

2020

Brain Disease Detection From EEGs: Comparing Spiking and Recurrent Neural Networks for Non-stationary Time Series Classification

Hristo Stoev
Technological University Dublin

Follow this and additional works at: <https://arrow.tudublin.ie/scschcomdis>

 Part of the [Computer Engineering Commons](#), and the [Computer Sciences Commons](#)

Recommended Citation

Stoev, H. (2020). *Brain disease detection from EEGs: comparing spiking and recurrent neural networks for non-stationary time series classification*. Masters Dissertation. Technological University Dublin.
DOI:10.21427/sv9j-t268

This Dissertation is brought to you for free and open access by the School of Computing at ARROW@TU Dublin. It has been accepted for inclusion in Dissertations by an authorized administrator of ARROW@TU Dublin. For more information, please contact yvonne.desmond@tudublin.ie, arrow.admin@tudublin.ie, brian.widdis@tudublin.ie.



This work is licensed under a [Creative Commons Attribution-NonCommercial-Share Alike 3.0 License](#)

Brain Disease Detection from EEGs - Comparing Spiking and Recurrent Neural Networks for Non-Stationary Time Series Classification



Hristo Stoev

A dissertation submitted in partial fulfilment of the requirements of
Dublin Institute of Technology for the degree of
M.Sc. in Computing (Stream)

6 January 2020

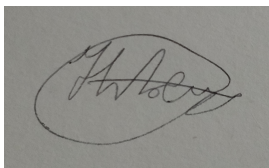
Declaration

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Stream), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed:

A square box containing a handwritten signature in dark ink. The signature is stylized and appears to be 'H. A. O.' or similar, written in a cursive script.

Date: 06 January 2020

Abstract

Modeling non-stationary time series data is a difficult problem area in AI, due to the fact that the statistical properties of the data change as the time series progresses. This complicates the classification of non-stationary time series, which is a method used in the detection of brain diseases from EEGs. Various techniques have been developed in the field of deep learning for tackling this problem, with recurrent neural networks (RNN) approaches utilising Long short-term memory (LSTM) architectures achieving a high degree of success. This study implements a new, spiking neural network-based approach to time series classification for the purpose of detecting three brain diseases from EEG datasets - epilepsy, alcoholism, and schizophrenia. The performance and training time of the spiking neural network classifier is compared to those of both a baseline RNN-LSTM EEG classifier and the current state-of-the-art RNN-LSTM EEG classifier architecture from the relevant literature. The SNN EEG classifier model developed in this study outperforms both the baseline and state-of-the-art RNN models in terms of accuracy, and is able to detect all three brain diseases with an accuracy of 100%, while requiring a far smaller number of training data samples than recurrent neural network approaches. This represents the best performance present in the literature for the task of EEG classification.

Keywords: artificial intelligence, classification, time series, spiking neural network, recurrent neural network, LSTM, Electroencephalogram (EEG), epilepsy, alcoholism, schizophrenia

Acknowledgments

I would like to thank my supervisor, Dr. Basel Magableh, for the ample guidance, advice, and attention provided over the course of the conducted research. I am extremely grateful and this dissertation would not have been possible without his help.

Additionally, I would like to thank Dr. Luca Longo for his assistance in formulating the original research project proposal. His contribution helped me determine an appropriate direction for my dissertation.

Finally, I would like to express my gratitude to my parents and to my girlfriend for offering their unyielding support and encouragement while I undertook this research project.

The complete statistical programming code used for this dissertation is available at the following URL:

github.com/HristoStoevAI/TUDublinDissertation

Contents

Declaration	I
Abstract	II
Acknowledgments	III
Contents	IV
List of Figures	VIII
List of Tables	X
List of Acronyms	XI
1 Introduction	1
1.1 Background	1
1.2 Research Problem	4
1.3 Research Question	7
1.3.1 Research Hypotheses	7
1.4 Research Objectives	7
1.5 Research Methodologies	9
1.6 Scope and Limitations	11
1.7 Document Outline	12
2 Literature Review	14
2.1 Data with Temporal Components	14

2.1.1	Time Series	15
2.1.2	EEG Time Series	15
2.1.3	Event-Based Data	16
2.2	Artificial Neural Networks	17
2.2.1	ANN Neuron Model	18
2.2.2	ANN Decision-Making	18
2.2.3	ANN Architecture	19
2.2.4	Learning in ANNs	19
2.3	Recurrent Neural Networks	21
2.3.1	Learning in RNNs	22
2.3.2	LSTM	23
2.4	Spiking Neural Networks	24
2.4.1	SNN Implementation Approaches	25
2.4.2	Spike Train Generation	29
2.4.3	Spiking Neuron Models	31
2.4.4	Learning Rules in SNNs	33
2.5	Summary, Limitations and Gaps in Literature Review	37
3	Design and Methodology	39
3.1	Data Understanding	39
3.1.1	EEG recordings of healthy adolescents and adolescents with symptoms of schizophrenia	39
3.1.2	EEG recordings of alcoholics and control subjects	40
3.1.3	EEG of healthy subjects and subjects with epilepsy	41
3.2	Data Preparation	42
3.2.1	Data Encoding	42
3.2.2	Dataset Construction	42
3.2.3	Dataset Balancing	43
3.2.4	Data Splitting	43
3.3	Software	44

3.4	Modelling	45
3.4.1	SNN Model Architecture	45
3.4.2	Baseline RNN Model Architecture	49
3.4.3	State-of-the-Art RNN Model Architecture	50
3.5	Performance Evaluation	51
4	Implementation and Results	55
4.1	Data Splitting	55
4.2	Hyperparameter Selection	56
4.2.1	SNN Hyperparameters	56
4.3	Number of Training Epochs	58
4.4	Final Model Implementations	59
4.4.1	Final Baseline RNN Implementations	59
4.4.2	Final State-of-the-Art RNN Implementations	61
4.4.3	Final SNN Implementations	62
4.5	Model Results	63
4.5.1	Baseline RNN Results	64
4.5.2	State-of-the-Art RNN Results	68
4.5.3	SNN Results	73
5	Evaluation, Analysis and Discussion	80
5.1	Comparison of Classifiers Results	80
5.2	Hypothesis Evaluation	82
5.2.1	Hypothesis 1 - Accuracy	82
5.2.2	Hypothesis 2 - Training Time	86
5.3	Summary of Key Findings	89
5.4	Strengths and Limitations	92
5.5	Considerations of Previous Research	92
6	Conclusions	94
6.1	Research Overview	94

6.2	Problem Definition	96
6.3	Contributions to Body of Knowledge	96
6.4	Future Work and Recommendations	97
	References	99
	A	118

List of Figures

2.1	Standard EEG electrode placements and channel names according to the 5% scheme.	17
2.2	The MP Computational Neuron	18
2.3	A Multi-Layer Perceptron with One Hidden Layer	19
2.4	A recurrent neuron, unrolled over time	21
2.5	RNN and LSTM cells	23
2.6	Biological neuron structure	24
2.7	A typical feedforward spiking neural network architecture, with a single hidden layer	26
3.1	Schizophrenia Dataset Target Class Proportions	40
3.2	Alcoholism Dataset Target Class Proportions	41
3.3	Epilepsy Dataset Target Class Proportions	41
3.4	Baseline RNN Topology	50
52figure.caption.43		
3.6	Binary Classification Confusion Matrix	53
4.1	Final Keras Model Description for Schizophrenia Baseline RNN Classifier	60
4.2	Final Keras Model Description for Alcoholism Baseline RNN Classifier	60
4.3	Final Keras Model Description for Epilepsy Baseline RNN Classifier . .	61
4.4	Schizophrenia Baseline RNN Classifier - Confusion Matrix	64
4.5	Alcoholism Baseline RNN Classifier - Confusion Matrix	66
4.6	Epilepsy Baseline RNN Classifier - Confusion Matrix	67

4.7	Schizophrenia State-of-the-Art RNN Classifier - Confusion Matrix . . .	69
4.8	Alcoholism State-of-the-Art RNN Classifier - Confusion Matrix	70
4.9	Epilepsy State-of-the-Art RNN Classifier - Confusion Matrix	72
4.10	Schizophrenia SNN Classifier - Confusion Matrix	73
4.11	Alcoholism SNN Classifier - Confusion Matrix	75
4.12	Epilepsy SNN Classifier - Confusion Matrix	76
4.13	Classifiers' Test Accuracies Compared to Number of Training Epochs (RNN Approaches)	78
4.14	Classifiers' Test Accuracies Compared to Number of Training Epochs (SNN Approach)	79
A.1	Final Keras Model Description for Schizophrenia State-of-the-Art RNN Classifier.	119
A.2	Final Keras Model Description for Alcoholism State-of-the-Art RNN Classifier.	120
A.3	Final Keras Model Description for Epilepsy State-of-the-Art RNN Clas- sifier	121

List of Tables

4.1	Hyperparameters Evaluated for the SNN Models	56
4.2	Schizophrenia Baseline RNN Classifier Performance	65
4.3	Alcoholism Baseline RNN Classifier Performance	66
4.4	Epilepsy Baseline RNN Classifier Performance	68
4.5	Schizophrenia State-of-the-Art RNN Classifier Performance	69
4.6	Alcoholism State-of-the-Art RNN Classifier Performance	71
4.7	Epilepsy State-of-the-Art RNN Classifier Performance	72
4.8	Schizophrenia SNN Classifier Performance	74
4.9	Alcoholism SNN Classifier Performance	75
4.10	Epilepsy SNN Classifier Performance	76
5.1	SNN and LSTM Classifiers Comparison	81
5.2	SNN and LSTM Training Time Comparison (in seconds)	82
5.3	Schizophrenia WSR Test: SNN - Baseline LSTM	83
5.4	Schizophrenia WSR Test: SNN - State-of-the-art LSTM	84
5.5	Epilepsy WSR Test: SNN - Baseline LSTM	85
5.6	Epilepsy WSR Test: SNN - State-of-the-art LSTM	85
5.7	Schizophrenia WSR Test: SNN - Baseline LSTM	87
5.8	Schizophrenia WSR Test: SNN - State-of-the-art LSTM	87
5.9	Epilepsy WSR Test: SNN - Baseline LSTM	88
5.10	Epilepsy WSR Test: SNN - State-of-the-art LSTM	89
5.11	Hypotheses Tests Results	90

List of Acronyms

ANN	Artificial Neural Network
RNN	Recurrent Neural Network
SNN	Spiking Neural Network
LSTM	Long Short Term Memory
GRU	Gated Recurrent Unit
HH	Hodgkin-Huxley
LIF	Leaky Integrate-and-Fire
SOTA	State-of-the-Art
WSR	Wilcoxon Signed Rank
EEG	Electroencephalogram
AI	Artificial Intelligence
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative

Chapter 1

Introduction

1.1 Background

What a peculiar privilege has this little agitation of the brain which we call 'thought' (Hume, 1779).

Within the past decade, truly significant strides in progress have been made in the field of AI, both in terms of the performance of its existing applications, and in terms of the breadth of its potential uses. Computers are being taught to perceive, understand, and learn from the world using newer, ever more powerful techniques. This evolution in AI has made many tasks, once considered to be exclusively achievable by humans due to their complexity, unpredictability or required creativity, to be fully or partially automated.

At the same time, the wider adoption of applied AI in industry and consumer products has made it play a more visible role than ever in people's daily lives. AI technologies have led to transformations in healthcare (Jiang et al., 2017), finance (Culkin & Das, 2017), language processing (Young, Hazarika, Poria, & Cambria, 2018), warfare (Bode & Huelss, 2018), autonomous vehicles (Tokody, Mezei, & Schuster, 2017), facial recognition (Taigman, Yang, Ranzato, & Wolf, 2014), and manufacturing (B.-h. Li, Hou, Yu, Lu, & Yang, 2017), to name just a few areas. While nobody knows precisely what the future has in store for AI, its potential has captured the interest of the public and, more importantly, of industry and business. A study conducted by

venture capital firm Atomico found that European AI firms gathered \$2.9B of funding in 2019 alone and close to \$10B since 2015, outperforming all other categories in the tech sectors (Atomico & Slush, 2019).

The expansive growth of AI is based on developments in the field of Artificial Neural Networks. Originally conceived in the 1940s (McCulloch & Pitts, 1943), ANNs were developed to replicate the information processing mechanics present in the human brain. The most sophisticated processing unit known to man, the functions of the brain are highly energy efficient, massively parallel, and are able to facilitate fast, generalisable learning with minimal supervision (Poo, 2018). ANNs attempts to capture this ability, but it hasn't been until recent years that they have experienced a significant growth in their capabilities. This has led to the emergence of the concept of 'deep learning', which employs the use of more complex topologies for AI models to be able to tackle more difficult and complex problems

One particularly intriguing advance made in AI is the development of recurrent ANN modelling techniques, such as the Recurrent Neural Network (RNN). Early ANNs were exclusively feed-forward, meaning that they were not able to model the relationship between the passage of time and the data. However, since humans perceive and interact with their environment in a fundamentally temporal manner (Peuquet, 1995), in order for AI to be truly useful to humans, it needs to be able to do the same. This is the motivation behind the development of recurrent ANN models, and the LSTM (Long Short Term Memory) architecture, a special type of RNN layer. LSTMs are not only able to conduct information processing of temporal data, but also to extract and store long-term memories from the data sequence using includes an additional set of learning mechanisms (Hochreiter & Schmidhuber, 1997), which are then used to influence the model's future decisions. This allows LSTMs to solve many problems in domains where temporal sequences of data are fundamental, and where past context is needed in order to make decisions about the future of the sequence, such as text and language processing.

This evolution in AI capabilities and applications is enabled to a large extent by advancements made in computing hardware, primarily the widespread adoption of

GPU acceleration hardware. In particular, LSTMs are highly reliant on GPUs' high memory bandwidth since they incur a large memory overhead (Nowak, Taspinar, & Scherer, 2017). This is particularly true for language modelling applications, where often very large vocabularies of millions of words need to be represented within the recurrent model, making learning difficult in the case of insufficient memory bandwidth (X. Li, Qin, Yang, Hu, & Liu, 2016). Another type of acceleration provided by GPUs is parallelisation, usually through Nvidia's CUDA platform (Garland et al., 2008). This has allowed the ANN learning process (known as 'training') to progress from traditionally being conducted on a single processor thread, to being carried out simultaneously across multiple threads. GPU acceleration has dramatically sped up learning in recurrent ANN models, which is generally quite computationally expensive.

One problem area where recurrent ANNs, specifically those with LSTM architectures, have been particularly useful is in the classification of sequences of data. This has applications in a number of domains (Karim, Majumdar, Darabi, & Harford, 2018), such as diagnosis of disease from recorded biometrics, and gesture recognition. Quite often, temporal data in the form of sequences collected from the environment tends to exhibit certain characteristics, one of which is non-stationarity, meaning that the statistical properties of the data, like its mean and variance, change as the sequence progresses. Non-stationarity in particular makes modelling the sequence data using parametric, non-AI methods problematic without first applying transformations on the data (Amjad & Shah, 2016). Non-stationary sequence modelling is key in tasks such as speech separation from ambient noise (Wöllmer, Zhang, Weninger, Schuller, & Rigoll, 2013), and interpretation of EEG data (Klonowski, 2009), all of which are important applications in the field of AI.

While RNN models and have proven to be highly capable of learning temporal patterns from non-stationary sequences of data, they are not the only AI paradigm that can conduct processing of memory information. One alternative is Spiking Neural Networks (SNNs), which differ from conventional Artificial Neural Networks due to their event-based nature. SNNs employ binary activation 'spikes' in time-space for information processing, while ANNs use scalar data. While ANNs are only abstractly

inspired by the brain’s learning methods, SNNs emulate its biology in a more stringent manner, allowing them to replicate the brain’s excellent processing capabilities more so than ANNs. For instance, SNNs are massively parallelizable, and this allows them to match and even exceed the performance of conventional AI models, all while exhibiting a lower energy and computational overhead (Pfeiffer & Pfeil, 2018).

1.2 Research Problem

In recent years, the application of SNNs for processing and learning from non-stationary sequences of data has become an exciting research topic in the field of AI. This is primarily due to the fact that SNNs demonstrate highly promising potential to alleviate many of the limitations of more commonly-used recurrent ANN models. As stated earlier, ANNs can be categorised into strictly feed-forward models, which are not able to process information in sequences of data, and recurrent models, which are able to do so. While recurrent ANNs have become very advanced in terms of their performance and range of applications, their functionality still uses strictly feed-forward ANN as a foundation. Essentially, one could conceptualise recurrent ANNs as being feed-forward ANNs where sequence modelling abilities were inserted in addition to their base operation, despite the fact that the original creators of the feedforward ANN did not develop them with this usage in mind. As such, the method used by RNNs to process temporal information is fundamentally inefficient. On the other hand, from the very inception of the SNN, information processing of sequence data has always been a central feature, since this is the form of information processing utilised by the human brain, the emulation of which is the very motivation behind the creation of the SNN.

The efficiency advantage of SNNs over recurrent ANNs does not just stem from the origins of the two approaches - it can be seen directly by comparing how the two approaches represent the passage of time and learn from how it affects the sequence data. Recurrent ANNs’ ability to recognise temporal patterns in sequence data stems from the fact that, when ‘unrolled’ over time, they are actually composed of many strictly feed-forward ANN models, which are linked together in a dependent manner

such that they can affect each other’s decision-making. This is a fundamentally inefficient approach to memory information processing, especially as the length of the data sequence being modelled increases (Nowak et al., 2017). Even worse, in the case of the LSTM architecture, the already inefficient recurrent ANN is augmented through the use of computationally expensive memory gates and hidden states, which add an additional cognitive load to the algorithm, resulting in the model taking even longer to learn temporal patterns (Z. Li et al., 2019). SNNs, on the other hand, do not rely on strictly feed-forward ANNs for information processing in any way whatsoever. Even the most fundamental SNN architectures are able to effectively perform temporal information processing by design, as a result of their event-based information processing. On top of this, SNNs do not require additional mechanisms to extract and learn long-term memories from data sequences, as is the case with LSTM-RNNs. This is because SNNs make use of learning methods which are designed from the ground up to be efficient at modelling patterns in sequences of data by incorporating the precise timing between spiking events into the learning process. This is a completely different approach to learning than the one used in RNNs, where the learning algorithms do not directly incorporate the passage of time, but are simply modified versions of those used in strictly feed-forward ANNs (Mozer, 1995). Research has shown that it is precisely due to these differences in learning approaches that SNNs exhibit shorter training times than recurrent ANNs when extracting temporal patterns present in non-stationary sequences of data (Neil, Pfeiffer, & Liu, 2016).

SNNs also have a more well-defined roadmap for future research than RNNs, which stems from their purpose to emulate the information processing of the human brain. The brain is excellent at a number of facets of learning that are, so far, infeasible for AI - it can multi-task, learn quickly with very little supervision and easily generalize existing skills to solve wide ranges of problems (Poo, 2018). The ultimate goal of the SNN approach to AI is to replicate these qualities of biological brains in computing systems, and research in the literature is focused on this goal. On the other hand, RNNs research lacks an overall goal such as this. The trend in RNNs is the development of approaches and solutions to problems using architectures which are

highly specialised to the task at hand, thus sacrificing generalisability. For instance, two RNNs designed for speech recognition, and for text classification, respectively, would require entirely different architectures and layer topologies. Therefore, novel applications of RNNs require extensive trial and error to find the optimal architecture for the problem. This inflexibility is another limitation that SNNs have the potential to alleviate - even without being fully biologically accurate. SNNs are already more suited to dynamic problem environments than RNNs because they can handle changes in the input data shape or feature space, such as different data sampling rates that change over time and across features. This is facilitated by their event-based processing. RNNs, on the other hand, need to have their entire architecture re-evaluated and their parameters re-trained in the event of such changes in the environment.

As to for which specific sequence data learning tasks the above advantages of SNNs hold true, the research is not conclusive. This is particularly the case for modelling non-stationary sequences of data, where the use of traditional modelling techniques is problematic due to their requirement that the statistical properties of the data sequence remain stationary (Amjad & Shah, 2016). Despite the potential for SNNs to become an alternative approach to AI learning from temporal data, studies directly comparing the performance of SNNs and RNNs for time series classification have not been conducted in the literature to this author’s knowledge. As a result, this research aims to directly investigate whether or not an SNNs-based approach to time series classification can be used instead of RNNs to avoid some of the current limitations faced by implementations using the latter. Due to the lack of research in this problem area, it was decided that an investigation would be academically useful. The research problem studied in this project focused specifically on the problem of detecting brain diseases using EEG recordings, as this is a type of non-stationary time series classification. Additionally, many EEG datasets have been made available to researchers for this purpose, eliminating the need for data collection as part of the research project.

1.3 Research Question

To what extent do neural network models, built using a Spiking Neural Network, have superior accuracy and/or training time to models built using a Recurrent Neural Network when implemented for classification of non-stationary time series datasets?

1.3.1 Research Hypotheses

The following list outlines the hypotheses that are evaluated by this project in order to answer the above research question:

Hypothesis 1: *H0: Spiking Neural Network models do not have higher accuracy than Recurrent Neural Network models when implemented for classification of non-stationary time series data.*

H1: Spiking Neural Network models have higher accuracy than Recurrent Neural Network models when implemented for classification of non-stationary time series data.

Hypothesis 2: *H0: Spiking Neural Network models do not have a faster training time than Recurrent Neural Network models when implemented for classification of time series data.*

H1: Spiking Neural Network models have a faster training time than Recurrent Neural Network models when implemented for classification of time series data.

1.4 Research Objectives

The research objectives for this project are as follows:

1. Determine which Spiking Neural Network simulator package to use

A literature review will be the research method used to compile and compare each of the potential options for the Spiking Neural Network simulator package. The criterion for selecting the appropriate simulator package is which option offers the most comprehensive and diverse implementations of network topologies and supervised learning rules.

2. Determine the Spiking Neural Network topology and learning rule

This includes deciding on a spike encoding strategy and which supervised learning rules to use (Tavanaei, Ghodrati, Kheradpisheh, Masquelier, & Maida, 2019). This will be done by conducting secondary research to determine the approach that will lead to the best classification results. One important requirement for the network topology and learning rule is that it is supported by the simulation package decided on in Step 1.

3. Determine the baseline and state-of-the-art recurrent neural network architecture

This will be done by reviewing the existing research on EEG classification using RNNs and basing the RNN architecture decision on the most effective existing implementations. Two model architectures will be selected - a baseline LSTM-RNN approach, and the current state-of-the-art model for EEG classification with LSTM-RNNs, and these will be determined by conducting secondary research to determine the approach that will lead to the best classification results.

4. Develop the optimal SNN classification model for each dataset

This will be done using whatever simulator, topology and learning rule was selected in Steps 1 and 2. This step also involves finding the optimal model hyperparameters for modelling each of the three experimental datasets.

5. Replicate the baseline and state-of-the-art RNN classification models for each dataset

The baseline and state-of-the-art RNN topologies decided upon in Step 3 will be developed using the deep learning library, Keras, as this allows for rapid prototyping and fine-tuning of models.

6. The optimal classification models will be trained using the SNN and RNN approaches on each of the three experimental EEG datasets

The optimal trained classifier models will then be used to collect the necessary training times and classification performance metrics. Training will be conducted until the models' maximum performance is reached.

7. The performance metrics collected by evaluating the optimal SNN and RNN classifiers will be analysed and compared

This will be done so that the research hypotheses described in Section 1.3.1 can be accepted or rejected based on the conclusions drawn from the available experimental results.

1.5 Research Methodologies

The overall methodology used to conduct this research project is one of empirical evaluation. The specific research methodologies used to investigate each hypothesis outlined in Section 1.3.1 are described in this section.

Hypothesis 1: For each of the three experimental EEG datasets, the performance metrics (described in detail in section 3.6) of the hyperparameter-optimized SNN and RNN classifiers are calculated. The performance metrics for the RNN and SNN classifiers are compared, to determine whether or not there is a significant increase

in classification performance when using the SNN-based model. This is quantitative research. Statistical analysis will be used to determine if the difference in performance metrics between the SNN and the RNN model is significant.

In order to ensure that the performance metrics and training times are reliable, k-fold cross-validation is employed, as this reduces the chance of over-fitting the model to the training data (Mitchell, 1997). Since there is no guarantee that the mean of the performance metrics over each data fold is normally distributed, and the samples are not independent, the non-parametric Wilcoxon Signed-Rank test is used to determine if there is a statistically significant difference on model classification performance by whether or not the architecture used was the RNN or the SNN. If the p-value from the Wilcoxon Signed-Rank test is below 0.05, there is a significant difference between the RNN and SNN performance metrics averaged over each fold. These findings relate to the research question as they will discern whether models built using a Spiking Neural Network have a superior performance than models built using an RNN when implemented for classification of time series data.

