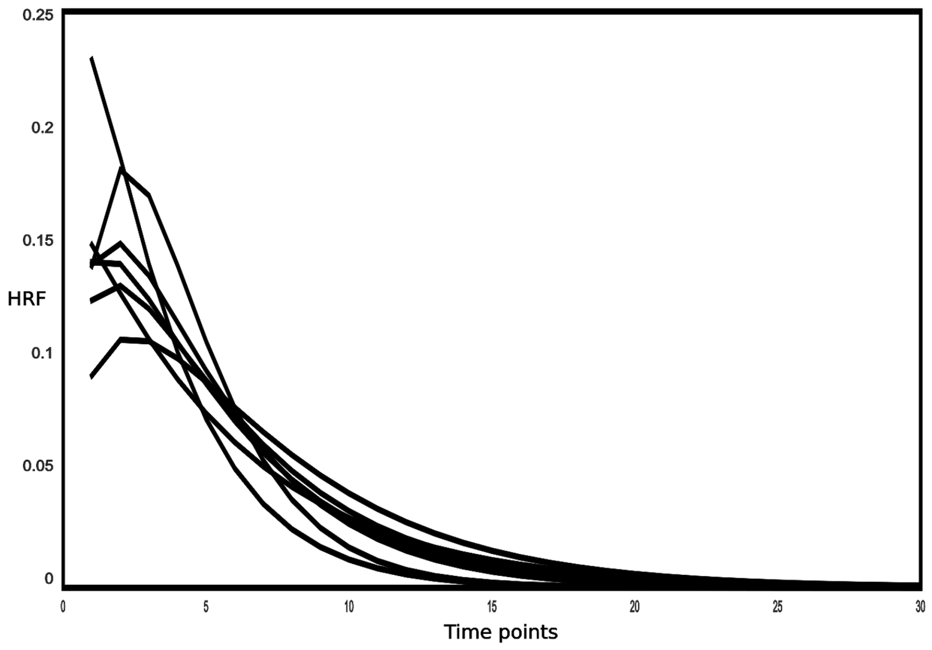


Figure 6.5.: Comparison of haemodynamic response function learned by GAGamma encoding method for voxel 8 of subject 04847 across 7 different trials.

as not only can it reconstruct the bigger trends in the signal but also the seasonal variations.

As part of the optimisation, the GA-gamma encoding method also optimises for the parameters of the response filter H. Figure 6.5 shows the gamma haemodynamic response filters learned by the model for a single voxel across 7 trials. It can be seen from the figure, that for a single voxel across trials, the shape of the HRF is nearly consistent, but varies in the scale. This result is consistent with the notion of the existence of minor variations of HRF across voxel and/or subject.

Analysis of the Spike-train Encoded by GAGamma-16 Method

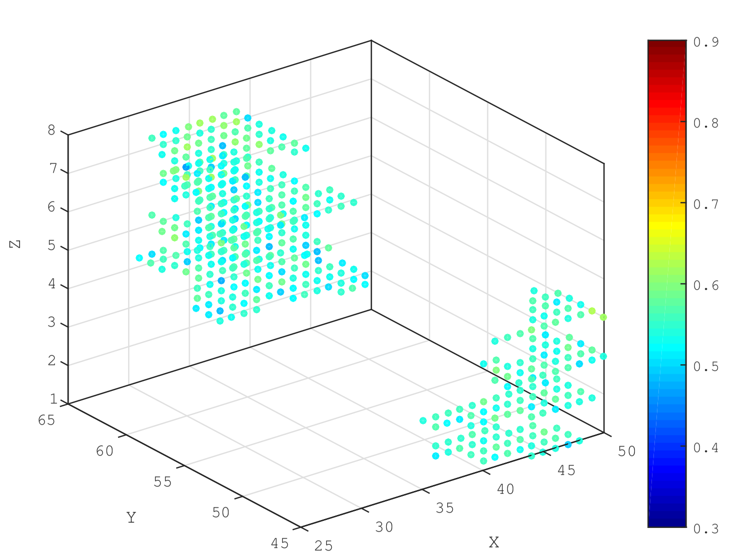
Additionally, GAGamma-16 encoded spikes for the ‘seeing a picture’, and the ‘reading a sentence’ stimuli were independently analysed for interpreting the discriminating spatio-temporal influence of the spikes. As described earlier in the experimental protocol, the presentation of a certain stimulus within a trial follows an order, *i.e.* for each stimuli class there exists subclasses of 'presented first' or 'presented second'. To analyse the effect of the first or second presentation of stimuli, the encoded dataset was separated into four classes, 'picture presented first', 'picture presented second', 'sentence presented first' and 'sentence presented second'. Figures 6.6 and 6.7 shows the comparison of the mean spike percentage across the trials for the four subclasses in subject 04847 and subject 07510. The points in the 3D plot correspond to the spatial location of the voxels. Each voxel belongs to two physiologically defined clusters or regions of interest, namely 'CALC' and 'LIPL.' The top row plots are the 'picture' trials, and the bottom row trials are the 'sentence' plots. The columns correspond to the stimulus ('picture' or 'sentence') being presented first or second. The two clusters in each of the 3D plots relate to the two ROI's (top left is 'LIPL' and bottom right is 'CALC') of the brain structure. Functionally, the 'CALC' region is responsible for central and peripheral vision whereas the 'LIPL' region is related to visual attention. In both the subjects, 'reading a sentence second' after 'seeing a picture first' has more spike activity on average across the trials than the other way around, especially in the 'LIPL' region. The mean spike activity in the 'LIPL' is observed to be relatively higher (0.59 and 0.57) when the subjects were 'reading a sentence' than when the subjects were 'seeing a picture' (0.54 and 0.55). A two-sample T-test was conducted between the 'seeing a picture' and the 'reading a sentence' class in the 'LIPL' region for the subjects to validate the previous result. The null hypothesis for the test conducted was the following, H_0: 'there is no difference between the picture spike activity and sentence spike activity'. The null hypothesis was rejected at 5% significance level with $p = 5.27 \times 10^{-18}$ for subject 04847 and with $p = 7.05 \times 10^{-12}$ for subject 07510. Hence, as per the T-test, the average spike activity across the trials over time for 'seeing a picture' is significantly different from the average spike activity across trials over time for 'reading a sentence'. Further, it must also be noted the sentences shown as part of the experiment are highly visual in nature (*e.g.* 'It is not true that the dollar is below the plus.') and requires a high image comprehension ability. This result is, therefore, consistent with the experimental results (Just, Newman, Keller, McEleney & Carpenter, 2004) obtained earlier which shows a greater degree of activation and functional connectivity in the 'LIPL' region during cognitive

Table 6.2.: Average pairwise asynchronicity of three different voxels at the end of ten independent runs of GAGamma encoding.

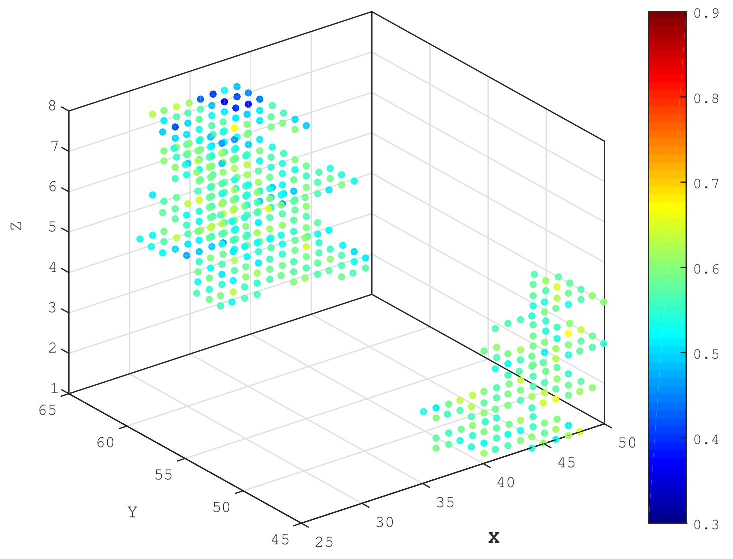
voxel ID	$\mathbf{d_p}$	$\mathbf{d_{vp}}$
30	24.18 ± 10.15	0.23 ± 0.09
468	27.78 ± 11.96	0.26 ± 0.10
3429	28.03 ± 11.31	0.28 ± 0.11

tasks associated with high imagery sentence comprehension. This, in fact, also validates the ability of the proposed encoding algorithm to preserve the useful discriminatory information in the compressed encoded space of data.

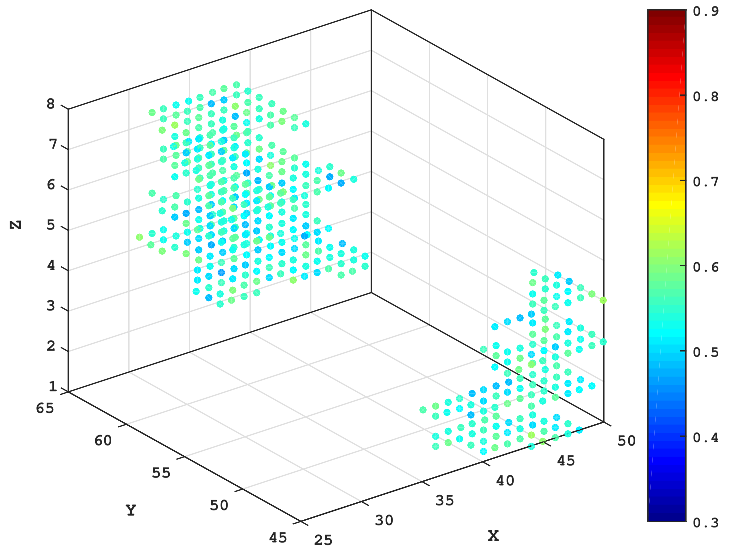
Table 6.2 relates to the reproducibility of the spike-timings produced by the mixed integer genetic algorithm solver for the GAGamma encoding. The genetic algorithm being an evolutionary optimisation solver produces a non-reproducible result when on multiple iterations. Nevertheless, a pareto-optimal fitness value is guaranteed on each iteration. To validate the reliability of the GAGamma optimisation, in this instance, ten independent runs of GAGamma encoding was applied on three random voxels (30468 and 3429) from trial 12 of subject 04847. Table 6.2 compares the similarity of the spike-trains produced by the GAGamma encoding using two spike-asynchronicity measures. They are the percentage asynchronicity d_p and Victor Purpura distance d_{vp} respectively. The Victor Purpura distance (d_{vp}) (Victor & Purpura, 1997) metric is a cost based distance metric. The distance is defined by the minimum cost of converting one spike-train into the other using three operations: insertion (cost 1); deletion (cost 1); and shifting a spike by an interval δt (cost $q|\delta t|$). For the smaller value of q the distance metric approximates the spike count difference and hence supports rate coding. A higher penalty value of q, on the contrary, supports the number of non-coincidental spikes and hence temporal encoding. The comparison of the spike synchronicity using d_p and d_{vp} in Table 6.2 shows that the spike-timings are correctly reproduced approximately 75% of times.



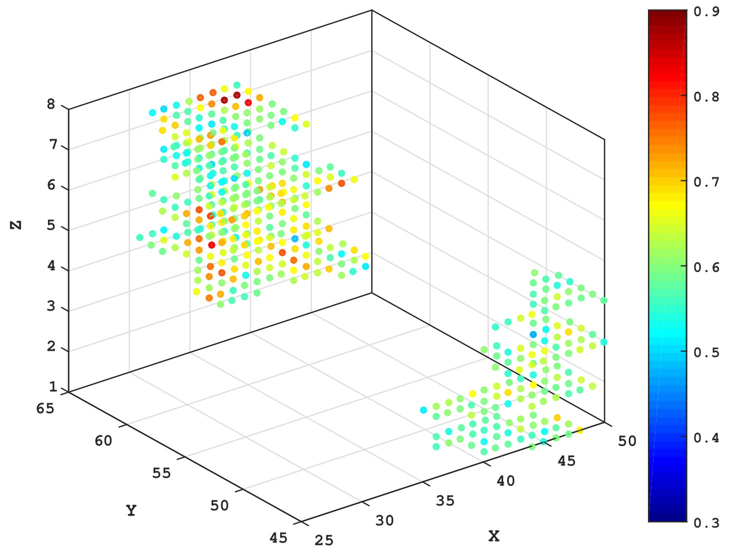
(a) ‘seeing a picture’ first during trial



(b) ‘seeing a picture’ second during trial

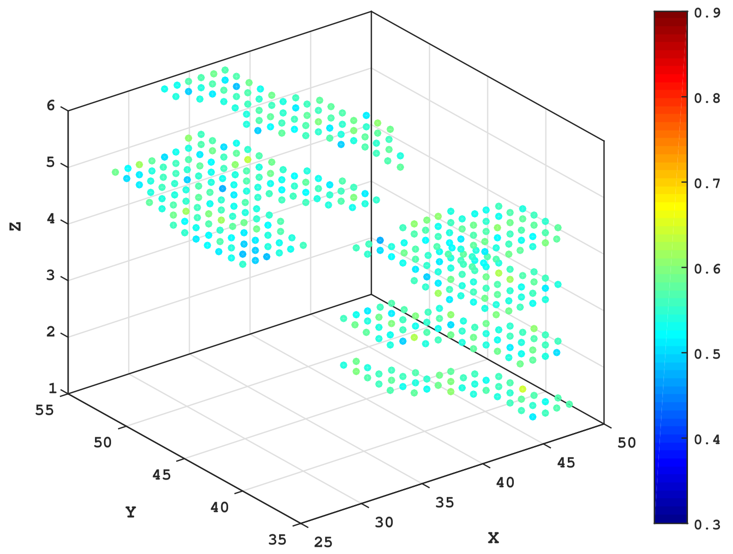


(c) ‘reading a sentence’ first during trial

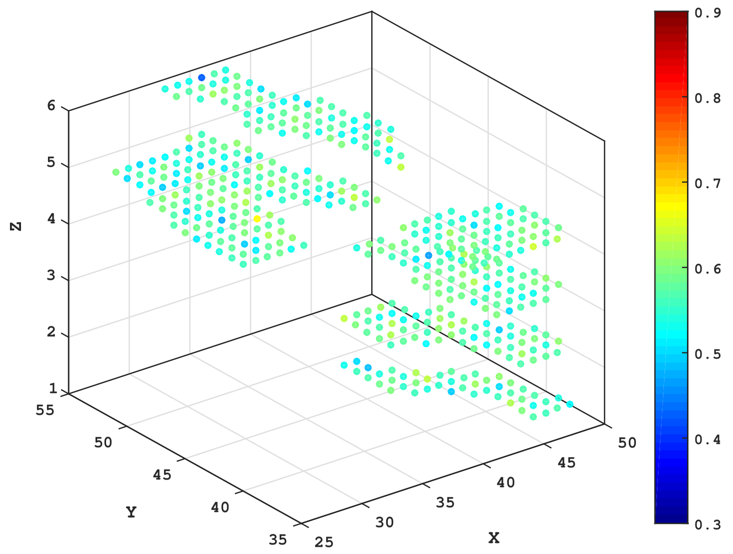


(d) ‘reading a sentence’ second during trial

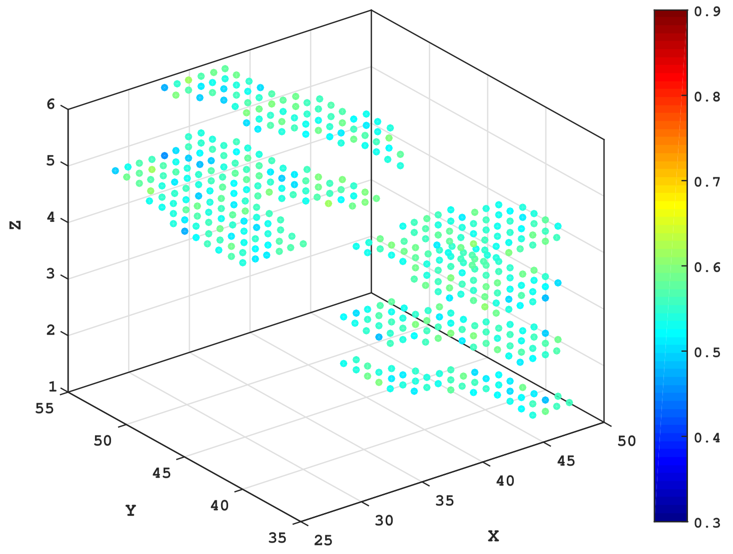
Figure 6.6.: Comparative analysis of spike frequencies of the subject 04847 seeing picture vs. reading a sentence.



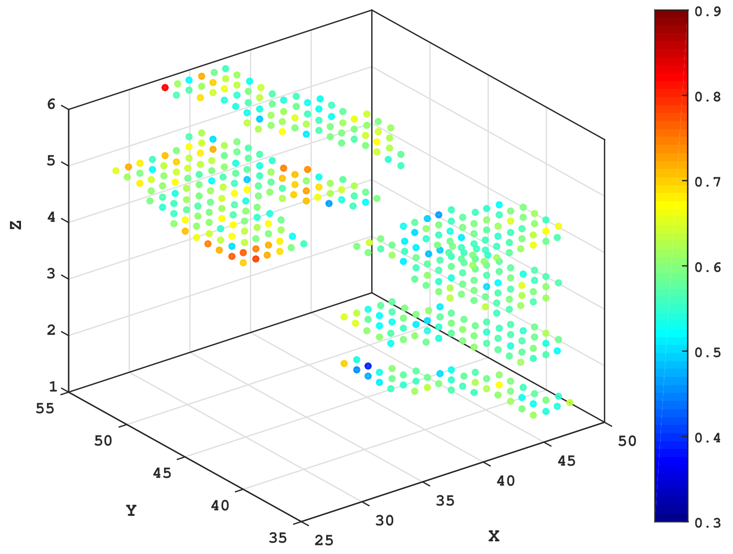
(a) ‘seeing a picture’ first during trial



(b) ‘seeing a picture’ second during trial



(c) ‘reading a sentence’ first during trial



(d) ‘reading a sentence’ second during trial

Figure 6.7.: Comparative analysis of spike frequencies of the subject 07510 seeing picture vs. reading a sentence.

6.7 Summary and Conclusion

In this Chapter, the focus was on using temporal encoding as a framework to concisely represent large volumes of data by spike-timings. By doing so, the existing discriminatory spatio-temporal information was preserved. In this regard, apart from using the existing temporal encoding techniques, a novel temporal encoding framework was formalised and a specific encoding algorithm for fMRI data, called GAGamma, was proposed. The experimental result on benchmark fMRI dataset shows the superiority of the temporal encoding algorithms, such as GAGamma and BSA, to succinctly represent the discriminatory information in the compressed encoded spike space without losing any appreciable amount of information. Thus, it achieves comparable or superior pattern recognition performance. It can be argued that the flexibility of the proposed encoding framework lies in its ability to inject known structure information about the data source and thus, provide the compression/encoding algorithms sufficient redundancy to represent the large dataset in an optimally concise manner.

6.8 Contributions and Publications

Contributions

1. A generalised *a priori* knowledge driven optimisation framework for spike-time encoding of continuous data was formalised.

2. To elaborate the characteristics of this framework, a realisation of the proposed framework, namely GAGamma, was proposed for the purpose of encoding fMRI data using *a priori* knowledge of the data source.

3. The proposed encoding algorithm was applied and compared with state of the art encoding algorithms on a benchmark pattern recognition problem involving fMRI data. The results showed, in general, the uniqueness of the proposed temporal encoding framework not only lies in its ability to significantly compress the data, but also in keeping the discriminatory information intact, which is extremely useful for pattern recognition tasks.

Publications

1. **Sengupta, N.**, & Kasabov, N. (2017). Spike-time encoding as a data compression technique for pattern recognition of temporal data. Information Sciences, 406, 133-145.

2. **Sengupta, N.**, Scott, N., & Kasabov, N. (2015). Framework for knowledge driven optimisation based data encoding for brain data modelling using spiking neural network architecture. In Proceedings of the Fifth International Conference on Fuzzy and Neuro Computing (FANCCO-2015) (pp. 109-118). Springer, Cham.

7 Orientation Influence Driven Spike-Time Dependent Plasticity Learning: A Novel Unsupervised Learning Algorithm for Integrating Spatial, Temporal and Orientation Information

7.1 Introduction

In the recent past, non-invasive brain data collection techniques such as functional magnetic resonance imaging (fMRI), electroencephalography (EEG), diffusion tensor imaging (DTI) and others have made significant contributions to understanding various structural and functional properties of the human brain. There has been consistent development in data sampling technology over the past few years which has enabled simultaneous sampling of multiple modalities of brain data while a subject performs or does not perform a task. This provided an opportunity to perform pattern recognition using large quantities of such data. It is evident that each data modality provides a unique but limited perspective of the brain. In the past, these data modalities were used independently for pattern recognition and overlooked the joint information present in the data (Sui, Adali, Yu, Chen & Calhoun, 2012). Algorithms with the ability to extract and integrate relevant information from various data sources into a single model are crucial not only for predictive modelling but also in terms of understanding the spatio-temporal relationships within the data.

Structural dysconnectivity, as measured by DTI, has been demonstrated in several psychiatric disorders and has been shown to reflect functional dysconnectivity in some cases (Greicius, Supekar, Menon & Dougherty, 2009; Stephan, Friston & Frith, 2009). In accordance with these theories, it would be appealing to incorporate dysconnectivity information into any algorithm that is designed to classify or predict outcomes in people with psychiatric disorders. This Chapter discusses the steps that have been undertaken to develop a new algorithm that can incorporate orientation information from DTI along with the EEG/fMRI activity data as a data fusion approach. The most commonly used method of integrated data analysis for this kind of problem is by converging evidence (Horwitz & Poeppel, 2002). Typically, each data type is analysed separately, and the results from other analysis that support one's findings are discussed. Horwitz and Poeppel (2002) also put forth discussions about an alternative data fusion analysis called computation neural modelling. This is done by creating biologically realistic neural network models, where each network simulates data of a certain type and is compared with observed data. One major setback of this paradigm of data analysis is that the hypothesis driven neural network model is built under several assumptions for simulated data generation. Hence, it would be difficult to know whether a disagreement between observed and simulated data is due to the assumptions in the model, or simply wrong. A comprehensive review of the research in the direction of multi-modal brain data analysis is summarised by Sui et al. (2012). Some prominent work includes integration of fMRI/EEG (Valdes-Sosa et al., 2009), fMRI/MEG (Plis et al., 2011), fMRI/DTI (Stämpfli et al., 2008; Kleiser, Staempfli, Valavanis, Boesiger & Kollias, 2010) and fMRI/-gene expression (Yang, Liu, Sui, Pearlson & Calhoun, 2010). There is a third alternative for multi-modal data integration known as data fusion, which is a more data driven approach. Direct data fusion encompasses the technique of directly fusing multiple datasets using statistical and machine learning algorithms. The data-driven methods span across the domain of the combined blind source separation techniques such as Independent Component Analysis (Calhoun, Liu &

Figure 7.1.: fMRI scanning device (source NIBIB (2017)).

Adal, 2009; Teipel et al., 2010) and its variants, multi-modal Cross-Correlation Analysis (N. M. Correa, Li, Adali & Calhoun, 2008; N. M. Correa, Eichele, Adal, Li & Calhoun, 2010; N. Correa, Li, Adali & Calhoun, 2008), Partial Least Squares (K. Chen et al., 2009), and others.

7.2 fMRI

Functional magnetic resonance imaging (fMRI) is a form of magnetic resonance imaging that takes advantage of magnetic susceptibility artefacts caused by the deoxygenated haemoglobin in the brain. Magnetic susceptibility measures the magnetic properties of the interaction between a tissue or other substance and the in-scanner magnetic field strength. Magnetically susceptible materials distort the homogeneity of a magnetic field: materials with negative magnetic susceptibility are known as diamagnetic, and those with positive magnetic susceptibility are referred to as paramagnetic. Introduction of a paramagnetic substance, such as deoxyhaemoglobin, into the scanner magnetic field causes variability in field strength, spin dephasing, geometric distortion and loss of signal; fMRI exploits this property by measuring changes in the relative ratio of oxygenated (diamagnetic) to deoxygenated (paramagnetic) haemoglobin in the blood.

fMRI measures the haemodynamic response to neuronal excitation and is therefore a secondary measure of neuronal activity. As the metabolic demands of neurons increase (as observed during task performance), astrocytes are signalled to produce prostaglandin E2 and epoxyeicosotrienoic acids, which diffuse to arteriolar smooth muscle and cause vasodilation (Hamilton, Attwell & Hall, 2010). Independently, adenosine, a breakdown product of adenosine triphosphate (ATP) produced during periods of high metabolic demand signals pericytes (contractile cells surrounding capillaries) to relax, permitting increased blood flow through capillaries (Hamilton et al., 2010). This increased blood flow delivers high concentrations of oxygenated haemoglobin and glucose to the activated region, increasing the ratio of oxygenated to deoxygenated haemoglobin. As discussed above, deoxyhaemoglobin is paramagnetic and causes dephasing and signal loss in the MR image. Increasing the ratio of oxyhaemoglobin to deoxyhaemoglobin reduces signal loss, because oxyhaemoglobin is diamagnetic. It is this decreased signal loss, corresponding to the peak of the haemodynamic response function (HRF), that is measured during fMRI experiments. This is referred to as the blood oxygen-level dependent (BOLD) signal. Importantly, the HRF produces only a $1 - 2\%$ change in signal following a single stimulus. For this reason, it is required that data be collected over a long period of time so that the signal to noise ratio can be improved. fMRI may be acquired during performance of a cognitive task or during rest. During rest, the brain exhibits patterns of spontaneous activity that coincide with those present during task performance (S. M. Smith et al., 2009), making resting-state fMRI (rs-fMRI) an excellent tool for investigating functional brain connectivity. In a standard setup, fMRI data is collected using an MRI device such as the one shown in Figure 7.1. Each fMRI scan is visualised as sequence of 2D slices of a 3D image. The pixel colours represents the indirect measure of neural activity. An example of fMRI data visualisation is shown in Figure 7.2. Each image in Figure 7.2 is showing a single slice horizontal view of the 3D image.

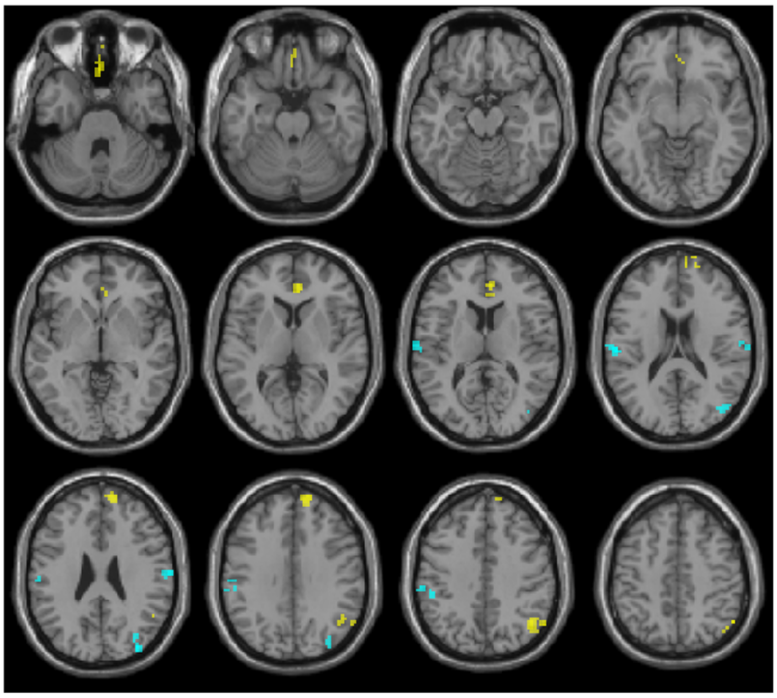


Figure 7.2.: Example of fMRI data represented as a sequence of images (source Quarantelli et al. (2013)).

7.3 DTI and Tractography

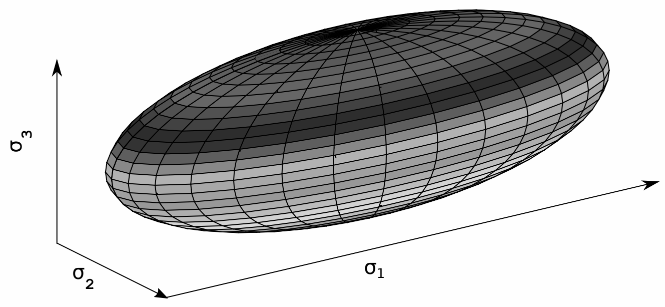
DTI is a magnetic resonance imaging technique that uses immense gradient amplitudes together with spin-echo or gradient-echo EPI sequences to provide a measure of the relative diffusion of molecules in tissues. In DTI, each voxel of the MR image has one or more pairs of parameters: a rate of diffusion and a preferred direction of diffusion described in terms of three-dimensional space for which that parameter is valid.

The diffusion of hydrogen within a voxel is described by the diffusion tensor D as represented by Equation 7.1. D is a reciprocal matrix, with six independent scalar elements (D_{xx}, D_{yy}, D_{zz}, D_{xz}, D_{xy}, D_{yz}). The six unique elements of the tensor are the coefficients of the ellipsoid equation given by $D_{xx}x^2 + D_{yy}y^2 + D_{zz}z^2 + D_{yx}yx + D_{zx}zx + D_{xy}xy = 1$. This equation can also be expressed in terms of orthogonal Eigenvectors σ and their respective eigenvalues λ. These variables describe the elements of the diffusion ellipsoid as shown in Figure 7.3.

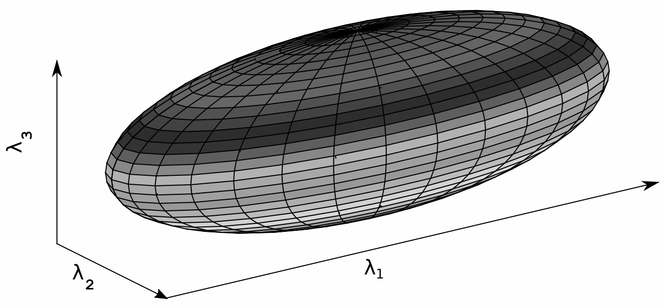
$$D = \begin{pmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{xy} & D_{yy} & D_{yz} \\ D_{xz} & D_{yz} & D_{zz} \end{pmatrix} = \begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \begin{pmatrix} \sigma_1 & \sigma_2 & \sigma_3 \end{pmatrix} \tag{7.1}$$

Isotropic diffusion within a voxel causes the diffusion ellipsoid to take a spherical shape with $\lambda_1 = \lambda_2 = \lambda_3$. In contrast, restriction of diffusion in certain directions leads to an elevated eigenvalue coupled with the principal diffusion direction as opposed to those corresponding to the secondary and tertiary diffusion direction. From the tensor equation depicted in Equation 7.1, a number of metrics corresponding to the properties of diffusion in the voxel can be derived. λ_1 corresponds to the primary direction of diffusion (principle eigenvector; σ_1) is referred to as the axial diffusivity (AD). Radial diffusivity (RD) is the mean of λ_2 and λ_3 and reflects the diffusion behaviour transverse to the axonal path.

From the raw data, it is possible to determine fractional anisotropy but not the fibre orientation. Diffusion tensor information can also be used to reconstruct white matter bundles in the brain. This technique is termed tractography. Fibre tractography is a very elegant method for delineating individual fibre tracts from diffusion images. Tractography uses diffusion orientation information from tensor imaging to calculate the direction of fibre bundles in-vivo. In deterministic, or streamline tractography, the local tract direction is defined by the major eigenvector of the diffusion tensor (Alexander, 2010). This causes issues in voxels with crossing, kissing or splitting fibres. However, as the algorithm is only capable of estimating one fibre orientation (Alexander, 2010), probabilistic tractography addresses these issues by estimating the orientations of two or three different fibre populations within a single voxel (Behrens, Berg, Jbabdi, Rushworth & Woolrich, 2007). Thereafter, at every voxel, the algorithm estimates the most probable fibre orientation and also provides a distribution representing the probability that every other orientation lies along that fibre. This process is repeated many times, each time using a slightly different orientation (according



(a) The primary, secondary and tertiary diffusion directions described by the Eigenvectors of Equation 7.1.



(b) The primary, secondary and tertiary Eigen values described in Equation 7.1.

Figure 7.3.: Visual intuition of the ellipsoid representation of a diffusion tensor voxel.

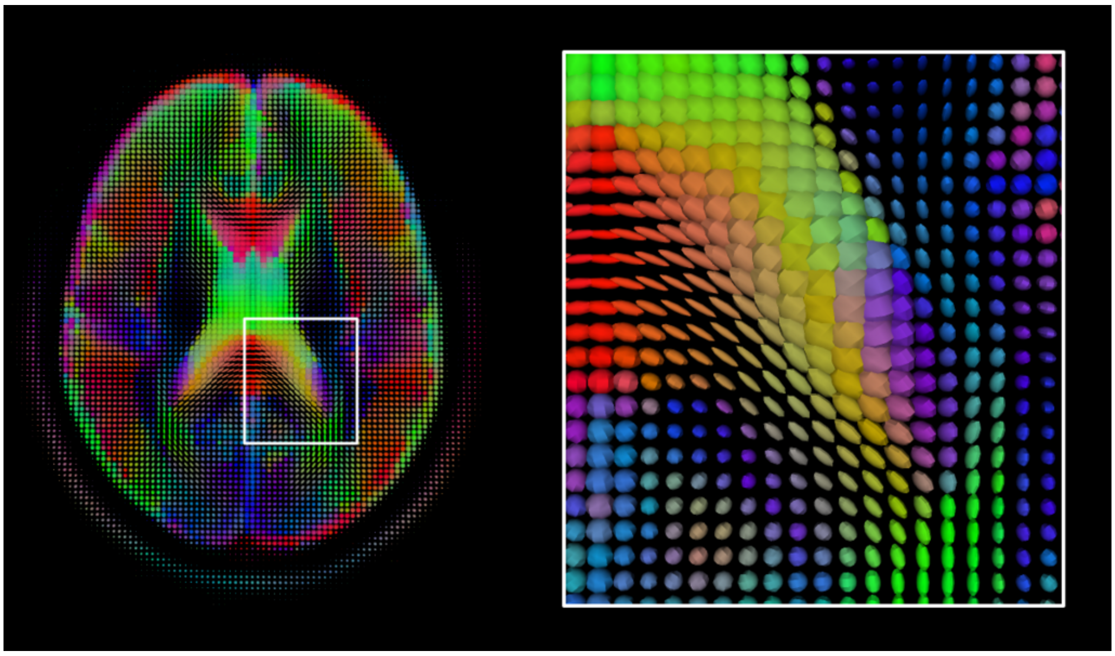


Figure 7.4.: Orientation information from DTI image. Left image shows an axial slice of a single subject's DTI data, registered to structural and MNI standard space. The Right image shows a close-up of the right posterior corpus callosum. Directions corresponding to each colour are as follows: Red - left to right/right to left, green - anterior to posterior/posterior to anterior and blue - superior to inferior/inferior to superior (source "Medical image computing" (n.d.)).

to its likelihood). The integration of all estimates provides a collective measure of connection probability along each tract (Behrens et al., 2007; Behrens et al., 2003). Figure 7.4 shows an example of a diffusion MRI image of the human brain.

