A PIONEER STUDY OF PROCESSING EEG SIGNALS FOR BRAIN COMPUTER INTERACTION - A SOFTWARE ENGINEERING APPROACH

Prithivi Sharma, 45705704

Bachelor of Engineering(Honours)
Software Engineering Stream

School of Engineering
Macquarie University

June 9, 2023

Supervisor: Dr, James Zheng

ACKNOWLEDGMENTS

I would like to acknowledge my thesis supervisor as well as the project team who helped me get familiarised with this project and its contents for me to be able to complete my thesis paper.

STATEMENT OF CANDIDATE

I, Prithivi, declare that this report, submitted as part of the requirement for the award of Bachelor of Engineering in the School of Engineering, Macquarie University, is entirely my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualification or assessment an any academic institution.

Student's Name: Prithivi Sharma

Student's Signature: Prithivi Sharma

Date: 9/6/2023

ABSTRACT

Electroencephalogram (EEG) signals and their use in Brain Computer Interface tasks have been an integral area of study in recent times due to its applications in deep-learning models. Because of its importance in many different fields of application in software engineering, this paper analyses the different BCI models, conventional and state-of-the-art which collect and classify EEG signals using many different frameworks and algorithms. This includes traditional Machine Learning Models as well as conventional euclidean based deep learning methods and state-of-the art non-euclidean models. In addition, this paper also proposes an end-to-end deep learning model and an EEG signal evaluation method which focuses on real-time applications of EEG signals to perform one of the key and important BCI tasks, Motor Imagery. The output of this model would then be used to control a Robot car based on an evaluation setting trajectory where its movements would be recorded. This would then be used to compare the trajectories of the movement of the same car within the same evaluation setting trajectory through a controller. The software development cycle used for this experiment focuses on the collection of raw EEG signals, filtering of the data through preprocessing, feature extraction of the preprocessed signals and their classification. In addition, the use of Real Time Location coordinate units would be used to feed the coordinates of the car with a tag placed in the front of the car. This model is then used as an area of learning to further venture into many other elements used to analyse EEG signals for conducting BCI tasks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Most recently, BCI has gained an incredible amount of notice as an effective pathway to establish human-computer interaction for a wide array of skills. BCI technology allows the human brain to be able to communicate with other systems and in some cases, even affect the behavior of other devices. This groundbreaking innovation can contribute in many different tasks like emotion regularity and motor movement of humans, allowing us to better understand how the human brain works [6, 17, 18].

EEG, similarly to BCI has been a very popular choice as a conduit in BCI research. This is not only because of its application in generating electrical activity of humans but also opening up new avenues of analysis like the identification of patterns based on different actions or procedures [1, 5, 7]. By being able to explore this area of analysis and being able to integrate that with BCI tasks, researchers are able to generate models that are capable of classifying BCI tasks based on the area they are looking to address.

The main aim of this research paper is to put forward different ways to evaluate EEG signals and the effectiveness of deep learning BCI models in real time experiment settings. We aim to do so with the introduction of a non-euclidean based end-to-end deep learning model which highlights relationships between EEG signals and the channels used to collect them with more fine-tuning through Riemannian Geometry [5]. In summary, we aim to evaluate the accuracy of real time signals through frequency related methods like Event-Related Desynchronisation and time and frequency domain graphs when collecting EEG signals [17, 18]. As a result, this allows to increase the accuracy and improve future use of BCI systems, resulting in solutions which provide a stronger understanding on EEG signals.

Explicitly, this research paper will address the research question of how EEG signals can be processed more effectively especially after the growth of many traditional, conventional and state-of-the art models that have been used to classify BCI tasks through EEG signals [15, 24]. In essence, we will be exploring the workflow of EEG signals within conventional and state-of-the art BCI models with data preprocessing through the lens of data collection, data filtering and training set creation. We will also be analysing the motivations behind the selection of these models and how their unique frameworks allow them to meet the aim for the model in terms of the analysis of EEG signals. To finish off, this paper will also analyse how these models are tested within their respective datasets to generate their accuracy [20].

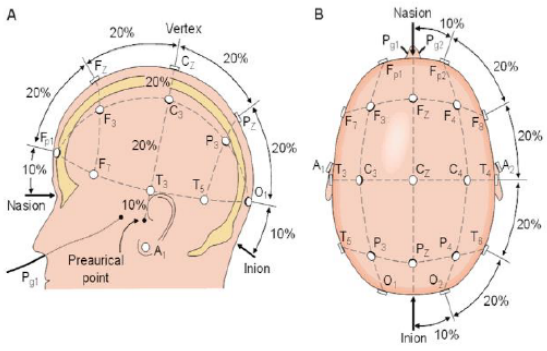


Figure 1.1: 10-20 EEG system

In addition, we will explore the real-world experimental setup for this research paper. It will provide a detailed walk through on how our purposed model, TensorBasic is trained with an emphasis on imagining motor movement during data collection [18]. We will also explore how it is tested with real-time experiments that ask for participants to imagine the same movement to move a Robot car. This will allow for the model to generate an accuracy and explore its application in generating accurate predictions for BCI related tasks.

Lastly, this thesis will contribute to the evaluation of the performance of EEG signals in practical and real-time BCI applications on real-time datasets from the analysis of the movement of the car based on an evaluation setting and a controller which performs the same movement as the model to the car when connected. Through this evaluation, we will be able to assess the useability and validity of our proposed BCI model in processing real-time EEG signals and classifying them correctly [5, 15, 17, 18].

In conclusion, this thesis contributes in increasing the overall comprehension of EEG signals and how they are used for BCI tasks. By investigating how raw EEG signals are processed and understood in real-time with a strong model, we look to explore more ways to build more reliable and efficient BCI systems that open opportunities to learn more about the importance of properly analysing EEG signals for BCI models.

1.0.1 Background

An Electroencephalogram (EEG) cap is used to collect information from a participant from the electrodes placed on a cap. Although there are many approaches applied in the placements of electrodes, the International standard of 10-20 is implemented as a means to describe the position of electrodes within an EEG cap as shown in as seen in Fig 1.1 [1]. The electrodes are used to look over brain activity of a user by recording signals from the brain. This is done by analysing the brain wave bands that are found in EEG signals that include Delta(1-4), Theta(4-8), Alpha(8-14), Beta(14-31) and Gamma(31-50). EEG caps, then transfer the raw signals to computer systems through different means.

Collection The collection of EEG signals used for processing in models are achieved through some major steps. The first steps include choosing the type of stimuli used to collect brain activity and data from EEG caps. Once decided, the number of participants whose data are collected is chosen. Once chosen, chosen participants interact with the

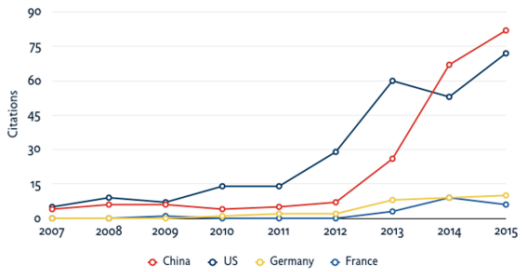


Figure 1.2: Publications about Deep Learning from - China, USA, Germany and France

stimuli picked for them and data is collected for processing.

Processing Raw EEG signals collected from all channels are then passed on for pre-processing. Pre-processing involves the removal of unwanted data, which are normally identified as noises. This process also includes the division of data into training and testing data, allowing for data to be clean, resulting in accurate models.

Feature Extraction The stage of feature extraction is used as an area to further process pre-processed data by extracting features through many different algorithms catered to certain elements of the EEG signals. This can range from differential entropy(DE) for temporal features [7, 30] to fast fourier transform(FFT) features. Consequently, this allows for the classifier at the end of the model to be able to extract an emotion based on the signal provided and the functions used to process it with more accuracy.

Classification The classification stage is used to select the classifiers that are used to allocate an emotion based on the number of classes and model selected. This can include valence-arousal which is selected for majority of the projects used for EEG signal processing models for emotion detection.

1.0.2 Related Work

Earlier iterations of BCI models used to collect, process and classify EEG signals are mostly centred around the implementation of hand-crafted algorithmic approaches with deep belief networks and machine learning classification libraries such as Support Vector Machines(SVM) and K-Nearest Neighbor(KNN). They had limited capabilities on feature extraction which would lead to increased computational resources in the creation of a classification model resulting in the application of deep learning methods over the years as seen in Fig 1.3 [21].

Deep Belief Network

Wei-Long Zhong et al investigated the use of Deep Belief Networks with EEG signals acquired from 15 participants for BCI on the DEAP dataset. [32] The main feature used for extraction involved the application of DE and value comparisons on the left and right directions of the EEG cap through the differential asymmetry and rational asymmetry. Classification was achieved using Restricted Boltzmann Machines to create a Deep

Belief Network. Although it generated relatively good scores with useful formulas and focus towards asymmetrical brain activity for accuracy, some practices in its application required more resources to be applied successful. This included the use of Restricted Boltzmann Machines which are known to be more difficult to train and requires weight adjustment [32]. This can be improved with a network architecture that includes formulas that are more robust and capable of handling large data without needing more time to train and classify signals through the means of BCI.

Hao Chao et al applied a Deep Belief Network Conditional Random Field by using AMIGOS, DEAP, SEED and HR-EEG4EMO datasets. Features were extracted using time-domain, frequency domain and time-frequency domain functions such as mean, variance, zero-crossing rate and power-spectral density [4]. Although the DBNCRF model applies useful features extraction methods based on frequency band placement and time based signals, its accuracy can improve by focusing on using samples from high frequency areas and by using classification features from state-of-the art deep learning algorithms especially with large scale data [4].

Mohammad Mehdi Hassan et al proposed a deep belief network with an implementation of the physiological signals of Electro Dermal Activity(EDA) along with Photoplethysomogram(PPG) and Zygomatics Electromyography(zEMG) and a Deep Belief Network to be able to find depth level features through BCI. The observations of the 3 physiological signals were determined using the principles of the DEAP dataset. The DBN is structured and trained through the successive training of the 3 Restricted Boltzmann machines and sigmoid belief network [13]. Feature extraction is achieved with the 9 statistical features of EDA,PPG and zEMG, 9 Power Spectral Density features of EDA, PPG and zEMG and 46 features of DBN and are then classified using a Fine Gaussian SVM classifier. Although this network focuses on ensuring high frequency EEG data from collected signals is used for processing and classification through a high pass filter, making more accurate assumptions, it also overlooks the increased training time required for the RBMs along with the classifier due to the large data it is inheriting, resulting in increased computational resources to train the model and generate an accurate assumption. One of the ways to tackle this, however, can be through the implementation of certified state-of-the art deep learning algorithms [7, 23]to minimise the likelihood of errors with built-in and reliable feature extraction and classification algorithms which can allow the model to have a high level of accuracy while also handling large amounts of EEG signal-based data at a better rate.

Support Vector Machines

Fabian Parsia George et al implements an SVM classifier with the standard collection, processing and classification procedure for EEG signals with the DEAP dataset for music videos. Band extraction and feature extraction is achieved by the application of FFT and segregated into 4 values of emotional feelings based on valence and arousal, High Arousal- High Valence(HAHV), Low Arousal- High Valence(LAHV), Low Arousal- Low Valence(LALV) and High Arousal- Low Valence(HALV). Classification is achieved by the use of 10-fold cross validation which takes in the scaled statistical features of the input signals for each of the 4 emotional quadrants [11]. Although this experiment was successful in the usage of efficient feature algorithms which are well known for achieving accurate classifications, this model is also hard to maintain due to the inability of the SVM classifiers being able to handle large-scale data as evident from the action taken

to reduce the channels that are related to the frontal lobe of the brain. Moreover, the manipulation of the data provided along with all important channels not being used for processing can lead to the model not being a reliable body for signal classification for EEG signals. This can be improved by moving to a classification model which is able to use feature extraction methods that can cover all areas and handle long sequential data which is essential for EEG signal processing based models. [7, 30].

Itsara Wichakam et al used a similar implementation process as [11] with an SVM classifier but with multiple normalisation techniques. However, some of the processing steps taken for their model are different due to their stimuli handling 20 songs. The signal processing involved self-assessment after every song was played from the participants which would directly be represented using the Self-Assessment Manikin(SAM) to show the different scales of valence and arousal with the data collection process focusing on the training of users on how the experiment is conducted and then being tested two times within 5 separate days. The EEG data collected was then passed for feature extraction methods involved FFT and PSD. The normalisation techniques used for the features extracted included Re-scaling, Z-score standardisation and Frequency Band Percentage with classification done by comparing the performance of the model on the 2 separate tests done through F1 score. Although this model made good use of feature extraction and normalisation methods to pick out high frequency EEG signal areas through BCI and process them as seen from its effort to pick the best normalisation technique based on the information provided by using multiple SVM kernels [27], one of the areas this project can be reflected on is of using more pre-processing techniques to remove noise and be able to input signals of strong frequency levels only [20]. One of the areas this project did introduce me to is multiple sessions of testing as it proved to be useful in generating better signals to process in the main network.

Noppadon Jatpaiboo et al on the other hand, propose an EEG signal processing network using BCI with less EEG channels and frequency bands with an SVM classifier achieving a high accuracy. Pre-processing involves choosing 100 stimuli based on the highest and lowest scores on valence within the Geneva Affective Picture Database and Artifact filtering for removing unwanted data. EMOTIV caps and dataset are used in the data collection phase, with 14 channels being used to collect EEG signals. Participants are shown all pictures chosen as their stimuli for 10 seconds with a 5 second time period to adjust signal readings after one picture has been shown and the process continues afterwards. Features are extracted to the 5 frequency bands of Delta, Theta, Alpha, Beta and Gamma through Wavelet Transform(WT) [**?**, 14]. Normalisation of the features from the 14 channels are done using scaling between 0 and 1 and classification is achieved using Gaussian SVM with 10-fold cross validation. The strong points highlighted with this model involve the use of less channels to still generate a strong level of accuracy while also implementing strong pre-processing techniques such as Blind Source Separation to remove unwanted information so that sequences that can be processed to generate a model with strong accuracy is created. Moreover, other practices to improve upon the generated accuracy like reducing pairs of channels and frequency bands also assisted in the learning aspect of this project as it supported how pairing channels close to one another can generate sequential data that is a lot stronger and can give more information on the model itself [7]. An area where improvements can be made is by making comparisons to other sections within the human brain which this project has overlooked which can allow the model to learn more about the relationships between other channels and create

pairs of channels that can make the model more accurate than it already is [15].

K-Nearest Neighbor

Mi Li et al proposed a multichannel EEG signal model using BCI with a K-Nearest Neighbor classifier [22]. Data collection involves using the DEAP dataset with 32 datasets and 40 videos of 60 seconds. EEG data from 10, 14, 18 and 32 channels from the left and right side of the EEG cap are selected for pre-processing which is done using Average Mean Reference(AMR) and normalised using min-max before the extraction of features. Feature extraction for this model involves the use of Discrete Wavelet Transform(DWT) and features for every channel are generated through Entropy and Energy. The value of K was set to 3 for classification after the extraction of important features. Extracted and selected features are classified using valence and arousal dimensions for the 32 participants of the experiment. *Mi Li et al* practice established methods for the analysis of EEG signals like the very well known Entropy and energy function [16, 23]but does contain some elements that can be improved in order to generate a stronger level of accuracy. One such example is to focus more on a different type of model for classification mainly due to the computational resources and effort to keep a model like operating efficiently using a KNN classifier as data increases in size.

Fatemeh Bahari et al propose a BCI network which processes eeg signal using a KNN classifier and recurrence plot analysis. Pre-processing actions included SAM being filled out by the 32 participants based on music and videos after displaying one of 40 music videos for one minute [3]. EEG signals were recorded using 512 Hz and downsampled to 128 Hz. Noise removal was conducted using bandpass frequency filter. Feature extraction involve the use of Recurrence Quantification Analysis and Recurrence Plot Analysis. Classification was achieved through the use of methods such as t-test and area under ROC curve. Case independent classification involved the exclusion of the first 10 sets of features randomly and the use of statistical methods and the Bhattacharyya distance. KNN and valence, arousal and liking were used as classifiers for the final classification. Case dependent classification was also implemented across all channels using a LOO procedure and included same optimal values for k values and distance metrics. Some areas which were responsible for the low accuracy generated from this model was the improper use of the channels during classification as the exclusion of extraction of temporal features as this model focused only on spectral features of EEG. This is further supported from the lack of research that was conducted on EEG signal processing [3]. Model accuracy can be improved by using different classifiers and the inclusion of other extraction methods that have been known to be effective in EEG signal processing methods that are made using BCI [23].

Kaundaya et al breaks down an EEG signal processing model with a KNN classifier. Pre-processing involves the collection of EEG signals from ground truth from visual and audio stimuli. Band pass filter is used to remove noises of 50Hz and the DC offset from each electrode of an EEG cap [19]. Feature extraction process involves statistical parameters calculation using WT and coefficients for four level decomposition are extracted using mean, Variance, standard deviation, entropy, root mean square, skewness and power [**?**]. KNN classification model is uses the new labeled sample passed through as testing data with the baseline data during preprocessing as the training data. The classifier is eval-

uated from the varied value of k from 1 to 10. Accuracy is obtained from the values of the total datasets that are selected for the model. Although this model is thorough with classification analysis through the examination of different k values from the KNN classifier, increased datasets can prove to be a challenge in terms of maintaining the accuracy of the model. Moreover, feature extraction methods like Linear Discriminant analysis can prove to be more efficient for this model as it allows for more maximisation within mean values of each class and can lead to less overlap between classes which is essential for KNN classifiers, therefore, improving its accuracy.

Chapter 2

Deep Learning Algorithms To Analyse EEG Signals

2.1 State Of The Art Algorithms (Deep Learning)

State-Of-The-Art Deep Learning Algorithms are a strong method in identifying patterns in computer vision and have been heavily used in image classification. Some recent studies implementing these models [7, 23, 30] have been in-part successful in collecting and processing EEG signals through BCI. RNN models have also been another type of model which has been implemented widely in analysing EEG signals using BCI through sequential data analysis within areas such as speech recognition, language translations and image processing. Although this suggests that they contain capabilities in examining EEG signals, other more advanced methods have been introduced recently that have been able to better explore EEG signals through a thorough study of time and frequency domain based understandings being implemented with a non-euclidean nature for EEG signal processing.

