Sean Tomlinson – C1673374

CM3203: One Semester Individual Project

Supervised by Dr Matthias Treder

Final Report

Evaluating Deep Learning Techniques for Automated Sleep Stage Scoring of EEG Data

---

10 May, 2019

Abstract

Sleep stage scoring is the first step in the quantitative analysis of polysomnographic recordings. Scoring is usually performed manually by at least two qualified sleep technologists. Manual scoring carries limitations of inter-rate reliability and time demands of sleep technologists. Recent developments have considered the use of artificial neural networks to solve this problem autonomously. This project evaluates the use of 8 different neural network models, specifically taking inspiration from newer developments in the wider fields of time series classification and digital signal processing. Model design is discussed in detail and the final evaluations are made from experiments on two datasets from Cardiff University's Neuroscience and Psychology of Sleep Laboratory. The resulting system designed for the project is a neural network test-ground where models are capable of classifying sleep stages to a high degree of accuracy.

Acknowledgements

I would first like to thank Julia Schneider and Anne Kooperman for providing the project with their personal research datasets and supporting me throughout the project. Next, I would like to thank all of my family and friends; without you I would not be the person I am today. Specifically, my parents Paul and Sarah, for raising me and supporting me throughout university, through thick and thin. Finally, my project supervisor, Matthias Treder, who was an exceptional source of knowledge and who kept the project on track from start to finish.

Contents

1. Introduction

1.1 Overview & Motivation

Roughly one third of human life is spent sleeping; it is as essential for survival as food and water. Yet the biological purposes of sleep remain shrouded in mystery. The field of sleep research is a core component of Medicine, Neuroscience and Psychology. The majority of Britons (74%) now sleep for seven hours or less per night, with more than a quarter (30%) experiencing poor quality sleep on a regular basis [1]. Therefore, there is urgent need for improved methods to diagnose sleep-related disorders. Sleep research is also a growing area outside of sleep medicine, with recent studies focusing on the relationship between sleep, memory [2] and cognitive ability [3].

The current standard and first step in both sleep medicine and sleep research is visual sleep stage scoring, performed by two sleep technologists. The technologists use a participant's *polysomnographic* (PSG) data which is recorded by specialist equipment, during an overnight sleep study. A PSG data recording measures a range of physiological activity parameters, such as Electroencephalogram (EEG) data, throughout the night. Manual sleep stage scoring carries limitations of inter-rate reliability and time demands of sleep technologists. An 8-hour PSG study can take over two hours of technologist's time to generate into a complete sleep report. Much of that time is spent on manual annotation. This project aims to find a solution to automate this process, in order to alleviate those manual efforts, and to assist or even replace manual stage scoring.

In its essence, sleep stage scoring from a PSG sleep study is a *Time Series Classification* (TSC) problem. Deep learning methods, particularly *Artificial Neural Networks* (ANNs), are known to have much success in modern classification problems such as ImageNet [4]. Specifically, the field of TSC is well researched; problems such as human activity classification [5][6] being one example. In this project, I will evaluate the use of a variety of ANN models with the purpose of developing a deep learning system for sleep stage classification. I am working with Cardiff University's *Neuroscience and Psychology of Sleep* (NaPS) laboratory and the *Cardiff University Brain Research Imaging Centre* (CUBRIC) to provide a system that can aid in the scoring of sleep data to a good degree of accuracy.

1.2 Project Objectives

The project revolved around these core objectives taken from my initial plan. Some changes that have been made are discussed in **Section 3.1**:

- Develop the back-end tools required to test several ANN architectures for use in a system that can aid sleep experts by automatically classifying EEG sleep recordings.
- Design and implement a custom *Convolutional Neural Network* (CNN) that will be used to classify EEG sleep recordings (based on problem specific literature).
- Design and implement additional, custom ANN models that have not previously been applied to this problem in literature.
- Gain knowledge and experience in the fields of computer science and data science. In particular, with regards to deep learning.
- Determine whether the use of ANNs is an appropriate solution to the problem of classifying EEG sleep recordings.

1.3 Report structure

The report is organised as follows. **Section 2** contains the background knowledge required to understand the report. This includes the specialist concepts behind sleep research, TSC, *Digital Signal Processing* (DSP), and deep learning. A background of related literature is also discussed, along with a description of the tools applied to the development of the project. **Section 3** goes into detail of the approach for the implementation of the system; beginning with how the goal of the project evolved over time, going on to explain why certain design choices were made, and ending with the model design process. The resulting 8 models from the design process are defined and described in **Section 4** along with a detailed description of the implementation of the system. Additionally, the structure of the experimentation phase is described in this section. **Section 5** presents the results of these experiments, and evaluates their significance. On the back of this, future work is proposed in **Section 6** and finally conclusions are drawn and reflections are made in **Sections 7 & 8**.

2. Background

2.1 Context and Concepts

This report assumes the reader has the knowledge of the average bachelor computer science graduate. Therefore, in order to understand the majority of the project, there are some concepts that need to first be explained in further detail.

Time Series Classification

Before delving into detail on the datasets studied, and introducing the concepts behind the ANN designs, I will introduce some formal background definitions for TSC [7]:

- **Definition 1** – A univariate time series $X = [x_1, x_2, \dots, x_T]$ is an ordered set of real values. The length of X is equal to the number of real values T.
- **Definition 2** – An M-dimensional multivariate time series, $X = [X_1, X_2, \dots, X_T]$ consists of M different univariate time series with $X^i \in \mathbb{R}^T$.
- **Definition 3** – For a dataset containing K classes, the one-hot label vector Y_i is a vector of length K where each element $j \in [1, K]$ is equal to 1 if the class of X_i is j and 0 otherwise.
- **Definition 4** – A dataset $D = \{(X_1, Y_1), (X_2, Y_2), \dots, (X_T, Y_T)\}$ is a collection of pairs (X_i, Y_i) where X_i is a time series with Y_i as its corresponding one-hot label vector.

The task of TSC consists of training a classifier on a dataset D in order to map from the space of possible inputs to a probability distribution over the class variable values (labels). I will be training a variety of ANN architectures as classifiers for my sleep dataset, and evaluate their performance. A clear demonstration of this framework can be seen in **Figure 1**.

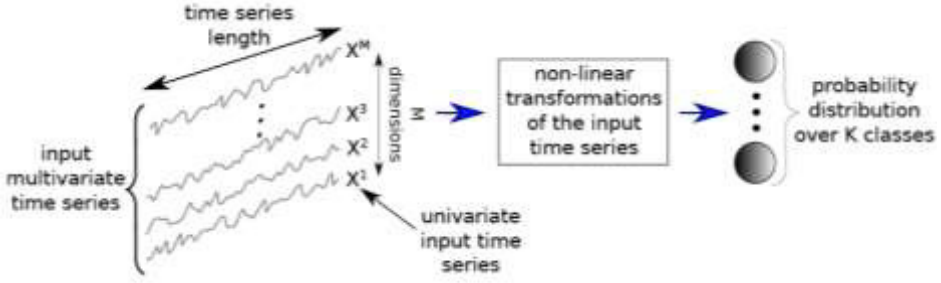

***Figure 1**: The unified deep learning framework for time series classification [7].*

Sleep Stages and Scoring Techniques

During natural human sleep, we oscillate between three phases in a cycle – *Wake* (W), *Rapid Eye Movement* (REM), and *Non-REM* (NREM) sleep. The NREM phase is further broken down into 3 sub-categories, which leads to the scientific standard of five stages sleep: W, REM, N1, N2, N3 (the final 3 are sometimes referred to as S1, S2, and S3). Each sleep stage is thought to correlate with different functions within the brain. For example, REM sleep is where most dreams occur, this is accompanied by random/rapid movement of the eyes and low muscle tone throughout the body.

During a sleep study, PSG data is recorded which is then scored by two sleep technologists. Common to all PSG monitoring are three physiological parameters:

- ***Electroencephalogram* (EEG)** measures changes of electrical potential in the brain from electrodes placed along a participant's scalp.
- ***Electro-Oculogram* (EOG)** measures changes of electrical potential to record muscle movements from each eye.
- ***Electromyogram* (EMG)** measures muscle twitch potentials, commonly from the chin/neck area, or a limb.

These electrical time series signals are separated into 30 second segments known as "epochs". Each epoch is classified to a sleep stage according to reference nomenclature, such as the guidelines of the *American Academy of Sleep Medicine* (AASM) [8]. The participant's data can then be summarised in a statistical diagram known as a "sleep histogram", that indicates the progression of sleep stages and cycles throughout the study, an example of which can be seen in **Figure 2**.

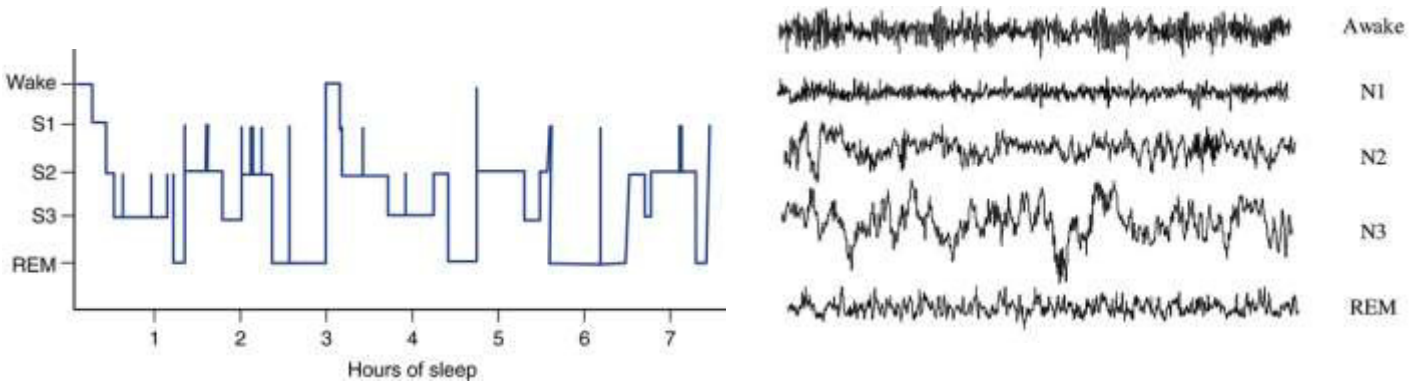

***Figure 2**: An example of a sleep histogram (left), and a sample of PSG data for each sleep stage (right).*

Deep Learning

Deep learning is a subfield of machine learning, concerned specifically on ANN methods. ANNs were initially theorised over 70 years ago, inspired by the biological nervous system [9]. ANNs have been a revolutionary development in machine learning for a wide variety of applications over the past 30 years. This improvement is thanks to developments in learning algorithms and hardware. Specifically, the field of data classification has benefitted the most from this development, with applications such as image classification [10], video classification [11], and automated cancer detection/ classification [12] all breaking accuracy and efficiency records thanks to ANNs. ANNs consist of a system of nodes modelling neurons and synapses [13], and are preferred to machine learning models due to their impressive ability to learn non-linear functions. ANNs are fed data and "learn" by adjusting the synaptic connections between nodes. An ANN is made up of multiple layers of nodes, as seen in **Figure 3**. An input layer (dimension determined by dimension of input dataset), the hidden layers (dimensions decided by network architect), and the output layer (for classification, dimension is determined by the number of output classes). ANNs make predictions through a process known as forward propagation [14], and train through a process known as back propagation [15].

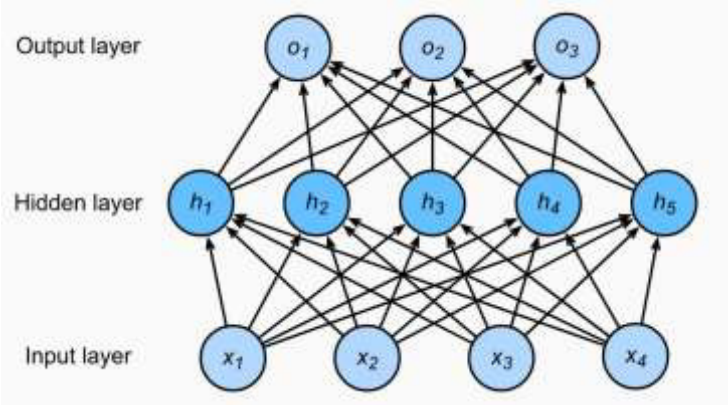

***Figure 3**: The basic structure of an ANN.*

CNNs introduced in 1980 [16], are a specialised type of neural network for processing data that has a grid-like topology (e.g. 1D time series data). CNNs use convolutional layers to act as filters that learn to detect specific features in the data, see **Figure 4**. This is achieved by weight sharing. The deeper layers in the network detect more complex features. Convolutional layers are followed by pooling layers. The function of a pooling layer is to lower the number of parameters and computations in the network by progressively reducing the spatial size of the representation [17]. Convolutional networks have had great success with multiple applications to TSC as seen in this review [18].

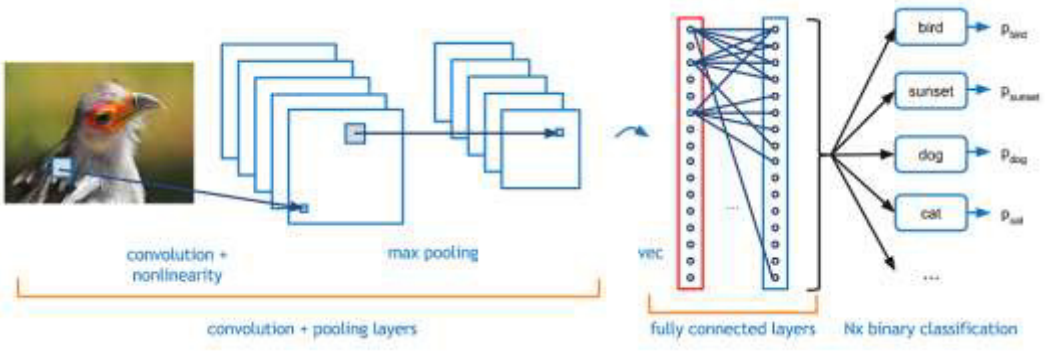

***Figure 4**: The flow of data through a typical CNN [74].*

A *Residual Neural Network* (ResNet) is a kind of ANN that builds on constructs known from pyramidal cells in the cerebral cortex [19]. After a certain point, adding more layers to a network will usually hinder performance. Usually this is attributed to overfitting. However, neural networks suffer from a variety of other problems when greatly expanded, such as the vanishing gradient problem [20]. The culmination of these issues is known as the degradation problem [21]. ResNets solve these problems by utilising skip connections, or short-cuts to jump over some of the convolutional layers in the network, as seen in **Figure 5**.

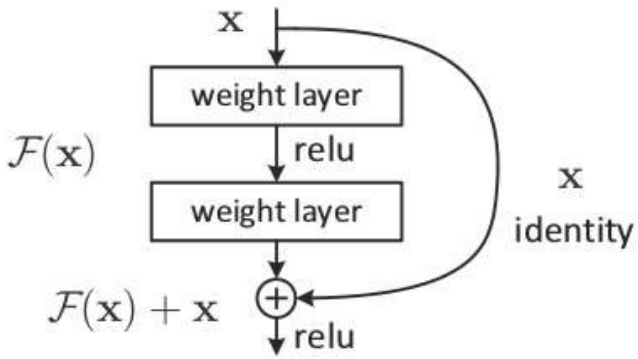

***Figure 5**: The structure of a residual block [20].*

A *Recurrent Neural Network* (RNN) [22] is a specialised type of neural network for processing sequences of values. This allows them to exhibit temporal dynamic behaviour. RNNs use their internal state to process sequences of inputs, which gives them a "memory" in order to store previous computations. The specific type of RNNs I have used in this project are *Long Short-Term Memory* (LSTM), introduced in 1997 [23]. They overcome the problem of handling long-term dependencies with which traditional RNNs struggle. LSTMs have been known to work well with time series data. They have been applied to such problems as predicting the Chinese stock market [24] and speech classification [25]. The difference in structure between a standard RNN cell and an LSTM cell is shown in **Figure 6**.

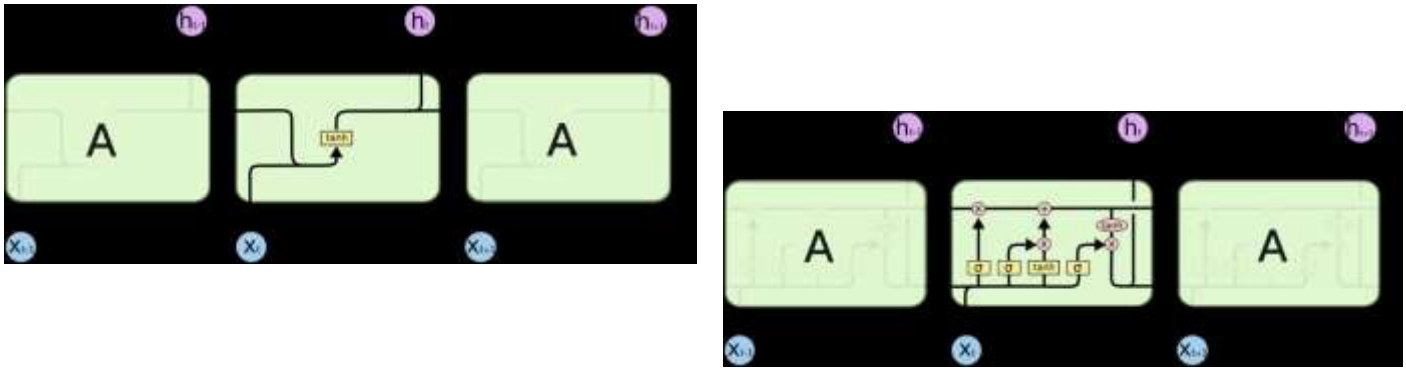

***Figure 6**: The repeating module in a standard RNN (left), compared to the repeating module in an LSTM (right) [75].*

Digital Signal Processing

DSP is the theory of using computational systems to perform signal processing operations. Signals processed in this manner are a sequence of numbers that represent samples of a continuous variable in a domain such as time, space, or frequency. ANNs are being widely explored in DSP due to the fact that they have the advantage of being able to apply self-learnt transformations in order to process a target signal [26][27]. The basic transformations that ANNs apply to incoming 1D data can be related to concepts in DSP theory. For example, convolutional layers are akin to applying the Fourier transform in order to filter and focus certain frequency waves; see convolution theory [28]. Additionally, the *Rectifier Linear Unit* (ReLU) activation function is similar to using power diodes as half-wave rectifiers to remove the negative parts of a signal [29]. As my models will have to process and learn from digital signals, I feel that it is suitable that model design will take inspiration from DSP.