7.4 NeuCube Architecture for Integrating Spatial, Temporal and Orientation Information

7.4.1 Formalisation of the Machine Learning Problem

The machine learning problem here is to learn a functional mapping of $f(D_{seq}, D_{stat}) \rightarrow C$, given a set of training samples $< D_{seq}, D_{stat}, C >$, so that the spatial temporal and orientation information present in D_{seq} and D_{stat} are used to not only increase prediction performance, but also impart robustness to the model. Here D_{seq} and D_{stat} represents the fMRI/EEG and DTI data respectively.

7.4.2 NeuCube Personalised SNNc Architecture

Here, a personalised SNNc based NeuCube architecture is proposed for the purpose of learning from multi-modal information. The personalised SNNc based NeuCube architecture is a modification of the NeuCube architecture described in Section 4.4 and depicted in Figure 4.6. Before proposing the modified architecture, it is necessary to elaborate certain characteristics of the SNNc in a canonical setting.

Saturation behaviour of canonical NeuCube SNNc: The classical NeuCube architecture shown in Figure 4.6 is made up of a single instance of encoder layer, SNNc layer and supervised readout layer. The instances of these layers are initialised before the training process and data is passed through sequentially over time for: (1) Encoding layer to continuously encode the data into spike-timings; (2) SNNc layer to digest the data and update its state; and (3) supervised readout layer to digest the transformed spike-train and create instances in the data space for the K-NN algorithm to act on. NeuCube is a pattern recognition system that learns continuously from data. During the course of development of NeuCube, it is observed that in a vanilla setting, learning continuously from streaming input data has some undesirable effects on the run-time efficiency and performance over time. Upon analysis, it was found this behaviour relates to the single instantiation of the NeuCube SNNc layer.

To further describe this behaviour, the spike density of SNNc can be defined as the number of neurons that fire at a certain time instance. It is observed that as input data is fed into the SNNc over time, the probability of the neurons in the SNNc firing increases. The spiking neuron of course has a mechanism, such as refractoriness, to avoid continuous spiking; however, over time, periodicity in spike density cannot be avoided. To demonstrate this statement, a set of experiments were conducted using synthetically generated spike data.

In the experimental setup, a small SNNc was initialised having 27 neurons. The neurons were spatially distributed in a 3×3 grid. The network of neurons were connected following the SWC algorithm ($r_{thr} = 0.3$). For input data, random input spikes (with spiking probability 0.3) data $D_{seq} = \{0, 1\}^{200 \times 4}$ were synthetically generated for each sample. Table 7.1 enlists three instances of the experiment wherein the canonical SNNc unsupervised learning algorithm was run using varying sample sizes and hyperparameter values. At the end of the SNNc simulation, the spike-rate over time (or samples as samples are presented over time) was plotted. It can be observed that irrespective of the hyperparameter settings, a periodicity/total saturation in spike-rate is seen in the SNNc. As a consequence, the spiking patterns after a certain number of sample presentations becomes predictable and does not relate to the variabilities in the sample any more. This fact can be verified from the 'sample distance matrix' column where the pairwise hamming distance matrix was plotted across the sample output of the SNNc. The colours (red is high and blue is low) in the plot represents the hamming distance between two samples. It must be noted that the hamming distance between samples not only considers how many times SNNc has spiked for a sample, but also considers when the neurons have spiked. It can be observed there is a lot of variability in the similarity values initially, but as more samples are presented, the spike patterns become more and more similar leading to very small values of dissimilarity.

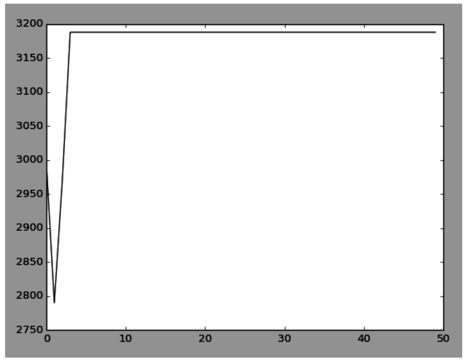
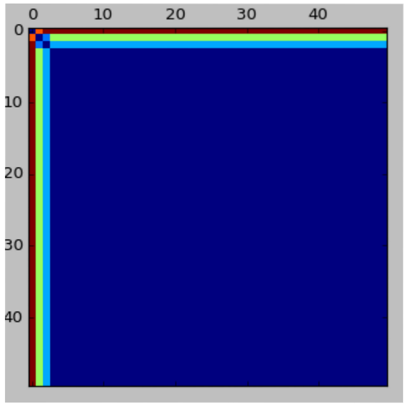
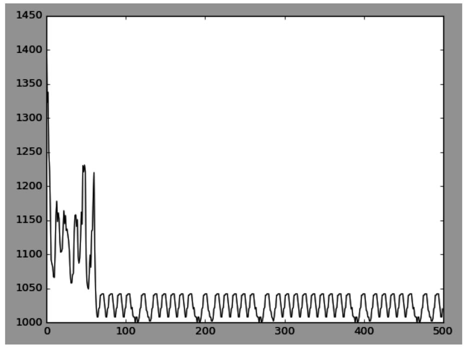
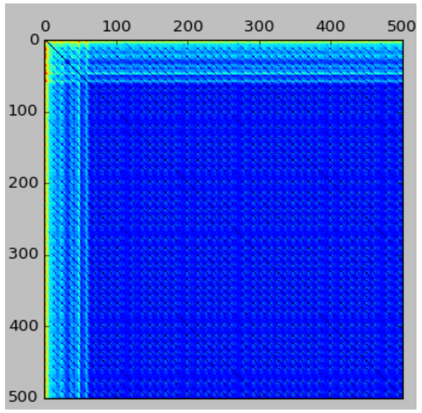
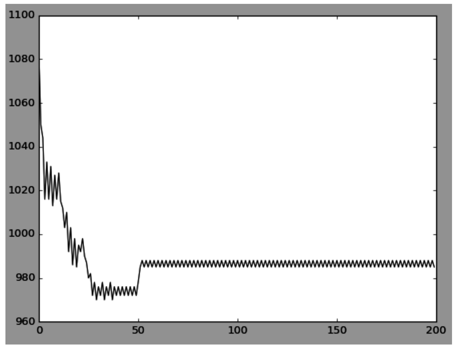
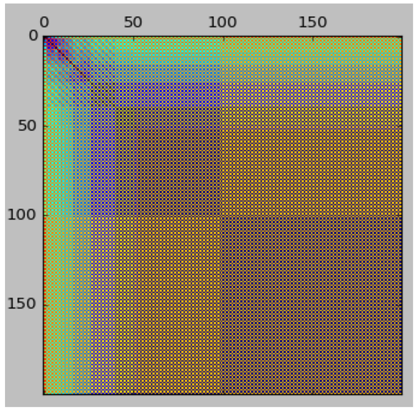
The point of including the SNNc layer is to improve pattern formation by expanding it into higher dimensional space. However, the behaviour described above does have a significant impact on pattern generation capabilities of the canonical NeuCube SNNc beyond a certain time. This can be defined as saturation threshold time. There are two directions where research can traipse in order to resolve or further analyse this issue:

- Improve the SNNc mechanisms and principles to keep it within sub-threshold limit. In this setting, a single instance of the SNNc can be simulated life-long without reduction in pattern generation abilities.

- Minimise the impact of saturation by using multiple instances of the SNNc and control the sub-threshold state using the hyperparameter values. This approach was used in this work.

The personalised SNNc based NeuCube architecture is depicted in Figure 7.5. In the personalised SNNc approach, usage of multiple instances of SNNc was proposed. In this approach one instance of SNNc is simulated per sample along with a single instance of the encoding layer and supervised readout layer. In this setup, each sample of input data is fed into a unique pre-initialised personal SNNc. The personal SNNc acts as a filter which evolves over time to capture the spatio-temporal relationships within a sample in the synaptic strengths. The knowledge of finite time horizon means the SNNc instance can be kept in a sub-saturation state by controlling the hyperparameter ranges. The other advantage of the personalised SNNc architecture is the ability to avoid the sequential sample presentation bias that may exist if all the samples are fed into a single instance of the SNNc. This means the sequence of the sample presentation does not have any impact on spike patterns generated by the SNNc. The other important aspect of the proposed personalised SNNc architecture is the ability to incorporate multi-modal nature of the input data which will be discussed further.

Table 7.1.: Experimental results showing saturation characteristics of canonical NeuCube SNNc.

# samples	υ_{thr}	η_{thr}	κ	sample vs spike rate	Hamming distance matrix	saturation type
50	0.5	0	0.1			total
500	10	0	0.1			periodic
200	1	15	0.1			periodic

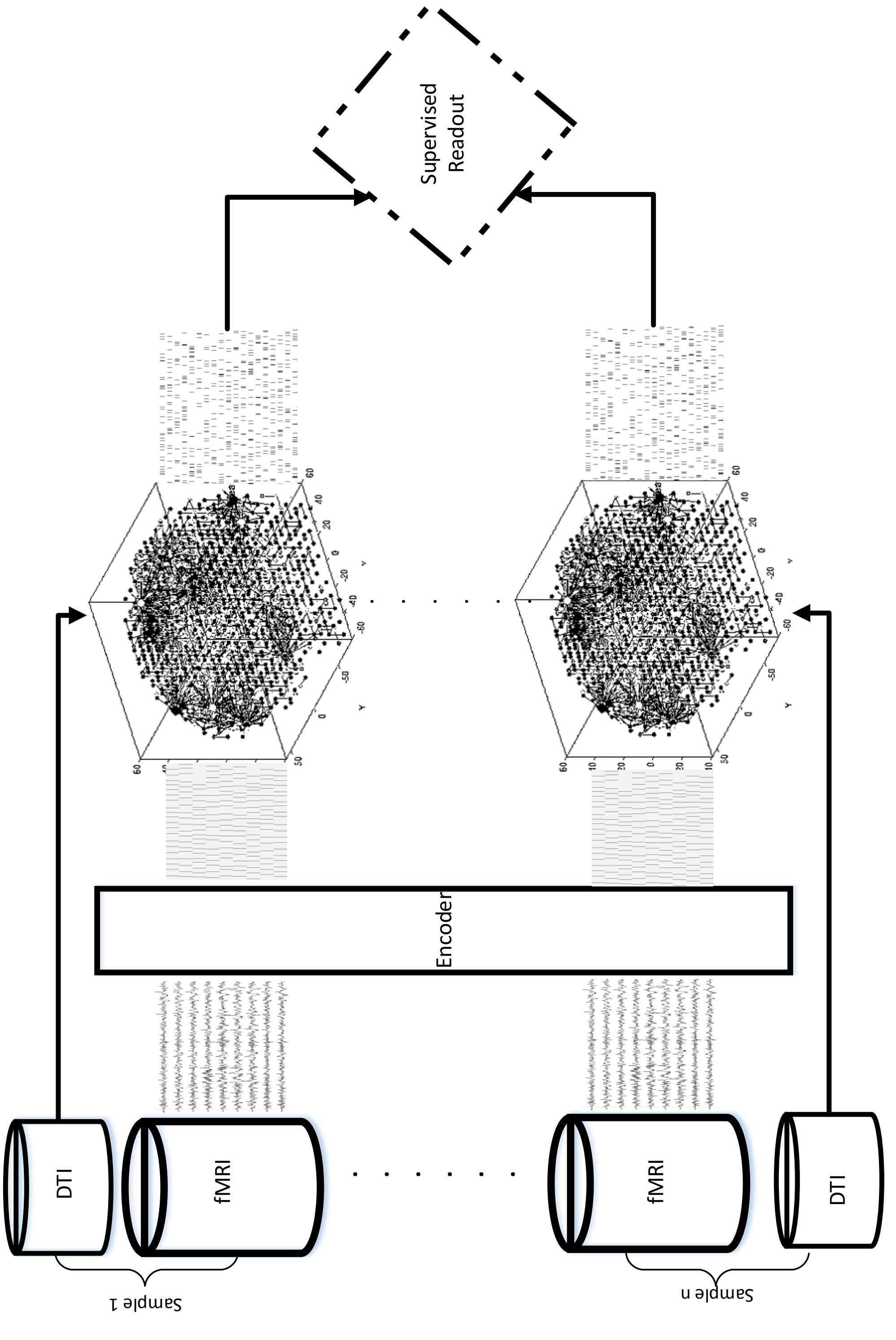


Figure 7.5.: Proposed NeuCube personalised SNNc architecture.

7.4.3 Multi-modal Information Integration in SNNc Using Orientation and Spike-time Data

In this Section, the proposed adaptations of the canonical SNNc unsupervised learning algorithm mentioned earlier in Section 5.3.4, for the purpose of fusing dynamic spatio-temporal and static orientation information from brain data, will be discussed.

SNNc Architecture, Mapping and Initialisation

The SNNc architecture, as described elaborately in Section 5.3.1, are spatially arranged (in three dimensions) set of neurons, partially connected by recurrent synapses forming a directed incomplete and acyclic graph $G = \{M, C, W\}$. The symbols and formalisations are consistent with Chapter 5 unless otherwise stated. The neurons within the network are input or spiking types. The spatial arrangement of the neurons follow a natural ordering and in the case of brain data, integration takes a brain-like shape. The connections of the network are established following the standard SWC (N. Kasabov et al., 2014) algorithm.

Neuron Dynamics

The SRM model has been used for implementing the leaky integrate and fire like behaviour of a neuron. The SRM neuron model is used to adapt the membrane potential $v(t)$ of a neuron over time. This neuron model specifies the membrane potential as the sum of: (1) temporal integration of the PSPs and (2) the refractoriness. The PSP kernel ϵ_0 is a function of $t - t^f$, representing the PSP trace over time generated by the firing of neuron j over time. τ_m is known as the membrane constant which controls the decay rate of the PSP. In the present experiments, a constant $\tau_m = 0.5$ has been used. This means the influence of a pre-synaptic spike diminishes from maximum to minimum within approximately five discrete time intervals. For a more detailed explanation and behaviour of the neuron dynamics, see Section 5.3.2.

$$
\begin{aligned}
v_i(t) &= v_{rest} + \sum_{j \in \mathrm{T}_i} w_{ji} \sum_{t_j^f \in F_j} \epsilon_0(t - t_j^f) + \eta(t - t_i^f) \\
\epsilon_0(t - t_j^f) &= \exp(-\tfrac{t - t_j^f}{\tau_m}) \mathcal{H}(t - t_j^f) \\
\mathcal{H}(t - t_j^f) &= \begin{cases} 1, & \text{if } t - t_j^f \geq 0 \\ 0, & \text{otherwise} \end{cases} \\
\eta(t - t_i) &= \begin{cases} -\infty, & \text{if } t - t_i^f < \eta_{thr} \\ 0, & \text{otherwise} \end{cases}
\end{aligned} \tag{7.2}
$$

Python implementation: The neuron model dynamics has been implemented in python $v3.5$. The neuron model dynamics were implemented as part of the $LeakyIntegrateAndFire$ class shown below. The $simulate()$ method is implemented to simulate the neuron dynamics using the input $spike\_train$ and pre-synaptic connection weights $weight$ and hyperparameter tau.

```python
class LeakyIntegrateAndFireNeuron:
  def __init__(self, predecessor_count, potential_threshold,
    refractory_threshold, potential_resting,
    spike_history_length):
     self.predecessor_count = predecessor_count
     self.refractory_threshold = refractory_threshold
     self.potential_threshold = potential_threshold
     self.potential_resting = potential_resting
     self.spike_history_length = spike_history_length
     self.time = 0
     # initialise neuron state
     self.potential = self.potential_resting
     self.refractory_state = 0
```

```python
        self.potential_state_history = [self.potential]
        self.spike_history = np.zeros((self.spike_history_length, self.predecessor_count), dtype=int)

    @staticmethod
    def spike_response_kernel(current_time, spike_time, tau):
        time_diff = current_time - spike_time
        potential = np.exp(-time_diff / tau)
        return potential

    def simulate(self, spike_train, weight, tau):
        self.time += 1
        self.spike_history = np.delete(self.spike_history, 0, axis=0)
        self.spike_history = np.append(self.spike_history, [spike_train], axis=0)
        current_time = self.spike_history_length
        potential = self.potential
        spike = 0
        if self.refractory_state == 0:
            for k in np.arange(current_time):
                count = np.count_nonzero(self.spike_history[k, :])
                indices = np.nonzero(self.spike_history[k, :])
                for i in range(count):
                    w = weight[indices[0][i]]
                    potential += w * self.spike_response_kernel(current_time=current_time - 1, spike_time=k, tau=tau)
            self.potential = potential
            self.potential_state_history.append(self.potential)
            if self.potential > self.potential_threshold:
                self.refractory_state = self.refractory_threshold
                self.potential = self.potential_resting
                spike = 1
        else:
            self.refractory_state = max(0, self.refractory_state -
 1)
            self.potential = max(self.potential_resting, self.
potential)
            self.potential_state_history.append(self.potential)
    return spike
```

code snippet 7.1: Python code for NeuCube modified SRM neuron model

Adaptation of Synaptic Strengths of the SNNc

The unsupervised learning algorithm in the SNNc is the most important aspect of the proposed architecture for integrating multi-modal information. In a neural network paradigm, learning is achieved through the synaptic strength updates of the network. The learning behaviour of the SNNc can be explained using the learning model of a single spiking neuron. Considering the single neuron architecture, as shown in Figure 5.1, the unsupervised learning problem is the a scheme of updating the w_{ji}'s by $\Delta w_{ji}(t)$ over the simulation time T. In a recurrent SNNc layer, the aim is to learn dynamic influence from dynamic data (fMRI) and static orientation influence from static data (DTI).

Dynamic Influence from fMRI/EEG

In the majority of the machine learning applications, models are trained on static data, where a sample is represented by a vector of numbers $\mathbf{d} = \{d_1, d_2, \cdots\}$, where each of the elements within the tuple d represents the scalar value of a particular feature. However, in this case with fMRI or EEG data, a sample is represented by a matrix $\mathbf{D_{seq}} = \{\mathbf{d_1}, \mathbf{d_2}, \cdots, \mathbf{d_n}\}$, where $\mathbf{d_i} = \{d_i(1), d_i(2), \cdots d_i(t)\}$. The sample representation is not only multidimensional but also is ordered in time (sequence). Learning from these types of data in machine learning is known as sequence learning and techniques, like the hidden Markov model and flavours of recurrent neural network have shown promise in learning from such sequences. In this instance, an unsupervised sequence learning algorithm will be described that is within the NeuCube SNNc layer and utilises the sequential information as part of its learning mechanism. The sequential information within the SNNc architecture is named as the dynamic influence and is denoted by ϕ.

The dynamic influence from the spike-time data using the STDP learning algorithm was modelled. As discussed elaborately in Section 5.3.3, STDP is a temporally asynchronous form of Hebbian learning ("neurons wire together, if they fire together") (Hebb, 1949) induced by the temporal correlation of the spikes. Due to the online nature of the learning within the SNNc layer, the canonical online formulation of the STDP learning rule have been used. Sjöström and Gerstner (2010a) proposed the on-line STDP update rule by modifying the canonical STDP rule. In the on-line setting, the ϕ_{ji} is calculated every time a neuron i fires a spike or receives a spike from neuron j. Equation 7.3 formalises the weight update rule that was used to calculate the dynamic influence. The first term in the right hand side of Equation 5.7 corresponds to the LTP update and is calculated when neuron i fires a spike at time t. The second term is the LTD update and is calculated when neuron i receives a spike from neuron j at time t.

$$\phi_{ji}(t) = \sum_f \kappa_+ \exp(-(t - t_j^f)) - \sum_g \kappa_- \exp(-(t - t_i^g)) \tag{7.3}$$

Python implementation: The python implementation of Equation 7.3 is given below. The $STDP()$ method takes the pre- and post-synaptic spike history t_j^fs and t_i^gs along with the learning rate hyperparameter κ to generate the dynamic influence phi.

```
def STDP(pre_synaptic_spike_history,
    post_synaptic_spike_history, kappa):
    assert isinstance(pre_synaptic_spike_history, list)
    assert isinstance(post_synaptic_spike_history, list)
```

```
if len(pre_synaptic_spike_history) != len(post_synaptic_spike_history):
raise ValueError("Length mismatch of pre and post synaptic spike history!!")
pre_synaptic_spike_energy = 0
post_synaptic_spike_energy = 0
importance_of_LTP = 0.3
importance_of_LTD = 0.3
time_history = len(pre_synaptic_spike_history)
pre_synaptic_spike_history = np.asarray(pre_synaptic_spike_history)
post_synaptic_spike_history = np.asarray(post_synaptic_spike_history)
"""
calculation of LTD
"""
if pre_synaptic_spike_history[time_history - 1] == 1:
   post_spike_indices = np.nonzero(post_synaptic_spike_history)[0]
   k = kappa * np.exp(-(1 - importance_of_LTD) * ((time_history - 1) - post_spike_indices))
   pre_synaptic_spike_energy = np.sum(k)

"""
calculation of LTP
"""
if post_synaptic_spike_history[time_history - 1] == 1:
   pre_spike_indices = np.nonzero(pre_synaptic_spike_history)[0]
   k = kappa * np.exp(-(1 - importance_of_LTP) * ((time_history - 1) - pre_spike_indices))
   post_synaptic_spike_energy = np.sum(k)
phi = post_synaptic_spike_energy - pre_synaptic_spike_energy
```

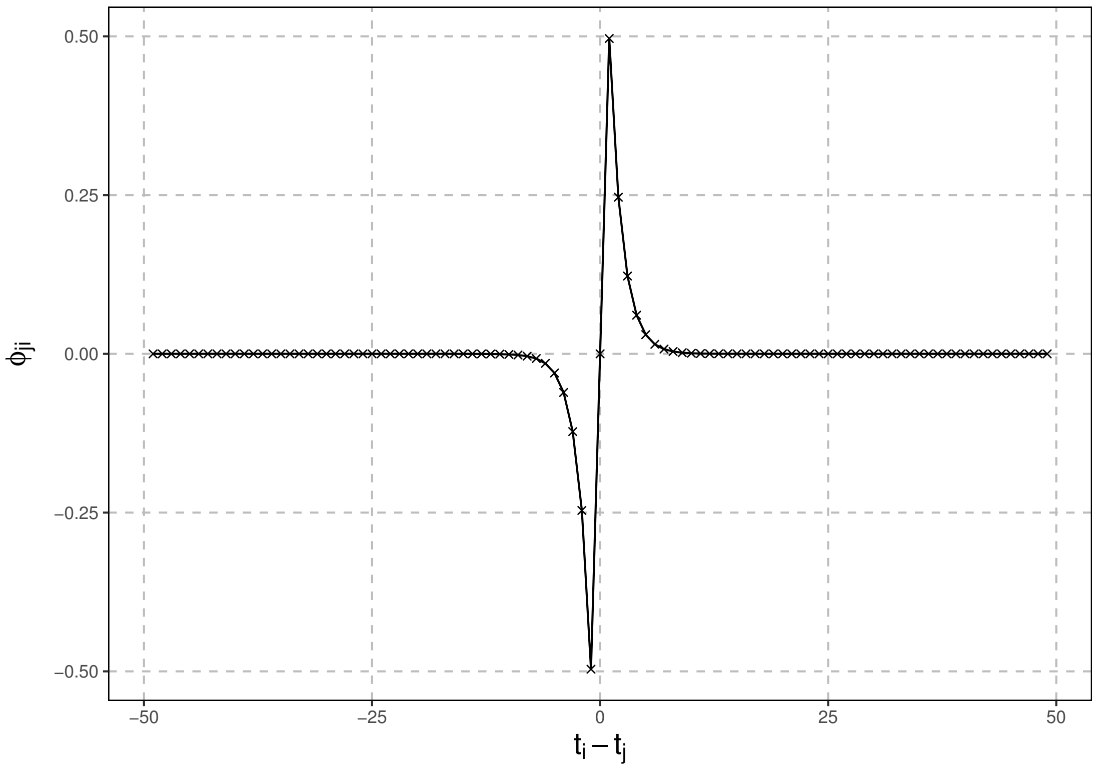


Figure 7.6.: Plot of the STDP weight update as a function of the relative time difference of the pre and post synaptic spikes. The data for this plot was generated using code snippet 7.2 with hyperparameter $\kappa_{+} = \kappa_{-} = 0.5$.

```
    return phi
```

code snippet 7.2: Python code for calculating dynamic influence

Static Orientation Influence from DTI Tractography Data

The present study has used the DTI data in the form of orientation vectors representing mean orientation of the fibre tract at different voxel locations. The process of generating the orientation data from a DTI image is described later in Section 7.6.3. The orientation vector of a sample DTI image is represented by a matrix $\mathbf{D_{stat}} \in \mathbb{R}^{|N| \times 3}$, where each feature/voxel is made up of a 3D vector describing the primary orientation of the fibre in the Cartesian coordinate system.

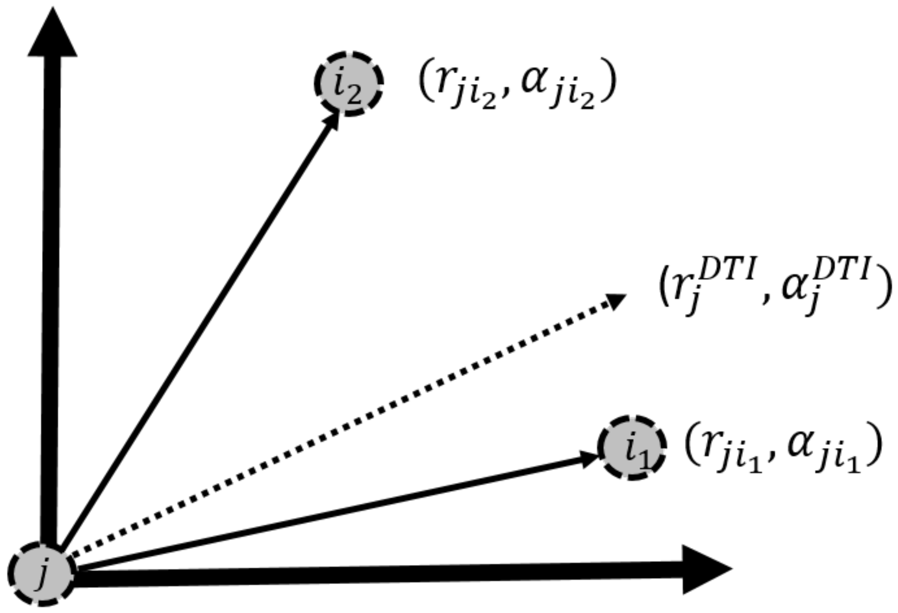

i_2
$(r_{ji_2}, \alpha_{ji_2})$
$(r_j^{DTI}, \alpha_j^{DTI})$
i_1
$(r_{ji_1}, \alpha_{ji_1})$
j

Figure 7.7.: Example of a pre-synaptic neuron j connected to two post synaptic neurons i_1 and i_2. Each neurons spatial location is defined by the polar coordinates (r, α).

Here, the interest lies in establishing a learning rule that does not only utilises dynamic data influence as described in Section 7.4.3, but can also make use of the static orientation influence from the DTI data. The intuition behind the orientation influence can be explained again by a small SNNc architecture consisting of three neurons as shown in Figure 7.7. The figure shows a single pre-synaptic neuron j connected to two post synaptic neurons i_1 and i_2. The important aspect to note here is that the neurons in this diagram have spatial allocations contrary to the one in Figure 5.1. The location of the neurons are defined by the radial and the angular coordinates in the polar coordinate system. Now, the intent is to measure the orientation influence ψ of neuron j on neurons $\{i_1, i_2\}$. The neuron j will be referred to as the pivot neuron. The orientation vector of the pivot neuron (from DTI data) is be represented by (r_j, α_j). The orientation influence of the pivot neuron on the post-synaptic neurons $\{i_1, i_2\}$ is defined by the angular proximity of the pivot neuron's DTI orientation vector to the synaptic orientations of the neuron pairs. In that way, as per the hypothesis, the pivot neuron wields a stronger angular influence on the neurons that lie in closer angular proximity to the orientation vector of the pivot neuron. Therefore in Figure

7.7, the orientation influence of neuron j can be arranged as $i_1 > i_2$ due to the angular proximity of i_1 and j being greater than i_2 and j.

Even though a 2D vector space has been used for explaining the intuition of angular influence, the SNNc neurons reside in a 3D space. The intuition can be extended to the 3D vector space by adding another dimension in the coordinate system. In 3D, the spherical coordinates of a point are given by (r, α, β), where r is the scalar distance of the point from the centre, α and β are the elevation and azimuth angle from the centre. A Gaussian radial basis (GRB) kernel has been utilised to realise the elevation and azimuth orientation influences given the elevation and azimuth data of the neurons. ψ is calculated as the mean of azimuth and elevation influences between pre- and post-synaptic neurons j and i according to Equation 7.4.

$$\begin{aligned}\psi_{ji} &= \frac{\psi_{ji}^{\alpha}+\psi_{ji}^{\beta}}{2}\\ \psi_{ji}^{\alpha} &= \exp(\frac{||\alpha_{ji}-\alpha_j^{dti}||^2}{2\gamma^2})\\ \psi_{ji}^{\beta} &= \exp(\frac{||\beta_{ji}-\beta_j^{dti}||^2}{2\gamma^2})\end{aligned} \tag{7.4}$$

Python implementation: The python implementation of Equation 7.4 is given below. The $angular\_influence()$ method takes the coordinates of two neurons and the orientation information, and outputs the orientation influence psi.

```
def angular_influence(pivot_coordinate, sphere_coordinate,
    elevation_angle, azimuth_angle):
    assert isinstance(pivot_coordinate, np.ndarray)
    assert isinstance(sphere_coordinate, np.ndarray)
    assert isinstance(elevation_angle, float)
    assert isinstance(azimuth_angle, float)

    standard_deviation = 8
```

```python
    relative_sphere_coordinate = np.subtract(sphere_coordinate, pivot_coordinate)
    radius, elevation, azimuth = cart2sph(relative_sphere_coordinate[0], relative_sphere_coordinate[1], relative_sphere_coordinate[2])
    el_prob = radial_decay(elevation, elevation_angle, standard_deviation)
    az_prob = radial_decay(azimuth, azimuth_angle, standard_deviation)
    psi = (el_prob + az_prob) / 2
    return psi

def cart2sph(x, y, z):
    radius = math.sqrt(x ** 2 + y ** 2 + z ** 2)
    elevation = math.atan2(math.sqrt(x ** 2 + y ** 2), z)
    azimuth = math.atan2(y, z)
    elevation *= 180 / math.pi
    azimuth *= 180 / math.pi
    return (radius, elevation, azimuth)

def radial_decay(x, mu, sigma):
    return math.exp(-1 * (x - mu) ** 2 / (2 * sigma ** 2))
```

code snippet 7.3: Python code for calculating static orientation influence

The GRB kernels exponentially decay the orientation influence as the Euclidean norm $||\alpha_{ji} - \alpha_j||$ and $||\beta_{ji} - \beta_j||$ increases. The variance hyperparameter γ^2 controls the speed with which the orientation influence decays with increasing radial distance (see Figure 7.8). The overall orientation influence is calculated as the mean of the elevation and azimuth influence, as shown in Equation 7.4.

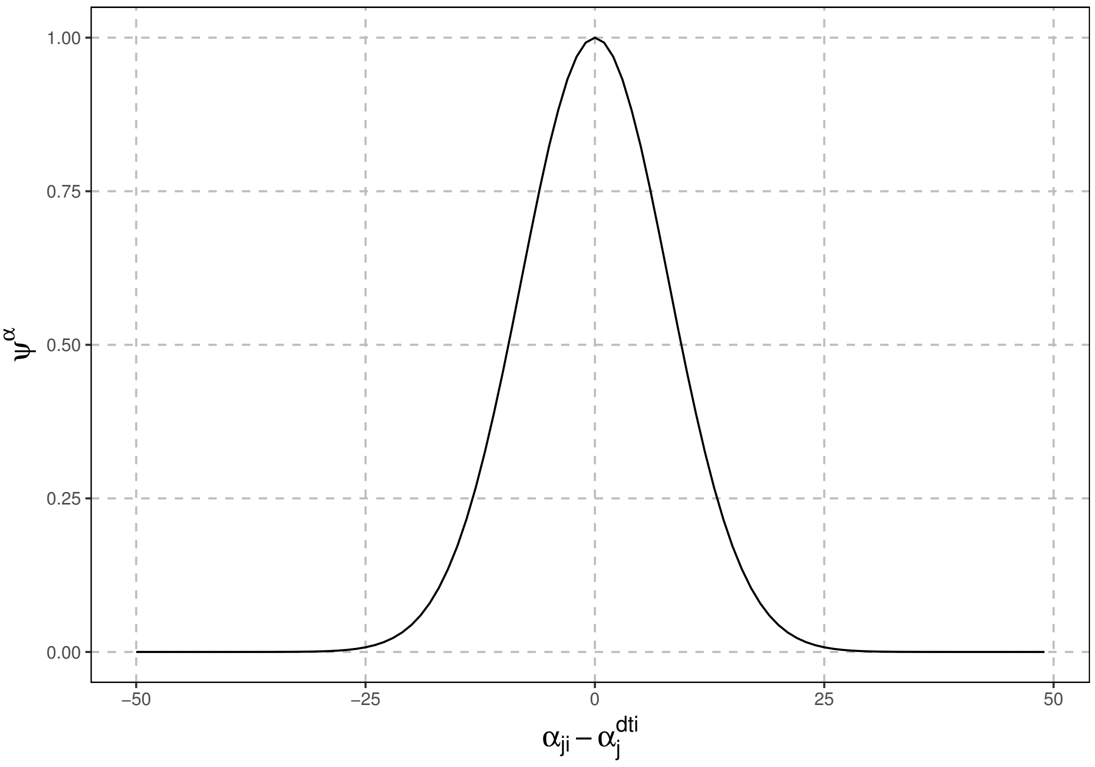


Figure 7.8.: Plot of the elevation influence ψ^{α} as a function of the radial distance $\alpha_{ji} - \alpha_j^{dti}$ and $\gamma = 8$.

