EEG Sleep Stage Classification and Prediction using Entropy Feature Extraction

Abdurrahmaan Idris-Agbabiaka

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical and Electronic Engineering

Eastern Mediterranean University
September 2022
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science in Electrical and Electronic Engineering.

Assoc. Prof. Dr. Rasime Uyguroğlu
Chair, Department of Electrical and Electronic Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Electrical and Electronic Engineering.

Asst. Prof. Dr. Shahla Azizi Alikamar
Supervisor

Examining Committee

1. Prof. Dr. Hasan Amca

2. Assoc. Prof. Dr. Kamil Yurtkan

3. Asst. Prof. Dr. Shahla Azizi Alikamar

ABSTRACT

Over the past 2 decades, EEG research has rapidly advanced due to the discoveries of active electrodes, machine learning algorithms and more interest in the field of neuroscience. This has led to EEG research being much cheaper and accessible, but due to its stigma of previously being an extremely high-end research field and the sensitivity of the data required for the research, it has resulted in even more complex EEG experiments and applications.

This thesis proposes a method of predicting sleep stages using a single EEG channel input BCI. The experiments were performed on Sleep-EDF's Sleep cassette and Sleep telemetry datasets. We used entropy feature extraction to identify different sleep stages which were then fed into a classifier enabling us to then predict subsequent sleep stages. Entropy has been proven to be able to numerically depict non-linear time series data and it has been used in many other EEG BCIs.

Three classifiers were compared to ascertain their various performances. SVM showed the most stable results at 75 percent prediction accuracy. While KNN has a higher maximum prediction accuracy than SVM at 88 percent accuracy, it has proven to be unstable and varies depending on the iterations and number of inputs. The final classifier used was the NN classifier which should result in the best accuracies compared to the SVM and KNN at 87 percent or higher prediction accuracies.

The results show that it is indeed possible to predict upcoming sleep stages with a single EEG channel with accuracies greater than 75 percent depending on the methods used.

Keywords: EEG, Feature Extraction, BCI, Entropy, Classifier

ÖZ

Son 20 yılda, aktif elektrotların keşfi, makine öğrenme algoritmaları ve sinirbilim alanına daha fazla ilgi nedeniyle EEG araştırmaları hızla ilerlemiştir. Bu, EEG araştırmalarının çok daha ucuz ve erişilebilir olmasına yol açtı, ancak daha önce son derece üst düzey bir araştırma alanı olarak damgalanması ve araştırma için gereken verilerin hassasiyeti nedeniyle, daha da karmaşık EEG deneyleri ve uygulamaları ile sonuçlandı.

Bu tez, tek bir EEG kanal girişi BCI kullanarak uyku aşamalarını tahmin etmek için bir yöntem önermektedir. Deneyler, Sleep-EDF'nin Uyku kaseti ve Uyku telemetri veri setleri üzerinde yapıldı. Farklı uyku aşamalarını belirlemek için entropi öznitelik çıkarımı kullandık ve bu aşamalar daha sonra sonraki uyku aşamalarını tahmin etmemizi sağlayan bir sınıflandırıcıya aktarıldı. Entropinin, doğrusal olmayan zaman serisi verilerini sayısal olarak gösterebildiği kanıtlanmıştır ve diğer birçok EEG BCI'da kullanılmıştır.

Çeşitli performanslarını belirlemek için üç sınıflandırıcı karşılaştırıldı. SVM, yüzde 75 tahmin doğruluğunda en kararlı sonuçları gösterdi. KNN, yüzde 88 doğrulukla SVM'den daha yüksek bir maksimum tahmin doğruluğuna sahip olsa da, kararsız olduğu kanıtlanmıştır ve yinelemelere ve girdi sayısına bağlı olarak değişiklik gösterir. Kullanılan son sınıflandırıcı, yüzde 87 veya daha yüksek tahmin doğruluklarında SVM ve KNN'ye kıyasla en iyi doğrulukla sonuçlanması gereken NN sınıflandırıcıydı.

Sonuçlar, kullanılan yöntemlere bağlı olarak yüzde 75'ten daha fazla doğrulukla tek bir EEG kanalı ile yaklaşan uyku aşamalarını tahmin etmenin gerçekten mümkün olduğunu gösteriyor.

Anahtar Kelimeler: EEG, Özellik Çıkarımı, BCI, Entropi, Sınıflandırıcı

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to Assist. Prof. Dr. Shahla Azizi Alikamar for her supervision, assistance, and guidance throughout this thesis, as well as providing me with incredible insight and advice during the process. Most importantly was the constant advice given by her on how to improve and better my thesis. Without her advice, this thesis would have been a far cry from what it currently is, and I could not be more grateful for that. Her thoughts, experiences, advice, and ambitions have enriched and motivated my development as a student, teacher, and person. She is a true mentor, and it is my greatest honour to be her student.

I would also like to express my gratitude to my dear parents who are the pillars that supported me in all aspects to whom I owe far too much. I would also like to thank Al-Khattab Ali Yaseen and all my friends who provided immense assistance and support throughout this whole ordeal, they are like my second family and I wish the best for them.

Lastly, I would like to thank all the electrical department staff for their work and effort in creating such a great learning, developing, and working environment filled with helpful, friendly and wonderful people and I am proud of being a student of this department.

To everyone not mentioned but in one way or another helped me during my time working on this study from the bottom of my heart, thank you very much.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AR	Autoregressive
BCI	Brain Computer Interface
CNN	Convolutional Neural Network
DTFT	Discrete Time Fourier Transform
EEG	Electroencephalogram
EOG	Electro-oculography
EMG	Electromyography
EP	Evoked Potential
FFT	Fast Fourier Transform
fMRI	Functional Magnetic Resonance Imaging
K	Number of Nearest Neighbours in KNN Algorithm
KNN	K-nearest Neighbour
MEG	Magnetoencephalography
MRI	Magnetic Resonance Imaging
NN	Neural Network
PSD	Power Spectral Density
RBF	Gaussian Radial Basis Function
ROC	Receiver Operating Characteristic Curve
SC	Sleep Cassette
ST	Sleep Telemetry
SVM	Support Vector Machine

Chapter 1

INTRODUCTION

There are currently many people around the world plagued with various sleep-related disorders like insomnia, sleep apnoea, narcolepsy and more. These strings of disorders tend to be more challenging than most to diagnose and treat with most of the treatments being done by the patients themselves in home settings without advanced medical equipment. This thesis aims to achieve a non-complex method for monitoring and controlling patients with sleep disorders by classifying and predicting their sleep states.

Brain computer interface (BCI) is a recent form of communication that allows people the possibility of monitoring, communicating, controlling external devices or navigating a virtual world using only the power of their thoughts [1]. It also allows us to research and understand even more of the great mystery organ that is the human brain and learn a lot more than we could before.

1.1 Creating a BCI

When creating a BCI, the major steps followed as depicted in figure 1 are first and foremost, signal acquisition in which different techniques are used to acquire information from a human brain, following which said data is put through the pre-processing section where it is processed to make the signal more suitable for use in the BCI by re-organising the data into a suitable format or by removing various kinds of noise disturbances in preparation for the next step which is feature extraction. In this step, crucial information is extracted from the already pre-processed data by using one

or a variety of techniques to create as robust of a feature space as possible. This allows the following step which is classification to focus on using various mostly machine learning algorithms to classify and make predictions based on the inputs received from the feature extraction steps. Finally, after creating the model, we complete the BCI by testing it on various parameters like accuracy, computational speeds etc. This allows us to further optimise the system to better suit the intended application.

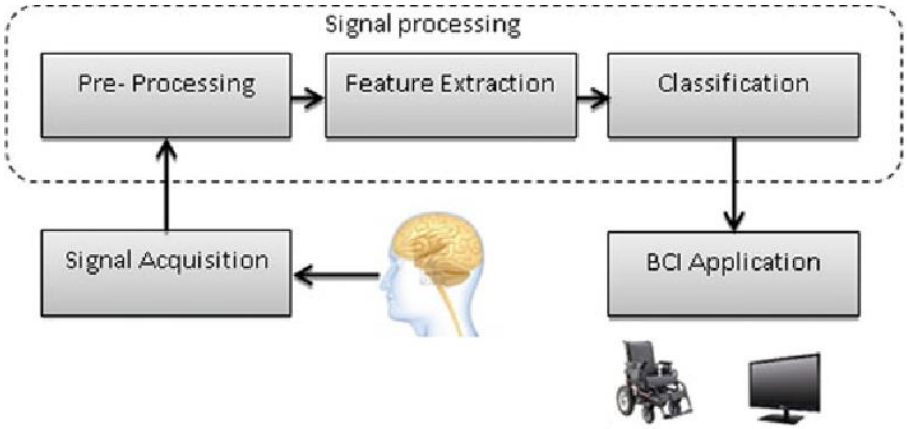

Figure 1: Steps taken to create a BCI [2]

1.1.1 Signal Acquisition

There are a variety of different technologies used in the signal acquisition process each with its pros and cons associated with them. To name a few:

- Electroencephalography (EEG): EEG is a method to record an electrogram of the electrical activity on the scalp which has proven to depict the activity of the brain. It is usually non-invasive, as EEG sensors (electrodes) are placed along the scalp.
- Magnetoencephalography (MEG): MEG is a functional neuroimaging technique used to map brain activity by recording magnetic fields using highly sensitive magnetometers. These magnetic fields are produced as a result of natural occurring electric currents in the brain.

- Functional Magnetic Resonance Imaging (fMRI): Medical resonance imaging (MRI) is a medical imaging technique that generates images of the organs in the body by using strong magnetic fields, gradients, and radio waves. fMRI measures brain activity by detecting changes associated with blood flow.

These various techniques have their advantages and disadvantages with MEG, and fMRI being non-portable, expensive, and requiring skilled technicians to use. The other methods excluding EEG require surgery making them even more difficult to use compared to the other technologies. In the case of EEG, apart from its spatial resolution disadvantage, it has many advantages of being portable, non-invasive, relatively inexpensive and easy to work with making it the most suitable choice for practical BCIs and by extension this thesis [3]. Below is a table comparing different BCI technologies.

Table 1: A comparative table of various brain activity monitoring technologies [4]

	Electrode placement	Frequency range	Typical amplitude	Advantage	Disadvantage
EEG	Scalp contact (usually cap)	0.1-100 Hz	< 100 µv	Non-invasive, portability	Spatial resolution
MEG	Remote (e.g. helmet)	2 - 100 Hz	< 10 - 14 Tesla	Non-invasive	Non-portability
MRI	Remote (bed inside a tubular equipment)	-	-	Non-invasive, non-contact	Non-portability, temporal resolution
ECoG (iEEG)	Intracranial cortical surface	1 - 100 Hz	Several hundred µv	quality, special and temporal resolution	Surgery required, highly invasive
Microelectrode (Microwire)	Intracranial, intracortical	0.5 – 5 kHz	A few mv	quality, simultaneous monitoring of multiple sites	High risk surgery requirement, gliosis and other medical complications
MEA	Intracranial, intracortical	0.5 – 5 kHz	A few mv		High risk surgery requirement, gliosis and other medical complications

1.1.2 Pre-processing

Due to EEG signals having very low amplitudes, as well as the frequency bands they occupy colliding with other commonly found electromagnetic signals, EEG signals tend to possess a lot of noise (aka. EEG Artifacts). Thus, in this step, the raw EEG signals are refined to achieve the best quality with various methods like amplifying, filtering and most importantly noise elimination.