2.2 Datasets

I have access to a few datasets from Cardiff University's NaPS lab. Here I describe in detail the two datasets that were selected for use in the project.

	Wake	N1	N2	N3	REM
DS1	8.2%	5.2%	45.5%	20.4%	20.6%
DS2	6.3%	9.8%	45.7%	18.4%	20.7%

***Table 1**: Exact distribution of classes in both datasets.*

Dataset 1 (DS1)

This is a dataset of over 345 hours of labelled PSG sleep recordings. The recordings were collected from 20 different participants, where each participant was studied for 2 nights. The recordings were scored by two independent sleep technicians using the standard guidelines of the AASM. The dataset was originally recorded to study the effects of stimulation anticipation on healthy sleep and cognition. In total, there are 40 sleep recordings which translates to 41,470 individually scored epochs. The split of the 5 classes is as follows (see **Table 1**): Wake – 3,420; N1 – 2,153; N2 – 18,863; N3 – 8,487; REM – 8,547. The data consists of 10 EEG channels, 1 EOG channel, and 1 EMG channel all sampled at 200Hz. This gives a dimensionality for each epoch of 6000 x 12. The data was originally filtered between 0.3-35 Hz (EEG & EOG) and 10-90 Hz (EMG).

Dataset 2 (DS2)

This is a dataset of over 180 hours of labelled PSG sleep recordings. The recordings were collected from 22 different participants, where each participant was studied for a single night. The recordings were scored by two independent sleep technicians using the standard guidelines of the AASM. Additionally, this dataset was scored by Z3Score, a commercial sleep scoring solution which is also based on ANN technology (the Z3 software is explained in detail later in this section); I will be comparing my results to this commercial software for one metric of the projects success. The dataset was originally recorded to examine the effects of targeted memory reactivation of an associative memory task during REM and N3. In total, there are 22 sleep recordings which translates to 21,978 individually scored epochs. The split of the 5 classes is as follows (see **Table 1**): Wake – 1,392; N1 – 1,927; N2 – 10,046; N3 – 4,057; REM – 4,556. The data consists of 15 EEG channels, 2 EOG channel, and 1 EMG channel all sampled at 500Hz. For this project, the data first was down sampled to 200Hz in MATLAB. This gives a dimensionality for each epoch of 6000 x 18. The only pre-processing effect applied to the data was a lowpass filter below 45Hz.

2.2 Methods and Tools

Some non-standard Python packages and development environments were used throughout the project. A brief overview is useful in order to understand what tools were combined for the final system, and to understand how the system overcame environmental constraints that arose during the design and development phases.

APIs and Libraries

Two key libraries that were used for developing, training, and testing ANNs are Keras & TensorFlow. Keras is an open-source ANN library written in Python. It is capable of running on top of TensorFlow. Keras is designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible [30].

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic maths library, and is also used for machine learning applications such as neural networks [31]. Both Keras and TensorFlow are designed to take advantage of NVIDIA CUDA [32] capable modern GPUs in order to accelerate training and testing times for ANNs.

Development Environments

Three different development environments were used to fulfil the differing needs of the project at different stages. The initial testing and data pre-processing were performed on a 2014 MacBook Pro [33] running Anaconda [34] as a package manager for Python, and JupyterLab [35] as a development environment. As the project progressed, testing outgrew this purely CPU based environment so other solutions were needed.

Google Colab [36] is a free, cloud based Jupyter notebook [37], with support for GPU computing. Colab uses one Tesla K80 (compute 3.7, having 2496 CUDA cores, 12GB GDDR5 VRAM). The majority of the project design was performed using Colab. However, Colab has the disadvantage of being incapable of running multiple notebooks per GPU. Additionally, Colab has a paywall for higher storage, which was needed for testing the entirety of both datasets. For the final development and testing, a more powerful solution was needed.

Supercomputing Wales (SCW) is a £15m programme of investment, part-funded by the *European Regional Development Fund* (ERDF) through the Welsh Government, to provide university research teams access to powerful computing facilities to undertake high-profile science and innovation projects within the consortium universities [38]. As a Cardiff University student, I had access to use SCW's "Hawk" system for the final development and testing of my project. Hawk is a supercomputing cluster made up of over 8000 cores, with access to 26 GPUs. The full technical specifications for the system can be found here [39]. Hawk uses a Slurm job scheduler [40], and has inbuilt software packages such as Anaconda and TensorFlow. This development environment met all of the project's needs, it was free, powerful, could process several jobs in parallel, and had large reserves for data storage. The only problem with Hawk was the initial learning curve for scheduling jobs, as the system had a unique architecture.

2.3 Related Work

There have already been several published attempts to solve similar sleep scoring problems using ANNs. Here is a summary of 4 main studies in order to highlight the strengths and weaknesses of each approach, and where my project differs from this current literature.

Study 1) SLEEPNET

"SLEEPNET: Automated Sleep Staging System via Deep Learning" [41] was a paper published in mid 2017 explaining the technical development of SLEEPNET, a specialised clinical tool for automated sleep stage scoring, powered by ANNs. A notable feature of SLEEPNET is that it was developed from a study on a huge private dataset of 10,000 overnight PSG recordings collected from the Massachusetts General Hospital Sleep Laboratory. These recordings are each roughly 8 hours long; this means the dataset contains around 9,600,000 scored epochs. 90% of the data was used for training, and 10% was used for testing. SLEEPNET was then deployed in two clinical environments and case studies were carried out to evaluate the performance of the system. The paper compares a variety of deep learning techniques with more classical machine learning methods such as *Logistic Regression* (LR). The deep learning techniques used are explained in full technical detail. However, the primary focus of the paper is the development of a system, rather than purely for research purposes. There are parts of the paper therefore, that are not relevant to my project, such as the deployment plan. The solution produced in my project is not intended to be an entirely finished application. Rather a research-focused system, exploring the latest developments in deep learning, aimed at this specific problem.

The development section of the paper focuses on 3 ANN models trained across 3 different representations of the dataset. The first of which is the raw time series data. It is worth noting their dataset is made up only using 6 EEG channels, all sampled at 200Hz. This gives a dimensionality for each epoch of 6000x6. This is half the size of DS1, and has arguably much less information, as SLEEPNET only uses EEG data (and no EOG or EMG). Secondly, this EEG signal is represented as a spectrogram using the Fourier transform. In essence, this allows each epoch to be represented as a 29x257 pixel, 2-Dimensional image, which can then be processed by the ANNs. The final representation of the dataset used is a feature vector of expert defined features. Each epoch is condensed into a vector of 96 features, such as power ratios, this is explained in more detail in the paper. These methods of data representation are a novel idea, and have clear benefits such as condensing data size in order to speed up training and testing of models, but they have also been studied in other published works [42] [43].

The models evaluated are one basic CNN, one basic RNN, and a novel hybrid *Recurrent-Convolutional Neural Network* (RCNN). For the CNN, the raw data was averaged from 6 channels to 1. The CNN had 3 convolutional layers with ReLU activation functions. Each layer is followed by a max pooling layer, and is finally

connected to a fully connected layer. The 3 convolutional layers were of sizes 32, 64, and 128 filters all of kernel size 3x3 (or 3x1 in the 1D case). On the raw data the 1D CNN achieved an accuracy of 77.31%. The 2D CNN achieved an accuracy of 77.83% on the spectrogram representation of the dataset. The CNN is the only model that was not tested on all 3 representations of the dataset because applying convolutions to a feature vector would be nonsensical. This reason for the comparative low accuracy, compared to other similar studies, is down to the simplicity of the model, the size of the dataset, and the simplicity of the input data; only using 6 EEG channels.

For the RNN, the input data is fed to the model as a series of features with every timestamp corresponding to a 30-second epoch. The RNN makes a prediction for each epoch, but has a recurrent "memory" where the previous epochs are also taken into account when a prediction is made. The RNN is implemented using 5 layers of 1000 LSTM cells with tanh activation functions, and a dropout probability of 0.9 to avoid overfitting. This architecture for an RNN is fairly complex; RNNs are sensitive to hyperparameter changes, and overfitting. Because the training set is so large, the team behind the paper have the luxury of being able to use so many LSTM cells in each layer. However, a large dropout value is needed to ensure the model, being this complex, does not overfit. The RNN achieves 79.46% accuracy on the raw data, 79.21% accuracy on the spectrogram features, and a notably high accuracy of 85.76% on the expert defined features; the best score in the study. RNNs are known to perform well with smaller feature vectors for applications such as text classification [44], so it is no surprise the model performed best here. The slight improvement over the CNN with regards to the other two representations of the dataset is most likely down to the improved complexity of the model.

For the final model, the team test a novel idea, a hybrid RCNN. This model is much more applicable to my project as I will be implementing novel deep learning ideas for a similar purpose. The inspiration behind the model is to be able to extract spatial features from each epoch, but also to be able to preserve the longer temporal relationships between epochs in the data. A CNN first processes each epoch, to produce a feature vector at each timestep. The features are then processed in sequence by an RNN. The exact specifications for the model architecture are not made clear in the paper, but one can assume they use the same hyperparameters in both parts of the model that were used for the parent models. The entire model is trained together from scratch. In theory this model makes sense, but simply "gluing together" two models that performed well is not the best way to go about this idea. Working on new hyperparameters from scratch would have been a better approach, although being more time consuming. The RCNN outperforms its two parent models on the raw dataset, and the spectrogram features; with accuracies of 79.81% and 81.47% respectively. However, the model only receives an accuracy of 81.67% on the expert defined feature vector. As previously mentioned, it is nonsensical to use a CNN for this; there are no spatial features to extract from a feature vector. It is surprising the accuracy was even that high; this is a good example of why it is beneficial to learn the temporal relations of sleep.

Overall the paper is very useful. It goes on to describe the success of use in clinical environment for aiding medical professionals with sleep stage scoring. However, the accuracy of the system is still too low, in my opinion for the project to have reached its goal of producing an automated sleep scoring system. There is much room for improvement, to explore novel ideas, and to apply newer breakthroughs in ANN TSC to this specific problem.

Study 2) Z3Score

Z3Score [45] is a new, client/server, commercial system for sleep research and scoring. It was designed off the back of a 2018 paper "An end-to-end framework for real-time automatic sleep stage classification" [46]. I have had access to the software, as Z3Score has been used to score DS2. This means there will be a clear comparison between my solutions and this published, commercial software.

The study uses a dataset that is comprised of 11,727 hours of PSG data which translates to 1,403,164 scored epochs. The dataset is the result of combining 4 smaller medical datasets, two of which were used only for validation. Similarly to SLEEPNET, the Z3 project's primary focus is the design of a system rather than purely testing multiple models for research purposes. Therefore, again there are parts of the paper that are not relevant to my project. Although the paper boasts about the system's ability to operate on raw PSG data, a complex feature extraction stage must first take place by the client before any of the data is processed by the ANN on the server side. To summarise the pre-processing stage, the system first receives two EEG channels and an EOG channel for each eye, all sampled at 100Hz. The system averages the EEG signals together. Several filters are then applied to each signal, and a short-time Fourier transform is used to form 32×32 spectrograms (x3 channels) for each epoch.

Unfortunately, the paper lacks the exact technical details of the model architecture for the server-side classification model, as the software is commercial intellectual property. The summary of the details that are included is as follows and are summarised in **Figure 7**. The first stage is made up of a 16-layer, deep CNN. The CNN processes the spectrograms for each epoch and outputs the probability for each of the 5 sleep stages. Finally, a *Multilayer Perceptron* (MLP) is used to make the final guess of the sleep stage along with the relative confidence of the estimate which are then sent back to the client. This is clearly a complex approach, but the model lacks the ability of learning the longer temporal relations of sleep that are more suited to the use of an RNN. This design system has the advantage of also outputting the model's confidence of each classification estimate, which is very useful while tweaking the design of a model, to see where the mistakes are made.

The model was tested on 3 separate sections of the dataset. The first was a section overlapping the same dataset used in the training of the model in which it achieved an accuracy of 89.8%. This is on par with other cutting-edge solutions, however the training and testing sets are from overlapping studies, so it is likely that the model

has overfit to this dataset. Luckily, the team performs two more tests on two smaller, completely separate datasets. For the first, they achieve an accuracy of 81.4%. Showing substantial agreement, the model is clearly transferable, however the results are far from outstanding. On the final dataset, the accuracy achieved is 71.8%, a much poorer result. This however, is because the dataset is made up of a study on patients with Parkinson's disease who often exhibit REM behaviour sleep disorder [47]. It is a useful comparison to make but it highlights the fact that ANNs commonly overfit, and they are not highly transferable, especially when classification patterns and rules change significantly.

Overall, the study is interesting. It signifies that more complex deeper models might be the best way forward in order to learn the complexities in the sleep data. The CNN used does not learn directly from the raw data, rather from a spectrogram produced by a complex pre-processing stage. My project aims to train networks directly using the raw data as I believe this area of research is much more interesting.

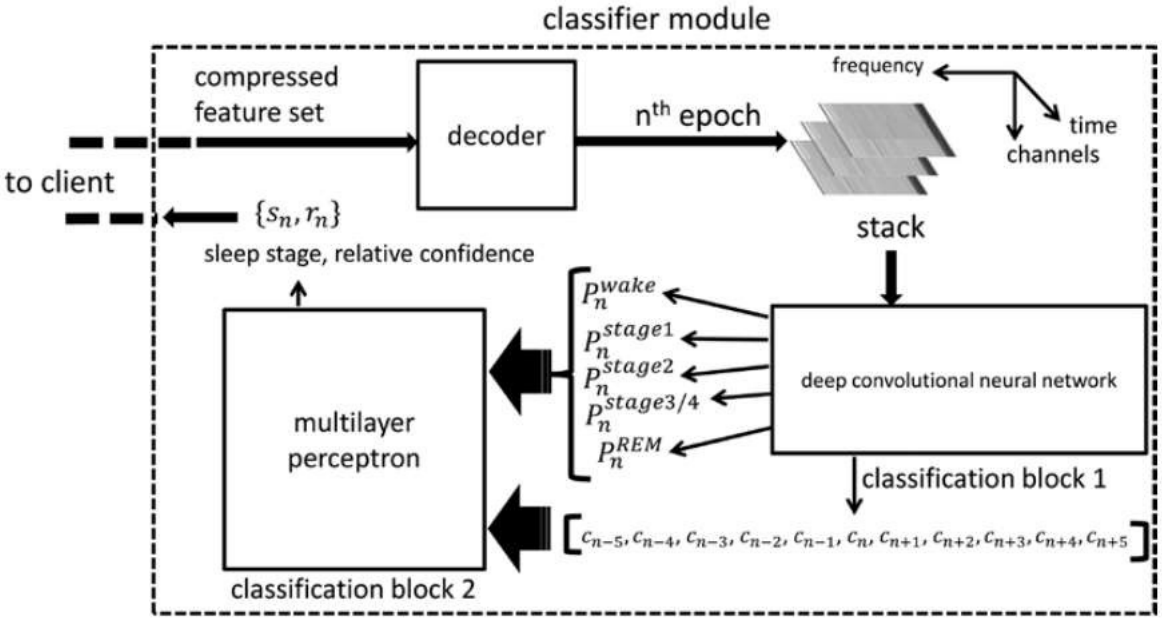

***Figure 7**: Overview of the classifier module used in Z3Score [46].*

Study 3) Automatic Human Sleep Stage Scoring Using Deep Neural Networks

This paper, published in late 2018 [48], implements two types of ANN models and tests them across two datasets. The models are also compared to a *Random Forest* (RF) model [49], a more basic type of machine learning algorithm used for classification. The paper achieves state of the art results, however, they have to be taken with a grain of salt, because the datasets used are small. The first dataset contains PSG recordings of 18 healthy young males, across 3 separate nights (54 PSGs in total). The second dataset contains PSG recordings from 43 nights of patients with either Hypersomnia or Narcolepsy. The study has the advantage of being able to compare the models on patients with a sleep condition, as well as healthy participants, to assess if the models are robust across the two differing sets. The technical details of both datasets can be seen in full detail in the methods section of the study. The datasets were recorded in separate labs with different technical specifications so both datasets had to be standardised before they could be used by the models.

An interesting focus of the study is the comparison between a feature-based network and a raw data based network. 6 different network configurations were explored for a feature-based LSTM network, all of which had 3 hidden layers, see **Figure 8**. Seven different network configurations were explored for a raw data based CNN-LSTM network. Here, very deep CNNs were used on the raw data, some with residual connections, before the results were processed by two bidirectional LSTM layers of 32 nodes each. In some models the input channels were separated, and in some they were not. For a better overview of the CNN-LSTM networks see **Figure 9**.