7.4.4 Orientation Influence Driven STDP (oiSTDP) Learning in SNNc

Algorithm 7.1 describes the oiSTDP learning algorithm step by step. The unsupervised learning is executed on a preinitialised SNNc. The algorithm takes: (1) the dynamic data D_{seq} (fMRI/EEG) in the form of spikes; (2) the static orientation data D_{stat} as 3D orientation vectors; and (3) the coordinates of the SNNc neurons as the input. The output of the learning algorithm are the spikes in the higher dimensional space in the form of O_{seq}. Over simulation time T, the execution of the algorithm can be divided into two sequential blocks:

1. Block 1 (line 6 to 17): First, each spiking neuron in set Q is queried for the pre-synaptic neurons J_i^{spk}, that has fired a spike at time t . These connections are stored in C_{learn} for later updating the weights as per the LTD rule. Then the spiking neuron is simulated to update the membrane potential. If the neuron spikes as a result of neuron simulation (line 10),

Algorithm 7.1 oiSTDP-based SNNc learning algorithm

```
1: input: $G = \{M, C, W\}$, $D_{seq} \in \{0,1\}^{|N|\times|T|}$, $D_{stat} \in \mathbb{R}^{|N|\times 3}$, $loc \in$
   $\mathbb{R}^{|M|\times 3}$,$\{hyperparameters = v_{thr}, \eta_{thr}, \kappa\}$
2: output: $O_{seq} \in \{0,1\}^{|M|\times|T|}$
3: $O_{seq}[N,T] \leftarrow D_{seq}$
4: for $t \in T$ do
5:     initialise $C_{learn} \leftarrow \{\}$
6:     for $i$ in spiking neurons $Q$ do
7:         find firing(at time $t-1$) pre-synaptic neurons, $J_i^{spk(t)}$
8:         set $C_i^{ltd} \leftarrow (J_i^{spk}, i)$
9:         $C_{learn} + \leftarrow C_i^{ltd}$
10:        simulate $i$ as per Equation 5.4
11:        if $i$ fires a spike then
12:            $O_{seq}[i, t+1] \leftarrow 1$
13:            find pre-synaptic neurons, $J_i$
14:            set $C_i^{ltp} \leftarrow (J_i, i)$
15:            set $C_{learn} + \leftarrow C_i^{ltp}$
16:        end if
17:    end for
18:    for $c_{ji}$ in $C_{learn}$ do
19:        initialise $\psi_{ji} \leftarrow 1$
20:        if $j$ in $N$ then
21:            $(r_j^{dti}, \alpha_j^{dti}, \beta_j^{dti}) \leftarrow cart2sph(D_{stat}[j])$
22:            $(r_{ji}, \alpha_{ji}, \beta_{ji}) \leftarrow cart2sph(loc[i] - loc[j])$
23:            calculate $\psi_{ji}$ as per Equation 7.4
24:        end if
25:        calculate $\phi_{ji}$ as per Eq. 5.7
26:        set $\Delta w_{ji} \leftarrow \psi_{ji} \cdot \phi_{ji}$
27:        update $w_{ji} \leftarrow w_{ji} + \Delta w$
28:    end for
29: end for
```

a spike is added to O_{seq} and the pre-synaptic connections are again added to the variable C_{learn} for later weight update as per the LTP rule.

2. Block 2 (line 18 to 28): This block implements the weight update rule. At every time iteration t, synaptic strengths are updated for all the connections stored in C_{learn}. Lines 20 to 24 are the steps for calculating the orientation influence ψ. The function $cart2sph(.)$ takes a 3D Cartesian coordinate (x, y, z) as input and outputs the spherical coordinate (r, α, β). The following formulae are used for this conversion:
   $r = \sqrt{x^2 + y^2 + z^2}$
   $\alpha = \tan^{-1} \frac{\sqrt{x^2+y^2}}{z}$
   $\beta = \tan^{-1} \frac{y}{z}$
   The weight update Δw_{ji} for connection C_{ji} is the product of ψ_{ji} and ϕ_{ji} (line 27). In this way, the orientation influence has been used as a modulation factor of the dynamic influence. phi and psi are bound to $[-1, 1]$ and $[0, 1]$ respectively. Henceforth, $Deltaw$ is bound between $[-1, 1]$. The most important characteristic of this learning rule is, of course, the inclusion of the orientation influence along with the dynamic influence in the weight update rule formulation. The rationale behind this formulation is to bind the coincidence (STDP) and tract information together in a way such that the strongest weight update occurs between a neuron pair when (1) there is maximal coincidence between the pre-synaptic and post-synaptic firing, and (2) the orientation of a neuron pair matches the DTI orientation data of the pre-synaptic neuron. Hence, the relations observed in the synaptic strengths of the SNNc network are representative of both the spatio-temporal coincidence generated from the encoded spike sequence and the orientation information produced by the DTI data. In this way, the spatial, temporal and orientation information in the synaptic strengths are able to be captured. Figure 7.9 shows the behaviour of the oiSTDP update rule. The X and Y axes are the radial orientation distance between the neuron pair and the temporal difference between the pre and post-synaptic firing times, respectively. The figure shows a mirrored inverted half Mexican hat behaviour. It is visible that every slice across r_{ji} axis mimics the STDP

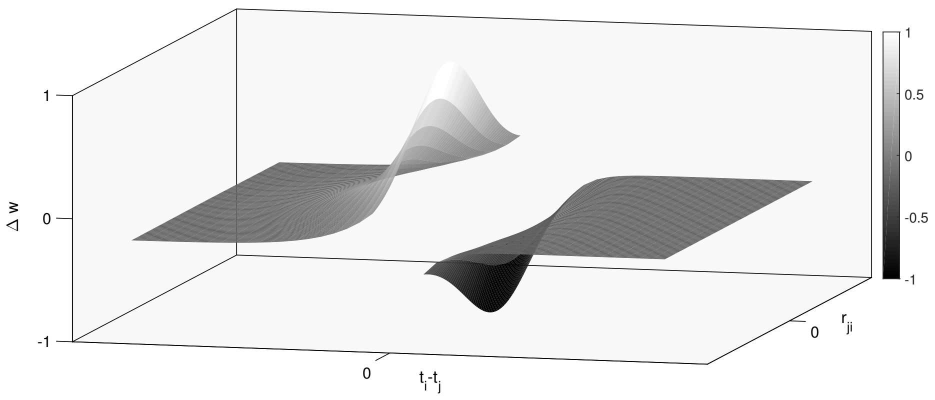
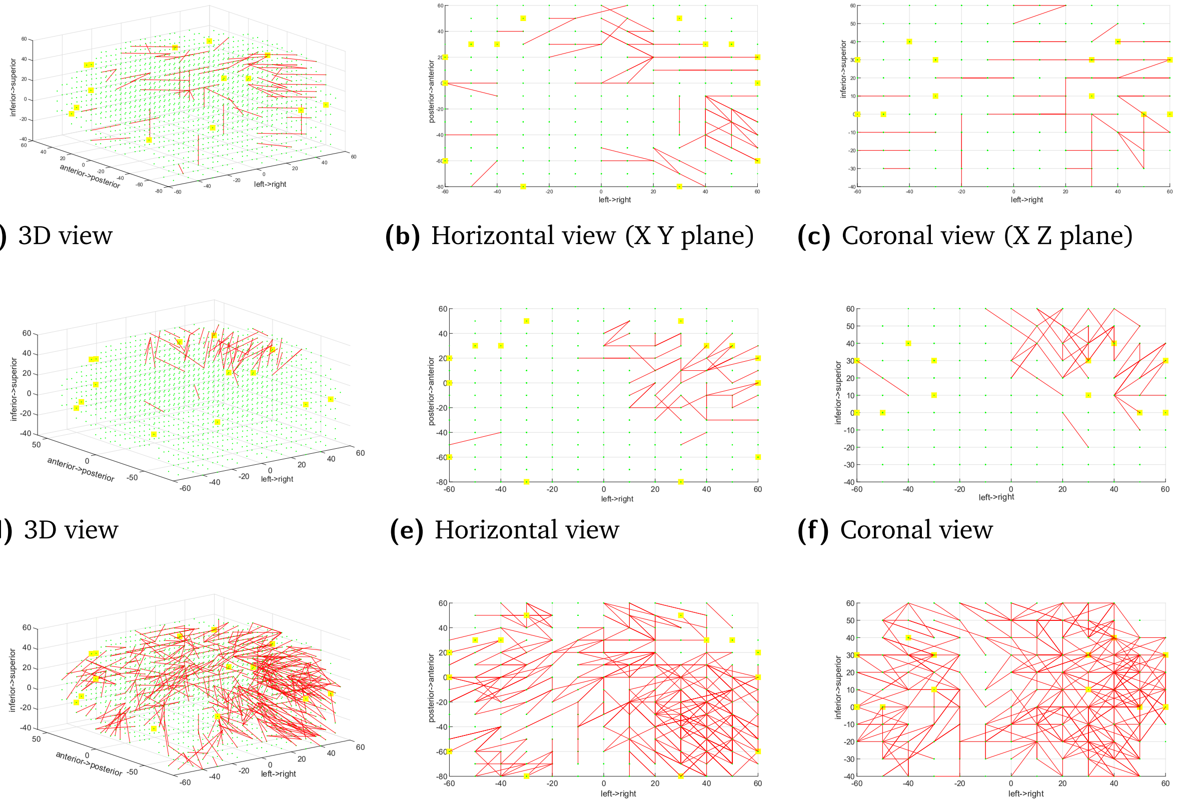


Figure 7.9.: Graph showing the relationship of oiSTDP weight update Δw with post and pre synaptic firing time difference $t_i - t_j$ and orientation distance r_{ji}. As the temporal difference between neuronal spikes decreases, the effect on weight updating increases, so that spikes timed closely together lead to greater increases in weight updating than spikes timed further apart. The order of spikes also affects weight updating. If neuron j fires before neuron i consistently, then the synaptic weight between them continues to increase; however, if the order switches, the weight is reduced.

behaviour shown in Figure 7.6. The top half of the Mexican hat relates to the positive LTP weight update, which peaks at the minimum angular distance and decays with increasing angular distance. The bottom inverse half, on the contrary, relates to negative LTD weight update which achieves a negative peak at the minimum angular distance. In this way, the angular proximity of the neurons plays a role in modulating the spike synchronicity driven dynamic influence.

7.5 Analysis of oiSTDP Algorithm Using Synthetic Data

To analyse the behaviours of the oiSTDP learning algorithm, synthetically generated dynamic spatio-temporal data D_{seq} and static orientation data D_{stat} have been used. The input spike data D_{seq} is of size 128×14, and was generated in a way that it mimics a random one second sample of a $128Hz$ 14 channel EEG device. The current experiment included $M = 1485$ neurons with sparse recurrent connections. The neurons in the SNNc are spatially distributed to resemble the



(a) 3D view **(b)** Horizontal view (X Y plane) **(c)** Coronal view (X Z plane)

(d) 3D view **(e)** Horizontal view **(f)** Coronal view

(g) 3D view **(h)** Horizontal view **(i)** Coronal view

Figure 7.10.: The columns show the 3D, horizontal and coronal view of the strongest connection weights in the 3D SNNc at the end of the unsupervised learning. Each row corresponds to a different input orientation data. Every neuron of the SNN in the first row were simulated with orientation data ($\alpha = 0°$, $\beta = 0°$). This resulted in the majority of the strong connections being oriented in the direction ($alpha = 0°$, $\beta = 0°$). This shows the systematic bias towards orientation information in absence of any dynamic bias. The second row shows similar systematic bias towards the input orientation ($\alpha = 45°$, $\beta = 45°$). The third row is the baseline with no orientation influence where no systematic orientation information bias can be observed.

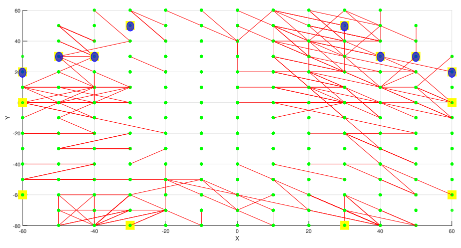
shape of the brain (N. K. Kasabov, 2014). The location of the input neurons in the SNNc are resolved as per the natural spatial ordering of EEG channels-AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4 and F8. For connection generation, $r_{swc} = 0.02$ (meaning connect neurons within 2% of the maximum distance), and $W_0 = 0.05$, have been used. The spiking neurons are simulated using the parameters ($\eta_{thr} = 4, \upsilon_{thr} = 0.1, \kappa_- = \kappa_+ = 0.01$).

7.5.1 Effect of the Orientation Information on SNNc

This segment demonstrates the systematic effect of orientation information in the SNNc map via the orientation influence ψ. The oiSTDP learning rule represents orientation information in conjunction with the spatio-temporal information in the connection strengths. The sample D_{seq} was taken from a Poissons' distribution to keep minimal spike synchronicity in the data. This is done to minimise the influence of synchronicity to the maximum. As per Algorithm 7.1, absence of synchronicity will mean that the spike data has minimal influence in the SNNc weight update. Figure 7.10 shows the systematic effect of different input orientation information on the final 3D SNNc map. In first of three experiments, all the SNNc neurons were fed with input orientation data ($\alpha = 0°$, $\beta = 0°$), *i.e.* parallel to the X axis and perpendicular to the Z axis. It is clearly visible from Figures 7.10a, 7.10b and 7.10c that the strongest connections in the SNNc are formed in the direction of the orientation information provided. The second and third experiment use ($\alpha = 45°$, $\beta = 45°$), and no orientation information respectively. It is evident from Figure 7.10 that in absence of the temporal information (synchronicity), the orientation information is reflected in the strongest connections of the SNNc, and, as such, in simple cases, they are visually discriminatory.

7.5.2 Effect of the spike synchronicity on SNNc

The aim of this experiment is to show the effect of spike synchronicity, *i.e.* the effect of STDP learning (Equation 5.7) on the SNNc map for different spatio-


(a) Synchronous input spike train at locations AF3, F7, F3, FC5, FC6, F4, F8 and AF4 (Horizontal view)

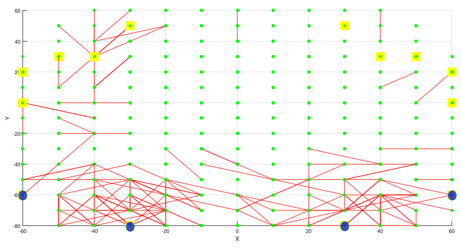

(b) Synchronous input spike train at locations P7, O1, O2 and P8 (Horizontal view)

Figure 7.11.: Comparison of the effect of synchronous input spikes on the SNNc map generated by Algorithm 7.1. The blue dots show the synchronous input channels.

temporal patterns. Per the STDP learning rule, greater synchronicity leads to stronger connections through LTP. To demonstrate the effect of the spatio-temporal synchronicity, two samples of the input spike data have been created. In the first sample, the spike sequences corresponding to the channels in the frontal lobe of the brain are kept the same (mimicking 100% synchronicity) and in the second sample, 100% spike synchronicity is kept at the occipital and parietal lobe. Figure 7.11 shows the comparison between the two SNNc maps created by the oiSTDP learning algorithm when fed with these two samples. The 'strongest connection' density is clearly more prominent in the frontal lobe in Figure 7.11a due to the greater input spike synchronicity in that region. Similar clusters (Figure 7.11b) at the parietal and occipital lobe can be seen with when the second sample is used. Through these analyses, it has been demonstrated how different temporal patterns and the spatial arrangement of such patterns can affect the visual map of SNNc through the oiSTDP learning.

7.6 Personalised Predictive Modelling of Treatment Outcome using NeuCube with oiSTDP Learning

This study was conducted as part of a large cross-sectional TRS study investigating Clozapine (CLZ) response in people with treatment resistant schizophrenia using EEG, MRI and genetic information. This study is a collaborative work with the school of pharmacy, University of Auckland. The University of Auckland conducted the participant recruitment, data collection and preprocessing. The current study's concentration and contribution lies in the data preprocessing, computational method development and data analysis. Strong efforts have been made to include all the relevant information in this study. However, for in-depth detail about the study, participants, data acquisitions and so on, the PhD thesis of McNabb (2017) is a highly recommended read. This study was approved by the Health and Disability Ethics Committee and received locality approval from Auckland and Counties Manukau, District Health Boards of New Zealand. The ethical approval was obtained by the University of Auckland and a copy of the approval is attached in Appendix E.

7.6.1 Schizophrenia and Antipsychotics

The Global Burden of Disease Study, in 2013, estimated that 23 million people, across the globe, were living with schizophrenia (Vos et al., 2015). Taking into account fewer than 0.01% of the total population (though it was previously reported by McGrath, Saha, Chant and Welham (2008) the prevalence rates between 0.3% and 0.7%), schizophrenia was found as a leading cause of years lived with disability and contributed to 3.7% of the global burden of disease (Vos et al., 2015). The annual national expenditure of schizophrenia, in a systematic review, was estimated to range between USD 92 billion and USD 102 billion, with indirect costs accounting for 50% to 85% of the overall cost (Chong et al., 2016).

Schizophrenia most commonly emerges during adolescence or early adulthood (Spitzer & Williams, 1980). This disease is a leading cause of disability in individuals aged between 15 and 44 years (Rössler, Salize, van Os & Riecher-Rössler, 2005). Schizophrenia affects its targets through hallucination, reduced emotional expression and relapsing episodes of delusions, along with persistent cognitive dysfunction and several other disruptive symptoms that hinder an individuals capacity to be an active and engaged member of society, thereby greatly diminishing their quality of life (Spitzer & Williams, 1980). Pharmacological treatment is one of the most successful tools for managing the symptoms affiliated with schizophrenia and has been shown to be effective in the majority of instances (Matheson, Shepherd & Carr, 2014). Nevertheless, despite best-practice, there are a small number of individuals who remain symptomatic. It has been estimated that as few as 5% to as much as 60% of the population inflicted with schizophrenia are resistant to first-line treatment (Elkis, 2007; Lehman et al., 2004; Essock et al., 1996; Juarez-Reyes et al., 1995). Contributing to some of the highest rates of hospitalisation and diminished functioning in mental health (Iasevoli et al., 2016; Lieberman & Murray, 2012), resistance to first-line antipsychotic intervention is estimated to cost USD 34 billion per annum in healthcare expenditure within the US alone (Kennedy, Altar, Taylor, Degtiar & Hornberger, 2014). Relieving patients of their most incapacitating symptoms, the atypical antipsychotic clozapine (CLZ) has been shown to be effective for treating between 30% and 70% of individuals who fail first-line treatment (Elkis, 2007; Essali, Al-Haj Haasan, Li & Rathbone, 2009; J. M. Kane & Correll, 2016). The severe and potentially life-threatening side-effects linked to CLZ, however, restrict its use in the clinic. There are no current means for identifying those who will respond to treatment with CLZ, or alternatively, for identifying who will fail to respond to first-line therapy.

7.6.2 Treatment Outcome in People with Schizophrenia

Since the advent of chlorpromazine during the middle of the twentieth century, pharmacological interventions have been recognised as an integral component of treatment of people with schizophrenia. Nevertheless, treatment using first-line antipsychotics has been shown to be effective in only 60% to 80% of individuals. Fewer options for treatment exist for those who do not respond to first-line therapy, thereby leaving many patients without an effective means for managing their symptoms. Research suggests that CLZ can successfully treat treatment-resistant schizophrenia (TRS) (Meltzer, 2010; J. Kane, Honigfeld, Singer & Meltzer, 1988), resulting in significant enhancements towards quality of life, as well as positive long-term functional outcomes in some individuals (Essali et al., 2009). However, CLZ also has the potential to cause serious, life-threatening side effects, such as agranulocytosis, myocarditis and cardiomyopathy, and has therefore, been restricted in its use, leading to delayed access for many individuals (Wheeler, 2008). Due to its noteworthy potential to cause improvement, psychiatrists are in need of reliable tools that can predict whether CLZ will be both an effective and a safe option for clients with schizophrenia.

Evidence suggests that schizophrenia is linked to disruptions to structural and functional connectivity (Yu et al., 2012; Fitzsimmons, Kubicki & Shenton, 2013). Work from previous research also suggests that functional connectivity may be different between individuals who respond to CLZ and those who are ultra-treatment-resistant (UTRS) (Creese, Burt & Snyder, 1976; Knott, Labelle, Jones & Mahoney, 2000). (Rodrguez, Andreé, Castejón, Garca et al., 1996) found that lower perfusion in the thalamus, left basal ganglia and right prefrontal regions are poor responders to CLZ prior to treatment, and that individuals who have high metabolic activity in the dorsolateral prefrontal cortex were more likely to experience improvements in negative symptoms after administrating CLZ. Another study found improvements in the Positive and Negative Syndrome Scale (PANSS) correlated with pretreatment inter- and intra-hemispheric spectral power asymmetry, which was measured using electroencephalography (EEG) (Knott et al., 2000).

Pattern recognition algorithms can provide a novel and practical means to understand the differences between patients and healthy controls and predict individual patients' responses to treatment. Within psychiatric research, in particular, machine learning has gained considerable momentum as a useful tool for developing predictive models of treatment response. Incorporation of multiple imaging modalities into these algorithms could provide increased reliability, especially in disorders where clinical diagnosis does not necessarily guide treatment. M. J. Patel et al. (2015) recently applied machine learning algorithms to predict treatment response in late-life depression using a combination of clinical and imaging data. Comparing several algorithms, they determined that alternating decision trees could most accurately predict treatment outcome in this cohort using a combination of structural and functional connectivity data (M. J. Patel et al., 2015). Khodayari-Rostamabad, Reilly, Hasey, de Bruin and MacCrimmon (2013) have used EEG data to predict the response to selective serotonin reuptake inhibitors in major depressive disorder and to CLZ in people with treatment-resistant schizophrenia (Khodayari-Rostamabad, Hasey, MacCrimmon, Reilly & de Bruin, 2010). C.-C. Lin et al. (2008) also employed machine learning algorithms to predict the response to CLZ, instead of using a combination of clinical and pharmacogenetic data as input. Doehrmann et al. (2013) employed machine learning techniques to predict treatment outcome in social anxiety disorder. Using task-based fMRI, they accounted for 40% of the variance in treatment response (Doehrmann et al., 2013). The challenge now is to create an algorithm that can incorporate brain data from different modalities.

7.6.3 Method

Participants

In accordance with the Diagnostic and Statistical Manual of Mental Disorders (DSM-IV), persons with a diagnosis of schizophrenia were recruited from mental health clinics within the Waitemata and Counties Manukau districts of Auckland, New Zealand. Participants were required to be: between the ages of 18 and 45; and clinically stable for at least six weeks prior to their inclusion in the study and also receiving atypical antipsychotics for the treatment of schizophrenia. Approximately twenty psychiatrically healthy control participants were also recruited as part of the research. The exclusion criteria involved current or previous diagnosis of any other axis I disorder, history of traumatic brain injury producing loss of consciousness of longer than three minutes, along with other significant physical disorders that were uncontrolled and may have affected brain structure, active substance dependence and contraindications for MRI. It is also noteworthy that participants were excluded from analysis if their functional image or supporting fieldmap image was overwhelmingly disrupted by motion.

As per the history of treatment and current antipsychotic regimen, participants were assigned into one of three studies: first-line responder (FLR); treatment resistant (TRS); and ultra-treatment resistant (UTRS). The FLR group involved those responding well to first-line atypical antipsychotic mono-therapy. The TRS group consisted of participants who failed at least two previous six-to-eight-week trials of atypical antipsychotics and were now receiving CLZ. Lastly, those who failed at least two previous six-to-eight-week trials of atypical antipsychotics and had also failed a trial of CLZ monotherapy were enrolled in the UTRS group. Participants in the UTRS group all received a mixture of two antipsychotic drugs; 79% received CLZ as part of an augmented treatment method.

The demographics of the participants were compared across cohorts using IBM SPSS Statistics Version 24. The Students t-test was used to analyse the variables that satisfied assumptions of homoscedasticity (Brown-Forsythe test for equality of variances) and normality (Shapiro-Wilk test for normality). Variables that violated assumptions of homoscedasticity and/or normality were analysed using the Mann-Whitney U tests or Z-score.

Data Acquisition

The Siemens Mangetom Skyra 3T scanner at the Centre for Advanced MRI (University of Auckland, New Zealand) was used to acquire the diffusion and resting-state (rs) functional magnetic resonance images. Magnetisation prepared 180-degrees radio-frequency pulses and rapid gradient-echo (MPRAGE) sequence were used to obtain Structural T1-weighted images. These were the acquisition parameters: Repetition time (TR) 1900 ms; echo time (TE) 2.39 ms; inversion time (TI) 900 ms; flip angle $9°$; repetition 1; acceleration (GRAPPA) factor of 2; field of view (FOV) 230 mm; matrix 256×256; voxel size $0.9 \times 0.9 \times 0.8$ mm.

Diffusion-weighted (DWI) images were obtained through an echo planar imaging (EPI) sequence with the following parameters: TR 8900 ms, TE 95 ms, FOV 240 mm, 122×122 matrix, 2.0 mm slice thickness, isotropic voxel size $2.0 \times 2.0 \times 2.0$ mm. An acceleration factor of 2 was used. 67 slices were acquired parallel to the anterior commissure-posterior commissure ($A >> P$) direction with diffusion-weighting factor $b = 1000$ s/mm2 in 64 gradient directions. There were a further 8 scans without diffusion weighting ($b = 0$ s/mm2) which were also acquired.

Gradient distortion (fieldmap) images for diffusion data were acquired using a gradient echo pulse sequence with the following parameters: TR 655 ms; TE1 4.92 ms; TE2 7.38 ms; voxel size $3.8x3.8x2.0$ mm; phase encode direction $R >> L$; FOV 240 mm.

Rs functional images were acquired using EPI with the following parameters: TR 3000 ms, TE 30 ms; echo spacing 0.65 ms (0.62 ms for last 4 participants, following software upgrade); phase-encode direction $A >> P$; slices 54; volumes 160; FOV 192 mm; acceleration factor of 2; matrix 64×64; voxel size $3.0 \times 3.0 \times 3.0$ mm. Participants were requested to lie immobile with eyes open and to focus their attention on a fixation cross that was presented on a screen in front of the scanner. Participants were given the instruction to keep a blank state of mind by not thinking of anything in particular.

Gradient distortion images for functional data were acquired using a gradient echo pulse sequence with the following parameters: TR 655 ms; TE1 4.92 ms; TE2 7.38 ms; voxel size $3.4 \times 3.4 \times 2.4$ mm; phase-encode direction $A >> P$; FOV 220 mm.

fMRI Image Preprocessing

fMRI image preprocessing was performed using FSL version $5.0.7$ (S. M. Smith et al., 2004). Brain tissue was extracted from raw magnitude files using FSLs BET (S. M. Smith, 2002), after which brain-extracted images were eroded to ensure that no voxels containing non-brain tissue remained. Fieldmaps were then created using the fsl_prepare_fieldmap function. Functional image registration to high resolution structural and MNI152 standard space was performed using FMRIB's Expert Analysis Tool (FEAT). Preprocessing parameters in FEAT were as follows: motion correction = MCFLIRT; b0 unwarping = on; echo spacing = 0.325 (0.31 for last 5 participants); TE = 30; spatial smoothing = 5mm; global intensity normalisation = on; temporal filtering = off; MELODIC = off; registration to structural image = boundary-based registration; registration to MNI152_2mm = non-linear; warp resolution = 10 mm.

ICA-based Automatic Removal Of Motion Artefacts (ICA-AROMA) (Pruim et al., 2015) was used to remove motion artefacts from the fMRI data utilising FSLs FEAT output as input. White matter and cerebro-spinal fluid (CSF) maps were segmented from high resolution structural images using FSLs FAST (Yongyue Zhang, Brady & Smith, 2001) and warped to functional space using linear registration to FEAT output (FSLs FLIRT (Jenkinson & Smith, 2001)). Nuisance time-series were generated from ICA-AROMA output (denoised functional data) using CSF and white matter maps as input. A general linear model (GLM) of residual activity was then generated from the denoised functional data and nuisance time-series using FSLs GLM. A temporal mean file of denoised functional data was created, to which high-pass temporal filtering (sigma=16.7) was applied in addition to removal of residual activity attributed to CSF and white matter. Filtered, de-noised functional data was then warped to MNI standard space for use in further analysis.

DTI Image Preprocessing

DTI image preprocessing was performed using FSL version $5.0.7$ (S. M. Smith et al., 2004). Structural images were reoriented to a standard template and brain tissue was extracted from raw image files using FSLs brain extraction tool (BET) (S. M. Smith, 2002). If automatic brain extraction failed to eliminate all non-brain tissue, the excess was removed manually. Magnitude images were subjected to the same process, after which brain-extracted images were eroded to ensure that no voxels containing non-brain tissue remained. Fieldmaps were then created using the fsl_prepare_fieldmap function. The gradient-free image was used to create a binary mask with BET. Gradient distortions were corrected using FSLs fugue function and output registered to gradient-free images using the linear registration function (FLIRT) (Jenkinson & Smith, 2001). Data were then corrected for head movement and eddy current distortions using FSLs eddy tool (Andersson & Sotiropoulos, 2016). Slices with average intensity at least four standard deviations lower than the expected intensity were interpolated with predictions made by the Gaussian Process (Andersson, Graham, Zsoldos & Sotiropoulos, 2016). DTIfit was used to independently fit diffusion tensors to each voxel, limited to brain space using the binary brain mask. Crossing fibres were modelled using Bayesian Estimation of Diffusion Parameters Obtained using Sampling Techniques (BEDPOSTX) (Behrens et al., 2003). BEDPOSTX estimates of primary fibre orientations (dyads1) were then warped to a standard MNI template for use in the initial NeuCube construction.

The second stage of data processing focused on selecting a set of voxels from the fMRI and DTI to be used to build the multi-modal NeuCube model. As discussed in Section 7.4.3, since a major component of this model captures temporal variations in data and the noise reduction capabilities of SNN architectures through encoding (N. Kasabov, Scott et al., 2016), it was hypothesised that the discriminatory information is hidden in the voxels with significant variability in the activity over time. A set of voxels with an absolute mean standard deviation of greater than 105 were selected for the experiments. Figure 7.12 shows the 3D atlas locations of the selected voxels in the MNI coordinate system. The selected voxels are found to be predominantly ($> 67\%$) located in the cerebellum area of the brain. The second and third column of the ROI frequency table (see Table 7.2) also lists to the number and the percentage of voxels belonging to the different ROIs.

In this study, the aim was to build a model for discriminating CLZ mono-therapy respondent and non-respondent individuals from multi-modal fMRI and DTI brain data. For this investigation, a subset of data was collected from the TRS study with the intention of classifying subjects into groups with either TRS or UTRS using resting-state fMRI and DTI data. A total of 30 subjects were chosen for the study. Sixteen subjects belonged to the TRS group and fourteen to the UTRS group.

Experimental Results

The final preprocessed dataset consists of dynamic fMRI trials $\mathbf{D_{seq}} \in \mathbb{R}^{30\times 2318\times 80}$ and static DTI orientation vector data $\mathbf{D_{stat}} \in \mathbb{R}^{30\times 2318\times 3}$ of 30 subjects and 2318 voxels. Each fMRI voxel is sampled over 80 time points within a trial. The voxels of the DTI orientation data are represented by 3D vectors signifying the primary orientation of the fibre tract at the voxel location.

Table 7.2.: Frequency table of ROI's of the selected voxels.

ROI	# voxel	%
Frontal Lobe	177	7.64
Insula	16	0.69
Temporal Lobe	138	5.95
Cerebellum	1557	67.17
Occipital Lobe	25	1.08
Parietal Lobe	134	5.78
Thalamus	187	8.07
Caudate	84	3.62

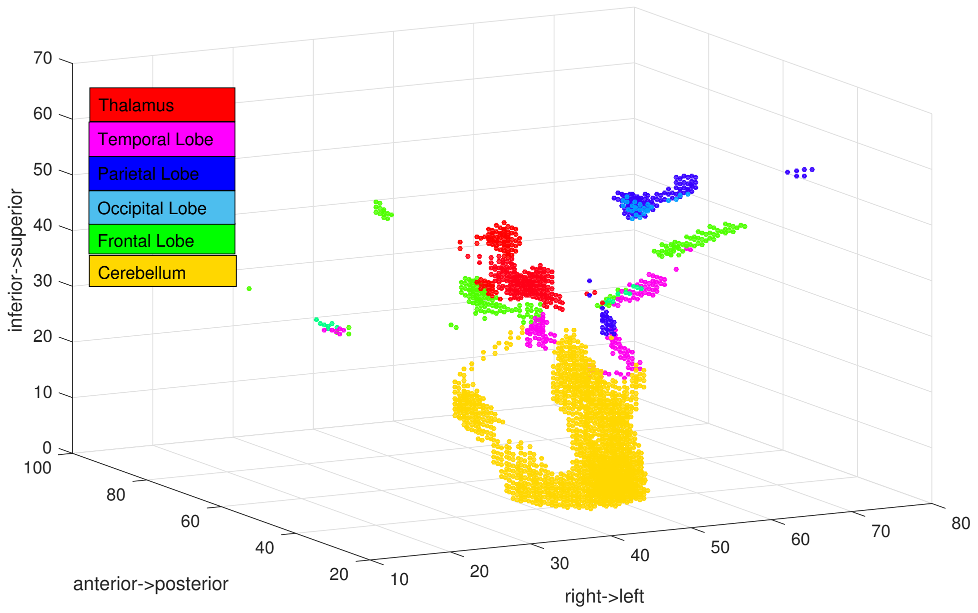


Figure 7.12.: Voxel selection using absolute mean standard deviation.

The NeuCube personalised architecture consists of three modules as described in Section 7.4.3. The first step of the process was to compress or encode the dynamic fMRI data from continuous real space to binary spike space. For the encoding step, the BSA (Schrauwen & Van Campenhout, 2003) algorithm was chosen due to its ability to represent brain data as important spike event and has shown promising results in (Sengupta & Kasabov, 2017; Nuntalid et al., 2011). For the second step, subject-specific SNNc's were initialised with $Q = 1000$ spiking neurons, $N = 2318$ input neurons and synapses were initialised using the SWC algorithm within the radial neighbourhood of $r_{swc} = 0.05$ and $w_{ij} = 0.05$. The spiking neurons were set with the hyperparameters $\{v_{rest} = 0, v_{thr} = 0.1, \eta_{thr} = 8\}$. These along with other hyperparameters were chosen by a grid based hyperparameter search strategy. The encoded fMRI data and the DTI orientation vector data for each subject were then passed through the initialised SNNc, generating O_{seq} for each subject using Algorithm 7.1. In the final step, a lazy K-NN binary classification model using 50% of the randomly chosen subjects was learned. It is to be noted that a custom distance function as part of the K-NN algorithm for learning binary spike data was used. A custom spike asynchronicity-based distance function as described in Section 6.6.3 and in (Sengupta & Kasabov, 2017) has been utilised. The K-NN learning algorithm can of course be replaced by any supervised learning module that can learn from discrete or binary input data.