1.1.3 Feature Extraction

In this step, person-specific information or features are computed using the pre-processed signals and combined into feature vectors using different techniques. This step is done to decrease redundancy and obtain the most important information in the signals, which helps facilitate easier and more efficient classification and prediction. Feature extraction is a crucial step in any BCI and there are many different feature extraction methods employed in various EEG based BCIs namely:

- Discrete Time Fourier Transform (DTFT): DTFT is a strong tool that allows us to find the spectrum of a finite-duration signal. This allows us to examine the signal in the frequency domain.
- Power Spectral Density (PSD): Power Spectral Density (PSD measures the power content of a signal versus its frequency. The amplitude of the PSD is normalized by the spectral resolution employed to digitize the signal.
- Wavelet Transform: This is used as an alternative to Fourier transform by decomposing a function to a set of wavelets. A wavelet is a wave-like oscillation that is localized in time.
- Entropy: Entropy is a numerical depiction of the amount of chaos in a given system [3]. It shows the degree to which one can predict each part of the trajectory based on their behaviours. Basically, the higher the entropy, the more

complex or chaotic systems the system is and the more unpredictable it becomes [5].

Other less common feature extractions methods used in EEG based BCIs include Linear Complexity, Energy spectrum density, etc.

1.1.4 Feature Reduction and Selection

In the case of multiple features being extracted, this step is required to reduce redundancy and further optimise the system. This step mostly consists of evaluating extracted features and selecting the most relevant ones to be used in the classification step. This step could be omitted if only the required features were extracted during the feature extraction step.

1.1.5 Classification

This step consists of model training which constructs a model using the extracted features and testing them. As a result, we can measure the accuracy of our model in predicting upcoming sleep states. Similar to the case of feature extraction there are likewise multiple classification techniques used in EEG based BCIs namely:

- Support vector machine (SVM): it is a machine learning algorithm used for both classification and regression. Its goal is to find a hyperplane in an N-dimensional space that distinctly classifies the data points [6].
- Neural Networks (NN): A NN is a method that imitates the human brain in the way it processes data. It achieves this by using interconnected nodes (neurons) in a layered structure similar to the human brain.
- K-Nearest Neighbour (KNN): The KNN algorithm, is a non-parametric, supervised learning classifier, that can classify and predict the grouping of single data points by using proximity to other points [7].

There are other classification techniques used but the aforementioned 3 are the most widely utilized. In this thesis, we use a binary SVM as our base classifier due to its implementary simplicity. Other classifiers will then be implemented and their performances in sleep stage prediction will be compared.

1.2 Problem Statements and Research Question

1.2.1 Problem Statements

The goal of this thesis was to design a simple EEG system that allows people to identify and potentially treat mental illnesses that otherwise take time and are expensive to diagnose. A lot of information on mental illnesses can be gained by observing the different stages of sleep [8]. However, due to the complexity of multichannel EEG systems both in terms of development and operation, it proved to be counter intuitive to the goal of this thesis. Thus, we identify our main aim to be sleep stage classification and prediction using single-channel EEG data. The challenge being faced is a way to extract adequate information from a single EEG channel to facilitate high-accuracy classification. To achieve this, we first need to obtain an appropriate database to work with.

A database can be obtained by an experimental recording of volunteers but due to many constraints i.e., time, equipment, data sensitivity and skilled personnel, the database needed to be sourced from the internet. The criteria for the sourced database were for it to be open source, have credible citations with usage in many other pieces of research and to be already filtered for some noise. The chosen database which fit all the criteria, and more was the Sleep-EDF database.

Feature extraction is an important step in any BCI research as it directly affects the results from the classifiers. Due to the discovery of various entropy feature extraction methods being more recent, BCIs using these extraction methods have not been fully explored yet thus there is room to experiment with the feasibility of this method.

Accuracy is also a very crucial requirement of any BCI system thus due to the sourced database being sleep data, we established our goal to enable sleep stage prediction using entropy feature extraction not just with a single EEG channel but also to an acceptable degree of accuracy.

1.2.2 Research Questions

With the knowledge of the stated problems, this thesis aims to achieve the following objectives:

1. Construct a BCI system using a single EEG channel data input
2. Sleep stage classification and prediction using entropy feature extraction on single channel data.
3. Identify the effects of different classifiers on prediction accuracy

1.3 Research Methodology

As shown in figure 2, the datasets used in this thesis were obtained from the Sleep-EDF database, which were then pre-processed into a suitable format for feature extraction. We then used Entropy feature extraction to acquire information on sleep states with Shannon Entropy as our baseline entropy calculation method.

Finally, different classifiers were used for classification and prediction and their accuracies were compared, with the SVM classifier used as a baseline method.

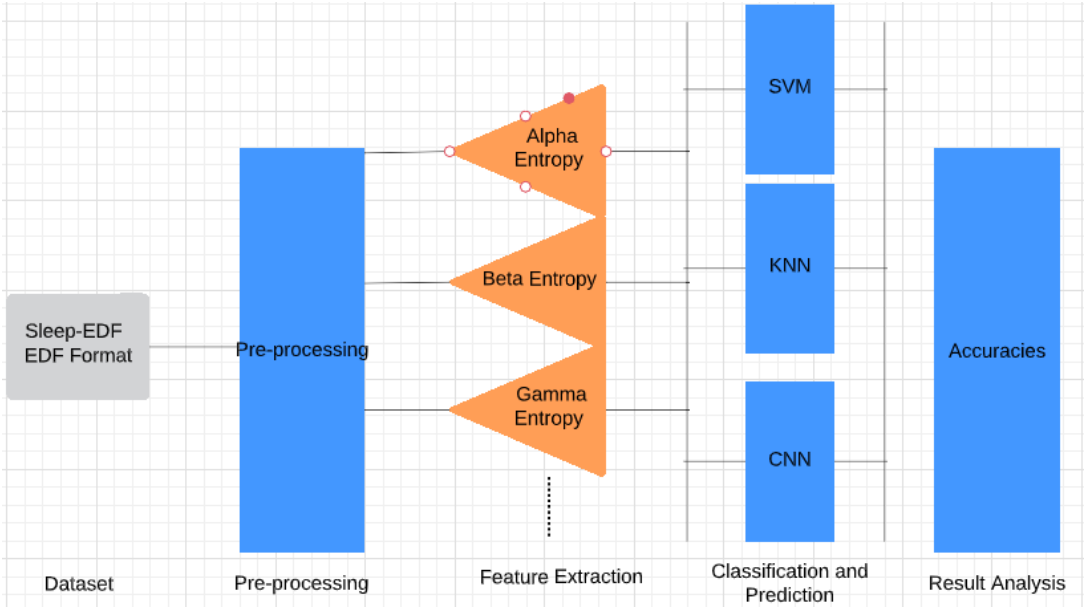

Figure 2: Proposed research methodology

1.4 Contribution of This Thesis

The contributions of this thesis include:

1. Enabling sleep stage classification and prediction using entropy feature extraction on single channel data.
2. Comparing different classification methods for sleep stage prediction.
3. Analysing the effect of increasing input channels to the BCI system.

Chapter 2

LITERATURE REVIEW

In this chapter, we will provide a background on the characteristics of EEG, brainwave patterns, pre-processing methods, feature extraction techniques and different classifiers that can be used for sleep stage classification and prediction.

2.1 EEG Characteristics

2.1.1 EEG Signals

EEG signals are composed of synaptic currents which are mainly related to neuron activities. There are likely at a potential of 60−70mV [3] but can vary depending on synaptic activity. Another facet of EEG signals are Evoked potentials (EP) shown in figure 3. They are produced due to quick changes in membrane potential which is caused by any stimuli such as pressure, light, chemical, touch etc.

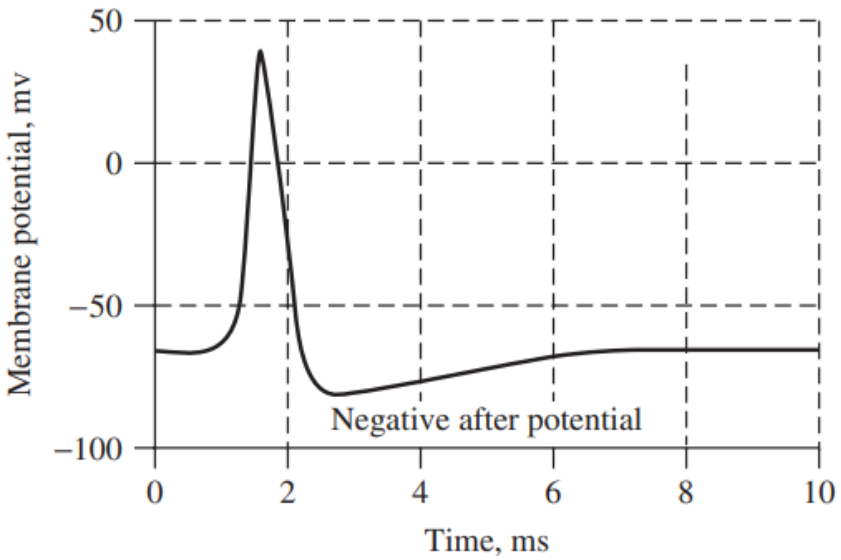

Figure 3: An example of EP, after [3]

To record these signals, electrodes are attached to the scalp and the voltage is measured between the pairs of electrodes these voltages are called channel voltages. There are various positions and contacts of the electrodes with the scalp as shown in figure 4 and they play a big role in EEG signal acquisition [9]. The electrodes can be placed in many ways however, an internationally recognised and most commonly utilised is the 10-20 system EEG placement method. This thesis focuses on the Fpz – Cz and Pz – Oz channels.

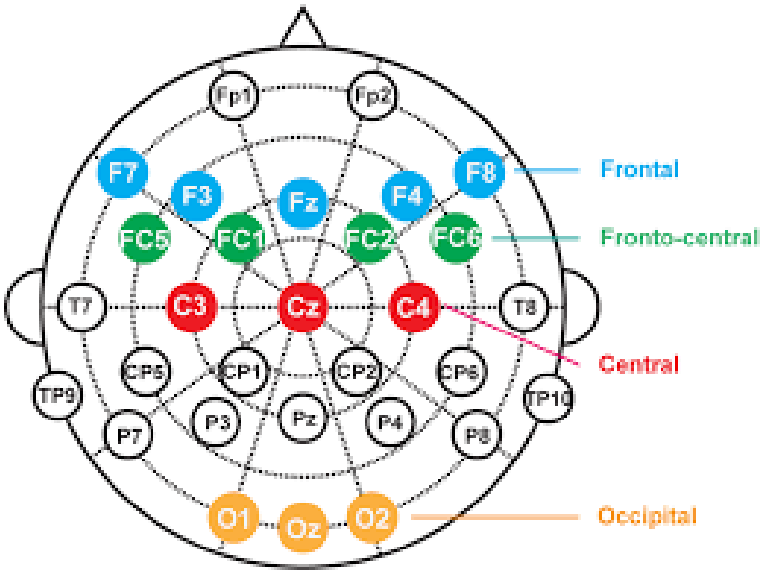

Figure 4: A diagram showing various electrode placement locations [10]

2.1.2 EEG Brain Wave Patterns

The brainwave pattern shows the state of a human's consciousness by variation in amplitudes and frequencies [3]. There are currently five main brainwave patterns separated by their frequency bands most of which were discovered and named by the German psychiatrist Hans Berger the creator of EEG namely alpha (α), beta (β), delta (δ), theta (θ), and gamma (γ):

- Delta waves: EEG delta waves are high-amplitude brain waves with frequencies from 0.5 to 3Hz and are associated with deep sleep stages. They are also

found in other brain functions apart from deep sleep, e.g., high frontal delta waves in awake subjects are associated with cortical plasticity [11].