Hypothesis 2: For each of the three experimental EEG datasets, the training times for the hyperparameter-optimized SNN and RNN models are collected. The training times for the RNN and SNN classifiers are then compared, to determine whether or not there is a significant decrease in training time when using the SNN-based model. The goal of this research is to determine whether classification using a SNNs requires less training time than using an RNN architecture. This is quantitative research. Statistical analysis will be used to determine if the difference in training time between the SNN model and the RNN model is significant.

In order to ensure that the collected training times are reliable, k-fold cross-validation is employed, as this reduces the chance of over-fitting the model to the training data (Mitchell, 1997). Since there is no guarantee that the mean of the training times over each data fold is normally distributed, and the samples are not independent, the non-parametric Wilcoxon Signed-Rank test is used to

determine if there is a statistically significant difference on model training times by whether or not the architecture used was RNN-based or SNN-based. If the p-value from the Wilcoxon Signed-Rank test is below 0.05, there is a significant difference between the RNN and SNN training times averaged over each fold. These findings relate to the research question as they will discern whether models built using a Spiking Neural Network have superior training times than models built using an RNN when implemented for classification of time series data.

1.6 Scope and Limitations

The scope of this research is limited to the detection of brain diseases from EEG datasets. The problem of EEG signal classification was chosen due to the nature of EEGs - they are examples of non-linear, and non-stationary time series data. Three different brain disease datasets are used, for the detection of schizophrenia, epilepsy, and alcoholism. Each dataset contains sequence EEG data, measured for different lengths of time, at different sampling rates, with different sample sizes and numbers of features, and contained recordings from subjects with both positive and negative brain disease diagnoses. Due to the differences in structure between each of three datasets, the results of the research experiment, and the conclusions drawn from them, are applicable to many different formats of recorded EEG data, making the experiment academically useful. However, a limitation of this research would be that since each of the three datasets used in the experiment features a strictly binary target variable, generalising the conclusions beyond binary classification problems would be problematic.

More generally, the research scope was limited specifically to the problem of EEG classification. Conclusions drawn from the results of this research would not be applicable to other types of problems involving modelling sequence data with SNNs and RNNs, such as prediction problems.

A significant limitation of this research is the lack of availability of neuromorphic hardware for academic use. Currently, the most efficient method of implementing

SNNs involves using specialised VLSI hardware to enable analog computation (Indiveri et al., 2011). This allows the researcher to assign physical quantities to represent the internal decision-making properties of SNNs, such as the neuron membrane potential (further discussed in section 2.4.3), reducing the total number of necessary computations, and allowing for a very efficient, low-power implementation. However, the majority of research efforts towards the development of neuromorphic chips are made in private corporations, rather than academic institutions, such as Intel’s Loihi chip (Davies et al., 2018) and IBM’s TrueNorth chip (Hsu, 2014). These technologies are not available for use by the general public. In academic spheres, the SpiNNaker project (Furber, Galluppi, Temple, & Plana, 2014) is the only neuromorphic computing architecture that is available to researchers through cloud access, but access to the SpiNNaker platform is only available through invitation and use-case approval. This researcher made a use-case request for access to the SpiNNaker architecture, but the request was not approved as of the time of writing of this dissertation. Outside of these options, the rest of the hardware-implemented SNNs in the literature all make use of custom hardware designed by the authors (Indiveri et al., 2011). This approach was not feasible for this researcher. As a result, the experiment was conducted with the use of a synchronous or ‘clock-driven’ SNN simulation (Brette et al., 2007). It should be kept in mind that this digital simulation approach to SNN implementation leads to lower model efficiency than hardware-based analog models.

1.7 Document Outline

The following section provides a guide to the chapters in this dissertation, and motivates their function.

- Chapter 2 provides a review of academic topics relevant to this research. Fields reviewed include the nature of temporal data, a contextual analysis of feed-forward ANNs, recurrent ANNs and memory architectures. The state-of-the-art in abstract mathematical neuron models and data encoding methods for SNNs are examined. This chapter also highlights the limitations and gaps present in

the literature.

- Chapter 3 outlines the overall design of the experiments conducted in this research. The data engineering process is described in detail, and the choice of software used to run the experiment is motivated. The methodologies of the AI models used in the research are also specified, including evaluation metrics and model optimization.
- Chapter 4 consists of a report of the specific implementation of the conducted experiment. This includes a description of the final, optimised architectures, topologies and hyperparameters, for the RNN and SNN models. The final results of the experiment are presented, in the form of the evaluation metrics described in Chapter 3.
- In Chapter 5, the experimental results are evaluated, and their significance is discussed. The classifiers are compared with one another, and the research hypotheses are evaluated based on the experiment results.
- In Chapter 6, the undertaken research project is summarised, and conclusions are drawn from the experiment results. The contribution of the research towards the field's advancement is examined, and directions for further academic work are suggested.

Chapter 2

Literature Review

This chapter consists of a review of the existing academic literature relevant to the classification of event-based data, with a specific focus on RNN and SNN classifier models. Firstly, the nature of temporal data is examined, and its unique characteristics are outlined. The academic context of SNNs and RNNs is reviewed using an overview of their precursor in the field of artificial intelligence, the ANN. The seminal literature on RNN architectures is examined, and the motivation for LSTM models is outlined. The academic state-of-the-art in SNNs is reviewed, with special attention being placed on specific spiking neuron models and learning rules. The literature on the biological inspirations behind SNNs is described only where it is relevant as motivation. The spike encoding methods for generating spiking datasets are also reviewed.

2.1 Data with Temporal Components

On a fundamental level, AI algorithms operate by learning to represent the intricate structures present in data, with the goal of using these learned representations for pattern detection or classification (LeCun, Bengio, & Hinton, 2015). As such, the choice of data used to train AI models is of central importance, with different data characteristics and properties leading to different model inference capabilities. This also applies to data that features a temporal component.

Temporal processes manifest themselves in nature very prominently. Phenomena

observed in reality tends to evolve its characteristics over periods of time, and in order to model and understand this, researchers need to make use of data that specifically incorporates temporality. Such temporal data is particularly useful due to the fact that interactions between humans and the environment are inherently temporal in nature (Peuquet, 1995). In the field of data science, there are two important paradigms for expressing the change of information in relation to the progression of time. These are the concepts of the time series and of event-driven data.

2.1.1 Time Series

A time series is any dataset where the times at which observations occur are tracked and included within the dataset itself (Brockwell & Davis, 2002). This can either be codified explicitly using a feature in the dataset representing time, or implicitly, as is in the case where the data is in the form of a sequence sampled at a certain rate. Time series can be either continuous-time, where the sample time can be any continuous point in time, or discrete-time, where the set of sample times is discrete (for instance, if the samples are taken every second).

The underlying structure of time series data can exhibit different distributions, such as linearity and stationarity. Linearity refers to data where the residuals are independent and identically distributed, (Berg, Paparoditis, & Politis, 2010), while stationarity describes processes where the statistical properties, such as mean and variance, do not vary over time (Blanco, Garcia, Quiroga, Romanelli, & Rosso, 1995). Many techniques for time series analysis rely on the assumption that the time series is either linear or stationary, or both. This can make the interpretation of their results problematic when they are applied to time series which do not follow these assumptions (Manuca & Savit, 1996).

2.1.2 EEG Time Series

EEG (electroencephalogram) datasets are one particular type of non-linear (Lo, Tsai, Lin, Lin, & Hsin, 2009) and non-stationary (Muñoz-Gutiérrez, Giraldo, Bueno-López,

& Molinas, 2018) time series. EEG signals are recorded from human subjects, and they reflect the dynamics of electrical activity in the brain which enables populations of neurons to work in a synchronous manner (Ernst Niedermeyer, 2005). Essentially, EEGs measure communications in networks of neurons.

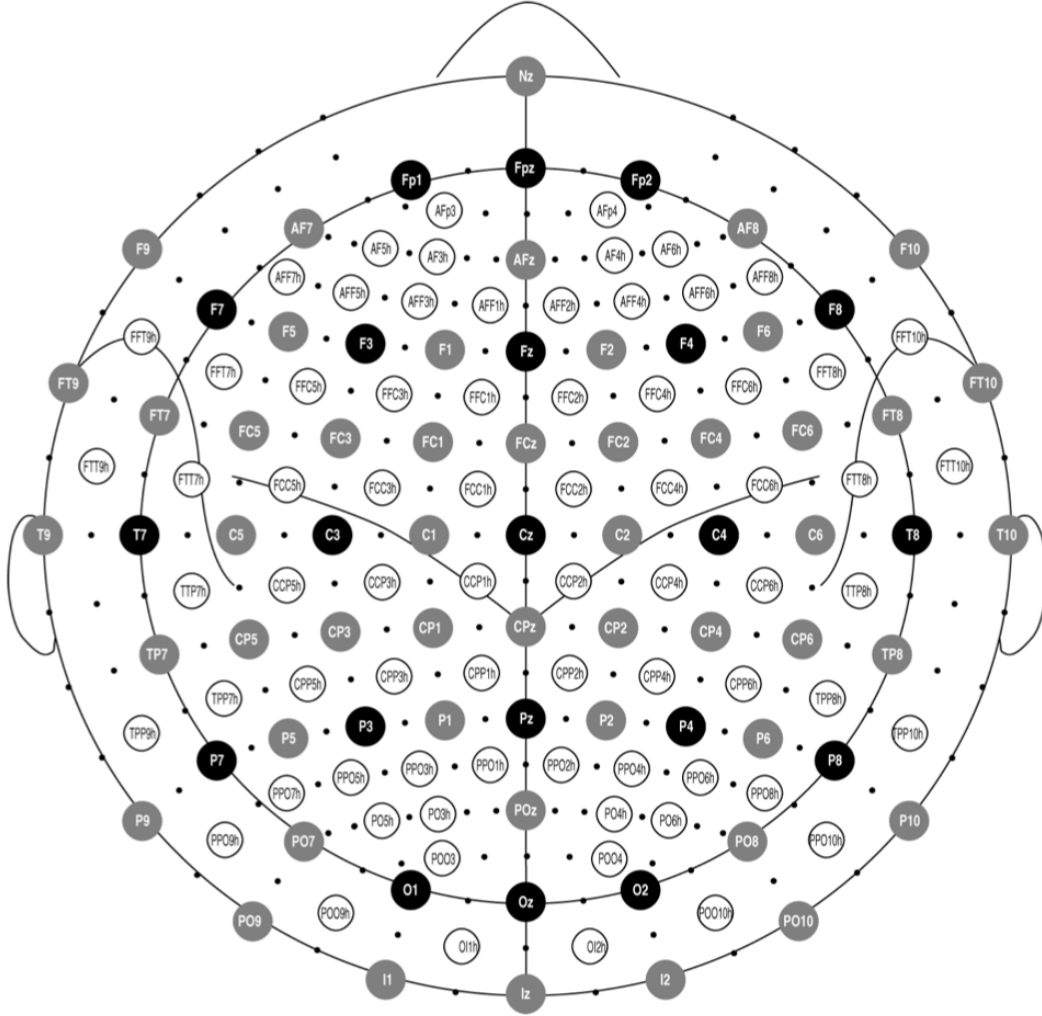
EEG measurements are taken using electrodes, attached to the scalp of the subject, with an electrode jelly applied between the electrode and the skin. The data gathered from a single electrode corresponds to a single EEG channel. There are several different electrode placement schemes in use in the EEG academic literature, but the most comprehensive of these is known as the 5%, or 10-5 system, in which the total number of electrode locations is around 345 (Oostenveld & Praamstra, 2001). This scheme is shown in figure 2.1. The 5% placement system does not require that researchers make EEG recordings at all of the listed electrode locations - it only defines the scalp locations and standard channel nomenclature to be used. For instance, the 3 datasets used in the experiment conducted in this research (outlined in section 3.2) feature different numbers of channels (specifically 64, 16 and 1). The creation of an EEG data requires that the EEG data collection takes place over a certain period of time, and that the EEG signal is sampled at a certain rate, measured in Hertz.

Certain patterns in EEG data correlate with the function of both healthy and unhealthy central nervous systems in humans (Ernst Niedermeyer, 2005). This has led to the widespread use of such data for the diagnoses of brain diseases, such as epilepsy, dementia, and neurological infections (Smith, 2005).

2.1.3 Event-Based Data

Another form of data that incorporates temporality is event-based data. The idea behind event-based data is that data acquisition occurs only when an “event” of some interest occurs, such as a notable change in a certain quantity being tracked (Tsividis, 2010). This is particularly useful in applications where energy and bandwidth resources are scarce.

Spike trains are an example of event-based data.



(Oostenveld & Praamstra, 2001)

Figure 2.1: Standard EEG electrode placements and channel names according to the 5% scheme.

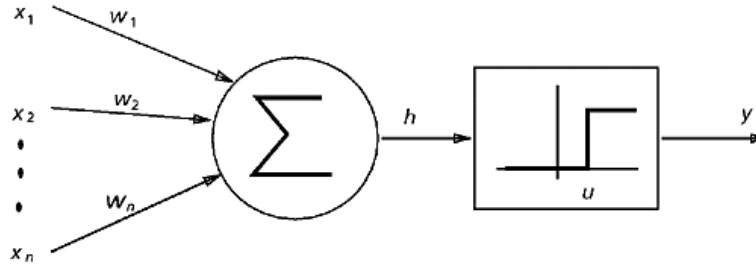
2.2 Artificial Neural Networks

Artificial Neural Networks (ANNs) are a field of Artificial Intelligence algorithms, the operation of which is inspired on an abstract level by the structures present in the human brain (Jain, Jianchang Mao, & Mohiuddin, 1996). This inspiration comes in the form of implementing computational models of biological neurons and axons, which are the connections between neurons.

One application of ANNs which is particularly relevant to this research is their

use in modelling non-stationary time series data. As outlined in section 2.1.2, this is a problem domain with which other methods of time series analysis encounter issues due to their underlying assumptions about data distributions. However, ANNs make no such assumptions, and have been demonstrated to be able to model non-stationary time series with high degrees of success (Kim, Oh, Kim, & Do, 2004).

2.2.1 ANN Neuron Model



(Jain et al., 1996)

Figure 2.2: The MP Computational Neuron

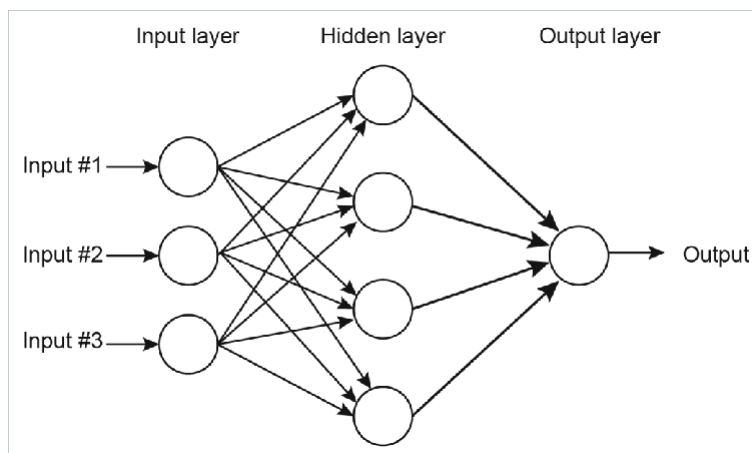
The basic computational neuron model in ANNs, known as the McCulloch and Pitts (MP) neuron (pictured in Figure 2.2), is essentially an input/output device (Hopfield, 1988). It works by calculating a weighted sum of its inputs, and applying an activation function on the result, often a simple sigmoid function, though the choice of activation function varies depending on the problem. Each input to the neuron is weighted using a connection weight. The process of learning, or 'training', in the ANN involves making changes to precisely these connection weights.

2.2.2 ANN Decision-Making

The activation function of the neuron is what gives it its decision-making properties. The use of a threshold-based activation function, such as the sigmoid function, allows the neuron to model non-linear decision frontiers, which is something that linear systems are not able to do (Jain et al., 1996). McCulloch and Pitts (1943) demonstrated that, in principle, if a number of MP neurons are connected in a certain manner, the

neurons' weights can be suitably optimised such that universal computation can be performed.

2.2.3 ANN Architecture



(Manning, Sleator, & Walsh, 2013)

Figure 2.3: A Multi-Layer Perceptron with One Hidden Layer

The typical architecture used in non-recurrent ANNs is known as a feed-forward network, where neurons are organised into layers and each neuron propagates its output into the ensuing layer's neurons. In the literature, these types of networks are also known as Multi-Layer Perceptrons (MLPs) (Rumelhart & McClelland, 1987). MLPs consist of a minimum of three layers of neurons: an input layer, at least one hidden layer, and an output layer. The hidden layers allow the network to create complex internal representations of the input, which is then propagated to the output layer. The classification/prediction results of the model are then determined from the output layer.

2.2.4 Learning in ANNs

ANNs are optimised by iteratively updating the connection weights between neurons so that the final output of the network is as close as possible to the structures observed

in the real data. This process is known as learning, or training. There exist a number of different categories of learning techniques used in ANNs, depending on what information is available to the network (Mohri, Rostamizadeh, & Talwalkar, 2012):

- **Supervised Learning:**

This category of learning algorithm makes use of not just the training data for learning, but also the target variable labels of the training data. This allows the ANN to calculate the error of its predicted labels compared to the true labels (known as the error signal) during learning, which can then be used to update the network's weights such that the error is reduced. Learning rules which operate on this principle are known as Error-Correction rules (Jain et al., 1996). This paradigm is the most common one used for classification problems.

- **Unsupervised Learning:**

In this case, the labels of the training data are not available to the network, and the ANN attempts to learn to recognise some kind of underlying pattern in the input data. The absence of the training data's true target labels during learning means that there is no way to use the model error during training. Examples of Unsupervised Learning problems include clustering and dimensionality reduction.

In addition to the above paradigms, ANNs can be trained using Semi-Supervised Learning, Reinforcement Learning, and Online Learning, among others (Mohri et al., 2012).

Backpropagation

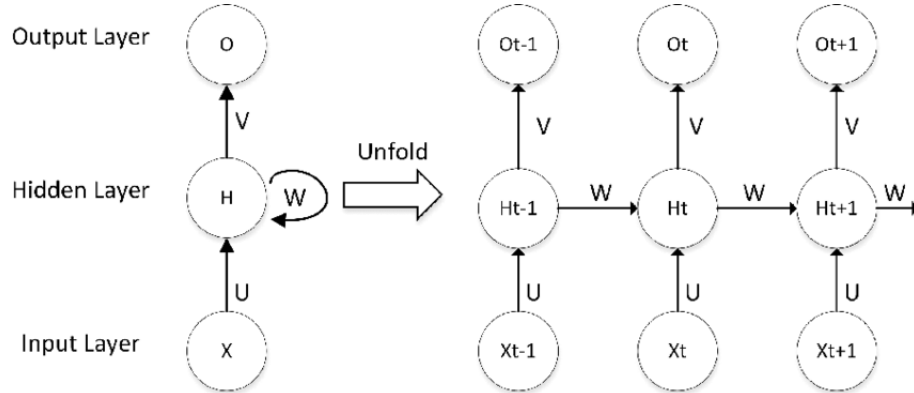
In the area of supervised learning, the Backpropagation algorithm is one of the most important developments leading to the widespread use of neural network models. Based on the Error-Correction principle, Backpropagation is a learning rule which uses the chain rule to find the derivative of the loss function with respect to the synaptic weights, and implements a Gradient Descent method for determining the optimum weights in a feedforward architecture (Rumelhart, Hinton, & Williams, 1988).

Network optimisation with Backpropation starts by computing the changes for the weights feeding into the output layer, and the process is repeated for each previous layer, such that the error is propagated backwards through the network.

2.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a subset of ANNs which differ from traditional feed-forward networks in that they employ a looping, or recurrent, mechanism. This allows them to operate in a temporal manner. Hidden neurons in RNNs aren't just connected to the hidden neurons in the next layer, but also recurrently to themselves, across all previous timesteps. These recurrent connections permit the hidden neurons to take into account information from their temporally previous output, which in turn shapes their subsequent behaviour.

The recurrent mechanism lets the RNN represent memory (Elman, 1990), which sets recurrent architectures apart from simple feed-forward ones, which are memory-less, meaning that their output is only a function of the current data, rather than of both current and previous samples (Ma & Principe, 2019).



(Z. Zhang et al., 2018)

Figure 2.4: A recurrent neuron, unrolled over time

RNNs' ability to process memories allow them to make use of the temporal context present in time-series data. As a result, RNN models trained on such data are of significant interest in fields that deal with elements that interact with each other

in space and time, therefore having temporal features. These fields include video-recognition (Le, Zou, Yeung, & Ng, 2011), robotics (Douillard, Fox, & Ramos, 2011), and self-driving vehicles (X. Zhang, Jiang, & Wang, 2014), all important problem areas in deep learning and fields where RNN applications are widespread.

In addition to this, said contexts can be of arbitrary length, due to RNNs' recurrent nature which allows information to repeatedly cycle inside the network (Mikolov, KarafiÅít, Burget, CernockÅœ, & Khudanpur, 2010). This makes them much more effective than feed-forward architectures at modelling behaviors which express themselves as temporal sequences, such as speech and language (Elman, 1990).

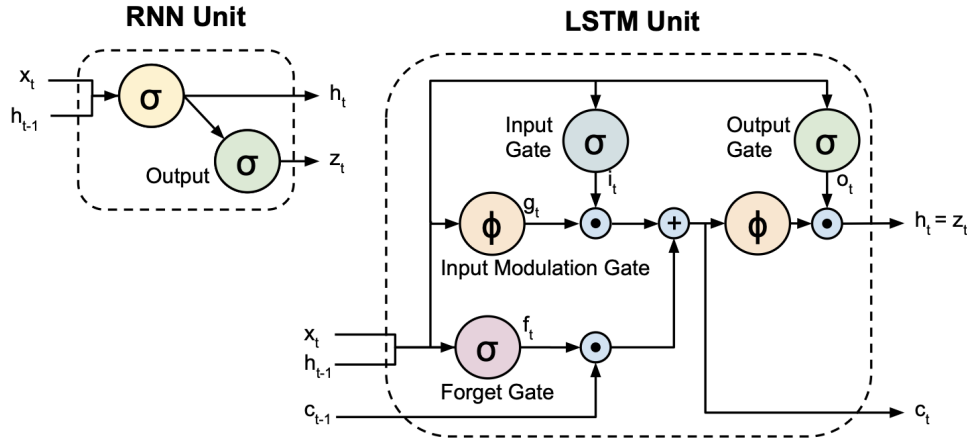
2.3.1 Learning in RNNs

The Backpropagation algorithm used for supervised learning with feed-forward neural networks can be adapted to train RNNs, with the Backpropagation Through Time (BPTT) algorithm. Derived from the classical Backpropagation algorithm (outlined in section 2.2.4), it also relies on the use of the chain rule to calculate and backpropagate the error gradient. In order to implement BPTT in RNNs, the network must first be unrolled over time, essentially leading to the creation of a feed-forward network for each timestep (Mozier, 1995). During optimisation, the gradient is backpropagated through each of these different networks, which means that as the number of timesteps increases, this method can become computationally expensive, and incur a very high memory requirement (Nowak et al., 2017). Additionally, each of the individual networks in the unrolled RNN contains internal dependencies to the network from the previous timestep, which introduces a bottleneck for GPU based training implementations (Hwang & Sung, 2015). This is because these internal dependencies limit the number of parallel operations that can be conducted using GPU hardware.

RNNs encounter a significant issue when they are applied to the learning of long-term temporal dependencies. Bengio, Simard, and Frasconi (1994) prove experimentally that learning these long-term dependencies with RNNs using gradient descent becomes increasingly inefficient as the temporal span of the dependencies becomes longer. In addition to this, the effect of noise on learning increases along with the

length of the dependencies. Also, RNNs encounter the 'vanishing gradient' problem, causing error signals propagated backwards in time to decay (Hochreiter, 1998), making long-term dependencies hard to learn with the BPTT algorithm due to insufficient weight changes.