2.2 Convolutional Neural Networks (CNN) And Recurrent Neural Networks (RNN)

A convolutional neural network is a feed forward, deep learning neural network that is applied to process structured pieces of information. These include images as well natural language processing for text classification. They are widely known to be a state-of-the-art neural network when it comes to computer vision and image classification. Recurrent Neural Networks (RNN) is also another deep learning neural network which The software development cycle involved with EEG signal analysis cover the initial stages of data preprocessing which cover the collection of signals, what hardware and software components are used to achieve that, data filtering which look at initial measures taken to capture information that is useful for modeling preprocessed data which is used in feature extraction and classification. These include other types of artifacts that are a part of the input EEG signal and signal-to-noise ratios. The final area of the software development cycle then involves the modeling of the preprocessed data which is used to select and extract features based on parameters set for a model and how that translates into generating a classifier based accuracy for the signals that are processed.

2.2.1 Data Preprocessing

Data Collection

Regional Asymmetric CNN *Heng Cui et al*'s regional asymmetric CNN model's signal collection process employs DEAP and DREAMER dataset's signal for analysis [6, 26]. For the two binary classification tasks of the DEAP dataset, the data for each trial from each of the datasets contained 3 seconds of baseline data and 60 seconds of trial data. EEG signals were used for the model with all EEG signals being sampled at 500 Hz and being down-sampled to 100Hz. The threshold value of 5 was used to divide the data into two classes of classification. After collecting data for each participant, their measures of Arousal, Valence, Liking and Dominance were collected for each video, ranging from 1 to 9 [9, 23]. In contrast, for the DREAMER dataset, the length of the clips were not fixed, with durations extending from 63-393 seconds long. In addition, baseline data of 60 seconds was collected before each film clip that was shown to the participant [6]. 3 was used as the threshold value which was the most optimal one to be able to divide the data based on their classification values.

2D CNN *Trun-Duc-Thinh-Phan et al* on the other hand, purely rely on the use of the public dataset, DEAP 32 participants of equal numbers of genders(male and female). The data was collected using 40 one-minute music videos similar to [7] with the intention of evoking an emotion out of a participant. After viewing each of the clips presented to a participant, they evaluate their emotional state amongst the classification labels of valence, arousal and dominance from 1 to 9. A trial for each participant amounted to 1280 signal sets as evident from the number of participants(32) and the number of videos(40) they watched [6, 7, 23]. For each set, the first 3 seconds were used as baseline signals where the participant has not watched a stimuli yet and 60 seconds were used as the EEG signals which were collected [23, 24].

Parallel Sequence Channel Projection CNN *Lili Shen et al* also used the same framework as to [23] as it used the DEAP dataset with 32 participants watching 40 music videos with the duration of 1 minute respectively. A self-assessment of each video using the SAM scale is done for valence, arousal and dominance classification [24, 25]. The sampling frequency of the signal is collected at 512 Hz and then downsampled to 128 Hz. By being able to do so, this allows for the EOG artifacts to be removed from each of the signals which are collected from the DEAP dataset. The change in their model's development structure was the separation of the collected signals for all participants with a focus on Temporal Stream (TS) and Spatial Stream (SS) respectively [25]. This allows for EEG signals to be filtered with more ease as each of the subnetworks would handle the extraction of temporal and spatial information of EEG signals that are collected with their Baseline noise filtering module.

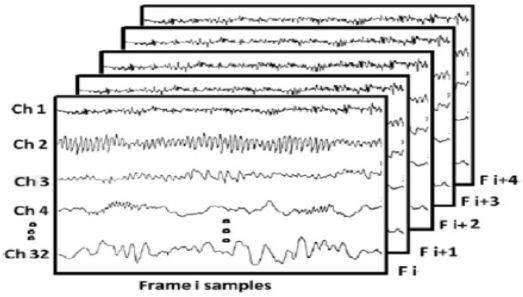


Figure 2.1: EEG signal chunk

3D CNN *Elham S. Salama et al*'s 3D CNN model uses the DEAP dataset and face data which is collected from input video. The signal collection process involves the use of 40 music videos that last one minute where the signals are collected using 512 Hz. The data from all channels are segmented into frames and the main input of EEG signals are 5 of those frames as shown in Fig 2.1 [24]. The first 3 seconds of the signals are removed as they display the initial emotional state of the participant which is not required for the model [23, 24]. Upon viewing every video, a self-assessment of the ability of the videos to express feelings of arousal, valence, dominance and like/dislike.

The facial expressions which are collected using the input video are collected using 22 participants. 60 frames are collected as they correspond to the total length for each video. This is done taking a frame from every 30 consecutive frames [24].

GRU-MCC The collection of data revolves around the use of two public datasets, SEED and MPED [9, 30]. The SEED dataset contains EEG, EOG and EMG signals whereas the MPED dataset comprises of types of data that are more diverse, ranging from EEG, ECG, respiration and galvanic skin response [7]. The trial data consisted of 3 sessions with 15 participants and 15 emotional video clips with an even distribution of 5 positive, neutral and negative clips. The sessions are conducted with an interval of 2 weeks. The collected data has 62 EEG channels with a sampling rate of 1000 Hz [7, 30]. Regarding the data collected from the MPED dataset, the data collected from this dataset contains only one session and comprises of physiological signals of 23 subjects from 7 kinds of emotional clips. This results in 28 trails for one participant.

Attention Based LSTM with Domain Discriminator (ATDD-LSTM) The signal collection procedure involved with this project uses EEG signals that are captured from multiple public datasets(DEAP, SEED, CMEED). The data collection procedure for each dataset is different in comparison to the other in order for the model to start its initial phase of filtering data in its software development cycle due to the difference in the number of channels and the stimuli presented. This can be evident from Fig 2.2 [9].

With the DEAP dataset, data is collected from 32 participants who watch 40 music videos. Each EEG signal lasts 63 seconds with 3 seconds of baseline data and 60 seconds of EEG signals [6, 9, 24]. After each video, self-assessment based on feelings of arousal, valence, like/dislike and dominance from the range of 1 to 9 are done. Furthermore, the collected EEG samples are downsampled to 200 Hz in order to remove EOG artifacts.

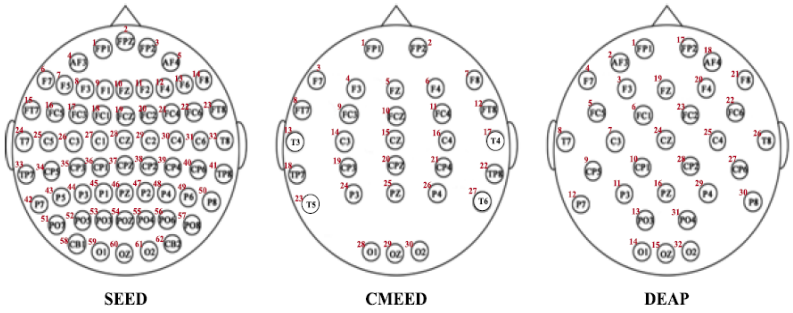


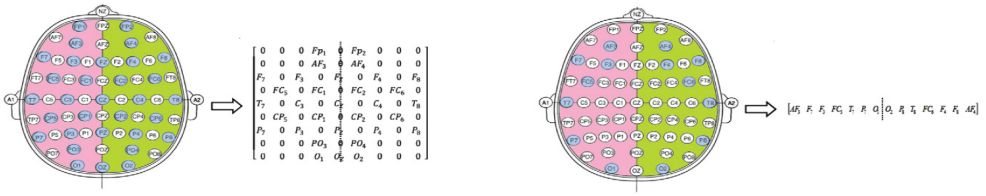
Figure 2.2: Order of electrodes for EEG signal collection for SEED, DEAP and CMEED

With the SEED dataset, data is collected from 15 participants upon watching 15 Chinese movie clips. EEG data that is collected is around 4 minutes long, corresponding to the length of each movie clips. Similar to the DEAP dataset, a self-assessment is done by each participant in giving their evaluation on the type of emotion the clip evoked out of them(positive, neutral, negative) and EEG recordings are downsampled to 200 Hz to remove other artifacts which are not required for processing the EEG signals [6, 30].

The CMEED dataset has collected EEG data from 37 participants after watching 16 Chinese movie clips which are 2 minutes long. Self-assessments liking to arousal and valence are done after each video as well. The recordings from this dataset, however, were downsampled to 128 Hz rather than 200 Hz like SEED and DEAP [9].

Spatial Temporal RNN The signal collection phase of the Spatial Temporal RNN uses data from the public dataset SEED with 15 participants with signals calculated in 4 sections. The stimuli used as videos comprised of 15 movie clips from 6 Chinese movies [9, 30]. 5 seconds were used before a clip began to get participants familiarised with the experiment environment. 4 minutes were used to capture EEG signals, 45 seconds were used as a period of self-assessment for participants to rate the video based on categories of positive, neutral and negative emotion and 15 seconds were used to rest the participants before the next video started playing [30]. Self-assessment questions focused on getting not just the emotional response of the participants upon watching the movie clip but also focused on understanding if they had watched that clip before and how much of the clip they understood. The sampling rate used in the collection of data was 1000 Hz through 62 channels [7, 30].

ALL-LSTM and WT-LSTM The signal collection conducted with the ALL LSTM and WT LSTM model uses a personalised dataset comprised of students of Menzin University of Beijing with 20 students acting as participants [16]. The signal collection framework of this model looks to collect data from 62 channels by presenting participants with 12 videos as an initial measure and pick 6 videos for another trial after participants rated videos based on their ability to elicit emotion in terms of arousal and valence classification from 1 to 9 [9, 16]. Those six videos were from 110 to 120 seconds as signals captured from videos that are 1 minute to 10 minutes long would allow for a single emotion to be evoked by participants [16]. Moreover, this form of data capture would also eliminate the likelihood of capturing multiple emotions within one signal, allowing for features to be extracted with more ease.



(a) DEAP (b) DREAMER

Figure 2.3: Spatial Encoding of EEG channels from DEAP and DREAMER

Data Filtering

RACNN For the Regional Asymmetric CNN Model of *Heng Cui et al*, The filtering of the data was done based on the aspect of baseline signals being used to improve emotion recognition. This was achieved by being able to divide the trial and baseline signals through n-s and m-s respectively into 1-s samples for each trial. Once completed, the baseline samples m are then used as a representation of basic emotional states of the participants when watching a clip. The final filtering process implemented is used to remove the basic emotional state of the participants is accomplished by removing basic emotional state of the participants by finding the difference of each of the trial samples and the mean value of the trial that is being looked at. This allows for the final preprocessing samples through Z score normalisation to be obtained as shown in the expression for the matrix: $x^N \epsilon \mathbb{R}^{c*s}$ where c and s correspond to the number of channels and the time points of the signal upon the implementation of Z score normalisation [6, 7].

Another process of filtering which is implemented for this model is the spatial encoding of the channels as seen in Fig 2.3 [6]. This is one area of difference with the models as this allows for the spatial relationship between electrodes to be identified [6, 20]. For the DEAP dataset, EEG electrodes are marked into a matrix of h x w with h and w being 9. Interpolation methods of Gauss Interpolation and linear interpolation are not added to channels that are not required for processing in order to reduce the likelihood of unrelated noise [6].

For the DREAMER dataset, the use of 14 electrodes allows for all channels to be encoded into a vector that allows for relationships of the electrodes to be formed collectively. This is achieved with the transformation of the vector into a h x w matrix with values 1 and 14 respectively. This allows for consistency to be maintained with the matrix and channels of the DEAP dataset [6, 26]. This lets adjacent EEG channels to maintain relationships within the matrix and to be able to form a channel pair for the left and right channels of the EEG framework while being symmetric to the middle area of the matrix, separating them [6]. Once implemented, the matrix obtained from the DEAP dataset filtering process($x^N \epsilon R^{c \times s}$) can be mapped to $x \epsilon \mathbb{R}^{h \times w \times s}$ to allow clean data that corresponds to each side of the head as shown in the matrix to be added to the main model for feature extraction [6]. Moreover, this allows the channels that are right next to one another maintain their relationships within the matrix and overlapping is prevented with a channel pairing to the left and right hemisphere of the cap is maintained in the vector as well [6, 15].

2D CNN With the 2D CNN's filtering process, *Tran-Dac-Thinh-Phun et al* employs the use of a band-pass filter with cut-off frequencies between 4 and 45 to the downsampled

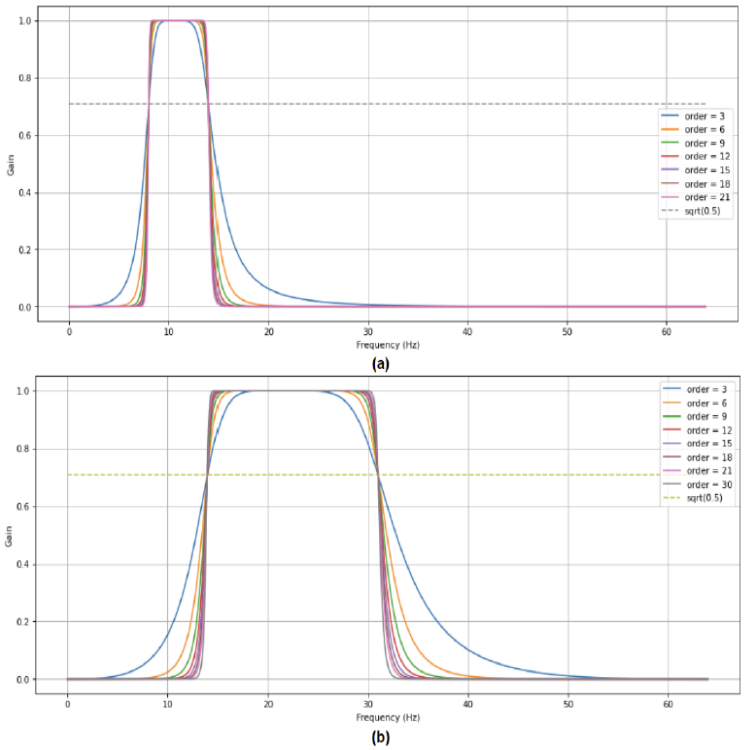


Figure 2.4: Effect of Butterwork band pass filter on Alpha and Beta Frequency band

version of the preprocessed version of the EEG signals from the DEAP dataset [23]. This was employed to remove EOG artifacts and noises [9, 23]. Once the EEG signals were obtained without any other form of interference, the Butterworth filter was employed into 4 frequency bands of Theta, Alpha, Beta and Gamma Bands. This allows for low frequency responses and ensures the minimisation of the areas of the transition band as shown in Fig 2.4 [23]. Frequencies within the delta band were removed as a result of the band-pass filter which was applied earlier.

Frequency domain values which were filtered from the band-pass filters were transformed to time signals to conduct more analysis [23, 24]. The high and low cut of the frequencies were instrumental for the model moving forward as this would allow for the area of transition to be accurately identified and minimise the stop band, resulting in the order of the Butterworth filter to be 30 [23]. Once the preprocessed data is transformed into time signals for each band, the signals of 60 seconds were then divided into 10 segments using a sliding window technique, resulting in each segment being 6 seconds long. This allowed to ensure there was no form of overlap between each of the segments for the training and testing datasets and treated as individualistic segments in a signal sequence.

PSCP-NET The filtering process involved with the PSCP-NET model involves the use of collecting the emotional and baseline signals that are highlighted before [6, 23, 25]. The differences between the emotional and baseline signals are emphasised with a filter that removes baseline signals that contain a strong fluctuation [6, 25].

The first 3 seconds of baseline signals of each channel are transformed into a dictionary, (i, p_i) with i used to record the initial order of the sampling pointp_i. The dictionaries are then sorted in ascending order based on the values of the sampling point. The middle 2 seconds of the baseline signals for each channel is intercepted and put in ascending

order. Once made, violent signals are removed and a filtered baseline vector that covers all channels are shown in (1).

$$FBV = [F_1, F_2, ..., F_32]^T \epsilon \quad \mathbb{R}^{32X256} \tag{1}$$

The emotional signals are then segmented into matrices of (32 X 256) which are then used along with the values of FVB for each channel for each participant. All of those matrices upon the removal of the FVB values are then combined into a matrix, combining the differential representations to be the same as the emotional representations without any interference [25]. This in turn allows for the model to decode information. After all differential samples are collected, they are sliced for the temporal (32 X 128) and spatial stream (128 X 32) with non-overlapping segments [25]. Once that is done, each sample is normalised throughout non-zero elements with Z score normalisation using (2):

$$z = \frac{x - \mu}{\sigma} \tag{2}$$

With x being the non-zero element, u denotes the average value of all non-zero elements and o represents the standard deviation of all sample elements [23, 25].

3D CNN The initial EEG signal filtering steps that are undertaken with the 3D CNN by Salama et al to improve the EEG signal that is being outputted involve the down-sampling of the signal into 128 Hz. This allows for the removal of eye artifacts which are a part of the DEAP dataset [9, 24]. Moreover, the use of a band-pass filter within the frequency range of 4–45 Hz is applied to EEG signals [7, 23, 24]. This is done in order to remove EOG artifacts that are a part of the signal. The frames that culminate to create one frame of EEG signals are used as a means to capture temporal domain data and to be added alongside the EEG chunk. This allows for the model to be able to better classify emotions better based on the temporal data is captured which can then be trained and tested with feature extraction methods to understand what emotions are being evoked and what channels are useful in doing so [22, 24]. The updated values are then passed on to the area of final extraction and classification element of the model for the classification which uses the fusion chunk to classify emotions.

The filtering methods that are used before the input video segments are added into the model firstly involve the use of a data augmentation step in order to ensure that they are not lacking data. These include flipping, updating the colour and adjusting the brightness. Flipping would allow reflect the frame around a vertical axis, updating the colour would convert the RGB pixel values to Hue-Scale-Variance(HSV) values and convert it back to RGB values by updating the hue, saturation and variance pixel values [24]. Upon the addition of a constant value to the input segments to update the brightness value, a Mask-RCNN instance segmentation is implemented in order to remove background displays within the face frame and then passed on to the final extraction and classification areas to classify the signals and facial data together.