- Theta waves: Frequencies between 4 and 7 Hz, and usually have amplitudes larger than 20 μV. Mental states like stress, disappointment or frustration are known to cause these waves and can be associated with deep thinking and creativity [9].
- Alpha waves: These waves occur during resting with eyes closed. They disappear during sleep and vanish when concentrating on a specific task. They are identified to have frequencies between 8 and 13 Hz [12].
- Beta waves: These frequencies are mostly seen in an awakened state and have high frequencies with low amplitudes. They are commonly attributed to conscious thought and logical thinking. These waves can be divided into three specific classifications namely, Low beta waves (12–15 Hz), Mid-range beta waves (15–20 Hz) and High beta waves (18–40 Hz) [11].
- Gamma waves: Gamma waves are high-frequency waves low amplitude signals. They are not commonly observed and they relate to the movement of some body parts [3]. They are observed at frequencies greater than 30 Hz.

2.2 EEG Pre-processing

In the case of EEG data, pre-processing usually means cutting out the noise from the data to get closer to the true neural signals. Pre-processing techniques can be generalised into two major categories:

- Spatial filtering: These are techniques used for extracting features with specific spatial distributions by combining data from multiple locations in the brain. Some examples are the Laplacian and Common average reference filters which work by using defined weights to combine electrodes linearly [1].

- Temporal Filtering: This works by separating the signal into smaller time windows and analysing them. Some methods are better suited to smaller time windows like band-pass filtering and autoregressive analysis while Fourier transform is more effective on larger time windows. In band-pass filtering, low-frequency noise (i.e., breathing), can be removed using a high-pass filter with a cut-off frequency of 0.5 Hz. For high-frequency noise, such as electrical noise, a low-pass filter with a cut-off frequency of approximately 50-70 Hz is used [3].

2.3 EEG Based Feature Extraction Methods

Feature extraction methods are used to find new features to increase the pattern space, it is used to separate classes when the original data is not adequate [13].

2.3.1 Fourier Transform

This is the most used signal transform method. This is a mathematical formula as shown in equation 1, that converts a signal sampled in the time or space domains to the frequency domain allowing us to see the frequency components of the signal [14].

$$X(f) = \int x(t) e^{-2j\pi ft} dt \quad (1)$$

2.3.2 Power Spectral Density

Spectral analysis is a standard method used for EEG quantification. The power spectrum shows the distribution of signal power over frequency.

To compute the PSD, we assume that x(n) is a discrete-time signal with n = 0, 1, . . ., N − 1 and a sequence of random variables with zero mean. The PSD is defined as the Discrete time Fourier transform (DTFT) shown in equation 2 of the autocovariance sequence of (n) shown in equation 3 [15].

$$\varphi(\omega) = \sum_{k=0}^{N-1} r(k) e^{-i\omega k}$$

(2)

where the autocovariance sequence r(k) is defined as

$$r(k) = E\{x(n)x^*(n-k)\}$$

(3)

2.3.3 Autoregressive Model

An Autoregressive model (AR) model predicts future behaviour based on known past instances. It's used to predict the correlation between values in a time series and the values before and after them. The AR model, with the order p, is defined to be linearly related to a number of its previous samples and can be computed using equation 4 [3] i.e.

$$x(n) = -\sum_{k=1}^{p} a_k x(n-k) + y(n)$$

(4)

where x(n) is the data of the signal at the sampled point n, a_k, where y(n) is the noise input and k = 1, 2, . . ., p are the AR coefficients.

2.3.4 Entropy

Entropy measures the uncertainty in EEG signals, which equates to randomness or predictability. We can extract or characterise EEG signal features by observing the changes in entropy across the signal. However, there are many methods and parameter choices that can fundamentally change the result and meaning [16]. In summary, the higher the entropy, the more complex or chaotic the system is therefore it is less predictable [5].

There are several methods of calculating entropy which can significantly alter the results. These entropy methods can be grouped into 2 major categories [5]:

1. Spectral Entropies: These methods use the amplitudes of the signals' power spectrum as probabilities to find the entropy values. Spectral Entropy, Wavelet Entropy and Renyis Entropy are typical examples of this category [1].
2. Embedding entropies: In these methods, the time series is directly estimated using the entropy. Typical examples of this category are Shannon Entropy, Approximate Entropy, Sample Entropy, Conditional Entropy and KolmogorovSinai Entropy [1].

2.3.5 Shannon Entropy

In this thesis, Shannon entropy is used as a baseline for its simplicity. To compute the Shannon entropy, we define X as X = {x1, x2, . . ., xN}, xi ∈ Rd, Shannon entropy, H(X), is calculated as shown in equation 5 [17].

$$H(X) = -c \sum_{i=0}^{N} p(x_i) \ln p(x_i) \quad (5)$$

c is a positive constant used as a measuring unit and p(xi) is the probability of xi ∈ X as shown in equation 6.

$$\sum_{i=0}^{N} p(x_i) = 1 \quad (6)$$

2.4 Classifiers

2.4.1 Support Vector Machine

SVM is highly preferred and used in most systems as its accuracy to computational power is very high. It is mostly used for classification but can also be used in regression tasks as well [18].

The goal of the SVM algorithm is to identify a demarcation (hyperplane) in a dimensional space with N number of features that most distinctively separates the data. As shown in the right graph of figure 5, the optimal hyperplane is that which is right in the middle of the Maximum margin.

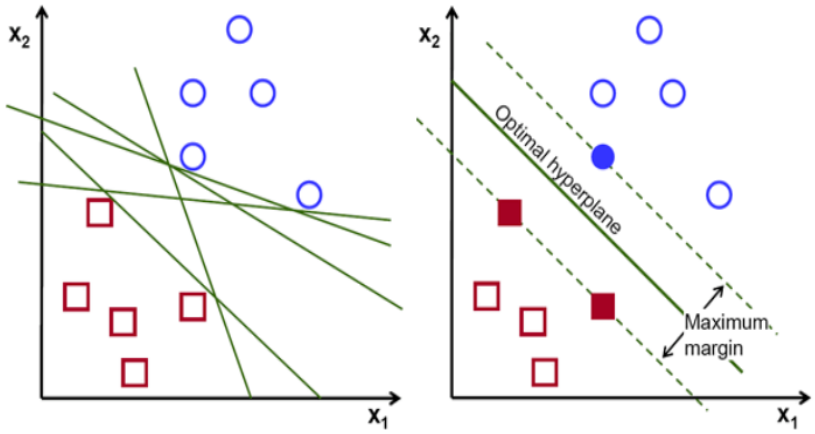

Figure 5: Possible hyperplanes [18]

As shown in the left graph of figure 5, to distinguish between the two classes, many possible hyperplanes could be chosen.

SVM algorithms use kernels which are mathematical functions that transform the input data into the required form. There are many different types of kernels however in this thesis we utilise the Gaussian radial basis function (RBF) shown in equation 7 [19].

$$K(X_1, X_2) = \exp(-\frac{\left||X_1 - X_2|\right|^2}{2\sigma^2})$$

(7)

Where σ is the variance and hyperparameter, and $||X_1 - X_2||$ is the Euclidean distance between X_1 and X_2 [20].

The goal of the SVM is to select the hyperplane with the most distance from the data points (optimal hyperplane). To help maximize the distance the hinge loss function shown in figure 6 is used.

$$min_w \lambda \left|\left|w\right|\right|^2 + \sum_{i=1}^{n} (1 - y_i(x_i, w))_+$$

(8)

We update our weights in figure 7 by using gradients obtained by taking partial derivatives w.r.t the weights as shown in figure 8.

$$\frac{\sigma}{\delta w_k} \lambda \, || \, w \, ||^2 = 2\lambda w_k$$

$$\frac{\sigma}{\delta w_k} (1 - y_i(x_i, w))_+ = \begin{cases} 0, & if \; y_i(x_i, w) \geq 1 \\ -y_i x_{ik}, & else \end{cases}$$

(9)

In the case of no misclassification, we only have to update the gradient in figure 7 from the regularization parameter shown in figure 9. This parameter dictates the amount of misclassification allowed in our SVM model.

$$w = w - \alpha \cdot (2\lambda w)$$

(10)

When there is a misclassification, i.e our model makes a mistake on the prediction of the class of our data point, we include the loss along with the regularization parameter to perform a gradient update as shown in figure 10.

$$w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda w)$$

(11)

2.4.2 Neural Network

A Neural network is a computing system composed of highly interconnected nodes which dynamically process information based on state response to external inputs.

As depicted in figure 6, the basic units of a neural network are the neurons, sometimes referred to as nodes. They receive inputs from other nodes, or an external source and compute an output. The inputs all have weights assigned to them based on their relative importance to each other. A function is then applied to the weighted sum of the inputs by the node. There are many types of neural networks however, this thesis uses a Convolutional Neural Network (CNN) which is a linear neural network categorised by the fact that they have no activation functions in any layers of the neural network.

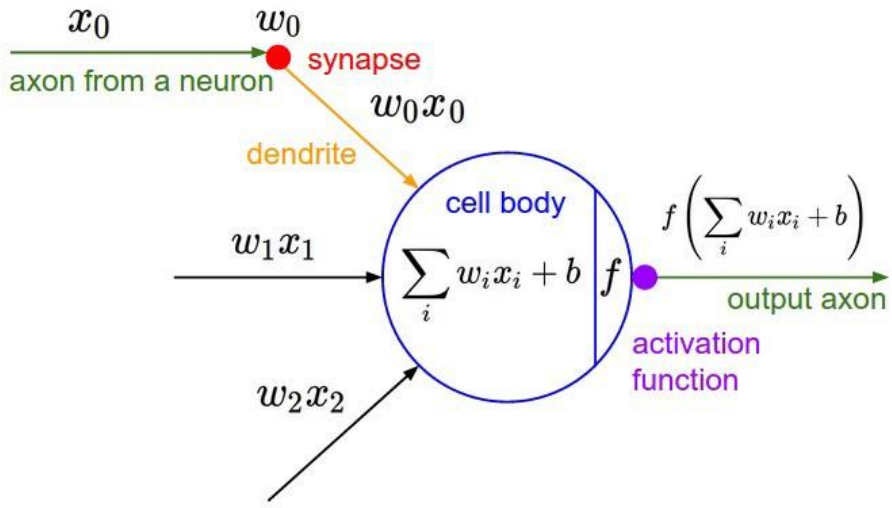

Figure 6: Mathematical model of a neuron [21]

They function by controlling the strength of the wights based on the learning rate to influence the neuron. The learning rate controls how fast the NN updates concepts it has learnt. To show the rate of spikes along the axon, the firing rate for each neuron is modified with activation functions like the sigmoid function [21].

2.4.3 K – Nearest Neighbour

It is a non-parametric supervised learning classifier that uses proximity to classify or predict the grouping of a data point. The K in KNN stands for the number of nearest neighbours considered in the grouping process.

For classification, classes are assigned based on a majority vote. In other words, the label around the most represented data point is used.