2.3.2 LSTM



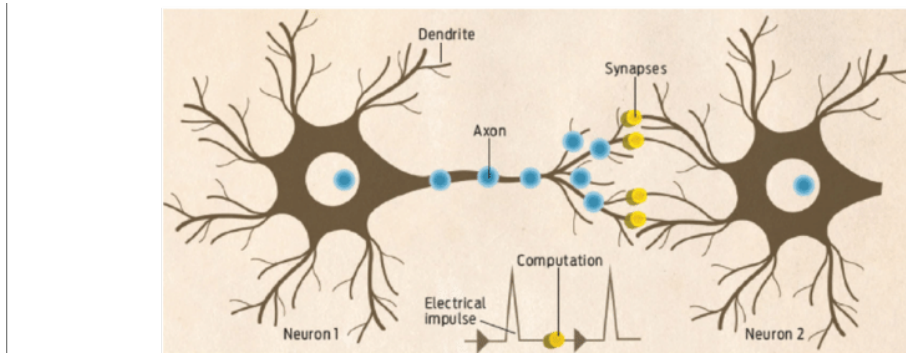
(Donahue et al., 2014)

Figure 2.5: RNN and LSTM cells

The vanishing gradient issue seen in RNNs can be alleviated by making use of the Long Short Term Memory (LSTM) network architecture (Hochreiter & Schmidhuber, 1997), which can capture long-term dependencies in excess of 1000 timesteps, while not diminishing the network's ability to model short-term structures. The LSTM architecture achieves this by keeping the error gradient flowing back through time constant. LSTM cells employ a hidden state, which tracks the internal memory contents of the cell. In addition to this, three different memory gates are used to control the information flow into and out of the CEC. These memory gates are the 'forget' (Gers, Schmidhuber, & Cummins, 2000), 'input' and 'output' gates.

2.4 Spiking Neural Networks

SNNs are a cutting-edge development in the field of machine learning, which are intended to more accurately imitate the biology of the human brain, as opposed to the abstract representation used by ANNs. The general motivation behind this approach is to capture the human brain’s capabilities for multi-tasking, learning with minimal supervision and generalizing learned skills, all done with high computational and energy efficiency (Poo, 2018).



(Wang, Lu, & Wen, 2017)

Figure 2.6: Biological neuron structure

In SNNs, information is not represented as matrices of scalars, as is the case with ANNs, but as sequences of binary events, or spikes. These spikes are inspired by the actions of Event-Related Potentials (ERPs) in the brain, which are voltages present in the brain in response to specific stimuli (Blackwood & Muir, 1990). Sequences of spikes over time are referred to as ‘spike trains’, and their temporal nature makes them a very powerful method of encoding information. In particular, the nature of spike trains makes them particularly efficient for representing temporal data. The advantage of using data encoded as spikes is that it can enable SNNs to process complex inputs faster than with scalar data (S. Thorpe, 1990).

Spike trains are not just an efficient method of encoding temporal information. The literature on neural information processing shows that the precise timing of ERPs, and their impact on the membrane potential, is an integral part of information processing in the brain (Bohte, 2004a). As such, significant research efforts have been made to

implement approaches mirroring the brain’s biological learning methods in SNNs.

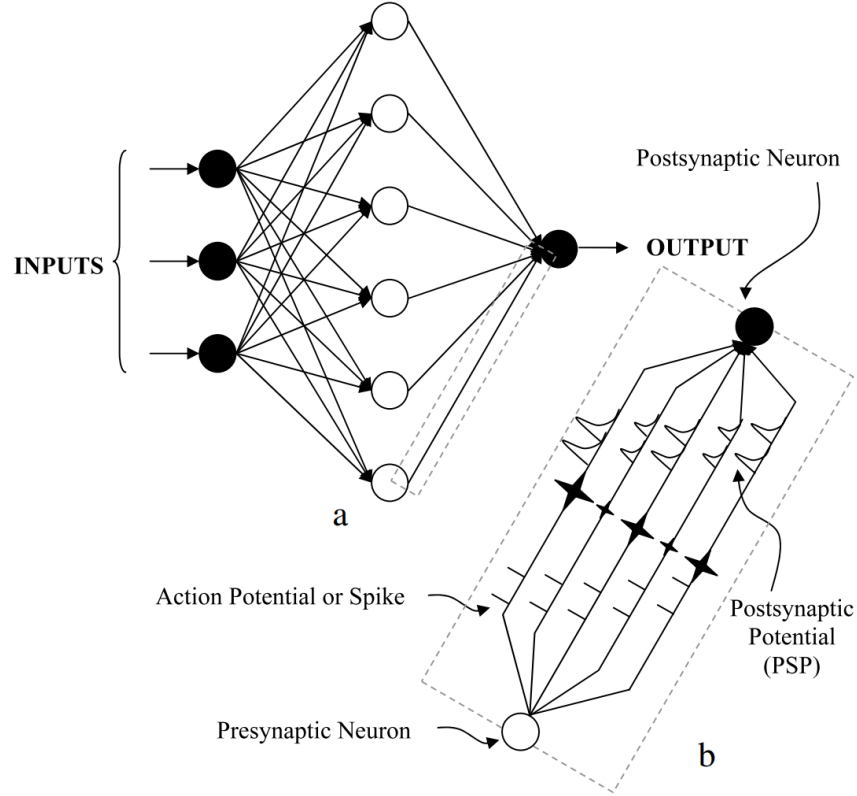
In addition to this, the event-based information processing in SNNs occurs on a massively parallel scale, allowing many neurons to communicate and process spike sequences simultaneously. This is similar to the phenomena observed in biological brains, (White & McDonald, 2002) where multiple, independent neural systems continually process information and influence behavior simultaneously and in parallel.

This event-based processing presents another advantage to using SNNs as opposed to traditional ANNs - while ANNs have to update the activation value of every neuron in the network at every timestep, SNNs only update activation values when a neuron ‘spikes’ (Neil et al., 2016), or when the ERP reaches a certain threshold. This sparse computation makes them more computationally and energy efficient than ANNs. Additionally, Maass (1996) demonstrate that individual spiking neurons have more processing power than the individual sigmoid neurons typically used in ANNs.

The literature on SNNs is extensive, and many different spiking neuron models have been proposed by researchers. These are outlined in section 2.4.3. Additionally, researchers have constructed SNN architectures which incorporate a variety of non-spiking layer types into the topology of the network. These hybrid models include spiking deep belief networks (O’Connor, Neil, Liu, Delbruck, & Pfeiffer, 2013), spiking convolutional networks (Cao, Chen, & Khosla, 2015), and reservoir-based spiking networks (Schliebs, Hamed, & Kasabov, 2011), as well as spiking recurrent neural networks (Buesing, Bill, Nessler, & Maass, 2011).

2.4.1 SNN Implementation Approaches

The modelling of SNNs provides a number of implementation challenges when compared to ANNs. Broadly speaking, ANNs attempt to find a solution to a problem where the decision frontier is non-linear, meaning that finding an analytical solution to the problem would be too complex. To counter this, ANNs use numerical simulations on digital computer hardware to find solutions to these problems (Schemmel, Grubl, Meier, & Mueller, 2006). Another approach to this involves implementing the neural network in analog, rather than digital hardware, on VLSI circuits, a concept



(Ghosh-Dastidar & Adeli, 2009)

Figure 2.7: A typical feedforward spiking neural network architecture, with a single hidden layer

known as neuromorphic computing (Mead, 1990). While this approach is not necessary for ANN implementation, it is currently the most feasible way to model SNNs, as having a physical neural network model allows for the assignment of physical quantities (such as voltage) to represent the physiological quantities which make biological neurons such effective information processing units (such as the neuron's spiking potential). Digital models of SNNs would have to simulate these physiologically-inspired quantities numerically, making them inefficient when compared to the physical model. As a result of this, there are two main approaches for implementing SNNs for practical applications - using on-chip training of SNNs with neuromorphic hardware, and using SNN simulators on conventional hardware.

Neuromorphic platforms for SNN implementation include Intel's Loihi chip (Davies

et al., 2018), IBM’s TrueNorth chip (Hsu, 2014), and SpinNNaker (Furber et al., 2014). However, access to this hardware was not available for this research, and as a result, an SNN simulator was used.

SNN Simulators

Due to the limited availability of neuromorphic hardware for research of SNNs, a number of SNN simulators have been developed by academics. Brette et al. (2008) provide a detailed review and comparison of 8 SNN simulation environments prevalent in the literature, though it does not cover the simulators that have been developed since 2007. The following list describes the capabilities and intended use of several of the most popular SNN simulators, including newer, cutting edge platforms.

1. GENESIS

GENESIS (the GEneral NEural SIMulation System) is a spiking neuron simulator written in C, and is the first such simulator presented in the literature (J. M. Bower & Beeman, 2012). Development in GENESIS is performed using its own proprietary scripting language. Its purpose is to provide a platform for conducting research into biological neural systems by constructing physiologically realistic models (J. Bower & Hale, 1991). As a result, conducting experiments in GENESIS may be excessively specific for researchers looking to model more simplified neuron models, such as LIF.

2. NEURON

NEURON is another platform designed primarily for biologically-stringent simulation of nerve cells (Hines & Carnevale, 2013). The problems for which it is most efficient are those involving the simulation of the neurophysiology of a small number or single neurons. As such, it is less appropriate for the domain of neural networks involving many neurons, and more so for neurobiology. NEURON does not make use of any particular mathematical model (described in section 2.4.3) for spiking neuron simulation, but rather deals with specific concepts at

the neuroscience level. The NEURON simulator has also been extended to use a Python interpreter (Hines, Davison, & Muller, 2009).

3. NEST

NEST is an open-source SNN simulator implemented in C++ (Gewaltig & Diesmann, 2007). It can be used only for simulation of homogeneous networks, where all of the neurons are of the same mathematical neuron model. Despite this, the NEST framework is very extensive, and supports the use of the LIF and HH neuron models, though only for unsupervised learning through STDP. While the framework originally required the use of a Typescript-derived SLI (simulation language interpreter) for development, a Python-based interface to NEST was later introduced, called PyNEST (Eppler, Helias, Muller, Diesmann, & Gewaltig, 2009).

4. Brian

The Brian simulator is the first SNN simulator written in pure Python (Goodman & Brette, 2008). Its primary aim is improved ease-of-use over other simulators available in the literature, as opposed to biological accuracy or simulation efficiency. The package developers intend that Brian is used mostly as a tool for learning and teaching computational modelling (Goodman & Brette, 2009). It supports the use of the LIF, HH and Izhikevich neuron models, and does not require that networks modelled with it are homogeneous. However, the current version of Brian does not provide any methods for supervised learning.

5. ANNarchy

ANNarchy is an open-source, flexible neuron simulator, written in C++, with a Python interface (Vitay, Dinkelbach, & Hamker, 2015). It also supports GPU training. It was originally developed for the simulation of exclusively rate-coded networks, where the precise spike timing is not considered during learning. However, it was later extended to permit simulation of precisely-timed spiking networks, as well as hybrid networks, using both rate-coded and precisely-timed

components. It is able to simulate LIF, HH, and Izhikevich neuron models, but it does not contain any supervised learning rules.

6. BindsNET

BindsNET is an open-source SNN simulation package written in C++, with a Python wrapper (Hazan et al., 2018b), and is one of the few SNN simulators designed for the express purpose of solving machine learning problems. Additionally, the package uses PyTorch for its matrix computations to make its simulations more efficient. One of the main advantages of BindsNET is that it contains a host of different learning rules and spiking neuron models already included in the package - it can be used for supervised, unsupervised, and reinforcement learning, using both Izhikevich and LIF neurons.

7. SpykeTorch

SpykeTorch is a simulation framework for SNNs written in Python. Its matrix calculations are conducted using PyTorch, which makes it highly efficient and able to use GPU hardware for training. However, the package is limited to network simulations with at most one spike per neuron (Mozafari, Ganjtabesh, Nowzari-Dalini, & Masquelier, 2019). It is also not able to accommodate supervised learning problems, as there are no such rules programmed in the package.

Due to the massively parallel nature of SNNs, computer hardware implementations of SNNs require vast amounts of electrical power and great lengths of time, which hinders scientifically-useful experimentation.

2.4.2 Spike Train Generation

As outlined in section 2.1.3, SNNs make use of sequences of binary events, known as spike trains. There are two main approaches to generating data in the form of spike trains. The first of these involves the use of event-driven neuromorphic sensors (Liu & Delbruck, 2010), which by design output data in the form of spike trains.

Neuromorphic sensors have been designed for a number of different applications, including computer vision (Costas-Santos, Serrano-Gotarredona, Serrano-Gotarredona, & Linares-Barranco, 2007) and audio processing (Wen & Boahen, 2009). The second method for generating spike train data is converting existing, non-spiking datasets made up of scalar data into spike trains.

Scalar data (such as image pixel data, EEG voltages, etc.) can be encoded as a spike train using a number of different approaches, which can be broadly categorised within two types. The first of these is rate-based coding, which is based on the concept that all of the information processed by the neuron can be represented simply using the neuron’s firing rate, as opposed to using the precise timing between spikes (Adrian & Zotterman, n.d.). This approach has been widely used in experiments on the operation of neurons, and has a biological basis in the firing patterns of motor sensory systems.

However, research conducted by Stein, Gossen, and Jones (2005) indicates that while simply encoding neural stimulus as a rate might be sufficient in motor systems, more sophisticated neural systems such as the somatosensory and visual systems (Gollisch & Meister, 2008) rely on the precise spiking times to encode information. This has led to the development of a second category of approaches to spike encoding, known as temporal, or spike-time coding. Bohte (2004b) also show that the precise timing, rather than the overall rate, of incoming spikes in animal brains encodes the majority of information.

With respect to SNNs, the literature supports the use of temporal coding rather than rate-based coding. Rate-based encoding models are more prevalent in the field of indirect supervised training of SNNs (Mostafa, 2018), but since this approach involves the training of ANNs using scalar data, it is not appropriate for processing event-based data. On top of this, spiking models which utilise rate-coded data for information are inefficient. In direct SNN training approaches, temporal coding is more appropriate, as it allows for more sparse and information-dense neuron firing (Mostafa, 2018). The literature on learning rules for SNNs also focuses nearly exclusively on the use of temporal spike-trains for information processing, with all of the learning rules discussed in section 2.4.4 relying on the temporal encoding scheme.

Spike Encoding Strategies

There are several different methods available for encoding scalar data as spike trains. These include:

- Rank Order Coding (S. Thorpe & Gautrais, 1998)
- Time to First Spike (Tuckwell & Wan, 2005)
- Threshold-Based (N. Kasabov, 2016)
- Step-Forward (SW) (N. Kasabov, 2016)
- Moving-Window (MW) (N. Kasabov, 2016)
- Ben’s Spiker Algorithm (Schrauwen & Van Campenhout, 2003)

Each of these encoding methods can be viewed as a method of data compression, and each of them exhibits different characteristics of lossiness. In addition to this, spike encoders often vary in the types of spike signals they are able to encode. One important distinction in spike signals relevant to this is whether the signal is unipolar or bipolar - in the case of EEG signals, it is unipolar. Petro, Kasabov, and Kiss (2019) conduct a review of several spike encoding algorithms and determine that the most effective method for encoding unipolar signals is BSA (Schrauwen & Van Campenhout, 2003).

2.4.3 Spiking Neuron Models

There exist in the literature a number of mathematical models for describing the actions of spiking neurons, which can be placed on a varying scale of biological accuracy when compared to the behaviour of physical neurons. All mathematical spiking neuron models emulate biological neurons in that they activate (ie. spike) only when a certain neuron action potential exceeds a threshold, though different models achieve this with differing levels of detail (Pfeiffer & Pfeil, 2018), and selecting the appropriate model often involves a trade-off between biological accuracy and computationally efficiency.

In this subsection, the three spiking neuron models which are most prevalent in the SNN literature are described. While there are many more varieties of neuron models, these three represent the most seminal and influential approaches.

Hodgkin-Huxley

The first of these is the Hodgkin-Huxley (HH) model (Hodgkin & Huxley, 1952), which explains the firing activity of a giant squid axon. The spiking activity in HH neurons is determined by the contributions of several ionic currents to the action potential, which are described by 4 non-linear differential equations (Kistler, Gerstner, & van Hemmen, 2000). While highly biologically accurate, HH is a complex model to implement, as it requires the use of 16 parameters corresponding to membrane capacitance, ionic conductivity, potentials and voltages of the currents involved (Meunier & Segev, 2002). As a result, more abstract neuron models tend to be used by researchers for large-scale spiking neural network studies (Herz, Gollisch, Machens, & Jaeger, 2006).

Leaky-Integrate-and-Fire

A model of spiking neuron that is highly prevalent in the literature (Burkitt, 2006) is the Leaky-Integrate-and-Fire (LIF) model. Unlike the HH model, the LIF model makes use of a single linear differential equation to model the evolution of the neuron's action potential (Gerstner, Kistler, Naud, & Paninski, 2014a). LIF neurons exhibit a vital behaviour present in biological neurons - time-dependent memory. This is due to the 'leaky' component of the model, which allows the action potential of the neuron to leak out during the time intervals between spikes (Dutta, Kumar, Shukla, Mohapatra, & Ganguly, 2017). While this 'leaky' characteristic is also present in the HH model (Gerstner, Kistler, Naud, & Paninski, 2014b), the LIF model has a significantly lower computational cost, requiring only 5 floating-point calculations per 1ms of neuron simulation compared to HH's 1200 calculations.

Additionally, there are several more sophisticated spiking neuron models that build on the basic LIF model described above. This includes the LIF model with Adaptive Spiking Thresholds, (Diehl & Cook, 2015), which increase the model's biological plau-

sibility by introducing a mechanism for limiting neurons' spike rates. Essentially, LIF neurons with Adaptive Spiking Thresholds cause a neuron's spiking threshold to increase after every spike. If a spike does not occur, the spiking threshold decays over time back to its original value. A review of the spiking neuron models which add to the standard LIF is conducted in Feng (2001).

Izhikevich

Another significant spiking neuron approach is the Izhikevich model (Izhikevich, 2003), which represents a compromise between the biological accuracy of the HH model, and the computation efficiency of the LIF model. Though the Izhikevich model consists of just two equations and has only one nonlinear term, Izhikevich (2003) consider it a 'canonical' neural model, as with the correct parameter optimisation, it is able to mirror the performance of even the most complex, biologically accurate spiking neuron models, such as that of HH.

Izhikevich (2004) review 20 features prominent in biological spiking neurons which contribute to the overall complexity of their spiking behaviour, with the aim of comparing the comprehensiveness of existing spiking neuron models by examining which features they are able to represent. From the 11 different spiking neuron models examined, only the Izhikevich model excels in both biological plausibility and computational efficiency, with the authors being able to experimentally reproduce all 20 biological features. For other biologically accurate but less computationally efficient models, such as the HH model, Izhikevich (2004) are unable to determine experimentally if they exhibit all 20 biological features, as the authors failed to find the necessary parameters for this purpose within a reasonable period of time.

(Diehl & Cook, 2015) that its ability to replicate biological neuron features does
The chosen spiking neuron

2.4.4 Learning Rules in SNNs

In order to perform optimisation of SNNs, specific learning rules are used. These rules dictate how the weights of the spiking neurons are adjusted, which is identical to the

role player by learning rules in ANNs. Learning rules can be split into a number of categories, namely supervised, unsupervised, semi-supervised and reinforcement learning.

Unsupervised Learning

The Rank Order learning rule, first described by S. J. Thorpe, Delorme, and van Rullen (2001), assumes that earlier spikes carry more information, therefore only considering the order of spike arrival when calculating weight changes. The rank order learning rule is motivated by the idea that animal brains' ability to rapidly process sensory inputs is due to earlier input spikes carrying more information than later ones (S. Thorpe, Fize, & Marlot, 1996). This concept is further supported by VanRullen, Guyonneau, and Thorpe (2005), who underline the critical importance of first-spike time encoded data in human sensory systems at the population level.

Another prominent learning rule is spike-timing dependent plasticity (STDP). In the SNN literature, this learning rule receives a large amount of attention, as it has been proven to mirror the biological processes used by animal brains for learning and memory (Caporale & Dan, 2008). It also requires a spiking architecture for implementation, and cannot be used for training ANNs. As such, research on STDP has driven a lot of development in the field of spiking neuron models and networks.

STDP allows connected neurons to learn consecutive temporal associations from data, forming chains of connections to represent patterns in the data.

Supervised Learning

In supervised learning, an artificial intelligence model is optimised by reducing the error between the observed labels in the training data and the model's predicted labels. Artificial Neural Networks achieve this using the Backpropagation algorithm (Rumelhart et al., 1988) which involves propagating the derivative of the loss function backwards through the network. However, SNNs encounter an issue with the implementation of Backpropagation, being that SNNs use spike trains for information processing, which are non-differential as they are a sequence of discrete events, repre-

sented by a sum of Dirac delta functions (Tavanaei et al., 2019). Therefore, methods for SNN supervised learning utilise approximations of the derivative.

1. SpikeProp

The SpikeProp learning rule (Bohte, Kok, & Poutré, 2001) is able to circumvent the problem of non-differentiable spike trains by approximating the neurons' spiking thresholds as a function, which allows the partial derivative of the relationship between the input data and the resultant spiking time to be found. However, the original SpikeProp learning rule as presented by Bohte et al. (2001) is only able to optimise networks in which the neurons are constrained to emit just a single spike. Ghosh-Dastidar and Adeli (2009) succeed in adapting the SpikeProp algorithm for use with SNNs that exhibit multiple spikes per neuron. This learning rule is dubbed Multi-SpikeProp.

2. ReSuMe

ReSuMe is a supervised learning rule for SNNs Ponulak and Kasiński (2009) which is based on the Widrow-Hoff algorithm, an optimisation method originally developed for non-spiking neurons (Widrow & Hoff, 1988). The ReSuMe algorithm is able to minimise the error between the target and output spike trains without having to explicitly calculate the gradient, and unlike the original SpikeProp algorithm, can process more than a single spike per neuron.

3. SPAN

The SPAN (spike pattern association neuron) learning rule is different from other supervised rules in that it converts the SNN's spike trains into analogue signals before optimisation takes place. Like ReSuMe, it also based on the Widrow-Hoff learning rule, but SPAN allows each neuron in the network to learn and memorise more information. N. K. Kasabov (2019) demonstrates this experimentally by comparing the classification accuracy of ReSuMe networks and SPAN networks with equal numbers of spiking neurons, concluding that ReSuMe has less memory capacity than SPAN.

4. Clamping

In addition to the supervised learning rules above, the unsupervised STDP learning rule can be modified to perform unsupervised learning tasks. This is done by 'clamping' certain neurons to particular labels (Hazan et al., 2018a). Spiking neurons in the hidden layer of the network are divided into groups, one group for each class label present in the dataset, with each group containing the same number of spiking neurons. During training, the class label of the current sample is recorded and a random neuron in the corresponding group is selected. The chosen hidden layer neuron is then induced to spike at every time step in the presented sample sequence. This lets the neuron's weights learn the features of the input sample.

5. Indirect Training

Another alternative to supervised training of SNNs is indirect supervised training. This involves the training of conventional ANNs using scalar data, and then transferring the learned model weights to an SNN with a specialised conversion algorithm. While these models tend to achieve state-of-the-art classification performance in the field of SNNs (Diehl et al., 2015), they are inefficient learners, and encounter limited effectiveness in certain problem domains. Pfeiffer and Pfeil (2018) detail the reasons behind these limitations. Firstly, only a subset of artificial neuron activation functions can be converted to SNN - the sigmoid activation function is an example of one that cannot. In the case of converting CNNs, pooling operations are difficult to implement in SNNs. Additionally, converted SNN models tend to require significantly more spikes than directly-trained SNNs to achieve the same inference performance.

6. Teacher Forcing

A novel supervised learning paradigm for spiking neural networks, Teacher Forcing (also known as Forced Teaching), allows the unsupervised STDP learning rule to be adapted to perform unsupervised learning tasks (Legenstein, Naeger, & Maass, 2005). Teacher Forcing is different from other supervised learning

techniques in that it is not error-based, that is, the error between the target and predicted spike trains is not used during training. Essentially, Teacher Forcing works by 'clamping' the output of the spiking neuron to the desired target, which forces the synaptic weights of the neuron to converge to the target weights. Legenstein et al. (2005) prove that spiking networks of LIF neurons with Adaptive Thresholds are able to learn any arbitrary transformation between input and output spike trains when using this learning paradigm, making it an effective supervised learning rule. This technique is inspired by a similar learning rule used in reservoir computing (Paugam-Moisy, Martinez, & Bengio, 2008). Unlike other techniques, the target behavior does not have to be specified as a closed form differential equation for training. There is no batch learning implementation for Teacher Forcing, and as such training with this approach uses with a batch size of 1.

2.5 Summary, Limitations and Gaps in Literature Review

The literature review in this study covered a number of topics relevant to the research problem being investigated. These included outlines of the type of data being studied, as well as AI methods such as ANNs, recurrent ANNs and SNNs for information processing of time series data.

However, there are a number of gaps present in the literature relevant to this research, such as in the area of spike encoding. While a number of effective encoding strategies have been proposed, such as rank-order encoding, and HSA encoding, representing data as temporal spike-trains tends to be a lossy process (Sengupta & Kasabov, 2017). In fact, there are currently no lossless encoding techniques for spike trains. This can be considered a significant gap in the literature on SNNs.

Even though there are many theoretical advantages to the use of SNNs over RNNs for time series classification, there are no comparative studies in the literature that prove or disprove the existence of these advantages experimentally. This is an impor-

tant gap in the literature on SNNs that this research aims to fill.