GRU-MCC The filtering of data for the model were different for both of the datasets although they contained the same number of channels for data collection and their respective sampling rates. As a part of the preprocessing steps for the data collected from

the SEED dataset [7, 9], EEG signals were downsampled to 200 Hz. In addition to that, recordings that contained EMG and EOG signals were gotten rid of. Moreover, bandpass frequency filters were applied to signals ranging from 0-75 Hz of the signals that were collected. This allowed for important data points to be only selected which further helped with finding features without any noise. Once done, the EEG signals after the bandpass frequency filter was applied, the samples are then divided into samples for every second with no overlap [7, 23, 24].

The MPED dataset, on the other hand, Independent Component Analysis(ICA) was applied to remove EOG samples within the data of the MPED dataset signals only. From doing so, EEG signals are only applied and not been contaminated by other types of signals. In similarity to the SEED signal, the EEG signals were filtered and divided into 1 second samples.

ATDD-LSTM

The filtering process involved with the ATDD-LSTM models involve a similar concept that is seen in [7]. This includes the collected EEG recordings being downsampled to 200 Hz [7, 9]. This ensures other artifacts from the collected signal are discarded and only include signals that are appropriate for further processing. The second element of the filtering process involves the application of the bandpass filter that ranges from 0-75Hz [7,9]. This is done to select frequency bands that correlate to processing EEG signals in accordance to the SEED dataset as the collected data from this dataset contains more electrodes and more artifacts in comparison to other public datasets like DEAP [6, 9, 23]. Frequency bands of Alpha, Delta, Beta, Gamma and Theta are then selected. These allow for signals to be segmented based on the frequency bands they belong to and allows for the feature extraction to be done more easily as shown in Fig 2.5 [9].

STRNN The filtering process involved with the Spatial Temporal RNN initially involves the initial filter of the signal for the 62 channels based on the five frequency bands that are used to identify brain activity, Delta, Theta, Alpha, Beta and Gamma in the SEED dataset [7, 9, 30]. If any continuous sequences are identified, a 256 point short time fourier transform with a non-overlapped window of 1 second is used to find signals that fall under the frequency bands and then the DE is calculated for each frequency band [30].

Once that is done, the discrete sequences in the 5 bands of the 62 channels are calculated. A slicing window of 9 seconds is used to examine each sequence by one step [23, 30]. For each step, the slicing window sequences are applied as the point which is in the centre of the slicing window. This ensures that temporal dependencies are covered for each part of the signal and are used as one of the indicators to recognise a human emotion at a specific time period [30].

ALL-LSTM and WT-LSTM The filtering process involved the division of multichannel EEG signals based on time and frequency domains through the use of fast fourier transform (FFT) [16]. Once the data is collected from those domains, contaminated pieces of data is discarded through the use of digital filtering with Finite Impulse Responses (FIR) and Infinite Impulse Response (IIR) [16]. IIR allows for impulse reactions

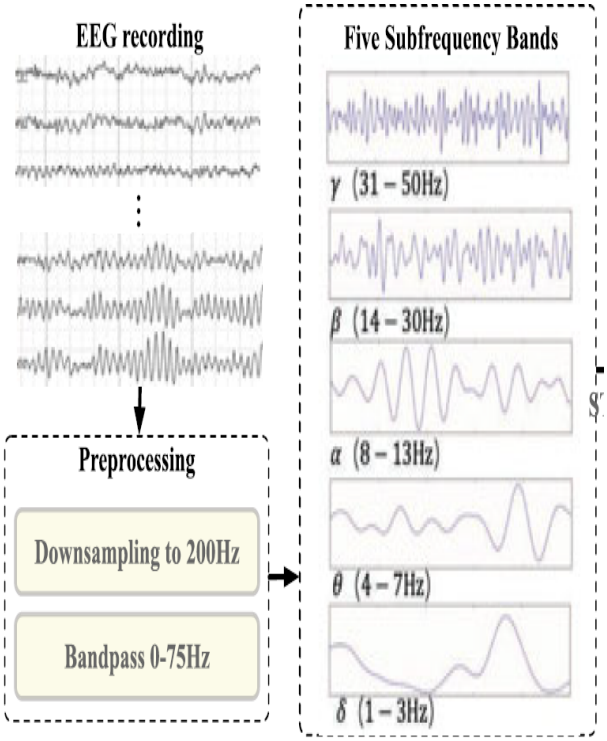


Figure 2.5: Preprocessing framework of ATDD-LSTM

of users to be captured when looking at a stimuli and can use that to be able to filter out data that is not useful in identifying emotional responses based on the time domain readings of participants when they have looked at a video in this instance. Furthermore, this filter also achieves the ability to capture sequential data for each signal when looking at changes within the sample to allow for a better reading of what they imply and how they can be used to generate readings for emotion recognition and classification.

FIR on the other hand, allows for features of the given signal to be identified within a particular set of time, making it generate linear phases which are non-recursive and their findings stable upon preprocessing of the signals that have been collected and filtered based on the domains mentioned previously [23].

Training Set Creation

RACNN The creation of the training set used for the RACNN model involves the collection of the samples from the DEAP and DREAMER datasets. The samples for DEAP that correspond to its 40 trials that are a minute long for each participant lead to 2400 samples for each subject with each sample being of the size 9 X 9 X 128 [6]. The DREAMER dataset, on the other hand, has 3728 samples for each participant. This allows for the label of the samples for each participant to be the same with the following sample. 10-fold cross validation is applied to each of the samples for each subject to generate a result based on training and testing [20, 26, 30]. The mean of the 10 fold cross validation for each participant's samples are used with 150 learning epochs as the result for each of the participants. The average of the result for all participants is used to create generate the final result of the model.

Array	Array shape	Array content
data	40 × 32 × 8064	video/trial × channel × data
labels	40 × 2	video/trial × label (valence, arousal)

Figure 2.6: Data format for participants in the PSCP-NET model

PSCP-NET The creation of the training set used for the PSCP-NET involves dividing the mixed data of the participants with their label data as in Fig 2.6 [25] being divided based on the ratio of 7:3 [25]. After successfully dividing the mixed data of all participants, a 10-fold cross validation is applied [6, 30] after 1000 training epochs to generate the accuracy of the model through implementation.

2D CNN With the model of *Tran Dac-Thinh Phan et al*, the training set creation framework involved the division of the data of all of the 32 participants from the DEAP dataset used for the PSCP-NET model with 10 non-overlapping segments which were made from each trial for each of the participants. Each segment that was generated through this process was 6 seconds long, similar to [6]. The number of samples for evaluation was set to 12,800 corresponding to 32 participants, 40 trials and 10 segments [23]. 10 cross validation was used as a means to evaluate the result of the model [6, 25].

3D CNN The training set creation for the 3D CNN model of *Salama et al* uses the ratio of 8:2 [15,24]. This was used with the intention of being able to combining the predictions for the face data, EEG data and the fusion data which generates accuracy scores with consideration to both EEG and face data. This is achieved through the resizing of the the EEG segments and the video data to 32 *128 respectively to be able to be analysed together [24].

GRU-MCC The training set creation used with the collected and preprocessed by *Heng Cui et al* with the GRU-MCC model employs the use of Leave-One-Subject-Out(LOSO) cross validation strategy [9, 15]. This involves the use of the data of one participant being used for the testing dataset and the remaining data of the participants for the training dataset. The implementation of the LOSO strategy involves 2 different paradigms : offline and online.

With the offline paradigm, the model assumes that all samples of the the participants used as the dataset is available, allowing for the unlabeled samples being implemented to train the GRU-MCC [7]. With the online paradigm, however, all samples of a new subject are not collected beforehand. The model assumes that only a few samples can be collected to make improvements to the model [7]. Similarly, with the samples of the participant whose data is used for testing, the data from the first three movie clips to improve the model with the remaining samples being used to test the accuracy of the model.

ATDD-LSTM With the creation of the training set for the model of ATDD-LSTM, there were 3 variations due to the exploration of 3 different datasets (SEED, DEAP, CMEED) that were explored and analysed as a means to generate the accuracy of the

model.

With the SEED dataset, the training set creation is generated through the division of samples for each participant with 15 trials of one participant divided within the ratio of 6:4 for training and testing. This allows for the generation of the recognition accuracy to be collected for each participant. In addition, to investigate the sessions of the SEED dataset, a Leave-one-session-out cross validation strategy is used for each participant where two sessions of the participant is used as the training dataset and the remaining one session is used for the testing dataset [7, 15]. With their subject independent model, a LOSO cross-validation strategy is applied [9].

For the DEAP dataset, the training set creation is divided into subject dependent and subject independent evaluation. For the subject dependent evaluation, a leave-one-clip out strategy was used in the division of training(**39 trials**) and testing data(**1 trial**) [9,15,26]. With the subject-independent evaluation, the division of the data is done in the same way as the SEED dataset with a binary classification task used for both evaluation strategies.

With the CMEED dataset, the subject-dependent evaluation strategy and the subject-independent evaluation strategy are performed in the same way as DEAP and SEED followed with a binary classification task.

STRNN The training set creation of STRNN by *Tong Zhang et al* involves the random division of the data from the 15 participants and their 30 experiments of the SEED dataset with a 6:4 ratio, similar to [9]. The data is then evaluated using 10 fold cross validation in order to generate the accuracy of the model.

ALL-LSTM and WT-LSTM The training set creation applied for the ALL-LSTM and WT-LSTM model involved the division of the data from each participant being divided into 10 sampling points, resulting in 100440 data for each participant. A 3:1 ratio is applied for training and testing with 75000 of the data being used for training and 25440 of the data used for testing [16]. Each training data was sent to a LSTM classification block in the creation of an epoch with the loss and accuracy of the training set is provided. The test set was then sent to the trained model to generate the accuracy of the model [16].

2.2.2 Model Selection And Testing

RACNN

The selection of the RACNN model was made and developed with the intention of being able to integrate regional and asymmetric features of EEG signals to improve emotion regularity which is one of the key tasks involved with BCI [6, 23]. The data modeling framework applied with the RACNN model of *Heng Cui et al* further emphasises with its investigation on Temporal, Regional and Assymetric EEG-based information collectively and applying a classifier to generate an accurate model that uses BCI in evaluating EEG signals. This includes the segmentation of the EEG signals that are filtered based on temporal-based parameters which are used to find important short and long-term temporal information [6, 30].

Regional based and asymmetric feature extractors are then used one after another as

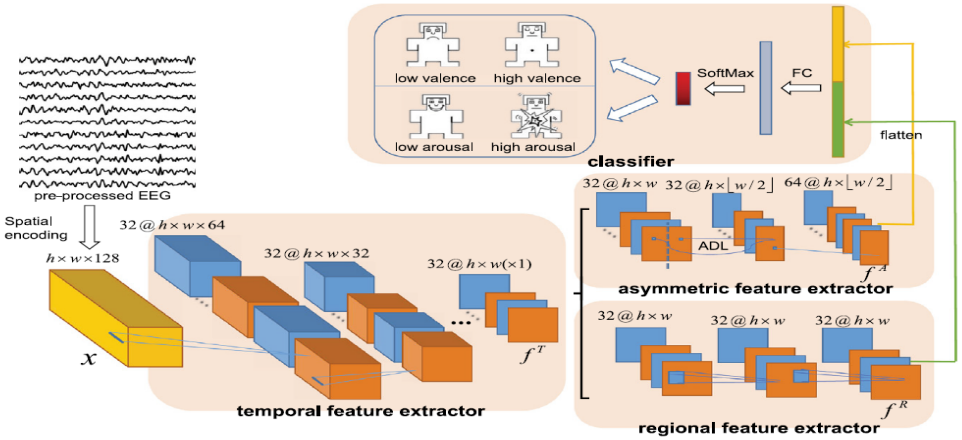


Figure 2.7: RACNN model

a means to be able to collect regional based information from the temporal features and are used to collect features through channel pairs [6, 15]. This is particularly achieved through the separation of EEG channels and their integration through a convolutional layer that captures channel pairs to maintain consistency with the output of the regional based feature extractors. The combined feature vector is passed to a fully connected layer that concatenates all findings and passes that towards the softmax classifier [6, 7, 23, 24]. A linear transformation process is applied in order to classify the probability of each feature that is found by the model with the 4 classes used for this model as displayed in Fig 2.7 [6].

The implementation of the model is achieved from setting parameters based on deep learning models. A cross entropy loss function is applied and minimised using the Adam optimiser where the learning rate is set to 0.001 [6, 7, 23]. The batch size is set to 100 and the learning epoch used for training is set to 150. The ReLU activation function is also used in the convolutional layers and fully connected layers [6, 25]. The dropout rate is set to 0.3 to ensure overfitting does not occur.

2D-CNN

The selection of the 2D-CNN model structure, similar to *Heng Cui et al* is motivated with the intention of integration of two different types of patterns through by being able to use channel and bands of channels as areas of reference to find local and global patterns of signals [6]. The 2D-CNN's data modeling framework with the preprocessed data it generates shows this with multi-feature extraction with an emphasis towards time-domain features through Differential Entropy(DE) [6, 23]. The selection and extraction of said features are achieved using 4 matrices that correspond to 4 of the frequency bands that were made during the filtering of the raw EEG signals.

Multi-feature representation is achieved by combining all features for each channels for each frequency band that were stored in their matrices. 4 multi-scale convolutional blocks with 16, 32, 64 and 128 output channels are used with a convolutional layer containing

batch normalisation [23]. A Rectified Linear Activation Unit(ReLU) is applied to find channels with higher correlation with their adjacent channels and to improve the efficiency of the model by cutting down parameters which are then passed to a fully connected layer and the classifier [21, 23].

A softmax classifier is applied to evaluate the features generated by the 2D CNN model similar to *Heng Cui et al* [6, 23] . The main difference that is visible with the implementation of the classifier for the 2D CNN model is within its application of linear transformation and a ReLU activation function [23].

With the implementation of the 2D CNN model for its classification accuracy, a binary cross-entropy loss was used due to its associated labels have high and low worth. With four emotion classification, categorical cross entropy loss function was used as the main loss function. Learning rate is set to 0.001 at first and is slowly lowered using a cosine annealing method as a means to allow the model to link up better and ensure there is not a significant dissimilarity with the performance of the training set. The total number of epochs is 150 and the batch size is 640 with Adam Optimiser being implemented for the model and the model was built using Python and Pytorch.

PSCP-NET

Lili Shen et al's PSCP-NET model uses a different area of focus in being able to better enhance EEG signal processing methods for BCI by exploring time-continuity and distance correlation amongst channels. This is evident from the modeling of preprocessd data for this model. This includes the use of the temporal sub-network and spatial sub-network that each focus on the capture of temporal and spatial information from the signals that are preprocessed when seen in Fig 2.8 [25].

The temporal sub-network uses sequence projection layers and temporal and convolutional kernels as a way to be able to generate a temporal feature vector. The sequence projection layers allow the model to analyse each sequence of the filtered signal individually. Temporal and convolutional kernels are used to collect the temporal information at a higher and reduce the length of the output of temporal convolutional kernels to be able to work with the spatial sub-network [25, 30].

The spatial sub-network uses spatial and convolutional kernels to process the preprocessed signals. These elements allow for similar operations that is done with the temporal sub-network to occur in the spatial sub-network. These include the capture of spatial feature representations of the preprocessed signals with 64 spatial convolutional filters [23, 25]. Convolutional filters are used to find features that contain higher spatial representations. 16 spatial convolutional layers are then used to minimise the length of the output vector to work alongside the output temporal feature vector of the temporal sub-network.

A fusion classification block is used as the main entity that concatenates the spatial feature vector and the temporal feature vector. The fusion block is then passed to a softmax classifier which is used to determine the state of the participants based on the classification values defined for the model [6, 23–25].

Lili Shen et al's model's implementation is quite similar to [6,23] with the implementation

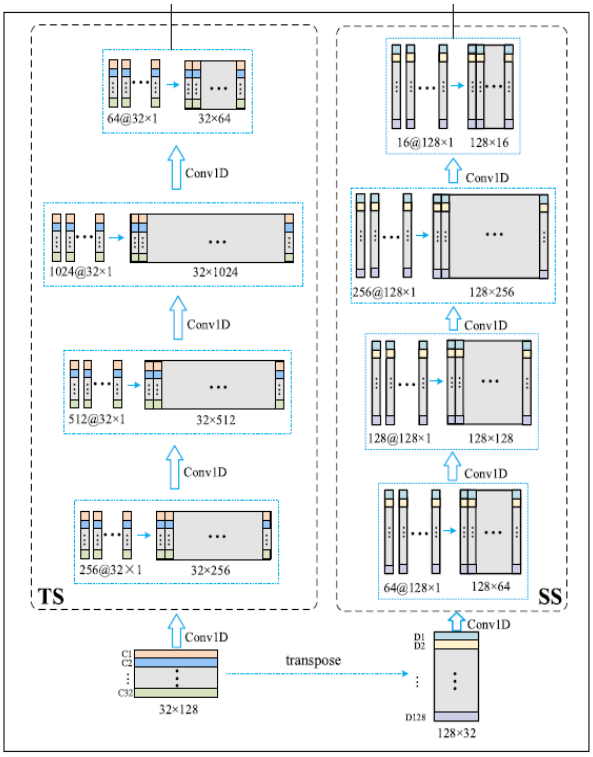


Figure 2.8: Temporal and Spatial Sub Network

of Adam optimiser with a learning rate of 0.0001 to cut back on the cross-entropy loss function. Within PSCP-NET itself, behind every convolutional layer and fully connected layer, ReLU is used as an activation function [6, 25]. In addition, a L2 regularisation measure is introduced with weight 0.0004 to reduce the likelihood of overfitting to occur with the model's data [23]. A decay rate of 0.997 and a batch size of 32 is maintained to accelerate the convergence rate [25].