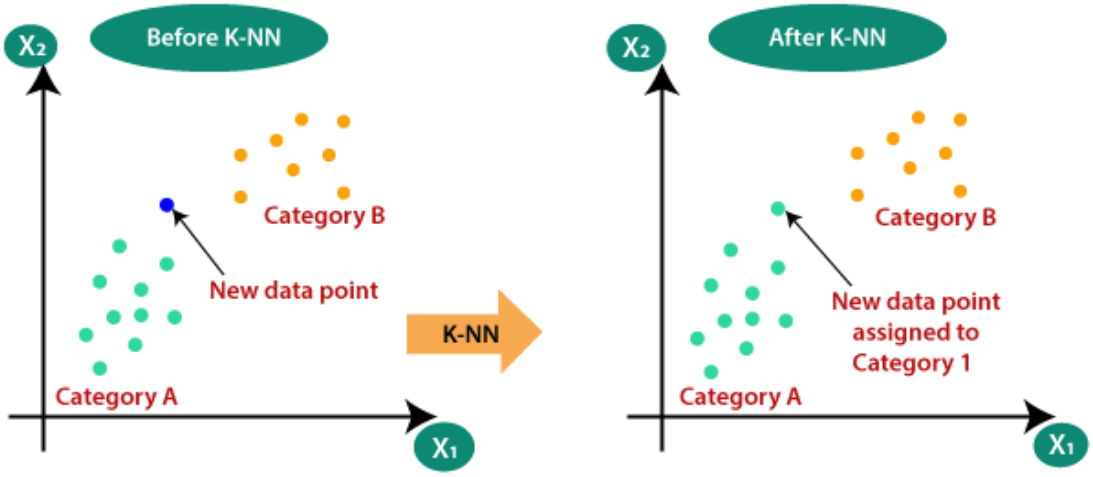

Figure 7: KNN diagram [22]

As shown in figure 7, the closest distance between the new data and query points is calculated. These distance metrics help to form decision boundaries, which separate query points into different regions. There are various distance measures but the most used is the Euclidean distance metric [7] shown below.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2} \quad (12)$$

2.4.4 Classifier Evaluation Methods

Classifiers are usually evaluated on 3 major methods namely [23];

1. Accuracy: The calculated success rate of the classifier with the model used. These accuracies can be calculated in multiple ways i.e., precision, recall, F1 and more.

   - $Precision = \frac{TP\ (True\ Positive)}{TP+FP\ (False\ Positive)}$
   - $Recall = \frac{TP}{TP+FN\ (False\ Negative)}$
   - $F1 = \frac{2 \times precision \times recall}{precision+recall}$

(13)

In the above equations, our positive state is assigned as the Lucid state and negative state is the Deep state. Therefore, TP represents the instance where our model correctly predicts the Lucid state and FP is when our model predicts the Lucid state incorrectly and vice-versa for the TN and FN.

2. Computational speed: The time taken for the model to arrive at a result.
3. Graphical representation of performance: These are methods applied to a graphical representation of a model to compare it with other similar systems an example of which is receiver operating characteristic curve (ROC).

The main evaluation used in this thesis will be precision accuracy but other methods such as computational speed will also influence the final results.

2.5 Existing Research

While still juvenile, a substantial amount of research has been conducted on EEG BCIs as they are a door to the study and cure for many mental-related diseases and have a wide range of applications from monitoring to control.

Most recently, research is being conducted into applications such as entropy feature extraction of EEG signals for automatic person identification [1], a comparative analysis of different classifiers on EEG signals for predicting epileptic seizure

[24], quantitative evaluation of EEG-biomarkers for prediction of sleep stages [25], automatic human sleep stage scoring using deep neural networks [26] and much more.

2.5.1 Entropy Feature Extraction of EEG Signals for Automatic Person Identification

In this thesis, 5 entropy feature extraction methods were tested and compared to identify which resulted in the best performance in automatic person identification. The Autoregression model was used as a baseline to compare with the other entropy feature extraction methods while an SVM classifier was used [1].

The results of their experiments proved entropy to be adequate as a feature extraction method for a multi-channel EEG based person identification system. With greater than 85% identification rates they showed better performances than the compared AR methods for most of the entropy extraction methods used. For the speed comparisons, in model building, some entropy methods were faster than corresponding AR methods however most of them were slower at computing times greater than 2000s. However, most entropy methods showed better results during model testing with times slower than AR methods [1].

2.5.2 Comparative Analysis of Different Classifiers on EEG Signals for Predicting Epileptic Seizures

In this study, 6 different classifiers were tested to ascertain which resulted in the best accuracy for epileptic seizure prediction [24]. Dataset was sourced from the Kaggle website and ten folds of the datasets were made in MATLAB and cross-validated.

Ten accuracies and average accuracies were obtained for each classifier with the average for all of them being 81% excluding the linear classification model with 58.339%. The best classifier was the Naïve Bayes classifier as it was found to have a

better average accuracy compared to the other classifiers used at an average of 95.739% with KNN as a close second at 95.165%.

2.5.3 Quantitative Evaluation of EEG-Biomarkers for Prediction of Sleep Stages

This study aimed to quantify EEG biomarkers and predict 5 class sleep stages using sleep EEG data [25].

The dataset used was the Haaglanden Medisch Centrum open access public dataset on the Physio Net website. Features were extracted using Fast Fourier transform (FFT), PSD and other methods.

Three different machine learning models were tested and compared with previous studies namely, C5.0, Neural network and CHAID for prediction of sleep stages. The results obtained were much better than other works, with accuracies of 91%, 95% and 84% for C5.0, NN and CHAID methods respectively.

2.6 Summary

There are also researches like Automatic human sleep stage scoring using deep neural networks [26], Sleep stage classification from single channel EEG using CNN [27] and many more. Most of the current research are conducted on multi-channel data to increase the accuracy and size of data as they are geared toward advanced applications.

For the topic of sleep stage prediction, entropy has not been used with single channel data for sleep stage prediction which has resulted in the main objective of this thesis.

Chapter 3

EXPERIMENTAL SETUPS

In this section, we will describe in detail the datasets used for the experiments, the pre-processing steps taken, the entropy feature extraction steps and finally the parameters used in the classifiers.

3.1 EEG Datasets

The dataset used in this research was the Sleep-EDF [26] database. This database contains EEG recordings of Caucasian males and females aged 25 – 101 years old without any kind of medication. The database contains 197 whole-night polysomnographic sleep recordings containing 2 EEG channels namely Fpz – Cz and Pz – Oz, 1 Electro-oculography (EOG) channel and 1 Electromyography (EMG) channel respiration and body temperature were also included in some recordings. The database has been pre-processed for noise reduction and scored by professionals. The database consists of 2 studies, Sleep cassette (SC) shown in figure 8 which contains recordings of 82 healthy Caucasians each with 2-night recordings and Sleep Telemetry (ST) shown in figure 9 which contains recordings of 22 Caucasians with mild difficulties falling asleep to see the effects of temazepam. Only the EEG channels are used in this thesis. EEG recordings were scored in sleep stages according to the R&K rule to obtain the hypnograms. The data used in this thesis was 20 recordings from 10 subjects from both SC and ST studies. We divided the data into 10s epochs; with 1s overlaps therefore, the number of points for each epoch is 1000 due to the 100 Hz sampling frequency. We labelled each epoch as 10s, 19s, 28s, 37s, 46s, 55s etc.

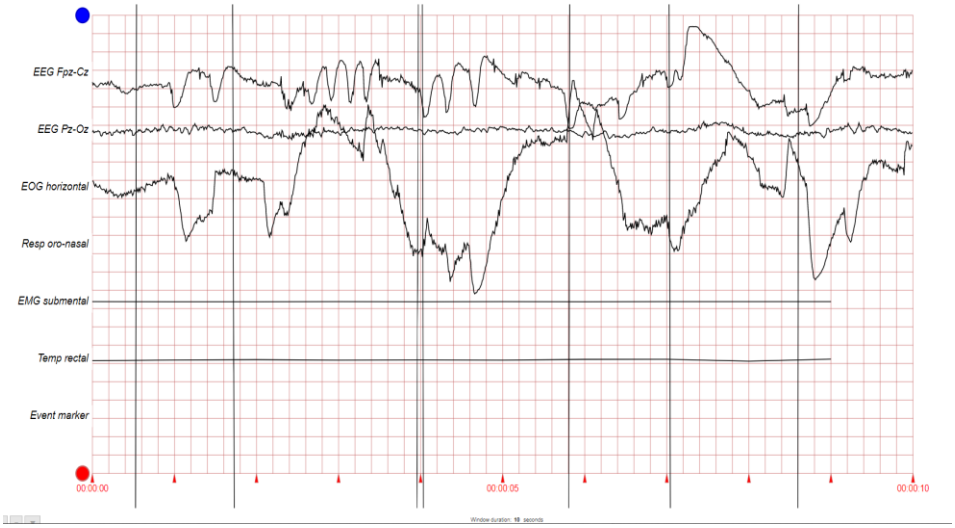

Figure 8: Image of SC 10s epoch recording [26]

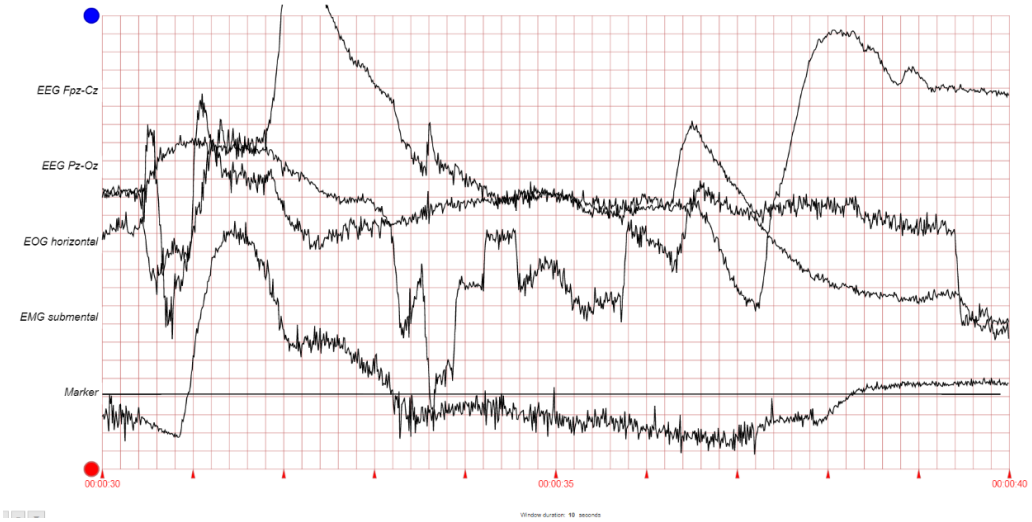

Figure 9: Image of ST 10s epoch recording [26]

3.2 EEG Pre-processing

Initially, tools like EEGLAB [33] on MATLAB were used for the computation but they proved to be too slow and it was difficult to segment the data. Thus, the computation was done in python due to it allowing for optimised selection of data being used thus algorithms can be tested on individual data before being applied to the entire project. The computer used contained a 16GB RAM with a 7th generation I7 intel core.