It is most beneficial to focus on the results of the healthy participant dataset, as this is most relevant to my project. The paper choses the performance measure of Cohen's kappa, a measure of inter-rate agreement that takes into account the possibility of the agreement occurring by chance, to evaluate the model performance Exact values can be found in the paper's supplementary material [50]. All of models perform well and beat the RF methods but the study is careful to make any concrete conclusions. The study is aware of some of the problems with the experiments such as the small size of the datasets and the use of the same datasets for training and testing. However, the main conclusions drawn are that the raw data based methods are superior to that of feature extraction. Another trend in the results is that the models in which the input channels are first separated perform slightly better than all other models. From these conclusions, my models will be applying some of these novel ideas.

The study is at the cutting edge of this research problem. It solidified that raw data methods will be the path forward to explore. Splitting the input channels had some success so this is another novel idea that my project will investigate. Similarly to this study, my datasets are both small, because of this, one of the experiments carried out in the project will be dedicated to testing transferability of the models.

A difference from this study however, is that my project has access to up to 18 channels of input data compared to 6.

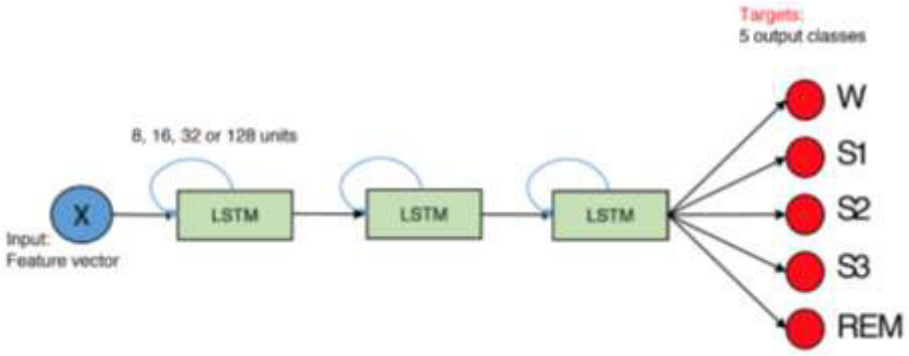

***Figure 8**: Feature based LSTM network explored in **Study 3** [48].*

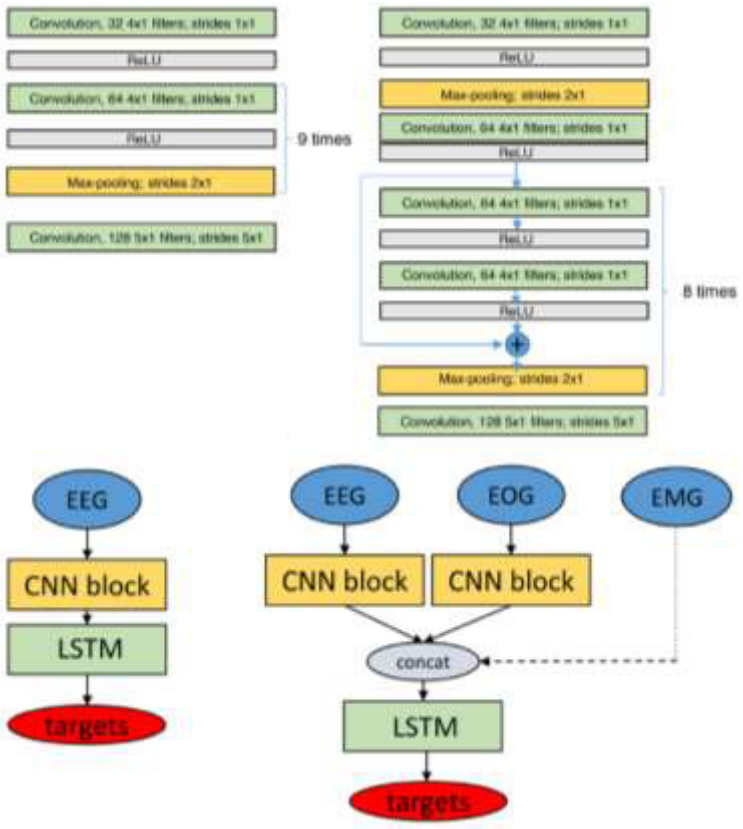

***Figure 9**: Raw data based CNN-LSTM networks explored in **Study 3** [48].*

Study 4) Deep Learning for Time Series Classification: A Review

This paper, published in late 2018 [18], aims to provide a comprehensive overview of deep learning methods for TSC in general. The paper is no attempt to solve a sleep stage scoring problem, however some of the datasets do contain EEG data. In total, the team trained 8,730 deep learning models across 97 time series datasets, making it the most exhaustive study of ANNs for TSC to date. 12 of these datasets contain multivariate time series data. Out of all the experiments, the paper chooses to describe 9 in detail and evaluates their results on each dataset. Before the models are introduced, the paper first discusses the theory behind TSC and ANNs in great detail. The paper then continues by justifying the approach used for model design.

Listing the hyperparameters for every model would be overwhelming, so the specifics for each model can be found in the paper. I will go into detail on the 3 best performing models. The first of which is a *Fully Convolutional Network* (FCN) [51]. FCNs are similar to CNNs, except that they omit local pooling layers. This means that the length of a time series is kept unchanged throughout the convolutions. In addition, one of the main characteristics of this architecture is the replacement of the traditional final FC layer with a *Global Average Pooling* (GAP) layer. The next model is a deep ResNet. The main characteristic of a ResNet is the shortcut residual connection between consecutive convolutional layers used so data can skip layers and prevent the vanishing gradient problem. The final model to discuss is an Encoder [52]. An Encoder is a hybrid deep CNN whose architecture is inspired by an FCN with the main difference being where the GAP layer is replaced with an attention layer [53].

The models are tested on two standard, publicly available TSC data archives, one univariate and one multivariate. It is most beneficial to focus on the multivariate results as that is most applicable to my project. The multivariate Baydogan's archive [54] contains 13 TSC datasets, one of which was left out for this review because of memory requirements. When tested on these datasets, the three deep CNNs discussed earlier (FCN, ResNet, Encoder), are the only 3 models which outperform the originally proposed models from this a previous study [51]. Full results can be seen in **Figure 10**.

The novel ideas explored in this paper will be applied to my specific TSC problem. Testing these cutting-edge ideas should mean my models perform better than the older ideas explained previously in this section. Some downsides to this paper are that no RNN solutions were discussed. As mentioned previously, RNNs learn in a very different way to CNNs, and seem suited to time distributed input data. Additionally, all the models are trained and tested within the same datasets which mean that the transferability of models is not explored.

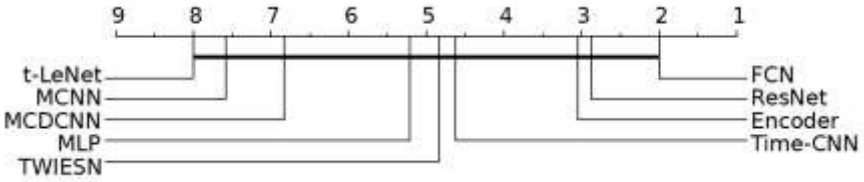

***Figure 10**: Critical difference diagram showing pairwise statistical difference comparisons of the 9 deep learning classifiers tested on the multivariate dataset in **Study 4** [18].*

2.4 Background Conclusions

The background and research provide key insight on how to tackle the problem. Here is a summary of the key points learnt so far:

- Raw data methods seem to have more promise than feature extraction methods. Raw data methods are less explored in literature, but recent developments show they hold more promise than well researched feature extraction methods.
- CNNs are the most popular method to solve the problem. This is the best and most consistent method, with lots of evidence to show this. Variations on CNNs seem to hold the most promise for recent developments in TSC.
- RNNs show promise to outperform standalone CNNs. The better methods combine models to receive the best results.
- Processing each input channel type separately (EEG, EOG, EMG) seems to lead to improved results.
- Studies training and testing on overlapping datasets seem to inflate accuracy results. To make a fair and balanced assessment, experiments will need to be carried out to test transferability of models.

From these points this section concludes with the research questions which the project aims to tackle. The overall goal of the project is to evaluate a variety of deep learning methods for the purpose of sleep stage classification of PSG data. In order to demonstrate the achievement of the stated aim, the project will need to explore newer and novel ANN models that have not been previously studied in similar research. Models will take inspiration from previous studies, as well as wider research into TSC, and the theoretical field of DSP. By the end of the project the basic tools will have been developed for a sleep scoring system that, with a few extra front-end features, could be used in a professional setting. In order to achieve this, the project will experiment with the transferability of models between datasets, and evaluate these results.

3. Approach

3.1 Initial Objective Changes

Throughout the research into and development of my project some of my core objectives changed from the initial plan. Here I justify why these core goals were changed. Firstly, I ruled out the need to specifically develop a pure stand-alone RNN to solve the problem. A stand-alone RNN would be useful for learning the longer temporal relations between epochs. However, literature has shown that most applications use CNNs for epoch by epoch sleep classification. I will still be assessing the use of RNNs combined with a CNN to learn the relationship between the sequences inside epochs in the project.

Secondly, I have avoided testing a variety of pre-processing methods to apply to the EEG data. I have, however, greatly explored them through literature, and came to the conclusion that training the networks on the raw data would be more interesting than spending time figuring out feature extraction methods. This is mainly because I feel testing on the raw data is underrepresented in literature, but also because I feel part of an ANNs learning process is to learn the best features to focus on by itself.

Finally, I originally set out to compare my ANN models to simpler, classical approaches of classification in machine learning such as LR or MLPs. I feel this comparison would have been interesting, however, this has already been explored in literature (SLEEPNET). The accuracy of ANNs far outperforms that of classical methods. Therefore, the comparison would be somewhat pointless. It will be much more useful to compare my results to similar problems in literature in order to assess the degree of the project's success.

Changing these goals means I have had the chance to test a much wider variety of ANN models and even theorise some fairly original ideas. I have produced 8 models in total, each with unique features. Their design is described in detail in section 4.1. The remainder of this section goes on to explain my approach of design for each part of the system.

3.2 Data Representation

The first issue in the project was to overcome the complicated datasets. This meant standardising the raw data for the system. In their original format, PSGs from both datasets were contained in MATLAB files. A standard format for input data was theorised and then applied to each dataset. A custom input-data script was written in Python to process both datasets. The script ran through several ordered methods, which included features such as:

- Reading specific trial data from each channel in the MATLAB file score data from an external text file.

- Slicing any unnecessary channel data if the recorded data didn't fit nicely into each epoch.
- Normalising the channel between each channel's *Interquartile Range* (IQR) as to remove the effect of noisy outlier data.
- Cleaning and formatting score data to fit into 5 numerical classes as specified by Keras.
- Reshaping and splitting the channel data into Numpy [55] files of exact and constant dimensionality.

The script produced individual epoch files of Numpy ".npy" files which contained a 2D Numpy array of size 6000xNumOfChannels. Each epoch was given an ID and recorded in a list. A dictionary was used to link each epoch ID to its sleep score.

3.3 System Design

A system was needed to build, train, and test the theorised model architectures. The approach of the project is primarily research focused. The final direction of the project will be a balanced evaluation of the techniques implemented. All software was designed with this in mind. This project should serve as the background research for a possible future system to be developed using the models and tools that have been designed and tested. The final system in the project serves as a playground to experiment with 8 different ANN architectures. Different models can be designed and trained interchangeably thanks to the design of the system. Models have been designed in an *Object Oriented* (OO) fashion; where a parent model class contains common methods for all models. This parent class also acts as a factory class where the specific type of model can be chosen. The individual child model classes contain a "build" method which includes all of the necessary Keras code to build the model.

The datasets used in the project are large (over 25Gb in total size); keeping all of this data in memory while the models are training is infeasible. To solve this problem the system uses custom data generator classes inspired by the Stanford design [56]. A data generator works by loading the specific part of the dataset needed by the model into memory. This is done across multiple cores and in near real-time. Data generators can also apply simple transformations to the data before it is fed into the network. My generators are in control of formatting the data so that it is the perfect shape to be used by the models straight away. Having these simple and fast generators allow models to be trained and tested easily. Therefore, the focus can be kept on tweaks to the model architecture without having to make constant changes to how the data is read into the system.

A main script is used to set all of the initial parameters for the system. Model parameters, testing sets, and generator parameters can all be easily changed from here, without the need for large changes to then be made elsewhere in the code. The system was designed in this way because testing has to be carried out on a large variety of models across different datasets. It was of prime importance that more time was spent

designing a flexible system to begin with, in order to save development resources in the long run. Designing and testing the networks was much easier because of this.

3.4 Model Design

My approach to model design was first to build and test simple models that have already had success within literature. This was to ensure that the remainder of the project would be feasible, and to allow for time to overcome any unforeseen issues. These models were initially trained and tested on a small subset of DS1. The final official comparisons between the models are comprised of two experiments. The first of which was on DS1. As development progressed, the whole of DS1 was split into training, validation, and test sets. The results for each model trained on this dataset are discussed later in the report. The second experiment aims to examine the transferability of the models to new datasets. The models are first trained on DS1 and tested on DS2. A comprehensive description of both experiments can be found in **Section 4**.

The methodology behind the model design process was a systemic life cycle of continuous improvement not too dissimilar from the System Development Life Cycle. My first stage was research. I researched novel ideas in published literature to inspire the design for a model. Often ideas from multiple papers and articles were combined to form the final model. Next was the design phase, where the model was implemented in the code, and any problems in the initial structure were smoothed over. This was followed by the training and testing phase. Models are trained on the training set, and tested on a separate test set. Multiple factors of performance are analysed: training and test accuracy, time taken to train, convergence speed, time taken to overfit, etc. Finally, the improvement phase. Based on the previous results, tweaks are made to the network's hyperparameters. This is more of an art than a science. At the beginning tweaks tend to be somewhat random, until a feel is obtained for which direction to push certain parameters in. This constant trial and improvement phase continued until I was happy with the final results produced by the model. This cycle of trying out new ideas, and tweaking the architectures and hyperparameters was what the bulk of the project was spent on. This is because training the models repeatedly on the larger datasets would often take hours.

4. Implementation

4.1 Model Architectures

I present and describe 8 unique ANN model architectures, specifically designed for sleep stage classification from multichannel raw signal input. Many of the designs are inspired by (i) progressions in TSC with new and novel ANN design from literature, or (ii) DSP theory, both of which have not been applied to this specific problem. Model titles will be annotated with the relevant markers to clearly indicate when a novel approach has been applied.

sCNN

sCNN is a shallow convolutional neural network comprised of two 1D convolutional layers. The input layer feeds into the first convolutional block made up of a 1D convolutional layer of 32 filters with a kernel size of 5. This is followed by a 1D max pooling layer of kernel size 5 to down-sample the data. The second convolutional block is made up of a 1D convolutional layer with 48 filters, again with a kernel size of 5. This is followed by a global average pooling layer to average each filter channel to one value. This was inspired by this article on classifying accelerometer sensor data [57]. This block connects to a *Fully Connected* (FC) layer of 40 nodes, before finally connecting to the FC classification layer of 5 nodes with a dropout value of 0.1 to prevent some of the overfitting. The penultimate FC layer acts to extract the key features, regardless of any temporal structure. All layers use the ReLU activation function where appropriate except for the final classification layer which must use the soft-max activation function in order to produce a multiclass probabilistic estimate. The network condenses the input data very quickly, forcing the network to retain only the most important information. However, this could mean that the network is not capable of learning all there is to learn from the dataset. This is the shallowest network tested and serves as a good baseline for the rest of the models. Full specifications for filters and kernel sizes can be seen in **Figure 11**. The network contains only 11,685 trainable parameters.

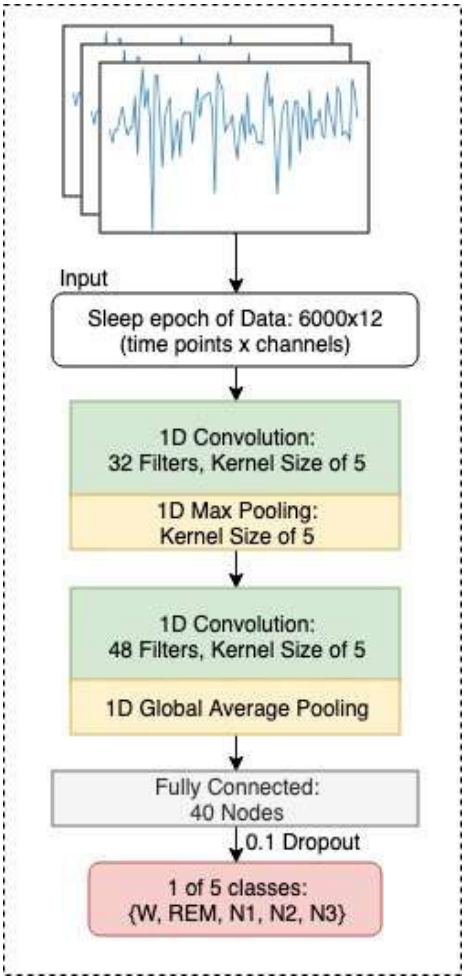

***Figure 11**: sCNN model architecture.*

dCNN

dCNN is a deeper CNN comprised of 4 convolutional blocks each made up of two 1D convolutional layers each. The input layer feeds into the first convolutional block. This is made up of two 1D convolutional layers with 16 filters each of kernel size 5. This is followed by a max pooling layer with kernel size 2, used to down-sample the data but at a slower rate compared to sCNN. The remaining blocks are again comprised of two convolutions followed by a pooling operation. Block 2's convolutional layers have 32 filters each, but a smaller kernel size of 3. This is followed by a max pooling operation identical to that of block 1. The third convolutional block is identical to block 2. The final block has 2 convolutional layers each with 256 filters of kernel size 3. This is followed by a global average pooling layer to average each filter channel to one value. The final block is connected to a FC layer with 64 nodes, before finally connecting to the FC classification layer of 5 nodes. Again, all layers use ReLU except for the final layer. Spatial dropout [58] is used between convolutional blocks with a rate of 0.01 and regular dropout is used before both FC layers with rates of 0.01 and 0.05 respectively. Using multiple convolutions before the pooling operation is standard for many successful image classification networks [59]. Deeper CNNs in general are better at learning the complex patterns in data. In theory this network should perform better than sCNN. Full specifications for filters and kernel sizes can be seen in **Figure 12**. The network contains 251,541 trainable parameters.