Due to the multi-modular and rather flexible nature of the NeuCube architecture, selecting baselines for comparison is a challenging task. In this work, the NeuCube architecture was used as a combination of temporal feature compressor, spatial expander and classifier. The compressor and the SNNc module together are used as a spatio-temporal feature extraction module. The supervised read-out layer then learns a model from the transformed feature transformed data. Hence it is appropriate to compare the BSA+oiSTDP feature extraction module against other feature extraction methods in continuous data domain. The following feature extraction algorithms have been compared:

1. Sparse autoencoder (Ng, 2011): Autoencoders are shallow single hidden layer neural networks that can perform identity mapping of the input. The hidden layer of the autoencoder learns non linear lower dimensional data representations. The sparse autoencoders are an extension of regular autoencoders that impose sparsity regularisation constraints on the loss function. In these present experiments, the fMRI data was used to learn a sparse autoencoder (with 1000 relu units in the hidden layer and L1 regularisation constraint of 10^{-5}) that encodes the data into 1000 dimensional feature space using the python keras API (Chollet et al., 2015).

2. Principal component analysis (PCA): PCA is a standard orthogonal linear feature transformation technique that transforms features into principal components. Using the scikit-learn API, 1000 principal components were fitted and transformed on the fMRI data.

3. Independent component analysis (ICA): ICA is another statistical feature transformation technique used to decompose feature space to statistically independent component space by maximising statistical independence of the estimated components. Once again, scikit-learn (Pedregosa et al., 2011) API's FastICA algorithm was used to fit and transform the fMRI data to 1000 independent components.

4. Restricted Boltzman's machine (RBM) (Hinton & Salakhutdinov, 2006): RBM is an unsupervised nonlinear feature learner based on a probabilistic model that has gained much popularity in the deep neural network domain. The scikit-learn API (Pedregosa et al., 2011) was used to learn a Bernoulli RBM network, with 1000 components using stochastic Maximum likelihood (Tieleman, 2008) learning.

Table 7.3 presents the experimental results. The rows of the table compares the methods for the classification task. (C) and (E) in the method names cor-

[1]TFC=Temporal feature compressor, NTFC=No-temporal feature compressor, SE=Spatial expander, C=classifier

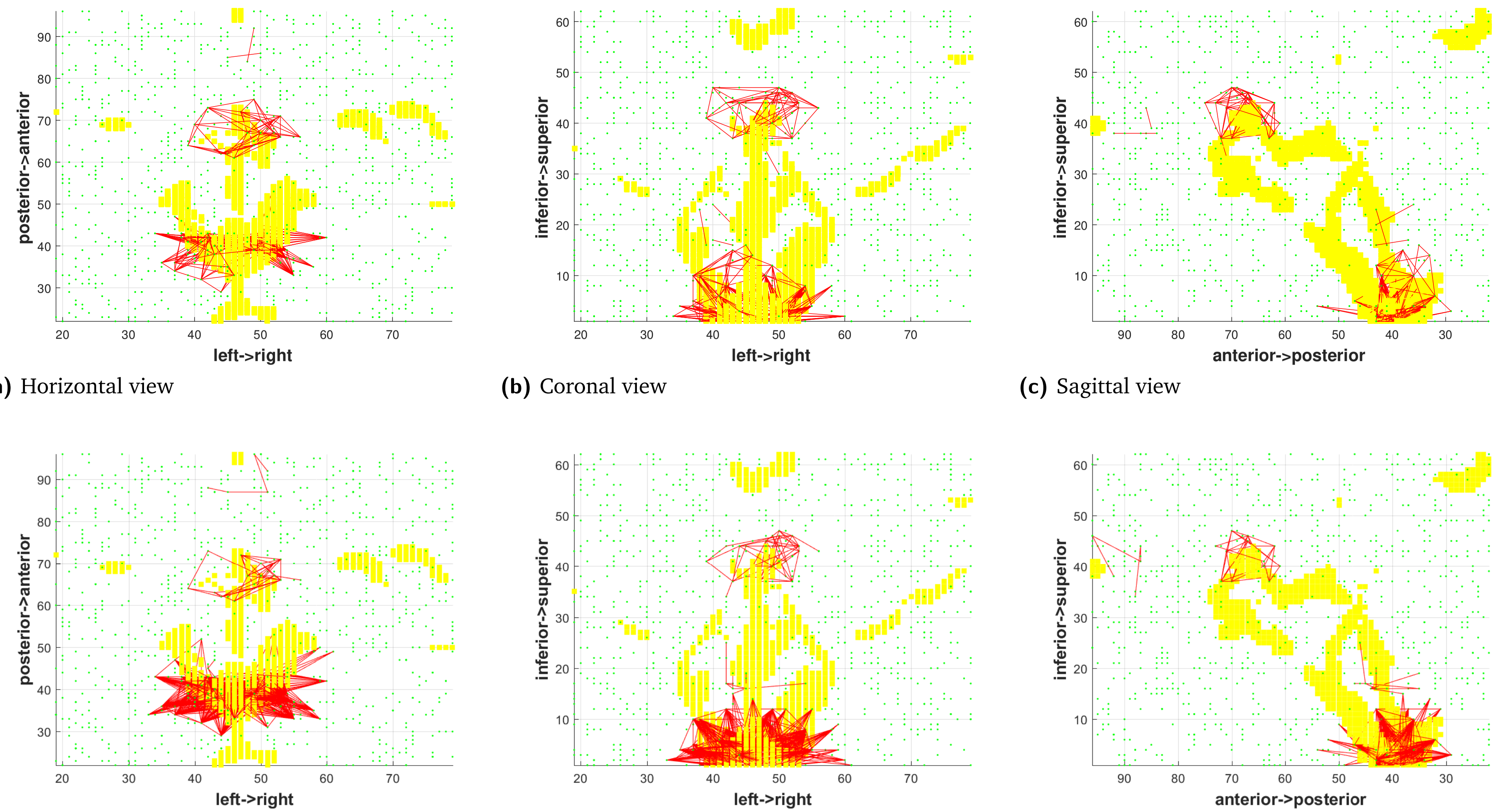
[2]The performance metrics are computed as $mean \pm standard\ deviation$ of 10 independent train/test runs of the classification module

Table 7.3.: Comparison of the pattern recognition methods on the binary classification task.

Method	Framework[1]	Data	Relation learning capabilities		Performance[2]	
			Temporal	Multi-modal	Accuracy (%)	Cohen's κ
BSA+oiSTDP+K-NN(C)	TFC+SE+C	fMRI+DTI	✓	✓	$\mathbf{72.3 \pm 12.3}$	$\mathbf{0.44 \pm 0.25}$
BSA+STDP+K-NN(C)	TFC+SE+C	fMRI	✓	✗	69.4 ± 13.9	0.38 ± 0.28
BSA+K-NN(C)	TFC+C	fMRI	✗	✗	64.2 ± 12.4	0.22 ± 0.26
Sparse Autoencoder+K-NN(E)	NTFC+C	fMRI	✗	✗	56.1 ± 7.2	0.01 ± 0.11
PCA+K-NN(E)	NTFC+C	fMRI	✗	✗	56.1 ± 11.3	0.13 ± 0.18
ICA+K-NN(E)	NTFC+C	fMRI	✗	✗	62.8 ± 12.3	0.26 ± 0.23
RBM+K-NN(E)	NTFC+C	fMRI	✗	✗	36.2 ± 4.9	-0.23 ± 0.11
LSTM	C	fMRI	✓	✗	45.7 ± 9.6	-0.15 ± 0.14
GRU	C	fMRI	✓	✗	45.2 ± 7.5	-0.018 ± 0.13

respond to the custom and euclidean distance function used as part of K-NN respectively. The framework column specifies the role of each component in the method names. For example, the proposed BSA+oiSTDP+K-NN is a combination of temporal feature compressor (TFC), spatial expander (SE) and classifier (C). The Performance of the binary classification task is measured by overall accuracy and Cohen's κ statistic. The first 3 rows of the table compare the different NeuCube architectures. The BSA+oiSTDP+K-NN is the proposed architecture for fMRI and DTI integrated learning. The next two methods systematically removes (1) orientation influence from SNNc learning (STDP) and (2) the SNNc module to show the effect of inclusion of these artefacts on the performance. The best performance across the different methods is achieved by the proposed BSA+oiSTDP+K-NN architecture with overall accuracy of $72.4 \pm 12.3\%$ and Cohen's kappa of 0.44 ± 0.25. The classification accuracy increases by $\approx 8\%$ and doubles the mean Cohen's κ statistic when oiSTDP-based SNNc learning is performed in the middle using fMRI and DTI data. Rows 4 to 7 are the non temporal feature extraction baselines described earlier. Due to the non temporal nature of the baseline feature compressors, the fMRI data for each subject is input to these feature extractors as a single vector (created by concatenating the temporal dimension) leading to a massive feature vector space. K-NN ($K = 1$) classifier was used for the classification task to keep the comparisons as fair as possible. The disadvantage of the large feature space is quite imperative as it leads to potential over fitting of the data. The DTI data was not added to the already large feature space to avoid further over fitting. As the SNNc of NeuCube is a spiking recurrent neural network framework with temporal or sequential learning capabilities, the binary classification task with other single hidden layer recurrent neural network framework, such as long short term memory (LSTM) (Hochreiter & Schmidhuber, 1997b) and gated recurrent units (GRU) (K. Cho et al., 2014) was also brought to light. Both LSTM and GRU networks were designed as shallow single hidden layered neural networks having 50 LSTM and GRU units. These networks were implemented in keras API (Chollet et al., 2015) and learned by optimising the binary crossentropy loss function using the adaptive momentum optimiser. The results for the baselines show that K-NN performs best on ICA-based feature representation. PCA and sparse autoencoders are similar in accuracy, but PCA achieves a stronger κ statistic. On the other hand, the baseline recurrent neural networks fail to learn the task, leading to poor performance statistics.

Furthermore, connection weights have been individually scrutinised for the TRS and the UTRS group, generated by the oiSTDP learning algorithm. Figure 7.13 shows a comparison of the strongest mean connection weights of the TRS and the UTRS groups. Most of the strong connections are shown to be created in the lower cerebellum and thalamus across UTRS and TRS group. It has been shown that by connections via the thalamus, the cerebellum innervates with motor cortical, pre-frontal and parietal lobes (W. Ou, Cameron & Thomas, 1992). Following cerebellar damage, neuro-cognitive symptoms and a cognitive affective syndrome including blunted affect and inappropriate behaviour have been shown (Baillieux, Verslegers, Paquier, De Deyn & Mariën, 2008). Recent fMRI and PET studies have also demonstrated the involvement of cerebellum and thalamus in sensory discrimination (Gao, Parsons, Bower, Xiong et al., 1996), attention (Courchesne, Akshoomoff, Townsend & Saitoh, 1994), and complex problem solving (S. Kim, Ugurbil, Strick et al., 1994). All of which may be impaired in people with schizophrenia (Yeganeh-Doost, Gruber, Falkai & Schmitt, 2011). A recent study also has shown that the administration of CLZ in people with schizophrenia can restore cerebellar functions by altering the glutamatergic system and neuro-plasticity (Yeganeh-Doost et al., 2011). The present study has additionally shown (Figure 7.13) the presence of a large density of strong connections in the cerebellum region of the model in the UTRS group compared to the TRS group. Similarly, a large number of strong connections are present in the thalamic region of the TRS as opposed to UTRS. As the oiSTDP learning reg-



(a) Horizontal view **(b)** Coronal view **(c)** Sagittal view

(d) Horizontal view **(e)** Coronal view **(f)** Sagittal view

Figure 7.13.: A visual comparison of the strongest connections (mean weight across subjects within a group) formed in the SNNc of the TRS and the UTRS group. The top row shows the connections in the TRS group and the bottom row corresponds to the UTRS group. The yellow coloured cluster represents the input neurons and the green neurons are the computational spiking neurons.

ulates the connection strength based on the joint information extracted from the spike activity and angular information, it can be stated that the distinctive connection density in the cerebellum and thalamus regions of the two groups in the Figure 7.13 is due to distinctive fMRI activity and DTI orientation information in the input data.

7.7 Summary and Conclusion

This is the first attempt to integrate multiple modalities of information in a spiking neural network architecture. The novelty of this approach lies in the proposed personalised SNNc-based architecture of NeuCube, and most importantly the proposed oiSTDP learning algorithm, which can integrate multiple modalities of information including time, space and orientation from data. Despite some assumptions being made on multi-modal brain data, the proposed algorithm is not limited to brain data and can be extended to data having spatial, temporal and orientation information. Examples of such data are weather (change in temperature, wind movement, and cloud movement) and traffic data.

The experiments shown here were conducted to demonstrate the ability of the algorithm to capture discriminative joint information present in the data and represent this information within its connection strengths. The current design has incorporated DTI and fMRI from individuals initiating antipsychotic therapy to create a personalised classifier of treatment response in people with schizophrenia. Interrogation of the SNNc network revealed increased network connectivity in the cerebellar region of the model, potentially implicating activity in this area of the brain as a bio-marker of treatment response in schizophrenia. Inclusion of more participants and studies using specific task-based designs may expose other markers not currently identified in the literature and provide novel hypotheses regarding why some individuals respond to CLZ mono-therapy while others do not. Additional applications of the algorithm may include other disorders where treatment or clinical outcome is poorly understood.

The ability to incorporate data from multiple imaging modalities simultaneously could increase the reliability of the model to predict treatment outcomes in future. To date, studies have achieved high rates of accuracy in patient samples combining single imaging techniques alongside clinical and pharmaco-genetic data (M. J. Patel et al., 2015; Khodayari-Rostamabad et al., 2013), though none have led to changes in clinical practice. This could potentially be a result of over fitting during training, which the algorithm proposed would be less likely to do. The learning algorithm is formulated in a way that it favours joint information over information that is contradictory. In this way the predictive outcomes are robust and trustworthy.

In the future, apart from algorithmic refinement to further improve the performance, the aim is also to include EEG data as part of the integrated brain data modelling using the NeuCube personalised SNNc. Further improvement of the classification performance through the addition of EEG data to the model. This strand of work will lead to new methods for integration of multi-modal data with heterogeneous spatial and temporal resolutions.

7.8 Contributions and Publications

Contributions

1. *This Chapter has put forth a proposal for an unsupervised SNN learning algorithm for learning and recognising patterns in the form of spatial, temporal and orientation information from data.*

2. *A novel personalised SNNc based architecture of NeuCube has also been proposed for achieving sub-criticality in the SNNc network.*

3. *The proposed algorithm has been applied to a case study of predicting treatment response in people with schizophrenia. The results have shown the superior ability of the proposed SNNc architecture to incorporate multiple dimensions of information and generate a better performing model when compared to the other state-of-the-art technique.*

Publications

1. **Sengupta, N.**, McNabb, C. B., Kasabov, N. & Russell, B. (2018), integrating space, time and orientation in spiking Neural Networks: A case study on multi-modal brain data modelling, IEEE Transactions on Neural Networks and Learning System. DOI: https://doi.org/10.1109/TNNLS.2018.2796023.

8 Conclusion

This thesis has delved into numerous areas relating to both the theory and practice in neuromorphic pattern recognition systems and advanced the current status quo in various ways. This final Chapter discusses the major contributions this present thesis has made towards the literature and further articulates the ways in which the research questions that were posed at the beginning in Section 1.2 have been answered. The key caveats and limitations of this work are then examined along with the review of future research and open questions identified.

8.1 Research Questions and Contributions

From Chapter 3 onwards, at the end end of each chapter, a list of contributions has been enumerated. Additionally, a list of peer-reviewed publications that has resulted from that Chapter has also been mentioned. Here, I summarise the key-contributions towards answering the research questions. The novel contributions on the basis of the research questions are as follows:

8.1.1 How to Design Architectures of SNN that are Capable of Digesting and Processing Large Volumes and Variety of Spatio-temporal Data?

It is no doubt that in the era of big data, a multitude of research is ongoing which delves into the subject of handling large volumes of data. Many of the research in this area concentrate on improving the processing capacities of the infrastructures, such as using GPUs or building elastic cloud infrastructures. This research, however, investigates the large volume data processing problem in the light of efficient data representation in the form of binary spikes, and therefore, aims to build SNN pattern recognition systems that can recognise patterns from such binary spikes. Through the research performed in this thesis, I have delved into various aspects of the ways in which such scalable SNN architectures can be built.

Neuromorphic Computing Beyond von Neumann Architecture

Chapter, 3 has followed the development of brain-inspired spiking neural networks and presented the basis of the present research as being paradigm-shifting in regards to computation architecture. This Chapter presented the neuromorphic thinking and design that has the potential of creating a novel, efficient and low power echo-system between neuromorphic hardware (such as SpiNNaker) and neuromorphic pattern recognition systems (such as, the NeuCube) architecture.

Software Design Principles of NeuCube SNN architecture

Through the work in Chapter 4 and 5, an in-depth overview of NeuCube, an SNN architecture was presented at first. The focus was especially on the NeuCube generic prototyping and testing tool as a case of implementation framework. Recognising the NeuCube SNNc layer as the scalability bottleneck of the system, the focus was centered around the scalability of the SNNc layer. It has been elaborated through experimentation and analysis that the adjacency forward-backward list serves as the most optimal data structure for representing the SNNc graph in relation to optimal storage ($S(3 \times C)$ complexity), and execution time ($O(1)$) of the SNNc unsupervised learning algorithm. Through simulation in a commodity hardware, it was demonstrated that using the adjacency forward-backward list data structure, it is possible to run neuromorphic SNNc with neurons in the order of 10^6 and connections in the order of 10^{10}.

Beyond volume, the human brain is extremely good at processing a variety of data, which relates to the modularity (functional specialisation of the parts of the brain) and heterogeneity (variety in the components such as learning mechanisms, synapses, neuron dynamics and so on) inside the brain. Incorporating modularity and heterogeneity, thus, is an important aspect of any SNN system. The design principles with which one can achieve such modular and heterogeneous architectures was demonstrated in these Chapters.

8.1.2 How to Perform Neural Encoding on Real-world Data to Represent Information as Timings of Spikes?

At the start of the research, incorporation of neural, and especially the temporal encoding, was envisioned to be a critical piece of the puzzle towards solving neuromorphic pattern recognition in large volume data. Chapter 6 was dedicated towards exploring this idea.

Neural Encoding from the Perspective of Data Compression and Information Theory

This research began with the very notion that, in terms of pattern recognition, one can in fact, only learn from data when there exists redundancy. In many data analysis tasks, the data is preprocessed or re-coded in a way that could be seen as a form of data compression. If such a preprocessing does not destroy the patterns of interest, it results in comparative performance of the learning algorithms. The motivation of the temporal encoding, thus, was to reduce large volumes of data into a compressed state with minimal loss and the maximal presence of discriminable information. Then, a qualitative comparison of temporal and rate encoding schemes was made, as per Shannon and Kolmogorov's information theory principles. It was found the main interest, the temporal encoding scheme, to be adherent to Kolmogorov's algorithmic information theory.

Framework for *a priori* Knowledge Driven Optimisation Based Temporal Encoding

The *a priori* knowledge driven encoding framework was proposed on the premise that (1) a universal data encoder does not exist; and (2) *a priori*-knowledge of a data source can be injected into a prediction system that can predict the data generation process. Temporal data encoding was formalised as a generalised constrained optimisation problem (Equation 6.6), where *a priori*-knowledge of data generation process is injected into the problem formulation.

GAGamma Encoding Algorithm and Case Study on Benchmark Data

To illustrate a concrete example, an encoding algorithm based on the *a priori*-knowledge driven optimisation framework, namely GAGamma, was proposed, and applied as part of a pattern recognition framework (encoding and classification) on the benchmark StarPlus fMRI dataset. A comparison of the proposed GAGamma algorithm against the state-of-the-art temporal encoding algorithms such as BSA and TC not only demonstrated its superior data compression quality in regards to decoding error and bit compression ratio, but also achieved superior classification performance (Table 6.1). Additionally, it was observed that on the benchmark data, applying temporal encoding operation compressed the data dramatically, between 6 to 25 times compared to the raw data, still keeping the classification performance high and thus could capture the discriminatory information well.

8.1.3 How to Integrate Spatial, Temporal and Orientation Information Present in Multi-modal Brain Data using SNN Architecture?

This research question relates to Chapter 7. This research question was envisaged as a direction towards the data fusion approach for pattern recognition in multi-source multi-modal data. In order to keep the research focused and constrained, this research question was directed towards specific use-case, in this case, recognising patterns in multi-modal brain data. The rationale behind fusing multiple modality of brain data revolved around not only a hypothesised improved performance, but superior reliability of the model as well.

Personalised SNNc Architecture of NeuCube

From a methodological point-of-view, the current research stayed within the NeuCube framework and proposed modification of the NeuCube architecture for dealing with large volumes of multi-modal data. Sub-criticality and saturation behaviours in the NeuCube SNNc unsupervised learning algorithm were analysed, and discussed how such criticality can be minimised by using the personalised SNNc architecture depicted in Figure 7.5.

Orientation Influence Driven Spike-time Dependent Plasticity (oiSTDP) Learning for NeuCube SNNc

A novel online unsupervised learning algorithm for the SNNc layer of NeuCube was proposed, namely oiSTDP learning algorithm (see Algorithm 7.1), that can jointly fuse and learn from the spatial, temporal and orientation information from multi-source brain data.

Case Study on Predicting Treatment Outcomes of Clozapine on People with Schizophrenia

A case study was performed in collaboration with the University of Auckland on predicting treatment outcomes of clozapine in schizophrenia patients. The results presented in Table 7.3 summarise the comparative performances and capabilities of the proposed method against numerous other state-of-the-art methods, including deep learning algorithms. The proposed method of BSA+oiSTDP+KNN has shown best performance in regards to accuracy and Cohen's kappa statistic for the classification task. Further, interrogation of the SNNc network revealed increased network connectivity in the cerebellar region of the model, potentially implicating activity in this area of the brain as a bio-marker of treatment response in schizophrenia.

8.2 Limitations of the Thesis

Generally, the limitations of the individual pieces of work in this thesis have been discussed in context in all of the Chapters. Here, thus, only the overall limitations of this work will be discussed.

The studies performed as part of this thesis are proof-of-concept, rather than comprehensive studies. These works provide certain empirical support towards the systems introduced. It was never the researcher's intention to perform large-scale comprehensive experiments, instead, the intention was to provide the systems and methodologies to support the empirical studies, which are in turn, handled in other literature.

8.3 Future Direction and Closing remarks

Concluding a research is almost always the most difficult part because a significant piece of research typically asks more than what it answers. During the process of conducting this research as well, the same was found to be true. It is rather an impossible task to exhaustively list the open-ended questions and future directions. Throughout the Chapters of this thesis, possible future directions of the research have been discussed. Therefore, here, without elaborating too much, some potentially interesting research directions this thesis could open up in the future are mentioned:

- **Towards more efficient representation of real world data through data encoding.** The efficiency (both time and power consumption) with which the brain can recognise patterns is second to none. The current state-of-the-art in pattern recognition and artificial intelligence is significantly lagging in this domain. The inherent sequential processing architecture of the von Neumann computer architectures is a significant bottleneck towards achieving efficiency. Through the present work, there has potentially been a paradigm shift in research towards more efficient and accurate pattern recognition algorithms made through highly compressed representation of data using data encoding. This, in conjunction with developments in neuromorphic hardware systems over the next decade, can push AI to be more neuromorphic.

- **How to fuse multi-modal data with heterogeneous spatio-temporal resolution?** A significant future direction towards pattern recognition by fusing multi-modal brain data would be to focus on methods for fusing data with heterogeneous spatio-temporal resolutions such as fMRI and EEG.

- **Towards brain-like multi-modular heterogeneous architecture for pattern recognition?** This direction in research relates to the domain of neural networks. Through the work in Chapter 5 possibilities and opportunities (from a software design perspective) have been discussed to create heterogeneous and modular neural network consisting of spatially distributed components of different types that mimics the human brain. However, the open question that remains is the type of scenario where such architecture could be used, and then, the ways in which it could be built in a way that is useful to the field of AI.

Bibliography

Abbott, A., Sengupta, N. & Kasabov, N. (2016). Which method to use for optimal structure and function representation of large spiking neural networks: A case study on the neucube architecture. In *2016 international joint conference on neural networks (ijcnn)* (pp. 1367–1372). IEEE.

Abel, S. & Fompeyrin, J. (n.d.). Neuromorphic devices & systems.

Adrian, E. D. (1926). The impulses produced by sensory nerve endings. *The Journal of physiology*, *61*(1), 49–72.

Afterhyperpolarization. (2007). https://en.wikipedia.org/wiki/Afterhyperpolarization. Accessed: 2018-02-25.

Alexander, A. L. (2010). Deterministic white matter tractography. *Diffusion MRI: Theory, methods, and applications*, 383–395.

Andersson, J. L., Graham, M. S., Zsoldos, E. & Sotiropoulos, S. N. (2016). Incorporating outlier detection and replacement into a non-parametric framework for movement and distortion correction of diffusion mr images. *NeuroImage*, *141*, 556–572.

Andersson, J. L. & Sotiropoulos, S. N. (2016). An integrated approach to correction for off-resonance effects and subject movement in diffusion mr imaging. *Neuroimage*, *125*, 1063–1078.

Arya, A. S., Vadlamani, R., Valadi, T., Sengupta, N. & Kasabov, N. (2016). Cyber fraud detection using evolving spiking neural network. In *2016 11th international conference on industrial and information systems (iciis)* (pp. 263–268). doi:10.1109/ICIINFS.2016.8262948

Ashby, F. G. (2011). *Statistical analysis of fmri data*. MIT press.

Aur, D. & Jog, M. S. (2010). *Neuroelectrodynamics: Understanding the brain language*. IOS Press.

Babu, B. & Jehan, M. (2003). Differential evolution for multi-objective optimization. In *Evolutionary computation, 2003. cec'03. the 2003 congress on* (Vol. 4, pp. 2696–2703). IEEE.

Baillieux, H., Verslegers, W., Paquier, P., De Deyn, P. P. & Mariën, P. (2008). Cerebellar cognitive affective syndrome associated with topiramate. *Clinical neurology and neurosurgery*, *110*(5), 496–499.

Ballings, M., Van den Poel, D., Hespeels, N. & Gryp, R. (2015). Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications*, *42*(20), 7046–7056.

Baum, E. B. (1988). On the capabilities of multilayer perceptrons. *Journal of complexity*, *4*(3), 193–215.

Beckmann, C. F., Jenkinson, M. & Smith, S. M. (2003). General multilevel linear modeling for group analysis in fmri. *Neuroimage*, *20*(2), 1052–1063.

Behrens, T. E., Berg, H. J., Jbabdi, S., Rushworth, M. F. & Woolrich, M. W. (2007). Probabilistic diffusion tractography with multiple fibre orientations: What can we gain? *Neuroimage*, *34*(1), 144–155.

Behrens, T. E., Woolrich, M., Jenkinson, M., Johansen-Berg, H., Nunes, R., Clare, S., ... Smith, S. M. (2003). Characterization and propagation of uncertainty in diffusion-weighted mr imaging. *Magnetic resonance in medicine*, *50*(5), 1077–1088.

Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T. C., Rasmussen, D., ... Eliasmith, C. (2013). Nengo: A python tool for building large-scale functional brain models. *Frontiers in neuroinformatics*, *7*.

Bengio, Y., Lamblin, P., Popovici, D. & Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Advances in neural information processing systems* (pp. 153–160).

Benuskova, L. & Kasabov, N. (2010). *Computational neurogenetic modeling*. Springer Science & Business Media.

Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Kötter, T., Meinl, T., ... Wiswedel, B. (2008). Knime: The konstanz information miner. In *Data analysis, machine learning and applications* (pp. 319–326). Springer.

BieSSmann, F., Plis, S., Meinecke, F. C., Eichele, T. & Muller, K.-R. (2011). Analysis of multimodal neuroimaging data. *IEEE Reviews in Biomedical Engineering*, *4*, 26–58.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Bisoi, R. & Dash, P. K. (2014). A hybrid evolutionary dynamic neural network for stock market trend analysis and prediction using unscented kalman filter. *Applied Soft Computing*, *19*, 41–56.

Boahen, K. (2006). Neurogrid: Emulating a million neurons in the cortex. In *Conf. proc. ieee eng. med. biol. soc* (p. 6702).

Bohte, S. M., Kok, J. N. & La Poutre, H. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, *48*(1), 17–37.

Boksman, K., Théberge, J., Williamson, P., Drost, D. J., Malla, A., Densmore, M., ... Neufeld, R. W. (2005). A 4.0-t fmri study of brain connectivity during word fluency in first-episode schizophrenia. *Schizophrenia research*, *75*(2), 247–263.

Boynton, G. M., Engel, S. A., Glover, G. H. & Heeger, D. J. (1996). Linear systems analysis of functional magnetic resonance imaging in human v1. *The journal of neuroscience*, *16*(13), 4207–4221.

Brandeis, D. & Lehmann, D. (1989). Segments of event-related potential map series reveal landscape changes with visual attention and subjective contours. *Electroencephalography and clinical neurophysiology*, *73*(6), 507–519.

Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J. M., ... Harris Jr, F. C. et al. (2007). Simulation of networks of spiking neurons: A review of tools and strategies. *Journal of computational neuroscience*, *23*(3), 349–398.

Brown, E. N., Kass, R. E. & Mitra, P. P. (2004). Multiple neural spike train data analysis: State-of-the-art and future challenges. *Nature neuroscience*, *7*(5), 456–461.

Bugmann, G., Christodoulou, C. & Taylor, J. G. (1997). Role of temporal integration and fluctuation detection in the highly irregular firing of a leaky integrator neuron model with partial reset. *Neural Computation*, *9*(5), 985–1000.

Cabeza, R. & Nyberg, L. (2000). Imaging cognition ii: An empirical review of 275 pet and fmri studies. *Journal of cognitive neuroscience*, *12*(1), 1–47.

Calhoun, V. D., Liu, J. & Adal, T. (2009). A review of group ICA for fMRI data and ICA for joint inference of imaging, genetic, and erp data. *Neuroimage*, *45*(1), S163–S172.

Calhoun, V. D., Stevens, M., Pearlson, G. D. & Kiehl, K. (2004). Fmri analysis with the general linear model: Removal of latency-induced amplitude bias by incorporation of hemodynamic derivative terms. *Neuroimage*, *22*(1), 252–257.

Calimera, A., Macii, E. & Poncino, M. (2013). The human brain project and neuromorphic computing. *Functional neurology*, *28*(3), 191–196.

Capecci, E., Kasabov, N. & Wang, G. Y. (2015). Analysis of connectivity in neucube spiking neural network models trained on eeg data for the understanding of functional changes in the brain: A case study on opiate dependence treatment. *Neural Networks*, *68*, 62–77.

Carlson, N. R. (1967). *Foundations of physiological psychology*. WW Publisher.

Cassidy, A. S., Merolla, P., Arthur, J. V., Esser, S. K., Jackson, B., Alvarez-Icaza, R., ... Feldman, V. et al. (2013). Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores. In *Neural networks (ijcnn), the 2013 international joint conference on* (pp. 1–10). IEEE.

Cawley, G. C. & Talbot, N. L. (2010). On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, *11*(Jul), 2079–2107.

Chaitin, G. J. (1966). On the length of programs for computing finite binary sequences. *Journal of the ACM (JACM)*, *13*(4), 547–569.

Chang, P.-C. et al. (2012). A novel model by evolving partially connected neural network for stock price trend forecasting. *Expert Systems with Applications*, *39*(1), 611–620.

Chen, C.-T. (1995). *Linear system theory and design*. Oxford University Press, Inc.

Chen, K., Reiman, E. M., Huan, Z., Caselli, R. J., Bandy, D., Ayutyanont, N. & Alexander, G. E. (2009). Linking functional and structural brain images with multivariate network analyses: A novel application of the partial least square method. *Neuroimage*, *47*(2), 602–610.

Cheung, B.-S., Langevin, A. & Delmaire, H. (1997). Coupling genetic algorithm with a grid search method to solve mixed integer nonlinear programming problems. *Computers & Mathematics with Applications*, *34*(12), 13–23.

Chiou, Y., Liu, S. & Tsaih, R. (1996). Applying reasoning neural networks to the analysis and forecast of taiwans stock index variation. *Taipei Economics Inquiry, Taipei*, *34*(2), 171–200.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Cho, Z., Park, S., Kim, J., Chung, S., Chung, S., Chung, J., . . . Wong, E. (1997). Analysis of acoustic noise in mri. *Magnetic resonance imaging*, *15*(7), 815–822.

Chollet, F. et al. (2015). Keras. https://github.com/fchollet/keras. GitHub.

Chong, H. Y., Teoh, S. L., Wu, D. B.-C., Kotirum, S., Chiou, C.-F. & Chaiyakunapruk, N. (2016). Global economic burden of schizophrenia: A systematic review. *Neuropsychiatric disease and treatment*, *12*, 357.

Chu, C., Ni, Y., Tan, G., Saunders, C. J. & Ashburner, J. (2011). Kernel regression for fmri pattern prediction. *NeuroImage*, *56*(2), 662–673.

Churchland, P. S. & Sejnowski, T. J. (1988). Perspectives on cognitive neuroscience. *Science*, *242*(4879), 741.

Clark, P. & Niblett, T. (1989). The cn2 induction algorithm. *Machine learning*, *3*(4), 261–283.

Cohen, W. W. (1995). Fast effective rule induction. In *Proceedings of the twelfth international conference on machine learning* (pp. 115–123).

Cooper, M. W. (1981). A survey of methods for pure nonlinear integer programming. *Management Science*, *27*(3), 353–361.

Correa, N. M., Eichele, T., Adal, T., Li, Y.-O. & Calhoun, V. D. (2010). Multi-set canonical correlation analysis for the fusion of concurrent single trial ERP and functional MRI. *Neuroimage*, *50*(4), 1438–1445.