Figure 10 below shows the steps taken during pre-processing to allow us to easily extract the required features in python which was the main programming language used for this thesis

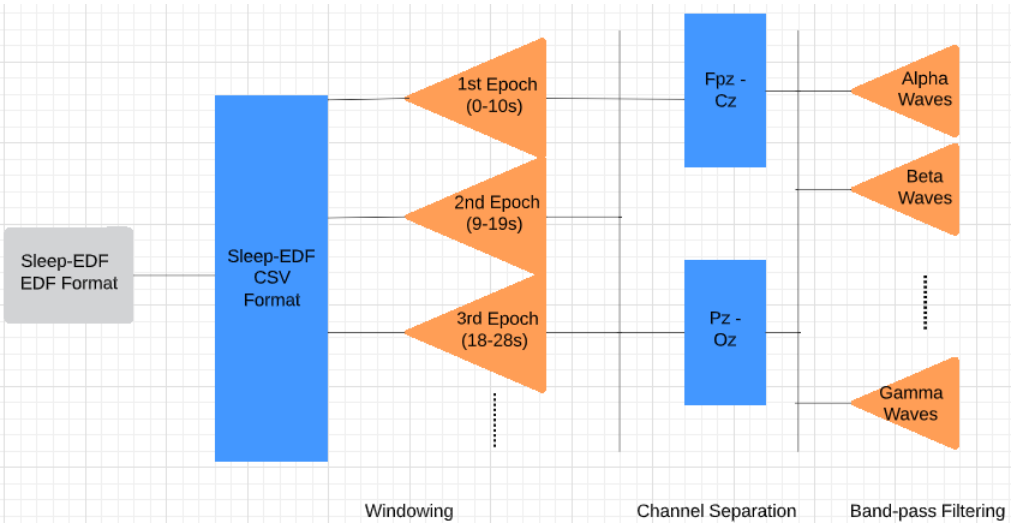

Figure 10: Flowchart of pre-processing steps

Initially, we loaded the data into MATLAB and applied a few noise reduction techniques including filtering but very little to no changes were noticed in the data due to it already being processed for noise reduction. Therefore, this step was skipped for the rest of the data used.

To begin pre-processing, the database was converted to a .csv format to make it easier to visualise and manipulate in python. However, due to the recordings being too large, the later parts of the recordings are lost when converted to a .csv format therefore only the first 349 seconds of each recording were used in this thesis.

The next step was to divide each recording into time windows of 10 s with 1 s overlaps to allow for easier processing time and more accurate visualisation of the data. Smaller time windows were tested however it was observed that when going lower than 10 s there is not enough variation in the Shannon entropy calculated for proper classification therefore 10 s were selected as the time windows.

Following the time window segmentation, all other channels except for the EEG channels were removed from the data leaving us with only the Fpz – Cz and Pz – Oz channels.

Finally, each window of each channel for each subject was put through band-pass filters to obtain the alpha, beta, gamma, delta and theta waves.

3.3 Feature Extraction

In this step, the entropy is calculated for all pre-processed waves for each subject as shown in figure 11.

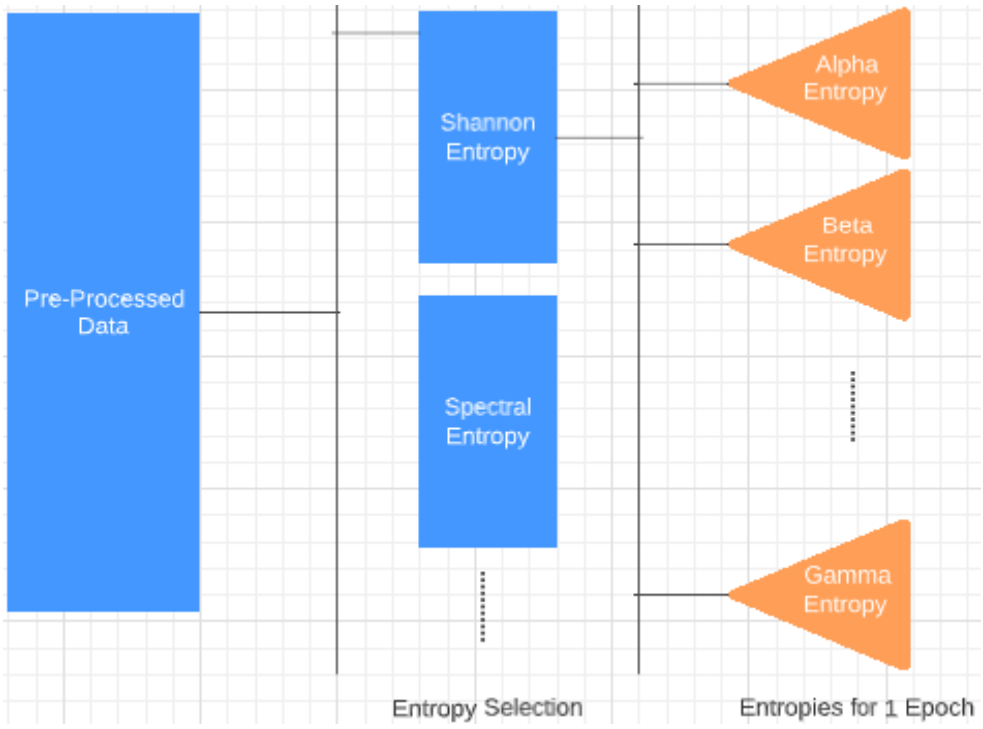

Figure 11: Flowchart of feature extraction steps

In this step, as shown in figure 11, we extract entropy features for each pre-processed wave to form a database that can be fed into the classifiers for classification and prediction.

In this thesis, only Shannon entropy was used as it is a good baseline for computing entropy but part of future testing will be trying out different entropy computation methods to identify which gives the best results.

3.4 Classifiers

3.4.1 Classification

All the extracted data were compiled into a spreadsheet containing the extracted entropies for each frequency band for each 10s-time epoch as shown in figure 12.

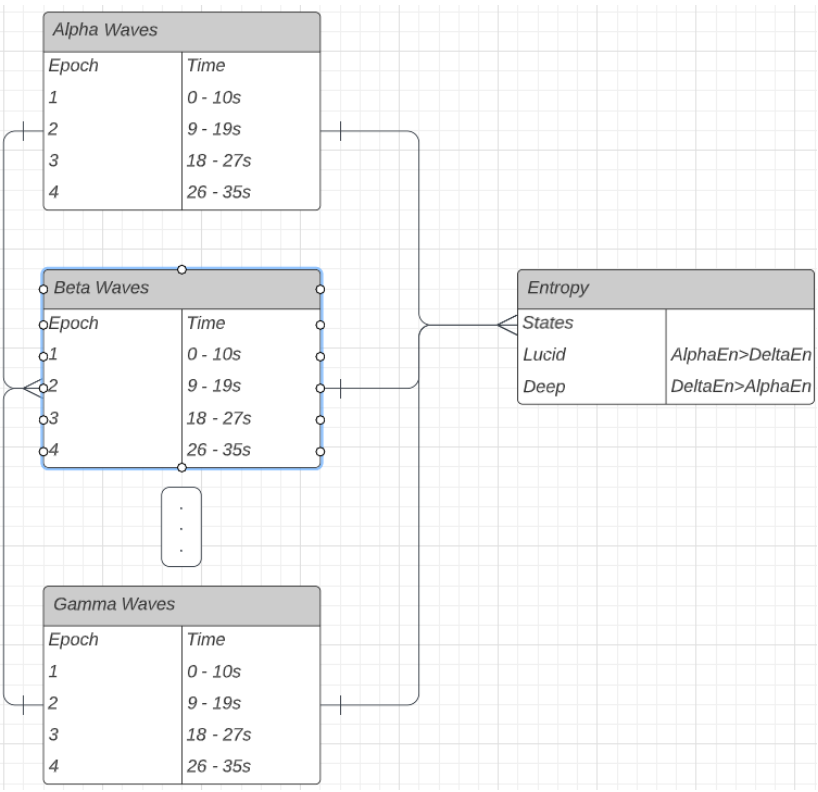

Figure 12: CNN classification flowchart

As shown in figure 19, we use a linear CNN due to the data being linear. The network uses 5 inputs namely, alpha, beta, gamma, delta and theta wave entropies to classify 2 outputs lucid and sleep states.

The network consists of 1 hidden layer with 64 neurones all with randomly generated weights and biases. The network does not have any activation function except at the output where a SoftMax function is applied to get the classes. A one-hot encoding has been applied to the data as a part of the standard data transformation for a classification task.

Since this is a linear network, the most suitable loss function to use is mean square error (MSE) loss which helps measure an accurate loss/error which then can be used to optimize the network using the stochastic gradient descent (SGD) optimizer which will update the values of the weights and biases to minimize the error.

The network has been trained for 500 epochs with a data size of 200 (40 by 5) with 80/20 split between training and testing. A learning rate of 0.001 is added to avoid any error spikes.

Two unique classes were identified and defined to be predicted by our classifiers based on the entropy characteristics described in [30]:

1. Lucid sleep state: this state was defined as when the extracted alpha entropy was greater than the delta entropy signifying that there was more “information” in the alpha frequency band.
2. Deep sleep state: this state was defined as when the extracted alpha entropy was less than the delta entropy signifying that there was more “information” in the delta frequency band.

3.4.2 Prediction

After the states were classified, the classified data is then fed into our prediction classifiers;

- SVM: A rbf Kernel SVM was used due to its flexibility and the size of our feature space (5-features). The SVM splits the input data into 2 with 80% used

as training data and 20% used for testing. After the training is complete, the predictions are made and then compared with the test data to determine their accuracies.

- KNN: The input data was also split into test and train data for the KNN however, different splits were tested and drastic results were observed when the train data was increased or reduced. However, for the sake of consistency in the comparisons, we settled at the same train-test split as the SVM at an 80 – 20 train-test split. The KNN used was tested with different amounts of k (nearest neighbours) to acquire which number of k resulted in an acceptable error rate.

3.4.3 Classifier Evaluation

All used classifiers are evaluated and compared mainly on their prediction accuracies as the goal of this thesis is to attain adequate accuracy with single channel data but other parameters will also play a role in the final decision.

All prediction accuracies were finalised using 5–fold cross-validation to obtain the most accurate results. We achieved this by the following steps:

1. The dataset was randomly shuffled
2. Then the dataset was split into 5 groups
3. Each group was then used as a test set while the other groups were used to train the model and their accuracies recorded
4. Finally, we compared and averaged all the acquired accuracies

This method was used to reduce the amount of bias in the model results as every part of the data is used for both testing and training [31]. In this method, the results are averaged preventing misinformation (higher or lesser accuracies) due to any

given input-output split. It also helps greatly stabilise the results of some classifiers i.e., KNN.