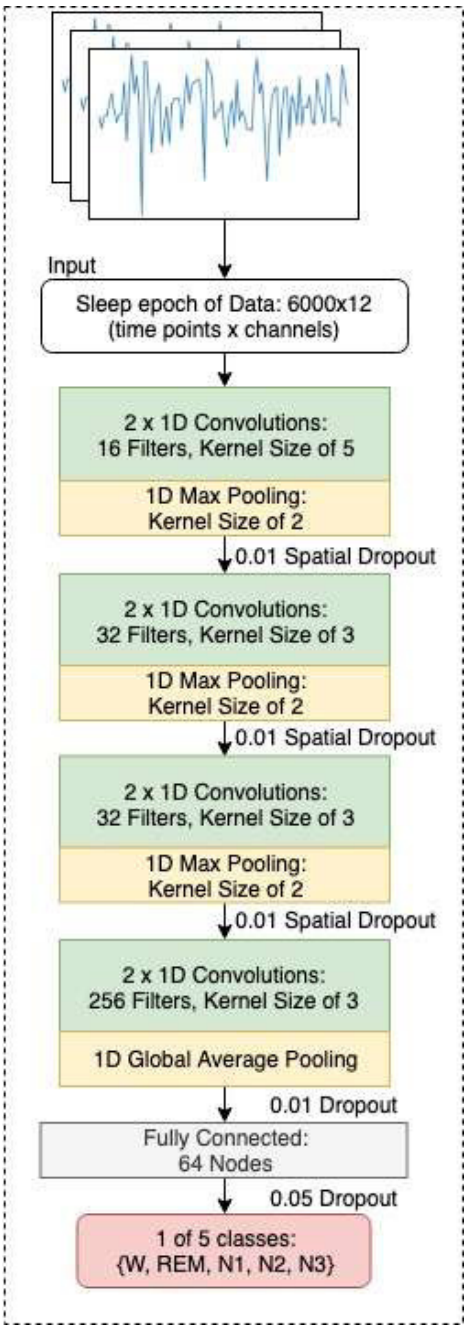

***Figure 12**: dCNN model architecture.*

Multi1D

Multi1D is a deep CNN similar to dCNN except that the input channels are initially processed separately by the network. Multi1D was designed after initial tests with dCNN proved successful. In dCNN all channels from the input data are processed by the same filters. Multi1D aims to use 3 different CNN paths to process the EEG, EOG, and EMG data separately, instead of applying the same rules to all channels of data. This is a custom solution for the problem, but the design has been inspired by **Study 3** [48]. The paths start with an individual input layer. The results from the CNN paths are concatenated before being passed to a FC layer of 500 nodes, and then finally the classification layer. The CNNs used in Multi1D are inspired by dCNN's design. They consist of blocks of two 1D convolutional layers, followed by a pooling operation. Each path ends in 1D global average pooling layers before concatenation. Full specifications for filters and kernel sizes can be seen in **Figure 13**. The final network contains 12,036,781 trainable parameters.

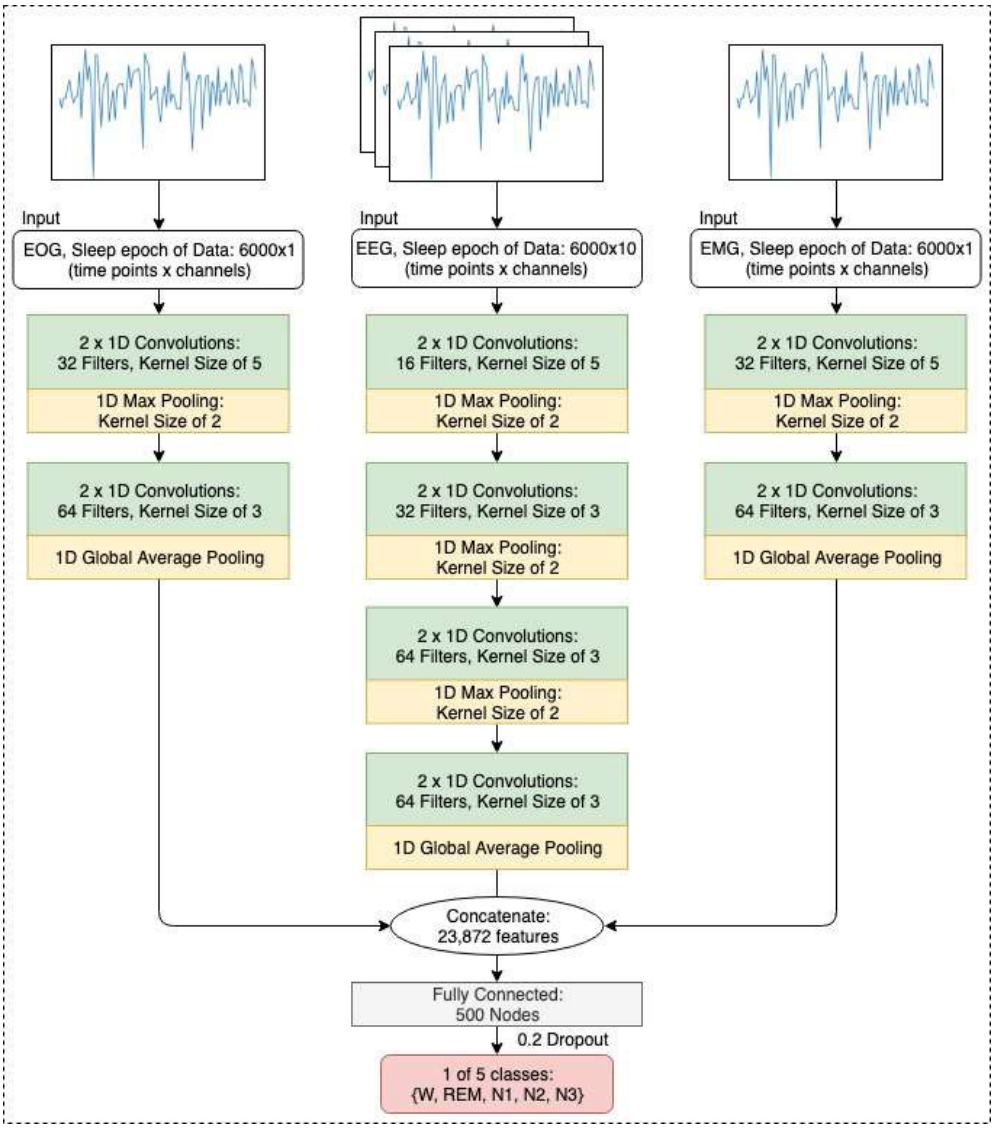

***Figure 13**: Multi1D model architecture.*

Multi2D [ii]

Multi2D is a deep, pathed, CNN with the multiple input channels processed separately, similar to Multi1D. The only difference between Multi1D and Multi2D is that the CNN responsible for processing the EEG data is a 2D CNN. The model was inspired by this paper [60] where 2D convolutions are applied to EEG data. The input EEG, is treated like an image with dimensionality 6000xNumberOfChannels. First, a spatial 2D convolution is applied across all channels to reduce the size of the EEG data to 6 channels. This has a linear activation due to the fact that relations between EEG nodes on the scalp are linear. Next regular temporal features are extracted from the data with 2D convolutions which emulate 1D filters, as the kernel size is 5x1 (to begin with). Treating the depth of channels like a 2D image means that the exact same filters are applied to the EEG data channel by channel. This means there are far fewer parameters to be trained by the model. The whole of this model architecture is inspired by the fact that CNNs can accurately self-teach 1D signal processing. The full specification of hyperparameter choices can be seen in **Figure 14**. The final network has 328,977 trainable parameters, almost 37 times less than Multi1D.

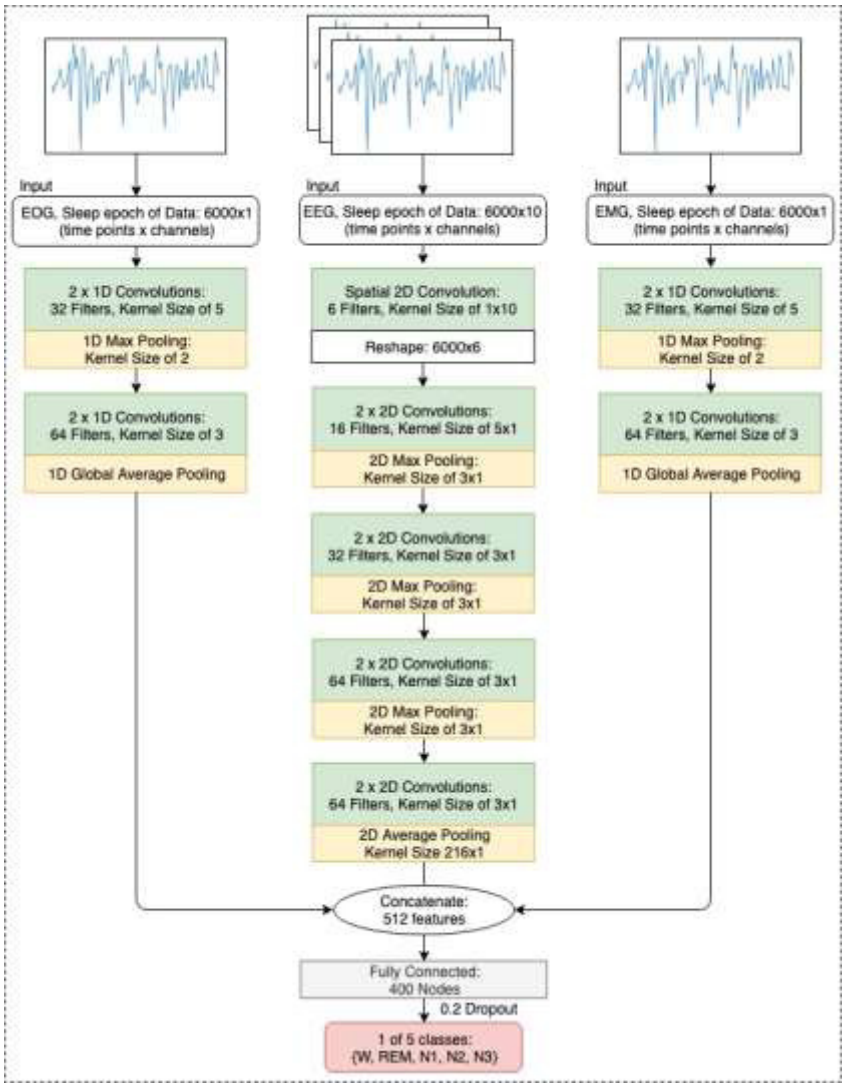

***Figure 14**: Multi2D model architecture.*

MultiF [ii]

Multi Filter (MultiF) is another variation of Multi1D inspired by signal processing theory. 1D convolutions simulate frequency filters when applied to 1D signal data. The larger the kernel size, the lower frequency of signal the 1D convolution is sampling, and vice versa for a smaller kernel size. The inspiration for MultiF is to apply multiple convolutions to each type of signal all with differing kernel sizes. This is in order to learn different information from performing the convolutions at different frequencies bands. This approach is in the same vein as GoogLeNet [61], where different filter sizes are used in parallel. Each different type of input has its own CNN path through the network. Each path is further split into 3 shallow identical CNN networks, where the only difference is the kernel sizes. The three subnetworks have kernel sizes of 13, 5, and 3 respectively and are made up of two single 1D convolutional layers each followed by a pooling operation. The information from the output of all 9 subnetworks are recombined using a concatenation layer, before it is passed to a FC layer of 300 nodes. This is connected to further FC layer of 50 nodes before finishing at the FC classification layer. The full specification for the network can be seen in **Figure 15**. The final network has 250,085 trainable parameters.

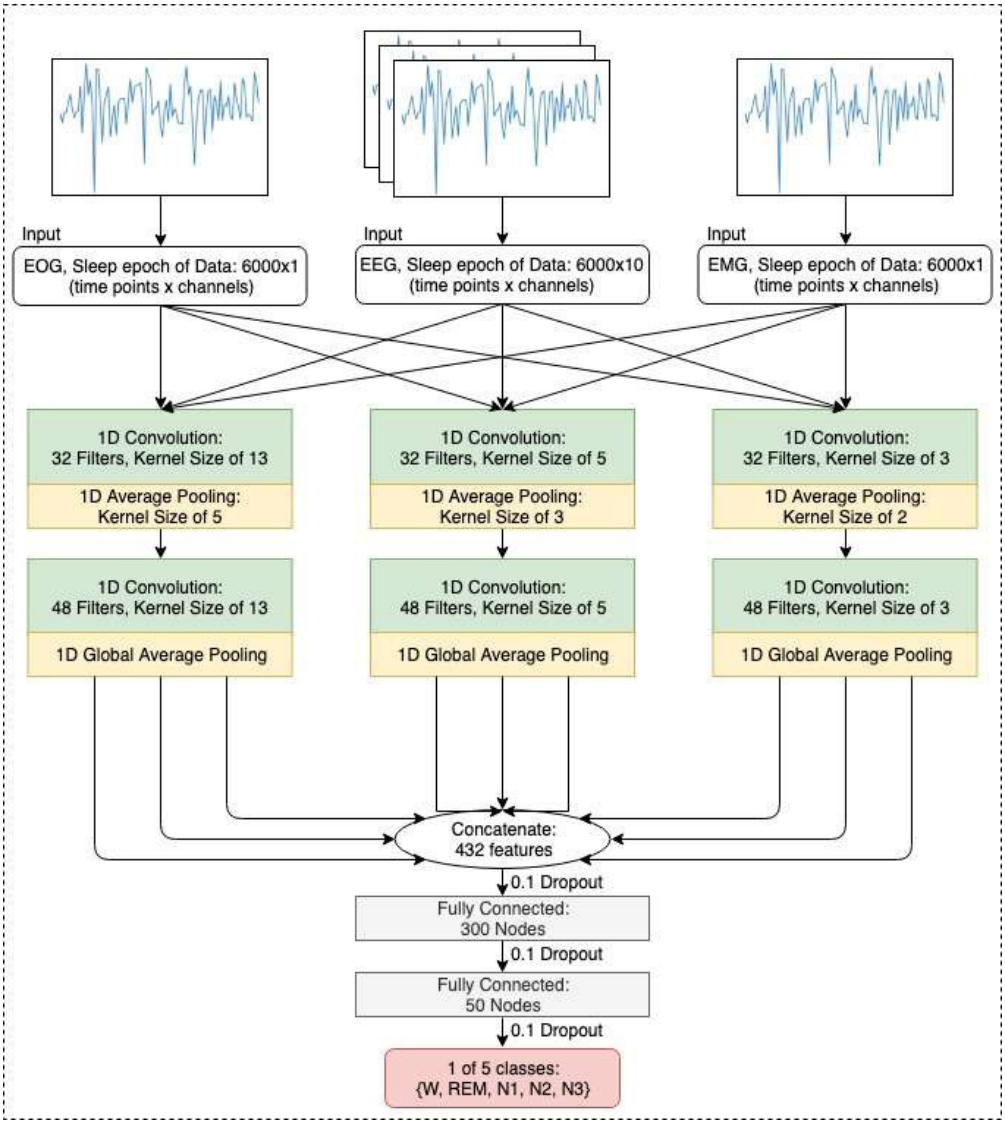

***Figure 15**: MultiF model architecture.*

ResNet [i]

ResNet is a CNN with residual skip connections otherwise known as a Residual Neural Network. ResNet solves the problem of wanting to design very deep networks. ResNet is an example of an FCN, with no pooling after convolutions, except for a final global average pooling layer before the FC classification layer. This model is inspired by the research explored in Study 4. The model consists of 3 residual blocks, all made up of 3 1D convolutional layers. Each block can be skipped by a residual connection. The result of each block and each skip is added together to produce a tensor of the same size for the next residual block. A single convolution is applied with a kernel size of one to reshape the skipped data after block one and two. Block one is made up of consecutive 1D convolutions with 32 filters of kernel sizes 8, 5, and 3. Blocks two and three are made up of consecutive 1D convolutions with 64 filters of kernel sizes 8, 5, and 3. The full specification for the network can be seen in **Figure 16**. The final network has 128,997 trainable parameters.