Chapter 3

Design and Methodology

This chapter describes the experiments used to generate the results from which a conclusion to the research question was reached. The experiments conducted are described, and the data, software and tools used are motivated and outlined.

3.1 Data Understanding

The research utilised a total of three EEG datasets. Each of these had a temporal component, and also had binary target variables.

3.1.1 EEG recordings of healthy adolescents and adolescents with symptoms of schizophrenia

The first of these is an EEG dataset from Moscow University (Gorbachevskaya & Borisov, n.d.). It contains EEG time series data from 84 subjects, with 16 features, one for each of the 16 EEG channels recorded. Each observation represents 1 minute of recorded EEG amplitude (mkV), and since the sampling rate is 128 Hz, there are 7680 samples for each observation. The target variable represents whether or not the subject exhibits symptoms of schizophrenia. The class balance in the target variable is 46.43% healthy subjects and 53.57% subjects with schizophrenia. This data will be referred to as the 'Schizophrenia Dataset' for conciseness.

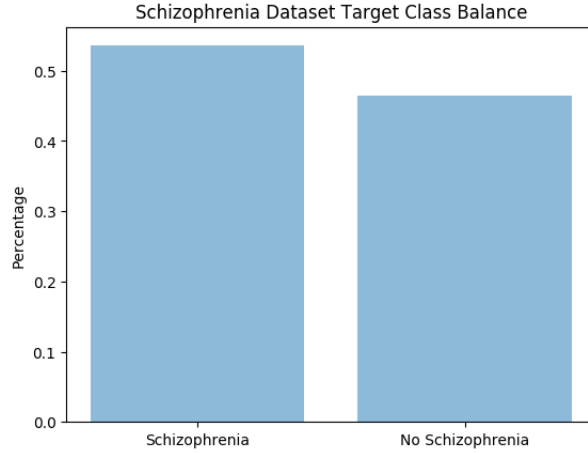


Figure 3.1: Schizophrenia Dataset Target Class Proportions

3.1.2 EEG recordings of alcoholics and control subjects

The second dataset utilised is from a study by the Neurodynamics Laboratory of the State University of New York Health Center (Lichman, n.d.), which investigates whether genetic predisposition to alcoholism can be predicted from EEG data. This EEG is recorded using 64 channels. The length of each trial is 1 second of EEG amplitude, sampled at 256 Hz, with 256 samples for each trial. The target variable is binary and shows if the recorded EEG is from an alcoholic subject or a control subject. The class balance in the target variable is 20% healthy subjects and 80% alcoholic subjects. Unlike the other 2 EEG datasets used in this study, this dataset has predetermined training and test sets, each containing 100 observations. This data will be referred to as the 'Alcoholism Dataset' for conciseness.

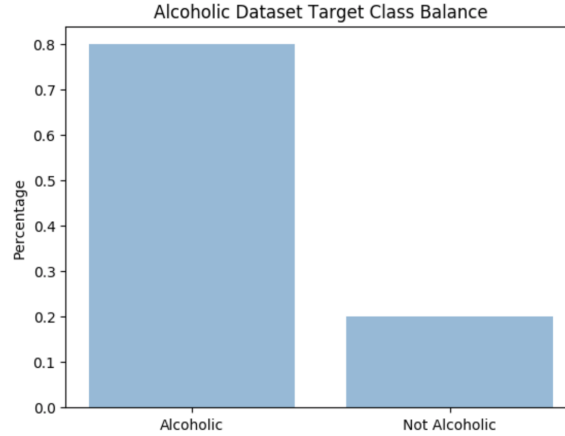


Figure 3.2: Alcoholism Dataset Target Class Proportions

3.1.3 EEG of healthy subjects and subjects with epilepsy

The third and final dataset used is from the University of Bonn (Andrzejak et al., 2002), which records single-channel EEG information from subjects with and without epilepsy. It consists of EEG time series data from 500 subjects, with readings lasting 23.5 seconds and containing 4097 data points. The original dataset also has 5 possible categories, with 1 category being used to identify subjects with epilepsy, while the other 4 all represent subjects without epilepsy. The class balance in the target variable is 20% subjects with epilepsy, and 80% subjects without epilepsy. This data will be referred to as the 'Epilepsy Dataset' for conciseness.

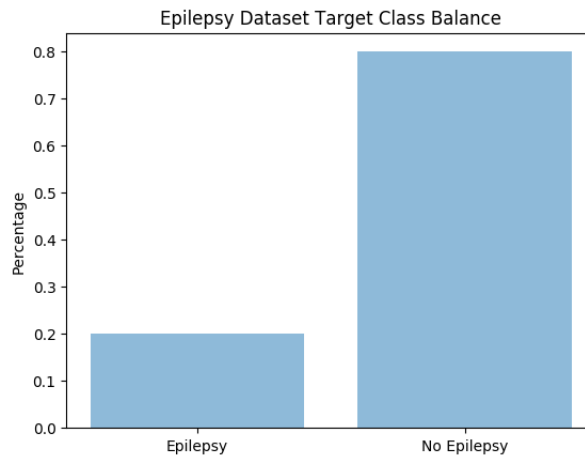


Figure 3.3: Epilepsy Dataset Target Class Proportions

3.2 Data Preparation

3.2.1 Data Encoding

No data encoding is necessary for training the RNN models, since all of the datasets used in this research (described in Section 3.2) are already in a time series format. However, Spiking Neural Networks utilise spike trains, which are sequences of neuron firing times, as opposed to time series of real-valued data. For this reason, the three datasets first were converted to spike trains before the Spiking Neural Network models could be trained on the data.

As outlined in Section 2.4.2, the optimal method for encoding unipolar signals as spike trains is BSA (Ben’s Spiker Algorithm). EEGs are unipolar, as it is impossible to record a negative EEG voltage. Since the datasets in this research that need to be encoded are all EEG time series, and these are by definition unipolar signals, the BSA encoding algorithm was selected.

3.2.2 Dataset Construction

All of the datasets were constructed to have this common shape:

([Number of Observations], [Number of Dimensions], [Sequence Length])

The Schizophrenia Dataset is received in the form of 2 data archives, containing 39 and 45 text files respectively. Each text file contains the EEG data of a single subject, with the first data archive representing healthy subjects and the second represent subjects with schizophrenia. Since the EEG time series is of length 7680, and amplitude was measured at 16 different positions on the scalp, each subject’s EEG text file contains 122880 amplitude values, as a sequence of 16 time series, one for each EEG channel. The text files’ contents were converted from single arrays into two-dimensional data matrices, and concatenated together to form the full dataset, leading to a final data matrix of shape (84, 16, 7680).

The Alcoholism Dataset consists of a predetermined training and test set. Both the training and test sets contain 10 subjects each, with each subject having 10 recorded

EEG trials Each trial file has 4 columns, representing the trial number, sensor position, sample number and sensor value. In total, each file contains 256 samples for each of the 64 different sensor channels. The file contents were grouped by the sensor position and converted into two-dimensional data matrices. Whether the trial was conducted by an alcoholic or control subject was determined by the presence of either an 'a' or 'c' as the fourth character in the filename, and this was used to construct the array of target labels. The final data matrix was of the shape (200, 64, 256).

The Epilepsy Dataset originally consisted of 5 sets (A, B, C, D, and E) of 100 files each, one file for each subject. The folders each represent subjects that belong to a certain class in the target variable. However, according to Andrzejak et al. (2002), only subjects that belonged to class E experienced epileptic seizures. As such, the dataset's target variable was re-coded to be binary, with classes 2, 3, 4, and 5 all becoming class 0. Each of the 100 files in each set contained just a time series of length 4096. Unlike the other datasets used in this research, the Epilepsy Dataset only has a single sensor channel. As such, the only dataset construction necessary was to concatenate the contents of each text file into a single data matrix of shape (500, 1, 4096).

3.2.3 Dataset Balancing

In each of the three datasets, the target variable was not balanced equally between positive and negative samples, though to differing degrees. This can cause problems for models' abilities to learn data structures, as having a target class be a minority means that the model will have a lower recall (outlined in section 3.6) when predicting this class. To counteract this, for each dataset, the training sets were balanced using simple undersampling of the majority target class. This was not done for the test sets.

3.2.4 Data Splitting

The datasets were split into training sets and test sets, to validate the model's performance on data that was not used for learning. The splitting was conducted using

stratified 10-fold cross-validation, ensuring that each split contained the same number of positive and negative target values. The final evaluation metrics for each model were calculated by averaging each individual fold’s evaluation metrics. This is especially important for the Schizophrenia and Epilepsy datasets, which have relatively few observations.

3.3 Software

The choice of software to use for the experiment was influenced to a high degree by the nature of the experiment.

1. SNN Simulator

As discussed in Section 2.4.1, it was determined that an SNN simulator framework would be necessary in order to construct the SNN classification models. The choice of which simulator to use is constrained by the nature of the experiment. Since the classification of EEG datasets is a supervised learning problem, the primary requirement for the simulator is the availability of supervised learning methods. The only such package from those outlined in Section 2.4.1 is BindsNET, which utilises a technique known as ‘clamping’ to adapt the unsupervised STDP learning rule into a supervised rule. In addition to this, BindsNET is unlimited in the quantity of neurons it can simulate, and supports the use of numerous different spiking neuron models, in a non-homogeneous network structure. As a result, the BindsNET simulator is used for this experiment.

2. Statistical Programming Language

For conducting data preprocessing and model construction, a statistical programming language would need to be used. Since the SNN simulator that would be used for the experiment, BindsNET, is Python-based, and due to the availability of a number of advanced deep learning libraries, Python was selected as the statistical programming language for the experiment.

3. RNN Model Framework

In order to construct the RNN models against which the SNN model would be compared, a deep learning library would need to be used. This library would have to support sequential modelling. Since Python was selected as the experiment’s statistical programming language, the choice of deep learning library was constrained to Python-based libraries. Keras was selected for this purpose, due to it being a high-level deep learning framework, leading to faster prototyping time.

3.4 Modelling

This section outlines the model architectures, layer topologies, and learning rules employed in both the RNN and SNN classification models. Additionally, the specific hyperparameter optimisation process in each case was outlined.

3.4.1 SNN Model Architecture

Layer Topology

The architecture of the SNN model was inspired by a number of state-of-the-art classification models present in the literature.

The number of hidden layers in the SNN is an important architectural decision that will influence the classification performance of the model. Even though it is standard practice in ANN architectures, in the SNN literature, implementations do not tend to utilise more than a single hidden layer. Ghosh-Dastidar and Adeli (2009) achieve a classification accuracy of 94.8% on the Epilepsy Dataset (Andrzejak et al., 2002) using one spiking hidden layer with 8 neurons, though unlike the experiment conducted in this research, the classification was between a total of 3 target classes, rather than 2. Kulkarni and Rajendran (2018) also use an SNN with a single hidden layer and 8112 spiking neurons to achieve 98.17% accuracy on the MNIST test database. Even in the unsupervised learning literature, multi-layer architectures are infrequent, with classification research tending to use a single layer with a high number of neurons

(Diehl & Cook, 2015) (Tavanaei & Maida, 2015). Research that investigates the classification performance of multi-layer SNNs includes Sporea and Grüning (2013), who use 9 hidden spiking layers, each with 10 hidden neurons. The biggest disadvantage of multi-layer SNNs comes from their optimisation. Each layer of spiking neurons requires that its hyperparameters are tuned to their optimal values. However, the parameters' optimal values are different for each hidden layer (Kheradpisheh, Ganjtabesh, Thorpe, & Masquelier, 2018), and as a result, hyperparameter optimisation for multi-layer spiking neurons is very computationally expensive. For the experiments conducted in this research, only 1 hidden layer is used, as the literature states that it is sufficient for classification model performance.

Spiking Neuron Model

The spiking neuron model used is another important element of the network architecture. The most prevalent spiking neuron models in the literature are outlined in detail in section 2.4.3. The choice of which spiking neuron model to use in the current research was limited to computationally efficient models, as the experiments relied on the simulation of a spiking network containing hundreds of neurons. As a result, the computationally expensive Hodgkin-Huxley neuron model was not considered for use. However, despite the Izhikevich model's superiority in biological accuracy over other efficient neuron types (Izhikevich, 2004), Jolivet, Rauch, and Gerstner (2005) demonstrate that LIF neurons with Adaptive Thresholds are sufficiently accurate simulators of neurons in the brain, with the threshold spiking voltage of LIF neurons differing from recorded thresholds in biological neurons by a few millivolts only. Similarly, the statistical structure of LIF spike outputs corresponds to that of the real neuron. In addition to this, Valadez-Godínez, Sossa, and Santiago-Montero (2020) compare the simulation capacities of LIF, Izhikevich and HH models, and determine that the conclusion reached by Izhikevich (2003) that the Izhikevich neuron model is canonical had been accepted erroneously. As a result, LIF neurons with Adaptive Thresholds are used for the hidden layer neurons. The chosen SNN simulation package, BindsNET, offers two different implementations of this neuron type - standard

LIF neurons with Adaptive Thresholds, and neurons replicating the model used in Diehl and Cook (2015). The latter are simply LIF neurons with Adaptive Thresholds that are limited to spike only once per timestep. Neurons emitting more than a single spiking per timestep can be caused if there are any input layer weights that are higher than the postsynaptic neuron threshold. Preliminary results from using both the regular Adaptive Threshold LIF neurons and those used by Diehl and Cook (2015) suggest that the latter achieve a better overall model classification performance. The Diehl and Cook (2015) neuron model is used in this research to form the first part of the SNN’s hidden layer.

Lateral Inhibition

While having a single layer of many Adaptive Threshold LIF neurons can be an effective architecture for classification using SNNs, there is another element inspired by biological neuron systems that has not been included at this stage, which is a mechanism for Lateral Inhibition. Lateral Inhibition is a concept in neurobiology which has been proven to be a vital dampener of spiking activity in biological neuron systems (Tomita, 1958). Nabet (2018) describe the function of Lateral Inhibition as a way of sharpening images in visual neuron systems. In the context of SNNs, Lateral Inhibition works by completely inhibiting the spike activity of a spiking neuron’s surrounding neurons within a specific distance. This helps the SNN learn more diverse features (Mozafari et al., 2019). Lateral Inhibition can be achieved in SNNs by simply adding an inhibitory layer of regular LIF neurons after the previous, excitatory layer of Adaptive Threshold LIF neurons, with both layers containing the same number of neurons. The connection between the excitatory and inhibitory layers is recurrent, and the weights in the inhibitory layer are negative. Despite the fact that there are now two layers in the network, their activity is synergistic, and they are both constrained to have the same number of neurons. As such, they can be considered as forming a single hidden layer.

SNN Learning Rule

The final decision made about the SNN model’s architecture is what learning rule would be used to train the network. The SNN supervised learning rules available in academic literature are outlined in detail in section 2.4.4. The choice of which learning rule to use was influenced primarily by what is available in the BindsNET package, which was selected as the simulation framework for conducting the experiment in section 3.4. Supervised Learning with BindsNET uses the Teacher Forcing technique, a non-error based learning paradigm. BindsNET’s implementation of Teacher Forcing works by modifying the unsupervised STDP learning rule to perform unsupervised learning tasks. This is done by ‘clamping’ certain neurons to particular labels (Hazan et al., 2018a). Spiking neurons in the hidden layer of the network are divided into groups, one group for each class label present in the dataset, with each group containing the same number of spiking neurons. During training, the class label of the current sample is recorded and a random neuron in the corresponding group is selected. The chosen hidden layer neuron is then induced to spike at every time step in the presented sample sequence. This lets the neuron’s weights learn the underlying features present in the input sample.

The SNN model architecture resulting from these decisions is a spiking network with a single hidden layer composed of N excitatory LIF neurons with Adaptive Thresholds, which use the Teacher Forcing technique as their learning rule, and N inhibitory LIF neurons, which are connected to the excitatory layer in a recurrent manner. Since the Teacher Forcing technique is being used for learning, the batch size is 1.

Hyperparameter Tuning

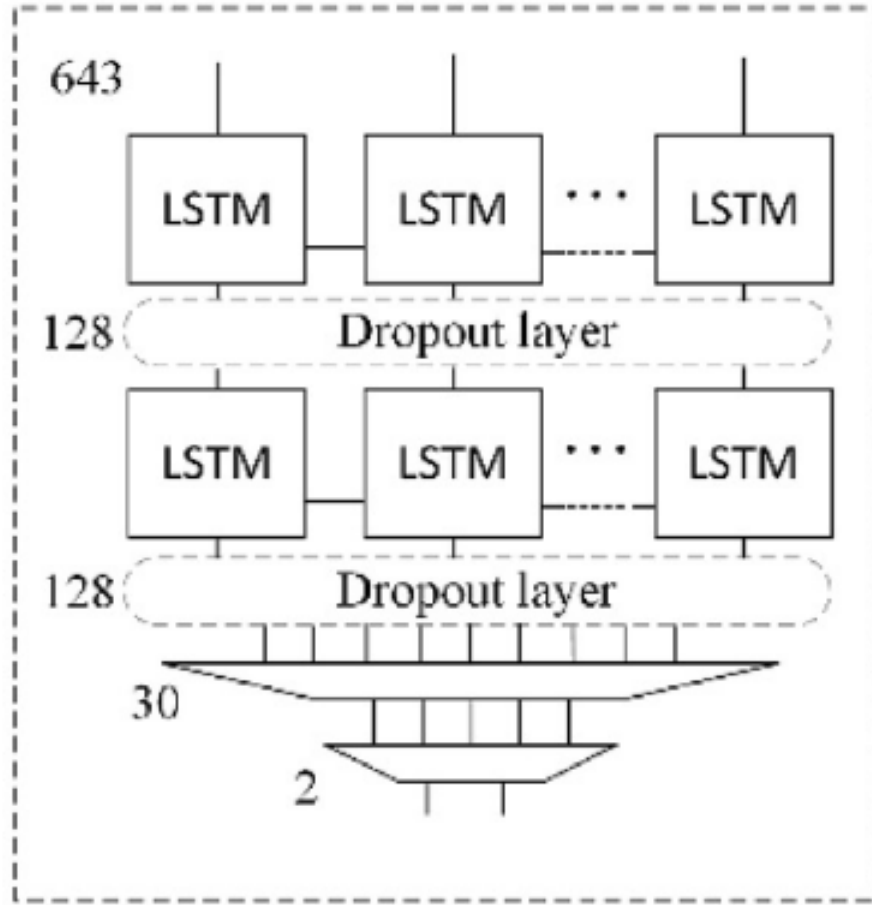
SNNs use several hyperparameters which affect how the model behaves and learns. In order to ensure that the SNN model exhibits optimal classification performance, a grid search is conducted to find the best values for these hyperparameters. This is referred to as Hyperparameter Tuning.

- Number of excitatory, inhibitory neurons
- Strength of synapse weights from excitatory to inhibitory layer
- Strength of synapse weights from inhibitory to excitatory layer
- Pre-synaptic events learning rate
- Post-synaptic events learning rate
- Input to excitatory layer connection weights normalization constant
- Batch size

3.4.2 Baseline RNN Model Architecture

Layer Topology

The layer topology of the baseline RNN model that would be used for this research was determined by examining the literature on EEG classification using RNNs with LSTM cells. Craik, He, and Contreras-Vidal (2019) conduct an extensive academic review of RNN-based approaches to EEG classification, and note that all such implementations use 2 LSTM layers. Specific research includes Kumar, Sharma, and Tsunoda (2019), who use 2 LSTM layers, with a maximum of 200 hidden units in each, the reasoning behind which is to reduce the computational complexity of the model. Tsiouris et al. (2018) also use a similar architecture for emotion recognition based on EEG, with 2 LSTM layers, and a final Dense layer for classification. The baseline RNN model used in this research uses the same layer topology, presented in Figure 3.4, without the Dropout Layers, as they were removed by Tsiouris et al. (2018) to reduce model complexity. The batch size is 10, and the activation function used in the LSTM layers is the basic Sigmoid function. The loss function is the binary cross-entropy loss function. This architecture (Tsiouris et al., 2018) was chosen as at the time of publication it achieved state-of-the-art performance on the Epilepsy dataset used in this research.



(Tsiouris et al., 2018)

Figure 3.4: Baseline RNN Topology

Hyperparameter Tuning

No hyperparameter tuning was conducted for the baseline RNN model. Since this step was already conducted by Tsiouris et al. (2018), it was considered superfluous for this research. As such, the exact hyperparameters used by Tsiouris et al. (2018) for EEG classification were replicated in the baseline RNN implementation of this research.

3.4.3 State-of-the-Art RNN Model Architecture

In addition to the baseline RNN classifier, the current state-of-the-art RNN model for EEG classification was also compared to the SNN approach. An extensive review of the literature revealed the state-of-the-art time series classifier to be Karim et al.

(2018), who develop an LSTM-RNN model which they then evaluate on 35 time series classification datasets. The code for this model is provided by the researchers online, allowing for their exact implementation to be replicated in the current study. Figure 3.5 shows the 35 time series classification datasets used by Karim et al. (2018) to evaluate their RNN classifier. The green cells denote instances where Karim et al. (2018) achieved the best accuracy. Specifically, the model achieves best performance on two EEG datasets, one of which, denoted as 'EEG2' in Figure 3.5, is the Alcoholism dataset used in this research. The SOTA RNN is trained using a batch size of 128, as per the original paper.

Since this classifier performs so well on time series classification problems, it was selected as the state-of-the-art RNN classifier for this research, the results of which would be compared against the SNN-based classifier.

3.5 Performance Evaluation

Traditionally, the results of classification models are represented using a confusion matrix, as seen in figure 3.6. This confusion matrix displays important elements of the results, which are also the starting point for more sophisticated evaluation techniques. These are described below.

- Number of True Positives (TP): The number of observations in the positive target class which were correctly classified as such by the model.
- Number of False Negatives (FN): The number of observations in the positive target class which were incorrectly classified as being in the negative target class by the model.
- Number of False Positives (FP): The number of observations in the negative target class which were incorrectly classified as being in the positive target class by the model.
- Number of True Negatives (TN): The number of observations in the negative target class which were correctly classified as such by the model.

Datasets	LSTM-FCN	MLSTM-FCN	ALSTM-FCN	MALSTM-FCN	SOTA
Action 3d	71.72	75.42	72.73	74.74	70.71 [DTW]
Activity	53.13	61.88	55.63	58.75	66.25 [DTW]
ArabicDigits	100.00	100.00	99.00	99.00	99.00 [27]
Arabic-Voice	98.00	98.00	98.55	98.27	94.55 [18]
AREM	76.92	84.62	76.92	84.62	76.92 [DTW]
AUSLAN	97.00	97.00	96.00	96.00	98.00 [27]
CharacterTraject	99.00	100.00	99.00	100.00	99.00 [27]
CK+	96.07	96.43	97.10	97.50	93.56 [18]
CMUsubject16	100.00	100.00	100.00	100.00	100.00 [26]
Daily Sport	99.65	99.65	99.63	99.72	98.42 [DTW]
DigitShapes	100.00	100.00	100.00	100.00	100.00 [26]
ECG	85.00	86.00	86.00	86.00	93.00 [27]
EEG	60.94	65.63	64.06	64.07	62.50 [RF]
EEG2	90.67	91.00	90.67	91.33	77.50 [RF]
Gesture Phase	50.51	53.53	52.53	53.05	40.91 [DTW]
HAR	96.00	96.71	95.49	96.71	81.57 [RF]
HT Sensor	68.00	78.00	72.00	80.00	72.00 [DTW]
JapaneseVowels	99.00	100.00	99.00	99.00	98.00 [25]
KickvsPunch	90.00	100.00	90.00	100.00	100.00 [27]
Libras	99.00	97.00	98.00	97.00	95.00 [25]
LP1	74.00	86.00	80.00	82.00	96.00 [27]
LP2	77.00	83.00	80.00	77.00	76.00 [27]
LP3	67.00	80.00	80.00	73.00	79.00 [27]
LP4	91.00	92.00	89.00	93.00	96.00 [27]
LP5	61.00	66.00	62.00	67.00	71.00 [27]
Movement AAL	73.25	79.63	70.06	78.34	65.61 [SVM-Poly]
NetFlow	94.00	95.00	93.00	95.00	98.00 [27]
Occupancy	71.05	76.31	71.05	72.37	67.11 [DTW]
OHC	99.96	99.96	99.96	99.96	99.03 [18]
Ozone	67.63	81.50	79.19	79.78	75.14 [DTW]
PenDigits	97.00	97.00	97.00	97.00	95.00 [25]
Shapes	100.00	100.00	100.00	100.00	100.00 [27]
UWave	97.00	98.00	97.00	98.00	98.00 [27]
Wafer	99.00	99.00	99.00	99.00	99.00 [27]
WalkvsRun	100.00	100.00	100.00	100.00	100.00 [27]
Arith Mean	4.63	3.29	4.17	3.17	-
Geo Mean	3.72	2.58	3.21	2.42	-
Count	22.00	28.00	26.00	27.00	-
MPCE	5.01	4.21	4.93	4.19	-

(Karim et al., 2018)

Figure 3.5: Time Series Classification Datasets Tested by Karim et al. (2018) (Green Denotes Best Performance)

To compare the performances of the SNN and RNN models on the three datasets, a number of evaluation metrics were used. Each of these metrics is an indicator of certain model characteristics, and the metrics' utilities are outlined in this section.