3.5 Summary

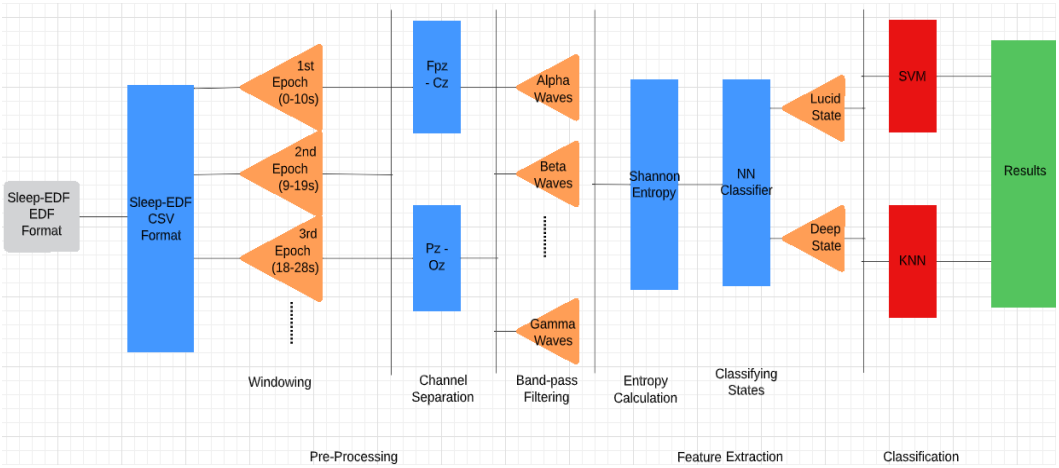

Figure 13: Flowchart of experimental setups

To summarise, as shown in figure 14, our model used the Sleep-EDF database which is pre-processed, the Shannon entropy is then computed and the results are fed into NN to assign classes (Lucid or Sleep). Finally, the data is fed into our two classifiers used for prediction and the results are compared.

Chapter 4

RESULTS

In this chapter, we will explain our modelling, training and testing methods as well as the results of our experiments.

4.1 Pre-processing Results

4.1.1 Channel Separation

Figures 14 and 15 below show the results obtained after channel separation.

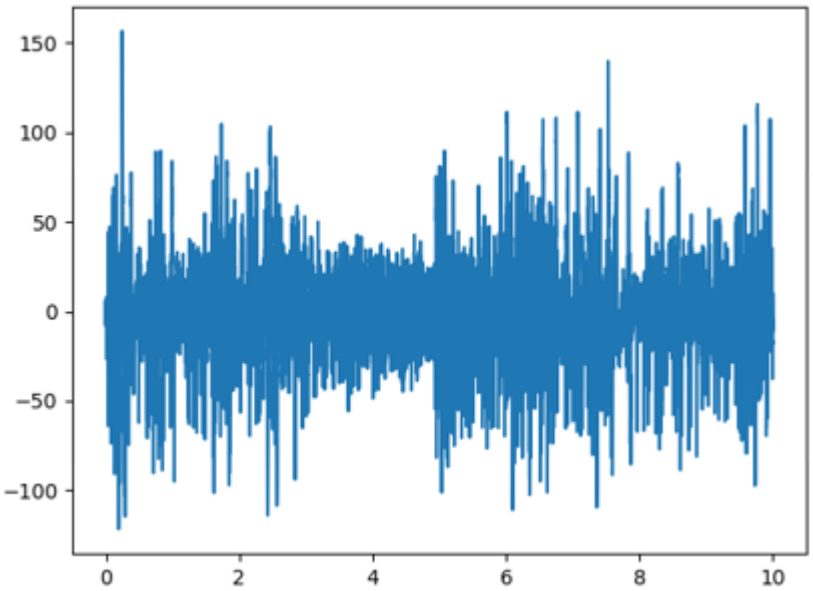

Figure 14: Voltage in mV (y-axis) over time in seconds (x-axis) plot of 1st epoch of Fpz-Cz channel

Figure 14 above shows the first epoch of the first recording for only the Fpz-Cz channel.

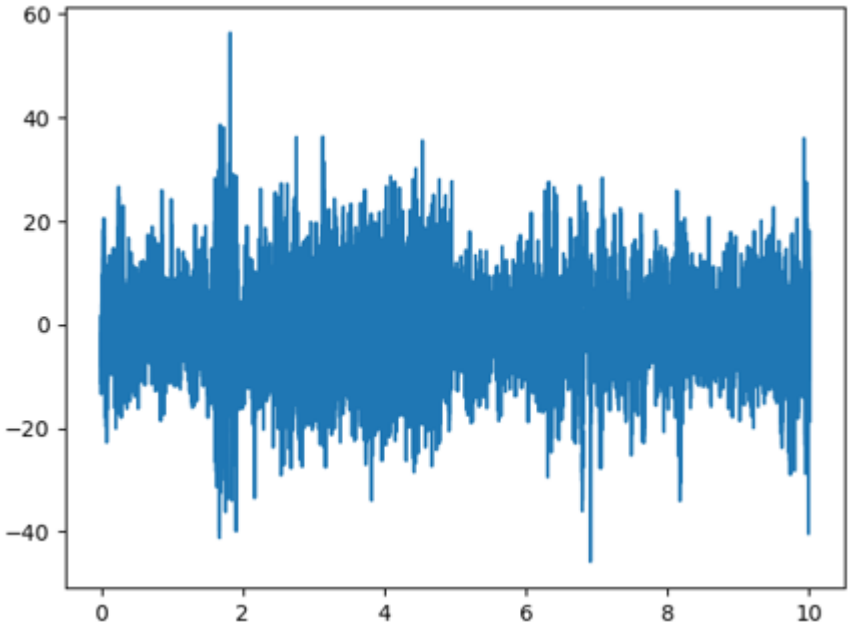

Figure 15: Voltage in mV (y-axis) over time in seconds (x-axis) plot of 1st epoch of Pz-Oz channel

Figure 15 above shows the first epoch of the first recording for only the Pz-Oz channel.

4.1.2 Frequency Filtering

Figures 16, 17 and 18 below show the results obtained after putting the channel-separated data through bandpass filters to obtain the different brain waves.

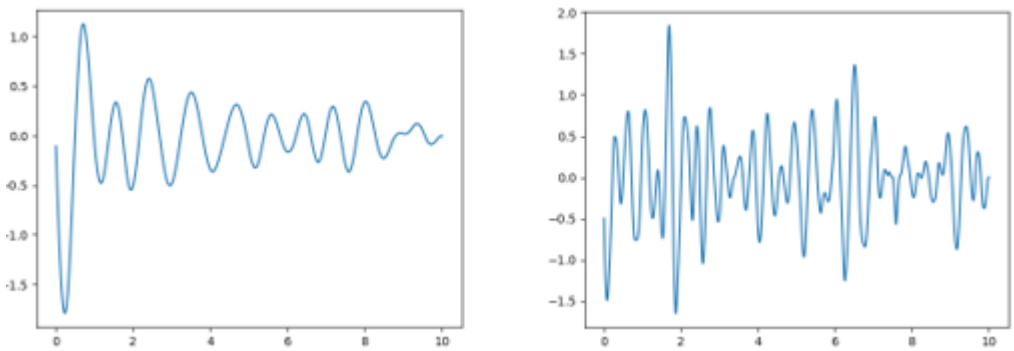
Figure 16: Voltage in mV (y-axis) over time in seconds (x-axis) plot of Alpha (left) and beta (right) responses of Fpz1_10s

Figure 16 above, shows the alpha and beta waves obtained after the first epoch of the first recording was put through an 8 – 13 Hz and 14 – 30 Hz band-pass filters respectively.

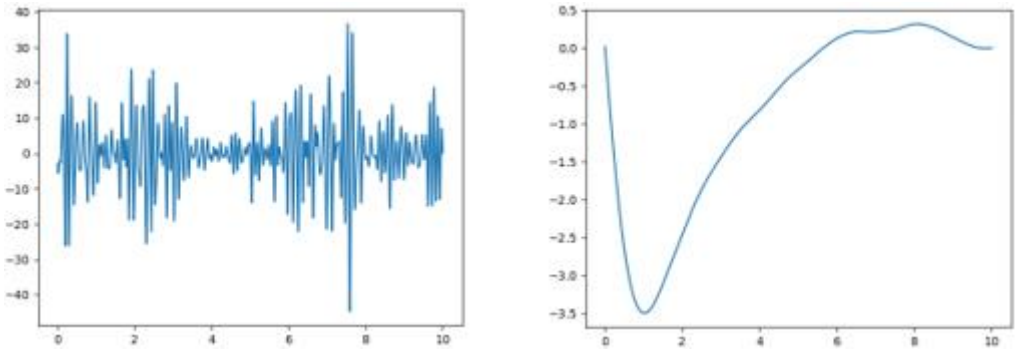

Figure 17: Voltage in mV (y-axis) over time in seconds (x-axis) plot of Gamma (left) and delta (right) responses of Fpz1_10s

Figure 16 above, shows the gamma and delta waves obtained after the first epoch of the first recording was put through a 30 – 100 Hz and 0.5 – 3 Hz band-pass filters respectively.

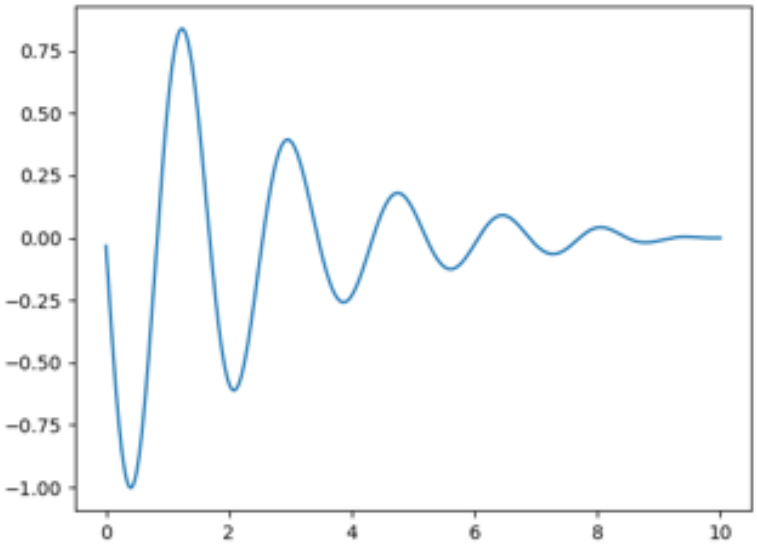

Figure 18: Theta response of Fpz1_10s

Figure 18 above, shows the theta waves obtained after the first epoch of the first recording was put through a 4 – 7 Hz band-pass filter.

4.2 Feature Extraction Results

Table 2 below shows the results of the feature extraction step using Shannon entropy.

Table 2: Output of first 2 epochs for 5 recordings of feature extraction

Channel	Epoch	Recordings	Alpha Entropy	Beta Entropy	Gamma Entropy	Delta Entropy	Theta Entropy
Fpz-Cz	1	1	14.834	14.854	14.871	14.828	14.781
Fpz-Cz	2	1	14.815	14.853	14.871	14.840	14.800
Fpz-Cz	1	2	14.797	14.863	14.872	14.868	14.854
Fpz-Cz	2	2	14.846	14.853	14.872	14.866	14.835
Fpz-Cz	1	3	14.859	14.866	14.871	14.869	14.858
Fpz-Cz	2	3	14.852	14.867	14.872	14.763	14.700
Fpz-Cz	1	4	14.841	14.859	14.871	14.817	14.835
Fpz-Cz	2	4	14.849	14.865	14.872	14.870	14.799
Fpz-Cz	1	5	14.846	14.855	14.872	14.863	14.743
Fpz-Cz	2	5	14.848	14.851	14.871	14.866	14.836

In table 2, the full database contains 40 entries with 2 entries for each recording. It contains 5 features after the feature extraction which can be used in the classifiers although not all of the extracted entropies are as useful for instance it can be observed that there is very little variation in the gamma entropies which is expected due to the data used being sleep data.