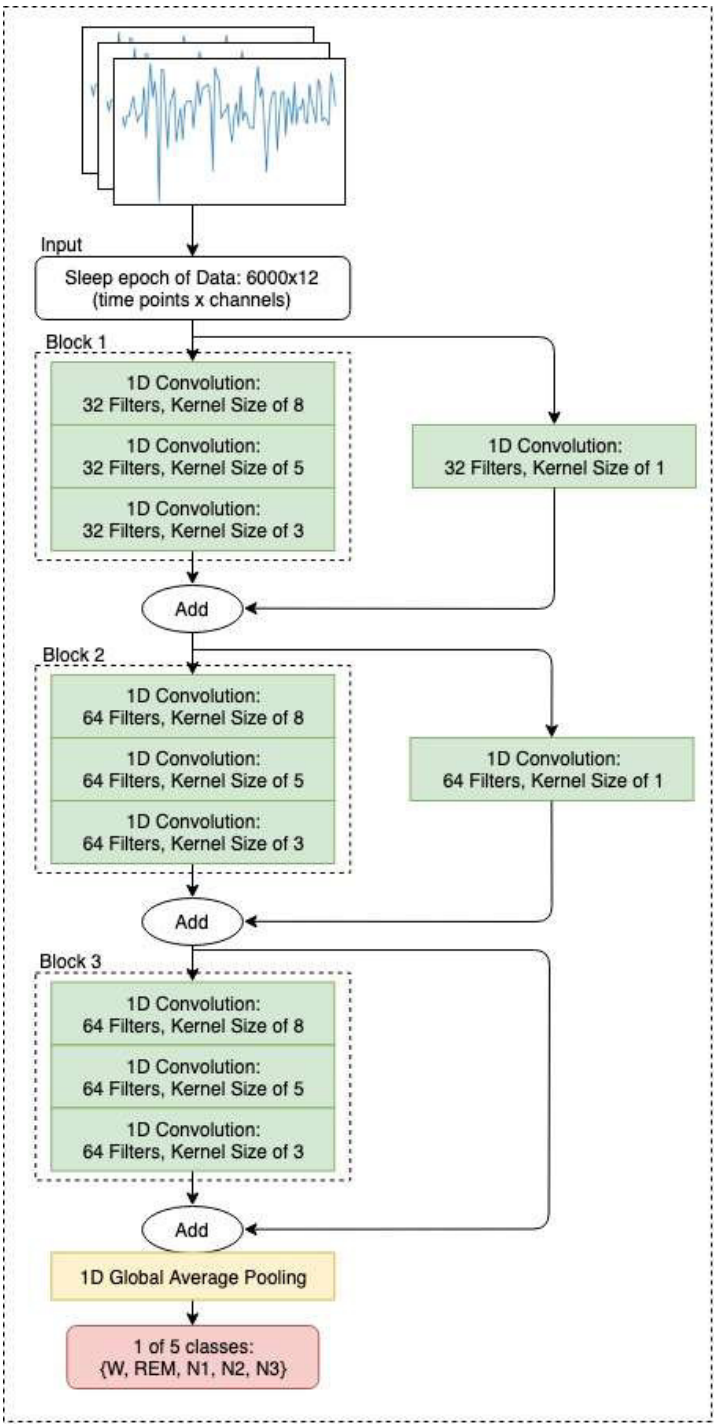

***Figure 16**: ResNet model architecture.*

CRNN [i]

CRNN is a recurrent evolution of sCNN. Compared to sCNN, the penultimate FC layer is replaced by a bidirectional LSTM layer made of 25 nodes. This is in order to learn temporal features from the epochs. This then feeds into the FC classification layer. Bidirectional recurrent layers connect two hidden layers of opposite directions to the same output. With this form of generative deep learning, the output layer can get information from past and future states simultaneously [62]. This means that the LSTM layer can process the epoch data twice; once in the positive time domain, and once in reverse. LSTMs work better when they have more data to process. Flipping the input sequence has had notable performance in speech recognition [63] which is what inspired this choice of design. In theory the LSTM layer should be able to learn sequence features from the epoch that sCNN could not have learnt alone. In this model the input to the LSTM is first processed by the convolutional layers, this is in theory to extract the important features from the raw data first. The CNN layers should learn to tune their filters to feed the most useful data to the LSTM, as they are trained in tandem. Full specification for the network can be seen in **Figure 17**. The final network has 24,735 trainable parameters.

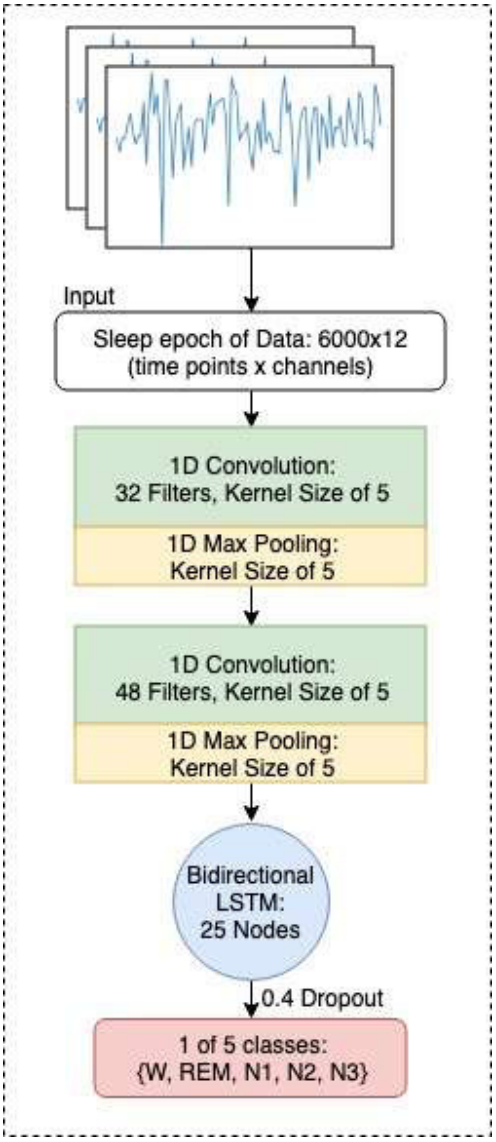

***Figure 17**: CRNN model architecture.*

ParaNet [i]

ParaNet features a deep CNN and an RNN to process the raw data separately, in parallel. The results of this are concatenated, before being processed by a series of FC layers to extract and combine the important features from both branches of the network. This paper's approach [64] is in the same vein, and has had success with generic TSC problems. A copy of dCNN is used as the convolutional branch of the network. For the RNN branch, the raw data is first transposed in order to present the data as feature vectors of 6000 time points for each channel to the RNN. The RNN is comprised of a simple LSTM of 8 nodes, with dropout at a rate of 0.5 before concatenation. After the two branches of the network are concatenated, a FC layer of 256 nodes processes the features. This finally feeds into the FC classification layer. In theory both branches of the network will extract different features from the raw data. The FC layer will then learn what combination of these features combine to match with the correct sleep class. Full specification for the network can be seen in **Figure 18**. The final network has 496,181 trainable parameters.

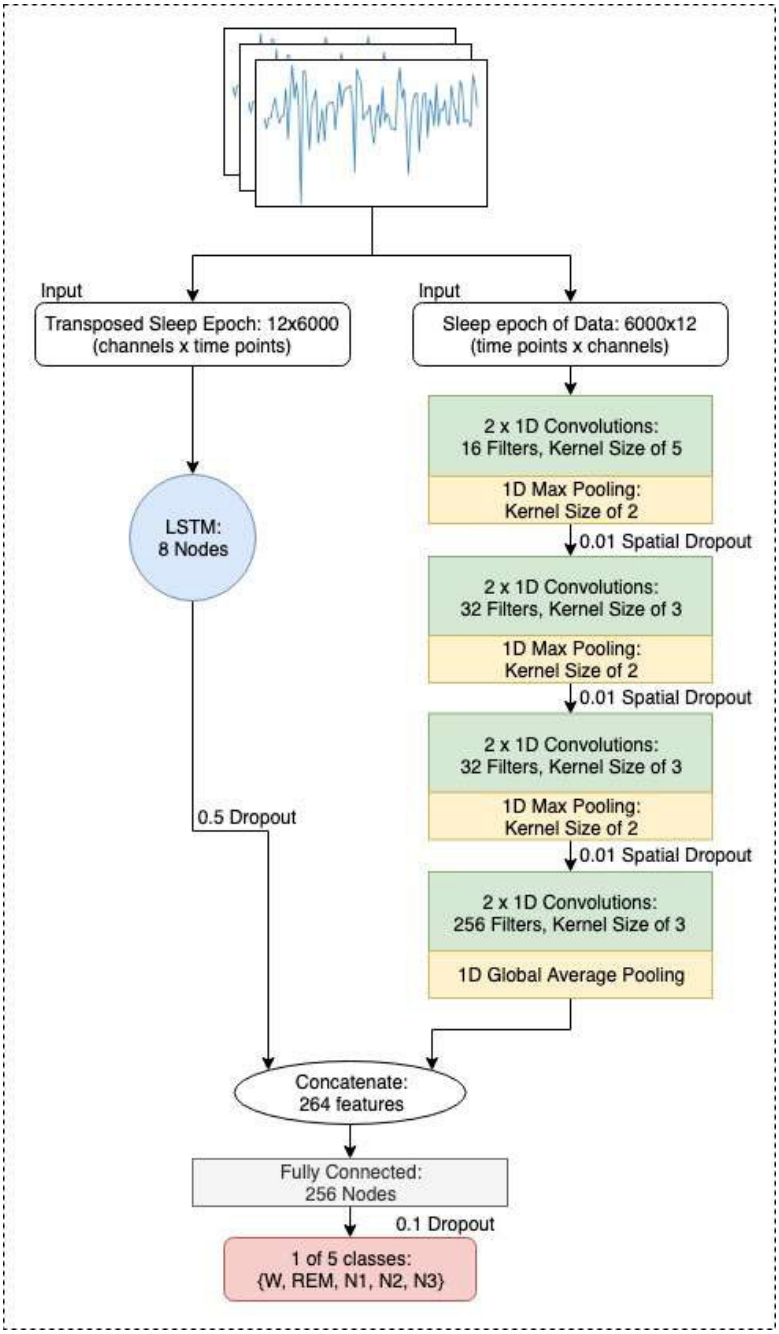

***Figure 18**: ParaNet model architecture.*

4.2 Model Training & Experiments

Two experiments are used to evaluate each model's performance. Here they are described in detail. The results of the experiments are presented in the following section. This section also describes the training parameters used when implementing the networks and ends with a brief comment on how the finalised models could be deployed to assist with sleep scoring in a professional setting.

Training

All models were trained to minimize categorical cross-entropy loss [65] with the Adam optimiser [66] having a learning rate of 0.001. All models are trained for a standard of 80 training epochs on their training sets. A model's accuracy is evaluated after each training epoch using a validation set; model weights are saved to an external file if this accuracy is an improvement on the current best. If the validation accuracy does not improve for 25 training epochs in a row, the system uses Keras' early stopping callback feature [67], to pre-emptively terminate the model's training. By the end of the model training the best weights will have been saved in the external weight file and the model is ready for testing on the test set.

Experiments

Experiment 1 (E1) was a baseline test performed solely on DS1. This experiment was the initial inspiration for the project; my success would be measured by these results. The PSGs in DS1 were first split up into individual sleep epochs of 6000 time points in each of the 12 channels. The sleep epochs were then shuffled to produce a randomly distributed mix of participant data between both the training and test set. 75% of the data was used for the training set, 15% as the test set, and 10% as a validation set. The exact size of the three final sets were:

- Training set: 31,102 sleep epochs
- Validation set: 4,148 sleep epochs
- Test Set: 6,220 sleep epochs

The same shuffled sets were used consistently across all models. The models are each trained on the training set for 80 training epochs, or until the validation accuracy did not improve for 25 consecutive training epochs. The best weights are recorded and then saved. This is measured by the validation accuracy after each training epoch. By the end of the training, each model is loaded with the best recorded weights and is tested on the test set. Predictions are made and a final accuracy measure is produced for each model.

Experiment 2 (E2) was performed on DS1 and DS2. This experiment was to test the transferability of models to an entirely separate dataset. This is a beneficial test because if the models are ever to be used in a professional setting, they will have to learn and classify new data. The PSGs in both datasets standardised by limiting the datasets to 11 common channels; 9 EEG, 1 EOG, and 1 EMG. They were next split up into individual sleep epochs of 6000 time points in each of the 11 channels. Participant data was not to be shuffled in this test in order to be a more robust measure of transferability. 10 participants from DS2 would be used exclusively for testing. The other 52 participants were used as a training set. The training set was first shuffled, then a validation set separated from the training set. The size of the validation set was matched to the test set. The size of the three final sets were:

- Training set: 43,142 of mixed DS1 & DS2 sleep epochs
- Validation set: 10,153 of mixed DS1 & DS2 sleep epochs
- Test Set: 10,153 of exclusively DS2 sleep epochs, from completely separate participants

The same sets were used consistently across all models. The models are each trained on the training set for 80 training epochs, or until the validation accuracy did not improve for 25 consecutive training epochs. The best weights are recorded and then saved. This is measured by the validation accuracy after each training epoch. By the end of the training, each model is loaded with the best recorded weights and is tested on the test set. Predictions are made and a final accuracy measure is produced for each model.

Finalised System

In order to make use of the full set of training data available between the two datasets, all finalised models will be trained across both datasets. The datasets will be combined; 36 of the 40 PSG recordings from DS1 and 19 of the 22 recordings from DS2 will be used as the training set. The remaining 4 and 3 recordings will be used together as a validation set. The models will be trained on the training set for 80 training epochs, or until the validation accuracy does not improve for 25 consecutive training epochs. The best weights will be recorded and saved. This is measured by the validation accuracy after each training epoch. In theory these models will be better suited in a system for professional adoption compared to using the weights from E1 or E2. This is because they will have trained on much more data. Any system may have to go through a transitional phase when it is first introduced to a new lab. This would be where the models continue to be trained for 10-15 participants in the new lab in order to be able to generalise to their participants. After which the models can be used to aid with sleep scoring. This process is modelled in E2 and achieves good results (see next section).

4.3 Development Problems & Scrapped Ideas

A portion of project time went into exploring other model architectures and other datasets. Here is a brief explanation of the substantial ideas that had to be scrapped due to problems encountered in the development phase.

CNN-LSTM

This model progressed through the design phase but encountered problems with training. The inspiration behind this model was from this article [68] and is similar to the original CRNN model; the network would first use convolutional layers to extract features to be then processed by an RNN. However, the difference in this network was that all epochs would be first processed by the CNN to produce a time distributed sequence of feature vectors. The RNN would then process this sequence of feature vectors, rather than just the individual sequence of timepoints inside an epoch. In essence, adding an extra time across all epochs for the RNN to process, see **Figure 19**. This model would have the benefit of remembering previous epochs, and possible looking at the next epoch. For example, if the previous epoch was scored as stage N1 and the next epoch also was scored as N2, the network could learn that the current epoch is much more likely to be either N1 or N2. The network would make use of Keras' *TimeDistributed* layer wrapper [69] around the CNN in order to produce a time sequence of feature vectors; one feature vector for each epoch. Another benefit to this model is that its CNN could be chosen from the best performing idea out of other the convolutional models. For example, the input signals could have been split like in the Multi CNNs, or multiple filter sizes could have been used such as in MultiF specifically.

There is nothing wrong with the model concept in theory; something went wrong with the implementation. The model would not learn throughout training; the accuracy value would initially fluctuate by a few percent, before sticking to a number in the 40's. I am unsure of the reason behind this but I assume it is down to the way data was fed into the network. Each epoch was being fed individually into the network by the generator, whereas it is possible that a sequence of epochs was needed instead. This would mean the input data shape would need to be dramatically altered; each input would be a whole nights PSG and the data generators would need to be specifically redesigned to reflect this. More research would certainly be required to get to the bottom of this problem. The finite time resources of the project meant that regrettably, this model had to be scrapped.

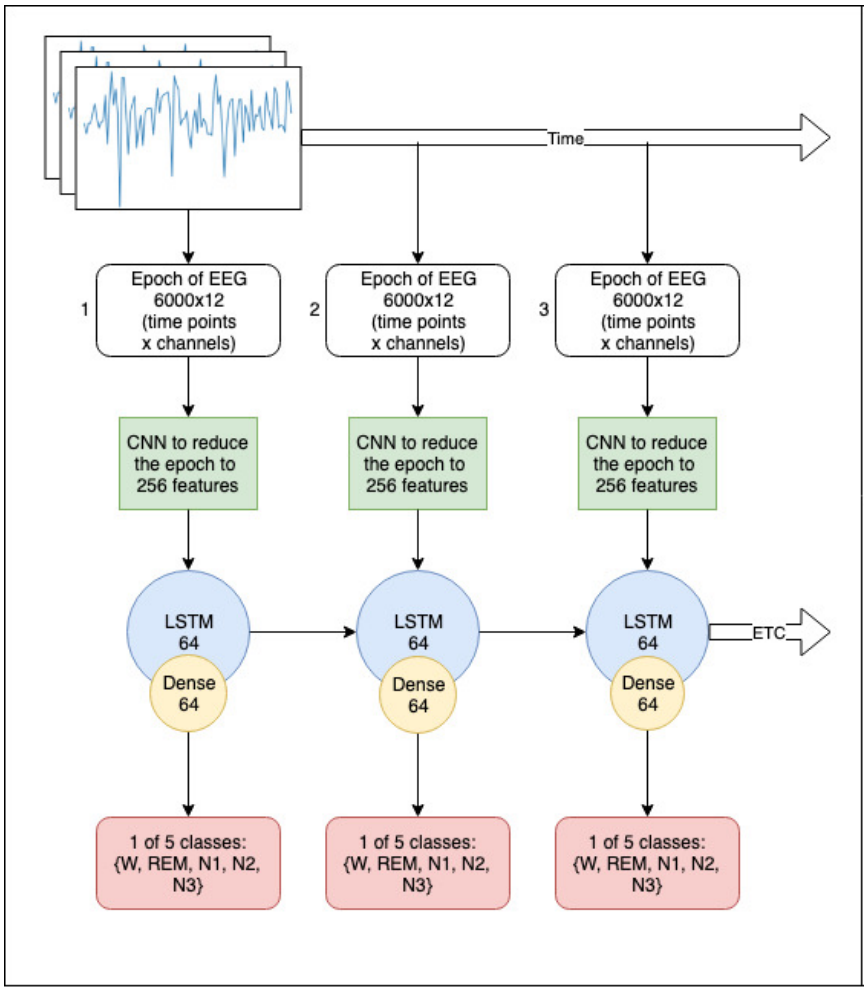

***Figure 19**: Theorised CNN-LSTM model architecture.*

ISRUC-SLEEP Dataset

The ISRUC-SLEEP Dataset [70] is a comprehensive public dataset for sleep researchers, with sleep data from 100 participants. It contains over 90,000 individually scored epochs. Having this large dataset would have been overwhelmingly beneficial to my project, as one of the biggest problems with training neural networks is when there is a lack of data, models will quickly overfit. The dataset has been used in other studies similar to mine focusing on automated sleep stage scoring with ANNs [71] [72]. This would have meant I could directly compare my solutions to previously published solutions on the same dataset. However, problems encountered with using the dataset meant that it could not be used.