3D CNN

With *Elham Salama et al*'s 3D CNN model, the selection of this model is made with the intention of focusing on the capture of multimodial and its use along with EEG signals in being able to generate more accurate results through its application with deep learning [24, 30]. This can be further shown from their 3D CNN model which employs a data modeling approach that is unique compared to the previous models mentioned here as it uses input video data that is preprocessed and EEG data that is preprocessed together [24]. This is achieved through the first 3 seconds of the EEG chunks and the face chunks being eliminated and the chunks of the same 5 seconds are combined together.

A standard convolutional model method is used for the fusion chunks with the classification values for the EEG chunks and face chunks separately generated and included into the convolutional training of the model for the fusion chunks as seen in Fig 2.9 [24]. This ensures that classifications for the fusion chunks can made without needing to be trained. The values generated from the model is then compared with the EEG signal model values and input video model values through Stacking and Bagging in Fig 2.10 [24].

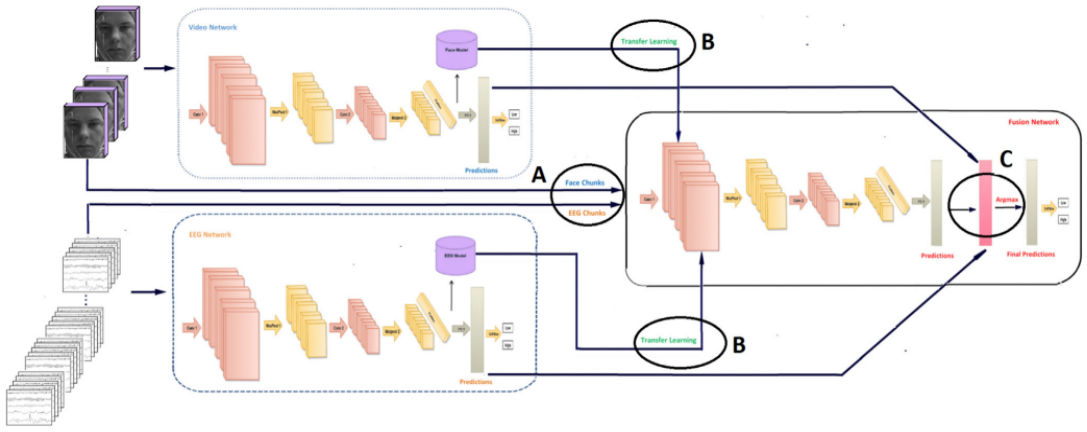


Figure 2.9: 3D CNN framework

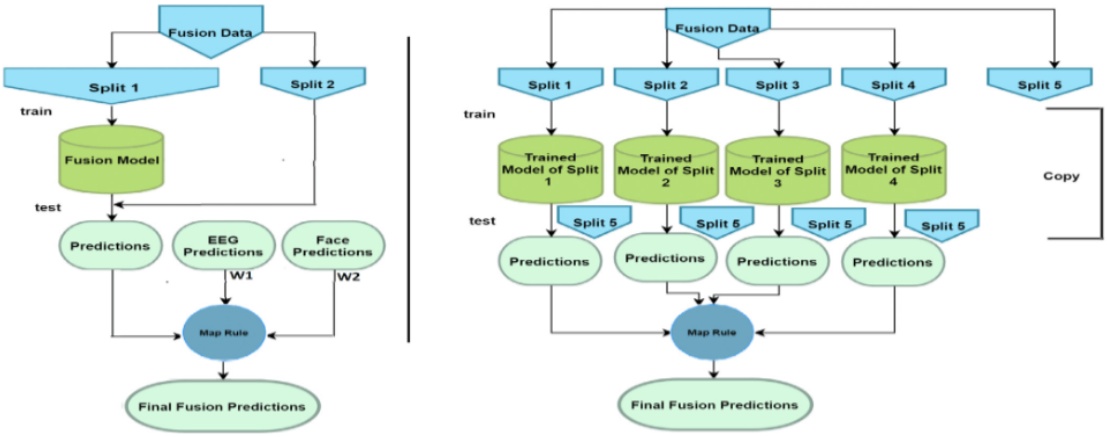


Figure 2.10: Stacking and Bagging Models

Stacking is used as a means to learn from the previous trained models and determine the strongest value from those models. This includes the use of model averaging ensemble, weighed sum ensemble and grid search ensemble leading to the final combined prediction results generated through stacking [24]. Bagging, on the other hand, uses a different approach even after using the same rule as stacking, splits the data into equal sized subsets and also ensures that duplicate values are not occurring on any of the subsets. This same methodology is applied in splitting training and testing data to generate three prediction values which are integrated with the multimodial data from the model for the final prediction values that are sent to the classifier.

The classifier process implemented uses the same softmax classifier like [6, 23, 25]. Only difference is found with the methods used in the generation of the final value that is put into the classifier with all learning methods being used and evaluated with the grid search ensemble method achieving the highest result in terms of classification [24].

In terms of the implementation of the 3D CNN model, the aforementioned resizing of collected face and EEG data as well as the bagging technique, the implementation of the model centres around the choosing of the number of features being chosen experimentally with 600, 600 and 200 features extracted for EEG, face data and fusion data respectively [24]. Upon the use of 5 fold cross validation through bagging, predictions of fusion data are generated, which are integrated with face data and EEG data using the Map Rule that picks the highest prediction from the three predictions that are computed.

GRU-MCC

The selection process that influences the modeling of the GRU-MCC model is motivated from its mission to discern processed signals with similar neural arrangement of EEG signals, allowing for local and global patterns of EEG signals and improve inter-subject transferability, allowing for its performance to improve in real-time and offline scenarios. The GRU-MCC model from *Heng Cui et al* further extends its argument with its different methods of processing which are universal and capable of finding present and hidden states of EEG signals from its focus towards the SEED and MPED datasets [7, 9].

The SEED dataset's EEG signal focuses on collecting DE features from the preprocessed signals. This lets data get selected from 5 sub bands: Delta, Theta, Alpha, Beta and Gamma [7,9]. The preprocessed signals from the MPED dataset, on the other hand, looks to find oscillary patterns within a particular time slot through Higher Order Crossings. These features are then added to a singular sample which is then used with a GRU cell.

The GRU cell of this model as shown in Fig 2.11 [7] where spatial information from the features are found through the implementation of an update and a reset gate [7]. The final activation function used before the features are sent to the classifier involves the flattening of the final activation sequence and added into a fully connected layer with a ReLU activation function to reduce feature dimension [7, 23].

The softmax classifier is applied in the final classification process of the GRU MCC model which determines the probability of a feature sequence belonging to a class [6,7,24].

With the implementation of the model on both of the datasets(SEED and MPED) used by *Heng Cui et al*, the number of classes are set to 3 and 7. An Adam optimiser is implemented to reduce total loss of the model itself while a dropout rate of 0.5 after the fully connected layer to avoid instances of overfitting [6, 7]. The size of an epoch is set to 200 with early stop and the batch size for origin and target data is 200. The learning rate value is set to 0.001, deviating from other reports such as [6, 23, 25].

ATDD-LSTM

With the framework of the ATDD-LSTM model, the main areas of influence are with finding channels that are most effective in the recognition of BCI tasks through the generation of non-linear relationships amongst channels [15, 26]. This can be shown from the use of SEED, CMEED and DEAP dataset for this model and forces the creation of domain-invariant data findings with EEG signals. This can be further shown from the

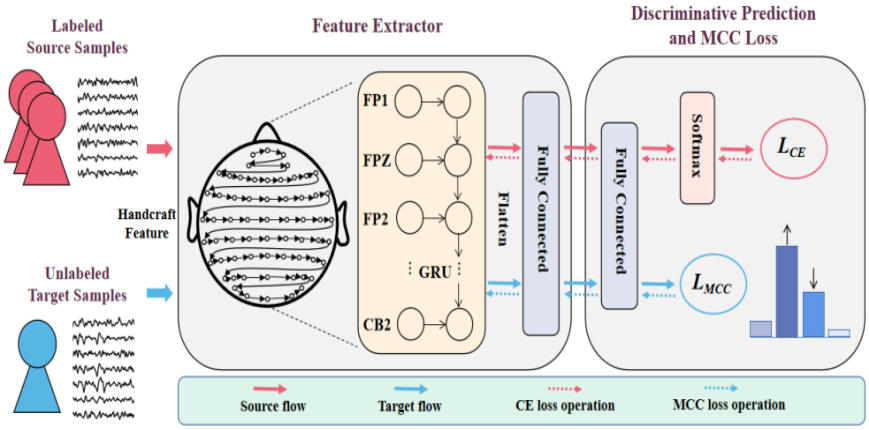


Figure 2.11: GRU,MCC Emotion Recognition Model

model's initial focus on extracting DE features on 5 sub bands, similar to [7] with the data from SEED, CMEED and DEAP. The DE features from those sub bands are then passed to a LSTM module that collects sequential features for the channel sequences. The LSTM model is made to specifically find connections and relationships amongst other frequencies within the frequency bands which were defined during preprocessing [9,20]. This allows to find dependencies between channels, resulting in being able to extract useful features from channels and frequencies without any ordering method. This along with hidden states of features and sequential features are collected together and passed to the Attention-Based Encoder-Decoder.

These findings are then used for the Attention-Based Encoder-Decoder which is responsible to construct relationships among other channels based on the sequential features that were collected previously and displayed in Fig 2.12 [9]. In addition to that, the attention mechanism within the Attention-Based Encoder-Decoder allows to integrate relationships between channels and different classes of EEG signals [9, 15, 20]. This in turn, results with the matrix that maps out the probability of a signal belonging to a class. The decoder is then applied to combine the output of the encoder and be able to show EEG representation of signals based on which class they belong to. This process allows for a weighed sum to be determined for features across all channels to be explored, ensuring classification is easier with a feature vector that corresponds to all channels.

The classification of the modeled data upon feature extraction for the ATDD-LSTM model is based around the use of the output of decoder which combines the probability of all signals belonging to a class with learnable parameters. The implementation of the model applied by *Xiaobing Du et al* relies on ensuring that non-linear relationships are maintained as shown from its selection. This is applied through the establishment of different measures applied based on the experimental strategies used to test the model based on its respective dataset.

With the SEED dataset and its cross session evaluation strategy through the use of LOSO cross validation strategy, the recognition accuracy in this instance is collected from averaging the three-fold cross validation and the standard deviation of all participants. With its subject independent evaluation, a 15 fold cross validation is applied on 15 subjects upon the division of data based on LOSO [9].

With the DEAP dataset, its subject dependent evaluation method involved the con-

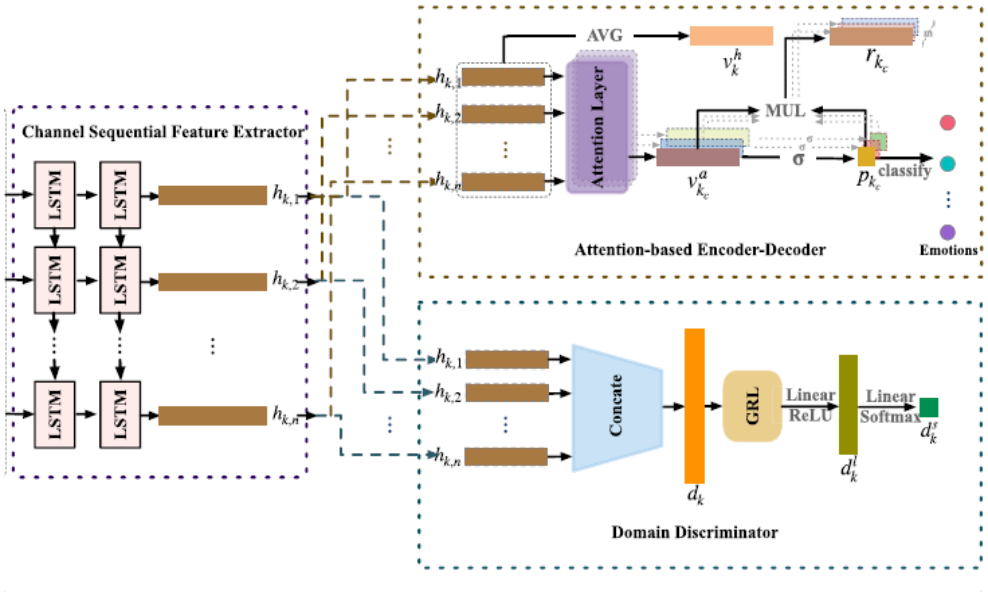


Figure 2.12: ATDD-LSTM model Feature Extractor

tinuation of LOSO with that logic being applied for all participants. Upon generating all accuracy scores, their average classification accuracy and standard deviations are used to generate the final performance of the model [9]. With the subject independent evaluation strategy, the accuracy score is generated in the same way as the SEED dataset with the implementation of LOSO with 32 folds corresponding to the 32 participants whose data is collected [9, 15].

With the CMEED dataset and its subject dependent and subject independent evaluation methods, they are both from following the same scheme used by DEAP and SEED respectively.

STRNN

The STRNN model's selected with a similar aim to that of [24] with the intention of modeling short and long-distance spatial dependencies with multimodial datasets like facial data areas and EEG signals. In addition, the model is also selected to be able to map out temporal variations within EEG signals to effectively display the relationships between EEG signals and BCI tasks. This is achieved by [7, 30] with the employment of 4 directional RNN's from specific angles and areas to collect a complete relationship graph.

Two major elements that are involved with the final projection matrix which is fed to the classifier for the STRNN involve the Spatial Recurrent Neural Network (SRNN) and the Temporal Recurrent Neural Network (TRNN) which have their own responsibilities in extracting high-level features based on their domains (spatial and temporal) from the signals it collects [6,23,30]. The SRNN uses a slicing boundary to display spatial elements which are then added into a vertex. Hidden regions of spatial features are done during traversal of the vector of spatial features using a projection that is correspondent to the direction of traversal.

The TRNN model uses the findings from the SRNN subsection to add them to each time slice from the TRNN section to create a temporal sequence, which are traversed through in a forward and backward manner through two RNNs [30]. Similar to the SRNN, projection matrices are then used to find hidden regions of temporal states within the

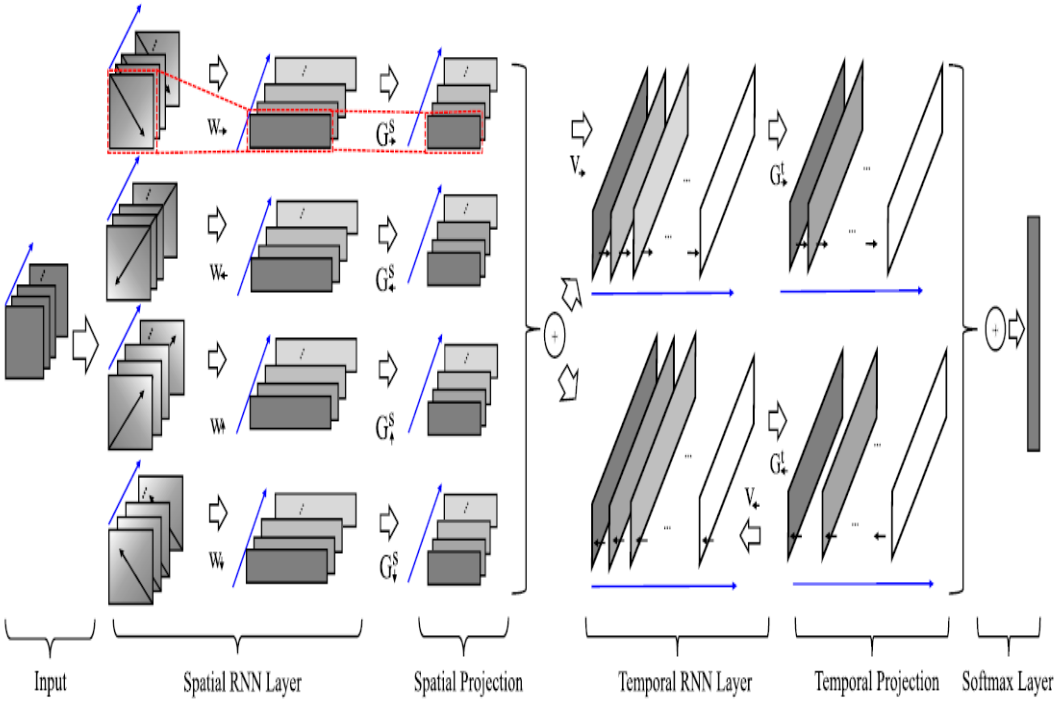


Figure 2.13: Spatial-Temporal RNN

signal [6, 30]. Once two projection matrices are generated for the forward and backward traversals, are used for the classifier to give the classifications for the features collected.

The softmax classifier is used for this model without the use of a fully connected layer where the findings from the forward and backward traversals of the temporal sequence of the TRNN layer are used in predicting the likelihood the generated output belonging to a class as evident from Fig 2.13 [30].

Tong Zhang et al's STRNN model is implemented with the basis of being able to model spatial and temporal dependencies and features together, allowing for the model to be tested on other datasets [6, 30]. This is achieved from the capture of DE descriptors for the five frequency bands of delta, theta, alpha, beta and gamma. Upon recognising a detailed continuous EEG segment, a 256-point short-time fourier transform with a non-overlapped Hanning window is applied and DE is calculated for the 5 bands [30]. Upon doing so, a slicing window of 9 seconds are generated with one step. This allows for signals with temporal dependencies to be included in the processing of EEG signals to classify BCI tasks in the STRNN model.

ALL-LSTM and WT-LSTM

The model selection process involved with the ALL-LSTM and WT-LSTM from *Huiping Jiang et al* stems from the initiation to create a model which is trained purely using sequential data as it helps enhance classification accuracy of the BCI task highlighted for this paper of emotional regularity [16]. This is further seen with the two different approaches of an ALL-LSTM and WT-LSTM model when modeling filtered EEG signals and using the subsequent trained model to classify EEG signals as shown in Fig 2.14 and Fig 2.15 [16].

The ALL LSTM model uses an LSTM layer to extract contextual related features that focus on domain features such as mean, variance and standard deviation [6, 16, 23].