Correa, N. M., Li, Y.-O., Adali, T. & Calhoun, V. D. (2008). Canonical correlation analysis for feature-based fusion of biomedical imaging modalities and its application to detection of associative networks in schizophrenia. *Selected Topics in Signal Processing, IEEE Journal of*, *2*(6), 998–1007.

Correa, N., Li, Y.-O., Adali, T. & Calhoun, V. D. (2008). Examining associations between fMRI and EEG data using canonical correlation analysis. In *Biomedical imaging: From nano to macro, 2008. isbi 2008. 5th ieee international symposium on* (pp. 1251–1254). IEEE.

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297.

Courchesne, E., Akshoomoff, N., Townsend, J. & Saitoh, O. (1994). A model system for the study of attention and the cerebellum: Infantile autism. *Electroencephalography and clinical neurophysiology. Supplement*, *44*, 315–325.

Cox, R. W. (1996). Afni: Software for analysis and visualization of functional magnetic resonance neuroimages. *Computers and Biomedical research*, *29*(3), 162–173.

Creese, I., Burt, D. R. & Snyder, S. H. (1976). Dopamine receptor binding predicts clinical and pharmacological potencies of antischizophrenic drugs. *Science*, *192*(4238), 481–483.

Cui, X., Li, J. & Song, X. (2011). Xjview: A viewing program for spm.

Davison, A. P., Brüderle, D., Eppler, J., Kremkow, J., Muller, E., Pecevski, D., . . . Yger, P. (2008). Pynn: A common interface for neuronal network simulators. *Frontiers in neuroinformatics*, *2*.

De Garis, H. (1994). An artificial brain ATR's CAM-Brain project aims to build/evolve an artificial brain with a million neural net modules inside a trillion cell cellular automata machine. *New Generation Computing, 12*(2), 215–221.

De Schutter, E. & Bower, J. M. (1994). An active membrane model of the cerebellar purkinje cell. i. simulation of current clamps in slice. *Journal of neurophysiology*, *71*(1), 375–400.

de Oliveira, F. A., Nobre, C. N. & Zárate, L. E. (2013). Applying artificial neural networks to prediction of stock price and improvement of the directional prediction index–case study of petr4, petrobras, brazil. *Expert Systems with Applications*, *40*(18), 7596–7606.

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering*, *186*(2), 311–338.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons.

Deep, K., Singh, K. P., Kansal, M. L. & Mohan, C. (2009). A real coded genetic algorithm for solving integer and mixed integer optimization problems. *Applied Mathematics and Computation*, *212*(2), 505–518.

Demar, J., Zupan, B., Leban, G. & Curk, T. (2004). *Orange: From experimental machine learning to interactive data mining*. Springer.

Diesmann, M., Gewaltig, M.-O. & Aertsen, A. (1999). Stable propagation of synchronous spiking in cortical neural networks. *Nature*, *402*(6761), 529–533.

Do, L.-N. & Yang, H.-J. (2014). A robust feature selection method for classification of cognitive states with fmri data. In *Advances in computer science and its applications* (pp. 71–76). Springer.

Doborjeh, M. G., Capecci, E. & Kasabov, N. (2014). Classification and segmentation of fmri spatio-temporal brain data with a neucube evolving spiking neural network model. In *Evolving and autonomous learning systems (eals), 2014 ieee symposium on* (pp. 73–80). IEEE.

Doborjeh, M. G., Gholami, Z., Gollahalli, A. R., Kumarasinghe, K., Breen, V., Sengupta, N., ... Ge, C. (2018). *From von neumann architecture and atanasoffs abc to neuromorphic computation and kasabovs neucube. part ii: Applications*. Springer.

Doehrmann, O., Ghosh, S. S., Polli, F. E., Reynolds, G. O., Horn, F., Keshavan, A., ... Hofmann, S. G. et al. (2013). Predicting treatment response in social anxiety disorder from functional magnetic resonance imaging. *JAMA psychiatry*, *70*(1), 87–97.

Dora, S., Suresh, S. & Sundararajan, N. (2015). A sequential learning algorithm for a spiking neural classifier. *Applied Soft Computing*, *36*, 255–268.

Dorigo, M., Maniezzo, V. & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, *26*(1), 29–41.

Drewes, R. (2005). *Brainlab: A toolkit to aid in the design, simulation, and analysis of spiking neural networks with the ncs environment* (Doctoral dissertation, University of Nevada Reno).

Eichele, T., Calhoun, V. D., Moosmann, M., Specht, K., Jongsma, M. L., Quiroga, R. Q., ... Hugdahl, K. (2008). Unmixing concurrent eeg-fmri with parallel independent component analysis. *International Journal of Psychophysiology*, *67*(3), 222–234.

Elkis, H. (2007). Treatment-resistant schizophrenia. *Psychiatric Clinics of North America*, *30*(3), 511–533.

Eluvathingal, T. J., Chugani, H. T., Behen, M. E., Juhász, C., Muzik, O., Maqbool, M., . . . Makki, M. (2006). Abnormal brain connectivity in children after early severe socioemotional deprivation: A diffusion tensor imaging study. *Pediatrics*, *117*(6), 2093–2100.

Eppler, J. M., Helias, M., Muller, E., Diesmann, M. & Gewaltig, M.-O. (2008). Pynest: A convenient interface to the nest simulator. *Frontiers in neuroinformatics*, *2*.

Esmaeilzadeh, H., Blem, E., St Amant, R., Sankaralingam, K. & Burger, D. (2011). Dark silicon and the end of multicore scaling. In *Acm sigarch computer architecture news* (Vol. 39, *3*, pp. 365–376). ACM.

Essali, A., Al-Haj Haasan, N., Li, C. & Rathbone, J. (2009). Clozapine versus typical neuroleptic medication for schizophrenia. *The Cochrane Library*.

Essock, S. M., Hargreaves, W. A., Dohm, F.-A., Goethe, J., Carver, L. & Hipshman, L. (1996). Clozapine eligibility among state hospital patients. *Schizophrenia bulletin*, *22*(1), 15–25.

Farquad, M., Ravi, V. & Raju, S. B. (2012). Analytical crm in banking and finance using svm: A modified active learning-based rule extraction approach. *International Journal of Electronic Customer Relationship Management*, *6*(1), 48–73.

Felleman, D. J. & Van Essen, D. C. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex (New York, NY: 1991)*, *1*(1), 1–47.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of human genetics*, *7*(2), 179–188.

Fitzsimmons, J., Kubicki, M. & Shenton, M. E. (2013). Review of functional and anatomical brain connectivity findings in schizophrenia. *Current opinion in psychiatry*, *26*(2), 172–187.

Floudas, C. A. (1995). *Nonlinear and mixed-integer optimization: Fundamentals and applications*. Oxford University Press.

Freund, Y. & Schapire, R. E. (1999). Large margin classification using the perceptron algorithm. *Machine learning*, *37*(3), 277–296.

Fristen, K. J. (1997). Imaging cognitive anatomy. *Trends in cognitive sciences*, *1*(1), 21–27.

Friston, K. J. (1994). Statistical parametric mapping.

Friston, K. J., Buechel, C., Fink, G., Morris, J., Rolls, E. & Dolan, R. (1997). Psychophysiological and modulatory interactions in neuroimaging. *Neuroimage*, *6*(3), 218–229.

Friston, K. J., Holmes, A. P., Poline, J., Grasby, P., Williams, S., Frackowiak, R. S. & Turner, R. (1995). Analysis of fmri time-series revisited. *Neuroimage*, *2*(1), 45–53.

Friston, K. J., Josephs, O., Rees, G. & Turner, R. (1998). Nonlinear event-related responses in fmri. *Magnetic resonance in medicine*, *39*(1), 41–52.

Furber, S. (2016). Large-scale neuromorphic computing systems. *Journal of neural engineering*, *13*(5), 051001.

Furber, S. B., Galluppi, F., Temple, S. & Plana, L. A. (2014). The spinnaker project. *Proceedings of the IEEE*, *102*(5), 652–665.

Gao, J.-H., Parsons, L. M., Bower, J. M., Xiong, J. et al. (1996). Cerebellum implicated in sensory acquisition and discrimination rather than motor control. *Science*, *272*(5261), 545.

Gautrais, J. & Thorpe, S. (1998). Rate coding versus temporal order coding: A theoretical approach. *Biosystems*, *48*(1), 57–65.

George, J. S., Aine, C., Mosher, J., Schmidt, D., Ranken, D., Schlitt, H., ... Belliveau, J. (1995). Mapping function in the human brain with magnetoencephalography, anatomical magnetic resonance imaging, and functional magnetic resonance imaging. LWW.

Gerstner, W. (1998). *Spiking neurons*. MIT-press.

Gerstner, W., Kempter, R., van Hemmen, J. L. & Wagner, H. (1996). A neuronal learning rule for sub-millisecond temporal coding. *Nature*, *383*(LCN-ARTICLE-1996-002), 76–78.

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

Gerstner, W., Kistler, W. M., Naud, R. & Paninski, L. (2014). *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press.

Geschwind, N. (1974). Disconnexion syndromes in animals and man. In *Selected papers on language and the brain* (pp. 105–236). Springer.

Ghosh-Dastidar, S. & Adeli, H. (2009). Spiking neural networks. *International journal of neural systems*, *19*(04), 295–308.

Girones, J. (n.d.). J48 decision tree.

Glover, G. H. (1999). Deconvolution of impulse response in event-related bold fmri 1. *Neuroimage*, *9*(4), 416–429.

Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning, 1989. *Reading: Addison-Wesley*.

Gollisch, T. & Meister, M. (2008). Rapid neural coding in the retina with relative spike latencies. *science*, *319*(5866), 1108–1111.

Goodman, D. F. (2010). Code generation: A strategy for neural network simulators. *Neuroinformatics*, *8*(3), 183–196.

Goodman, D. F. & Brette, R. (2009). The brian simulator. *Frontiers in neuroscience*, *3*(2), 192.

Gordon, E. (2007). Integrating genomics and neuromarkers for the era of brain-related personalized medicine. *Personalized Medicine*, *4*(2), 201–215.

Greicius, M. D., Supekar, K., Menon, V. & Dougherty, R. F. (2009). Resting-state functional connectivity reflects structural connectivity in the default mode network. *Cerebral cortex*, *19*(1), 72–78.

Grossmann, I. E. (2002). Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and engineering*, *3*(3), 227–252.

Grunwald, P. & Vitányi, P. (2004). Shannon information and kolmogorov complexity. *arXiv preprint cs/0410002*.

Hafezi, R., Shahrabi, J. & Hadavandi, E. (2015). A bat-neural network multi-agent system (bnnmas) for stock price prediction: Case study of dax stock price. *Applied Soft Computing*, *29*, 196–210.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. & Witten, I. H. (2009). The weka data mining software: An update. *ACM SIGKDD explorations newsletter*, *11*(1), 10–18.

Hallock, R. M. & Di Lorenzo, P. M. (2006). Temporal coding in the gustatory system. *Neuroscience & Biobehavioral Reviews*, *30*(8), 1145–1160.

Hamilton, N. B., Attwell, D. & Hall, C. N. (2010). Pericyte-mediated regulation of capillary diameter: A component of neurovascular coupling in health and disease. *Frontiers in neuroenergetics*, *2*.

Hamstra, D. A., Rehemtulla, A. & Ross, B. D. (2007). Diffusion magnetic resonance imaging: A biomarker for treatment response in oncology. *Journal of clinical oncology*, *25*(26), 4104–4109.

Hassabis, D., Kumaran, D., Summerfield, C. & Botvinick, M. (2017). Neuroscience-inspired artificial intelligence. *Neuron*, *95*(2), 245–258.

Haxby, J. V., Gobbini, M. I., Furey, M. L., Ishai, A., Schouten, J. L. & Pietrini, P. (2001). Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science*, *293*(5539), 2425–2430.

Haynes, J.-D. & Rees, G. (2005). Predicting the orientation of invisible stimuli from activity in human primary visual cortex. *Nature neuroscience*, *8*(5), 686–691.

Hebb, D. O. (1949). *The organization of behavior: A neuropsychological approach*. John Wiley & Sons.

Hines, M. L. & Carnevale, N. T. (1997). The neuron simulation environment. *Neural computation*, *9*(6), 1179–1209.

Hines, M. L. & Carnevale, N. T. (2006). The neuron simulation environment. *NEURON*, *9*(6).

Hinton, G. E., Mcclelland, J. L. & Rumelhart, D. E. (1986). Distributed representations, parallel distributed processing: Explorations in the microstructure of cognition, vol. 1: Foundations. MIT Press, Cambridge, MA.

Hinton, G. E. & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504–507.

Hinton, G. E. & Sejnowski, T. J. (1986). Learning and releaming in boltzmann machines. *Parallel Distrilmted Processing*, *1*.

Historical - Indices. (n.d.). http://www.bseindia.com/indices/IndexArchiveData.aspx.

Hochreiter, S. & Schmidhuber, J. (1997a). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hochreiter, S. & Schmidhuber, J. (1997b). Lstm can solve hard long time lag problems. In *Advances in neural information processing systems* (pp. 473–479).

Hodgkin, A. L. & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, *117*(4), 500–544.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, *79*(8), 2554–2558.

Horovitz, S. G., Skudlarski, P. & Gore, J. C. (2002). Correlations and dissociations between bold signal and p300 amplitude in an auditory oddball task: A parametric approach to combining fmri and erp. *Magnetic resonance imaging*, *20*(4), 319–325.

Horvath, G. (2003). Cmac neural network as an svm with b-spline kernel functions. In *Instrumentation and measurement technology conference, 2003. imtc'03. proceedings of the 20th ieee* (Vol. 2, pp. 1108–1113). IEEE.

Horwitz, B., Grady, C. L., Haxby, J. V., Schapiro, M. B., Rapoport, S. I., Ungerleider, L. G. & Mishkin, M. (1992). Functional associations among human posterior extrastriate brain regions during object and spatial vision. *Journal of Cognitive Neuroscience*, *4*(4), 311–322.

Horwitz, B. & Poeppel, D. (2002). How can eeg/meg and fmri/pet data be combined? *Human brain mapping*, *17*(1), 1–3.

Horwitz, B., Soncrant, T. T. & Haxby, J. V. (1992). Covariance analysis of functional interactions in the brain using metabolic and blood flow data. In *Advances in metabolic mapping techniques for brain imaging of behavioral and learning functions* (pp. 189–217). Springer.

Horwitz, B., Tagamets, M. & McIntosh, A. R. (1999). Neural modeling, functional brain imaging, and cognition. *Trends in cognitive sciences*, *3*(3), 91–98.

Hough, M., De Garis, H., Korkin, M., Gers, F. & Nawa, N. E. (1999). SPIKER: Analog waveform to digital spiketrain conversion in atrs artificial brain (cam-brain) project. In *International conference on robotics and artificial life*. Citeseer.

Hsu, J. (2014). Ibm's new brain [news]. *IEEE spectrum*, *51*(10), 17–19.

Hu, J., Hou, Z.-G., Chen, Y.-X., Kasabov, N. & Scott, N. (2014). Eeg-based classification of upper-limb adl using snn for active robotic rehabilitation. In *Biomedical robotics and biomechatronics (2014 5th ieee ras & embs international conference on* (pp. 409–414). IEEE.

Hua, Z. & Huang, F. (2006). An effective genetic algorithm approach to large scale mixed integer programming problems. *Applied Mathematics and computation*, *174*(2), 897–909.

Huang, C.-J., Yang, D.-X. & Chuang, Y.-T. (2008). Application of wrapper approach and composite classifier to the stock trend prediction. *Expert Systems with Applications*, *34*(4), 2870–2878.

Huang, W., Nakamori, Y. & Wang, S.-Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, *32*(10), 2513–2522.

Hubel, D. H. & Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, *148*(3), 574–591.

Hubel, D. H. & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, *160*(1), 106–154.

Iakymchuk, T., Rosado-Munoz, A., Bataller-Mompean, M., Guerrero-Martinez, J., Frances-Villora, J., Wegrzyn, M. & Adamski, M. (2014). Hardware-accelerated spike train generation for neuromorphic image and video processing. In *Programmable logic (spl), 2014 ix southern conference on* (pp. 1–6). IEEE.

Iasevoli, F., Giordano, S., Balletta, R., Latte, G., Formato, M. V., Prinzivalli, E., ... de Bartolomeis, A. (2016). Treatment resistant schizophrenia is associated with the worst community functioning among severely-ill highly-disabling psychiatric conditions and is the most relevant predictor of poorer achievements in functional milestones. *Progress in Neuro-Psychopharmacology and Biological Psychiatry*, *65*, 34–48.

Izhikevich, E. M. (2001). Resonate-and-fire neurons. *Neural networks*, *14*(6), 883–894.

Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *IEEE transactions on neural networks*, *15*(5), 1063–1070.

Jenkinson, M. & Smith, S. (2001). A global optimisation method for robust affine registration of brain images. *Medical image analysis*, *5*(2), 143–156.

Jong, K. A. D. (1975). *An analysis of the behavior of a class of genetic adaptive systems* (Doctoral dissertation).

Juarez-Reyes, M. G., Shumway, M., Battle, C., Bacchetti, P., Hansen, M. S. & Hargreaves, W. A. (1995). Effects of stringent criteria on eligibility for clozapine among public mental health clients. *Psychiatric services*, *46*(8), 801–806.

Just, M. A., Newman, S. D., Keller, T. A., McEleney, A. & Carpenter, P. A. (2004). Imagery in sentence comprehension: An fmri study. *Neuroimage*, *21*(1), 112–124.

Kamitani, Y. & Tong, F. (2005). Decoding the visual and subjective contents of the human brain. *Nature neuroscience*, *8*(5), 679–685.

Kamitani, Y. & Tong, F. (2006). Decoding seen and attended motion directions from activity in the human visual cortex. *Current biology*, *16*(11), 1096–1102.

Kandel, E. R. [Eric R], Schwartz, J. H., Jessell, T. M., Siegelbaum, S. A., Hudspeth, A. J. et al. (2000). *Principles of neural science*. McGraw-hill New York.

Kane, J. M. & Correll, C. U. (2016). The role of clozapine in treatment-resistant schizophrenia. *JAMA psychiatry*, *73*(3), 187–188.

Kane, J., Honigfeld, G., Singer, J. & Meltzer, H. (1988). Clozapine for the treatment-resistant schizophrenic: A double-blind comparison with chlorpromazine. *Archives of general psychiatry*, *45*(9), 789–796.

Kara, Y., Boyacioglu, M. A. & Baykan, Ö. K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. *Expert systems with Applications*, *38*(5), 5311–5319.

Karpathy, A. (2015). The unreasonable effectiveness of recurrent neural networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

Kasabov, N. (1998). Evolving fuzzy neural networks-algorithms, applications and biological motivation. *Methodologies for the conception, design and application of soft computing, World Scientific, 1*, 271–274.

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

Kasabov, N. (2007). *Evolving connectionist systems: The knowledge engineering approach*. Springer Science & Business Media.

Kasabov, N. (2012). NeuCube evospike architecture for spatio-temporal modelling and pattern recognition of brain signals. In *Iapr workshop on artificial neural networks in pattern recognition* (pp. 225–243). Springer.

Kasabov, N. K. (2014). NeuCube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data. *Neural Networks*, *52*, 62–76.

Kasabov, N. K., Doborjeh, M. G. & Doborjeh, Z. G. (2017). Mapping, learning, visualization, classification, and understanding of fmri data in the neucube evolving spatiotemporal data machine of spiking neural networks. *IEEE transactions on neural networks and learning systems*, *28*(4), 887–899.

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*, *41*, 188–201.

Kasabov, N., Feigin, V., Hou, Z.-G., Chen, Y., Liang, L., Krishnamurthi, R., … Parmar, P. (2014). Evolving spiking neural networks for personalised modelling, classification and prediction of spatio-temporal patterns with a case study on stroke. *Neurocomputing*, *134*, 269–279.

Kasabov, N., Scott, N. M., Tu, E., Marks, S., Sengupta, N., Capecci, E., … Hartono, R. et al. (2016). Evolving spatio-temporal data machines based on the NeuCube neuromorphic framework: Design methodology and selected applications. *Neural Networks*, *78*, 1–14.

Kasabov, N., Sengupta, N. & Scott, N. (2016). From von neumann, john atanasoff and abc to neuromorphic computation and the neucube spatio-temporal data machine. In *Ieee 8th international conference on intelligent systems (is)* (pp. 15–21). IEEE.

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

Kennedy, J. L., Altar, C. A., Taylor, D. L., Degtiar, I. & Hornberger, J. C. (2014). The social and economic burden of treatment-resistant schizophrenia: A systematic literature review. *International clinical psychopharmacology*, *29*(2), 63–76.

Khodayari-Rostamabad, A., Hasey, G. M., MacCrimmon, D. J., Reilly, J. P. & de Bruin, H. (2010). A pilot study to determine whether machine learning methodologies using pre-treatment electroencephalography can predict the symptomatic response to clozapine therapy. *Clinical Neurophysiology*, *121*(12), 1998–2006.

Khodayari-Rostamabad, A., Reilly, J. P., Hasey, G. M., de Bruin, H. & MacCrimmon, D. J. (2013). A machine learning approach using EEG data to predict response to ssri treatment for major depressive disorder. *Clinical Neurophysiology*, *124*(10), 1975–1985.

Kim, K.-j. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, *55*(1), 307–319.

Kim, K.-j. & Han, I. (2000). Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert systems with Applications*, *19*(2), 125–132.

Kim, K.-j. & Lee, W. B. (2004). Stock market prediction using artificial neural networks with optimal feature transformation. *Neural computing & applications*, *13*(3), 255–260.

Kim, S., Ugurbil, K., Strick, P. et al. (1994). Activation of a cerebellar output nucleus during cognitive processing. *SCIENCE-NEW YORK THEN WASHINGTON-*, 949–949.

Kistler, W. M., Gerstner, W. & van Hemmen, J. L. (1997). Reduction of the hodgkin-huxley equations to a single-variable threshold model. *Neural computation*, *9*(5), 1015–1045.

Kleiser, R., Staempfli, P., Valavanis, A., Boesiger, P. & Kollias, S. (2010). Impact of fmri-guided advanced dti fiber tracking techniques on their clinical applications in patients with brain tumors. *Neuroradiology*, *52*(1), 37.

Knott, V., Labelle, A., Jones, B. & Mahoney, C. (2000). Eeg hemispheric asymmetry as a predictor and correlate of short-term response to clozapine treatment in schizophrenia. *Clinical Electroencephalography*, *31*(3), 145–152.

Koch, M. (1999). The neurobiology of startle. *Progress in neurobiology*, *59*(2), 107–128.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, *78*(9), 1464–1480.

Kohonen, T. (1998). The self-organizing map. *Neurocomputing*, *21*(1), 1–6.

Kolmogorov, A. N. (1965). Three approaches to the quantitative definition ofinformation'. *Problems of information transmission*, *1*(1), 1–7.

Kostal, L., Lansky, P. & Rospars, J.-P. (2007). Neuronal coding and spiking randomness. *European Journal of Neuroscience*, *26*(10), 2693–2701.

Kotsiantis, S. B., Zaharakis, I. & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques.

Kumar, M. & Thenmozhi, M. (2006). Forecasting stock index movement: A comparison of support vector machines and random forest.

Langford, J., Li, L. & Strehl, A. (2007). Vowpal wabbit online learning project. Technical report, http://hunch. net.

Laufs, H., Holt, J. L., Elfont, R., Krams, M., Paul, J. S., Krakow, K. & Kleinschmidt, A. (2006). Where the bold signal goes when alpha eeg leaves. *Neuroimage*, *31*(4), 1408–1418.

Lehman, A. F., Lieberman, J. A., Dixon, L. B., McGlashan, T. H., Miller, A. L., Perkins, D. O., . . . Altshuler, K. et al. (2004). Practice guideline for the treatment of partients with schizophrenia. *American Journal of Psychiatry*, *161*(2 SUPPL.).

Li, J.-l., Wang, L.-q., Li, C.-x., Han, B. & Zhang, Z.-c. (2005). Multi-pwm pulse generator based fpga. *PROCEEDINGS-CHINESE SOCIETY OF ELECTRICAL ENGINEERING*, *25*(10), 55.

Li, Y.-X. & Gen, M. (1996). Nonlinear mixed integer programming problems using genetic algorithm and penalty function. In *Systems, man, and cybernetics, 1996., ieee international conference on* (Vol. 4, pp. 2677–2682). IEEE.

Lieberman, J. A. & Murray, R. M. (2012). *Comprehensive care of schizophrenia: A textbook of clinical management*. Oxford University Press.

Lin, C.-C., Wang, Y.-C., Chen, J.-Y., Liou, Y.-J., Bai, Y.-M., Lai, I.-C., . . . Li, Y.-C. (2008). Artificial neural network prediction of clozapine response with combined pharmacogenetic and clinical data. *Computer methods and programs in biomedicine*, *91*(2), 91–99.

Lin, Y., Guo, H. & Hu, J. (2013). An svm-based approach for stock market trend prediction. In *Neural networks (ijcnn), the 2013 international joint conference on* (pp. 1–7). IEEE.

Littlestone, N. & Warmuth, M. K. (1994). The weighted majority algorithm. *Information and computation*, *108*(2), 212–261.

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

Luo, Y.-C., Guignard, M. & Chen, C.-H. (2001). A hybrid approach for integer programming combining genetic algorithms, linear programming and ordinal optimization. *Journal of Intelligent Manufacturing*, *12*(5), 509–519.

Maass, W. (1997). Networks of spiking neurons: The third generation of neural network models. *Neural networks*, *10*(9), 1659–1671.

Maass, W. & Bishop, C. M. (2001). *Pulsed neural networks*. MIT press.

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.

Mainen, Z. F. & Sejnowski, T. J. (1995). Reliability of spike timing in neocortical neurons. *Science*, *268*(5216), 1503–1506.

Mainen, Z. F. & Sejnowski, T. J. (1998). Modeling active dendritic processes in pyramidal neurons. *Methods in neuronal modeling*, 171–210.

Maiti, A., Bhunia, A. & Maiti, M. (2006). An application of real-coded genetic algorithm (rcga) for mixed integer non-linear programming in two-storage multi-item inventory model with discount policy. *Applied Mathematics and computation*, *183*(2), 903–915.

Marian, I. (2002). A biologically inspired model of motor control of direction. *Master's thesis, University College Dublin*.

Markram, H., Meier, K., Lippert, T., Grillner, S., Frackowiak, R., Dehaene, S., . . . DeFelipe, J. et al. (2011). Introducing the human brain project. *Procedia Computer Science*, *7*, 39–42.

Matheson, S., Shepherd, A. & Carr, V. (2014). How much do we know about schizophrenia and how well do we know it? evidence from the schizophrenia library. *Psychological medicine*, *44*(16), 3387–3405.

Maunsell, J. H. & Gibson, J. R. (1992). Visual response latencies in striate cortex of the macaque monkey. *Journal of neurophysiology*, *68*(4), 1332–1344.

McClure, E. B., Adler, A., Monk, C. S., Cameron, J., Smith, S., Nelson, E. E., . . . Pine, D. S. (2007). Fmri predictors of treatment outcome in pediatric anxiety disorders. *Psychopharmacology*, *191*(1), 97–105.

McCulloch, W. S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

McGrath, J., Saha, S., Chant, D. & Welham, J. (2008). Schizophrenia: A concise overview of incidence, prevalence, and mortality. *Epidemiologic reviews*, *30*(1), 67–76.

McNabb, C. B. (2017). *Brain dysconnectivity as a biomarker of treatment resistance in schizophrenia* (Doctoral dissertation, The University of Auckland).

Mead, C. (1990). Neuromorphic electronic systems. *Proceedings of the IEEE*, *78*(10), 1629–1636.

Medical image computing. (n.d.). https://en.wikipedia.org/wiki/Medical_image_computing.

Mel, B. W. (1994). Information processing in dendritic trees. *Neural computation*, *6*(6), 1031–1085.

Meltzer, H. Y. (2010). Role of clozapine in treatment-resistant schizophrenia. In *Therapy-resistant schizophrenia* (Vol. 26, pp. 114–128). Karger Publishers.

Menon, V. & Crottaz-Herbette, S. (2005). Combined eeg and fmri studies of human brain function. *Int Rev Neurobiol*, *66*, 291–321.

Mesulam, M. et al. (1990). Large-scale neurocognitive networks and distributed processing for attention, language, and memory. *Annals of neurology*, *28*(5), 597–613.

Michalski, R. S. (1980). Learning by being told and learning by examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease analysis. *Int. J. of Policy Analysis and Information Systems*, *4*(2).

Minsky, M. & Papert, S. (1969). Perceptrons.

Mitchell, T. M. et al. (1997). Machine learning. wcb. McGraw-Hill Boston, MA:

Mitchell, T. M., Hutchinson, R., Just, M. A., Niculescu, R. S., Pereira, F. & Wang, X. (2003). Classifying instantaneous cognitive states from fmri data. In *American medical informatics association annual symposium*.

Mitchell, T. M., Hutchinson, R., Niculescu, R. S., Pereira, F., Wang, X., Just, M. & Newman, S. (2004). Learning to decode cognitive states from brain images. *Machine learning*, *57*(1), 145–175.

Mitchell, T. M., Shinkareva, S. V., Carlson, A., Chang, K.-M., Malave, V. L., Mason, R. A. & Just, M. A. (2008). Predicting human brain activity associated with the meanings of nouns. *science*, *320*(5880), 1191–1195.

Mitchell, T. M. & Wang, W. (2001). Starplus fmri data.

Mohemmed, A., Schliebs, S., Matsuda, S. & Kasabov, N. (2012). Span: Spike pattern association neuron for learning spatio-temporal spike patterns. *International Journal of Neural Systems*, *22*(04), 1250012.

Mostafa, M. M. (2010). Forecasting stock exchange movements using neural networks: Empirical evidence from kuwait. *Expert Systems with Applications*, *37*(9), 6302–6309.

Murthy, S. K. (1998). Automatic construction of decision trees from data: A multi-disciplinary survey. *Data mining and knowledge discovery*, *2*(4), 345–389.

Muthukumaraswamy, S. D. & Singh, K. D. (2008). Spatiotemporal frequency tuning of bold and gamma band meg responses compared in primary visual cortex. *Neuroimage*, *40*(4), 1552–1560.

Nair, B. B., Mohandas, V. & Sakthivel, N. (2010). A decision treerough set hybrid system for stock market trend prediction. *International Journal of Computer Applications*, *6*(9), 1–6.

Natschläger, T., Maass, W. & Markram, H. (2002). The" liquid computer": A novel strategy for real-time computing on time series. *Special issue on Foundations of Information Processing of TELEMATIK*, *8*(LNMC-ARTICLE-2002-005), 39–43.

Natschläger, T., Markram, H. & Maass, W. (2003). Computer models and analysis tools for neural microcircuits. In *Neuroscience databases* (pp. 123–138). Springer.

NeuCom. (n.d.). http://www.theneucom.com. Accessed: 2015-08-15.

Ng, A. (2011). Sparse autoencoder. *CS294A Lecture notes*, *72*(2011), 1–19.

NIBIB. (2017). Fmri scanner. http://www.advancedimagingsouthbay.com/services-mri-scan.html.

Nicolini, C. & Bifone, A. (2016). Modular structure of brain functional networks: Breaking the resolution limit by surprise. *Scientific reports*, *6*.

Nissen, S. & Nemerson, E. (2000). Fast artificial neural network library. *Available at leenissen. dk/fann/html/files/fann-h. html*.

Norman, K. A., Polyn, S. M., Detre, G. J. & Haxby, J. V. (2006). Beyond mind-reading: Multi-voxel pattern analysis of fmri data. *Trends in cognitive sciences*, *10*(9), 424–430.

NSE - National Stock Exchange of India Ltd. (n.d.). https://www.nseindia.com/products/content/equities/indices/historical_index_data.htm.

Nunez, M. D., Nunez, P. L. & Srinivasan, R. (2016). Handbook of neuroimaging data analysis. (Chap. Electroencephalography (EEG): neurophysics, experimental methods, and signal processing). Chapman & Hall/CRC.

Nuntalid, N., Dhoble, K. & Kasabov, N. (2011). Eeg classification with bsa spike encoding algorithm and evolving probabilistic spiking neural network. In *International conference on neural information processing* (pp. 451–460). Springer.

Olah, C. (2015). Understanding lstm. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Olshausen, B. A. & Field, D. J. (2006). What is the other 85 percent of v1 doing. *L. van Hemmen, & T. Sejnowski (Eds.) 23*, 182–211.

Ou, P. & Wang, H. (2009). Prediction of stock market index movement by ten data mining techniques. *Modern Applied Science*, *3*(12), 28.

Ou, W., Cameron, P. & Thomas, D. (1992). Anatomical evidence for cerebellar and basal ganglia involvement in higher cognitive function. *Biol*, *2*, 227.

Pachitariu, M., Brody, C. D., Jun, P. J. K. & Holmes, P. J. (n.d.). Probabilistic models for spike trains of single neurons.

Pandya, D. N. & Yeterian, E. H. (1985). Architecture and connections of cortical association areas. In *Association and auditory cortices* (pp. 3–61). Springer.

Park, J. & Sandberg, I. W. (1991). Universal approximation using radial-basis-function networks. *Neural computation*, *3*(2), 246–257.

Patel, J., Shah, S., Thakkar, P. & Kotecha, K. (2015). Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications*, *42*(1), 259–268.

Patel, M. J., Andreescu, C., Price, J. C., Edelman, K. L., Reynolds, C. F. & Aizenstein, H. J. (2015). Machine learning approaches for integrating clinical and imaging features in late-life depression classification and response prediction. *International journal of geriatric psychiatry*, *30*(10), 1056–1067.

Patil, P. G., Carmena, J. M., Nicolelis, M. A. & Turner, D. A. (2004). Ensemble recordings of human subcortical neurons as a source of motor control signals for a brain-machine interface. *Neurosurgery*, *55*(1), 27–38.

Pecevski, D. (n.d.). Oger: Modular learning architectures for large-scale sequential processing.

Pecevski, D., Natschläger, T. & Schuch, K. (2009). Pcsim: A parallel simulation environment for neural circuits fully integrated with python. *Frontiers in neuroinformatics*, *3*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Perrin, D. (2011). Complexity and high-end computing in biology and medicine. In *Software tools and algorithms for biological systems* (pp. 377–384). Springer.

Plis, S. M., Weisend, M. P., Damaraju, E., Eichele, T., Mayer, A., Clark, V. P., ... Calhoun, V. D. (2011). Effective connectivity analysis of fMRI and MEG data collected under identical paradigms. *Computers in biology and medicine*, *41*(12), 1156–1165.