		Classifier Prediction	
		Positive	Negative
Actual Value	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

(Banda, Angryk, & Martens, 2013)

Figure 3.6: Binary Classification Confusion Matrix

Accuracy

$$Formula : \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy refers to the number of correctly classified observations as a proportion of the total number of observations in the dataset. However, this metric does not discriminate between the model's ability to classify positive and negative members of the target class.

Precision

$$Formula : \frac{TP}{TP + FP}$$

Precision refers to the number of correct positive classifications as a proportion of the total number of positive classifications made by the model. It measures the classifier's ability to correctly identify positive members of the target class while not making the mistake of labelling negative observations as positive.

Recall

$$\text{Formula : } \frac{TP}{TP + FN}$$

The recall metric (also known as sensitivity) is the proportion of the amount of positive target class members that are classified as such by the model. This represents the model's ability to correctly identify positive cases while avoiding false negative classifications of these positive cases.

F1 Score

$$\text{Formula : } 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1 score is the harmonic mean of the precision and recall metrics. This metric allows for a balanced expression of the model's recall and precision.

Chapter 4

Implementation and Results

This chapter describes in detail how the experiment was implemented. The specific architectures of the final model are outlined. Additionally, the results of the experiment are presented.

4.1 Data Splitting

K-fold stratified sampling was used to split the research data. In order to implement this, the `StratifiedShuffleSplit` class from the `sklearn.model_selection` module was used. The splitting criteria were exactly the same for each of the three EEG datasets, as well as for both the SNN and LSTM model implementations. This included using the same random seed value for random data sampling. The motivation behind this was to ensure that the evaluation metrics of the SNN and LSTM models could be directly compared to each other.

The number of data folds created from the split was 4. Each of the data folds contained a training and a test dataset. The proportion of training to test data was 80:20. Since the data splits are stratified, the proportion of positive and negative target variables from the overall dataset was preserved in the individual folds.

In the case of the Alcoholism dataset, the data is already split into a training and test set (Lichman, n.d.). In AI, it is convention to use the preexisting data splits where available for classification problems. In addition to this, Karim et al. (2018)

use the preexisting training/test split, so in order to accurately compare the accuracy of the SNN approach to that of Karim et al. (2018), no custom data splitting of the Alcoholism dataset was conducted.

4.2 Hyperparameter Selection

This section outlines the process used for finding the optimal hyperparameters for modelling the three experimental datasets using the LSTM and SNN architectures.

4.2.1 SNN Hyperparameters

For the SNN models, hyperparameter optimisation was conducted as discussed in section 3.5.1. Each of the three models used to fit the three experimental datasets had its hyperparameters tuned separately according to what achieved the best relative test performance on the dataset. The following Table 4.1 shows the different hyperparameters that were optimized for each classifier model, along with what specific hyperparameter values were evaluated:

Hyperparameter	Values Tested
Strength of synapse weights from excitatory to inhibitory layer	5, 25, 125
Strength of synapse weights from inhibitory to excitatory layer	5, 25, 125
Pre-synaptic events learning rate	0.1, 1
Post-synaptic events learning rate	0.1, 1
Number of Spiking Neurons	5, 10

Table 4.1: Hyperparameters Evaluated for the SNN Models

The hyperparameter optimization process employed in this research involved using a Grid Search, implemented using the ParameterGrid class, from the sklearn.model_selection module. The above hyperparameters list forms 32 possible SNN combinations. Each hyperparameter combination was then used to train an SNN classifier for 1 epoch,

and the mean accuracy over the 4 data folds was calculated. Due to the high classification performance obtained after just 1 epoch of training, training with a higher number of epochs was conducted during the hyperparameter optimization step. The hyperparameter combination with the highest test accuracy was then selected for the development of the final SNN model.

Schizophrenia Classifier

The Grid Search optimisation determined that a classifier with the following SNN hyperparameters displayed the highest accuracy metric for the Schizophrenia Dataset:

- Strength of synapse weights from excitatory to inhibitory layer: 5
- Strength of synapse weights from inhibitory to excitatory layer: 125
- Pre-synaptic events learning rate: 1
- Post-synaptic events learning rate: 1
- Number of Spiking Neurons: 5

Alcoholism Classifier

The Grid Search optimisation determined that a classifier with the following SNN hyperparameters displayed the highest accuracy metric for the Alcoholism Dataset:

- Strength of synapse weights from excitatory to inhibitory layer: 5
- Strength of synapse weights from inhibitory to excitatory layer: 5
- Pre-synaptic events learning rate: 0.1
- Post-synaptic events learning rate: 0.1
- Number of Spiking Neurons: 5

Epilepsy Classifier

The Grid Search optimisation determined that a classifier with the following SNN hyperparameters displayed the highest accuracy metric for the Epilepsy Dataset:

- Strength of synapse weights from excitatory to inhibitory layer: 25
- Strength of synapse weights from inhibitory to excitatory layer: 25
- Pre-synaptic events learning rate: 0.1
- Post-synaptic events learning rate: 0.1
- Number of Spiking Neurons: 5

4.3 Number of Training Epochs

For both RNN and SNN classification approaches, the optimal number of epochs used for training of the final model needed to be determined. This is necessary to both ensure that the final models had learned to classify the data as well as possible, and to avoid without wasting time on additional training epochs that didn't contribute to the models' overall classification performance.

The 'Early Stopping' technique is typically used to determine the number of epochs over which to train the model (Prechelt, 1998). However, it was decided that in order to accurately replicate both the Baseline (Tsiouris et al., 2018) and SOTA (Karim et al., 2018) RNN classifiers from the literature, the number of training epochs used in the current research would be the same as those used in the original implementations. Karim et al. (2018) use 250 epochs to train the SOTA RNN model, while Tsiouris et al. (2018) use a maximum of 55 epochs. For consistency's sake, it was decided that the greater number of training epochs would be used for both the Baseline and SOTA RNNs. As a result, each of the RNN classifiers was trained for a total of 250 epochs, with the test set accuracy being recorded after each epoch of training. The model was saved after every epoch where the test accuracy was improved upon. This saved optimal model was then used for the final model evaluation. The high training epoch

number of 250 helped avoid the scenario where the model parameters were trained to a suboptimal state due to a local minima in the error surface, which is a common problem with ANN learning using backpropagation (Fukumizu & Amari, 2000). The problem of error surface local minima has also been observed in the supervised learning of SNNs (Fujita, Takase, Kita, & Hayashi, 2008).

With respect to the SNN training, preliminary results showed that maximum accuracy was achieved after a single full training pass of the dataset (or after a single 'epoch'). Therefore, each SNN classifier was only trained for a maximum of a single epoch.

4.4 Final Model Implementations

4.4.1 Final Baseline RNN Implementations

Functions were developed to calculate the Precision, Recall, and F1 metrics using the true and predicted labels. Keras callback functions were used to track the training time of the model. The Unix time is recorded when the epoch starts and when it ends, and the difference is returned. The model is trained and evaluated for each of the 4 data folds. The optimizer used is the Adam optimizer, and the model architecture and hyperparameters are the same as those outlined in section 3.4.2. The loss function used was the binary cross-entropy loss function. The final evaluation metrics are calculated by averaging the metrics from each data fold.

Schizophrenia Dataset

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 16, 64)	1982720
dropout (Dropout)	(None, 16, 64)	0
lstm_1 (LSTM)	(None, 32)	12416
dropout_1 (Dropout)	(None, 32)	0
dense (Dense)	(None, 30)	990
dense_1 (Dense)	(None, 1)	31
Total params: 1,996,157		
Trainable params: 1,996,157		
Non-trainable params: 0		

Figure 4.1: Final Keras Model Description for Schizophrenia Baseline RNN Classifier

As seen in Fig. 4.1, the final baseline RNN Keras model used to model the Schizophrenia dataset features a total of 1,996,157 trainable parameters. For each of the 4 data folds, the model is trained on 62 samples, and validated on 17 samples. The model input shape is (10, 64, 7680).

Alcoholism Dataset

Layer (type)	Output Shape	Param #
lstm_4 (LSTM)	(None, 256, 128)	98816
lstm_5 (LSTM)	(None, 128)	131584
dense_4 (Dense)	(None, 30)	3870
dense_5 (Dense)	(None, 1)	31
Total params: 234,301		
Trainable params: 234,301		
Non-trainable params: 0		

Figure 4.2: Final Keras Model Description for Alcoholism Baseline RNN Classifier

As seen in Fig. 4.2, the final baseline RNN Keras model used to model the Alcoholism dataset features a total of 234,301 trainable parameters. For each of the 4 data folds,

the model is trained on 100 samples, and validated on 100 samples. The model input shape is (10, 16, 256).

Epilepsy Dataset

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 1, 64)	1065216
dropout (Dropout)	(None, 1, 64)	0
lstm_1 (LSTM)	(None, 32)	12416
dropout_1 (Dropout)	(None, 32)	0
dense (Dense)	(None, 30)	990
dense_1 (Dense)	(None, 1)	31
Total params: 1,078,653		
Trainable params: 1,078,653		
Non-trainable params: 0		

Figure 4.3: Final Keras Model Description for Epilepsy Baseline RNN Classifier

As seen in Fig. 4.3, the final baseline RNN Keras model used to model the Epilepsy dataset features a total of 1,078,653 trainable parameters. For each of the 4 data folds, the model is trained on 160 samples, and validated on 100 samples. The model input shape is (10, 1, 4096).

4.4.2 Final State-of-the-Art RNN Implementations

The functions developed to calculate the Accuracy, Precision, Recall, and F1 metrics of the Baseline RNN classifier were reused for the same purpose for the State-of-the-Art RNN classifier. Additionally, the training time was also tracked using Keras callback functions. These were all recorded in text documents as the training progressed. The final Keras model descriptions for the State-of-the-Art RNN implementations can be found in Appendix A as follows:

Schizophrenia Dataset: Appendix A.1

Alcoholism Dataset: Appendix A.2

Epilepsy Dataset: Appendix A.3

4.4.3 Final SNN Implementations

If a model is being trained, the following process is used to determine the neural network object to be used. If the current epoch is the first, a new network is initialised following the topology defined in 3.4.1. If the current epoch is not the first, then a new network object is not created, preserving the model weights determined from the first epoch of training. After the model training has completed, it is saved to the local memory. This allows it to then be recovered, with the trained weights, for the model testing and evaluation phase, without having to retrain the model.

If the model is being used for testing, rather than training, the learning capabilities of the model are disabled. This is done by changing the learning rules between each layer to a learning rule with no effect (called NoOp). Additionally, since the weights of the model are affected by spikes, the voltage of each neuron is set to 0 for the duration of the model testing, so that spikes do not occur.

Before the model training begins, the input datapoints' feature scalars are multiplied by an intensity coefficient. The input data is then iterated through, and the model is trained on each datapoint. Training occurs by propagating the input spike data through the network, which then optimizes the weights of the network. The model's output spikes at each timestep are kept recorded in an array, known as the spike record.

Supervised learning is done using the Teacher Forcing method (outlined in Section 2.4.4.) which is implemented by injecting an additional training signal. This signal is constructed by dividing the n hidden spiking neurons in the network into n/m

equally sized groups, where m is the number of unique target classes. During training, a random neuron is selected from the group corresponding to the true label of the datapoint that the network is currently being trained on. Spikes are then artificially introduced at every time step in the spike record of this neuron. This forces the weights of neuron to learn to recognise samples with this particular label.

Each output neuron can be seen as being responsible for assigning observations to one particular certain target class. Target classes are not limited to being represented by a single output neuron - multiple output neurons can classify the same target class. After the input data for a sample is propagated through the SNN, the number of spikes in the output neurons are recorded. The neuron which exhibits the highest number of afferent spikes in the spike record is then used to classify the sample, according to the neuron's class assignment.

The training times for each training epoch are recorded. After each epoch is complete, the test dataset is used to calculate the classification metrics. Training is complete for all epochs, the final performance metrics are recorded for the overall model. This process is repeated for all 4 data folds, and the final classification metrics are calculated by averaging the ones from the individual data folds. These results are all recorded in a text document.

The SNN model input shape is:

(10, [Number of Dimensions], [Sequence Length]).

4.5 Model Results

For each model, training the classifier, as well as inferring the test results, is conducted on identical computing architecture. This is done to ensure that the training times of the SNN and LSTM models can be compared meaningfully to each other. The cloud-based computing architecture utilized for the experiment includes an Intel® Xeon® E5-2630 v3 8-Core 2.4GHz CPU, and 16GB of RAM. No GPU accelerator hardware is used. The cloud provider is Paperspace, and the specific Paperspace cloud server type used is the 'C6 Server'.

4.5.1 Baseline RNN Results

This section details the experimental results obtained from the baseline RNN classifiers on the three EEG datasets.

Schizophrenia Baseline RNN Classifier

Figure 4.4 shows the confusion matrix of the Schizophrenia Baseline RNN classifier, averaged over each data fold. Table 4.2 shows the evaluation metrics of the Schizophrenia Baseline RNN classifier specific to each of the 4 data folds used for cross-validation.

		True Label		Total
		Positive	Negative	
Predicted Label	Positive	$TP : 6.5$	$FP : 4$	10.5
	Negative	$FN : 2.5$	$TN : 4$	6.5
Total		9	8	$n = 17$

Accuracy = 0.618	Precision = 0.664	Recall = 0.722	F1 = 0.669
Number of Epochs = 4	Mean Epoch Time = 1.395	Total Training Time = 5.58	

Figure 4.4: Schizophrenia Baseline RNN Classifier - Confusion Matrix

	Data Fold			
Evaluation Metric	Fold 1	Fold 2	Fold 3	Fold 4
True Positives (TP)	8	5	7	6
True Negatives (TN)	2	7	4	6
False Positives (FP)	6	1	4	2
False Negatives (FN)	1	4	2	3
Accuracy	0.588	0.706	0.647	0.706
Precision	0.571	0.833	0.636	0.75
Recall	0.888	0.556	0.778	0.667
F1	0.696	0.667	0.7	0.706
Number of Epochs	6	2	5	3
Mean Epoch Time	0.882	1.841	1.132	1.725
Total Training Time	5.292	3.682	5.66	5.175

Table 4.2: Schizophrenia Baseline RNN Classifier Performance

Alcoholism Baseline RNN Classifier

Figure 4.5 shows the confusion matrix of the Alcoholism Baseline RNN classifier, averaged over each data fold. Table 4.3 shows the evaluation metrics of the Alcoholism Baseline RNN classifier specific to each of the 4 data folds used for cross-validation.

		True Label		
		Positive	Negative	Total
Predicted Label	Positive	$TP : 122$	$FP : 87$	209
	Negative	$FN : 178$	$TN : 213$	391
Total		300	300	$n = 600$

Accuracy = 0.558	Precision = 0.518	Recall = 0.26	F1 = 0.331
Number of Epochs = 1	Mean Epoch Time = 4.244	Total Training Time = 4.244	

Figure 4.5: Alcoholism Baseline RNN Classifier - Confusion Matrix

	Data Fold
Evaluation Metric	Fold 1
True Positives (TP)	122
True Negatives (TN)	213
False Positives (FP)	87
False Negatives (FN)	178
Accuracy	0.558
Precision	0.518
Recall	0.26
F1	0.331
Number of Epochs	1
Mean Epoch Time	4.244
Total Training Time	4.244

Table 4.3: Alcoholism Baseline RNN Classifier Performance

Epilepsy Baseline RNN Classifier

Figure 4.6 shows the confusion matrix of the Epilepsy Baseline RNN classifier, averaged over each data fold. Table 4.4 shows the evaluation metrics of the Epilepsy Baseline RNN classifier specific to each of the 4 data folds used for cross-validation.

		True Label		Total
		Positive	Negative	
Predicted Label	Positive	$TP : 15.25$	$FP : 1.5$	16.75
	Negative	$FN : 4.75$	$TN : 78.5$	83.25
Total		20	80	$n = 100$

Accuracy = 0.938	Precision = 0.91	Recall = 0.763	F1 = 0.822
Number of Epochs = 9	Mean Epoch Time = 0.468	Total Training Time = 4.212	

Figure 4.6: Epilepsy Baseline RNN Classifier - Confusion Matrix

	Data Fold			
Evaluation Metric	Fold 1	Fold 2	Fold 3	Fold 4
True Positives (TP)	17	17	11	14
True Negatives (TN)	79	80	80	78
False Positives (FP)	1	0	0	2
False Negatives (FN)	3	3	9	6
Accuracy	0.96	0.97	0.91	0.92
Precision	0.944	1	1.0	0.875
Recall	0.85	0.85	0.55	0.7
F1	0.895	0.919	0.71	0.778
Number of Epochs	13	5	9	9
Mean Epoch Time	0.319	0.606	0.452	0.494
Total Training Time	4.147	3.03	4.068	4.446

Table 4.4: Epilepsy Baseline RNN Classifier Performance

4.5.2 State-of-the-Art RNN Results

This section details the experimental results obtained from the State-of-the-Art RNN classifiers on the three EEG datasets.

Schizophrenia State-of-the-Art RNN Classifier

Figure 4.7 shows the confusion matrix of the Epilepsy State-of-the-Art RNN classifier, averaged over each data fold. Table 4.5 shows the evaluation metrics of the Epilepsy State-of-the-Art RNN classifier specific to each of the 4 data folds used for cross-validation.

		True Label		
		Positive	Negative	Total
Predicted Label	Positive	$TP : 7.5$	$FP : 0.75$	8.25
	Negative	$FN : 0.5$	$TN : 8.25$	8.75
Total		8	9	$n = 17$

Accuracy = 0.927	Precision = 0.917	Recall = 0.938	F1 = 0.924
Number of Epochs = 41.75	Mean Epoch Time = 16.532	Total Training Time = 690.211	

Figure 4.7: Schizophrenia State-of-the-Art RNN Classifier - Confusion Matrix

	Data Fold			
Evaluation Metric	Fold 1	Fold 2	Fold 3	Fold 4
True Positives (TP)	8	8	7	7
True Negatives (TN)	8	9	9	7
False Positives (FP)	1	0	0	2
False Negatives (FN)	0	0	1	1
Accuracy	0.941	1	0.941	0.824
Precision	0.889	1	1	0.778
Recall	1	1	0.875	0.875
F1	0.941	1	0.933	0.824
Number of Epochs	42	28	55	42
Mean Epoch Time	17.743	16.26	16.026	16.099
Total Training Time	745.206	455.28	881.43	676.158

Table 4.5: Schizophrenia State-of-the-Art RNN Classifier Performance

Alcoholism State-of-the-Art RNN Classifier

Figure 4.8 shows the confusion matrix of the Alcoholism State-of-the-Art RNN classifier, averaged over each data fold. Table 4.6 shows the evaluation metrics of the Alcoholism State-of-the-Art RNN classifier specific to each of the 4 data folds used for cross-validation.

		True Label		
		Positive	Negative	Total
Predicted Label	Positive	$TP : 419$	$FP : 23$	442
	Negative	$FN : 181$	$TN : 577$	758
Total		20	80	$n = 1200$

Accuracy = 0.83	Precision = 0.948	Recall = 0.698	F1 = 0.804
Number of	Mean Epoch	Total Training	
Epochs = 133	Time = 21.266	Time = 2828.245	

Figure 4.8: Alcoholism State-of-the-Art RNN Classifier - Confusion Matrix

	Data Fold
Evaluation Metric	Fold 1
True Positives (TP)	419
True Negatives (TN)	577
False Positives (FP)	23
False Negatives (FN)	181
Accuracy	0.83
Precision	0.948
Recall	0.698
F1	0.804
Number of Epochs	133
Mean Epoch Time	21.266
Total Training Time	2828.245

Table 4.6: Alcoholism State-of-the-Art RNN Classifier Performance

Epilepsy State-of-the-Art RNN Classifier

Figure 4.9 shows the confusion matrix of the Epilepsy State-of-the-Art RNN classifier, averaged over each data fold. Table 4.7 shows the evaluation metrics of the Epilepsy State-of-the-Art RNN classifier specific to each of the 4 data folds used for cross-validation.

		True Label		
		Positive	Negative	Total
Predicted Label	Positive	$TP : 77.75$	$FP : 0.5$	78.25
	Negative	$FN : 2.25$	$TN : 19.5$	21.75
Total		80	20	$n = 100$

Accuracy = 0.973	Precision = 0.994	Recall = 0.972	F1 = 0.983
Number of Epochs = 76.25	Mean Epoch Time = 23.316	Total Training Time = 1777.845	

Figure 4.9: Epilepsy State-of-the-Art RNN Classifier - Confusion Matrix

	Data Fold			
Evaluation Metric	Fold 1	Fold 2	Fold 3	Fold 4
True Positives (TP)	77	79	77	78
True Negatives (TN)	20	20	20	18
False Positives (FP)	0	0	0	2
False Negatives (FN)	3	1	3	2
Accuracy	0.97	0.99	0.97	0.96
Precision	1	1	1	0.975
Recall	0.963	0.988	0.963	0.975
F1	0.981	0.994	0.981	0.975
Number of Epochs	3	99	152	51
Mean Epoch Time	22.242	23.228	23.933	23.859
Total Training Time	66.726	2299.57	3637.781	1216.817

Table 4.7: Epilepsy State-of-the-Art RNN Classifier Performance

4.5.3 SNN Results

Initial results for the SNN classifier models showed that for all three experimental EEG datasets, the models achieved 100% classification accuracy after just 1 epoch of training. Due to this, the SNN classifier test performance was not evaluated after each training epoch, but rather after each individual input sample was presented to the model for training. This was done to determine how many training input samples were necessary for the test accuracy to reach 100%. The section details the experimental results obtained from the State-of-the-Art RNN classifiers on the three EEG datasets.

Schizophrenia SNN Classifier

Figure 4.10 shows the confusion matrix of the Schizophrenia SNN classifier, averaged over each data fold. Table 4.8 shows the evaluation metrics of the Schizophrenia SNN classifier specific to each of the 4 data folds used for cross-validation.

		True Label		
		Positive	Negative	Total
Predicted Label	Positive	$TP : 8$	$FP : 0$	8
	Negative	$FN : 0$	$TN : 9$	9
Total		8	9	$n = 17$
Accuracy = 1		Precision = 1		Recall = 1
Number of		Mean Training		Total Training
Training Samples = 3		Sample Time = 8.994		Time = 26.982
				F1 = 1

Figure 4.10: Schizophrenia SNN Classifier - Confusion Matrix

	Data Fold			
Evaluation Metric	Fold 1	Fold 2	Fold 3	Fold 4
True Positives (TP)	8	8	8	8
True Negatives (TN)	9	9	9	9
False Positives (FP)	0	0	0	0
False Negatives (FN)	0	0	0	0
Accuracy	1	1	1	1
Precision	1	1	1	1
Recall	1	1	1	1
F1	1	1	1	1
Number of Training Samples	3	3	3	3
Mean Training Sample Time	9.145	8.835	9.135	8.862
Total Training Time	27.435	26.505	27.405	26.586

Table 4.8: Schizophrenia SNN Classifier Performance

Alcoholism SNN Classifier

Figure 4.11 shows the confusion matrix of the Alcoholism SNN classifier, averaged over each data fold. Table 4.9 shows the evaluation metrics of the Alcoholism SNN classifier specific to single data fold used for validation.

		True Label		
		Positive	Negative	Total
Predicted Label	Positive	$TP : 300$	$FP : 0$	300
	Negative	$FN : 0$	$TN : 300$	300
Total		300	300	$n = 600$
Accuracy = 1		Precision = 1		Recall = 1
F1 = 1		F1 = 1		
Number of		Mean Training		Total Training
Training Samples = 9		Sample Time = 0.303		Time = 1.818

Figure 4.11: Alcoholism SNN Classifier - Confusion Matrix

	Data Fold
Evaluation Metric	Fold 1
True Positives (TP)	300
True Negatives (TN)	300
False Positives (FP)	0
False Negatives (FN)	0
Accuracy	1
Precision	1
Recall	1
F1	1
Number of Training Samples	6
Mean Training Sample Time	0.303
Total Training Time	1.818

Table 4.9: Alcoholism SNN Classifier Performance

Epilepsy SNN Classifier

Figure 4.12 shows the confusion matrix of the Epilepsy SNN classifier, averaged over each data fold. Table 4.10 shows the evaluation metrics of the Epilepsy SNN classifier

specific to each of the 4 data folds used for cross-validation.