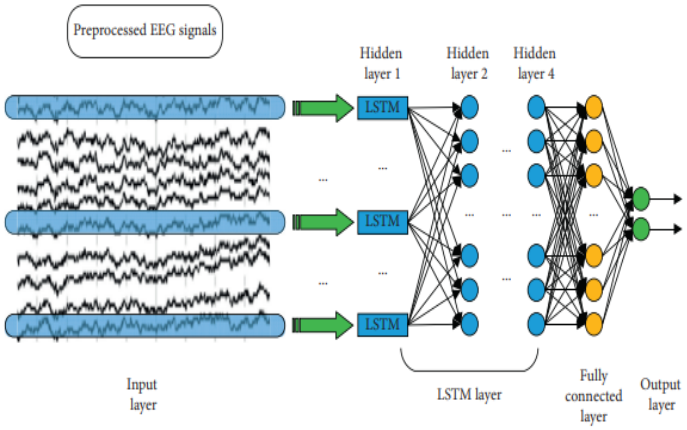


Figure 2.14: All LSTM model

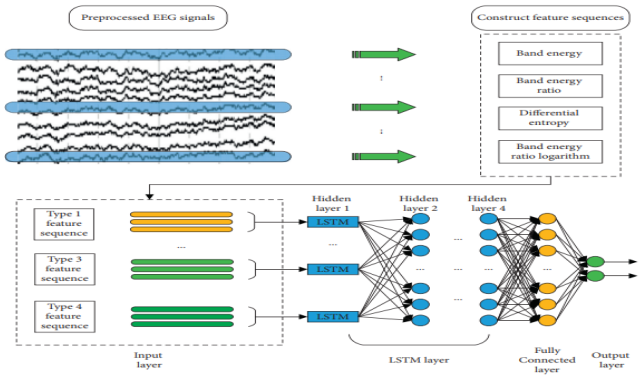


Figure 2.15: WT LSTM model

Upon the collection of those features, a fully connected layer is applied and used similarly to a general LSTM network in finding and joining features that are learned from each LSTM layer of each area of the preprocessed signal [6, 16]. These findings are then passed on to the classifier of the model.

The WT-LSTM model, on the other hand applies a different concept when modeling the data that is used to classify the preprocessed EEG signals that it collects. This involves the collection of wavelet-transform based features which include Band Energy, Band Energy Ratio, Logarithm Of Band Energy Ratio and DE are generated and used with the parameters that cater to each of the frequency bands of the preprocessed signals [16, 23]. Upon running each of the features for the frequency bands through the LSTM layers, they are concatenated and sent for classification.

The classifier for both of the models mentioned here use softmax like with the GRU-MCC model [7, 16]. The feature vectors from the ALL-LSTM model as well as the WT-LSTM model are classified separately and compared with one another in determining which model is superior in processing and classifying EEG signals [16].

In achieving the mission of implementing the model and testing it on existing datasets, *Hulping Jiang et al* use Adam Optimiser, allowing for active learning to be achieved for other parameters involved in testing [16]. Learning rate for the model is set to 0.005. Dropout is also added to ensure that overfitting does not occur with the model, with the value of 0.5 [16, 24]. Experiments were conducted to find the appropriate batch size with 64 batch size numbers being appropriate in improving the speed of the model and its use of memory [15, 16].

2.2.3 Discussion

Upon the exploration of these models, they present many strengths on processing EEG signals through BCI. These include their capabilities in being able to remove unwanted noise from multimodial datasets and being able to divide signals based on the frequency ranges necessary to process EEG signals and find features that correspond to those ranges [6, 9]. In addition, they are also capable in being able to construct connections among channels based on the spatio-temporal features they contain, leading to the creation and implementation of channel pairs that correlate to features that are the strongest [6, 9, 23].

Although these models are useful in being able to represent multiple areas of generalisation of features and finding patterns based on brain structure, they are incapable of extracting the structure between EEG signals and images. This in turn leads to the weakening of the interpretation of features in BCI tasks [5, 17]. This is mainly due to its use in analysing images that are secure and stable. Due to EEG signals following a more non-Euclidean nature [8, 18], local statistical properties change based on the spatial distribution of regions. Due to that, models that use spatial convariance matrix (SCM) using Riemannian geometry allow for a stronger focus on EEG signals that follow a non-Euclidean format and focus on generating stronger relationships with time and frequency based features for certain channels without needing to connect to all channels for classification, unlike current models that process EEG signals [5, 17, 18]. This is achieved through time and frequency segmentation using fixed short-length intervals and Chebyshev filtering, ensuring no overlap is achieved while raw EEG signals are decomposed more effectively [8, 17].

2.3 Self-Attention Based Models

Self attention Models is another state-of-the-art artificial neural network where its application focuses on language understanding. This is mainly done through its ability to find connections within different sections of input by allowing them to highlight the sections of input sequence that contain the most amount of information. Its recent implementation on image processing alongside other state-of-the art models has been a staple with processing EEG signals through BCI [15, 20]. Its graph based structure acts as a foundation of the implementation of non-euclidean based geometry which provides more in-depth understandings with time and frequency based information for the channels and the signals generated from those channels [5, 17, 18]. This in turn, allows to create better models which are better at processing EEG signals with BCI.

2.3.1 Data Preprocessing

Data Collection

Attention-Based Convolutional Recurrent Neural Network(ACRNN) For the ACRNN model, experiments were done using the DEAP and DREAMER like [6]. The implementation of the model was done using a TensorFlow framework and trained on an NVIDIA TITAN Xp pascal GPU. For the DEAP dataset that holds EOG and EEG signals. The signals with the DEAP dataset were collected with 32 participants who watched 40 emotional music videos [6, 23, 24, 26].

Other channels within that signal like the 8 channel peripheral psychological signals and EOG artifacts were disregarded during collection, the latter of which was removed using

the blind source separation technique [9, 26]. Upon collection, participants had recorded their level of arousal, valence, liking and dominance for each video they watched from 1 to 9 [6, 26]. This allowed the experiment faculty to be able to recognise which videos were influential in evoking certain which would be useful in the recognition of emotion and their classification of valence and arousal later in the model. The signals from EEG artifacts were sampled at 512 Hz and downsampled to 128 Hz [20, 26].

With the multimodal dataset of DREAMER, data from 23 participants, 14 of which were male and 9 of which were female [6, 26]. Each participant watched 18 clips where upon viewing each stimuli, participants recorded their levels of valence, arousal and dominance, similar to the data collected from the DEAP dataset. Baseline signals for each EEG signal were 4 seconds before each film clip. Before filtering the data from EEG signals, ECG signals were discarded along with ocular artifacts which were removed using linear phase FIR filters [26, 30]. To avoid instances of multiple emotions for participants and due to the length of each video, the last 180 seconds of each clip for each participant in each signal were examined further. By being able to do so, the data from the EEG signals from both DEAP and DREAMER could be able to integrate upon preprocessing since the number of channels looked at in these models were different [6, 26]. This in turn, would let new discoveries of emotions be found more easily that compliment the EEG signals from each dataset.

GraphNet Spatial Temporal Graph Convolution Network(ST-GCN) The signal collection framework involves the use of data from the Monreal Archive of Sleep Studies (MASS)-SS3 dataset. The data type that is recorded are of polysomnography (PSG) type from 62 participants with 28 male and 34 female discrepancy. The data that is collected from each recording for each participant comprises of 20 EEG channels, 2 EOG channels, 3 EMG channels and 1 ECG channel [15]. Unlike other models presented before and due to the main objective of the GraphSleepNet model, the signal collection framework does not contain any form of self-assessment as seen in [6, 7, 23, 24].

Spatial Frequency Convolutional Self-Attention Network(SFCSAN) The signal collection framework of the Spatial frequency convolutional self-attention network (SFCSAN) model uses both DEAP and DREAMER dataset [6, 20, 26]. The EEG signals that are collected from the DEAP dataset comprise of 32 participants watching 40 music videos that are a minute long. Once a video is finished, a self-assessment is done by participants for each video where they evaluate their current state based on valence, arousal, dominance and liking [6, 26, 30].

The signal collection framework of the DREAMER dataset, on the other hand has 23 participants who interact with 18 audio visual stimuli with a sampling rate of 128 Hz [6, 20]. Participants initially watch neutral movie clips in order to be able to identify their neutral emotion and act as baseline signals. The duration of each visual audio stimulus are 65 to 393 seconds long with the intention to evoke a single emotion. Similar to the DEAP dataset, each participant self-assesses their current emotion based on valence, arousal and dominance from 1 to 5 [20].

Data Filtering

ACRNN The filtering process of the ACRNN centres around the removal of the baseline signals and sliding windows [23, 26]. This can be seen in (3) where EEG signals are recorded as:

$$X_R = [X_B, X_T] \quad \epsilon \quad \mathbb{R}^{\mathbb{M}\times\mathbb{N}} \tag{3}$$

with $H\ Hz$ sampling frequency rate and T_1 representing the duration of the signal and M and N are the number of electrode notes and number of sampling points respectively [26]. Moreover, (4) is used to indicate a baseline signal with the duration time of T_2 within the number of sampling points L and X_i is used to house all of the baseline signals in the second i. This results in the mean value of baseline signal to be made as seen in (5) [26]:

$$X_B \quad \epsilon \quad \mathbb{R}^{M\times L} \tag{4}$$

$$\overline{X}_B = \frac{\sum_{i=1}^{T_2} X_i}{T_2} \tag{5}$$

with X_B acting as the mean value of baseline signal per second. The trial EEG signals with the duration T_3 is done using (6) where J represents the number of sampling points within that trial EEG signal. The baseline of the trial EEG signals are removed with the addition of X_T into the slices of X_j with a one-second non overlapping slicing window [23, 26], resulting in (7):

$$X_T \epsilon \mathbb{R}^{M\times J} \tag{6}$$

$$X_j^{'} = X_j - \overline{X}_B \tag{7}$$

The baseline signals that are removed are appended into a new matrix which is then used to segment into several temporal slices by the sliding window of 3 seconds as it achieves high classification accuracy as it allows for channel-wise attention to be used more easily[].

ST-GCN The filtering framework of the GraphSleepNet model involves the use of the band-pass filters of 0.30-100, 0.10-100 and 10-100 Hz for EEG, EOG, ECG and EMG respectively [9, 15]. This lets EEG signal based channels to be identified much more easily to process and use for classification of sleep states for participants. The final area of filtering which is done before the features are fully extracted, DE features are collected for each channel across 9 frequency band segments of : 0.5-4, 2-6, 4-8, 6-11, 8-14, 11-22, 14-31, 22-40 and 31-50 Hz [15]. This allows for the model to identify features and be able build relationships between channels that identify said features as shown in Fig 2.16 [15].

SFCSAN The filtering process involved with the SFCSAN model comprises of different processes based on the datasets used for data collection which in this case are the DEAP and DREAMER dataset.

The DEAP dataset downsamples its original data to 128 Hz [20, 24]. In order to remove EOG artifacts, the input EEG signal is decomposed with the use of a bandpass filter of 4-45 Hz is passed in order to be able to separate the data based on 4 frequencies of Beta, Gamma, Theta and Alpha [9, 20, 23]. This allows for the model to be able to

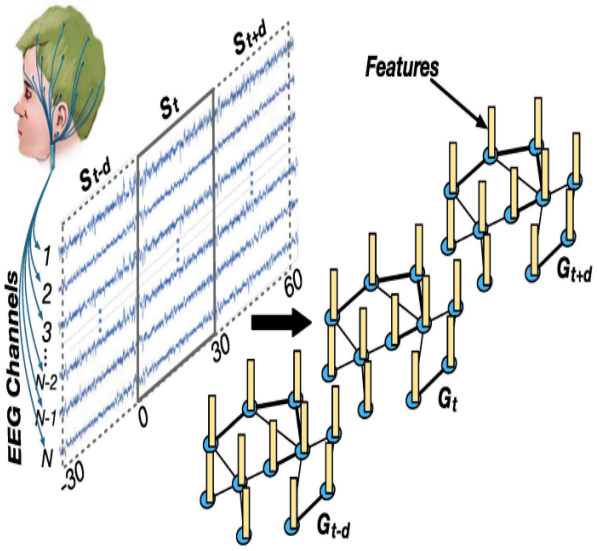


Figure 2.16: Sleep stage and connection between channel based on initial features

collect powerful features of EEG signals for each frequency during feature extraction. Eye artifacts are removed with a blind source separation technique.

With the DREAMER dataset, on the other hand, the process of filtering their EEG data consists of using a butterworth filter with a bandpass type is used to decompose the input signal into the same 4 frequency bands as DEAP [6, 20, 23]. Once applied, a three bandpass Hanning sinc linear phase FIR filters to remove a wide variety of elements within the EEG signals like muscle movements and power line noise [20].

Once all signals are divided into 4 frequency bands, the EEG signals of each frequency band are segmented through the use of a 3 second non-overlapped Hanning Window.

Training Set Creation

ACRNN The training set creation was applied through the use of a 10 fold cross validation scheme based on a subject-independent evaluation method. For both DEAP and DREAMER datasets, a 9:1 ratio was used in the division of the total samples collected for each participant on both datasets.(**800- DEAP, 1250- DREAMER**) This resulted in 720 samples of each participant and 80 samples of each participant being used for training and testing respectively and 1125 samples and 125 samples being used for training and testing with the DREAMER dataset. Upon performing 10-fold cross validation for all participants, the average of the accumulated accuracy score was used for classification.

ST-GCN Similar to [7, 26], the training set creation of the ST-GCN model involved the application of a subject independent model with a random division of its collected using the 8:2 ratio in the division between training and testing data [15]. 31 fold cross validation is then performed with the intention of evaluating the performance of the model with the BCI task of emotion regulation [15].

SFCSAN *Dongdong Li et al*, on the other hand, present a different idea evaluation method with a subject dependent method where classification is calculated each time.

For each of the participants, 972 samples are used for training and 108 are used for testing, following the 9:1 ratio in the division of training and testing data like [26].

2.3.2 Data Modeling and Testing

ACRNN

The model selection thought process used to create the ACRNN model came with intention to address an issue within all BCI tasks. Primarily, extracting features and classifying EEG signals directly through raw data without requiring any form of processing beforehand has been an issue that has not been fully explored [26]. This aspect is further practiced with the modeling practices of the ACRNN model follows the same concept that has been seen on many occasions throughout this project which is the collection and joining of the spatial and temporal features with the employment of concepts that are seen with CNN and RNN models through the Spatial Feature Extractor module and the Temporal Feature Extractor module within the ACRNN structure as seen in Fig 2.17 [26]. The one key area of difference which is prevalent with attention-based models like ACRNN is the emphasis towards finding relationships amongst other channels based on the features that are collected [15, 26] which closely follow the idea of the latest models that process and classify EEG signals for BCI [18]. They achieve through attention based mechanisms which are tied in with selecting and extracting features, falling under the same umbrella as the emerging non-Euclidean based methods which have been more successful in being able to process and learn EEG signals a lot better for BCI [26].

The Spatial Feature Extractor is used as a means to find channels that are important within the preprocessed signals which have been fed to the model [20, 26, 30]. Once all channels have been restructured, spatial features amongst all channels are found. The first process within this module involves differentiating the channels within the preprocessed EEG signals through the means of channel-wise attention with the mean pooling of each channel and their channel-wise statistics [6, 15, 26]. A gating mechanism uses 2 fully connected layers in finding channels that are the most important using the softmax classifier [6, 26]. This is particularly used to improve generalisation of the model.

The resulting vector is a channel-wise attentive features which are processed using a CNN. The CNN model then explores and gathers spatial data from the EEG signals. Exponential Linear Unit (ELU) is used as the activation function for operations rather than ReLU which has been used often [26]. MaxPool is then used to reduce parameters and collect more features. The feature vectors are then fed to the Temporal Feature Extractor for further processing [26].

The Temporal Feature Extractor module applies the findings from the Spatial Feature Extractor with a 2 layer LSTM and an extended self-attention mechanism to learn temporal based data on channel-wise features as displayed in Fig 2.18 [26]. This is achieved by combining the features with the hidden states as an activation functions for both spatial and temporal information [6, 26]. After putting them both together, an extended self-attention mechanism is implemented to find the natural importance of each EEG sample by finding more discriminative temporal based information through its comparison with findings in other areas using findings from hidden states [26].

The probability of the EEG samples are generated and multiplied with their respective hidden state as the final spatiotemporal value which is fed to the classifier for model accuracy.

The spatiotemporal feature vectors are then classified with a softmax classifier [7, 23,

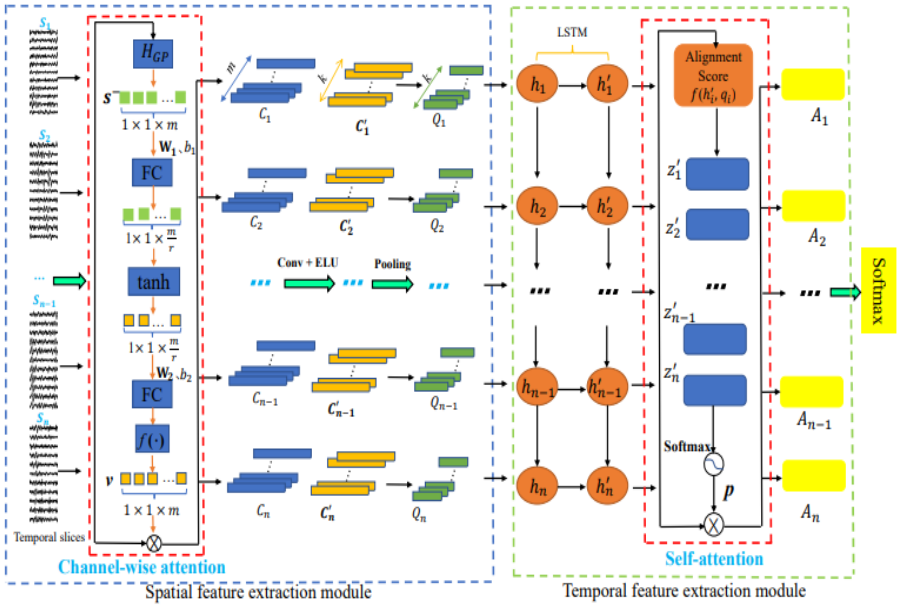


Figure 2.17: ACRNN model

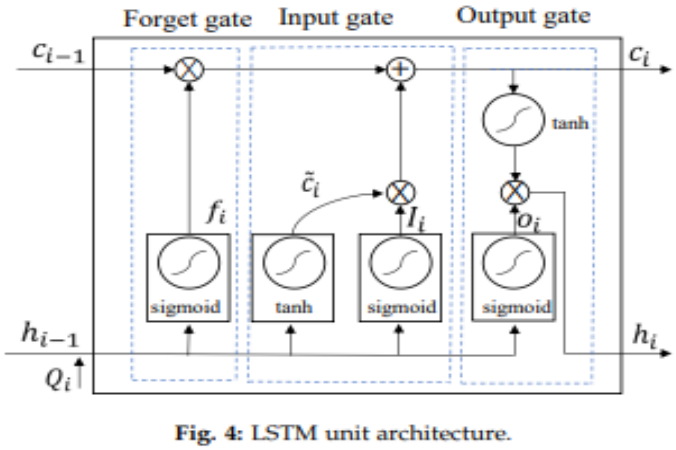


Fig. 4: LSTM unit architecture.