Ponulak, F. & Kasiski, A. (2010). Supervised learning in spiking neural networks with resume: Sequence learning, classification, and spike shifting. *Neural Computation*, *22*(2), 467–510.

Posner, M. I., Petersen, S., Fox, P. & Raichle, M. (1988). Localization of cognitive operations in the human brain. *Science*, *240*(4859), 1627–1631.

Price, G., Cercignani, M., Parker, G. J., Altmann, D. R., Barnes, T. R., Barker, G. J., ... Ron, M. A. (2007). Abnormal brain connectivity in first-episode psychosis: A diffusion mri tractography study of the corpus callosum. *Neuroimage*, *35*(2), 458–466.

Pruim, R. H., Mennes, M., van Rooij, D., Llera, A., Buitelaar, J. K. & Beckmann, C. F. (2015). Ica-aroma: A robust ica-based strategy for removing motion artifacts from fmri data. *Neuroimage*, *112*, 267–277.

Quandl Financial, Economic and Alternative Data. (n.d.). https://www.quandl.com/.

Quarantelli, M., Salvatore, E., Giorgio, S. M. D. A., Filla, A., Cervo, A., Russo, C. V., ... De Michele, G. (2013). Default-mode network changes in huntingtons disease: An integrated mri study of functional connectivity and morphometry. *PloS one*, *8*(8), e72159.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, *1*(1), 81–106.

Quinlan, J. R. (2014). *C4. 5: Programs for machine learning*. Elsevier.

Randell, B. (2013). *The origins of digital computers: Selected papers*. Springer.

Ravan, M., Hasey, G., Reilly, J. P., MacCrimmon, D. & Khodayari-Rostamabad, A. (2015). A machine learning approach using auditory odd-ball responses to investigate the effect of clozapine therapy. *Clinical Neurophysiology*, *126*(4), 721–730.

Rodrguez, V. M., Andreé, R. M., Castejón, M., Garca, E. C. et al. (1996). Spect study of regional cerebral perfusion in neuroleptic-resistant schizophrenic patients who responded or did not respond to clozapine. *The American Journal of Psychiatry*.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.

Rössler, W., Salize, H. J., van Os, J. & Riecher-Rössler, A. (2005). Size of burden of schizophrenia and psychotic disorders. *European Neuropsychopharmacology*, *15*(4), 399–409.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, *5*(3), 1.

Russell, S. (2003). Norvig (2003). *Artificial intelligence: a modern approach*, 25–26.

Saad, E. W., Prokhorov, D. V. & Wunsch, D. C. (1998). Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. *IEEE Transactions on neural networks*, *9*(6), 1456–1470.

Schaller, R. R. (1997). Moore's law: Past, present and future. *IEEE spectrum*, *34*(6), 52–59.

Schaul, T., Bayer, J., Wierstra, D., Sun, Y., Felder, M., Sehnke, F., ... Schmidhuber, J. (2010). Pybrain. *The Journal of Machine Learning Research*, *11*, 743–746.

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

Schrauwen, B. & Van Campenhout, J. (2003). Bsa, a fast and accurate spike train encoding scheme. In *Proceedings of the international joint conference on neural networks* (Vol. 4, pp. 2825–2830). IEEE Piscataway, NJ.

Schrauwen, B. & Van Campenhout, J. (2004). Extending spikeprop. In *Neural networks, 2004. proceedings. 2004 ieee international joint conference on* (Vol. 1, pp. 471–475). IEEE.

Schrauwen, B., Verstraeten, D. & Van Campenhout, J. (2007). An overview of reservoir computing: Theory, applications and implementations. In *Proceedings of the 15th european symposium on artificial neural networks. p. 471-482 2007* (pp. 471–482).

Schuler, I. (2015). Neuromorphic computing: From materials to systems architecture. Accessed: 2016-07-16.

Scott, N. (2015). *Evolving spiking neural networks for spatio- and spectro- temporal data analysis: Models, implementations, applications* (Doctoral dissertation, Auckland University of Technology).

Segev, I., Burke, R. E. & Hines, M. (1998). Compartmental models of complex neurons. *Methods in neuronal modeling: From ions to networks*, *2*.

Sengupta, N. & Kasabov, N. (2017). Spike-time encoding as a data compression technique for pattern recognition of temporal data. *Information Sciences*, *406-407*, 133–145. doi:https://doi.org/10.1016/j.ins.2017.04.017

Sengupta, N., McNabb, C. B., Kasabov, N. & Russell, B. R. (2018). Integrating space, time, and orientation in spiking neural networks: A case study on multimodal brain data modeling. *IEEE Transactions on Neural Networks and Learning Systems*, *PP*(99), 1–15. doi:10.1109/TNNLS.2018.2796023

Sengupta, N., Ramos, J. I. E., Tu, E., Marks, S., Scott, N., Weclawski, J., ... Abbott, A. (2018). *From von neumann architecture and atanasoffs abc to neuromorphic computation and kasabovs neucube: Principles and implementations*. Springer.

Sengupta, N., Scott, N. & Kasabov, N. (2015). Framework for knowledge driven optimisation based data encoding for brain data modelling using spiking neural network architecture. In *Proceedings of the fifth international conference on fuzzy and neuro computing (fancco - 2015)* (pp. 109–118). Cham: Springer International Publishing. doi:10.1007/978-3-319-27212-2_9

Shallice, T. (1988). *From neuropsychology to mental structure*. Cambridge University Press.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, *27*(3), 379–423.

Shinkareva, S. V., Mason, R. A., Malave, V. L., Wang, W., Mitchell, T. M. & Just, M. A. (2008). Using fmri brain activation to identify cognitive states associated with perception of tools and dwellings. *PLoS One*, *3*(1), e1394.

Siddique, M. & Tokhi, M. (2001). Training neural networks: Backpropagation vs. genetic algorithms. In *Neural networks, 2001. proceedings. ijcnn'01. international joint conference on* (Vol. 4, pp. 2673–2678). IEEE.

Sierra, C. V. R., Versluis, M. J., Hoogduin, J. M. & Duifhuis, H. (2008). Acoustic fmri noise: Linear time-invariant system model. *IEEE Transactions on Biomedical Engineering*, *55*(9), 2115–2123.

Sjöström, J. & Gerstner, W. (2010a). Spike-timing dependent plasticity. *Spike-timing dependent plasticity*, 35.

Sjöström, J. & Gerstner, W. (2010b). Spike-timing dependent plasticity. *Scholarpedia*, *5*(2), 1362. revision #151671. doi:10.4249/scholarpedia.1362

Smith, S. M. (2002). Fast robust automated brain extraction. *Human brain mapping*, *17*(3), 143–155.

Smith, S. M., Fox, P. T., Miller, K. L., Glahn, D. C., Fox, P. M., Mackay, C. E., ... Laird, A. R. et al. (2009). Correspondence of the brain's functional architecture during activation and rest. *Proceedings of the National Academy of Sciences*, *106*(31), 13040–13045.

Smith, S. M., Jenkinson, M., Woolrich, M. W., Beckmann, C. F., Behrens, T. E., Johansen-Berg, H., ... Flitney, D. E. et al. (2004). Advances in functional and structural mr image analysis and implementation as fsl. *Neuroimage*, *23*, S208–S219.

Song, S., Miller, K. D. & Abbott, L. F. (2000). Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature neuroscience*, *3*(9), 919–926.

Specht, D. F. (1990). Probabilistic neural networks. *Neural networks*, *3*(1), 109–118.

Spitzer, R. L. & Williams, J. B. (1980). Diagnostic and statistical manual of mental disorders. In *American psychiatric association*. Citeseer.

Sporns, O. & Betzel, R. F. (2016). Modular brain networks. *Annual review of psychology, 67*, 613–640.

Squire, L. & Kandel, E. (1999). Memory: From molecules to memory: Scientific american library. New York, Freeman Press.

Stämpfli, P., Reischauer, C., Jaermann, T., Valavanis, A., Kollias, S. & Boesiger, P. (2008). Combining fmri and dti: A framework for exploring the limits of fmri-guided dti fiber tracking and for verifying dti-based fiber tractography results. *Neuroimage, 39*(1), 119–126.

Stephan, K. E., Friston, K. J. & Frith, C. D. (2009). Dysconnection in schizophrenia: From abnormal synaptic plasticity to failures of self-monitoring. *Schizophrenia bulletin*, sbn176.

Sui, J., Adali, T., Yu, Q., Chen, J. & Calhoun, V. D. (2012). A review of multivariate methods for multimodal fusion of brain imaging data. *Journal of neuroscience methods, 204*(1), 68–81.

Supekar, K., Menon, V., Rubin, D., Musen, M. & Greicius, M. D. (2008). Network analysis of intrinsic functional brain connectivity in alzheimer's disease. *PLoS computational biology, 4*(6), e1000100.

Sutton, R. S. & Barto, A. G. (1998). *Reinforcement learning: An introduction*. MIT press Cambridge.

Taylor, D., Scott, N., Kasabov, N., Capecci, E., Tu, E., Saywell, N., . . . Hou, Z.-G. (2014). Feasibility of neucube snn architecture for detecting motor execution and motor intention for use in bciapplications. In *Neural networks (ijcnn), 2014 international joint conference on* (pp. 3221–3225). IEEE.

Teipel, S. J., Bokde, A. L., Meindl, T., Amaro, E., Soldner, J., Reiser, M. F., . . . Hampel, H. (2010). White matter microstructure underlying default mode network connectivity in the human brain. *Neuroimage, 49*(3), 2021–2032.

Theunissen, F. & Miller, J. P. (1995). Temporal encoding in nervous systems: A rigorous definition. *Journal of computational neuroscience, 2*(2), 149–162.

Thompson, S. E. & Parthasarathy, S. (2006). Moore's law: The future of si microelectronics. *Materials today, 9*(6), 20–25.

Thorpe, S. J. & Gautrais, J. (1997). Rapid visual processing using spike asynchrony. *Advances in neural information processing systems*, 901–907.

Thorpe, S. J. & Imbert, M. (1989). Biological constraints on connectionist modelling. *Connectionism in perspective*, 63–92.

Thorpe, S., Delorme, A. & Van Rullen, R. (2001). Spike-based strategies for rapid processing. *Neural networks, 14*(6), 715–725.

Tieleman, T. (2008). Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on machine learning* (pp. 1064–1071). ACM.

Toumey, C. (2016). Less is moore. *Nature Nanotechnology, 11*, 2–3.

Tsaih, R., Hsu, Y. & Lai, C. C. (1998). Forecasting s&p 500 stock index futures with a hybrid ai system. *Decision Support Systems, 23*(2), 161–174.

Tsien, J. Z., Li, M., Osan, R., Chen, G., Lin, L., Wang, P. L., . . . Liu, T. et al. (2013). On brain activity mapping: Insights and lessons from brain decoding project to map memory patterns in the hippocampus. *Science China. Life Sciences*, *56*(9), 767.

Tu, E., Kasabov, N., Othman, M., Li, Y., Worner, S., Yang, J. & Jia, Z. (2014). Neucube (st) for spatio-temporal data predictive modelling with a case study on ecological data. In *Neural networks (ijcnn), 2014 international joint conference on* (pp. 638–645). IEEE.

Tu, E., Kasabov, N. & Yang, J. (2017). Mapping temporal variables into the neucube for improved pattern recognition, predictive modeling, and understanding of stream data. *IEEE transactions on neural networks and learning systems*, *28*(6), 1305–1317.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, *59*(236), 433–460.

Valdes-Sosa, P. A., Sanchez-Bornot, J. M., Sotero, R. C., Iturria-Medina, Y., Aleman-Gomez, Y., Bosch-Bayard, J., . . . Ozaki, T. (2009). Model driven EEG/fMRI fusion of brain oscillations. *Human brain mapping*, *30*(9), 2701–2721.

Van Rullen, R., Gautrais, J., Delorme, A. & Thorpe, S. (1998). Face processing using one spike per neurone. *Biosystems*, *48*(1), 229–239.

Van Rullen, R. & Thorpe, S. J. (2001). Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex. *Neural computation*, *13*(6), 1255–1283.

van Veen, F. (2016). The neural network zoo. http://www.asimovinstitute.org/neural-network-zoo/.

Vazquez, A. L. & Noll, D. C. (1998). Nonlinear aspects of the bold response in functional mri. *Neuroimage*, *7*(2), 108–118.

Veropoulos, K., Campbell, C., Cristianini, N. et al. (1999). Controlling the sensitivity of support vector machines. In *Proceedings of the international joint conference on ai* (pp. 55–60).

Victor, J. D. & Purpura, K. P. (1997). Metric-space analysis of spike trains: Theory, algorithms and application. *Network: computation in neural systems*, *8*(2), 127–164.

Vivarelli, F. & Williams, C. K. (2001). Comparing bayesian neural network algorithms for classifying segmented outdoor images. *Neural Networks*, *14*(4), 427–437.

Vos, T., Barber, R. M., Bell, B., Bertozzi-Villa, A., Biryukov, S., Bolliger, I., . . . Dicker, D. et al. (2015). Global, regional, and national incidence, prevalence, and years lived with disability for 301 acute and chronic diseases and injuries in 188 countries, 1990–2013: A systematic analysis for the global burden of disease study 2013. *The Lancet*, *386*(9995), 743–800.

Wang, Y. (2014). Stock price direction prediction by directly using prices data: An empirical study on the kospi and hsi. *International Journal of Business Intelligence and Data Mining*, *9*(2), 145–160.

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

Wheeler, A. J. (2008). Treatment pathway and patterns of clozapine prescribing for schizophrenia in new zealand. *Annals of Pharmacotherapy*, *42*(6), 852–860.

Widrow, B. & Lehr, M. A. (1990). 30 years of adaptive neural networks: Perceptron, madaline, and backpropagation. *Proceedings of the IEEE*, *78*(9), 1415–1442.

Wilson, E. C. (2001). *Parallel implementation of a large scale biologically realistic neocortical neural network simulator* (Doctoral dissertation, University of Nevada Reno).

Wilson, M. A., Bhalla, U. S., Uhley, J. D. & Bower, J. M. (1989). Genesis: A system for simulating neural networks. In *Advances in neural information processing systems* (pp. 485–492).

Wilson, R. I. (2008). Neural and behavioral mechanisms of olfactory perception. *Current opinion in neurobiology*, *18*(4), 408–412.

Woolrich, M. W., Ripley, B. D., Brady, M. & Smith, S. M. (2001). Temporal autocorrelation in univariate linear modeling of fmri data. *Neuroimage*, *14*(6), 1370–1386.

Wysoski, S. G., Benuskova, L. & Kasabov, N. (2006). Adaptive learning procedure for a network of spiking neurons and visual pattern recognition. In *International conference on advanced concepts for intelligent vision systems* (pp. 1133–1142). Springer.

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

Yang, H., Liu, J., Sui, J., Pearlson, G. & Calhoun, V. D. (2010). A hybrid machine learning method for fusing fMRI and genetic data: Combining both improves classification of schizophrenia. *Front Hum Neurosci*, *4*(192.10), 3389.

Yeganeh-Doost, P., Gruber, O., Falkai, P. & Schmitt, A. (2011). The role of the cerebellum in schizophrenia: From cognition to molecular pathways. *Clinics*, *66*, 71–77.

Yen, G. G. & Lu, H. (2000). Hierarchical genetic algorithm based neural network design. In *Combinations of evolutionary computation and neural networks, 2000 ieee symposium on* (pp. 168–175). IEEE.

Yiqing, L., Xigang, Y. & Yongjian, L. (2007). An improved pso algorithm for solving non-convex nlp/minlp problems with equality constraints. *Computers & chemical engineering*, *31*(3), 153–162.

Yokota, T., Gen, M. & Li, Y.-X. (1996). Genetic algorithm for non-linear mixed integer programming problems and its applications. *Computers & industrial engineering*, *30*(4), 905–917.

Yu, Q., A Allen, E., Sui, J., R Arbabshirani, M., Pearlson, G. & D Calhoun, V. (2012). Brain connectivity networks in schizophrenia underlying resting state functional magnetic resonance imaging. *Current topics in medicinal chemistry*, *12*(21), 2415–2425.

Zhang, X.-d., Li, A. & Pan, R. (2016). Stock trend prediction based on a new status box method and adaboost probabilistic support vector machine. *Applied Soft Computing*, *49*, 385–398.

Zhang, Y. [Yongyue], Brady, M. & Smith, S. (2001). Segmentation of brain mr images through a hidden markov random field model and the expectation-maximization algorithm. *IEEE transactions on medical imaging*, *20*(1), 45–57.

Zhang, Y. [Yudong] & Wu, L. (2009). Stock market prediction of s&p 500 via combination of improved bco approach and bp neural network. *Expert systems with applications*, *36*(5), 8849–8854.

A Sequential eSNN Architectures for Cyber Fraud Detection

The sequential eSNN classifier is inspired by the architecture proposed in (Dora, Suresh & Sundararajan, 2015). The proposed architecture is a two layer fully connected feed-forward network as shown in Figure A.1. The input layer consists of m input neurons that convert real-valued inputs to spike-patterns (Figure A.2) and the output layer consist of spiking neurons (Figure A.3), The network consists of a decision block which monitors the output of the intermediate neurons to determine the predicted class for the presented sample. The intermediate neurons are modelled as IF neurons. These neurons can generate multiple spikes. The intermediate neurons process the input spike patterns from the input neurons. Each intermediate neuron is associated with a particular class and this association is stored in the decision block. The decision block identifies the intermediate neuron that fires first and returns its associated class label as the predicted class label. It should be noted here that although, an intermediate neuron can generate multiple spikes,the decision block uses only the first spike to determine the predicted class.

The proposed method is based on the evolving spiking neural network model for classification. It builds on the Thorpes model (S. Thorpe et al., 2001), in which early spikes are given more importance. Thorpe's model is highly influenced by the visual pattern recognition system. This method has fast supervised one pass learning. The eSNN have two layers:(1) an input layer and (2) an output layer. Initially, the output layer is empty. The output neurons are added to the output layer depending on the input samples during the training phase.

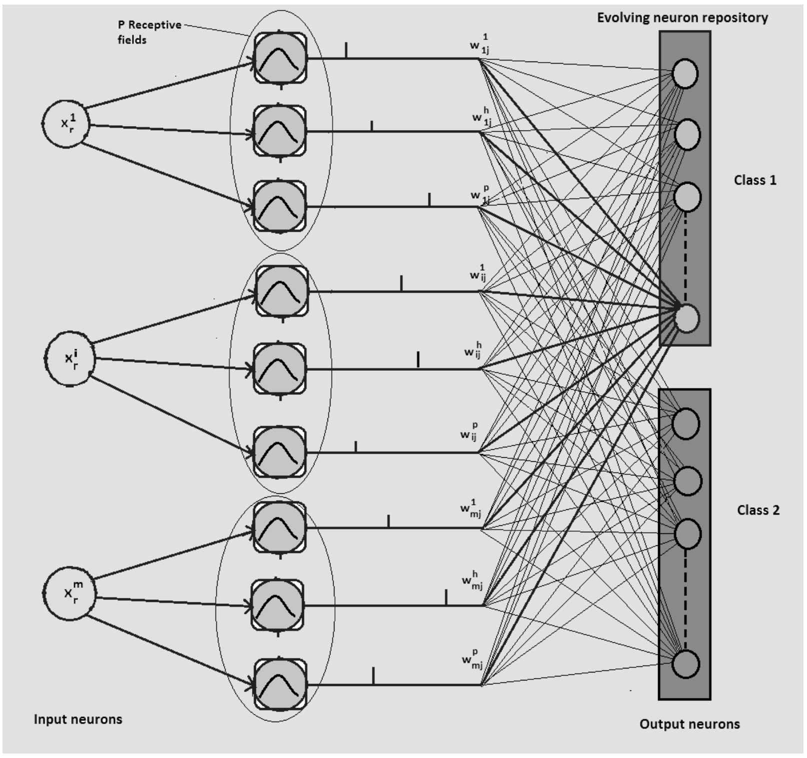


Figure A.1.: Proposed eSNN architecture for phishing website classification problem.

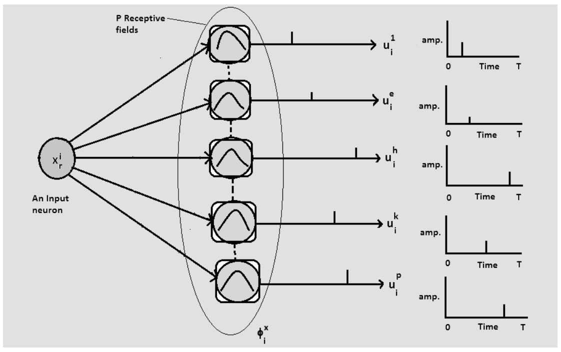


Figure A.2.: Input neuron model

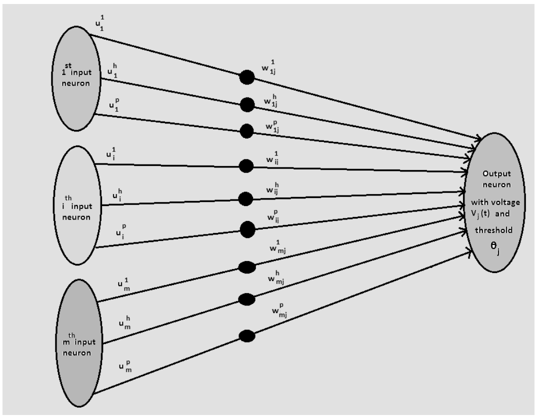


Figure A.3.: Intermediate neuron model

To deal with real-valued data sets, each data sample needs to map with the sequence of spikes using a precise neural encoding technique. Rank order population encoding is used for this purpose. The population encoding uses the Gaussian receptive fields (GRF) to encode the real-valued data. In this method, each input goes through a fixed number of GRF, and it generates a peak at the certain point of time. Following steps are performed during the classification process:

In rank order coding, every input is encoded individually using a set of P responders (*i.e.* receptive fields). For i^{th} input neuron with P receptive fields $(P > 2)$

whose input feature varies from I^i_{min} to I^i_{max}. The centre and width of the h^{th} receptive field is given by:

$$\mu^h_i = I^i_{min} + \frac{(2h-3)}{2}\frac{(I^i_{max} - I^i_{min})}{(P-2)} \tag{A.1}$$

and

$$\sigma^h_i = \frac{1}{\gamma}\frac{(I^i_{max} - I^i_{min})}{(P-2)} \tag{A.2}$$

where γ directly controls the width of the receptive field. The width controls the overlap between the two receptive fields. The τ^h_i is the firing time of the neuron calculated using:

$$\tau^h_i = \left\lfloor T(1-\phi^h_i) \right\rfloor \tag{A.3}$$

Where T is the simulation interval and ϕ^h_i is the GRF output defined as follows:

$$\phi^h_i = \exp(-\frac{(x^i_r - \mu^h_i)^2}{2(\sigma^h_i)^2}) \tag{A.4}$$

The output of the h^{th} responder of the i^{th} input neuron is given by:

$$u^h_i(t, x^i_r) = f^h_i(x^i_r)\delta^h_i(t - \tau^h_i) \tag{A.5}$$

Where $f^h_i(\cdot)$ is the spike amplitude function and $\delta^h_i(.)$ is the dirac delta function or firing time function.

The amplitude of the spikes

$$f^h_i(x^i_r) - \frac{\lambda^{r^h_i}}{1 + |x^i_r - \mu^h_i|} \tag{A.6}$$

Where r_i^h is the rank of the h^{th} responder of i^{th} neuron and λ is the slope of the amplitude function. The rank of the spikes is determined using the ranking function as follows:

$$F_R(x,y) = \begin{cases} 1, & \text{if } \phi_i^x \geq \phi_i^y \\ 0, & \text{otherwise} \end{cases} \tag{A.7}$$

Where x and y are the indices of any two receptive fields of the i^{th} neuron is given by:

$$r_i^h = 1 + \sum_{y=1, y \neq h}^{P} F_R(h,y) \tag{A.8}$$

The equations above describes the spike generation process using the population encoding framework and forms the first layer. The second layer also known as the output layer consists of another set of neurons and next, we describe the rules for the establishment of the synaptic connection between the input and output neuron and synaptic weight initialisation scheme. It is a sequential learning architecture that starts with no output neuron. The algorithm either chooses to add the new neuron at output layer or update the synaptic weight for training sample.

- Addition of output neuron: output neuron addition strategy to evolve the neuron if the current sample satisfies the following condition:

  $$\hat{c} = \{\phi\} \; OR \; (c \notin c_{overall}) \; OR \; (c \neq \hat{c} \; AND \; ||f - w_{nrs}|| > \beta_a) \tag{A.9}$$

  Where nrs is nearest output neuron of same class, β_a is distance threshold constant. f is the set of current sample spike-amplitude-response and w_{nrs} is the existing weight of synapse of the same class in the network. ϕ represents for the current sample none of the output neurons fired. coverall is the class label associated with the current output layer neurons and c

is the actual class label input training sample. The nearest output neuron is evaluated using the Euclidean distance between the current sample amplitude response and existing synaptic weight of all the output neurons of the same class. If the new sample satisfies the above condition then new output $(k+1)$ neuron is added its synaptic weight and threshold is given as:

$$w_{k+1} = f \tag{A.10}$$

$$\theta_{k+1} = \alpha w_{k+1}^T w_{k+1} \tag{A.11}$$

- In this strategy, the training sample associated class is not the same as the class associated with the fired output neuron. It means the different class label neuron fire first (nrl) and there exists another nearest output neuron of the same class (nrs). So the weight vector of different classes associated with output neurons is conflicting in nature. To detect this conflict problem following condition should be satisfied:

$$c \neq \hat{c} \; AND \; ||f - w_{nrs}|| < \beta_a \tag{A.12}$$

To resolve this conflict, the nearest neuron of the same class goes into long-term potentiation and the output neuron goes into different class goes into long-term depression. The synaptic weight update is as follows:

$$w_{nrs} = (1 - \eta_{nrs})w_{nrs} + \eta_{nrs} f \tag{A.13}$$

$$\theta_{nrs} = (1 - \eta_{nrs})\theta_{nrs} + \eta_{nrs}\alpha f^T f \tag{A.14}$$

Where η_{nrs} is self-adaptive learning factor. It is also called self-adaptive potentiation factor of output neuron. The factor η_{nrs} for output neuron given as:

$$\eta_{nrs} = \frac{\eta_{nrs}}{1 + \eta_{nrs}} \tag{A.15}$$

The output neuron of the other class goes to long-term depression. Due to the effect of this, the output neuron will not fire for the similar samples and the weights are updated as follows:

$$w_{nrl} = (1 + k)w_{nrl} - kf \tag{A.16}$$

Where k is the depression factor, which controls the synaptic weight depression factor. It is close to zero because the higher value of k results in a massive shift in the synaptic weight. Resulting information loss is stored in the network.

- Synaptic weight update approach: If the actual class label c is same as the class label associated with the fired output neuron then the synaptic weight of connection and threshold of the output neuron are updated as following:

$$w_{nrs} = (1 - \eta_{nrs})w_{nrs} + \eta_{nrs}f \tag{A.17}$$

$$\theta_{nrs} = (1 - \eta_{nrs})\theta_{nrs} + \eta_{nrs}\alpha f^T f \tag{A.18}$$

The voltage of the output neuron at any time calculated as follows:

$$v_j(t_1) = \sum_{t=0}^{t+1} \sum_{h=1,i=1}^{P} u_i^h(t, x_r^i)w_{ij}^h \tag{A.19}$$

A.1 Hyperparameter Selection of eSNN Classifier

There are a number of hyperparameters that needs to be set for the algorithm described above. The first parameter is the number of receptive fields P. During the experiment, we observed that the number of receptive field increases as the firing time decreases. The number of receptive fields also depends on the dataset. P also controls the amplitude of the input neuron. For our dataset, the number of the receptive field is set to 8. The second parameter is the overlap factor γ, which controls the overlap between receptive fields and regulates the width of the receptive field. γ the width of the receptive field and thus impact only the firing time function. In the experiment, the γ value is set to 3, which means 30% overlap exists between two subsequent receptive fields. It controls the range of the of receptive field and useful for temporal coding. The third factor is λ. It represent the slope of the amplitude function $f(\cdot)$. There are other parameters as well. The parameter α is the threshold fraction which controls the firing time of the output neuron and is set between 0.5 and 0.9. γ is the self-adaptive potential factor. Its value is initialised to 0.5 and decayed to 0. β is a constant that controls the addition of the neuron at the output layer. The range of the parameter β depends on the feature count and has an inverse relationship with values ranging between 0.5 and 0.8. The depression factor k controls if the neuron fired wrongly during training, the corresponding weights goes into long-term depression. The value of k is typically set to a very small value between 0.01 to 0.35. Large value of k leads to information loss in the network.

A.2 Testing Method of eSNN Classifier

At the end of the training phase, the knowledge is stored in the network. For testing, first, need to convert the real-valued input data into the spikes. Each output neuron associated with certain threshold value are chosen during the training. If the incoming potential (signal) summation crosses the specific threshold value of the particular neuron, then the neuron is fired, and the corresponding class label is predicted. It is often the case that many output neurons fire for the given input. To resolve this issue, k-nearest neighbour is applied to predict the class label. To get the optimal accuracy we check the different values of k to predict the class label, the value of k are 3, 5, 7, 9, 11, 13, 15 and so on. The k-nearest neighbour is performed using the Euclidean distance between the amplitude of spikes of the current input testing sample and the synaptic weight corresponding to the fired neuron which is stored by the network at the time of training *i.e.* $||f - w||$ with weights corresponding to each fired output neuron.

B

eSNN Architectures for Stock Price Movement Forecast

In the last few years, there have been a large number of studies performed on predicting the stock market trend. Both researchers and practitioners have used numerous approaches to predict the stock market trend. Due to the chaotic nature and the complexity of the stock market indices, researchers are still struggling to design techniques that can accurately model the behaviour of their trends. In Table B.1, we have given an overview of the different prediction techniques and stock market indices used in the literature to predict the stock price direction by taking technical indicators as input variables.

Table B.1 presents a concise literature review on machine learning techniques applied on stock price movement prediction. It is clear that many traditional machine learning techniques have been explored to predict stock price direction and also most of the algorithms have been applied to a single stock index to measure the performance of the model. In the third generation of neural networks the spiking neural networks (SNN) now offers a new perspective to explore for the solution of the problem. SNN uses spike information representation and spike-time learning rules to capture temporal associations between a large number of temporal variables in streaming data and to predict future events. One of the successful SNN models is the eSNN model, where the number of spiking neurons evolve incrementally in time to capture temporal prototypes from data.

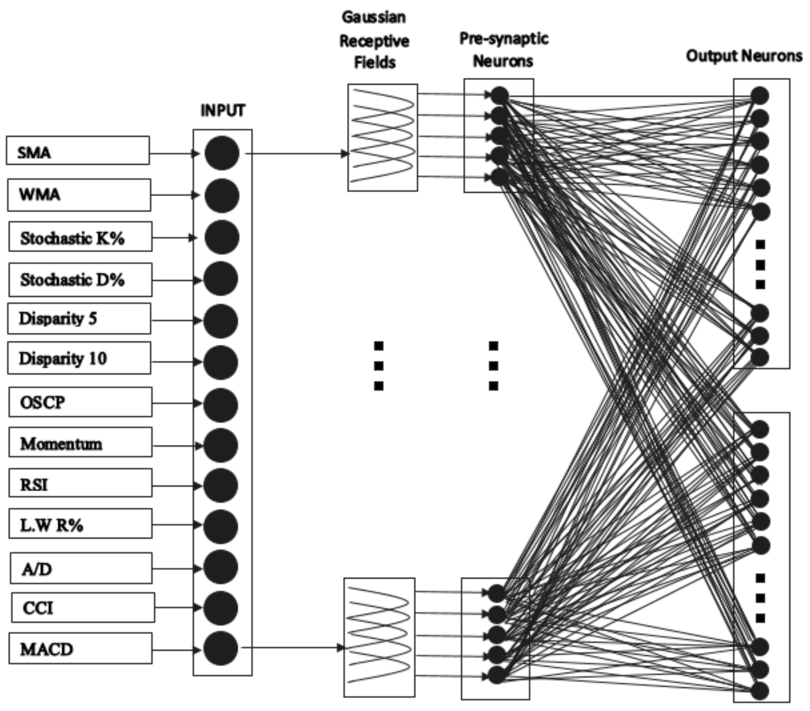


Figure B.1.: Architecture of the proposed technical stock indicator SI-eSNN model for stock price direction prediction

B.1 The SI-eSNN Model for Stock Trend Prediction Based on Stock Indicators

B.1.1 Overall Architecture

The architecture of the SI-eSNN model is presented in Figure B.1. The first layer is the set of inputs to the model, each of them representing a technical stock indicator. The research so far has demonstrated that using technical indicators can lead to better results than using real stock values as time series and also that there is a lot of research done on selecting the most appropriate technical indicators. In the model presented in Figure B.1 the input technical indicators have been selected from (J. Patel et al., 2015; K.-j. Kim, 2003; Kara et al., 2011) and explained in Table B.2, but these indicators can vary across stock prediction applications.

[0] C_t is the closing price, L_t is the low price and H_t is the high price at time t, $DIFF_t = EMA(12)_t - EMA(26)_t$, EMA is the moving average, $EMA(k)_t = EMA(k)_{t-1} + \alpha \times (C_t - EMA(k)_{t-1})$, α is the smoothing factor which is equal to $\frac{2}{k+1}$, k is the time period of

.

Layer 2 is the encoding layer, where the real value of each input variable (technical indicator) is encoded as trains of spikes generated by several encoding spiking neurons (or also, pre-synaptic neurons), each of them having a receptive field. The receptive fields of neighbouring neurons are overlapping as Gaussian or Logistic functions and all of them covering the whole range of the values of this variable. The number of these encoding neurons (receptive fields) can vary, and this is a user-defined parameter that is optimised for a better performance of the model.

Layer 3 is the output evolving layer, which evolves output spiking neurons that represent clusters (prototypes) of input vectors that belong to the same class, in this case class UP and class DOWN. Each output neuron is connected to all the input neurons, and the connection weights are subject to learning from data.

The architecture of the SI-eSNN model for stock price direction prediction allows for incremental learning. It is adaptive to new data when it becomes available. Hence, it can learn new samples without retraining the model on old data. The details of the functioning of the SI-eSNN model is presented below.