		True Label		
		Positive	Negative	Total
Predicted Label	Positive	$TP : 80$	$FP : 0$	80
	Negative	$FN : 0$	$TN : 20$	20
Total		80	20	$n = 100$
Accuracy = 1		Precision = 1		Recall = 1
F1 = 1		Mean Training		Total Training
Number of		Sample Time = 11.227		Time = 72.976
Training Samples = 6.5				

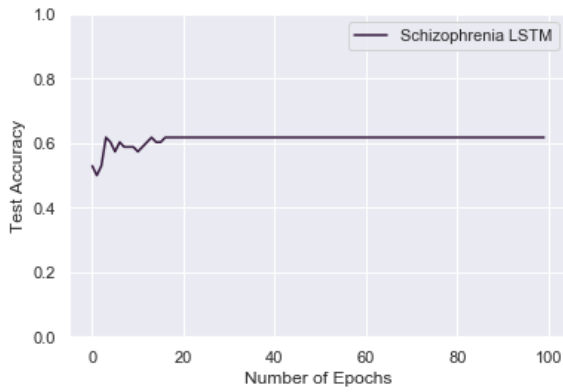
Figure 4.12: Epilepsy SNN Classifier - Confusion Matrix

	Data Fold			
Evaluation Metric	Fold 1	Fold 2	Fold 3	Fold 4
True Positives (TP)	80	80	80	80
True Negatives (TN)	20	20	20	20
False Positives (FP)	0	0	0	0
False Negatives (FN)	0	0	0	0
Accuracy	1	1	1	1
Precision	1	1	1	1
Recall	1	1	1	1
F1	1	1	1	1
Number of Training Samples	8	8	5	5
Mean Training Sample Time	29.479	5.009	5.13	5.289
Total Training Time	235.832	40.072	25.65	26.445

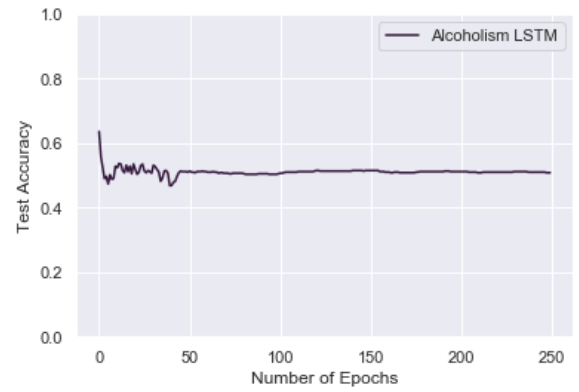
Table 4.10: Epilepsy SNN Classifier Performance

Figure 4.13 shows the relationship between number of training epochs and test set accuracy for the RNN-based approaches, while Figure 4.14 shows it for the SNN-based

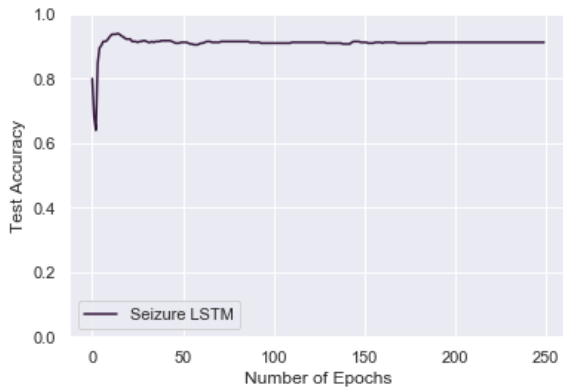
classifier.



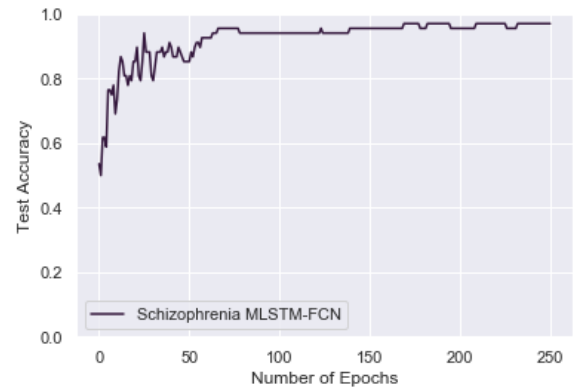
(a) Schizophrenia Baseline RNN Classifier



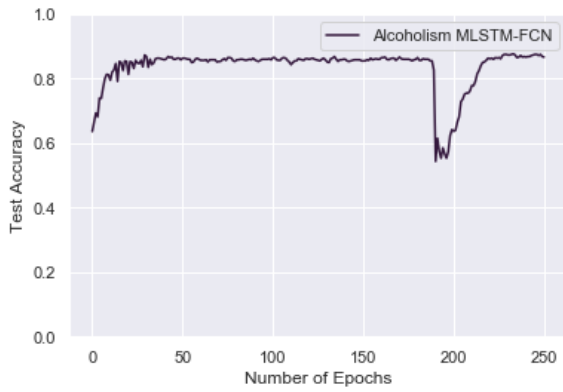
(b) Alcoholism Baseline RNN Classifier



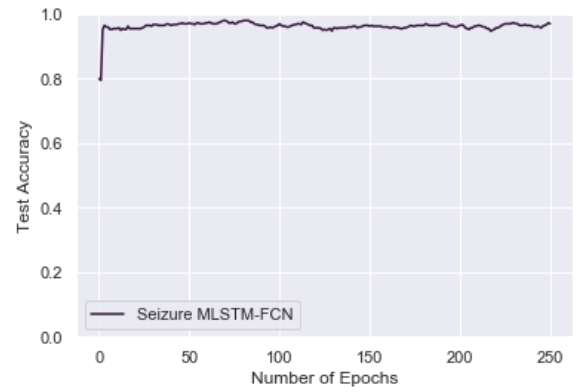
(c) Seizure Baseline RNN Classifier



(d) Schizophrenia State-of-the-Art RNN Classifier

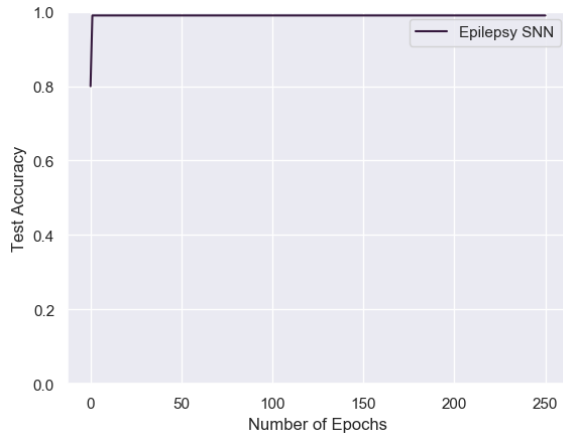


(e) Alcoholism State-of-the-Art RNN Classifier

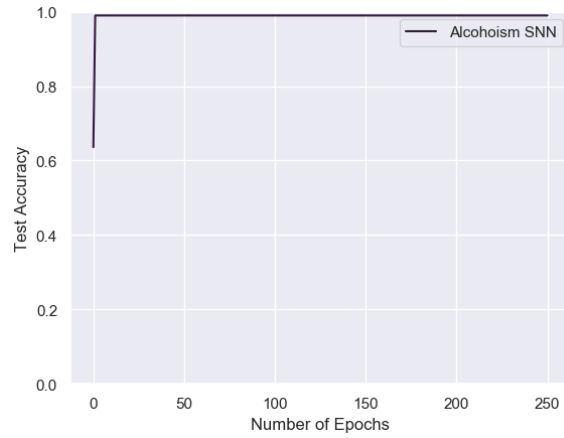


(f) Seizure State-of-the-Art RNN Classifier

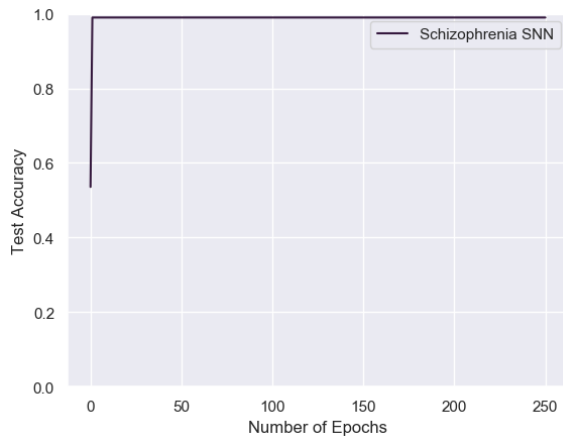
Figure 4.13: Classifiers' Test Accuracies Compared to Number of Training Epochs (RNN Approaches)



(a) Epilepsy SNN Classifier



(b) Alcoholism SNN Classifier



(a) Schizophrenia SNN Classifier

Figure 4.14: Classifiers' Test Accuracies Compared to Number of Training Epochs (SNN Approach)

Chapter 5

Evaluation, Analysis and Discussion

This chapter deals with the evaluation of the results of the research experiment, outlined in Chapter 4. The results are also analysed in the context of the existing academic literature, and their significance and contribution to the field are outlined and discussed.

5.1 Comparison of Classifiers Results

Figure 5.1 presents a comparison of the Accuracy, Precision, Recall and F1 metrics for each of the three experimental datasets, using the baseline RNN, state-of-the-art RNN, and SNN classifiers. In the case of each dataset, the highest metrics are presented in bold. For the Schizophrenia and Epilepsy datasets, the classifier test performance metrics are validated using 4-fold cross-validation. Cross-validation was not employed for the Alcoholism dataset, since the dataset features a predetermined training and test set (Lichman, n.d.).

Figure 5.2 presents a comparison of the training time for each of the three experimental datasets, using the baseline RNN, state-of-the-art RNN, and SNN classifiers. In the case of each dataset, the lowest training times are presented in bold. For the Schizophrenia and Epilepsy datasets, the classifier test performance metrics are

validated using 4-fold cross-validation. Cross-validation was not employed for the Alcoholism dataset, since the dataset features a predetermined training and test set (Lichman, n.d.).

		EEG Dataset		
Model	Metric	Schizophrenia	Alcoholism	Epilepsy
Baseline	Accuracy	0.618	0.558	0.938
LSTM	Precision	0.664	0.518	0.91
	Recall	0.722	0.26	0.763
	F1	0.669	0.311	0.822
MLSTM-	Accuracy	0.927	0.83	0.973
FCN	Precision	0.917	0.948	0.994
	Recall	0.938	0.698	0.972
	F1	0.924	0.804	0.983
SNN	Accuracy	1.0	1.0	1.0
	Precision	1.0	1.0	1.0
	Recall	1.0	1.0	1.0
	F1	1.0	1.0	1.0

Table 5.1: SNN and LSTM Classifiers Comparison

Model	Metric (Mean Over 4 Data Folds)	EEG Dataset		
		Schizophrenia	Alcoholism	Epilepsy
Baseline	Mean No. of Epochs	4	1	9
LSTM	Mean Time per Epoch	1.395s	4.244s	0.468s
	Total Training Time	5.58s	4.244s	4.212s
MLSTM-	Mean No. of Epochs	41.75	133	76.25
FCN	Mean Time per Epoch	6.532s	21.266s	23.16s
	Total Training Time	690.211s	2828.378s	1777.845s
SNN	Mean No. of Samples	3	9	6.5
	Mean Time per Sample	8.994s	0.303s	11.227s
	Total Training Time	26.982s	2.727s	72.976s

Table 5.2: SNN and LSTM Training Time Comparison (in seconds)

5.2 Hypothesis Evaluation

This section deals with the evaluation of the two hypothesis used to answer the research question. All of the hypotheses are evaluated to a significance level (α) of 0.05.

5.2.1 Hypothesis 1 - Accuracy

- *H0: Spiking Neural Network models do not have higher accuracy than Recurrent Neural Network models when implemented for classification of non-stationary time series data.*
- *H1: Spiking Neural Network models have higher accuracy than Recurrent Neural Network models when implemented for classification of non-stationary time series data.*

For each of the three datasets, two One-Tailed Wilcoxon Signed Rank Test (WSR) were used to evaluate the above hypothesis. The One-Tailed WSR is a non-parametric test used to determine if the SNN's mean accuracy over the 4 data folds is greater than that of the baseline RNN and state-of-the-art RNN classifiers.

Schizophrenia EEG Dataset

Data Fold	1	2	3	4
	Test Accuracies			
X: SNN	1	1	1	1
Y: Baseline LSTM	0.588	0.706	0.647	0.706
One-Tailed Wilcoxon Signed Rank Test (X-Y)				
Test Statistic (W)	10.000			
p-value	0.033			
Significance Level (α)	0.05			

Table 5.3: Schizophrenia WSR Test: SNN - Baseline LSTM

As seen in Table 5.7, which compares the baseline LSTM and SNN classifier accuracies, the p-value of the One-Tailed WSR Test is less than the Significance Level (α) of 0.05. Therefore, the Null Hypothesis (H_0) is rejected for the Schizophrenia EEG Dataset. Therefore, it can be said that the SNN model has a higher accuracy than the baseline RNN model when implemented for classification of the Schizophrenia EEG Dataset.

Data Fold	1	2	3	4
	Test Accuracies			
X: SNN	1	1	1	1
Y: MLSTM-FCN	0.941	1	0.941	0.824
One-Tailed Wilcoxon Signed Rank Test (X-Y)				
Test Statistic (W)	6.000			
p-value	0.051			
Significance Level (α)	0.05			

Table 5.4: Schizophrenia WSR Test: SNN - State-of-the-art LSTM

As seen in Table 5.8, which compares the state-of-the-art RNN and SNN classifier accuracies, the p-value of the One-Tailed WSR Test is greater than the Significance Level (α) of 0.05. Therefore, the Null Hypothesis (H_0) fails to be rejected for the Schizophrenia EEG Dataset. It cannot be said that the SNN model has a higher accuracy than the state-of-the-art RNN model when implemented for classification of the Schizophrenia EEG Dataset.

Alcoholism EEG Dataset

No cross-validation was employed to validate the classifiers' test performance on the Alcoholism EEG Dataset. This is because the Alcoholism EEG Dataset has predetermined Training and Test datasets (Lichman, n.d.). In order to ensure that the performance metrics of the SNN classifier can be compared to those of the state-of-the-art metrics (Karim et al., 2018), it's important that the test metrics are calculated on the same test dataset. As a result, the WSR test was not used on this dataset.

Epilepsy EEG Dataset

Data Fold	1	2	3	4
	Test Accuracies			
X: SNN	1	1	1	1
Y: Baseline LSTM	0.96	0.97	0.91	0.92
One-Tailed Wilcoxon Signed Rank Test (X-Y)				
Test Statistic (W)	10.000			
p-value	0.034			
Significance Level (α)	0.05			

Table 5.5: Epilepsy WSR Test: SNN - Baseline LSTM

As seen in Table 5.5, which compares the baseline LSTM and SNN classifier accuracies, the p-value of the One-Tailed WSR Test is less than the Significance Level (α) of 0.05. Therefore, the Null Hypothesis (H_0) is rejected for the Epilepsy EEG Dataset. Therefore, it can be said that the SNN model has a higher accuracy than the baseline RNN model when implemented for classification of the Epilepsy EEG Dataset.

Data Fold	1	2	3	4
	Test Accuracies			
X: SNN	1	1	1	1
Y: MLSTM-FCN	0.97	0.99	0.97	0.96
One-Tailed Wilcoxon Signed Rank Test (X-Y)				
Test Statistic (W)	10.000			
p-value	0.033			
Significance Level (α)	0.05			

Table 5.6: Epilepsy WSR Test: SNN - State-of-the-art LSTM

As seen in Table 5.6, which compares the state-of-the-art RNN and SNN classifier

accuracies, the p-value of the One-Tailed WSR Test is less than the Significance Level (α) of 0.05. Therefore, the Null Hypothesis (H_0) is rejected for the Epilepsy EEG Dataset. It cannot be said that the SNN model has a higher accuracy than the state-of-the-art RNN model when implemented for classification of the Epilepsy EEG Dataset.

5.2.2 Hypothesis 2 - Training Time

- *H0: Spiking Neural Network models do not have a faster training time than Recurrent Neural Network models when implemented for classification of time series data.*
- *H1: Spiking Neural Network models have a faster training time than Recurrent Neural Network models when implemented for classification of time series data.*

For each of the three datasets, two One-Tailed Wilcoxon Signed Rank Test (WSR) were used to evaluate the above hypothesis. The One-Tailed WSR is a non-parametric test used to determine if the SNN's mean training time over the 4 data folds is less than that of the baseline RNN and state-of-the-art RNN classifiers.

Schizophrenia EEG Dataset

Data Fold	1	2	3	4
	Training Times (seconds)			
X: SNN	27.435	26.505	27.405	26.586
Y: Baseline LSTM	5.292	3.682	5.66	5.175
One-Tailed Wilcoxon Signed Rank Test (Y-X)				
Test Statistic (W)	10.000			
p-value	0.966			
Significance Level (α)	0.05			

Table 5.7: Schizophrenia WSR Test: SNN - Baseline LSTM

As seen in Table 5.7, which compares the baseline LSTM and SNN classifier training times, the p-value of the One-Tailed WSR Test is greater than the Significance Level (α) of 0.05. Therefore, the Null Hypothesis (H_0) fails to be rejected for the Schizophrenia EEG Dataset. Therefore, it can be said that the SNN model does not have a shorter training time than the baseline RNN model when implemented for classification of the Schizophrenia EEG Dataset.

Data Fold	1	2	3	4
	Training Times (seconds)			
X: SNN	27.435	26.505	27.405	26.586
Y: MLSTM-FCN	745.206	455.28	881.43	676.158
One-Tailed Wilcoxon Signed Rank Test (Y-X)				
Test Statistic (W)	10.000			
p-value	0.034			
Significance Level (α)	0.05			

Table 5.8: Schizophrenia WSR Test: SNN - State-of-the-art LSTM

As seen in Table 5.8, which compares the state-of-the-art RNN and SNN classifier accuracies, the p-value of the One-Tailed WSR Test is less than the Significance Level (α) of 0.05. Therefore, the Null Hypothesis (H_0) is rejected for the Schizophrenia EEG Dataset. Therefore, it can be said that the SNN model does have a shorter training time than the state-of-the-art RNN model when implemented for classification of the Schizophrenia EEG Dataset.

Alcoholism EEG Dataset

No cross-validation was employed to validate the classifiers' training time on the Alcoholism EEG Dataset. This is because the Alcoholism EEG Dataset has predetermined Training and Test datasets (Lichman, n.d.), and as a result, no WSR test was employed as there was only 1 predetermined data fold used for training and testing.

Epilepsy EEG Dataset

Data Fold	1	2	3	4
	Training Times (seconds)			
X: SNN	235.832	40.072	25.65	26.445
Y: Baseline LSTM	4.147	3.03	4.068	4.446
One-Tailed Wilcoxon Signed Rank Test (Y-X)				
Test Statistic (W)	0.000			
p-value	0.966			
Significance Level (α)	0.05			

Table 5.9: Epilepsy WSR Test: SNN - Baseline LSTM

As seen in Table 5.9, which compares the baseline LSTM and SNN classifier accuracies, the p-value of the One-Tailed WSR Test is greater than the Significance Level (α) of 0.05. Therefore, the Null Hypothesis (H_0) fails to be rejected for the Epilepsy EEG Dataset. Therefore, it can be said that the SNN model does not have a shorter training

time than the baseline RNN model when implemented for classification of the Epilepsy EEG Dataset.

Data Fold	1	2	3	4
	Training Times (seconds)			
X: SNN	235.832	40.072	25.65	26.445
Y: MLSTM-FCN	66.726	2299.57	3637.781	1216.817
One-Tailed Wilcoxon Signed Rank Test (Y-X)				
Test Statistic (W)	1.000			
p-value	0.072			
Significance Level (α)	0.05			

Table 5.10: Epilepsy WSR Test: SNN - State-of-the-art LSTM

As seen in Table 5.10, which compares the state-of-the-art RNN and SNN classifier accuracies, the p-value of the One-Tailed WSR Test is greater than the Significance Level (α) of 0.05. Therefore, the Null Hypothesis (H_0) is not rejected for the Epilepsy EEG Dataset. Therefore, it can be said that the SNN model does not have a shorter training time than the state-of-the-art RNN model when implemented for classification of the Epilepsy EEG Dataset.

5.3 Summary of Key Findings

This section outlines the key findings reached over the course of the research project. For clarity, the results of all the research hypotheses and whether or not the null hypothesis is rejected is also presented in Table 5.11.

		EEG Dataset		
Research Hypotheses	Comparison	Schizophrenia	Alcoholism	Epilepsy
Hypothesis 1 - Accuracy H0: Spiking Neural Network models do not have higher accuracy than Recurrent Neural Network models when implemented for classification of non-stationary time series data. H1: Spiking Neural Network models have higher accuracy than Recurrent Neural Network models when implemented for classification of non-stationary time series data.	SNN vs. Baseline RNN	Reject H0	Reject H0	Reject H0
	SNN vs. SOTA RNN	Fail to Reject H0 (borderline)	Reject H0	Reject H0
Hypothesis 1 - Accuracy H0: Spiking Neural Network models do not have a faster training time than Recurrent Neural Network models when implemented for classification of time series data. H1: Spiking Neural Network models have a faster training time than Recurrent Neural Network models when implemented for classification of time series data.	SNN vs. Baseline RNN	Fail to Reject H0	Reject H0	Fail to Reject H0
	SNN vs. SOTA RNN	Reject H0	Reject H0	Fail to Reject H0

Table 5.11: Hypotheses Tests Results

- For the Epilepsy and Schizophrenia datasets, the SNN classifier outperforms the baseline RNN classifier in terms of accuracy, to a statistical significance level of 0.05. For the Alcoholism dataset, the accuracy on the predetermined test set reached using the SNN classifier is superior to that reached using the baseline RNN classifier.
- For the Epilepsy dataset, the SNN classifier outperforms the state-of-the-art RNN classifier in terms of accuracy, to a statistical significance level of 0.05. Additionally, for the Alcoholism dataset, the accuracy on the predetermined test set reached using the SNN classifier is superior to that reached using the state-of-the-art RNN classifier.
- While the SNN classifier does outperform the state-of-the-art RNN classifier for the Schizophrenia dataset in terms of accuracy, it does not do so to a statistically significant degree, as the p-value from the WSR test is 0.051. However, this WSR test result can be perceived as being just on the boundary of being considered statistically significant.
- For each of the three EEG datasets, the SNN classifier requires only a handful of input samples for the model test accuracy to reach the maximum. This is significantly less input data than what is necessary for the RNN-based approaches.
- For the Alcoholism dataset, the SNN classifier achieved the shortest total training time compared to both the baseline and state-of-the-art RNNs.
- For the Schizophrenia dataset, the SNN classifier achieved a shorter total training time than the state-of-the-art RNN, but a longer total training time than the baseline RNN, to a statistical significance level of 0.05
- For the Epilepsy dataset, the SNN classifier exhibited a longer total training time than both the baseline and state-of-the-art RNN classifiers, to a statistical significance level of 0.05

5.4 Strengths and Limitations

The research conducted in this project has a number of strengths. Firstly, multiple datasets were used to investigate the viability of SNN-based approaches to non-stationary time series classification. The decision to use 3 datasets as opposed to 1 makes the results of the research project more academically useful, as a classifier model that performs well on a number of datasets rather than just a single one is more generalisable.

In addition to this, this project made use of an appropriate research methodology for the problem at hand - the test metrics of the models were evaluated using cross-validation to make the results more reliable. Also, the correct, non-parametric statistical hypothesis test (WSR) was used.

One limitation of the research design is the fact that only one type of layer topology was evaluated for the SNN classifier. This was an SNN with a single hidden layer. While this decision is motivated based on existing literature (as is discussed in further detail in Section 3.4.1), it would have been academically interesting also test an approach using multiple hidden layers, and it is not outside the realm of possibility that other SNN topologies or may have led to superior classification performance or training time. In addition to adjusting the number of hidden layers, there are a multitude of different approaches to SNN architecture which build on the standard spiking layers. One of these is Convolutional Spiking layers (Matsugu, Mori, Ishii, & Mitarai, 2002), which were not employed in this research. The choice to not use these alternative architectures is based on the lack of academic consensus present in the academic literature supporting their superiority in terms of classification performance. Had this consensus existed, this researcher would have made use of them for the research task.

5.5 Considerations of Previous Research

The most significant element of the results achieved by this project's SNN approach is the perfect test set classification accuracy on all three experimental datasets. Consulting the previous research, one can find that the current best result for the Alcoholism

dataset achieved an accuracy of 91.33% (Karim et al., 2018), while the SNN classifier used in this project achieves 100% accuracy. Therefore, when taking the context of previous research into account, this project achieves best performance for the Alcoholism dataset (Lichman, n.d.).