Figure 2.18: LSTM unit in the ACRNN model

26] in recognising EEG signal states with weight parameters.

With the implementation of the ACRNN model and testing it with existing datasets, there were many factors that were considered to ensure there was a better performance during training. This included the implementation of the Adam Optimiser to minimise cross entropy loss function, setting the model's learning rate to 0.0001 and a dropout rate of 0.5 to avoid overfitting to occur within the model [16]. Batch normalisation was also implemented to make sure the model has better performance during training.

ST-GCN

The model selection process with the ST-GCN model was mainly centred around the automatic classification of EEG signals for BCI tasks using a model with a graph structure. Moreover, it also needs to be ensured that changes between different stages of a BCI task can be addressed and resolved correctly like sleep transition in this case. The data modeling process involving the ST-GCN model applied a similar concept as highlighted here and previously with the ACRNN by *Wei Tao et al* with a focus on understanding

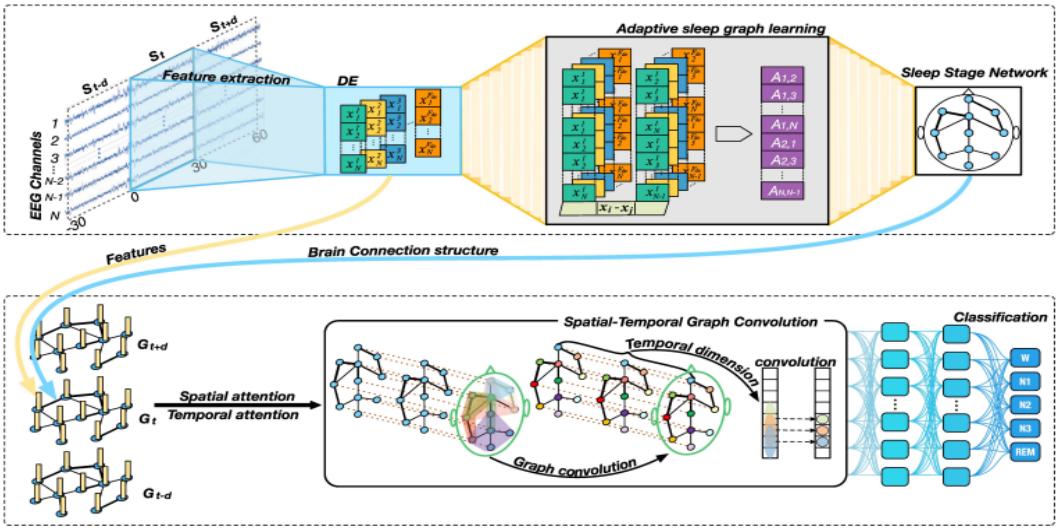


Figure 2.19: GraphSleepNet framework

the relationship between the preprocessed channels and the feature matrix of the DE features [15, 16]. The reasoning behind the modeling of the ST-GCN requiring DE features in their preliminary filtering process is to make fair comparisons of existing methods that are used for classification [15].

Preprocessed signals are added to an undirected graph along with the feature matrix in the Spatial Temporal Convolution sub-network as shown in Fig 2.19 [15]. Separations with the main EEG signal sequence are found with 30 second periods, forming a learnable adjacency matrix. This highlights the relationships that are present between each of the preprocessed signals with the feature matrix that was generated previously. This information is then passed to the sub-network of Spatial Graph Convolution [6, 15]. The Spatial Graph convolution uses the adjacency matrix to determine the mental state of the participant with the use of graph convolution that is employed to find spatiotemporal information that corroborates this view [15, 17]. This is done using methods like the Chebyshev expansion to find spatial information of adjacent channels of a particular node and extracts them. Upon finding enough spatial features for each state, temporal features are extracted using a temporal convolutional function for a particular state using ReLU [15, 23].

The spatiotemporal features which are found through Spatial Temporal Graph Convolution are sent to the Spatial Temporal Attention sub-network. These findings are then used with an attention matrix to capture and normalise attentive dynamics found from those signals. The attention matrix that is found, however, looks at finding dynamic within each of the states [15]. This allows to find temporal changes within the channels which are then fed to the classifier of the model [15, 30].

The classification of the changes made with the Spatial Temporal Attention sub-network based on the findings of the Spatial Temporal Convolution sub-network. This is achieved by finding the state of each participant based on the signals collected using the softmax classifier [6, 15, 20].

The implementation of the model in testing its evaluation with the MASS-SS3 dataset is achieved with the inclusion of several hyperparameters as displayed in Fig 2.20 where each of them serve their separate purpose. Examples include the use of the Adam Optimiser which is used to ensure efficient memory use for the model as well as a dropout rate of 0.5 to avoid overfitting [16, 26] and a learning rate of 0.001 to improve the performance of the model itself.

Hyperparameter description	Value
Sleep stage networks length T_n	5
Layer number of ST-GCN	1
Standard convolution kernels	10
Graph convolution kernels	10
Chebyshev polynomial K	3
Regularization parameter	0.001
Dropout probability	0.5
Batchsize	64
Number of training epochs	50
Learning rate	0.001
Optimizer	Adam

Figure 2.20: Hyperparameter table for ST-GCN

SFCSAN

With the model selection behind the SFCSAN model by [**?**], it is mostly devolved around the capture of more superior temporal and spatial based information in comparison to other conventional models as EEG signals are capable of clarifying neural processes and their application within BCI tasks [20,26]. The SFCSAN does exactly that with a detailed breakdown of understanding frequency based relations based on spatial relationships and contextual variations of signal sequences. This process begins from the decomposition of the signals based on the frequency bands identified earlier, DE features are extracted from each clip, resulting in a feature vector that corresponds to the number of electrodes with all 4 bands.

For EEG signals that contain continuous information and follow the Gaussian distribution through a Partial Differential Equation to learn uninterrupted information. The deviation between trial DE features extracted from the baseline before each trial data is taken s final classification features in order to remove artifacts from the processed data [20, 24]. The model's connectivity speed with other feature vectors are improved using Z score normalisation [6]. This also removes the effect of EEG features that are disordered. After these processes are conducted, the vectors are then passed to the Parallel Convolution Neural Network(PCNN).

The feature vectors for each of the frequency bands are assigned one PCNN layer to traverse through and to find more information within each layer that summarises the feature vector as presented in Fig 2.21 [20], resulting in spatial encoding. This is achieved using convolutional filtering with 3 PCNN layers. Scaled Experimental Linear Units(SELU) are used to eliminate the likelihood of exploding and/or vanishing gradient as kernels grow and feature vectors for each feature vector need to integrate with one another [20]. The vectors for each of the videos for each of the frequency bands are amplified to be better understood on a conceptual level through self attention in the Intra-Frequency Band Self-Attention sub-network.

The Intra-Frequency Band Self-Attention sub-network aims to be able to analyse each of the frequency bands' findings from the PCNN in order to develop a self-attention map that explores cross-channel EEG signal feature information and be able to combine them together [15, 17, 20]. This is achieved using the discriminative features found from the PCNN layer being applied into 2 feature spaces.

These feature spaces are used to find cross-channel information through matrix multiplication with higher values to find areas of the feature spaces that has the most amount of information. The number of channels upon the concatenation of the feature spaces are reduced and a softmax classifier [6, 20, 23] is applied to find the importance of each feature within each matrix based on their relationship with other features. The self-

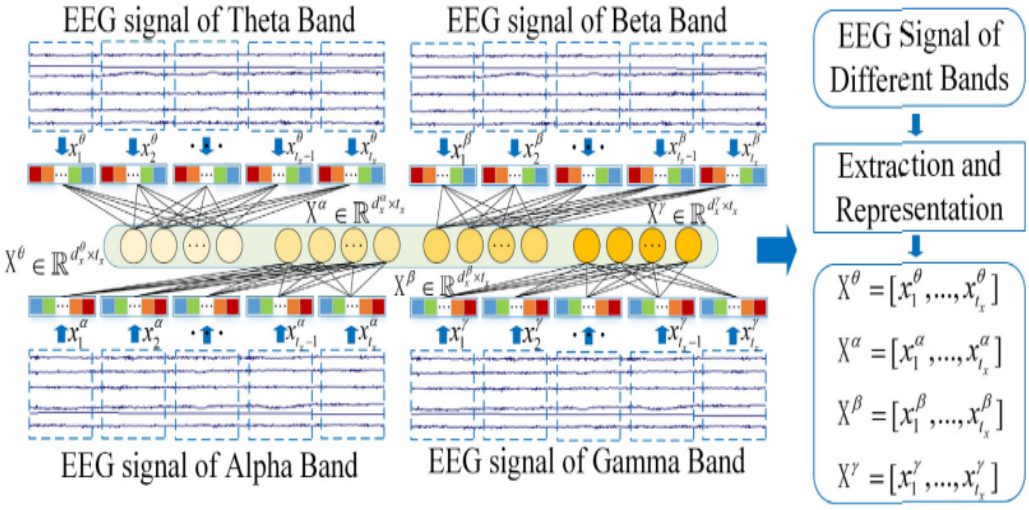


Figure 2.21: SFSCAN Feature Extraction framework

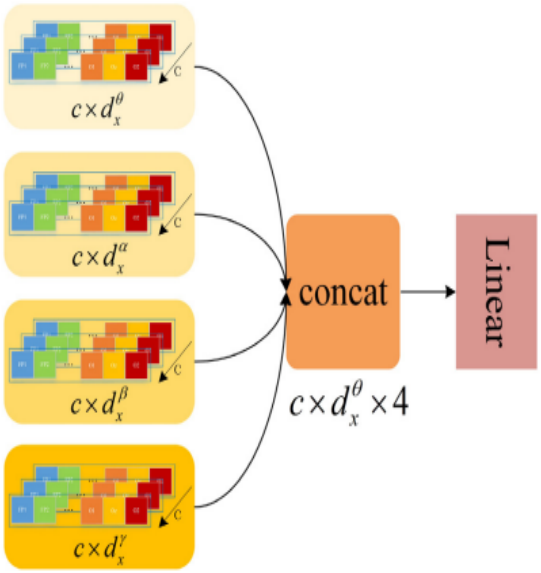


Figure 2.22: Inter-Frequency Mapping

attention map for each frequency band is passed through inter-frequency band mapping module.

The Inter-frequency band mapping flattens the self-attention map for each of the frequency bands which are concatenated to a fully connected layer with the weight matrix which is then passed to the classifier [20]. The classification of the flattened self-attention map is achieved with the feature vector with the learnable parameters of the Intra-Frequency Band Self-Attention sub-network with a softmax classifier and an argmax classifier subsequently as shown from the inter-frequency mapping process in Fig 2.22 [20].

The implementation of the SFCSAN model and how it is tested on existing datasets is mostly dependent on the performance of the training set generated previously [20]. This can be corroborated from the initial value of the learning rate being set to 0.0001 and how it is adjusted based on the performance of the training set after ten fold cross validation has occurred [26, 30]. Apart from that, the batch size is set to 128 for 50 iterations. As

an additional measure, a F1-score is also used as an indicator for classification accuracy.

2.3.3 Discussion

Self-attention models are capable of being able to process EEG signals while also being able to execute non-linear based actions when generating features with channels that have the highest level of frequencies and strongest of feature-based information through signals. This is evident from its capability with being able to decompose raw signals effectively due to their robustness to noise from being able to pick which channels are essential in being able to process EEG signals effectively [20, 26]. This can then allow for temporal and spatial based connections with other channels to be done with more ease as seen from the work of *Ziya Jia et al* where the signals are able to form temporal and spatial based connections with other channels based on the connection they have with them as well as [26] where multi-channel based feature extraction can be achieved based on connections they share with one another [15]. This, along with its structure that only selects informative features that correspond to the processing of EEG data allow it to be a strong model when it comes to EEG signal processing using the means of BCI.

They also, however, possess some areas of concern upon implementation. One such area is their overall structure upon the removal of unwanted noise and multimodial based channels which still focus on a Euclidean based formation with channels that although contain strong information still having connections when not required, resulting in a classification which is weakened [8, 15, 17]. This in turn, leads to limited application of BCI tasks that provide more in-depth information like Event-related desynchronisation(ERS/ERDS) as they are directly responsible in showing changes in neural activity and the importance of stimuli in being able to do so through rhythmic oscillations [5, 17, 18]. The inclusion of self-attention models for EEG signal processing also pose other disadvantages like an increased level of computational complications as their complicated structure results in slower training time as shown in [15] and can lead to the possibility of other problems like overfitting and requiring larger datasets as evident from [15, 20] unlike [18] and [5]. Therefore, although they are more accurate than the conventional deep learning models, they still share some drawbacks when compared to end-to-end deep learning models which are more precise and less complex due to their use of time domain to automatically capture features from the raw signals without requiring increased levels of processing [5, 20]. Since these issues are still persistent with current deep learning model frameworks that are used in the classification of BCI tasks, this leads to an issue of EEG signals collected in real-time to be put under question. TensorBasic addresses this issue through the implementation of a signal processing module which aims to process real-time EEG signals findings through time domain based information as well as frequency domain based information which is able to use raw signals as a direct source to find noises immediately and eliminate before being processed. This also acts as a pathway to the implementation of more refined preprocessing techniques which further filter data based on areas which contain more features through the analysis of EEG signals done previously. Moreover, the issue of reducing errors with more efficiency is analysed through the application of an end-to-end graph based network which analyse raw EEG signals and are able to use them in directly collecting spatiotemporal features. As a result, this allows for the model to be more accurate as it only collects data that is needed directly from source in real time and then further processes them, ensuring there is no external factor that can reduce the performance accuracy of BCI based models and also allow them to be more effective in classifying BCI tasks in the future. Due to the issues presented here, a new emerging method of collecting features directly from collected raw

EEG signals have been on the rise. One such example is the inclusion of time-domain analysis and frequency domain analysis of EEG signals. This subject matter is further explored in the following chapter where we discuss the preprocessing module used for the raw EEG data collected in the real-world experiment and is used in the removal of noise more effectively upon directly obtaining it. In addition, the use of the TensorBasic through experimental setup parameters and its implementation with a stable connection with other software components is further explored in the coming chapters. This allows us to evaluate the approaches that are used throughout all elements of the software development cycle of this project to assess the accuracy of TensorBasic.

Chapter 3

Preprocessing Module

The preprocessing module which is applied for the TensorBasic model contains three key areas of operation that are influenced by the real-time transmission of EEG signals in our real world experiment and how they are transmitted. This includes the initial transmission phase which handles the collection and segmentation of the collected raw data based on experimental factors and pass on that data to be analysed for further processing by analysing the time and frequency domain of the collected data. This allows us to be able to find external noises within the data and be able to remove them, allowing for more present and hidden features to be collected during the real world experiment.

3.1 Initial transmission

The initial transmission of the raw data collected during the real-time experiment involves the collecting data from the EEG cap every 3 seconds. This allows for the retrieved data to be analysed more effectively, allowing the model to be more accurate in the classification of the collected EEG signal. This is achieved from the use of the time library which ensures that any data that is collected within the three second mark is sent for further segmentation before being analysed based on their time and frequency domain. The first area of segmentation of the 3 second data involves ensuring that only data from the first 16 channels is used for analysis. This includes removing the other 17 columns of data which are used to capture other data which is not essential for analysis on TensorBasic. Upon the reduction of the channels and only choosing channels that are essential for the model, the second area of processing is to ensure that all of the collected data contain the same number of data points as this allows to amplify the results from filtering techniques [18]. Moreover, making sure that all collected data during real-time projects have the same number of data points allows to maintain consistency and are incredibly helpful in time domain analysis. This makes sure that there are no distortions amongst other signals when generating temporal based features amongst the all of the collected data [17].

3.2 Time Domain Analysis

Time Domain Analysis is another concept which is applied with the collected real time data from the participants as it mainly devolves around the time-locked analysis and removal of external data to allow us to explore the dynamic relationships like temporal patterns shared amongst channels and data points, allowing for the implementation of time series analysis as well [15, 17, 18]. For the TensorBasic's successful implementation in the real world experiment, one of the main areas of focus that is explored with the

collected data involves the removal of artifacts within the data that is collected in real-time that are particularly categorised with some sort of physical action. This in turn, allows to remove multimodial based data directly and access feature based data from the signal that is useful for the model's accuracy as a whole. Moreover, due to the continuous nature of the data that is collected throughout the experiment, another major area of time-domain analysis which is further explored is time-locked analysis. With the implementation of this technique, it acts as the main force of processing which allows for exploring specific features within a time period, allowing to explore other areas of understanding for EEG signals based on the time domain like ERDS graphs as done with our training data [5,17]. These signals would represent patterns that are used to recognise a particular brain activity responsible for certain BCI tasks like the movement of a car in this case from TensorBasic, making it more accurate in recognition of tasks from future signals, making it more accurate.