4.3 Classification Results

4.3.1 NN Classification Results

Table 3 below shows the outputs of the classification step using a linear NN classifier.

Table 3: Output of NN classifier and input of SVM and KNN for prediction

Channel	Epoch	Recordings	Alpha Entropy	Beta Entropy	Gamma Entropy	Delta Entropy	Theta Entropy	Class
Fpz-Cz	1	1	14.834	14.854	14.871	14.828	14.781	Lucid
Fpz-Cz	2	1	14.815	14.853	14.871	14.840	14.800	Deep
Fpz-Cz	1	2	14.797	14.863	14.872	14.868	14.854	Deep
Fpz-Cz	2	2	14.846	14.853	14.872	14.866	14.835	Deep
Fpz-Cz	1	3	14.859	14.866	14.871	14.869	14.858	Deep
Fpz-Cz	2	3	14.852	14.867	14.872	14.763	14.700	Lucid
Fpz-Cz	1	4	14.841	14.859	14.871	14.817	14.835	Lucid
Fpz-Cz	2	4	14.849	14.865	14.872	14.870	14.799	Deep
Fpz-Cz	1	5	14.846	14.855	14.872	14.863	14.743	Deep
Fpz-Cz	2	5	14.848	14.851	14.871	14.866	14.836	Deep

In table 3, the identified states of lucid and deep correspond directly with the alpha and delta values with the lucid state being assigned to states where the alpha entropy is greater than that of delta and the deep state being assigned otherwise. The fully classified data resulted in 30% (12 entries) being lucid and 70% (28 entries) classified as the deep state.

4.3.2 SVM Prediction Results

We were able to achieve a maximum prediction accuracy of 87.5% with an average of 67.5% after 5-fold cross-validation with 40 inputs. It is expected to increase to 80% with double the inputs.

4.3.3 KNN Prediction Results

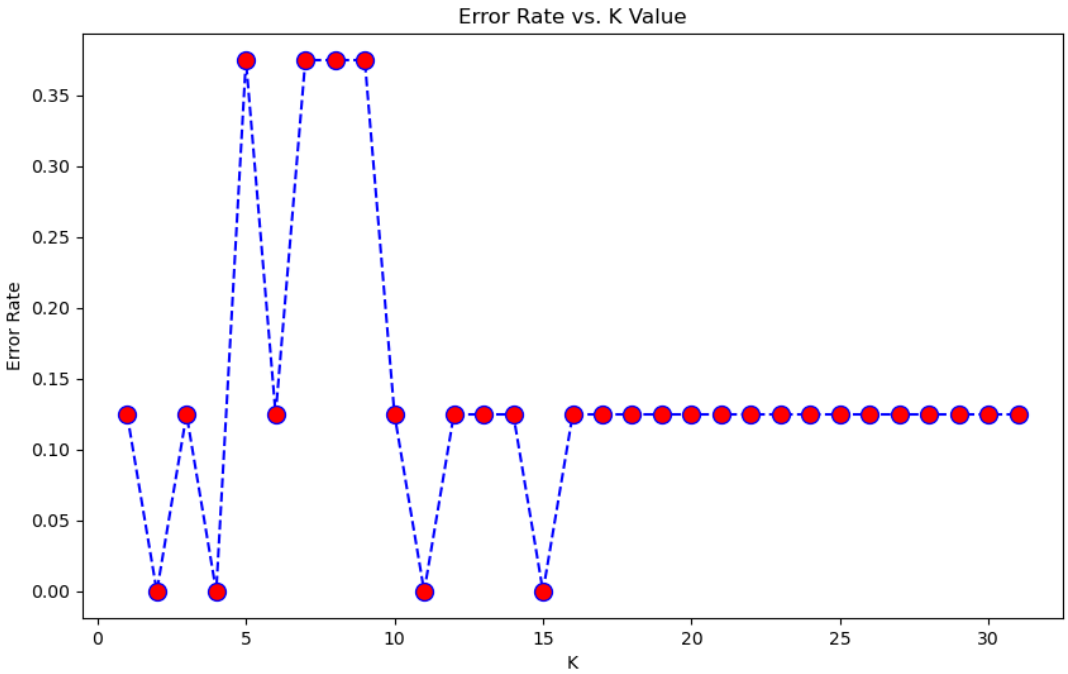

Figure 19: Error rate with different values of K (nearest neighbours)

The KNN used was tested with different amounts of k (nearest neighbours) as shown in figure 27 we achieved a balanced error rate of 0.125 after 15 iterations. We were able to achieve a maximum prediction accuracy of 100% with an average of 81.25% after 5-fold cross-validation.

4.3.4 NN Prediction

While prediction is possible it proved to be more challenging to implement thus it was omitted but in theory, should result in higher accuracies than the SVM and KNN methods with accuracies higher than 87% according to other research [32].

4.4 Discussions

4.4.1 Classifier Accuracies

The confusion matrices for our prediction classifiers can be observed in tables 4 and 5 below. It can be observed that, due to the large difference between the deep and lucid inputs in the test data (approx. 24 deep and 8 lucid), the predictions made by the classifiers were always Deep. In order to ascertain the correctness of our models, we increased the input data by adding the extracted data from the Pz – Oz channel (i.e., doubling the number of inputs). Following which the model made some predictions as Lucid. This was also how we hypothesized an estimate for the increase in accuracy for doubling the inputs.

Table 4: SVM confusion matrix

Cross-validation Folds											
		1st Fold		2nd Fold		3rd Fold		4th Fold		5th Fold	
True State	Deep	6	0	5	0	7	0	4	0	5	0
	Lucid	2	0	3	0	1	0	4	0	3	0
		Deep	Lucid	Deep	Lucid	Deep	Lucid	Deep	Lucid	Deep	Lucid
Predicted State											

Table 5: KNN confusion matrix

Cross-validation Folds											
		1st Fold		2nd Fold		3rd Fold		4th Fold		5th Fold	
True State	Deep	5	0	6	0	6	0	7	0	4	0
	Lucid	3	0	2	0	2	0	1	0	4	0
		Deep	Lucid	Deep	Lucid	Deep	Lucid	Deep	Lucid	Deep	Lucid
Predicted State											

Table 6: Prediction performance comparisons between classifiers

In-puts	Method	Average Entropy					Avg Class	Accuracy	
		Alpha	Beta	Gamma	Delta	Theta		Max	Avg
40	SVM	14.834	14.854	14.871	14.828	14.781	Lucid	87.5%	67.5%
40	KNN	14.815	14.853	14.871	14.840	14.800	Deep	100%	81.25%

Table 6 above shows the performance comparisons between the different classifiers regarding their accuracies. It shows the best accuracies achieved using each method.

4.5 Summary

To summarise KNN seems to show better prediction accuracy compared to SVM at low input levels. However, it is much slower in practice with wide variation in the prediction accuracies at this number of inputs. It is also difficult to automate as any changes to the input data changes the amount of K required for maximum accuracy.

As for the NN prediction accuracy, if implemented should result in the best accuracies and consistency compared to the other 2 methods with accuracies greater than 87% [33].

Increasing the channels will provide even more inputs being fed into the classifiers resulting in an overall increase in prediction accuracies.

4.5.1 Methods for Accuracy Improvements

There are a few ways in which the system could be improved namely:

- Increasing input channels
- Using other entropy feature extraction methods.
- Combining the different classification methods to form an Ensemble Classifier

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

The main objectives for this thesis have been achieved however, the results can be greatly improved upon by solving some limitations faced during this project. Below are the objectives completed and limitations faced during this project:

5.1.1 Accurately Predicting Sleep Stages with a Single EEG Channel Using Entropy Feature Extraction

We have been able to achieve average prediction accuracies between 70% and 90% which falls in line with other EEG sleep stage classification and prediction research but with significantly less input due to only using a single BCI channel [34].

5.1.2 Identifying the Performance of Different Classifiers for Sleep Stage Prediction

In terms of maximum accuracy, KNN showed the best results however its computational time was many times that of the SVM with the same number of inputs and due to the unstable nature of KNN any change in inputs causes significant changes to the accuracy and an optimal iteration has to be calculated again. On the other hand, SVM shows decent accuracy and is expected to result in better accuracies than KNN given enough inputs [32]. Thus, for this research's application, SVM will be the preferred choice.

5.1.3 Computational and Time Limitations

Due to the massive nature of EEG data, processing and experimenting on it requires a lot of computational power. This severely restricted the scope of the experiments as we were unable to fully utilise our datasets in their entirety, thus much better results could be acquired with more computational power. However, this limitation can be easily solved with better equipment.

For the same reasons mentioned above, with more time more data could be fed to the algorithm which would significantly increase the performance of our best classifier SVM and potentially increase that on KNN at the cost of increased computational time. It would also allow us to implement a CNN classifier which as seen from previous research has the best accuracies out of the 3 machine learning algorithms.

5.2 Future Work

Achieving entropy feature extraction for EEG sleep stage prediction is just the beginning and needs to be furthered upon and polished. Future work will focus on the following:

5.2.1 Identifying and Comparing the Best Entropy Feature Extraction Methods for EEG Sleep Stage Prediction

This is a priority in the continuation of this project as Shannon entropy was only used as a baseline and the results are expected to be significantly better if other entropy calculation methods are used [1].

5.2.2 Developing Better Classifier Algorithms and Implementation of a Prediction Algorithm using NN

The aim is to find the classifier algorithm that provides the best accuracy with the least number of inputs, with a significant increase in inputs to the classifiers so will

the importance of faster computational speeds which will be even more apparent in real-time implementations.

5.2.3 Extending Experiments to Non-Entropy Methods for Feature Extraction

Another avenue to be explored will be non-entropy feature extraction methods which could result in better accuracies than the best entropy extraction method and various extracted features like Autoregression, Fourier transform, Power spectral density etc.,

5.2.4 Building and Testing of Algorithm in a Non-Simulated Environment

The final step for this project would be to create a working prototype with similar accuracies to the simulations and while they will certainly be more complications due to the most likely more chaotic nature of the input data more pre-processing steps would most likely have to be added to achieve accuracies similar to the simulations.

REFERENCES

[1] D. V. Phung, "Entropy feature extraction of EEG signals for automatic person identification," 2016.

[2] T. Al-ani and D. Trad, "Signal processing and classification approaches for brain-computer interface," 2010.

[3] S. Sanei and J. A. Chambers, "EEG signal processing," John Wiley & Sons Ltd, 2007.

[4] B. I. Morshed and A. Khan, "A brief review of brain signal monitoring technologies for BCI applications: challenges and prospects," *Bioengineering & Biomedical Science,* 2014.

[5] N. Kannathal, U. R. Acharya, C. M. Lim and P. K. Sadasivan, "Characterization of eeg a comparative study," *Computer methods and Programs in Biomedicine,* 2005.

[6] aswathisasidharan, "Support vector machine algorithm," *GeeksforGeeks,* 2021.

[7] "IBM analytics," [Online]. Available: https://www.ibm.com/topics/knn#:~:text=The%20k-

nearest%20neighbors%20algorithm%2C%20also%20known%20as%20KNN%20or,of%20an%20individual%20data%20point.

[8] E. Suni, "Stages of sleep," 2022.

[9] J. B. Ochoa, "EEG signal classification for brain computer interface applications," 2002.

[10] W. Yang, J. Yang, Y. Gao, X. Tang, R. Yanna, S. Takahashi and J. Wu, "Effects of sound frequency on audiovisual integration: an Event-related potential study".