To this researcher’s knowledge, there is no previous research which classifies the Schizophrenia dataset (Gorbachevskaya & Borisov, n.d.) used in this research. As such, there is no existing best accuracy against which to compare it.

When taking previous research into consideration, the results of this project are quite interesting. Classification of non-stationary time series is a difficult problem in the field of AI, with EEG classification being one example of a sub-problem in this domain. Existing EEG classification implementations in the literature make use of either highly complex deep learning solutions (Karim et al., 2018), or very large numbers of training epochs (Tsiouris et al., 2018), often in the hundreds. By contrast, the SNN approach used in this research uses a non-deep layer topology of just 1 hidden layer, and, as can be seen in Table 5.2, requires training on less than 10 input data samples before the maximum accuracy is reached. This is a unique situation in the context of the existing academic literature.

Another interesting realisation made when examining previous research is that the Forced Teaching supervised learning paradigm (Legenstein et al., 2005) utilised in this study is not very popular in the broader literature. Efforts were made to the best of this researcher’s abilities to find additional studies that make use of this technique for SNN supervised learning, but no results were found. One can hope that the results achieved by the use of this paradigm in this research will inspire more academic interest in Forced Teaching, as it is both a fast and accurate method of supervised learning.

Chapter 6

Conclusions

This chapter concludes the dissertation by providing an overall retrospective and motivation of the research conducted. The project’s contribution to the academic body of knowledge is also examined, and recommendations are made for directions that future work in this field can go.

6.1 Research Overview

The main purpose of this research was to investigate the validity of using SNNs over RNNs for EEG time series classification. This was motivated by a number of factors, including certain limitations posed by RNN modelling of time series data. These include (but are not limited to) LSTM layers’ necessity for expensive GPU acceleration, computationally inefficient training, and the sensitivity of its performance to the model topology. A more detailed explanation of LSTM’s limitations can be found in section 2.3.2. On the other hand, existing literature on SNNs has shown promising results for their potential to exceed the performance of LSTMs for the purpose of time series modelling, primarily due to their fundamentally temporal nature, efficient data encoding, and low training time. Additionally, their more stringent biological accuracy means that they are able to leverage a number of the relative computational advantages of the human brain, such as low-power operation, massive parallelism, and event-based information processing.

The research conducted in this dissertation consisted of 7 main objectives, which are listed as follows:

1. Determine which Spiking Neural Network simulator package to use.
2. Determine the Spiking Neural Network topology and learning rule.
3. Determine the baseline and state-of-the-art recurrent neural network architecture.
4. Develop the optimal SNN classification model for each dataset.
5. Replicate the baseline and state-of-the-art RNN classification models for each dataset.
6. The optimal classification models will be trained using the SNN and RNN approaches on each of the three experimental EEG datasets.
7. The performance metrics collected by evaluating the optimal SNN and RNN classifiers will be analysed and compared.

The final RNN and SNN topologies were determined by examining the existing literature and replicating the topologies used in the existing implementations with the best performance. For each SNN model, hyperparameter optimization is conducted with the use of a grid search, where a classifier is constructed using each different combination of hyperparameters. The hyperparameter set of the model with the highest test accuracy is then used for the final model.

All of the statistical programming used to conduct the experiment was implemented using the Python programming language. The RNN classifiers were modelled using the Keras deep learning framework, while the SNN classifiers were built using the BindsNET SNN simulator, which is based on PyTorch.

6.2 Problem Definition

To what extent do neural network models, built using a Spiking Neural Network, have superior accuracy and/or training time to models built using a Recurrent Neural Network when implemented for classification of non-stationary time series datasets?

6.3 Contributions to Body of Knowledge

This research provides an answer to the research problem defined in Section 6.2. Firstly, SNNs have superior accuracy to RNNs when implemented for classification of non-stationary time series datasets to a statistical significance of 0.05. The relevant hypotheses and the results of their evaluation can be seen in Table 5.11 - all of the tests conducted for Hypothesis 1 reject H_0 (except one test result which is a borderline case). This contributes to the body of knowledge on AI by demonstrating experimentally that SNNs are more accurate classifiers than RNNs for EEG time series.

The other aspect of the research problem studied in this project deals with training time - to what extent do SNNs have superior training time to RNNs when implemented for classification of non-stationary time series datasets? Again, consulting Table 5.11 which contains the results of the hypothesis tests, one can see that there is not enough evidence to reject H_0 , as 3 out of 6 WSR tests for Hypothesis 2 reject H_0 , and the other 3 fail to do so. However, this research still contributes to the body of knowledge by showing that SNNs can be somewhat competitive in terms of shorter training time when compared to RNNs.

6.4 Future Work and Recommendations

Based on the results of this research, multiple different recommendations for future academic work can be made.

The research design stage of this project was difficult due to the limitations presented by the lack of publicly available SNN implementations. While there are a number of SNN simulators available, many of these are decades old and have not been updated in recent years. Because of this, implementations of recent advances in the field of SNNs, such as newly developed supervised learning rules, are few and far in between. An apt recommendation for future research in SNNs would be the development of more extensive tools and software packages for simulating SNNs. One important factor that has led to the rapid growth observed in the field of AI and ANNs in recent years is the development of specialised frameworks such as Tensorflow, PyTorch and Keras. Frameworks like these are vital as they allow researchers to quickly construct and iterate high abstraction level implementations of ANNs, without having to spend time and resources on developing more low-level functionality. Additionally, the use of a small number of standard frameworks means that new knowledge and information is exchanged between academics more quickly, as the widespread use of a handful of tools makes it unnecessary to spend time learning to develop with a lot of different frameworks. Since such widely-adopted frameworks do not exist as of yet in the field of SNNs, their future research and development could serve to significantly expedite academic progress in the field.

The results of this research are highly encouraging for the potential of SNNs for EEG time series classification. In addition to this, the comparative nature of this research helps build a case based on empirical evidence for the use of SNNs over RNNs for time series classification. While 100% accuracy is a very rare result for a classifier in the field of AI, it remains unclear whether SNNs' ability to classify EEG time series extends to non-EEG time series, particularly in non-stationary environments. It is possible that, since EEGs are essentially the recorded voltages of spikes in the human brain, the SNN algorithm is particularly suitable for modelling EEGs, but not so much

for other types of time series. While some studies on non-EEG time series classification with SNNs exist (A. Zhang, Zhu, & Li, 2018), such as for sound classification (Wu, Chua, Zhang, Li, & Tan, 2018), only one directly compares RNN and SNN performance (Ghosh-Dastidar & Adeli, 2007), and this study includes a convolutional layer in the SNN architecture, unlike the architecture used in this experiment. Additionally, Ghosh-Dastidar and Adeli (2007) conducted their research in a time when RNN and SNN classifier tools were less advanced. More research is necessary to determine if SNNs are able to classify all types of time series with the same efficacy.

This research implemented the Forced Teaching supervised learning paradigm (Legenstein et al., 2005) for SNNs. While this training approach proved to be successful at modelling the datasets of the conducted experiment, more research needs to be done on the application of this learning paradigm. Specifically, Batch learning for Forced Teaching has not been developed as of the writing of this report - the only option currently available for researchers who want to implement this paradigm is on-line learning, where the weight updates to the SNN model are calculated and applied for every single observation in the dataset.

References

- Adrian, E. D., & Zotterman, Y. (n.d.). The impulses produced by sensory nerve-endings: Part ii. the response of a single end-organ. *The Journal of physiology*, 61 2, 151-71.
- Amjad, M. J., & Shah, D. (2016). Trading bitcoin and online time series prediction. In *Nips time series workshop*.
- Andrzejak, R., Lehnertz, K., Mormann, F., Rieke, C., David, P., & Elger, C. (2002, 01). Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 64, 061907. doi: 10.1103/PhysRevE.64.061907
- Atomico, & Slush. (2019). The state of european tech 2019. Retrieved from <https://2019.stateofeuropeantech.com/>
- Banda, J., Angryk, R., & Martens, P. (2013, 05). Steps toward a large-scale solar image data analysis to differentiate solar phenomena. *Solar Physics*, 288. doi: 10.1007/s11207-013-0304-x
- Bengio, Y., Simard, P., & Frasconi, P. (1994, March). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2), 157-166. doi: 10.1109/72.279181
- Berg, A., Paparoditis, E., & Politis, D. N. (2010). A bootstrap test for time series linearity. *Journal of Statistical Planning and Inference*, 140(12), 3841 -

REFERENCES

3857. Retrieved from <http://www.sciencedirect.com/science/article/pii/S0378375810002272> (Special Issue in Honor of Emanuel Parzen on the Occasion of his 80th Birthday and Retirement from the Department of Statistics, Texas AM University) doi: <https://doi.org/10.1016/j.jspi.2010.04.047>
- Blackwood, D. H. R., & Muir, W. J. (1990). Cognitive brain potentials and their application. *British Journal of Psychiatry*, 157(S9), 96–101. doi: 10.1192/S0007125000291897
- Blanco, S., Garcia, H., Quiroga, R. Q., Romanelli, L., & Rosso, O. A. (1995, July). Stationarity of the eeg series. *IEEE Engineering in Medicine and Biology Magazine*, 14(4), 395-399. doi: 10.1109/51.395321
- Bode, I., & Huelss, H. (2018). Autonomous weapons systems and changing norms in international relations. *Review of International Studies*, 44(3), 393–413. doi: 10.1017/S0260210517000614
- Bohte, S. (2004a, 06). The evidence for neural information processing with precise spike-times: A survey. *Nat. Comput.*, 3. doi: 10.1023/B:NACO.0000027755.02868.60
- Bohte, S. (2004b, 06). The evidence for neural information processing with precise spike-times: A survey. *Nat. Comput.*, 3. doi: 10.1023/B:NACO.0000027755.02868.60
- Bohte, S., Kok, J., & Poutré, H. (2001, 02). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48, 17-37. doi: 10.1016/S0925-2312(01)00658-0
- Bower, J., & Hale, J. (1991). Exploring neuronal circuits on graphics workstations. *Scientific computing and automation*, 35–45.
- Bower, J. M., & Beeman, D. (2012). *The book of genesis: exploring realistic neural models with the general neural simulation system*. Springer Science & Business Media.

- Brette, R., Lilith, M., Carnevale, T., Hines, M., Beeman, D., Bower, J., . . . Destexhe, A. (2008, 01). Simulation of networks of spiking neurons: A review of tools and strategies. *Journal of computational neuroscience*, *23*, 349-398. doi: 10.1007/s10827-007-0038-6
- Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J. M., . . . Destexhe, A. (2007, December). Simulation of networks of spiking neurons: a review of tools and strategies. *Journal of computational neuroscience*, *23*(3), 349–398. doi: 10.1007/s10827-007-0038-6
- Brockwell, P., & Davis, R. (2002). *An introduction to time series and forecasting* (Vol. 39). doi: 10.1007/978-1-4757-2526-1
- Buesing, L., Bill, J., Nessler, B., & Maass, W. (2011, 11). Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons. *PLoS computational biology*, *7*, e1002211. doi: 10.1371/journal.pcbi.1002211
- Burkitt, A. (2006, 08). A review of the integrate-and-fire neuron model: I. homogeneous synaptic input. *Biological cybernetics*, *95*, 1-19. doi: 10.1007/s00422-006-0068-6
- Cao, Y., Chen, Y., & Khosla, D. (2015, 05). Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, *113*, 54-66. doi: 10.1007/s11263-014-0788-3
- Caporale, N., & Dan, Y. (2008). Spike timing-dependent plasticity: A hebbian learning rule. *Annual Review of Neuroscience*, *31*(1), 25-46. Retrieved from <https://doi.org/10.1146/annurev.neuro.31.060407.125639> (PMID: 18275283) doi: 10.1146/annurev.neuro.31.060407.125639
- Costas-Santos, J., Serrano-Gotarredona, T., Serrano-Gotarredona, R., & Linares-Barranco, B. (2007, July). A spatial contrast retina with on-chip calibration for neuromorphic spike-based aer vision systems. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *54*(7), 1444-1458. doi: 10.1109/TCSI.2007.900179

- Craik, A., He, Y., & Contreras-Vidal, J. (2019, 02). Deep learning for electroencephalogram (eeg) classification tasks: A review. *Journal of Neural Engineering*, 16. doi: 10.1088/1741-2552/ab0ab5
- Culkin, R., & Das, S. R. (2017). Machine learning in finance: the case of deep learning for option pricing. *Journal of Investment Management*, 15(4), 92–100.
- Davies, M., Srinivasa, N., Lin, T., China, G., Cao, Y., Choday, S. H., ... Wang, H. (2018, January). Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1), 82-99. doi: 10.1109/MM.2018.112130359
- Diehl, P., & Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, 9, 99. Retrieved from <https://www.frontiersin.org/article/10.3389/fncom.2015.00099> doi: 10.3389/fncom.2015.00099
- Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S., & Pfeiffer, M. (2015, July). Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 international joint conference on neural networks (ijcnn)* (p. 1-8). doi: 10.1109/IJCNN.2015.7280696
- Donahue, J., Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., & Darrell, T. (2014, 11). Long-term recurrent convolutional networks for visual recognition and description. *Arxiv, PP*. doi: 10.1109/TPAMI.2016.2599174
- Douillard, B., Fox, D., & Ramos, F. (2011). A spatio-temporal probabilistic model for multi-sensor multi-class object recognition. In M. Kaneko & Y. Nakamura (Eds.), *Robotics research* (pp. 123–134). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Dutta, S., Kumar, V., Shukla, A., Mohapatra, N., & Ganguly, U. (2017, 12). Leaky integrate and fire neuron by charge-discharge dynamics in floating-body mosfet. *Scientific Reports*, 7. doi: 10.1038/s41598-017-07418-y

REFERENCES

- Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2), 179 - 211. Retrieved from <http://www.sciencedirect.com/science/article/pii/036402139090002E> doi: [https://doi.org/10.1016/0364-0213\(90\)90002-E](https://doi.org/10.1016/0364-0213(90)90002-E)
- Eppler, J., Helias, M., Muller, E., Diesmann, M., & Gewaltig, M.-O. (2009). Pynest: a convenient interface to the nest simulator. *Frontiers in Neuroinformatics*, 2, 12. Retrieved from <https://www.frontiersin.org/article/10.3389/neuro.11.012.2008> doi: 10.3389/neuro.11.012.2008
- Ernst Niedermeyer, F. L. d. S. M. P. (2005). *Electroencephalography : basic principles, clinical applications, and related fields* (Fifth edition ed.). Lippincott Williams Wilkins.
- Feng, J. (2001). Is the integrate-and-fire model good enough?—a review. *Neural Networks*, 14(6), 955 - 975. Retrieved from <http://www.sciencedirect.com/science/article/pii/S0893608001000740> doi: [https://doi.org/10.1016/S0893-6080\(01\)00074-0](https://doi.org/10.1016/S0893-6080(01)00074-0)
- Fujita, M., Takase, H., Kita, H., & Hayashi, T. (2008, June). Shape of error surfaces in spikeprop. In *2008 ieee international joint conference on neural networks (ieee world congress on computational intelligence)* (p. 840-844). doi: 10.1109/IJCNN.2008.4633895
- Fukumizu, K., & Amari, S. (2000). Local minima and plateaus in hierarchical structures of multilayer perceptrons. *Neural Networks*, 13(3), 317 - 327. Retrieved from <http://www.sciencedirect.com/science/article/pii/S0893608000000095> doi: [https://doi.org/10.1016/S0893-6080\(00\)00009-5](https://doi.org/10.1016/S0893-6080(00)00009-5)
- Furber, S. B., Galluppi, F., Temple, S., & Plana, L. A. (2014, May). The spinaker project. *Proceedings of the IEEE*, 102(5), 652-665. doi: 10.1109/JPROC.2014.2304638

REFERENCES

- Garland, M., Le Grand, S., Nickolls, J., Anderson, J., Hardwick, J., Morton, S., ... Volkov, V. (2008, July). Parallel computing experiences with cuda. *IEEE Micro*, 28(4), 13-27. doi: 10.1109/MM.2008.57
- Gers, F., Schmidhuber, J., & Cummins, F. (2000, 10). Learning to forget: Continual prediction with lstm. *Neural computation*, 12, 2451-71. doi: 10.1162/089976600300015015
- Gerstner, W., Kistler, W. M., Naud, R., & Paninski, L. (2014a). *Neuronal dynamics: From single neurons to networks and models of cognition*. New York, NY, USA: Cambridge University Press.
- Gerstner, W., Kistler, W. M., Naud, R., & Paninski, L. (2014b). *Neuronal dynamics: From single neurons to networks and models of cognition*. New York, NY, USA: Cambridge University Press.
- Gewaltig, M., & Diesmann, M. (2007). NEST (NEural Simulation Tool). *Scholarpedia*, 2(4), 1430. (revision #130182) doi: 10.4249/scholarpedia.1430
- Ghosh-Dastidar, S., & Adeli, H. (2007, August). Improved spiking neural networks for eeg classification and epilepsy and seizure detection. *Integr. Comput.-Aided Eng.*, 14(3), 187-212. Retrieved from <http://dl.acm.org/citation.cfm?id=1367089.1367090>
- Ghosh-Dastidar, S., & Adeli, H. (2009). A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection. *Neural Networks*, 22(10), 1419 - 1431. Retrieved from <http://www.sciencedirect.com/science/article/pii/S0893608009000653> doi: <https://doi.org/10.1016/j.neunet.2009.04.003>
- Gollisch, T., & Meister, M. (2008). Rapid neural coding in the retina with relative spike latencies. *Science*, 319(5866), 1108-1111. Retrieved from <https://science.sciencemag.org/content/319/5866/1108> doi: 10.1126/science.1149639

REFERENCES

- Goodman, D., & Brette, R. (2008, 02). Brian: A simulator for spiking neural networks in python. *Frontiers in neuroinformatics*, 2, 5. doi: 10.3389/neuro.11.005.2008
- Goodman, D., & Brette, R. (2009, 09). The brian simulator. *Frontiers in neuroscience*, 3, 192-7. doi: 10.3389/neuro.01.026.2009
- Gorbachevskaya, N., & Borisov, S. (n.d.). *Eeg of healthy adolescents and adolescents with symptoms of schizophrenia*. Retrieved from http://brain.bio.msu.ru/eeg_schizophrenia.htm
- Hazan, H., Saunders, D. J., Khan, H., Patel, D., Sanghavi, D. T., Siegelmann, H. T., & Kozma, R. (2018a). Bindsnet: A machine learning-oriented spiking neural networks library in python. *Frontiers in Neuroinformatics*, 12, 89. Retrieved from <https://www.frontiersin.org/article/10.3389/fninf.2018.00089> doi: 10.3389/fninf.2018.00089
- Hazan, H., Saunders, D. J., Khan, H., Patel, D., Sanghavi, D. T., Siegelmann, H. T., & Kozma, R. (2018b, Dec). Bindsnet: A machine learning-oriented spiking neural networks library in python. *Frontiers in Neuroinformatics*, 12. Retrieved from <http://dx.doi.org/10.3389/fninf.2018.00089> doi: 10.3389/fninf.2018.00089
- Herz, A., Gollisch, T., Machens, C., & Jaeger, D. (2006, 11). Modeling single-neuron dynamics and computations: A balance of detail and abstraction. *Science (New York, N.Y.)*, 314, 80-5. doi: 10.1126/science.1127240
- Hines, M., & Carnevale, T. (2013). Neuron simulation environment. In D. Jaeger & R. Jung (Eds.), *Encyclopedia of computational neuroscience* (pp. 1–8). New York, NY: Springer New York. Retrieved from https://doi.org/10.1007/978-1-4614-7320-6_795-1 doi: 10.1007/978-1-4614-7320-6_795-1
- Hines, M., Davison, A., & Muller, E. (2009, 02). Neuron and python. *Frontiers in neuroinformatics*, 3, 1. doi: 10.3389/neuro.11.001.2009

REFERENCES

- Hochreiter, S. (1998, 04). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6, 107-116. doi: 10.1142/S0218488598000094
- Hochreiter, S., & Schmidhuber, J. (1997, 12). Long short-term memory. *Neural computation*, 9, 1735-80. doi: 10.1162/neco.1997.9.8.1735
- 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. Retrieved from <https://physoc.onlinelibrary.wiley.com/doi/abs/10.1113/jphysiol.1952.sp004764> doi: 10.1113/jphysiol.1952.sp004764
- Hopfield, J. J. (1988, Sep.). Artificial neural networks. *IEEE Circuits and Devices Magazine*, 4(5), 3-10. doi: 10.1109/101.8118
- Hsu, J. (2014, October). Ibm's new brain [news]. *IEEE Spectrum*, 51(10), 17-19. doi: 10.1109/MSPEC.2014.6905473
- Hume, D. (1779). *Dialogues concerning natural religion [electronic resource]*. by david hume, esq (The second edition. ed.) [Book, Online]. London.
- Hwang, K., & Sung, W. (2015). Single stream parallelization of generalized lstm-like rnns on a gpu. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1047-1051.
- Indiveri, G., Linares-Barranco, B., Hamilton, T., van Schaik, A., Etienne-Cummings, R., Delbruck, T., ... Boahen, K. (2011). Neuromorphic silicon neuron circuits. *Frontiers in Neuroscience*, 5, 73. Retrieved from <https://www.frontiersin.org/article/10.3389/fnins.2011.00073> doi: 10.3389/fnins.2011.00073
- Izhikevich, E. M. (2003, Nov). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14(6), 1569-1572. doi: 10.1109/TNN.2003.820440
- Izhikevich, E. M. (2004, Sep.). Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, 15(5), 1063-1070. doi: 10.1109/TNN.2004.832719

- Jain, A. K., Jianchang Mao, & Mohiuddin, K. M. (1996, March). Artificial neural networks: a tutorial. *Computer*, 29(3), 31-44. doi: 10.1109/2.485891
- Jiang, F., Jiang, Y., Zhi, H., Dong, Y., Li, H., Ma, S., ... Wang, Y. (2017). Artificial intelligence in healthcare: past, present and future. *Stroke and Vascular Neurology*, 2(4), 230–243. Retrieved from <https://svn.bmj.com/content/2/4/230> doi: 10.1136/svn-2017-000101
- Jolivet, R., Rauch, A., & Gerstner, W. (2005, 01). Integrate-and-fire models with adaptation are good enough: Predicting spike times under random current injection. *Advances in Neural Information Processing Systems*.
- Karim, F., Majumdar, S., Darabi, H., & Harford, S. (2018, 01). Multivariate lstm-fcns for time series classification. *Neural Networks*, 116. doi: 10.1016/j.neunet.2019.04.014
- Kasabov, N. (2016). Evolving spatio-temporal data machines based on the neucube neuromorphic framework: Design methodology and selected applications. *Neural Networks*, 78, 1 - 14. (Special Issue on)
- Kasabov, N. K. (2019). Evolving spiking neural networks. In *Time-space, spiking neural networks and brain-inspired artificial intelligence* (pp. 169–199). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from https://doi.org/10.1007/978-3-662-57715-8_5 doi: 10.1007/978-3-662-57715-8_5
- Kheradpisheh, S. R., Ganjtabesh, M., Thorpe, S. J., & Masquelier, T. (2018). Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, 99, 56 - 67. Retrieved from <http://www.sciencedirect.com/science/article/pii/S0893608017302903> doi: <https://doi.org/10.1016/j.neunet.2017.12.005>
- Kim, T. Y., Oh, K. J., Kim, C., & Do, J. D. (2004). Artificial neural networks for non-stationary time series. *Neurocomputing*, 61, 439 - 447. Retrieved from <http://www>

- .sciencedirect.com/science/article/pii/S0925231204002371 (Hybrid Neuro-computing: Selected Papers from the 2nd International Conference on Hybrid Intelligent Systems) doi: <https://doi.org/10.1016/j.neucom.2004.04.002>
- Kistler, W., Gerstner, W., & van Hemmen, L. (2000, 09). Reduction of the hodgkin-huxley equations to a single-variable threshold model. *Neural Computation*, 9. doi: 10.1162/neco.1997.9.5.1015
- Klonowski, W. (2009, 02). Everything you wanted to ask about eeg but were afraid to get the right answer. *Nonlinear biomedical physics*, 3, 2. doi: 10.1186/1753-4631-3-2
- Kulkarni, S. R., & Rajendran, B. (2018). Spiking neural networks for handwritten digit recognition - supervised learning and network optimization. *Neural networks : the official journal of the International Neural Network Society*, 103, 118-127.
- Kumar, S., Sharma, A., & Tsunoda, T. (2019, 12). Brain wave classification using long short-term memory network based optical predictor. *Scientific Reports*, 9. doi: 10.1038/s41598-019-45605-1
- Le, Q. V., Zou, W. Y., Yeung, S. Y., & Ng, A. Y. (2011, June). Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Cvpr 2011* (p. 3361-3368). doi: 10.1109/CVPR.2011.5995496
- LeCun, Y., Bengio, Y., & Hinton, G. (2015, 05). Deep learning. *Nature*, 521, 436-44. doi: 10.1038/nature14539
- Legenstein, R., Naeger, C., & Maass, W. (2005). What can a neuron learn with spike-timing-dependent plasticity? *Neural Computation*, 17(11), 2337-2382. Retrieved from <https://doi.org/10.1162/0899766054796888> doi: 10.1162/0899766054796888
- Li, B.-h., Hou, B.-c., Yu, W.-t., Lu, X.-b., & Yang, C.-w. (2017, Jan 01). Applications of artificial intelligence in intelligent manufacturing: a review. *Frontiers of Information Technology & Electronic Engineering*, 18(1), 86-96. Retrieved from <https://doi.org/10.1631/FITEE.1601885> doi: 10.1631/FITEE.1601885