A significant amount of time was spent acquiring the dataset; initially the dataset could not be downloaded online, and I had to directly contact the owners. Next, much time was spent extracting the data from the MATLAB files using a similar process to DS1 & DS2; a custom script was designed to split the data into scored epochs. Time was also taken to upload the dataset to the Hawk environment. This again was time consuming because of the dataset's large size. Unfortunately, the final straw came

down to problems when training models on the dataset. My models would only reach accuracies in the 60%-70% range at best. After some investigation I realised that the IQR normalisation stage in the processing script was causing the range of some channel values to change dramatically (an example is from between -30 and 30 to between -4000 and 4000). The reason for this strange behaviour is unknown (I could only assume the data was much noisier), but in order for tests to be consistent, the same pre-processing methods had to be applied to all of the datasets. At this stage in the project, the time remaining had to be spent pursuing meaningful results, so the dataset was left behind.

5. Results & Evaluation

5.1 Results of Experiments

After carrying out both experiments the results are as follows. There are a few factors used to assess model performance. The first, is overall accuracy. This is the percentage of correct predictions each model achieved on the test sets. Accuracy is further deconstructed into a confusion matrix to see the performance by individual class. Next is training time; how long each model took before stopping. No model reached the maximum of 80 training epochs so we can be sure the best possible weights for each model have been used (it is worth noting that each model trained an extra 25 training epochs to ensure the maximum validation accuracy had been achieved). Finally, the convergence of a model is measured by how many training epochs taken before each model reached a maximum validation accuracy. Results are presented clearly in **Tables 2 & 3**. In this section the results are presented for each model before they are discussed in detail.

Model Name	E1 Accuracy (%)	E1 Training Time (m)	E1 Convergence (training epochs)	E2 Accuracy (%)	E2 Training Time (m)	E2 Convergence (training epochs)
sCNN	87.3	133.8	37	83.6	140.4	25
dCNN	87.2	95	17	84.1	182.7	34
Multi1D	86.9	82.7	3	81.8	97	3
Multi2D	**87.7**	129.6	17	83.1	139.5	19
MultiF	87.6	166.9	29	**86**	154.7	24
ResNet	87.4	92.1	14	84.8	111.7	16
CRNN	**87.7**	126.1	33	83.8	180.7	26
ParaNet	**87.7**	108.4	23	84.8	107.9	21

***Table** 2: Results table for E1 and E2.*

Participant	Z3Score	sCNN	dCNN	Multi1D	Multi2D	MultiF	ResNet	CRNN	ParaNet
T1	78.3	91	91	85	**92**	90	**92**	89	91
T2	85.1	80	77	78	76	**82**	80	79	78
T3	88.4	73	76	80	75	77	74	**81**	**81**
T4	80.2	89	**91**	87	87	89	90	88	89
T5	77.6	80	81	79	81	**82**	80	78	80
T6	84.1	77	80	79	75	**89**	78	80	77
T7	86.5	87	86	86	88	**89**	88	87	87
T8	84.9	87	87	71	83	86	**88**	81	85
T9	86.5	90	**91**	86	87	88	90	89	89
T10	88.2	82	81	87	87	**88**	**88**	86	**88**
AVG	**83.98**	83.6	84.1	81.8	83.1	**86**	84.8	83.8	84.5

***Table** 3: Results of each model by individual participant in E2, with comparison to Z3Score. Green entries are where the model in question outperforms Z3Score, bold entries are where the model in question outperforms every other model.*

sCNN

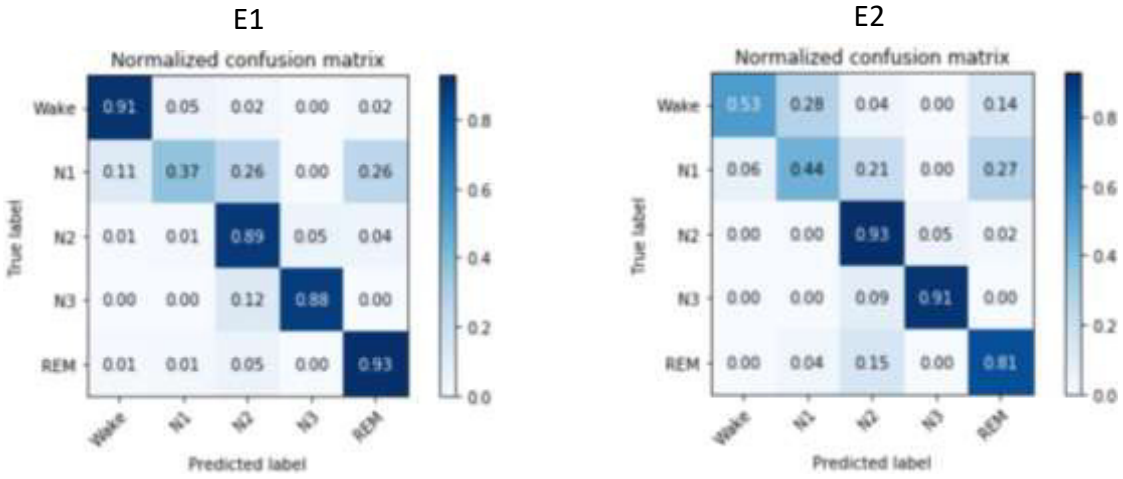

***Figure 20**: Resulting confusion matrices for sCNN.*

sCNN was the simplest model tested, comprised only of 4 layers. It achieved the 4th highest accuracy in E1 and the 3rd lowest in E2. However, sCNN was only 0.4% behind from the highest accuracy in E1 and 0.4% away from the industry standard Z3Score in E2. This being said, sCNN beat Z3 on 6 out of 10 of the participants; the margins of defeat are what brought the overall accuracy down. In E1 sCNN was particularly effective at classifying REM sleep effectively, with 93% accuracy, joint highest with MultiF. For quite a very simple model, this performance is very notable. Although the model was the simplest to train (averaged 3 minutes 37 seconds per training epoch in E), the overall training time for E1 is the second highest. This is because it took the longest time to converge, 37 training epochs. This is a good sign that the model didn't overfit.

dCNN

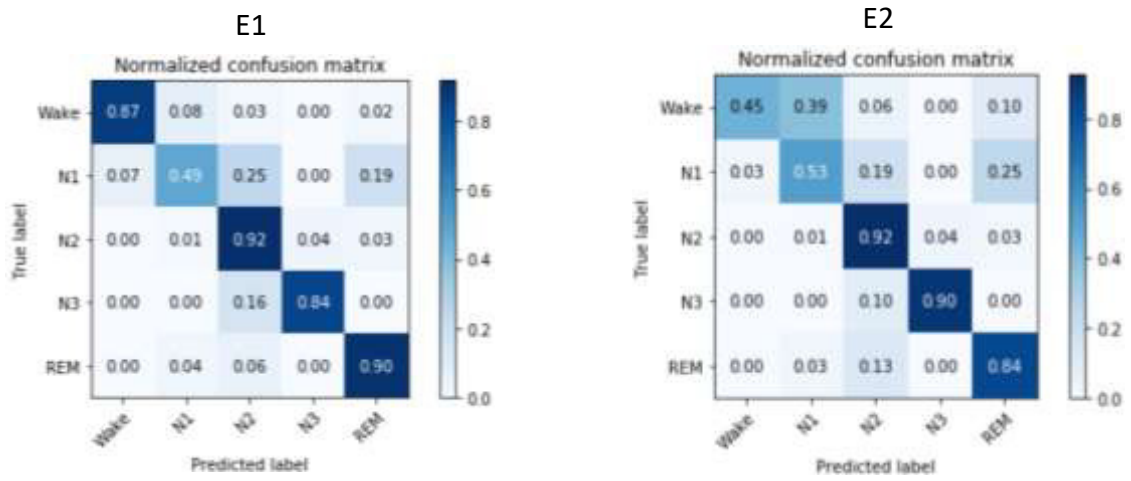

***Figure 21**: Resulting confusion matrices for dCNN.*

dCNN is a deeper version of sCNN with double convolutions per layer. It achieved the 5th highest accuracy in E1, being beaten by its shallower brother by only 0.1%. This difference is somewhat negligible. It too beat Z3 in the same 6 out of 10 participants but had higher accuracy results overall beating Z3 by 0.1% thus achieving the 4th highest accuracy in E2. For two of the participants, dCNN achieved the best accuracy scores out of any of the models. dCNN was consistently better across both tests at classifying N1 correctly compared to the other models being correct around 50% of the time. For a more complex model, it is disappointing that dCNN didn't perform much better than sCNN across the test. One advantage is that it did converge much more quickly in E1 than sCNN. However, in E2 dCNN had the longest training time of all the models and the longest convergence time. This means that the model likely didn't overfit.

Multi1D

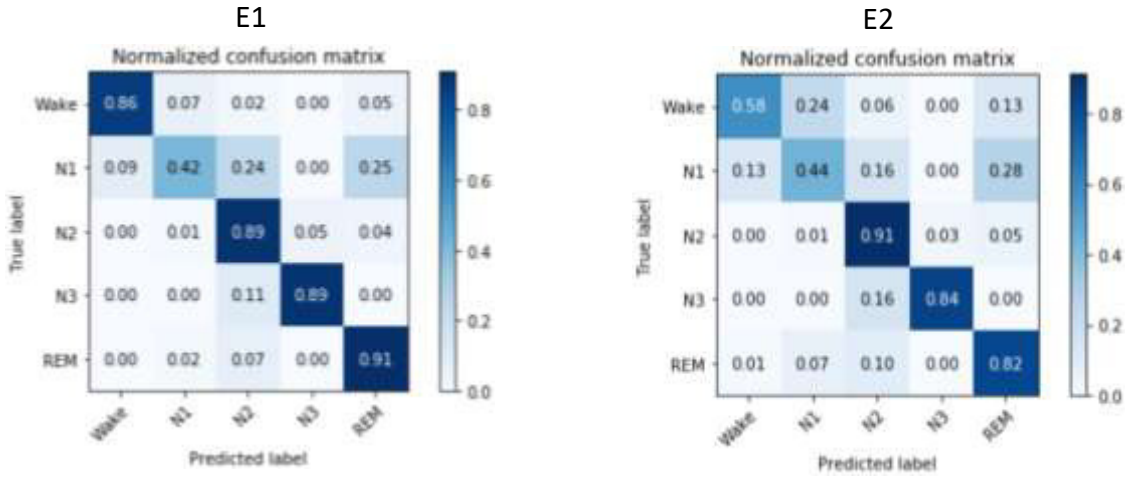

***Figure 22**: Resulting confusion matrices for Multi1D.*

Multi1D performed worst across both tests; by 0.3% in E1, and 1.3% in E2. This is clearly down to overfitting. In both tests the model only took 3 training epochs to converge to its best validation accuracy score. This is case even with the model having the longest average time per training epoch (27 mins 34 secs in E1, 32 mins 20 secs in E2). These results actually show promise for the model if it could train on a larger dataset. However, it is likely that the model architecture is too complex. One of the only good results is that Multi1D was the best model at classifying stage N3 of sleep in E1 at an accuracy of 89%. An even better result is that in E2 the model was the best by far at classifying wake data, beating even the best model by 3%.

Multi2D

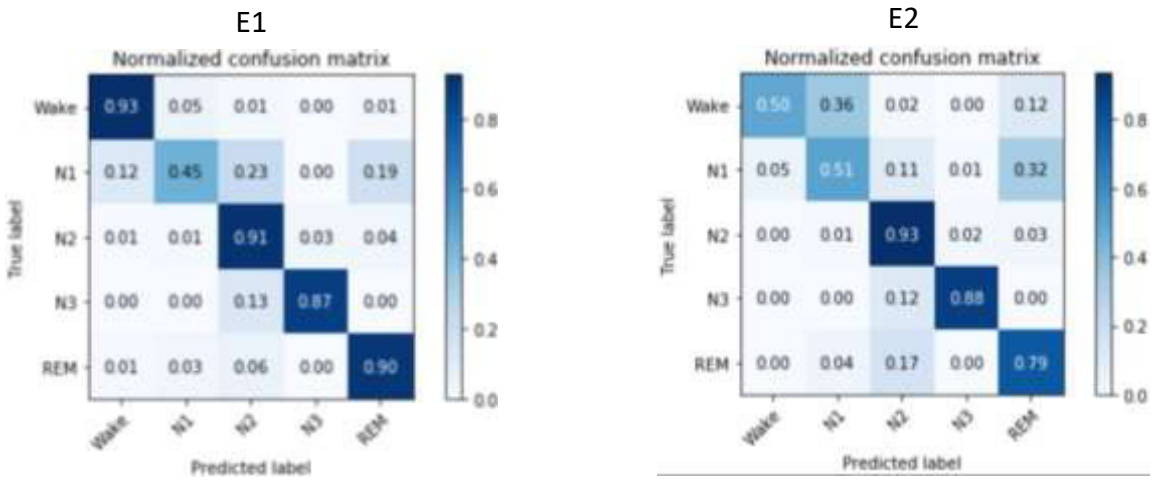

***Figure 23**: Resulting confusion matrices for Multi2D.*

Multi2D has the same architecture as Multi1D except for using 2D convolutions on the EEG data, thus greatly reducing the number of parameters and simplifying the architecture. This model achieves the join highest accuracy in E1, but the second lowest in E2. Although the model did not beat Z3 in overall accuracy, it did classify 5 of the 10 participants more accurately. Multi2D was the best model at classifying wake data in E1 with an accuracy of 93%, beating the average by 4.5%. The model had standard training time and convergence speed with no signs of severe overfitting. For such a complex model architecture, the model was fast at training, the average time per training epoch was only 7 minutes 37 seconds in E1 and even dropping in E2.

MultiF

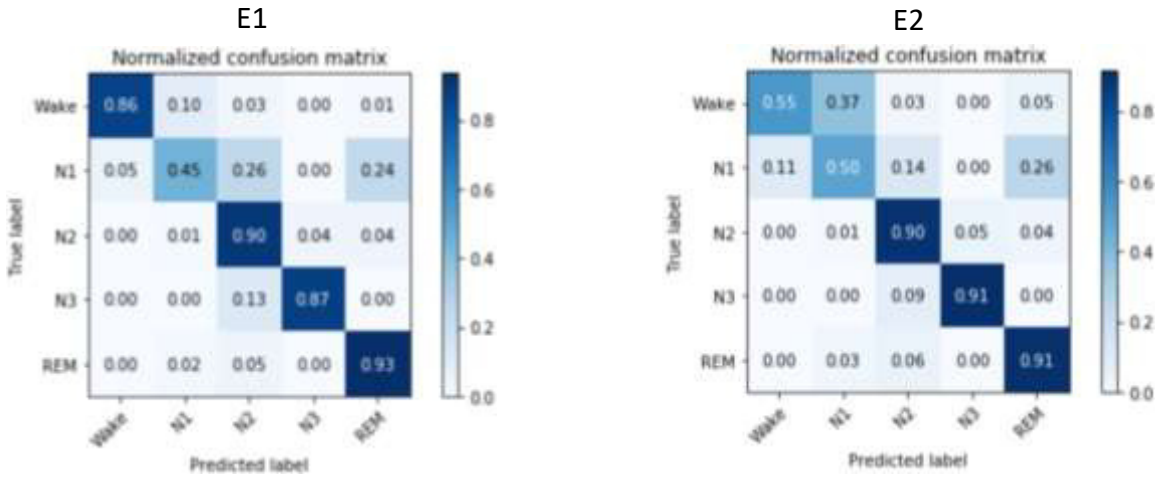

***Figure 24**: Resulting confusion matrices for MultiF.*

MultiF is similar to both Multi1D and Multi2D in the fact that it splits the initial input channels. However, MultiF uses multiple convolutions in parallel with different filter sizes to extract different features across different signal frequencies. MultiF achieves the 2nd highest accuracy in E1 losing out to first place by only 0.1%, a fairly negligible difference. In E2, MultiF achieves the highest accuracy by far, beating the next closest model by 1.2%. Additionally, MultiF beats Z3 in 8 of the 10 tests, and an overall accuracy of 2%. MultiF achieved the best accuracy on 4 of the 10 participants compared to any of the other models. This indicates that MultiF is the most transferable model. In E1 and E2, MultiF was the best at classifying REM sleep of all the models. As for convergence and training speed, MultiF performed well for being a complex model.