[11] P. A. Abhang, B. Gawali and S. C. Mehrotra, "Introduction to EEG- and speech-based emotion recognition," 2016.

[12] J. Moini and P. Piran, "Functional and clinical Neuroanatomy," 2020.

[13] D.-C. Li, C.-W. Liu and S. C. Hu, "A fuzzy-based data transformation for feature extraction to increase classification performance with small medical data sets," *Artificial Intelligence in Medicine,* 2011.

[14] M. Poulos, M. Rangoussi, N. Alexandris and A. Evangelou, "On the use of EEG features towards person identification via neural networks," 2001.

[15] P. Stoica and R. Moses, "Spectral analysis of signals," 2005.

[16] N. P. Subramaniyam, "Measuring entropy in the EEG," *Sapien labs*, 2018.

[17] C. Shannon, "The mathematical theory of communication," *The Bell System Technical Journal*, 1948.

[18] R. Gandhi, "Support vector machine — introduction to machine learning algorithms," *Towards data science*, 2018.

[19] "Support vector machine (SVM) in Python," DatabaseTown, [Online]. Available: https://databasetown.com/implementing-support-vector-machine-svm-in-python/. [Accessed 27 09 2022].

[20] S. Screenivasa, "Radial basis function (RBF) Kernel: the go-to Kernel," *Towards Data Science*, 2020.

[21] D. Fumo, "A gentle introduction to Neural networks series — part 1," *Towards data science*, 2017.

[22] "Machine learning JavaTpoint," javaTpoint, [Online]. Available: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning.

[23] J. Lever, M. Krzywinski and N. Altman, "Classification evaluation," *Nature methods,* 2016.

[24] M. K. Sharma, K. Ray, P. Yupapin, M. S. Kaiser, C. T. Ong and J. Ali, "Comparative analysis of different classifiers on EEG signals for predicting epileptic seizure," in *Advances in Intelligent Systems and Computing*, 2020.

[25] I. Hussain, M. A. Hossain, R. Jany, M. Abdul Bari, M. Uddin, M. A. R. Kamal, Y. Ku and . J.-S. Kim, "Quantitative evaluation of EEG-biomarkers for prediction of sleep stages," *Sensors,* 2022.

[26] A. Malafeev, D. Laptev, S. Bauer, X. Omlin, A. Wierzbicka, A. Wichniak, W. Jernajczyk, R. Riener, J. Buhmann and P. Achermann, "Automatic human sleep stage scoring using deep Neural networks," *Frontiers Sec. Sleep and Circadian Rhythms,* 2018.

[27] Y. Mansar, "Sleep stage classification from single channel EEG using convolutional Neural networks," *Towards Data Science,* 2018.

[28] B. Kemp, A. H. Zwinderman, B. Tuk, H. A. C. Kamphuisen and J. L. Oberye, "Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the EEG," *IEEE Transactions on Biomedical Engineering*, 2000.

[29] A. Delorme and S. Makeig, "EEGLAB: an open-source toolbox for analysis of single-trial EEG dynamics," *Journal of Neuroscience Methods,* 2004.

[30] S. Vajepeyam, "Understanding Shannon's entropy metric for information," 2014.

[31] M. Sanjay, "Why and how to Cross Validate a Model?," *Towards Data Science,* 2018.

[32] Q. Ma, M. Wang, L. Hu, L. Zhang and Z. Hua, "A novel recurrent neural network to classify EEG signals for customers' decision-making behavior prediction in brand extension scenario," *Frontiers Sec. Cognitive Neuroscience,* 2021.

[33] M. S. Aldayel, M. Ykhlef and A. N. Al-Nafjan, "Electroencephalogram-based preference prediction using deep transfer learning," 2020.

[34] K. A. I. Aboalayon, M. Faezipour, W. S. Almuhammadi and S. Moslehpour, "Sleep stage classification using EEG signal analysis: A comprehensive survey and new investigation," *Entropy and Electroencephalography II,* 2016.

[35] D. Bzdok, M. Krzywinski and N. Altman, "Machine learning: Supervised methods, SVM and kNN," *HAL,* 2018.

APPENDIX

Shannon Entropy Calculation Code

#Calculates Shannon Entropy of csv files

```
"""
csv_entropy.py
Author: MBJ (2017)
Purpose: Command line utility to assess the most "Informative" columns in a CSV file.
Detail:
  * Reads the contents of a CSV file
     - Assumes header row is at the top
     - Data is in rows 2 onwards
  * For each column calculates the shannon entropy of the row
  * Writes input CSV to output CSV (to stdout) but Row N+2 indicates the entropy
"""

import csv
import math
import sys
from collections import defaultdict


def usage():
    """ Print usage and exit """
    print("Usage: csv_entropy.py yourfile.csv results.csv")
    sys.exit()
```

```
def entropy(value_counts):
    """ Input is a db of value counts e.g. {True: 10, False:100, NULL: 5} """
    # Compute the shannon entropy of a column
    size = sum(value_counts.values())
    h_entropy = 0.0
    for _, count in value_counts.items():
        proportion = (count/size)
        h_entropy -= proportion * math.log(proportion, 2)
    return h_entropy


def analyse(infpath, outfpath):
    """ Analyse input file, write output file """
    header = None
    track = defaultdict(lambda: defaultdict(int))
    output = []


    # Read input
    with open(infpath, 'r') as inf:
        csvrows = csv.reader(inf)
        for row in csvrows:
            if not header:
                header = row
            else:
                for colix, value in enumerate(row):
                    track[colix][value] += 1
```

```
            output.append(row)


    entropies = {ix: entropy(db) for (ix, db) in track.items()}


    # Write output
    with open(outfpath, 'w') as ouf:
        csvwriter = csv.writer(ouf, lineterminator='\n')
        for row in output:
            csvwriter.writerow(row)
        csvwriter.writerow([])
        entropy_row = [entropies[colix] for colix in sorted(entropies.keys())]
        csvwriter.writerow(entropy_row)


if __name__ == '__main__':
    if len(sys.argv) != 3:
        usage()
    input_file, output_file = sys.argv[1], sys.argv[2]
    print("Analysing {} and writing {}".format(input_file, output_file))
    analyse(input_file, output_file)
    print("That's all folks!")
```

NN Classifier Code

```
import numpy as np
import torch
from torch.utils.data import DataLoader
from torch.utils.data import TensorDataset
from torch.autograd import Variable
import torch.nn as nn
import pandas as pd
import sklearn
```

Import the CSV data

```
df = pd.read_csv("KNN_Input.csv",index_col=0)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df.drop('State',axis=1))
df_feat = pd.DataFrame(scaled_features)

feat = scaled_features
label = df['State'].to_numpy()
```

One hot encoding the labels(targets)

```
nsize = label.size
```

```
nmax = int(label.max())
b = np.zeros((nsize, nmax+1))
b[np.arange(nsize),label] = 1
label = b
```

Transform the data into tensors

```
feat = torch.from_numpy(feat)
label = torch.from_numpy(label)
```

Create a dataset

```
dataset = TensorDataset(feat, label)
```

```
train_set, val_set = torch.utils.data.random_split(dataset, [int(len(dataset)*0.8),int(len(dataset)*0.2) ])
```

Split the data into training and testing

```
train_dataloader = DataLoader(train_set, batch_size=training_batch_size, shuffle=True)
test_dataloader = DataLoader(val_set, batch_size=testing_batch_size, shuffle=True)
```

```
train_iterator = iter(train_dataloader)
x_batch, y_batch = train_iterator.next()
```

```
test_iterator = iter(test_dataloader)
```

```
a_batch, b_batch = test_iterator.next()


Tunable parameters
training_batch_size = 20
testing_batch_size = 5
learning_rate = 0.001
epochs = 500


Building the model
model = nn.Sequential(
    nn.Linear(5, 64),
    nn.Linear(64, 2),
    nn.Softmax(dim = 1)
)


Defining loss and and optimization methods
loss_function = torch.nn.MSELoss()
optimizer = torch.optim.SGD(model.parameters(), lr= learning_rate)


Training
for epoch in range(epochs):
    for x, y in train_dataloader:
        optimizer.zero_grad()
        y_pred = model(x.float())
        loss = loss_function(y_pred.float(), y.float())
```

```
        loss.backward()
    optimizer.step()
    print(f'epoch: {epoch}, loss: {loss}')
```

Saving the model

```
torch.save(model.state_dict(), 'checkpoint.pth')
```

Testing the model

```
def validation(model, testloader, criterion):
    test_loss = 0
    accuracy = 0
    val_losses = []
    for x, y in testloader:
        y_pred = model(x.float())
        test_loss += criterion(y_pred, y).item()
        val_losses.append(test_loss)
        ps = torch.exp(y)
        equality = (torch.exp(y.data) == ps.max(dim=1)[1][1])
        accuracy += equality.type(torch.FloatTensor).mean()
    return test_loss, accuracy, val_losses
```

```
loss, acc , lp = validation(model , test_dataloader , loss_function)
print(acc)
import matplotlib.pyplot as plt
plt.show()
```

SVM Code

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.svm import SVC

from sklearn.metrics import confusion_matrix

from sklearn.preprocessing import LabelEncoder

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.colors import ListedColormap

from sklearn.metrics import accuracy_score



data = pd.read_csv('SVM_Inputnumbered.csv')

#print(data)

training_set,test_set = train_test_split(data,test_size=0.2,random_state=1)

#print("train:",training_set)

#print("test:",test_set)

x_train = training_set.iloc[:,0:2].values  # data

y_train = training_set.iloc[:,2].values  # target

x_test = test_set.iloc[:,0:2].values  # data

y_test = test_set.iloc[:,2].values  # target

#print(x_train)

#print(y_train)

#print(y_train.shape)

#print(y_test.shape)

# using labelencoder to convert string target value into no.
```

```
lb = LabelEncoder()
y_train = lb.fit_transform(y_train)
#print(y_train)

classifier = SVC(kernel='linear',random_state=1,C=1,gamma='auto')
print()
classifier.fit(x_train,y_train)

classifier_predictions = classifier.predict(x_test)
print(classifier_predictions)
print(accuracy_score(y_test, classifier_predictions)*100)

# visualizing the training data after model fitting
plt.title('Lucid Vs Deep Sleep')
plt.xlabel('Alpha Entropy')
plt.ylabel('Delta Entropy')
plt.legend()
plt.show()
# visualizing the predictions
plt.title('Lucid Vs Deep Sleep Predictions')
plt.xlabel('Alpha Entropy')
plt.ylabel('Delta Entropy')
plt.legend()
plt.show()
```

KNN Code

```
#K Nearest Neighbors with Python

#Import Libraries

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt


import numpy as np


#Load the Data

df = pd.read_csv("KNN_Input.csv",index_col=0)

df.head()


from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(df.drop('State',axis=1))

scaled_features = scaler.transform(df.drop('State',axis=1))

df_feat = pd.DataFrame(scaled_features,columns=df.columns[:-1])

df_feat.head()

#import train test split from sklearn


from sklearn.neighbors import KNeighborsClassifier


knn = KNeighborsClassifier(n_neighbors=1)
```

```
knn.fit(X_train,y_train)



KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=1, p=2,
           weights='uniform')
pred = knn.predict(X_test)

#Predicting and evavluations

from sklearn.metrics import classification_report,confusion_matrix

print(X_test)
for i in range(1,32):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))

plt.show()
```