REFERENCES

- Li, X., Qin, T., Yang, J., Hu, X., & Liu, T.-Y. (2016). Lightrnn: Memory and computation-efficient recurrent neural networks. In *Nips*.
- Li, Z., Ding, C., Wang, S., Wen, W., Zhuo, Y., Liu, C., ... Wang, Y. (2019, 3 26). E-rnn: Design optimization for efficient recurrent neural networks in fpgas. In *Proceedings - 25th ieee international symposium on high performance computer architecture, hpc 2019* (pp. 69–80). Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/HPCA.2019.00028
- Lichman, M. (n.d.). *Uci machine learning repository*. Retrieved from <http://archive.ics.uci.edu/ml>
- Liu, S.-C., & Delbruck, T. (2010). Neuromorphic sensory systems. *Current Opinion in Neurobiology*, 20(3), 288 - 295. Retrieved from <http://www.sciencedirect.com/science/article/pii/S0959438810000450> (Sensory systems) doi: <https://doi.org/10.1016/j.conb.2010.03.007>
- Lo, M.-T., Tsai, P. H., Lin, P.-F., Lin, C., & Hsin, Y.-L. (2009, 07). The nonlinear and nonstationary properties in eeg signals: Probing the complex fluctuations by hilbert-huang transform. *Advances in Adaptive Data Analysis*, 1, 461-482. doi: 10.1142/S1793536909000199
- Ma, Y., & Principe, J. C. (2019). A taxonomy for neural memory networks. *IEEE Transactions on Neural Networks and Learning Systems*, 1-14. doi: 10.1109/TNNLS.2019.2926466
- Maass, W. (1996). Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons. In *Nips*.
- Manning, T., Sleator, R., & Walsh, P. (2013, 12). Biologically inspired intelligent decision making. *Bioengineered*, 5. doi: 10.4161/bioe.26997
- Manuca, R., & Savit, R. (1996). Stationarity and nonstationarity in time series analysis. *Physica D: Nonlinear Phenomena*, 99(2), 134 - 161. Retrieved from <http://www>

- .sciencedirect.com/science/article/pii/S016727899600139X doi: [https://doi.org/10.1016/S0167-2789\(96\)00139-X](https://doi.org/10.1016/S0167-2789(96)00139-X)
- Matsugu, M., Mori, K., Ishii, M., & Mitarai, Y. (2002, Nov). Convolutional spiking neural network model for robust face detection. In *Proceedings of the 9th international conference on neural information processing, 2002. iconip '02.* (Vol. 2, p. 660-664 vol.2). doi: 10.1109/ICONIP.2002.1198140
- McCulloch, W. S., & Pitts, W. (1943, Dec 01). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4), 115–133. Retrieved from <https://doi.org/10.1007/BF02478259> doi: 10.1007/BF02478259
- Mead, C. (1990, Oct). Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10), 1629-1636. doi: 10.1109/5.58356
- Meunier, C., & Segev, I. (2002, 12). Playing the devil's advocate: Is the hodgkin-huxley model useful? *Trends in neurosciences*, 25, 558-63. doi: 10.1016/S0166-2236(02)02278-6
- Mikolov, T., Karafiřt, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010, 01). Recurrent neural network based language model. In (Vol. 2, p. 1045-1048).
- Mitchell, T. M. (1997). Artificial neural networks. *Machine learning*, 45, 81–127.
- Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2012). *Foundations of machine learning*. Mit Press. Retrieved from <http://www.jstor.org/stable/j.ctt5hhcw1>
- Mostafa, H. (2018, July). Supervised learning based on temporal coding in spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 29(7), 3227-3235. doi: 10.1109/TNNLS.2017.2726060
- Mozafari, M., Ganjtabesh, M., Nowzari-Dalini, A., & Masquelier, T. (2019, Jul). Spyketch: Efficient simulation of convolutional spiking neural networks with at most one spike per neuron. *Frontiers in Neuroscience*, 13. Retrieved from <http://dx.doi.org/10.3389/fnins.2019.00625> doi: 10.3389/fnins.2019.00625

- Mozer, M. (1995, 01). A focused backpropagation algorithm for temporal pattern recognition. *Complex Systems*, 3.
- Muñoz-Gutiérrez, P. A., Giraldo, E., Bueno-López, M., & Molinas, M. (2018). Localization of active brain sources from eeg signals using empirical mode decomposition: A comparative study. *Frontiers in Integrative Neuroscience*, 12, 55. Retrieved from <https://www.frontiersin.org/article/10.3389/fnint.2018.00055> doi: 10.3389/fnint.2018.00055
- Nabet, R. B., Bahram; Pinter. (2018). *Sensory neural networks : lateral inhibition*. CRC Press.
- Neil, D., Pfeiffer, M., & Liu, S.-C. (2016). Learning to be efficient: algorithms for training low-latency, low-compute deep spiking neural networks. In *Proceedings of the 31st annual acm symposium on applied computing* (pp. 293–298). New York, NY, USA: ACM. Retrieved from <http://doi.acm.org/10.1145/2851613.2851724> doi: 10.1145/2851613.2851724
- Nowak, J., Taspinar, A., & Scherer, R. (2017). Lstm recurrent neural networks for short text and sentiment classification. In L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, & J. M. Zurada (Eds.), *Artificial intelligence and soft computing* (pp. 553–562). Cham: Springer International Publishing.
- O'Connor, P., Neil, D., Liu, S.-C., Delbruck, T., & Pfeiffer, M. (2013, 10). Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in neuroscience*, 7, 178. doi: 10.3389/fnins.2013.00178
- Oostenveld, R., & Praamstra, P. (2001). The five percent electrode system for high-resolution eeg and erp measurements. *Clinical Neurophysiology*, 112, 713-719.
- Paugam-Moisy, H., Martinez, R., & Bengio, S. (2008). Delay learning and polychronization for reservoir computing. *Neurocomputing*, 71(7), 1143 - 1158. Retrieved from <http://www.sciencedirect.com/science/article/pii/>

- S0925231208000507 (Progress in Modeling, Theory, and Application of Computational Intelligence) doi: <https://doi.org/10.1016/j.neucom.2007.12.027>
- Petro, B., Kasabov, N., & Kiss, R. M. (2019). Selection and optimization of temporal spike encoding methods for spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 1-13. doi: 10.1109/TNNLS.2019.2906158
- Peuquet, D. (1995, 01). An event-based spatiotemporal data model (estdm) for temporal analysis of geographical data. *Int. J. Geographical Information Systems*, 9, 7-24. doi: 10.1080/02693799508902022
- Pfeiffer, M., & Pfeil, T. (2018). Deep learning with spiking neurons: Opportunities and challenges. *Frontiers in Neuroscience*, 12, 774. Retrieved from <https://www.frontiersin.org/article/10.3389/fnins.2018.00774> doi: 10.3389/fnins.2018.00774
- Ponulak, F., & Kasiński, A. (2009, 10). Supervised learning in spiking neural networks with resume: Sequence learning, classification, and spike shifting. *Neural computation*, 22, 467-510. doi: 10.1162/neco.2009.11-08-901
- Poo, M.-m. (2018, 10). Towards brain-inspired artificial intelligence. *National Science Review*, 5(6), 785-785. Retrieved from <https://doi.org/10.1093/nsr/nwy120> doi: 10.1093/nsr/nwy120
- Prechelt, L. (1998). Early stopping - but when? In G. B. Orr & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade* (pp. 55-69). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from https://doi.org/10.1007/3-540-49430-8_3 doi: 10.1007/3-540-49430-8_3
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Neurocomputing: Foundations of research. In J. A. Anderson & E. Rosenfeld (Eds.), (pp. 696-699). Cambridge, MA, USA: MIT Press. Retrieved from <http://dl.acm.org/citation.cfm?id=65669.104451>

- Rumelhart, D. E., & McClelland, J. L. (1987). Learning internal representations by error propagation. In *Parallel distributed processing: Explorations in the microstructure of cognition: Foundations* (p. 318-362). MITP. Retrieved from <https://ieeexplore.ieee.org/document/6302929>
- Schemmel, J., Grubl, A., Meier, K., & Mueller, E. (2006, July). Implementing synaptic plasticity in a vlsi spiking neural network model. In *The 2006 ieee international joint conference on neural network proceedings* (p. 1-6). doi: 10.1109/IJCNN.2006.246651
- Schliebs, S., Hamed, H. N. A., & Kasabov, N. (2011, 11). Reservoir-based evolving spiking neural network for spatio-temporal pattern recognition. In (Vol. 7063, p. 160-168). doi: 10.1007/978-3-642-24958-7_19
- Schrauwen, B., & Van Campenhout, J. (2003, July). Bsa, a fast and accurate spike train encoding scheme. In *Proceedings of the international joint conference on neural networks, 2003.* (Vol. 4, p. 2825-2830 vol.4). doi: 10.1109/IJCNN.2003.1224019
- 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. Retrieved from <http://www.sciencedirect.com/science/article/pii/S0020025517306631> doi: <https://doi.org/10.1016/j.ins.2017.04.017>
- Smith, S. J. M. (2005). Eeg in neurological conditions other than epilepsy: when does it help, what does it add? *Journal of Neurology, Neurosurgery & Psychiatry*, 76(suppl 2), ii8-ii12. Retrieved from https://jnnp.bmj.com/content/76/suppl_2/ii8 doi: 10.1136/jnnp.2005.068486
- Sporea, I., & Grüning, A. (2013, Feb). Supervised learning in multilayer spiking neural networks. *Neural Computation*, 25(2), 473-509. Retrieved from http://dx.doi.org/10.1162/NECO_a.00396 doi: 10.1162/neco_a.00396
- Stein, R., Gossen, E., & Jones, K. (2005, 01). Neuronal variability: Noise or part of the signal? *Nature Reviews Neuroscience*, 6, 389-397.

- Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014, June). Deepface: Closing the gap to human-level performance in face verification. In *2014 IEEE conference on computer vision and pattern recognition* (p. 1701-1708). doi: 10.1109/CVPR.2014.220
- Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., & Maida, A. (2019). Deep learning in spiking neural networks. *Neural Networks*, 111, 47 - 63. Retrieved from <http://www.sciencedirect.com/science/article/pii/S0893608018303332> doi: <https://doi.org/10.1016/j.neunet.2018.12.002>
- Tavanaei, A., & Maida, A. (2015, 06). A minimal spiking neural network to rapidly train and classify handwritten digits in binary and 10-digit tasks. *International journal of advanced research in artificial intelligence*, 4, 1-8. doi: 10.14569/IJARAI.2015.040701
- Thorpe, S. (1990, 01). Spike arrival times: A highly efficient coding scheme for neural networks. *Parallel Processing in Neural Systems and Computers*.
- Thorpe, S., Fize, D., & Marlot, C. (1996, 07). Speed of processing in the human visual system. *Nature*, 381, 520-2. doi: 10.1038/381520a0
- Thorpe, S., & Gautrais, J. (1998). Rank order coding. In J. M. Bower (Ed.), *Computational neuroscience: Trends in research, 1998* (pp. 113-118). Boston, MA: Springer US. Retrieved from https://doi.org/10.1007/978-1-4615-4831-7_19 doi: 10.1007/978-1-4615-4831-7_19
- Thorpe, S. J., Delorme, A., & van Rullen, R. (2001). Spike-based strategies for rapid processing. *Neural networks : the official journal of the International Neural Network Society*, 14 6-7, 715-25.
- Tokody, D., Mezei, I. J., & Schuster, G. (2017). An overview of autonomous intelligent vehicle systems. In K. Jármai & B. Bolló (Eds.), *Vehicle and automotive engineering* (pp. 287-307). Cham: Springer International Publishing.

REFERENCES

- Tomita, T. (1958). Mechanism of lateral inhibition in eye of limulus. *Journal of Neurophysiology*, 21(5), 419-429. Retrieved from <https://doi.org/10.1152/jn.1958.21.5.419> (PMID: 13576184) doi: 10.1152/jn.1958.21.5.419
- Tsiouris, K. M., Pezoulas, V. C., Zervakis, M., Konitsiotis, S., Koutsouris, D. D., & Fotiadis, D. I. (2018). A long short-term memory deep learning network for the prediction of epileptic seizures using eeg signals. *Computers in Biology and Medicine*, 99, 24 - 37. Retrieved from <http://www.sciencedirect.com/science/article/pii/S001048251830132X> doi: <https://doi.org/10.1016/j.compbiomed.2018.05.019>
- Tsividis, Y. (2010, Sep.). Event-driven data acquisition and continuous-time digital signal processing. In *Ieee custom integrated circuits conference 2010* (p. 1-8). doi: 10.1109/CICC.2010.5617618
- Tuckwell, H., & Wan, F. (2005, 06). Time to first spike in stochastic hodgkin–huxley systems. *Physica A: Statistical Mechanics and its Applications*, 351, 427-438. doi: 10.1016/j.physa.2004.11.059
- Valadez-Godínez, S., Sossa, H., & Santiago-Montero, R. (2020). On the accuracy and computational cost of spiking neuron implementation. *Neural Networks*, 122, 196 - 217. Retrieved from <http://www.sciencedirect.com/science/article/pii/S0893608019303065> doi: <https://doi.org/10.1016/j.neunet.2019.09.026>
- VanRullen, R., Guyonneau, R., & Thorpe, S. J. (2005). Spike times make sense. *Trends in Neurosciences*, 28(1), 1 - 4. Retrieved from <http://www.sciencedirect.com/science/article/pii/S0166223604003546> doi: <https://doi.org/10.1016/j.tins.2004.10.010>
- Vitay, J., Dinkelbach, H. U., & Hamker, F. H. (2015, July). Annarchy: a code generation approach to neural simulations on parallel hardware. *Frontiers in Neuroinformatics*.

- Wang, L., Lu, S.-R., & Wen, J. (2017, 05). Recent advances on neuromorphic systems using phase-change materials. *Nanoscale Research Letters*, *12*. doi: 10.1186/s11671-017-2114-9
- Wen, B., & Boahen, K. (2009). A silicon cochlea with active coupling. *IEEE transactions on biomedical circuits and systems*, *3*(6), 444–455.
- White, N. M., & McDonald, R. J. (2002). Multiple parallel memory systems in the brain of the rat. *Neurobiology of Learning and Memory*, *77*(2), 125 - 184. Retrieved from <http://www.sciencedirect.com/science/article/pii/S1074742701940080> doi: <https://doi.org/10.1006/nlme.2001.4008>
- Widrow, B., & Hoff, M. E. (1988). Neurocomputing: Foundations of research. In J. A. Anderson & E. Rosenfeld (Eds.), (pp. 123–134). Cambridge, MA, USA: MIT Press. Retrieved from <http://dl.acm.org/citation.cfm?id=65669.104390>
- Wu, J., Chua, Y., Zhang, M., Li, H., & Tan, K. C. (2018). A spiking neural network framework for robust sound classification. *Frontiers in Neuroscience*, *12*, 836. Retrieved from <https://www.frontiersin.org/article/10.3389/fnins.2018.00836> doi: 10.3389/fnins.2018.00836
- Wöllmer, M., Zhang, Z., Weninger, F., Schuller, B., & Rigoll, G. (2013, May). Feature enhancement by bidirectional lstm networks for conversational speech recognition in highly non-stationary noise. In *2013 ieee international conference on acoustics, speech and signal processing* (p. 6822-6826). doi: 10.1109/ICASSP.2013.6638983
- Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018, Aug). Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, *13*(3), 55-75. doi: 10.1109/MCI.2018.2840738
- Zhang, A., Zhu, W., & Li, J. (2018, 12). Spiking echo state convolutional neural network for robust time series classification. *IEEE Access*, *7*, 4927-4935. doi: 10.1109/ACCESS.2018.2887354

REFERENCES

- Zhang, X., Jiang, P., & Wang, F. (2014, 06). Overtaking vehicle detection using a spatio-temporal crf. In (p. 338-343). doi: 10.1109/IVS.2014.6856546
- Zhang, Z., Zhao, Y., Liao, X.-K., Shi, W., Li, K., Zou, Q., & Peng, S. (2018, 09). Deep learning in omics: a survey and guideline. *Briefings in functional genomics*, 18. doi: 10.1093/bfgp/ely030

Appendix A

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 16, 7680)	0	
permute_1 (Permute)	(None, 7680, 16)	0	input_1[0][0]
conv1d_1 (Conv1D)	(None, 7680, 128)	16512	permute_1[0][0]
batch_normalization_1 (BatchNor	(None, 7680, 128)	512	conv1d_1[0][0]
activation_1 (Activation)	(None, 7680, 128)	0	batch_normalization_1[0][0]
global_average_pooling1d_1 (Glo	(None, 128)	0	activation_1[0][0]
reshape_1 (Reshape)	(None, 1, 128)	0	global_average_pooling1d_1[0][0]
dense_1 (Dense)	(None, 1, 8)	1024	reshape_1[0][0]
dense_2 (Dense)	(None, 1, 128)	1024	dense_1[0][0]
multiply_1 (Multiply)	(None, 7680, 128)	0	activation_1[0][0] dense_2[0][0]
conv1d_2 (Conv1D)	(None, 7680, 256)	164096	multiply_1[0][0]
batch_normalization_2 (BatchNor	(None, 7680, 256)	1024	conv1d_2[0][0]
activation_2 (Activation)	(None, 7680, 256)	0	batch_normalization_2[0][0]
global_average_pooling1d_2 (Glo	(None, 256)	0	activation_2[0][0]
reshape_2 (Reshape)	(None, 1, 256)	0	global_average_pooling1d_2[0][0]
dense_3 (Dense)	(None, 1, 16)	4096	reshape_2[0][0]
dense_4 (Dense)	(None, 1, 256)	4096	dense_3[0][0]
multiply_2 (Multiply)	(None, 7680, 256)	0	activation_2[0][0] dense_4[0][0]
conv1d_3 (Conv1D)	(None, 7680, 128)	98432	multiply_2[0][0]
masking_1 (Masking)	(None, 16, 7680)	0	input_1[0][0]
batch_normalization_3 (BatchNor	(None, 7680, 128)	512	conv1d_3[0][0]
attention_lstm_1 (AttentionLSTM	(None, 8)	553328	masking_1[0][0]
activation_3 (Activation)	(None, 7680, 128)	0	batch_normalization_3[0][0]
dropout_1 (Dropout)	(None, 8)	0	attention_lstm_1[0][0]
global_average_pooling1d_3 (Glo	(None, 128)	0	activation_3[0][0]
concatenate_1 (Concatenate)	(None, 136)	0	dropout_1[0][0] global_average_pooling1d_3[0][0]
dense_5 (Dense)	(None, 2)	274	concatenate_1[0][0]
Total params: 844,930			
Trainable params: 843,906			
Non-trainable params: 1,024			

Figure A.1: Final Keras Model Description for Schizophrenia State-of-the-Art RNN Classifier.

APPENDIX A.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 64, 256)	0	
permute_1 (Permute)	(None, 256, 64)	0	input_1[0][0]
conv1d_1 (Conv1D)	(None, 256, 128)	65664	permute_1[0][0]
batch_normalization_1 (BatchNor	(None, 256, 128)	512	conv1d_1[0][0]
activation_1 (Activation)	(None, 256, 128)	0	batch_normalization_1[0][0]
global_average_pooling1d_1 (Glo	(None, 128)	0	activation_1[0][0]
reshape_1 (Reshape)	(None, 1, 128)	0	global_average_pooling1d_1[0][0]
dense_1 (Dense)	(None, 1, 8)	1024	reshape_1[0][0]
dense_2 (Dense)	(None, 1, 128)	1024	dense_1[0][0]
multiply_1 (Multiply)	(None, 256, 128)	0	activation_1[0][0] dense_2[0][0]
conv1d_2 (Conv1D)	(None, 256, 256)	164096	multiply_1[0][0]
batch_normalization_2 (BatchNor	(None, 256, 256)	1024	conv1d_2[0][0]
activation_2 (Activation)	(None, 256, 256)	0	batch_normalization_2[0][0]
global_average_pooling1d_2 (Glo	(None, 256)	0	activation_2[0][0]
reshape_2 (Reshape)	(None, 1, 256)	0	global_average_pooling1d_2[0][0]
dense_3 (Dense)	(None, 1, 16)	4096	reshape_2[0][0]
dense_4 (Dense)	(None, 1, 256)	4096	dense_3[0][0]
multiply_2 (Multiply)	(None, 256, 256)	0	activation_2[0][0] dense_4[0][0]
conv1d_3 (Conv1D)	(None, 256, 128)	98432	multiply_2[0][0]
masking_1 (Masking)	(None, 64, 256)	0	input_1[0][0]
batch_normalization_3 (BatchNor	(None, 256, 128)	512	conv1d_3[0][0]
attention_lstm_1 (AttentionLSTM	(None, 8)	18800	masking_1[0][0]
activation_3 (Activation)	(None, 256, 128)	0	batch_normalization_3[0][0]
dropout_1 (Dropout)	(None, 8)	0	attention_lstm_1[0][0]
global_average_pooling1d_3 (Glo	(None, 128)	0	activation_3[0][0]
concatenate_1 (Concatenate)	(None, 136)	0	dropout_1[0][0] global_average_pooling1d_3[0][0]
dense_5 (Dense)	(None, 2)	274	concatenate_1[0][0]
Total params: 359,554			
Trainable params: 358,530			
Non-trainable params: 1,024			

Figure A.2: Final Keras Model Description for Alcoholism State-of-the-Art RNN Classifier.

APPENDIX A.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 1, 4096)	0	
permute_1 (Permute)	(None, 4096, 1)	0	input_1[0][0]
conv1d_1 (Conv1D)	(None, 4096, 128)	1152	permute_1[0][0]
batch_normalization_1 (BatchNor	(None, 4096, 128)	512	conv1d_1[0][0]
activation_1 (Activation)	(None, 4096, 128)	0	batch_normalization_1[0][0]
global_average_pooling1d_1 (Glo	(None, 128)	0	activation_1[0][0]
reshape_1 (Reshape)	(None, 1, 128)	0	global_average_pooling1d_1[0][0]
dense_1 (Dense)	(None, 1, 8)	1024	reshape_1[0][0]
dense_2 (Dense)	(None, 1, 128)	1024	dense_1[0][0]
multiply_1 (Multiply)	(None, 4096, 128)	0	activation_1[0][0] dense_2[0][0]
conv1d_2 (Conv1D)	(None, 4096, 256)	164096	multiply_1[0][0]
batch_normalization_2 (BatchNor	(None, 4096, 256)	1024	conv1d_2[0][0]
activation_2 (Activation)	(None, 4096, 256)	0	batch_normalization_2[0][0]
global_average_pooling1d_2 (Glo	(None, 256)	0	activation_2[0][0]
reshape_2 (Reshape)	(None, 1, 256)	0	global_average_pooling1d_2[0][0]
dense_3 (Dense)	(None, 1, 16)	4096	reshape_2[0][0]
dense_4 (Dense)	(None, 1, 256)	4096	dense_3[0][0]
multiply_2 (Multiply)	(None, 4096, 256)	0	activation_2[0][0] dense_4[0][0]
conv1d_3 (Conv1D)	(None, 4096, 128)	98432	multiply_2[0][0]
masking_1 (Masking)	(None, 1, 4096)	0	input_1[0][0]
batch_normalization_3 (BatchNor	(None, 4096, 128)	512	conv1d_3[0][0]
attention_lstm_1 (AttentionLSTM	(None, 8)	295280	masking_1[0][0]
activation_3 (Activation)	(None, 4096, 128)	0	batch_normalization_3[0][0]
dropout_1 (Dropout)	(None, 8)	0	attention_lstm_1[0][0]
global_average_pooling1d_3 (Glo	(None, 128)	0	activation_3[0][0]
concatenate_1 (Concatenate)	(None, 136)	0	dropout_1[0][0] global_average_pooling1d_3[0][0]
dense_5 (Dense)	(None, 2)	274	concatenate_1[0][0]
Total params: 571,522			
Trainable params: 570,498			
Non-trainable params: 1,024			

Figure A.3: Final Keras Model Description for Epilepsy State-of-the-Art RNN Classifier