ResNet

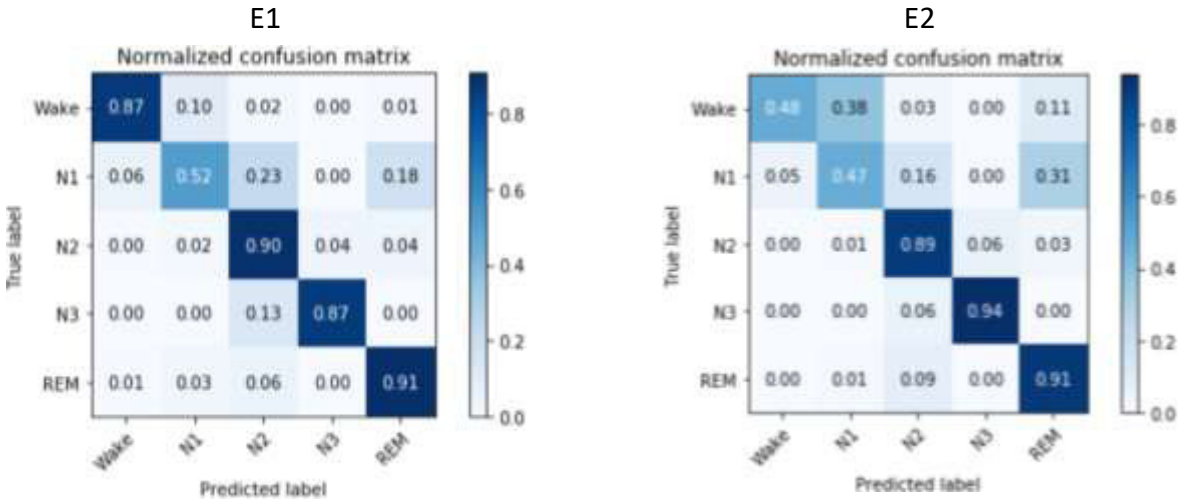

***Figure 25**: Resulting confusion matrices for ResNet.*

ResNet is a deeper CNN making use of residual connections. This model had the 3rd best performance in E1 and the 2nd highest in E2. These are very good results and show promise for this model architecture. In E1, ResNet was the best model at classifying N1, which all of the models struggled with. In E2, ResNet classified 7 out of 10 participants better than Z3, beating overall accuracy by 0.8%. ResNet achieved highest accuracy on 3 of the participants specifically compared to the rest of the models. Furthermore, ResNet was the best model at classifying stage N3 in E2 beating some models by 10%. Training time was the second fastest in E1 and the third fastest in E2. The model also had fairly fast convergence, the second lowest training epochs in both experiments. This could indicate that some overfitting occurred, although not as extreme as Multi1D. However, ResNets in general are known to have faster convergence than regular CNNs.

CRNN

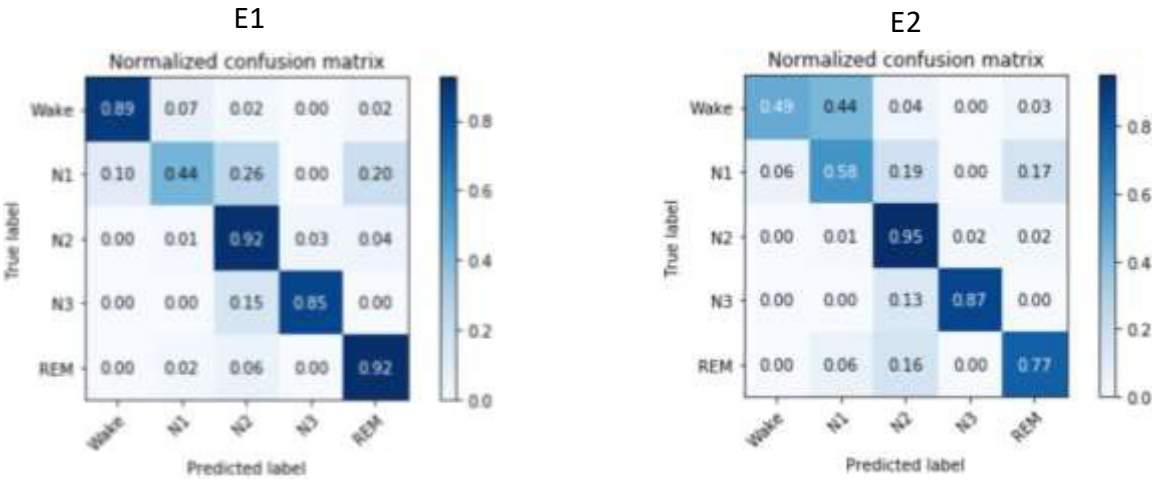

***Figure 26**: Resulting confusion matrices for RCNN.*

CRNN makes use of both convolutional and recurrent layers, to process individual epochs in a unique manner. CRNN achieves the joint highest accuracy in E1 but only the 5th highest in E2. This is promising evidence for the use of RNNs for TSC. In both E1 and E2, CRNN was the best model at classifying stage N2 with 92% and 95% accuracy respectively. In E2 specifically, CRNN was also the best model at classifying stage N1, a notably significant result, as most models struggled in this area. CRNN classified 5 out of 10 participants more accurately than Z3 but lost out on overall accuracy by 0.2%. CRNN had long overall training times, with slower convergence; a good indication that the model didn't suffer from overfitting. As CRNN is a shallower model comparatively, it boasted the 2nd fastest average training time per epoch in E1, at 3 mins 50 secs.

ParaNet

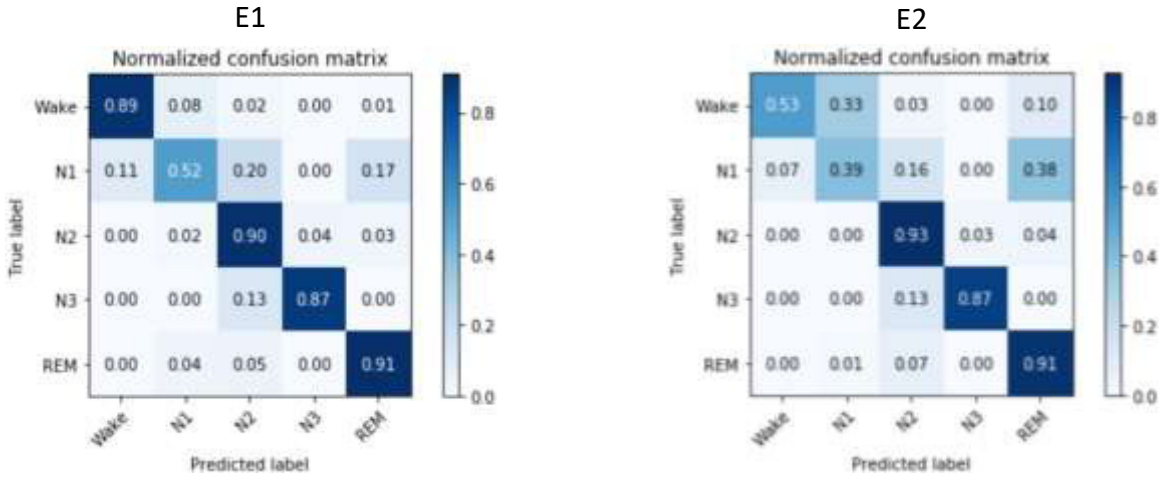

***Figure 27**: Resulting confusion matrices for ParaNet.*

ParaNet used a deep CNN and RNN to process data in parallel, before two common FC layers were used to make the final classification. ParaNet achieved the joint highest accuracy in E1, and the 3rd highest in E2. Specifically, ParaNet shared the highest accuracy in classification of stage N1 with ResNet in E1. This is completely countered in E2, as ParaNet is the model that struggles with N1 the most, only classifying it correctly 39% of the time. In E2, ParaNet performed better than Z3 on 7 out of 10 participants, with an overall accuracy increase of 0.5%. Additionally, ParaNet was the best model at classifying wake data in E2. ParaNet boasts fast training times for the complexity of the model and somehow achieves more efficient training times per epoch that dCNN even though it uses a more complex architecture and has nearly twice the parameters.

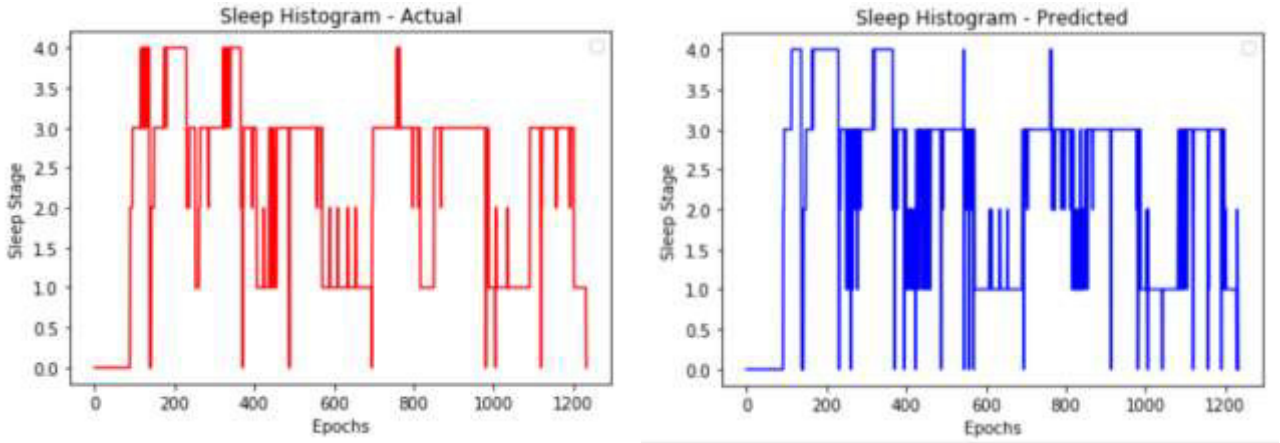

***Figure 28**: A actual sleep histogram from one participant in DS1 (left) VS Multi2D's prediction for the same participant (right).*

5.2 Evaluation of Model Performance

In general, all models achieved a good level of accuracy across both tests. Models were able to capture the cyclic structure of sleep which can be seen in **Figure 28**. Classification performance across all models was similar, especially in E1, varying by only 0.8% but there are some clear trends in the data. It appears that only deepening a CNN is not a sufficient method to increase accuracy as in E1 sCNN outperformed dCNN. For an overall increase in accuracy, models which used novel features were required, such as splitting input channels, using ideas inspired from DSP, and using RNNs to learn different features from the data. This was the case in all but one of the models Multi1D. This anomaly occurred as previously discussed, because the model overfit to the training data too quickly. A possible reason for the success of these ideas over solely increasing the depth of a model is that the input data did not have many complex features. The input data was one dimensional; deeper convolutional networks have had great success when applied to two dimensional image data because they are accustomed to complex feature detection. Instead of learning more complex features, the novel models learnt more high-level features in general. A combination of these features leads to a more accurate final classification.

Another trend of note is the specific success of models with recurrent layers. Previous research had shown the promise of this technology but I feel the results are truly clear that these networks hold the most potential going forward. Overall there is much less research into these networks for classification, specifically in TSC problems. My experiments show that these networks should be the focus of TSC research going forward. As expected, all models performed better at classifying all other stages than N1. This is somewhat due to the uneven spread of classes in the datasets; additionally, the effect of this can also be seen in E2 with the wake class receiving an average accuracy of 51.4% because DS2 made up of only 6.3% of wake epochs. It is worth noting however that stage N1 is notoriously the hardest stage to score. There is much evidence of low human inter-score agreement for this stage [73]. Therefore, the results for this stage are not as bad as they first appear.

As to be expected, the performance of models was somewhat lower when testing the transferability of models on a new dataset in E2. The results for this experiment are still promising, showing that all models transferred relatively well. The most exciting result was for MultiF, achieving the highest accuracy score and only lowering its overall accuracy between tests by 1.6%. This suggests having a wide variety of filter sizes means that the model can learn patterns in the data that can be more easily transferred across to new datasets. When comparing all of the E2 results against Z3, it is clear that my models beat the commercial software in the majority of tests. Half of the models achieved higher overall accuracy. This is an outstanding result that suggests modern advances in TSC can outperform current methods for this specific problem. However, a true test of transferability would need to be performed to confirm these results. A 3rd experiment was investigated where the models were trained in DS1 and only tested on DS2, with no transfer of data between the sets. The accuracy results dropped significantly to 40-50%. This is because of the lack of training data. To truly test a robust measure of accuracy, larger datasets would be needed.

The tests could not be subject to repeated statistical analysis due to the fact that the time needed to carry out 100's of repetitions was unfeasible for the scope of the project; models take hours to retrain even on Hawk, this meant the project did not have to time or computational resources necessary. That being said, these tests were repeated several times and the results stayed mostly consistent, with CRNN and MultiF being the best two performing models overall.

5.3 System Evaluation

Clearly this project is mostly research focused, however it is still worth evaluating the tools that were developed to help with the process. The system that was developed to design the models and implement the experiments performed sufficiently. It was as basic as necessary. The two main features, the data generator and model factory, were the bare bones of what was needed to efficiently and effectively carry out testing of multiple ANN architectures. Without these two developments, the project would have been tedious; I could not have performed the extensive testing that was required, and the variety of models would have been much more reserved. That being said more features could have been developed to assess the results of the experiments. Custom scripts were used for each experiment to produce the final diagrams such as confusion matrices and sleep histograms. This was a time consuming process that could have been avoided if additional tools were implemented to aid evaluation. Having to learn almost every aspect of this project from scratch however, the system was a good first effort and I expect development to continue into the future.

Keras was the correct choice of development tool to be used in the project. I found that there was plenty of online documentation for every aspect of initial model development. When delving into novel ideas, Keras' functional API was incredibly flexible and allowed for much experimentation with model design that had not been considered in the literature. As research and development in the field of deep learning improves, I expect superior tools to emerge; in the next 10 to 20 years, a project of this scope could be considered relatively simple. As for the development environment, I definitely made use of the best resources available to me with Hawk. It allowed me to perform extensive test and experiments, the scale of which would not have been possible for a regular bachelor project.

6. Future Work

The project has several directions to explore going forward. The main two ideas are either an applied system, or further theoretical research. Here is a summary of how my ideas grew throughout the project and where I see the project going in the future.

6.1 Continued Model Research

The results of the project are impressive and inspire further research to solidify the conclusions made. Applying the models to larger datasets would be the best path going forward to gain an additional accurate assessment of model performance. This would specifically allow for more rigorous testing of models. The transfer test performed in E2 was not a robust measure of model transferability due to the fact that the datasets used were small so participants from the same dataset were used in both training and testing. Future exploration of a dataset similar to ISRUC-SLEEP would be the best metric of model performance going forward. Furthermore, this would improve model performance on stage N1 and wake specifically, as models had much less data for these stages comparatively.

Another avenue for continued model research is to explore other model designs. As can be seen from my results, novel models hold the most promise going forward to push classification accuracy above 90%. I think research into a combined model, using the best parts from each of the models in this project, could be the next step forward. My personal suggestion would be a combined CNN and RNN that splits the input channels. For the convolutional section of the model I would explore using ideas inspired by DSP. Possibly a model which uses multiple 2D convolutions with varying filter sizes; in essence a combination of Multi2D and MultiF. Ideally this model would take inspiration from CNN-LSTM, the model which I failed to implement. The model would use a CNN to produce a learnt feature for each epoch, to then be processed by a bidirectional LSTM. The model could look at previous and future epochs to assess its position in the sleep cycle.

6.2 Professional Application

The opposite path the project could take is to develop one of the better models into a fully-fledged application to professionally aid with sleep stage scoring, similar to Z3 score or SLEEPNET, see **Figure 29**. The model provided from this project would be the back-end classifier for the application. First a standard pre-processing module would need to be built in order to standardise any input data for the model. If the system would be web-based, a client/server model would need to be theorised and ethical considerations for data protection would need to be discussed. A front-end system with a user interface would be the final part of the overall application. For this I would suggest receiving input from a professional sleep expert in order to gather

information about what features and are specifically needed. Some tools could be transferred from this project but custom newer system would be better in my opinion.

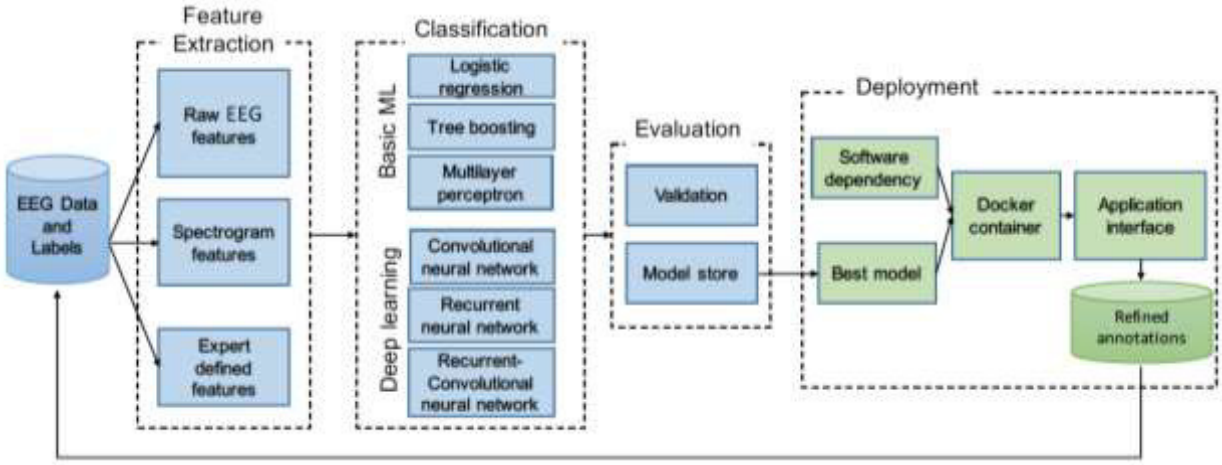

Figure 29: Analytic pipeline of SLEEPNET. Blue colour components correspond to model training module, green colour components correspond to the model deployment module.