3.3 Frequency Domain Analysis

Frequency Domain Analysis is the study of analysing the signals from the real-time data based on the use of FFT or Fourier series [6, 15]. This allows for a more in-depth understanding of the representations of frequency domains. They also open an avenue to explore patterns amongst frequency domains and their strengths, which are incredibly useful in designing filters that cater to the collected data themselves. This idea is further explored in the frequency domain analysis operation which is conducted with the real-time signals which are collected during the real-world experiment, Based on readings from previous mock real-world experiments as well as the training data collected beforehand, the frequency domain analysis used for TensorBasic focuses on all frequency bands but with more emphasis on the patterns that are present within the area with the highest increase in amplitude values. This implies that the majority of frequency based features are located within that range and more importantly are the focal point of analysis in the design of the main filter which processes the signals collected from the time and frequency domain specifically [8, 10]. As a result, this leads to compression of signals, allowing for signal representations to be highlighted where there is increased brain activity as shown from the area of the frequency domain that contains the highest amplitude values. Moreover, this also allows for the removal of noise from the design of a filter which only collects data where fluctuations within the amplitude values are present and also understand the behavior and properties of the collected signal from each of the channels used in the collection of the EEG data [18].

3.4 Conclusion

As shown from the three main components of operation with the real-time experiment data conducted in the implementation of the TensorBasic, it analyses signals based on raw data at the beginning where the stage for the extraction of spatiotemporal features from signals are already set. This can be further shown through the initial transmission of data which is implemented on the basis of collecting data every 3 seconds to allow for enough continuous data to be collected with consistency that can be readable and used to make accurate predictions. In addition, our model also ensures that the data that is collected for further processing is only collected from frequency domains where the strongest frequency based signals are located as shown from the implementation

of frequency domain findings which allow for a more thorough understanding of signal representation and the patterns they hold from the collected signal.

Chapter 4

Model Selection

4.1 Proposed Model Description

Upon being able to extract features from channels and signals separately, the TensorBasic model is implemented based on the idea of being able to further process signals which have been collected from the time and frequency domain analysis domains respectively. As a result, not only well known features which are present within areas of strong activity of participants during training and real-timed data are collected, hidden states for each channel is generated much more easily. This is achieved from the utilisation of spatiotemporofrequency features which are collected from the preprocessed EEG signals by TensorBasic [17, 18]. This is done with tensors being the main area of analysis which focuses on highlighting links between channels based on time-based and frequency based features and uses them in the creation of channels that are not connected to all channels and more importantly allows the model to recognise which channel is the most important in terms of the amount of information it contains compared to the rest and how important it is towards successful classification values.

4.2 Baseline Implementation with result

The Baseline model which is used to employ the accuracy of the filtered and unfiltered data earlier is a CNN- Multi Layer Perception(MLP) model which also focuses on 5 classes of MI, similar to TensorBasic that handles left hand, right hand, legs, tongue and no movement of participants upon BCI. The CNN-MLP model is used as an evaluation model to compare these preprocessing methods as they are a strong foundational system used to process EEG signals. This can be shown through its ability to classify states and conditions of the human brain where there is a consistent stream of BCI within the context of EEG signals, therefore allowing it to collect meaningful and important features and temporal and spatial information [6, 23]. In addition to that, the accuracy of CNN-MLP models are purely based on the effectiveness of the processing methods it deploys for its accurate results which is essential for TensorBasic as it uses mathematical modeling techniques to classify EEG signals under the umbrella of BCI, allowing for higher processing of EEG signals and more accurate performance representations [5, 17]. The preprocessing techniques used with our collected data involve Chebyshev and Bandpass filtering using a butterworth module. Moreover, unfiltered data is also analysed to show the importance of filtering techniques in being able to precisely process and classify EEG signals [17].

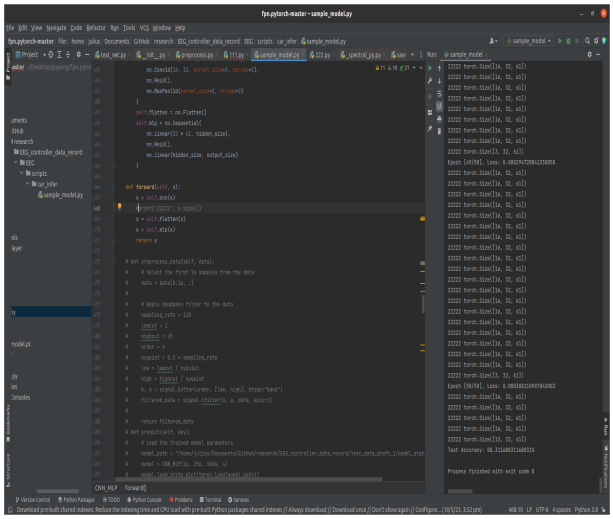


Figure 4.1: Baseline Model Evaluation with Unfiltered data

Comparison of Filters to classify MI using baseline models			
Model Type	Unfiltered	BandPass(Butterworth Module)	Chebyshev
MLP	27.2 %	88 %	62.3%

Table 4.1: The Evaluation Of The Different Filters On MLP Dataset

As seen from Table 4.1 and Fig 4.1-4.3, it is evident that the preprocessing method of Bandpass filter is the most effective in being able to process the EEG signals that are generated through our data collection process. This further shows how the bandpass filtering using a Butterworth module allows for a smoother transition with the passband and stopband which is essential in temporal based data with sliding windows which is collected as a part of the real-world experiment used with our model. In contrast, the Chebyshev filter creates a wrinkle in the passband and stopband, therefore losing important information when separating information with the way preprocessed data is segmented.

4.3 Conclusion

As seen from the performance of the bandpass-filtering method with the CNN-MLP model, it can be seen that it is more favorable in terms of use to train the model based on the data collected from participants and for generating accurate classification values during the real-world experiment.

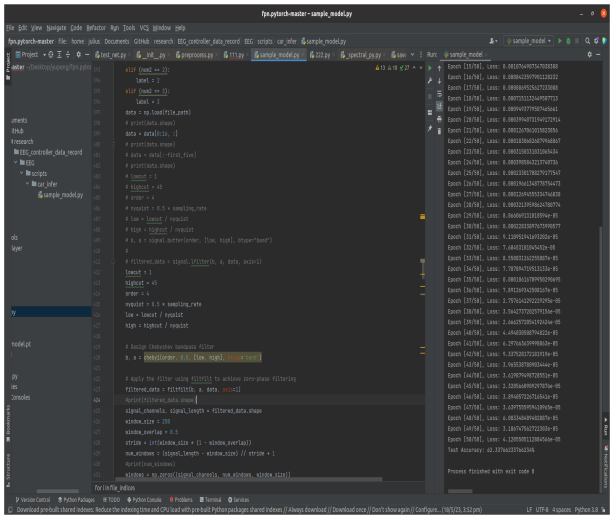


Figure 4.2: Baseline Model Evaluation with Chebyshev based data

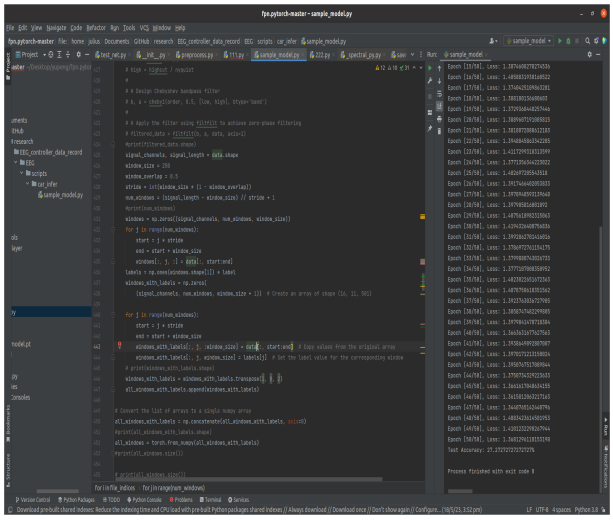


Figure 4.3: Baseline Model Evaluation with Bandpass filtered data

Chapter 5

Real World Experiment Setup

In this section, we provide a detailed description of the elements that are involved with the setup of the real world experiment we use for TensorBasic. This includes signal collection and preprocessing techniques used for our training data alongside hardware elements and used for this experiment alongside its application within the software paradigm and how they contribute in the efficient accuracy of the model.

5.1 Proposed Model And Its Workflow

TensorBasic submitted in this report aims to generate time-frequency graphs and apply a Riemann Geometry based deep learning model to predict the brain true activities quickly, accurately and with much small model size [5, 18]. Its implementation was motivated by the investigation of ensuring that the data collection procedure as well as the sections of preprocessing, feature extraction and classification were applied correctly and with efficiency. The accuracy generated by this model acts as an indicator of ensuring the flow of data is continuous within the model. The output of this model is compared with a controller in performing the same actions as the signal with the car as shown in Fig 5.1. The connection, collection and processing EEG signals is also examined to ensure they follow the intended network architecture that will be used for future works.

5.2 Experimental Setup

The experimental setup used for this experiment are divided into 2 main components, Real-Time and Offline. Both of these components are equally important in the evaluation of our proposed as the real-time system is influential in the generating the accuracy of the data and the offline paradigm is responsible in ensuring that the data that is sent to the model for further processing and classification is accurate and useful, thus enhancing its accuracy in terms of processing EEG signals.

5.2.1 Real-Time

The setup elements involved with the real-time system used for our model involves the understanding of the different areas of connection that are established in the real world experiment itself. These include the connections that are setup with the hardware and software components include the RTLS units, the EEG Cap and the Robot Car.

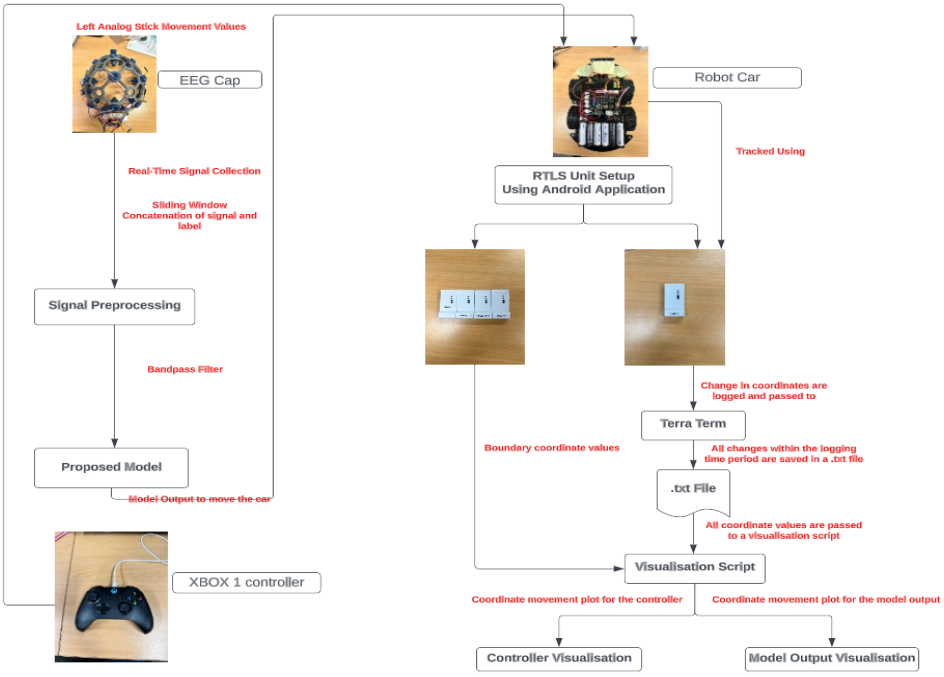


Figure 5.1: TensorBasic workflow

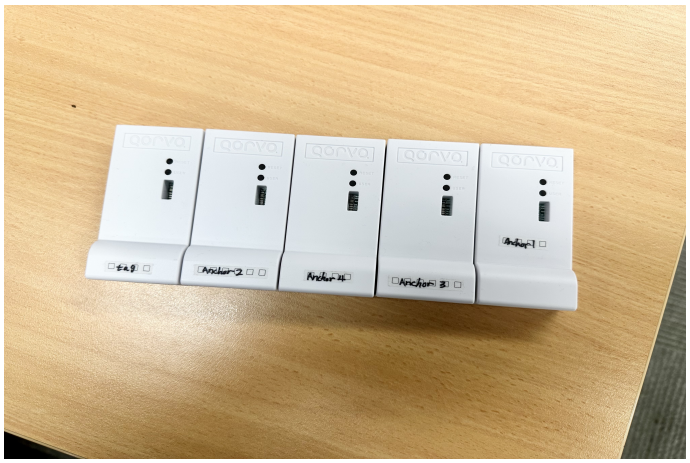


Figure 5.2: RTLS units

RTLS units

The first major set of devices that evaluates the output of our model are the RTLS units. These devices, as shown in Fig 5.2 are a part of the **Module Development & Evaluation Kit (MDEK1001)** for the **Decawave DWMC1001C**. Each unit serves its purpose in the capture of the real-time trajectory of the Robot with 4 units being situated in a square based boundary area as shown in Fig 5.3 with the units circled in red being the anchor units and the unit placed on the car being the tag RTLS unit. Other areas were considered to conduct the experiment. They were discarded, however, due to issues with data flow from the listener unit to the tag unit, disrupting the collection process of the coordinates.

Another anchor unit is placed as a passive unit to ensure it does not interfere with the readings of the anchor and tag units and is only used in to capture the changing coordinates of the tag unit and the Car through its USB connection with Tera Term and is connected through a Micro USB cable as seen in Fig 5.4.

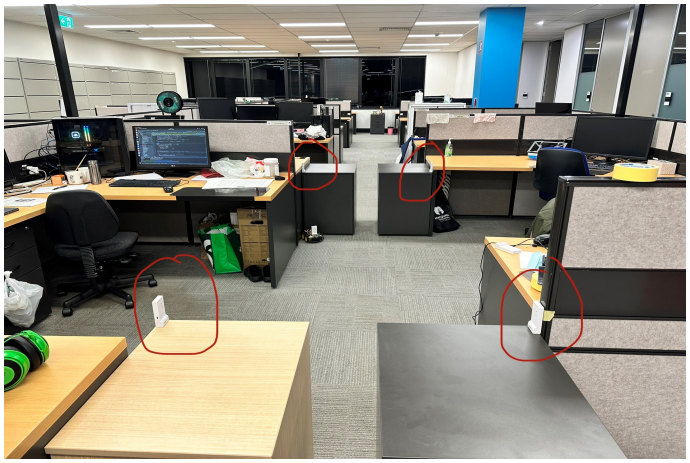


Figure 5.3: Experiment environment with square based boundary

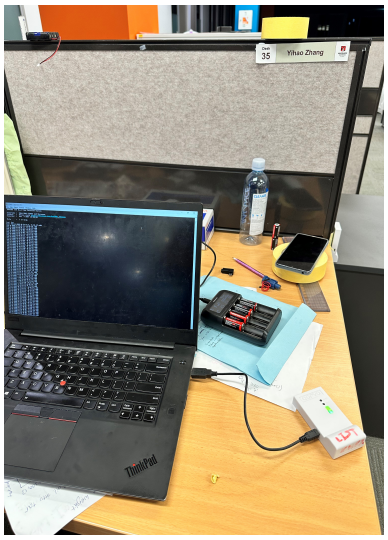


Figure 5.4: Passive Anchor Unit Connection

The efficient operation of the RTLS units upon tracking trajectories of the Robot car is achieved through its connection to each other and the OS which keeps track of the changes in the position of the tag RTLS unit. The connection of the RTLS units to one another is made through an android smartphone device and an application named **DRTLS** as seen in Fig 5.6. This application acts as a catalyst to connect the anchor RTLS units and the tag RTLS unit with one another and provides measurements based on how far apart they are with each other as displayed in Fig 5.5. As a result, this ensures that the units are operational and can be used to keep track of the trajectories of all unit in real-time as evident from Fig 5.7.

The connection of the RTLS units with the OS used to track the changing trajectories of the tag RTLS unit is mainly focused on the tag unit as it tracks the position of the Robot Car based on where it is moving when conducting the experiment. This is achieved through the use of Terra Term as seen in Fig 6.3 where it is used to begin the process of the capturing of the coordinates of the tag within the boundary area set from the anchor RTLS units.

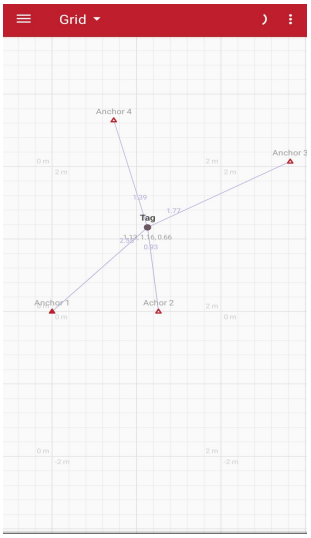


Figure 5.5: Connection of the RTLS systems to the application with measurements

Figure 5.6: Smartphone that tracks RTLS units digitally

EEG Cap

The setup of the EEG cap with the experiment conducted with TensorBasic involves the correct positioning of the electrodes that are used to collect the signals. The signals are collected based on the 10-20 international system, similar to the way signals were collected to train the model as seen in Fig 5.8. The connection of the OpenBCI headset with the OS is done from connecting the OS with the headset through the USB port of the OS. This is achieved by running the piece of the code below in the main terminal of the OS we use to conduct the experiment. Upon the successful connection with the system, the electrodes are tested for connection with the OpenBCI application and the real-time experiment is conducted.

```
sudo chmod 666 /dev/ttyUSB0
```

The software setup used to connect the EEG Cap to give EEG signals is achieved using the same method as the offline paradigm to connect the EEG Cap. This can be further evident from the following code excerpt where the upon the successful connection of the port and the Cyton Board, the data collection stream begins with a sampling rate of **128**

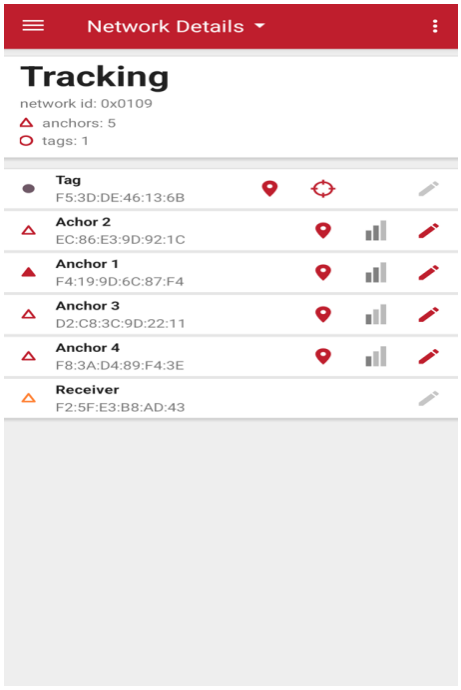


Figure 5.7: Connection of the RTLS systems to the application

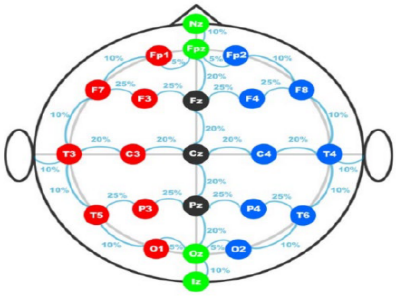


Figure 5.8: 10-20 International System

Hz [20,26].Each collected signal is filtered to a 16 channel based EEG signal like the data used to train our model and then preprocessed similarly. The preprocessed data is then sent to the model to predict the movement associated with the signal collected.

```
BoardShim.enable_dev_board_logger()
params = BrainFlowInputParams()
params.serial_port = "/dev/ttyUSB0"
board = BoardShim(2,params)
board.prepare_session()
board.start_stream()
```

Robot Car

The third and most important major device that is used to evaluate the accuracy of the model in identifying signals that convey motor imagery for the real world-experiment is the 4 Wheel Driven Bluetooth-compatible Robot Car. It is powered using 6 AA batteries and used along with a tag RTLS unit.