B.1.2 Neural Encoding

To learn real-valued data, each instance or sample (input vector) is encoded in the form of spikes over time using a neural encoding technique such as rank order population coding (S. J. Thorpe & Gautrais, 1997; Bohte, Kok & La Poutre, 2002). In our study, we have used rank order population encoding as per (Schliebs, Defoin-Platel, Worner & Kasabov, 2009). Population encoding maps the input value into a series of spikes over time using an array of Gaussian receptive fields that describe pre-synaptic neurons. The center (C_j) and width

k-day exponential moving average, LL_t and HH_t implies lowest low and highest high in the last t days respectively. $M_t = \frac{H_t+L_t+C_t}{3}$, $SM_t = \frac{\sum_{i=1}^{n} M_{t-i+1}}{n}$, $D_t = \frac{\sum_{i=1}^{n} |M_{t-i+1} - SM_t|}{n}$, UP_t means upward price change while DW_t is the downward price change at time t

(W_j) of each of the Gaussian or Logistic receptive field of pre-synaptic neurons j are defined as:

$$C_j = I^n_{min} + \frac{2j-3}{2} \times \frac{I^n_{max} - I^n_{min}}{N-2} \tag{B.1}$$

and

$$W_j = \frac{1}{\beta} \times \frac{I^n_{max} - I^n_{min}}{N-2}, 1 \leq \beta \leq 2 \tag{B.2}$$

Where N is the number of receptive fields; n is the range of input variable $n = [I^n_{min}, I^n_{max}]$, the parameter β defines the width of each receptive field. Output of each of the pre-synaptic neuron j using Gaussian receptive field is defined as:

$$output_j = \exp(-\frac{(x - C_j)^2}{2 \cdot W_j^2}) \tag{B.3}$$

Output of each of the pre-synaptic neuron j using Logistic receptive field is defined as:

$$output_j = \frac{\exp(\frac{-(x-C_j)}{W_j})}{(1 + \exp(\frac{-(x-C_j)}{W_j}))^2} \tag{B.4}$$

The firing time of each of the pre-synaptic neurons is defined as:

$$\tau_j = \lfloor T(1 - output_j) \rfloor \tag{B.5}$$

where: T is the simulation or spike time interval.

When a real value input is presented to the N pre-synaptic spiking neurons, the first spike is generated by this neuron to which receptive field the input value belongs to the highest degree, etc.

B.1.3 Neural Model

For the context of SI-eSNN, Thorpes neuron model (S. J. Thorpe & Gautrais, 1997) has been used since it is simple and effective. The Thorpes model is based on the timing of each spike, that is, earlier spike defines stronger weight as compared to the later spike. Each neuron in this model can spike at most once. A neuron in this model fires when its post-synaptic potential (PSP) reaches the threshold value. The PSP of neuron i is defined as:

$$PSP_i = \begin{cases} 0, & \text{if } fired \\ \sum w_{ji} * mod^{order(j)}, & \text{otherwise} \end{cases} \tag{B.6}$$

Where w_{ji} represents the weight of the synaptic connection between pre-synaptic neuron j to the output neuron i; mod represents modulation factor with a range in between 0 to 1; $order(j)$ defines the rank of pre-synaptic neurons spike. The first rank will be assigned as 0 and subsequently, rank will be increased by 1 based on firing time of each pre-synaptic neurons.

B.1.4 Learning in the Output Neurons

The eSNN algorithm was first introduced in (Wysoski, Benuskova & Kasabov, 2006; N. Kasabov, 2007). The learning techniques used by the eSNN model is one-pass learning, that is, the model requires one-time presentation of a sample in a feed-forward manner. It will create an output neuron for each input sample.

The weight vector and a threshold value for each of the output neuron generated towards the training pattern are learned and stored in the repository. However, if this weight vector is similar to the weight vector of the already trained neuron in the repository with some similarity threshold, then it will merge with the most similar one. Merging here means updating the weights and the threshold value of the merged neurons. The weight vector and threshold value of the merged neurons update their values by taking the average value of new output neuron weight vector and merged neuron weight vector and the average value of new output neuron threshold and merged neuron threshold respectively.

B.1.5 Algorithm for eSNN Training

Algorithm B.1 eSNN training algorithm

```
1: Initialize neuron repository, $R = \{\}$
2: Set eSNN parameter $mod = [0,1], C = [0,1], sim = [0,1]$
3: for $\forall$ input pattern $i$ that belongs to the same class do
4:    Encode input pattern into firing time of multiple pre-synaptic neurons $j$
5:    Create a new output neuron $i$ for this class and calculate the connection
      weights as $w_{ji} = mod^{order(j)}$
6:    Calculate $PSP_{max(i)} = \sum_j w_{ji} \times mod^{order(j)}$
7:    Get $PSP$ threshold value $\gamma_i = PSP_{max(i)} \times C$
8:    if The new neuron weight vector $\leq sim$ of trained output neuron weight
      vector in $R$ then
9:       Update the weight vector and threshold of the most similar neuron in
         the same output class group
10:      $w = \frac{w_{new} + w * N}{N+1}$
11:      $\gamma = \frac{\gamma_{new} + \gamma * N}{N+1}$
12:      where $N$ is the number of previous merges of the most similar neuron
13:   else
14:      Add the weight vector and threshold of the new neuron to the neuron
         repository $R$
15:   end if
16: end for
17: Repeat above for all input patterns of other output classes
```

The eSNN algorithm creates a repository of output neurons for the training patterns. For each training pattern that belongs to the same class, a new output neuron is created and connected to all the pre-synaptic neurons in the previous layer. The weight for each of the connection from pre-synaptic neuron j to output neuron i is denoted as w_{ji}. The value of w_{ji} is calculated based on the spike order through a synapse j, which is given in line 5 of Algorithm B.1 as $w_{ji} = mod^{order(j)}, \forall j$ where j is the pre-synaptic neuron of output neuron i.

The threshold γ_i of newly created output neuron would be defined as the fraction $C \in \mathbb{R}, 0 < C < 1$, of the maximum post-synaptic potential $PSP_{max(i)}$: $\gamma_i = PSP_{max(i)} \times C$

The weight vector of newly created output neuron is then compared with the already trained output neurons in the repository. If the newly created output neuron weight vector is less than sim of trained neuron weight vectors, then the threshold and weight vector of newly created output neuron is merged with most similar neuron according to $w = \dfrac{w_{new} + w * N}{N + 1}$ and $\gamma = \dfrac{\gamma_{new} + \gamma * N}{N + 1}$.

Here, N is the number of previous merges of the most similar neurons. After the merge operation, the newly created output neuron weight vector is discarded, and the new pattern is presented to model. If none of the trained neurons in the repository is similar, the new output neuron is added to the repository.

B.1.6 Testing (Recall of the Model on New Data)

The testing process is done by propagating the spikes that encode the test vector (sample) to all trained output neurons. The class label for the testing sample will be defined based on the output class label of the output neuron which fires first.

B.2 The CUDA-SI-eSNN Model: A Parallel eSNN Model for GPU Machines

The training and recall procedures for the SI-eSNN model consists of two operations. In the first stage, encoding of the input patterns, calculations of the

Algorithm B.2 CUDA-SI-eSNN algorithm

```
1: Initialize input patterns array in device memory with size is equal to $N \times M$,
   Where $N$ is number of input patterns and $M$ is number of input variable.
2: Declare weights matrix in device memory with size is equal to $N \times P$ Where
   $P$ is number of pre-synaptic neurons which is equal to the number of input
   variable multiplied by number of Gaussian receptive fields.
3: Transfer the input patterns data into the GPU memory.
4: Encode input patterns into firing time of multiple pre-synaptic neurons $j$. In
   this step, each of the threads in the GPU device will calculate the firing-time
   of each of the pre-synaptic neurons independently. The number of threads
   is equal to $N \times P$.
5: Calculate the connection weights as $w_{ji} = mod^{order(j)}$. In this step, each
   tread in the GPU device will calculate the weight independently.
6: Transfer the resulted weight matrix into CPU memory.
7: Calculation of maximum post-synaptic potential $PSP_{max(i)}$, threshold value
   $\gamma_i$ and the merging operation are similar to the equivalent operations of
   eSNN, which are performed on CPU sequentially.
```

weights and the thresholds are performed, which is from step-4 to step-7 of the eSNN algorithm. In the second stage, merging of the output neurons is performed, which is from step-8 to step-14 of the eSNN algorithm. Since the encoding and learning of weights of one input pattern is independent of others except for the merging operation, it is possible to perform both encoding and weight learning operations parallelly in GPU device using multiple threads. The procedure for the GPU-based eSNN algorithm is described in Algorithm B.2.

B.3 Sliding Window (SW)-eSNN for Incremental Learning and Stock Movement Prediction

While learning in the SI-eSNN and CUDA-SI-eSNN models were vector by vector and each vector was learned separately from the others even though the vectors of consecutive days would have some temporal associations. Here we present an SW-eSNN model where a window based block of the time series of technical indicator data is used for training a model and a future section to test it. Then

Algorithm B.3 SW-eSNN learning algorithm

```
1: Train an eSNN model on the whole existing historical data of a stock till time
   $t$ (as per Algorithm B.1)
2: Recall the model to predict the next month (t+1) stock movement.
3: When the next month results are known, train the model incrementally on
   this data.
4: Aggregate the output neurons if necessary using the aggregation operator
   and the $sim$ parameter.
5: Evaluate the classification error and the AUC so far.
6: Optimize parameters to improve future time accuracy.
```

the window moves in time, to select the next section for training and testing, etc.

Table B.1.: Prediction Techniques and Stock Market Indices for Stock Price Direction Prediction used in Literature

	Techniques	Stock Market Indices	GPU-Based
(Chiou, Liu & Tsaih, 1996)	RNN	TSI	No
(Tsaih, Hsu & Lai, 1998)	RBS + RNN	S & P 500	No
(Saad, Prokhorov & Wunsch, 1998)	TDN, RN, PNN	AAPL, IBM, MOT, MSFT	No
(K.-j. Kim & Han, 2000)	GA + ANN	KOSPI	No
(K.-j. Kim, 2003)	SVM	KOSPI	No
(K.-j. Kim & Lee, 2004)	FS + ANN	KOSPI	No
(W. Huang, Nakamori & Wang, 2005)	SVM, LDA, QDA, EBNN	Nikkie-225	No
(Kumar & Thenmozhi, 2006)	RF , SVM	NIFTY, CNX, S & P 500	No
(C.-J. Huang, Yang & Chuang, 2008)	Wrapper, LR, SVM, C 4.5 DT, BP, KNN, Voting	KOSPI, TSE	No
(P. Ou & Wang, 2009)	LDA, QDA, KNN, NB, SVM, LS-SVM, ANN, TBC, BC, Logit	HSI	No
(Yudong Zhang & Wu, 2009)	IBCO + BPNN	S & P 500	No
(Mostafa, 2010)	MLP, GRNN	KSE	No
(Nair, Mohandas & Sakthivel, 2010)	DT + RS	BSE	No
(Kara, Boyacioglu & Baykan, 2011)	SVM, ANN	ISE	No
(Chang et al., 2012)	EPCNNs	S & P 500	No
(Y. Lin, Guo & Hu, 2013)	SVM	TW50	No
(de Oliveira, Nobre & Zárate, 2013)	ANN	PETR4	No
(Wang, 2014)	PCA + SVM	KOSPI, HSI	No
(Bisoi & Dash, 2014)	ANN	BSE, IBM, RIL, OSI	No
(J. Patel, Shah, Thakkar & Kotecha, 2015)	ANN, SVM, NB, RF	NIFTY, BSE-Sensex, Infosys, Reliance	No
(Hafezi, Shahrabi & Hadavandi, 2015)	BNNMAS	DAX	No
(Ballings, Van den Poel, Hespeels & Gryp, 2015)	LR, ANN, KNN, SVM, AB, RF, KF	European Stock Index	No
(X.-d. Zhang, Li & Pan, 2016)	AB + PSVM +GA	SZSE, NASDAQ	No
Our study	eSNN, CUDA-eSNN	BSE, NIFTY-50, NASDAQ, Nikkie-225, DAX, NYSE-Amex,S & P-500, SSE	Yes

Table B.2.: Selected technical indicators and their formulas

Name of the technical indicators (SI)	Formulas [1]
Simple 10-day moving average	$\frac{C_t + C_{t-1} + \cdots + C_{t-9}}{10}$
Weighted 10-day moving average	$\frac{(n) \times C_t + (n-1) \times C_{t-1} + \cdots + C_{t-9}}{(n + (n-1) + \cdots + 1)}$
Stochastic K%	$\frac{C_t - LL_{t-(n-1)}}{HH_{t-(n-1)} - LL_{t-(n-1)}} \times 100$
Stochastic D%	$\frac{\sum_{t=0}^{n-1} K_{t-i}}{10}\%$
Disparity 5	$\frac{C_t}{MA_5} \times 100$
Disparity 10	$\frac{C_t}{MA_{10}} \times 100$
OSCP	$\frac{MA_5 - MA_{10}}{MA_{10}}$
Momentum	$C_t - C_{t-9}$
RSI (Relative Strength Index)	$100 - \frac{100}{1 + \frac{\sum_{i=0}^{n-1} UP_{t-i}/n}{\sum_{i=0}^{n-1} DW_{t-i}/n}}$
Larry William R%	$\frac{H_n - C_t}{H_n - L_n} \times 100$
A/D (Accumulation/Distribution)	$\frac{H_t - C_{t-1}}{H_t - L_t}$
CCI (Commodity Channel Index)	$\frac{M_t - SM_t}{0.015 \times D_t}$
MACD (moving average convergence divergence)	$MACD(n)_{t-1} + \frac{2}{n+1} \times (DIFF_t - MACD(n)_{t-1})$

Table B.3.: The number of instances of UP and DOWN categories in Stock Market Indices

Datasets (No. of instances)	Years Covered	Training (70%)			Testing (30%)		
		UP	Down	Total	UP	DOWN	Total
BSE	Jan 2005- Dec 2015	1020	892	1912	437	382	819
Nikkie-225	Jan 1987 - Jul 2016	2609	2483	5092	1115	1067	2182
NASDAQ	Jan 2005 - Dec 2015	1058	865	1923	446	378	824
NIFTY-50	Jan 2008 - Dec 2015	709	644	1353	300	279	579
S&P-500	Jan 1962 - July 2016	5118	4473	9591	2146	1964	4110
Sanghai Stock Exchange	Jan 1998 - July 2016	1679	1470	3149	685	665	1350
DJUS	Apr 2005 - July 2016	997	827	1824	438	344	782
DAX Index	Jan 1991 - July 2016	2413	2114	4527	1044	897	1941
NYSE-Amex	Jan 1996 to July 2016	1957	1668	3625	850	704	1554

Table B.4.: GPU Specifications

GeForce GT 730 Specifications	
Version	GT 730 DDR3, 64-bit
CUDA Cores	384
Base Clock (MHz)	902
Memory Clock	1.8 Gbps
Standard Memory Config	2048 MB
Memory Interface	DDR3
Memory Interface Width	64-bit
Memory Bandwidth (GB/sec)	14.4

C Miscellaneous Code Snippets

C.1 Matlab Code for the SNNc Unsupervised Learning Using Forward Adjacency Lists for Connection C and Weight W

```
function [O] =
    neucube_unsupervised_learning_forward_adjacency_list( W,C,D,
    indices_of_input_neuron, V_THR, ETA_THR, KAPPA)
   N=128;
   Q=1485;
   T=size(D,1);
   O= int16(zeros(T,Q));
   for t=1:T
      if(mod(t,N)==1)
         firing_neuron=int16(zeros(1,Q));
         last_spike_time=uint16(zeros(1,Q));
         eta=int16(zeros(1,Q));
         v=single(zeros(1,Q));
      end
      last_spike_time(firing_neuron==1)=t;
      firing_neuron_indices=find(firing_neuron==1);
      for i=1:size(firing_neuron_indices,2)
         index_i=firing_neuron_indices(i);
         post_synaptic_neuron_indices=C{index_i};
         if(~isempty(post_synaptic_neuron_indices))
            for j=1:size(post_synaptic_neuron_indices,2)
               index_j=post_synaptic_neuron_indices(j);
               if(eta(index_j)==0)
                  v(index_j)=v(index_j)+W{index_i}(1,j);
               end
            end
         end
         firing_neuron(index_i)=0;
      end
      a=find(v>=V_THR);
      not_a=find(v<V_THR);
      v(a)=0;
      eta(a)=ETA_THR;
      firing_neuron(a)=1;
      eta(not_a)=max(0,eta(not_a)-1);
      v(not_a)=max(0,v(not_a)-0.002);
      firing_neuron_indices=find(firing_neuron==1);
      if(~isempty(firing_neuron_indices))
         for i=1:size(firing_neuron_indices,2)
            index_firing=firing_neuron_indices(i);
            for k=1:Q
               if(find(C{index_firing}==k))
                  index_j=find(C{index_firing}==k);
                  deltaw=KAPPA/single(t-last_spike_time(k)+1);
                  W{index_firing}(1,index_j)=W{index_firing}(1,
   index_j)-deltaw;
               end
               if(find([C{k}]==index_firing))
                  index_k=find([C{k}]==index_firing);
                  deltaw=KAPPA/single(t-last_spike_time(k)+1);
                  W{k}(1,index_k)= W{k}(1,index_k)+deltaw;
               end
         end
      end
```

```
    end
    firing_neuron(indices_of_input_neuron)=D(t,1:14);
    firing_neuron(end-14+1:end)=D(t,14+1:end);
    O(t,:)=firing_neuron;
    end
end
```

code snippet C.1: Matlab code for SNNc unsupervised learning algorithm using forward adjacency list data structure for storing connection C and weight W

C.2 Matlab Code for the SNNc Unsupervised Learning Using Forward-backward Adjacency Lists for Connection C and Weight W

```
function [O] =
    neucube_unsupervised_forward_backward_adjacency_list( W,C,D,
    indices_of_input_neuron, V_THR, ETA_THR, KAPPA )
    N=128;
    Q=1485;
    T=size(D,1);
    O= int16(zeros(T,Q));
    for t=1:T
       if(mod(t,N)==1)
          firing_neuron=int16(zeros(1,Q));
          last_spike_time=uint16(zeros(1,Q));
          eta=int16(zeros(1,Q));
          v=single(zeros(1,Q));
       end
       last_spike_time(firing_neuron==1)=t;
       firing_neuron_indices=find(firing_neuron==1);
       for i=1:size(firing_neuron_indices,2)
```

```
         index_i=firing_neuron_indices(i);
         post_synaptic_neuron_indices=find(C(index_i,:));
         if(~isempty(post_synaptic_neuron_indices))
            for j=1:size(post_synaptic_neuron_indices,2)
               index_j=post_synaptic_neuron_indices(j);
               if(eta(index_j)==0)
                  v(index_j)=v(index_j)+W(index_i,index_j);
               end
            end
         end
         firing_neuron(index_i)=0;
      end
      a=find(v>=V_THR);
      not_a=find(v<V_THR);
      v(a)=0;
      eta(a)=ETA_THR;
      firing_neuron(a)=1;
      eta(not_a)=max(0,eta(not_a)-1);
      v(not_a)=max(0,v(not_a)-0.002);
      firing_neuron_indices=find(firing_neuron==1);
      if(~isempty(firing_neuron_indices))
         for i=1:size(firing_neuron_indices,2)
            index_firing=firing_neuron_indices(i);
            post_neuron_indices=find(C(index_firing,:)==1);
            for k=1:size(post_neuron_indices,2)
               index_j=post_neuron_indices(1,k);
               deltaw=KAPPA/single(t-last_spike_time(index_j)+1);
               W(index_firing,index_j)=W(index_firing,index_j)-deltaw;
            end
            pre_neuron_indices=find(C(:,index_firing)==1);
            for k=1:size(pre_neuron_indices,1)
```

```
                index_j=pre_neuron_indices(k,1);
                deltaw=KAPPA/single(t-last_spike_time(index_j)
    +1);
                W(index_j,index_firing)=W(index_j,index_firing)+
    deltaw;
            end
        end
    end
    firing_neuron(indices_of_input_neuron)=D(t,1:14);
    firing_neuron(end-14+1:end)=D(t,14+1:end);
    O(t,:)=firing_neuron;
  end
end
```

code snippet C.2: Matlab code for SNNc unsupervised learning algorithm using forward-backward adjacency list data structure for storing connection C and weight W

C.3 Python Code for the Modular Implementation of the SNNc

```
class SNNc:
    def __init__(self, spiking_neuron_coordinates,
    input_neuron_coordinates, SNNc_max_distance=None,
    connection_algorithm="swc", radial_distance_coverage=0.1):
        assert isinstance(connection_algorithm, str)
        assert isinstance(radial_distance_coverage, float)
        assert isinstance(spiking_neuron_coordinates, np.ndarray)
        assert isinstance(input_neuron_coordinates, np.ndarray)

        if connection_algorithm not in "swc":
```

```python
            raise ValueError('see doc string for possible values of connection algorithm parameter!!')
        if not 0 < radial_distance_coverage <= 1:
            raise ValueError('The radial distance coverage parameter should be between (0,1]!!')
        self.spiking_neuron_coordinates = spiking_neuron_coordinates
        self.input_neuron_coordinates = input_neuron_coordinates
        self.SNNc_coordinates = np.vstack((input_neuron_coordinates, spiking_neuron_coordinates))
        self.spiking_neuron_count = len(self.spiking_neuron_coordinates)
        self.input_neuron_count = len(self.input_neuron_coordinates)
        self.SNNc_neuron_count = self.input_neuron_count + self.spiking_neuron_count

        self.input_coordinate_indices = np.arange(self.input_neuron_count)
        self.spiking_coordinate_indices = np.arange(self.input_neuron_count, self.SNNc_neuron_count)
        """
        Calculate maximum distance between the SNNc neurons if SNNc_max_distance=None. Uses a brute force approach due
        to the irregularity in the shape. Consider updating it to a non bruteforce approach later.
        """
        if SNNc_max_distance is None:
            self.SNNc_MAX_DISTANCE = 0
            for ind1, neuron1 in enumerate(self.SNNc_coordinates):
                for ind2, neuron2 in enumerate(self.SNNc_coordinates):
                    distance = np.linalg.norm(neuron1 - neuron2)
```

```
               if distance > self.SNNc_MAX_DISTANCE:
                  self.SNNc_MAX_DISTANCE = distance
               else:
               self.SNNc_MAX_DISTANCE = SNNc_max_distance

      self.network = None
         self.create_network(connection_algorithm,
   radial_distance_coverage)

   def create_network(self, connection_algorithm,
   radial_distance_coverage):
      self.network = nx.DiGraph()
      nodes_list = []
      for neuron in np.arange(self.SNNc_neuron_count):
      if neuron in self.input_coordinate_indices:
         attributes = dict(type="i", location=self.
   SNNc_coordinates[neuron, :])
         attributes["neuron"] = None
         nodes_list.append((neuron, attributes))
      else:
         attributes = dict(type="s", location=self.
   SNNc_coordinates[neuron, :])
         nodes_list.append((neuron, attributes))
      self.network.add_nodes_from(nodes=nodes_list)
      if connection_algorithm is "swc":
         self.swc(radial_distance_coverage=
   radial_distance_coverage)
         self.network = nx.freeze(self.network)

   def swc(self, radial_distance_coverage):
      abs_radial_distance_coverage = radial_distance_coverage *
    self.SNNc_MAX_DISTANCE
      edge_list = []
```

```python
        points = self.SNNc_coordinates
        points_tree = scipy.spatial.cKDTree(points)
        input_neuron_indices = self.input_coordinate_indices
        for query_index in np.arange(self.SNNc_neuron_count):
            neighbour_indices = points_tree.query_ball_point(points[query_index], abs_radial_distance_coverage)
            neighbour_indices_without_input = [index for index in neighbour_indices if index not in input_neuron_indices]
            for index in neighbour_indices_without_input:
                edge_list.append((query_index, index))
                if (index, query_index) not in edge_list:
                    self.network.add_edge(query_index, index, w=0.05)

    def create_spiking_neurons(self, neuron_type='lif', **parameters):
        """

        Parameters
        ----------
        neuron_type:str
        current options: 'LIF'(Leaky integrate and fire).
        potential_threshold: float
        'lif' parameter(optional, default=1.0). The firing threshold potential
        refractory_threshold: int
        'lif' parameter(optional, default=1.0). The number of discrete times
        the neuron stays quiet after it fires a spike
        leak_rate: float
        'lif' parameter(optional, default=0.000). The absolute leak of
        potential when the neuron is refractory
        potential_resting: float
        ’lif’ parameter(optional, default=0.0). The resting potential of the neuron.
        spike_history_length:int
        ’lif’ parameter(optional, default=10. The historical spike information that
        the neuron remembers and calculates the states upon


        Returns
        -------
        None

        """
        if neuron_type not in ("lif"):
            raise ValueError("See docstring for possible values of neuron_type!!")
        if neuron_type is "lif":
        for neuron_index in self.spiking_coordinate_indices:
            predecessor_count = len(self.network.predecessors(neuron_index))
            self.network.node[neuron_index]["neuron"] = LeakyIntegrateAndFireNeuron(
            predecessor_count=predecessor_count, **parameters)

    def learn_stdp(self, sample, learning_rate):
        """

        Parameters
        ----------
        sample: numpy.ndarray
        input samples consisting of spike data(time x feature)
        input_learns: boolean(optional, default:True)
        Input connections weights are not changed during learning
     if set to False
        learning_rate: float(optional, default:0.5)
        The learning rate of the learning algorithm

        Returns
        --------
        numpy.ndarray
        The array of spikes generated by the SNNc for the given
    input sample. The dimension of the array is
        time x SNNc neuron count


        """

        assert isinstance(sample, np.ndarray)

        input_data = sample
        (simulation_time, _) = input_data.shape
        spike_data = np.zeros(shape=(simulation_time, self.
    SNNc_neuron_count), dtype=int)
        spike_data[:, self.input_coordinate_indices] = input_data

        for time in np.arange(simulation_time):
            connections_of_neuron_receiving_spike = []
            connections_of_neuron_emitting_spike = []
            """
            block for network simulation using spike information
    present in spike data
            """
            for neuron_id in self.spiking_coordinate_indices:
                predecessor_indices = self.network.predecessors(
    neuron_id)
```

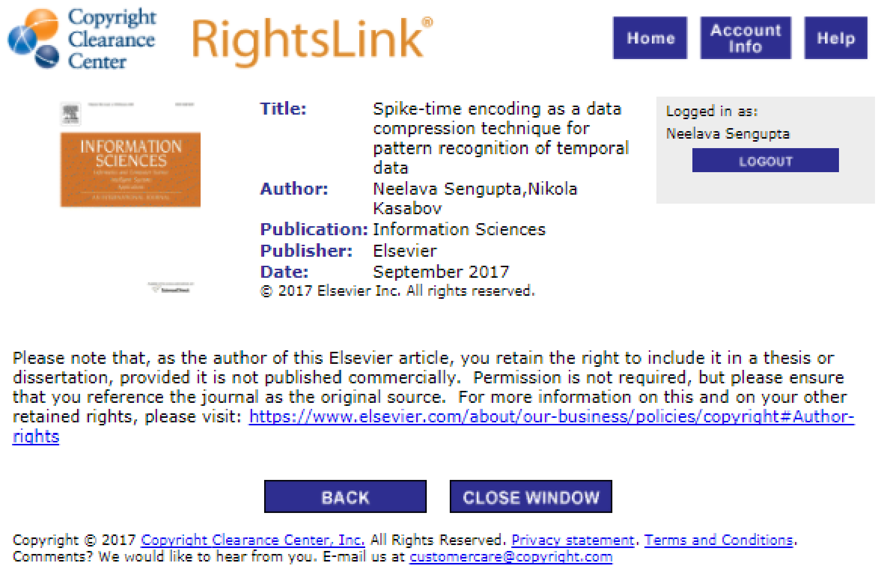
```
        predecessor_spike_data = spike_data[time, predecessor_indices]

        """
        Block to compute values to be used during learning
        """
        s_ind = np.nonzero(spike_data[time, predecessor_indices])[0]
        predecessor_spike_indices = []
        if len(s_ind) != 0:
            predecessor_spike_indices = list(np.array(predecessor_indices)[s_ind])
        for identifier in predecessor_spike_indices:
            connections_of_neuron_receiving_spike.append((identifier, neuron_id))
        predecessor_weight_data = []
        for k in self.network.predecessors(neuron_id):
            predecessor_weight_data.append(self.network[k][neuron_id]["w"])
        spk = self.network.node[neuron_id]["neuron"].simulate(spike_train=predecessor_spike_dat,
                                                             weight=predecessor_weight_data)
        """
        Block to compute values to be used during learning
        """
        if spk == 1:
            predecessor_indices = self.network.predecessors(neuron_id)
            for identifier in predecessor_indices:
                connections_of_neuron_emitting_spike.append((identifier, neuron_id))

                if time < simulation_time - 1:
                    spike_data[time + 1, neuron_id] = spk
        """
        block for learning
        """
        for connection in
connections_of_neuron_receiving_spike:
            pre_spike = spike_data[0:time + 1, connection[0]]
            post_spike = spike_data[0:time + 1, connection[1]]
            deltaw = self.stdp(pre_synaptic_spike_history=np.
abs(pre_spike).tolist(),
                               post_synaptic_spike_history=np.abs(
post_spike).tolist(),
                               max_weight_update=learning_rate)
            self.network[connection[0]][connection[1]]['w'] +=
deltaw
        for connection in connections_of_neuron_emitting_spike
:
            pre_spike = spike_data[0:time + 2, connection[0]]
            post_spike = spike_data[0:time + 2, connection[1]]
            deltaw = self.stdp(pre_synaptic_spike_history=np.
abs(pre_spike).tolist(),
                               post_synaptic_spike_history=np.abs(
post_spike).tolist(),
                               max_weight_update=learning_rate)
            self.network[connection[0]][connection[1]]['w'] +=
deltaw

    return spike_data

@staticmethod
def stdp(pre_synaptic_spike_history,
post_synaptic_spike_history, max_weight_update):
    """

    Parameters
    ----------
    pre_synaptic_spike_history: list
    It is the binary vector of size n specifying the presynaptic spike history of the neuron.
    The last element signifies a spike/no spike at current timepoint t, second last element
    signifies a spike/no spike at t-1 and so on.

    post_synaptic_spike_history: list
    It is a binary vector of size n specifying the post synaptic spike history.
    The last element signifies a spike/no spike at current timepoint t, second
    last element signifies a spike/no spike at t-1 and so on.

    max_weight_update: float
    hyperparameter to control the upper bound of deltaw.


    Returns
    -------
    float

    """

    assert isinstance(pre_synaptic_spike_history, list)
    assert isinstance(post_synaptic_spike_history, list)
    if not 0 < max_weight_update <= 1:
        raise ValueError("See docstring for value range of learning_rate")
```

```python
    if len(pre_synaptic_spike_history) != len(post_synaptic_spike_history):
        raise ValueError("Length mismatch of pre and post synaptic spike history!!")
    pre_synaptic_spike_energy = 0
    post_synaptic_spike_energy = 0
    importance_of_LTP = 0.3
    importance_of_LTD = 0.3
    time_history = len(pre_synaptic_spike_history)
    pre_synaptic_spike_history = np.asarray(pre_synaptic_spike_history)
    post_synaptic_spike_history = np.asarray(post_synaptic_spike_history)
    """
    calculation of LTD
    """
    if pre_synaptic_spike_history[time_history - 1] == 1:
        post_spike_indices = np.nonzero(post_synaptic_spike_history)[0]
        k = max_weight_update * np.exp(
            -(1 - importance_of_LTD) * ((time_history - 1) - post_spike_indices))
        pre_synaptic_spike_energy = np.sum(k)

    """
    calculation of LTP
    """
    if post_synaptic_spike_history[time_history - 1] == 1:
        pre_spike_indices = np.nonzero(pre_synaptic_spike_history)[0]
        k = max_weight_update * np.exp(
            -(1 - importance_of_LTP) * ((time_history - 1) - pre_spike_indices))
        post_synaptic_spike_energy = np.sum(k)

    deltaw = post_synaptic_spike_energy -
    pre_synaptic_spike_energy
    return deltaw
```

code snippet C.3: Python code for modular implementation of SNNc

D

Third Party Copyright Materials



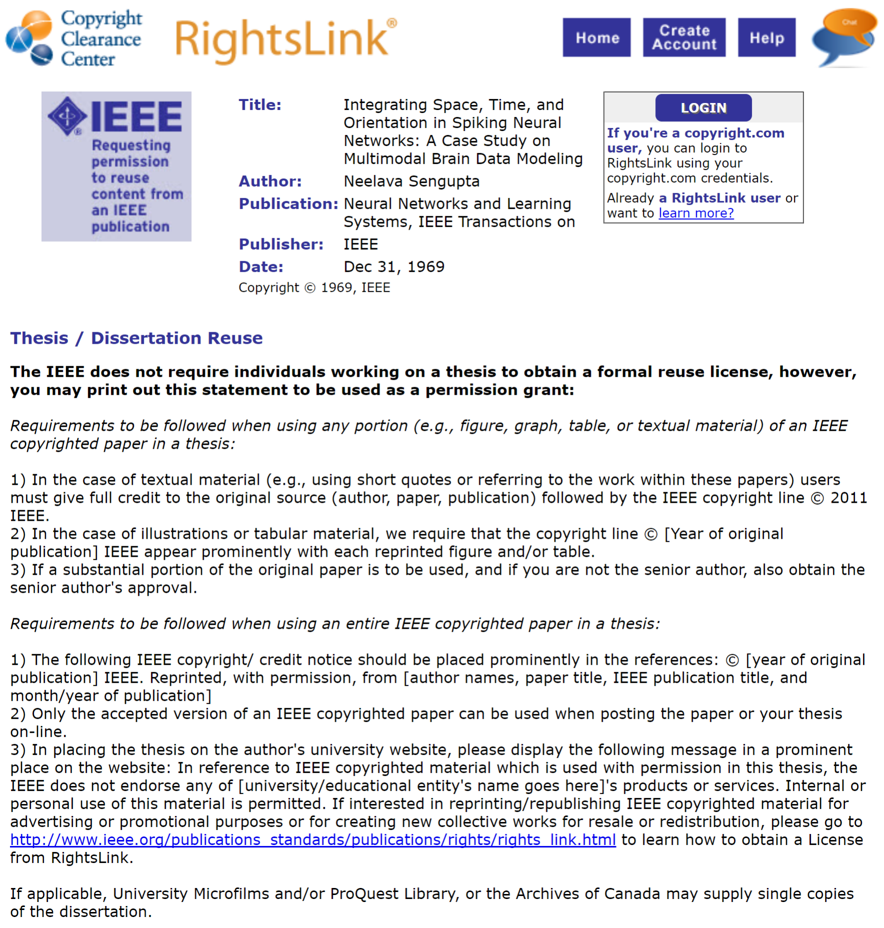
Title: Spike-time encoding as a data compression technique for pattern recognition of temporal data
Author: Neelava Sengupta,Nikola Kasabov
Publication: Information Sciences
Publisher: Elsevier
Date: September 2017
© 2017 Elsevier Inc. All rights reserved.