7. Conclusions

Here I conclude with a summary of main results and findings, followed by a discussion as to what extent the initial objectives were met. My results show that applying ideas from recent advances in deep learning and TSC hold the most promise for progression in the problem of accurate automated sleep stage scoring. In particular RNNs and ideas from DSP show the most promise. However, much more rigorous testing and evaluation needs to be performed on larger datasets in order to solidify these conclusions.

The project's initial objectives can be summarised into 3 succinct goals, which I believe were met:

1. Develop the tools required to design and ANN architectures for solving the problem of automated sleep stage scoring.
2. Using the developed tools, investigate novel advances in deep learning to design multiple different architectures.
3. Evaluate whether the use of ANNs is an appropriate solution to the problem of automated sleep stage scoring.

The tools I developed were sufficient for the progress of the project. They allowed me to design and test many different types of ANNs, of which 8 achieved state of the art performance and were worth discussing in the project. 5 of the 8 models used novel methods at the time of development; using ideas from wider research in TSC, and theoretical DSP. Looking forward, the project is in a position where my solution can be developed into a professional system to aid with sleep scoring. Overall, from my results, I believe I can conclude that ANNs are an appropriate solution for the problem of automated sleep stage scoring. This is because of the rapid rate of progression in deep learning as a field of research. As wider research is explored, and more powerful tools are produced, I see this type of problem following a similar path as image classification, where the latest use of ANNs achieve classification accuracies of above 95%. In order to see this type of performance in the problem of sleep stage classification, larger public datasets are needed to allow for robust model testing and exploration.

8. Reflection of Learning

In reflection, I am overwhelmingly pleased with the project. The project started strongly as the initial planning and preparation went well. Although the time scale was a little ambitious, and some of my initial objectives were not fulfilled, this actually led to a more focused and streamlined project. This specifically allowed me to explore deeply into one area of the project, model research and development. This meant some core objectives were cut, but some extended desirable objectives were completed and even taken further.

I have also acquired valuable communication skills from the project. Supervisor meetings were held weekly throughout the project as well as regular group meetings with other final year students. During these meetings I learnt to present my work clearly and present studies to the rest of the group; improving my skills of public speaking and teaching. Additionally, having access to NaPS lab sleep datasets, I had to communicate with professionals and postgraduates on a regular basis. This has given me an insight to the world beyond university, and has prepared me for the next step in my academic life.

Report writing and understanding is another key area that I have grown in. Before the project I don't believe I had ever read a single technical paper for computer science. Now I have read hundreds and fully understand their value. Reading these reports has also vastly improved my ability to write this final report. Initially, this was a daunting task. I would have even considered myself a bit of a "write-ophobe" (hence my choice of degree). But, after learning from other works, and from the support I've had at university, the writing of this report has flown by and was not as bad as I thought it would be.

Finally, the extensive technical skills I have learnt from the project are invaluable. Along with support of my supervisor, the majority of the project was self-taught, with the exception of programming skills. All of the theory behind sleep medicine, TSC, ANNs, DSP, and all of the practical skills such as model design, data analysis, literature breakdown, and many more, were all new concepts to me. I feel this project is as comparable in value as my whole degree for the new knowledge learnt. Additionally, I can't stress enough how this knowledge has truly been ingrained into me; I had to fully understand everything I was doing before I could use it. Compared to final year exams, where sometimes the concepts behind topics do not have to be understood as long as the superficial answer is known, or at least they don't have to be understood for an extended period of time. After reflecting on all of these skills, I now plan to take everything I have learnt into future endeavours, and to continue with deep learning based work for the foreseeable future.

9. References

1. The Sleep Council. 2017. *The Great British Bedtime Report.* [https://sleepcouncil.org.uk/wp-content/uploads/2018/04/The-Great-British-Bedtime-Report-2017.pdf]

2. Rasch, B. and Born, J. 2013. About sleep's role in memory. *Physiological Reviews.* [https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3768102/]

3. Alhola P. and Polo-Kantola P. 2007. Sleep deprivation: Impact on cognitive performance. *Neuropsychiatric disease and treatment.* [https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2656292/]

4. Krizhevsky, A., Sutskever, I. and Hinton, G. 2017. ImageNet classification with deep convolutional neural networks. *Communications of the ACM* [http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf]

5. De Vries, S., Garre, F., Engbers, L., Hildebrandt, V. and Van Buuren, S. 2011. Evaluation of Neural Networks to Identify Types of Activity Using Accelerometers. *Medicine & Science in Sports & Exercise.* [https://pdfs.semanticscholar.org/8c5d/0b6d7b96f090b1a748254e2f8c66e01ec4a7.pdf]

6. Yang J.B., Nguyen, M.N., San, P.P., Li, X.L. and Krishnaswamy, S. 2015. Deep convolutional neural networks on multichannel time series for human activity recognition. *IJCAI'15 Proceedings of the 24th International Conference on Artificial Intelligence.* [https://www.aaai.org/ocs/index.php/IJCAI/IJCAI15/paper/view/10710/11297]

7. Zheng Y., Liu Q., Chen E., Ge Y. and Zhao J.L. 2014. Time Series Classification Using Multi-Channels Deep Convolutional Neural Networks. *Web-Age Information Management. WAIM 2014. Lecture Notes in Computer Science* [https://link.springer.com/chapter/10.1007/978-3-319-08010-9_33]

8. Iber, C. 2007. The AASM manual for the scoring of sleep and associated events. *American Academy of Sleep Medicine.* [https://aasm.org/clinical-resources/scoring-manual/]

9. McCulloch, W. and Pitts, W. 1990. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology.* [http://waldirbertazzijr.com/wp-content/uploads/2018/10/mcp.pdf]

10. He, K., Zhang, X., Ren, S. and Sun, J. 2016. Deep Residual Learning for Image Recognition. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* [https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf]

11. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R. and Fei-Fei, L. 2014. Large-Scale Video Classification with Convolutional Neural Networks, *IEEE Conference on Computer Vision and Pattern Recognition.* [https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Karpathy_Large-scale_Video_Classification_2014_CVPR_paper.pdf]

12. Khan, J., Wei, J., Ringnér, M., Saal, L., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C., Peterson, C. and Meltzer, P. 2001. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine.* [https://www.nature.com/articles/nm0601_673]

13. Van Gerven, M. and Bohte, S. 2017. Editorial: Artificial Neural Networks as Models of Neural Information Processing. *Frontiers in Computational Neuroscience.* [https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5742181/]

14. Svozil, D., Kvasnicka, V. and Pospichal, J. 1997. Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems.* [https://www.sciencedirect.com/science/article/pii/S0169743997000610]

15. Hecht-Nielsen, R. 1988. Theory of the backpropagation neural network. *Neural Networks.* [https://www.sciencedirect.com/science/article/pii/B9780127412528500108]

16. Fukushima, K. 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics.* [https://www.rctn.org/bruno/public/papers/Fukushima1980.pdf]

17. Pokharna, H. (2019). The best explanation of Convolutional Neural Networks on the Internet!. *(online) Medium.* [https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8]

18. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L. and Muller, P. 2019. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery.* [https://arxiv.org/pdf/1809.04356.pdf]

19. Bekkers, J.M. 2011. Pyramidal neurons. *Cell Press, Current Biology.* [https://www.cell.com/current-biology/pdf/S0960-9822(11)01198-5.pdf]

20. Wang, C.F. 2019. The Vanishing Gradient Problem. *(online) Towards Data Science* [https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484]

21. Sahoo, S. 2018. Residual blocks—Building blocks of ResNet. *(online) Towards Data Science.* [https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec]

22. Zhang, A., Lipton, Z., Li, M. and Smola, A. (2018). Dive into Deep Learning, chapter 8. *(online).* [https://d2l.ai/chapter_recurrent-neural-networks/index.html

23. Hochreiter, S. and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation.* [https://www.bioinf.jku.at/publications/older/2604.pdf]

24. Chen, K., Zhou, Y. and Dai, F. 2015. A LSTM-based method for stock returns prediction: A case study of China stock market. *IEEE International Conference on Big Data* [https://ieeexplore.ieee.org/abstract/document/7364089]

25. Graves A., Fernández S., Schmidhuber J. 2005. Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition. *Artificial Neural Networks: Formal Models and Their Applications – ICANN.* [https://link.springer.com/chapter/10.1007/11550907_126]

26. Smith, S. 1997. The scientist and engineer's guide to digital signal processing, chapter 26. *California Technical Pub.* [https://www.dspguide.com/ch26.htm]

27. Jueschke, P. and Fischer, G. 2017. Machine learning using neural networks in digital signal processing for RF transceivers. *IEEE AFRICON.* [https://ieeexplore.ieee.org/document/8095513]

28. Weisstein, E. Convolution Theorem. *(online) MathWorld--A Wolfram Web Resource.* [http://mathworld.wolfram.com/ConvolutionTheorem.html]

29. Electronics Tutorials. Power Diodes and Rectifiers. *(online) Electronics Tutorials.* [https://www.electronics-tutorials.ws/diode/diode_5.html]

30. Chollet, F. Keras Documentation. *(online).* [https://keras.io/]

31. Google Brain Team. TensorFlow. *(online).* [https://www.tensorflow.org/]

32. NVDIA. CUDA Zone. *(online) NVIDIA Developer.* [https://developer.nvidia.com/cuda-zone]

33. Apple, Inc. MacBook Pro (Retina, 15-inch, Mid 2014) - Technical Specifications. *(online) Apple Support.* [https://support.apple.com/kb/sp704?locale=en_US]

34. Anaconda, Inc. Anaconda. *(online).* [https://www.anaconda.com/]

35. Project Jupyter. JupyterLab. *(online) Github.* [https://github.com/jupyterlab/jupyterlab]

36. Alphabet Inc. Colaboratory FAQs. *(online) Google research.* [https://research.google.com/colaboratory/faq.html]

37. Project Jupyter. Jupyter.org. *(online).* [https://jupyter.org/]

38. Supercomputing Wales. About SCW. *(online).* [https://www.supercomputing.wales/about/]

39. Supercomputing Wales Portal. About Hawk. *(online).* [https://portal.supercomputing.wales/index.php/about-hawk/]

40. SchedMD. SLURM Documentation. *(online).* [https://slurm.schedmd.com/documentation.html]

41. Biswal, S., Kulas, J., Sun, H., Goparaju, B., Westover, M., Bianchi, M. and Sun, J. 2017. SLEEPNET: Automated Sleep Staging System via Deep Learning. *(online) arxiv.* [https://arxiv.org/pdf/1707.08262.pdf]

42. Hsu, Y., Yang, Y., Wang, J. and Hsu, C. 2013. Automatic sleep stage recurrent neural classifier using energy features of EEG signals. *Neurocomputing.* [https://www.sciencedirect.com/science/article/pii/S0925231212008387]

43. Vilamala, A., Madsen, K.H. and Hansen, L. K. 2017. Deep convolutional neural networks for interpretable analysis of EEG sleep stage scoring. *IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP).* [https://ieeexplore.ieee.org/abstract/document/8168133]

44. Lai, S., Xu, L., Liu, K. and Zhao. 2015. Recurrent Convolutional Neural Networks for Text Classification. *AAAI Conference on Artificial Intelligence.* [https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9745/9552]

45. Neurobit Tech. Z3Score. *(online).* [https://z3score.com/]

46. Patanaik, A., Ong, J., Gooley, J., Ancoli-Israel, S. and Chee, M. 2018. An end-to-end framework for real-time automatic sleep stage classification. *Sleep.* [https://z3score.com/validation]

47. Schrempf, W., Brandt, M., Storch, A. and Reichmann, H. 2014. Sleep Disorders in Parkinson's Disease. *Journal of Parkinson's disease.* [https://www.ncbi.nlm.nih.gov/pubmed/24796235]

48. Malafeev, A., Laptev, D., Bauer, S., Omlin, X., Wierzbicka, A., Wichniak, A., Jernajczyk, W., Riener, R., Buhmann, J. and Achermann, P. 2018. Automatic Human Sleep Stage Scoring Using Deep Neural Networks. *Frontiers in Neuroscience.* [https://www.frontiersin.org/articles/10.3389/fnins.2018.00781/full]

49. Breiman, L. 2001. Random Forests. *Machine Learning.* [https://link.springer.com/article/10.1023/A:1010933404324]

50. Malafeev, A., Laptev, D., Bauer, S., Omlin, X., Wierzbicka, A., Wichniak, A., Jernajczyk, W., Riener, R., Buhmann, J. and Achermann, P. 2018. Automatic Human Sleep Stage Scoring Using Deep Neural Networks, Supplementary Material. *Frontiers in Neuroscience.* [https://www.frontiersin.org/articles/10.3389/fnins.2018.00781/full#supplementary-material]

51. Wang, Z., Yan, W. and Oates, T. 2016. Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline. *(online).* [https://arxiv.org/pdf/1611.06455.pdf]

52. Serrà, J., Pascual, S. and Karatzoglou, A. 2018. Towards a universal neural network encoder for time series. *(online).* [https://arxiv.org/pdf/1805.03908.pdf]

53. Synced. 2017. A Brief Overview of Attention Mechanism. *(online) Medium.* [https://medium.com/syncedreview/a-brief-overview-of-attention-mechanism-13c578ba9129]

54. Baydogan, M. 2018. Baydogan's Dataset. *(online)* [http://www.mustafabaydogan.com/]

55. The SciPy Community. 2018. numpy.lib.format. *(online).* [https://www.numpy.org/devdocs/reference/generated/numpy.lib.format.html]

56. Amidi, A. and Amidi, S. 2018. A detailed example of data generators with Keras. *(online) Stanford.* [https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly]

57. Ackermann, N. 2018. Introduction to 1D Convolutional Neural Networks in Keras for Time Sequences. *(online) Good Audience.* [https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf]

58. Allaire, J., Chollet, F. and RStudio. Spatial 1D version of Dropout. *(online) Keras.rstudio.com* [https://keras.rstudio.com/reference/layer_spatial_dropout_1d.html]

59. Brownlee, J. 2019. Architectural Innovations in Convolutional Neural Networks for Image Classification. *(online) Machine Learning Mastery.* [https://machinelearningmastery.com/review-of-architectural-innovations-for-convolutional-neural-networks-for-image-classification/]

60. Schirrmeister, R., Springenberg, J., Fiederer, L., Glasstetter, M., Eggensperger, K., Tangermann, M., Hutter, F., Burgard, W. and Ball, T. 2017. Deep learning with convolutional neural networks for EEG decoding and visualization. *Human Brain Mapping.* [https://arxiv.org/pdf/1703.05051.pdf]

61. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. 2015. Going deeper with convolutions. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* [https://ai.google/research/pubs/pub43022]

62. Schuster, M. and Paliwal, K. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing.* [https://ieeexplore.ieee.org/abstract/document/650093]

63. Graves, A., Jaitly, N. and Mohamed, A. 2013. Hybrid speech recognition with Deep Bidirectional LSTM. *IEEE Workshop on Automatic Speech Recognition and Understanding.* [https://ieeexplore.ieee.org/abstract/document/6707742]

64. Karim, F., Majumdar, S., Darabi, H. and Harford, S. 2018. Multivariate LSTM-FCNs for time series classification. *Neural Networks.* [https://arxiv.org/pdf/1801.04503.pdf]

65. Nielsen, M. 2018. Improving the way neural networks learn. *(online).* [http://neuralnetworksanddeeplearning.com/chap3.html]

66. Keras. Keras Optimizers. *(online).* [https://keras.io/optimizers/]

67. Keras. Keras Callbacks. *(online).* [https://keras.io/callbacks/]

68. Brownlee, J. 2017. CNN Long Short-Term Memory Networks. *(online) Machine Learning Mastery.* [https://machinelearningmastery.com/cnn-long-short-term-memory-networks/]

69. Keras. Keras Layer Wrappers. *(online).* [https://keras.io/layers/wrappers/]

70. Khalighi, S., Sousa, T., Santos, J. and Nunes, U. 2016. ISRUC-Sleep: A comprehensive public dataset for sleep researchers. *Computer Methods and Programs in Biomedicine.* [https://www.sciencedirect.com/science/article/pii/S0169260715002734]

71. Cui, Z., Zheng, X., Shao, X. and Cui, L. 2018. Automatic Sleep Stage Classification Based on Convolutional Neural Network and Fine-Grained Segments. *Complexity.* [https://www.hindawi.com/journals/complexity/2018/9248410/]

72. Yang, Y., Zheng, X., & Yuan, F. 2018. A Study on Automatic Sleep Stage Classification Based on CNN-LSTM. *Proceedings of the 3rd International Conference on Crowd Science and Engineering - ICCSE'18*. [https://doi.org/10.1145/3265689.3265693]

73. Danker-Hopfe, H., Kunz, D., Gruber, G., Klösch, G., Lorenzo, J., Himanen, S., Kemp, B., Penzel, T., Röschke, J., Dorn, H., Schlögl, A., Trenker, E. and Dorffner, G. 2004. Interrater reliability between scorers from eight European sleep laboratories in subjects with different sleep disorders. *Journal of Sleep Research*. [https://www.ncbi.nlm.nih.gov/pubmed/14996037]

74. Deshpande, A. 2017. A Beginner's Guide to Understanding Convolutional Neural Networks. *(online) Github*. [https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/]

75. Olah, C. 2015. Understanding LSTM Networks -- colah's blog. *(online) Github*. [http://colah.github.io/posts/2015-08-Understanding-LSTMs/]