Logged in as:
Neelava Sengupta
LOGOUT

Please note that, as the author of this Elsevier article, you retain the right to include it in a thesis or dissertation, provided it is not published commercially. Permission is not required, but please ensure that you reference the journal as the original source. For more information on this and on your other retained rights, please visit: https://www.elsevier.com/about/our-business/policies/copyright#Author-rights

Copyright © 2017 Copyright Clearance Center, Inc. All Rights Reserved. Privacy statement. Terms and Conditions.
Comments? We would like to hear from you. E-mail us at customercare@copyright.com

Figure D.1.: Copyright permission of Sengupta and Kasabov (2017)


Copyright Clearance Center
RightsLink®
Home
Create Account
Help
IEEE
Requesting permission to reuse content from an IEEE publication

Title: Integrating Space, Time, and Orientation in Spiking Neural Networks: A Case Study on Multimodal Brain Data Modeling
Author: Neelava Sengupta
Publication: Neural Networks and Learning Systems, IEEE Transactions on
Publisher: IEEE
Date: Dec 31, 1969
Copyright © 1969, IEEE

LOGIN
If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials.
Already **a RightsLink user** or want to learn more?

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Figure D.2.: Copyright permission of Sengupta, McNabb, Kasabov and Russell (2018)

Title: Evolving spatio-temporal data machines based on the NeuCube neuromorphic framework: Design methodology and selected applications
Author: Nikola Kasabov,Nathan Matthew Scott,Enmei Tu,Stefan Marks,Neelava Sengupta,Elisa Capecci,Muhaini Othman,Maryam Gholami Doborjeh,Norhanifah Murli,Reggio Hartono,Josafath Israel Espinosa-Ramos,Lei Zhou,Fahad Bashir Alvi,Grace Wang,Denise Taylor et al.
Publication: Neural Networks
Publisher: Elsevier
Date: June 2016
© 2015 Elsevier Ltd. All rights reserved.

If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials. Already **a RightsLink user** or want to learn more?

Please note that, as the author of this Elsevier article, you retain the right to include it in a thesis or dissertation, provided it is not published commercially. Permission is not required, but please ensure that you reference the journal as the original source. For more information on this and on your other retained rights, please visit: https://www.elsevier.com/about/our-business/policies/copyright#Author-rights

BACK | CLOSE WINDOW

Copyright © 2017 Copyright Clearance Center, Inc. All Rights Reserved. Privacy statement. Terms and Conditions.
Comments? We would like to hear from you. E-mail us at customercare@copyright.com

Figure D.3.: Copyright permission of N. Kasabov, Scott et al. (2016)

Copyright Clearance Center
RightsLink®
Home
Account Info
Help
IEEE
Requesting permission to reuse content from an IEEE publication
LOGOUT

Title: Which method to use for optimal structure and function representation of large spiking neural networks: A case study on the NeuCube architecture
Conference Proceedings: Neural Networks (IJCNN), 2016 International Joint Conference on
Author: Anne Abbott
Publisher: IEEE
Date: July 2016
Copyright © 2016, IEEE

Logged in as:
Neelava Sengupta

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

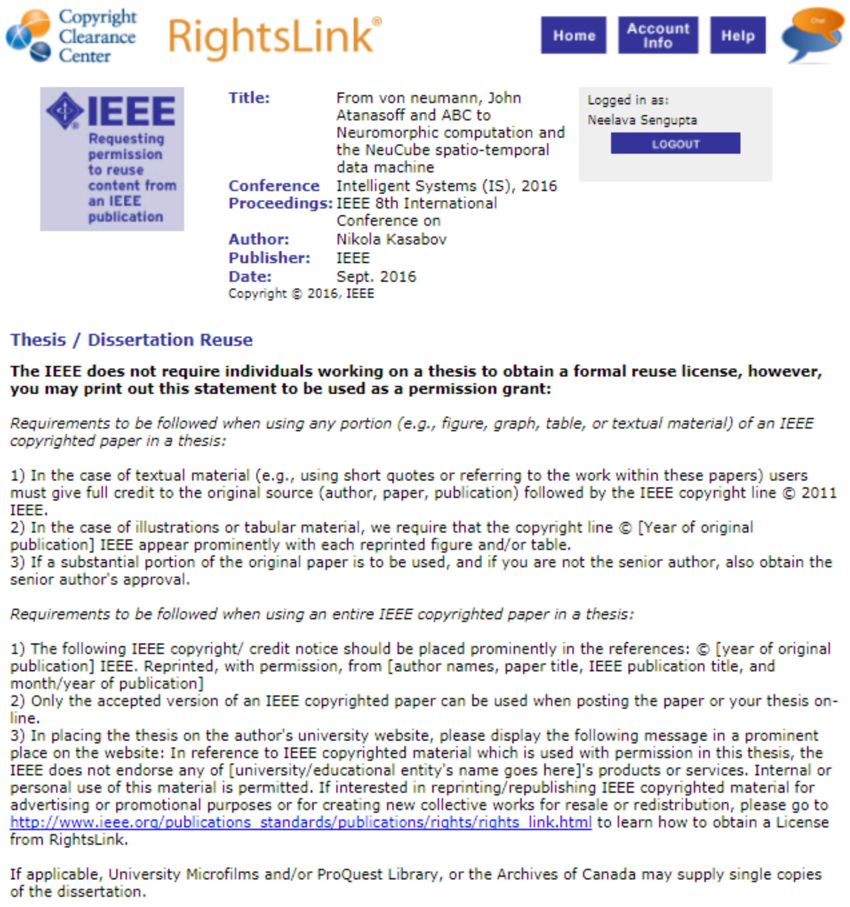
1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Figure D.4.: Copyright permission of Abbott, Sengupta and Kasabov (2016)


Copyright Clearance Center
RightsLink®
Home
Account Info
Help
IEEE
Requesting permission to reuse content from an IEEE publication
LOGOUT

Title:	From von neumann, John Atanasoff and ABC to Neuromorphic computation and the NeuCube spatio-temporal data machine
Conference Proceedings:	Intelligent Systems (IS), 2016 IEEE 8th International Conference on
Author:	Nikola Kasabov
Publisher:	IEEE
Date:	Sept. 2016

Copyright © 2016, IEEE

Logged in as:
Neelava Sengupta

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

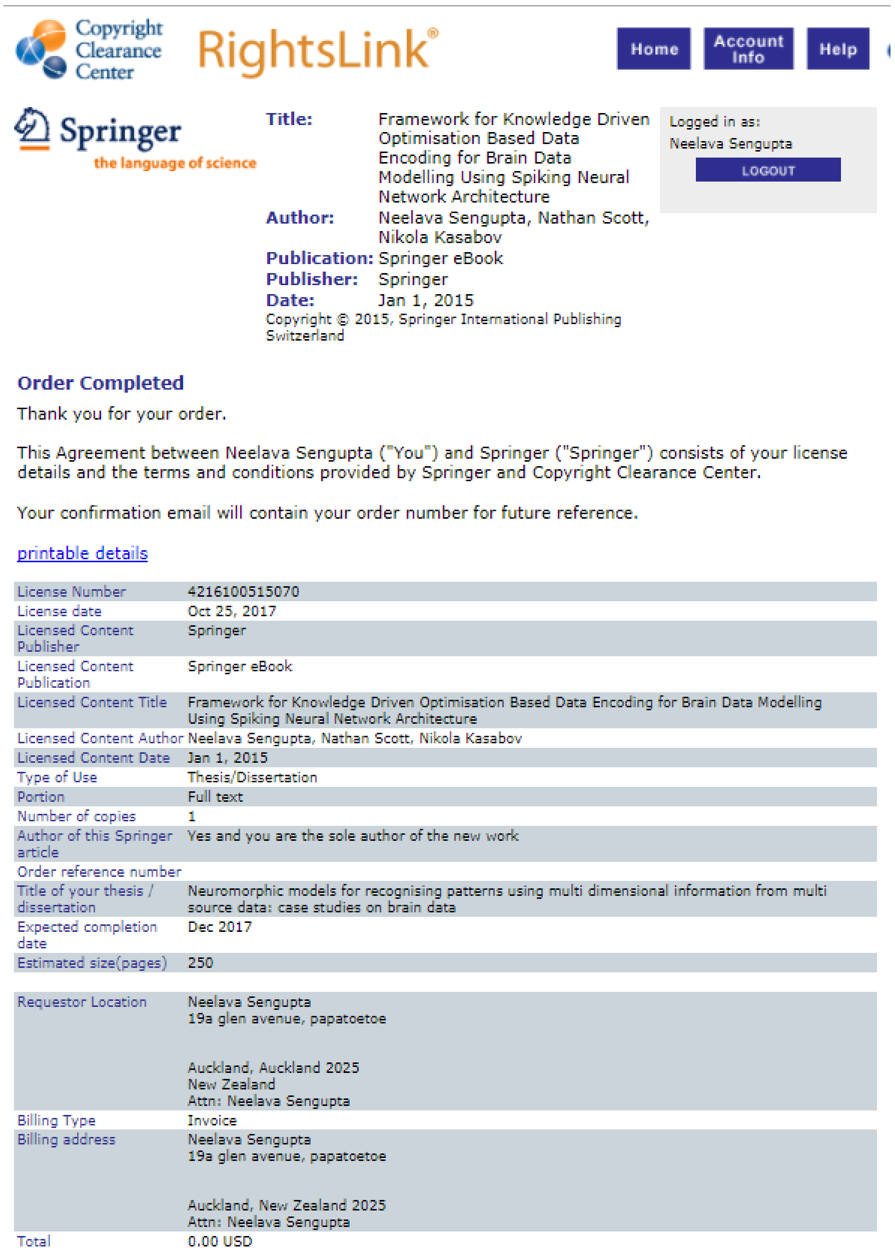
1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Figure D.5.: Copyright permission of N. Kasabov, Sengupta and Scott (2016)



Title: Framework for Knowledge Driven Optimisation Based Data Encoding for Brain Data Modelling Using Spiking Neural Network Architecture
Author: Neelava Sengupta, Nathan Scott, Nikola Kasabov
Publication: Springer eBook
Publisher: Springer
Date: Jan 1, 2015
Copyright © 2015, Springer International Publishing Switzerland

Logged in as:
Neelava Sengupta

Order Completed

Thank you for your order.

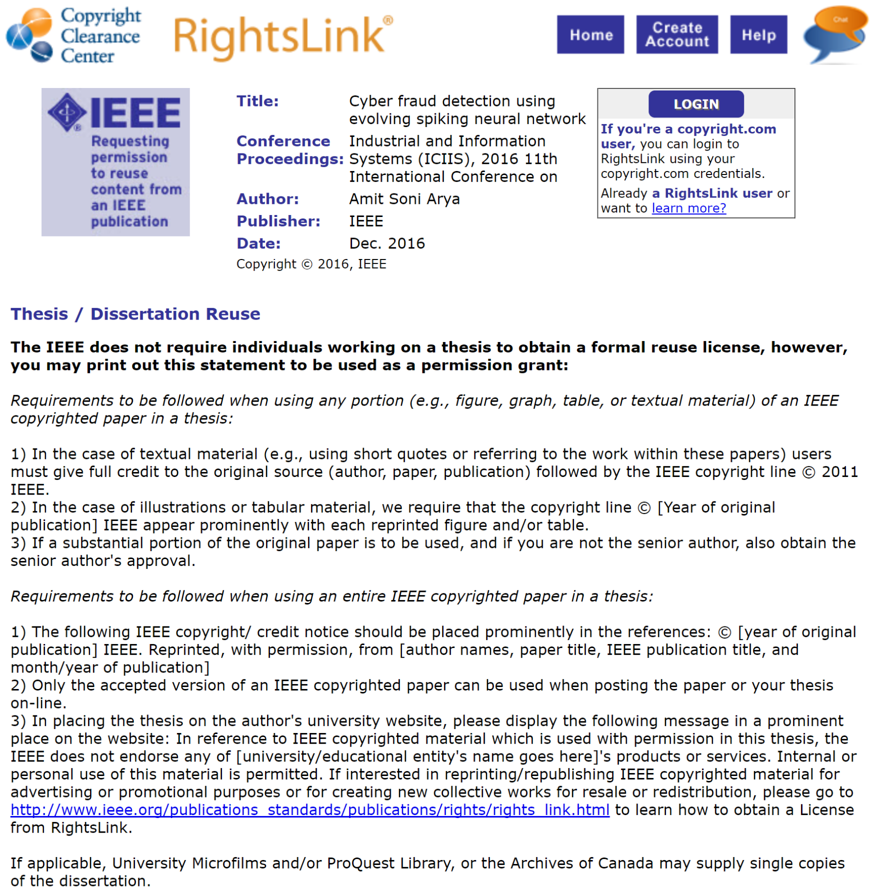
This Agreement between Neelava Sengupta ("You") and Springer ("Springer") consists of your license details and the terms and conditions provided by Springer and Copyright Clearance Center.

Your confirmation email will contain your order number for future reference.

printable details

License Number	4216100515070
License date	Oct 25, 2017
Licensed Content Publisher	Springer
Licensed Content Publication	Springer eBook
Licensed Content Title	Framework for Knowledge Driven Optimisation Based Data Encoding for Brain Data Modelling Using Spiking Neural Network Architecture
Licensed Content Author	Neelava Sengupta, Nathan Scott, Nikola Kasabov
Licensed Content Date	Jan 1, 2015
Type of Use	Thesis/Dissertation
Portion	Full text
Number of copies	1
Author of this Springer article	Yes and you are the sole author of the new work
Order reference number	
Title of your thesis / dissertation	Neuromorphic models for recognising patterns using multi dimensional information from multi source data: case studies on brain data
Expected completion date	Dec 2017
Estimated size(pages)	250
Requestor Location	Neelava Sengupta 19a glen avenue, papatoetoe Auckland, Auckland 2025 New Zealand Attn: Neelava Sengupta
Billing Type	Invoice
Billing address	Neelava Sengupta 19a glen avenue, papatoetoe Auckland, New Zealand 2025 Attn: Neelava Sengupta
Total	0.00 USD

Figure D.6.: Copyright permission of Sengupta, Scott and Kasabov (2015)


Copyright Clearance Center
RightsLink
Home
Create Account
Help
IEEE
Requesting permission to reuse content from an IEEE publication
LOGIN

Title: Cyber fraud detection using evolving spiking neural network

Conference Proceedings: Industrial and Information Systems (ICIIS), 2016 11th International Conference on

Author: Amit Soni Arya

Publisher: IEEE

Date: Dec. 2016

Copyright © 2016, IEEE

If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials.
Already a RightsLink user or want to learn more?

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

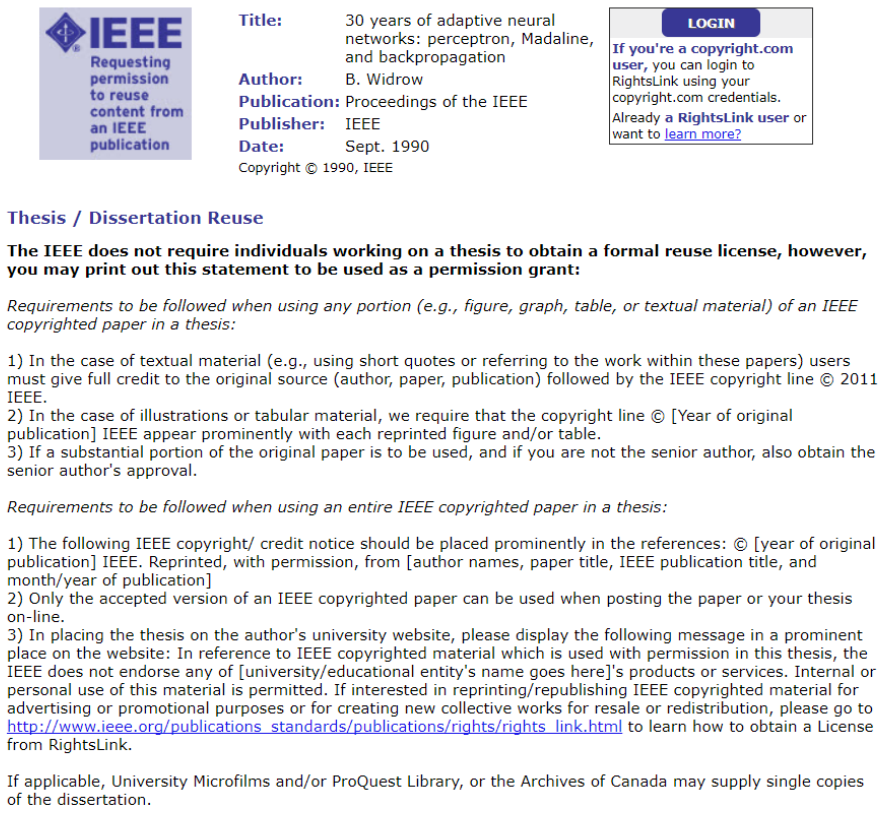
1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Figure D.7.: Copyright permission of Arya, Vadlamani, Valadi, Sengupta and Kasabov (2016)


IEEE
Requesting permission to reuse content from an IEEE publication
LOGIN

Title: 30 years of adaptive neural networks: perceptron, Madaline, and backpropagation

Author: B. Widrow

Publication: Proceedings of the IEEE

Publisher: IEEE

Date: Sept. 1990

Copyright © 1990, IEEE

If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials.
Already a RightsLink user or want to learn more?

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Figure D.8.: Copyright permission of Widrow and Lehr (1990)

Title: Support-vector networks
Author: Corinna Cortes, Vladimir Vapnik
Publication: Machine Learning
Publisher: Springer Nature
Date: Jan 1, 1995
Copyright © 1995, Springer Nature

Logged in as:
Neelava Sengupta
Account #:
3001210280

Order Completed

Thank you for your order.

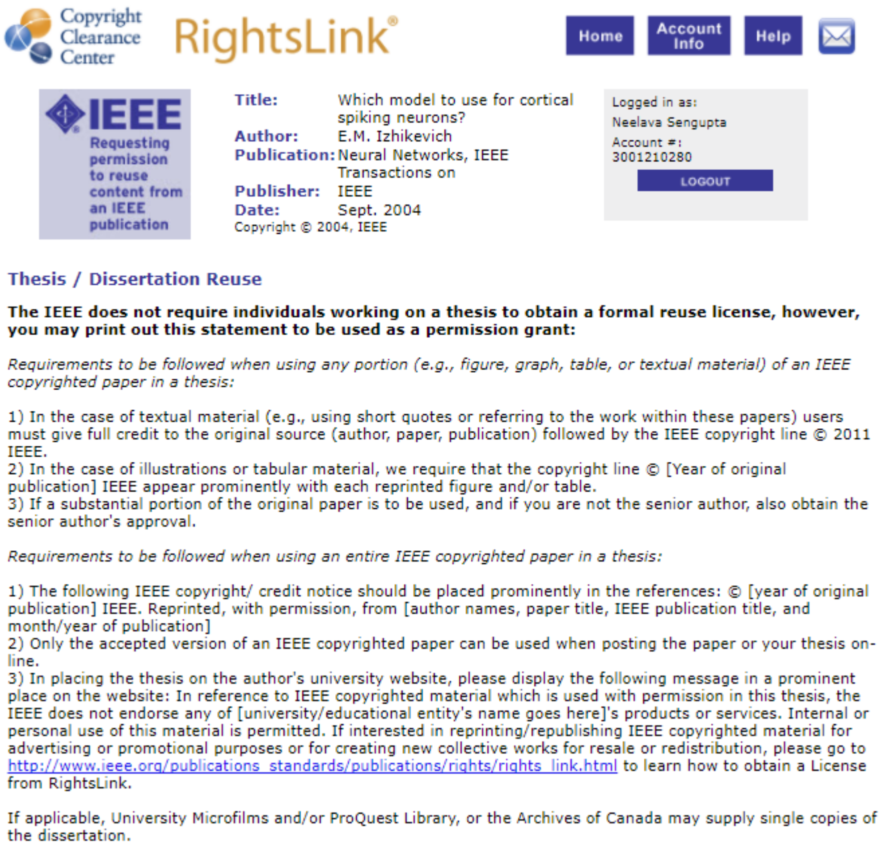
This Agreement between Neelava Sengupta ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

Your confirmation email will contain your order number for future reference.

printable details

License Number	4295680634767
License date	Feb 24, 2018
Licensed Content Publisher	Springer Nature
Licensed Content Publication	Machine Learning
Licensed Content Title	Support-vector networks
Licensed Content Author	Corinna Cortes, Vladimir Vapnik
Licensed Content Date	Jan 1, 1995
Licensed Content Volume	20
Licensed Content Issue	3
Type of Use	Thesis/Dissertation
Requestor type	academic/university or research institute
Format	print and electronic
Portion	figures/tables/illustrations
Number of figures/tables/illustrations	1
Will you be translating?	no
Circulation/distribution	<501
Author of this Springer Nature content	no
Title	Neuromorphic Computational Models for Machine Learning and Pattern Recognition from Multi-modal Time-series Data
Instructor name	Nikola Kasabov
Institution name	AUT
Expected presentation date	Mar 2018
Portions	Fig. 2
Attachment	
Requestor Location	Neelava Sengupta 19a glen avenue, papatoetoe Auckland, Auckland 2025 New Zealand Attn: Neelava Sengupta
Billing Type	Invoice
Billing address	Neelava Sengupta 19a glen avenue, papatoetoe Auckland, New Zealand 2025 Attn: Neelava Sengupta

Figure D.9.: Copyright permission of Cortes and Vapnik (1995)


Copyright Clearance Center
RightsLink®
Home
Account Info
Help
IEEE
Requesting permission to reuse content from an IEEE publication
LOGOUT

Title:	Which model to use for cortical spiking neurons?
Author:	E.M. Izhikevich
Publication:	Neural Networks, IEEE Transactions on
Publisher:	IEEE
Date:	Sept. 2004

Copyright © 2004, IEEE

Logged in as:
Neelava Sengupta
Account #:
3001210280

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

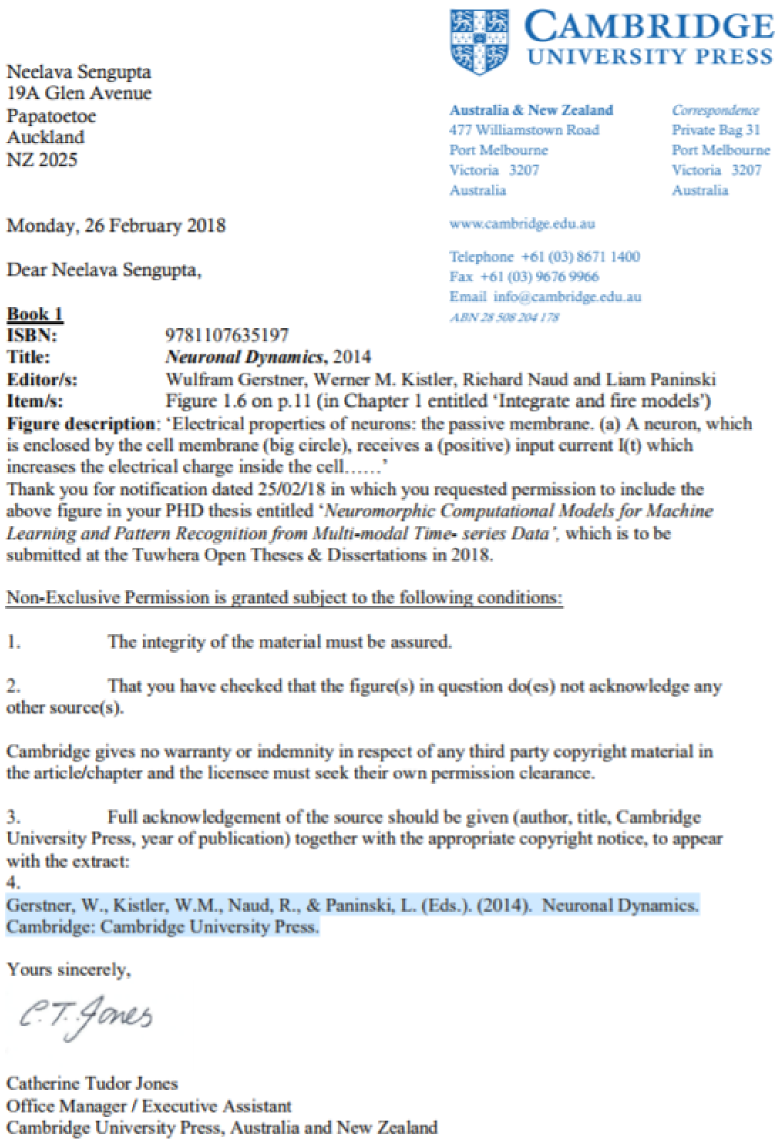
1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Figure D.10.: Copyright permission of Izhikevich (2004)



CAMBRIDGE
UNIVERSITY PRESS

Neelava Sengupta
19A Glen Avenue
Papatoetoe
Auckland
NZ 2025

Australia & New Zealand
477 Williamstown Road
Port Melbourne
Victoria 3207
Australia

Correspondence
Private Bag 31
Port Melbourne
Victoria 3207
Australia

www.cambridge.edu.au

Telephone +61 (03) 8671 1400
Fax +61 (03) 9676 9966
Email info@cambridge.edu.au
ABN 28 508 204 178

Monday, 26 February 2018

Dear Neelava Sengupta,

Book 1

ISBN:	9781107635197
Title:	***Neuronal Dynamics***, 2014
Editor/s:	Wulfram Gerstner, Werner M. Kistler, Richard Naud and Liam Paninski
Item/s:	Figure 1.6 on p.11 (in Chapter 1 entitled 'Integrate and fire models')

Figure description: 'Electrical properties of neurons: the passive membrane. (a) A neuron, which is enclosed by the cell membrane (big circle), receives a (positive) input current I(t) which increases the electrical charge inside the cell……'

Thank you for notification dated 25/02/18 in which you requested permission to include the above figure in your PHD thesis entitled '*Neuromorphic Computational Models for Machine Learning and Pattern Recognition from Multi-modal Time- series Data'*, which is to be submitted at the Tuwhera Open Theses & Dissertations in 2018.

Non-Exclusive Permission is granted subject to the following conditions:

1. The integrity of the material must be assured.

2. That you have checked that the figure(s) in question do(es) not acknowledge any other source(s).

Cambridge gives no warranty or indemnity in respect of any third party copyright material in the article/chapter and the licensee must seek their own permission clearance.

3. Full acknowledgement of the source should be given (author, title, Cambridge University Press, year of publication) together with the appropriate copyright notice, to appear with the extract:

4.

Gerstner, W., Kistler, W.M., Naud, R., & Paninski, L. (Eds.). (2014). Neuronal Dynamics. Cambridge: Cambridge University Press.

Yours sincerely,

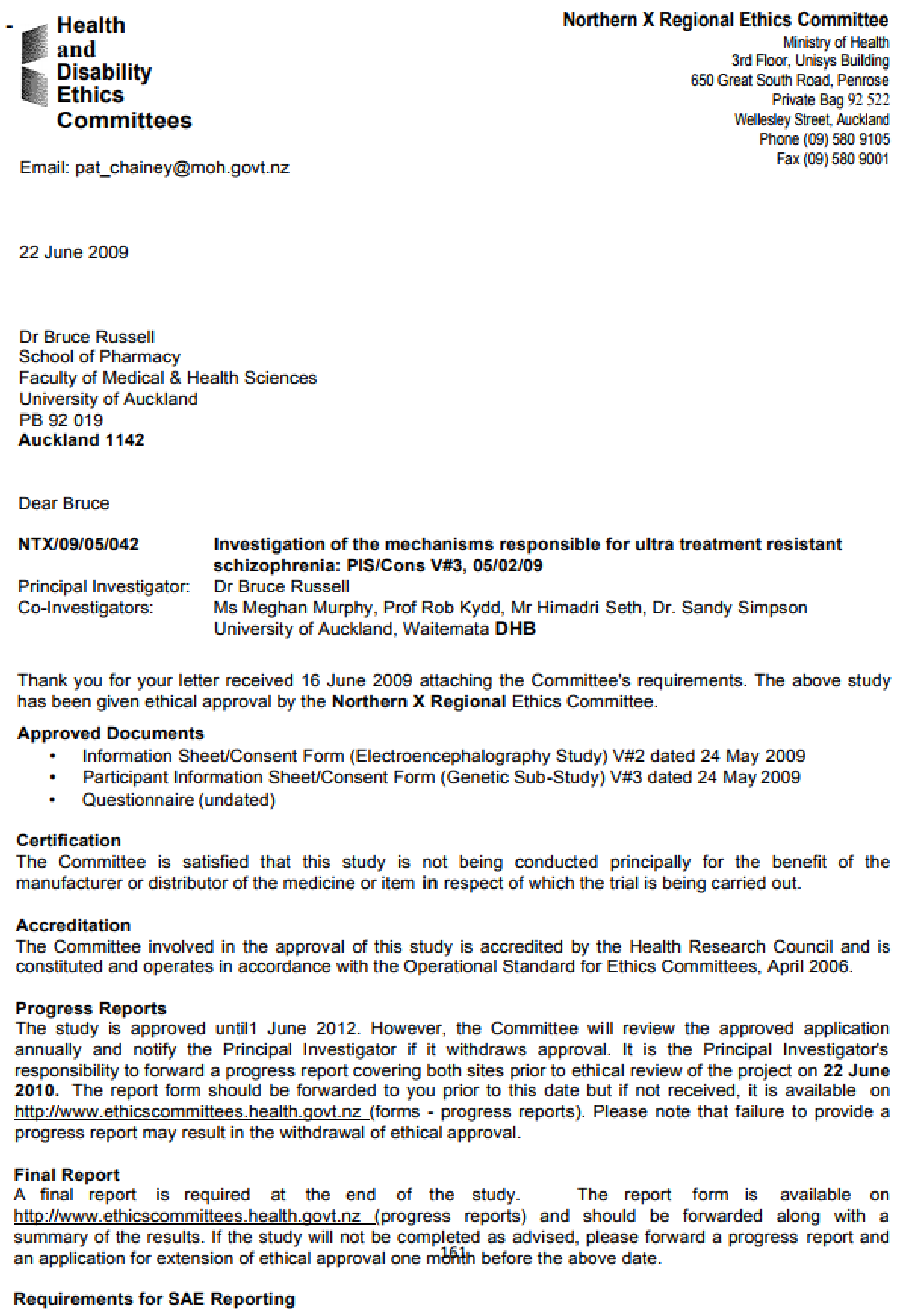
C.T. Jones

Catherine Tudor Jones
Office Manager / Executive Assistant
Cambridge University Press, Australia and New Zealand

Figure D.11.: Copyright permission of Gerstner, Kistler, Naud and Paninski (2014)

E

Ethical Approval of TRS Study



Health and Disability Ethics Committees

Email: pat_chainey@moh.govt.nz

Northern X Regional Ethics Committee
Ministry of Health
3rd Floor, Unisys Building
650 Great South Road, Penrose
Private Bag 92 522
Wellesley Street, Auckland
Phone (09) 580 9105
Fax (09) 580 9001

22 June 2009

Dr Bruce Russell
School of Pharmacy
Faculty of Medical & Health Sciences
University of Auckland
PB 92 019
Auckland 1142

Dear Bruce

NTX/09/05/042	**Investigation of the mechanisms responsible for ultra treatment resistant schizophrenia: PIS/Cons V#3, 05/02/09**
Principal Investigator:	Dr Bruce Russell
Co-Investigators:	Ms Meghan Murphy, Prof Rob Kydd, Mr Himadri Seth, Dr. Sandy Simpson University of Auckland, Waitemata **DHB**

Thank you for your letter received 16 June 2009 attaching the Committee's requirements. The above study has been given ethical approval by the **Northern X Regional** Ethics Committee.

Approved Documents

- Information Sheet/Consent Form (Electroencephalography Study) V#2 dated 24 May 2009
- Participant Information Sheet/Consent Form (Genetic Sub-Study) V#3 dated 24 May 2009
- Questionnaire (undated)

Certification
The Committee is satisfied that this study is not being conducted principally for the benefit of the manufacturer or distributor of the medicine or item **in** respect of which the trial is being carried out.

Accreditation
The Committee involved in the approval of this study is accredited by the Health Research Council and is constituted and operates in accordance with the Operational Standard for Ethics Committees, April 2006.

Progress Reports
The study is approved until1 June 2012. However, the Committee will review the approved application annually and notify the Principal Investigator if it withdraws approval. It is the Principal Investigator's responsibility to forward a progress report covering both sites prior to ethical review of the project on **22 June 2010.** The report form should be forwarded to you prior to this date but if not received, it is available on http://www.ethicscommittees.health.govt.nz (forms - progress reports). Please note that failure to provide a progress report may result in the withdrawal of ethical approval.

Final Report
A final report is required at the end of the study. The report form is available on http://www.ethicscommittees.health.govt.nz (progress reports) and should be forwarded along with a summary of the results. If the study will not be completed as advised, please forward a progress report and an application for extension of ethical approval one month before the above date.

Requirements for SAE Reporting

The Principal Investigator will inform the Committee as soon as possible of the following:

- Any related study in another country that has stopped due to serious or unexpected adverse events
- all serious adverse events occurring during the study in New Zealand which result in hospitalisation or death.
- all serious adverse events occurring during the study worldwide which are considered related to the study.

All SAE reports must be signed by the Principal Investigator and include a comment on whether you consider there are any ethical issues relating to this study continuing due to this adverse event. It is assumed by signing the report, the Principal Investigator has undertaken to ensure that New Zealand investigators are made aware of the event.

Amendments
All amendments to the study must be advised to the Committee prior to their implementation, except in the case where immediate implementation is required for reasons of safety. In such cases the Committee must be notified as soon as possible of the change.

Please quote the above ethics committee reference number in all correspondence.

The Principal Investigator is responsible for advising any other study sites of approvals and all other correspondence with the Ethics Committee.

It should be noted that Ethics Committee approval does not imply any resource commitment or administrative facilitation by any healthcare provider within whose facility the research is to be carried out. Where applicable, authority for this must be obtained separately from the appropriate manager within the organisation.

Yours sincerely

[illegible]

Pat Chainey
Administrator
Northern X Regional Ethics Committee

Cc: L. Neave, Waitemata DHB Research Office