The use of the 4WD Robot Car in the real world experiment implementation and the evaluation of the model is accomplished through its association with other elements. These include its connection to comparative components and the OS that is used to run

the model and to track the trajectories of the car itself. The first major component involved with the evaluation of the model based on the movement of the robot car is its connection with a video game console controller. For this experiment, the controller used to track trajectories of the aforementioned RTLS tag unit and the movement of the Robot Car it is on is the Black XBOX One controller as seen in Fig 6.1. The connection of the controller to the OS is done through a wired connection with a USB cable. The setup of the controller to correspond with the Robot Car and its movements is done from its connection to a User Develpment Protocol(UDP) server and the **pygame** library. The connection to the UDP server as seen in the code excerpt below is made in order to be able to send commands of the controller for moving the Robot Car. The connection with the **pygame** library of Python is used in order for the controller to be used to move the Robot Car.

```
import pygame
import matplotlib.pyplot as plt
import os
from pyOpenBCI import OpenBCICyton
import numpy as np
import socket
import time

pygame.init()
joysticks = []
direction  = 0

localPort = 1234
UDPServerSocket = socket.socket(family=socket.AF_INET,
type=socket.SOCK_DGRAM)
```

The second area of connection the 4WD Robot Car conducts is with TensorBasic and its output. This is achieved with the use of a UDP server where the car is connected to a pipeline with a connection to the script where the model is used to control to the car(**car_control_pipe.py**) and another script, which uses the model and its predicted value to begin the real-time collection of EEG signals to use for prediction and transmission. The trained model is therefore connected through the script where an instance of our model is called and evaluated based on parameters that are set for the model as seen in the code excerpt from **model_connect_debug.py** below. These include imports of the libraries and the model that is trained and used to move the car through its prediction from real-time data.

```
import sys
import tty
import termios
import select
import socket
import curses
import os
import time
import torch
import numpy as np
import queue
import threading
from distutils.command.config import config
from brainflow.board_shim import BoardShim, BrainFlowInputParams
from new_model import EEGPredictor
```

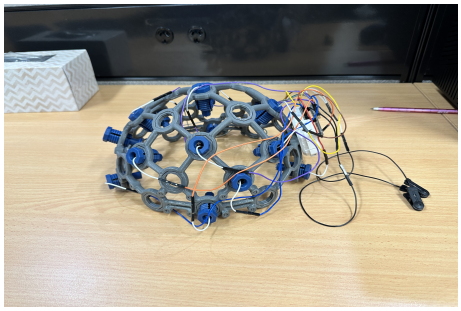


Figure 5.9: EEG Cap used throughout the experiment

```
from model import XXXPNet_Basic
write_path = "tmp/command"
wf = os.open(write_path, os.O_SYNC | os.O_CREAT | os_RDWR)
localPort = 1234
UDPServerSocket=socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
model_path = "/home/julius/Documents/GitHub/research/EEG_controller_data_rec
model = XXPNet_Basic()
model.load(load_state_dict(torch.load(model_path)))
model.eval()
eeg_predictor = EEGPredictor()
eeg_preditor.set_model(model)
data_queue = queue.Queue()
```

5.2.2 Offline

The experimental setup involved with the offline paradigm is applied to train the model using preprocessed data. The EEG cap used for collecting our training data is assembled using OpenBCI. The headset made from the OpenBCI framework contains 16 channels as seen in Fig 5.9. The distribution of these channels are done with the 10/20 system, similar to Fig 5.8 [6, 17, 23]. This framework is applied as this allows for spatial relationships to be modeled with the channels of the headset which is also applied in our non-euclidean based model [5, 18]. In addition, independent channel pairs are identified more easily when using this structure [6, 15], allowing for the signals to be evaluated much more easily. Moreover, this structure also allows the model to better detect channels that are important in terms of EEG signals. The Cyton Daisy Board is used to connect 16 channels for the EEG headset. To capture EEG signals effectively based on the parameters set for them, the software components that are used for the acquisition of accurate training data are divided based on software development cycle sections of signal collection, filtering and data modeling. The EEG Cap is connected to the offline experimental domain in the same way as the real-time domain.

Collection

With the collection of signals, Visual Studio is used alongside the Tkinter Library in the creation of the Graphical User Interface(GUI) which the participants interact with in order to obtain the EEG signals which are respective to each of the pictures presented to the participants as shown in Fig 5.10.

The GUI contains 2 groups of pictures of the labels of images displayed above. During the display of each picture in the GUI, the user is asked to imagine moving the limb that is being moved in the picture that they are looking at. After each picture is shown to a

Figure 5.10: Sample Left Hand Movement Picture from GUI for collecting training data

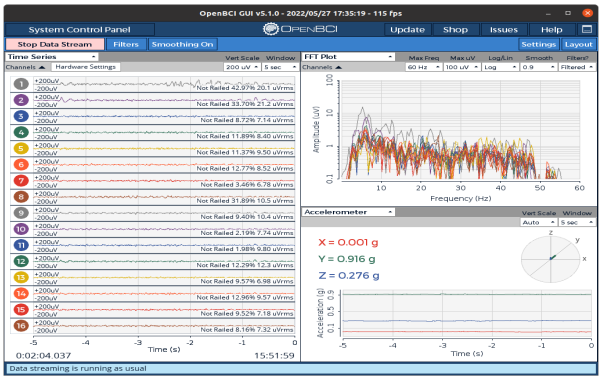


Figure 5.11: EEG Cap used throughout the experiment

participant like the one in Fig 5.10 for 20 seconds, there is a brief 5 second white screen which is used as a buffer period to separate the readings for each of the pictures. This process continues for 2 iterations as this allows for the readings to be better processed and for the model to be more accurate during preprocessing and training. The filtering of the data which is done upon the collection of signals for each of the participants are done using PyCharm. The data modeling framework is made using PyTorch [18, 23].

The experimental procedure employed with collecting training data comprise of procedures which are used to set up the hardware devices used to collect data of EEG signals and the trajectories of the Robot Car as well as the procedures that participants undertake when EEG signals are collected at the starting phase of the experiment.

OpenBCI The connection of the OpenBCI headset with the main data collection system in Visual Studio firstly involves the connection of the OpenBCI headset with the computer system. This is done with the USB based connection of the OpenBCI headset with the computer system once it is turned on. The OpenBCI application of version 5.10 is used to check if the electrodes in the headset are properly placed on the head of participants. This is determined with **"not-railed"** readings for each channel in the OpenBCI application for 15 seconds which results in running the program which collects the signals for the participants as shown in Fig 5.11.

Participants The data collection for TensorBasic's training involves the collection of signals of 14 participants with an even distribution of male and female participants(7). Participants have a mean age of 23 years. To ensure that a successful study is conducted for our model, signal is collected from 5 participants from our research team before collecting the data from 14 participants. This will help our team to identify any underlying

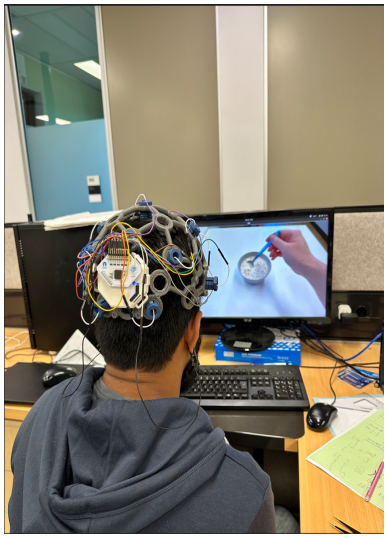


Figure 5.12: Training Data Capture

issues with all facets of the signal collection and processing before all of the data is collected from all participants. This allows for the model that is collected to be efficient and effective, allowing for the model to generate accurate results.

Participant procedure Participants will be seated in a comfortable armchair, watching a 27" monitor from a distance of approximately 2 meters. At the beginning of a trial, a fixation cross will appear on the black screen, accompanied by a short acoustic warning tone [12]. Two seconds later, an image corresponding to one of the four classes (*left hand, right hand, feet, or tongue*) will appear on the screen, signaling the participant to begin the motor imagery task [2]. They will be asked to continue the task until the image disappears from the screen (around 10 seconds), imagining the kinesthetic experience of movement while staying relaxed and avoiding any physical motion as seen in Fig 5.12.

These tasks have been commonly used in research studies because they engage different body parts associated with diverse regions of the brain, and are relatively simple for individuals to perform, making them ideal for use in studies with little to no experience in motor imagery [29, 31]. The study was composed of six runs, each with 40 trials, for a total of 60 repetitions of each mental task type, with 3 to 5-minute breaks between runs. Tasks were performed in a random order within each run to prevent adaptation. The EEG signals will be sampled at 128 Hz [2, 12, 29, 31]. The data collected from the participants acts as the data which is used to train our model in order to learn signals which correspond to the movement of the 4 areas of movement needed.

Data segmentation Upon the collection of the required data of the participants to train the model, the collected data go through a segmentation process where the data collected is reduced and collated together to be preprocessed and sent to be filtered and preprocessed.

The first element of segmentation that is conducted for the collected data of the participants involves the removal of data based on the recording paradigm as seen in Fig 5.13. First 5 seconds of the collected data which includes time periods of the fixation cross and the cue are discarded along with the break which occurs in the last 5 seconds of the collected data for each of the stimuli that is looked at. This in turn, leaves the data where motor imagery based signals are collected to be filtered.

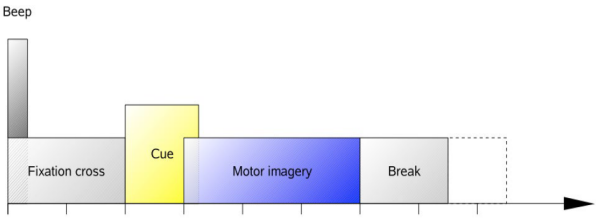


Figure 5.13: Data Collection Paradigm

Data Transmission The second area of segmentation which is practiced with the collected data for training our model involves the use of sliding window being independently applied to all files of the participants which are stored in their separate folders with an overlap rate of 50% [23, 30]. This is mainly used as an indicator to the influence of the previous window segment in the value generated for the next window segment of the data that is collected for the participants.

During this time, the label associated to each file for each participant is separated and eventually added onto a separate entity that contains the same number of windows that are generated upon the use of sliding window for each of the participants. Upon the execution of the filtering methods used for the signals collected in this experiment, the value of the labels are added towards the final filtered signal which is then sent to the model.

This in turn, allows the model to recognise which label each of the signals are associated to when the user is looking at a stimuli when EEG signals are collected. After performing sliding window on each of the files that have been collected, they are then concatenated to one another into a singular *.npy* file which is then filtered to generate findings of channels that share strong relationships and which is channel is more important than the other for classification.

Data pre-processing

The main pre-processing technique that has been used with the signals that correspond to the time period of motor imagery is the Bandpass filter. The Bandpass filter is particularly used for TensorBasic as this method allows to separate frequency bands from one another which is essential in this case. This is evident from [17,28] where frequency based calculations are required in generating effective and accurate classifications of states of participants from EEG signals. The Bandpass filter also ensures a lower insertion error which is extremely important for TensorBasic as it involves a large number of different artifacts which are used in the data collection phase as well as external noises which are bound to present within the EEG signal itself. The generation of the ERDS graph as shown in Fig 5.14 from the use of bandpass filtering paves the way for being able to find channels that have relationships with one another and which channel contains more information compared to others [5, 17, 28].

Setup of Hyperparamters The hyperparameters that are present with the preprocessing method of our model are divided into two different areas. FFT function, which is used for time-frequency relations between the channels have hyperparameters like the sampling rate with the values of **128** Hz resulting in the frequency domain graph in Fig 5.15 [2, 5, 17]. This is implemented with the intention of ensuring consistency is maintained with the data collection procedure implemented in this experiment [12, 29]. The second element of hyperparamters setup to particular values involves bandpass filtering

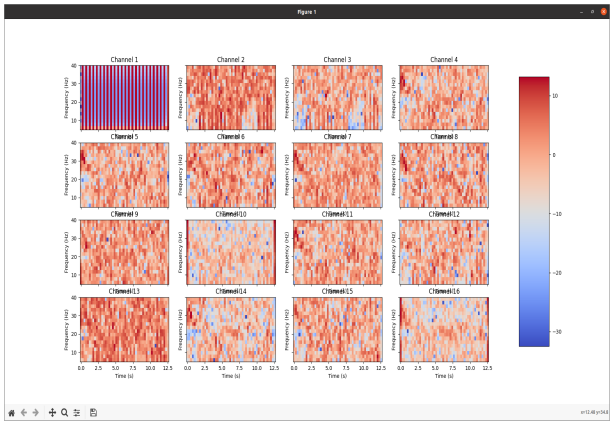


Figure 5.14: ERDS graph for one stimuli for one participant

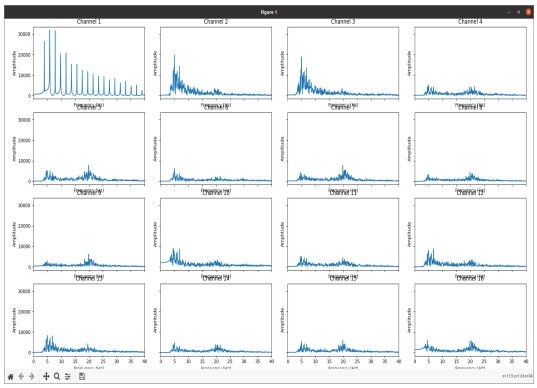


Figure 5.15: Time Frequency Graph for one stimuli viewed by a participant

which is employed with a sampling frequency of *128 Hz* and a low pass and high pass value of *4* and *45*. This is mainly done to be able to separate the signal based on frequency bands, allowing for noise reduction and identification of frequency bands that are the strongest and most active when identifying strong signals. Moreover, this also plays a strong part in the creation of the ERDS graph which highlights the channels that have the strongest correlation with changes within EEG signals based on the stimuli associated with a limb movement that is presented to the participant [5, 17, 18].

5.3 Discussion

The glaring change that is noticeable from the current experiment setup is evident from its generated end-product as it focuses on motor imagery. This was attributed to the lack of participants for this experiment as it would not allow for a definitive model that is efficient in being able to detect and predict human emotion of the participants as it would not be able to cover areas for all students. However, the processing methodologies implemented in this experiment cater to detecting and predicting brain activity [5]. Therefore, the main objective and general idea of this experiment is applying a prediction model to predict limb movement of participants based on the capture of brain waves collected using EEG signals.

In addition, the implementation of the workflow that is involved with capturing limb movement for motor imagery can be generalized into emotion detection provided we have the participants data [5, 20]. An implementation like this is necessary as the detection of limb movement is identified through EEG signals and their strength in being able to

move objects can allow to find other areas of exploration which can be helpful in future models used for emotion detection [2]. This in turn, makes the methods used for this model appropriate to be used in emotion detection in our future work.

Chapter 6

Experiment Results

This section displays the result of different preprocessing techniques with a baseline model. In addition, it also acts as an entity that provides an overview of the workflow involved with the real-world experiment framework and how the model's output are evaluated in establishing a robust BCI pipeline and identifying signals based on their appropriate characteristic. This is achieved through the generation of the results of this experiment which delve into the movements of the Robot Car through the use of the output of TensorBasic, how those movements are tracked and visualised and finally, how they compare to a controller in the movement of the Robot car, leading to the comparison of the model in identifying MI based EEG signals and their ability to project it.

6.1 Real world experiment implementation

The real-world experiment implementation with the current methodology involves the application of the real-time results of our model when a participant imagines motor movement upon instructions to move a robot car whose movements are tracked by a RTLS system. In addition, the real world experiment implementation will also observe the capabilities of a video game console controller in being able to perform the same movements as our model for the car and use its performance to evaluate TensorBasic's accuracy and efficiency. The implementation of this experiment is achieved through 4 software development cycle processes that are specifically for this experiment. These include Setup of experimental components, Movement of The Robot Car, Trajectory tracking and Movement Visualisation.

6.1.1 Movement

The movement of the Robot Car performed as a part of this experiment varies from the components used to do so. This is evident from the way a controller is used to operate the Robot Car compared to the model.

The movement of the Robot Car from the controller is done using the left analog stick of the controller as seen in Fig 6.1. The direction of the car is determined from the direction the left analog stick is pushed to move the car.**(Up- Forward, Left Turn- Left, Right Turn- Right, Down- Backward)** The default position that is established for the Robot Car if the controller's left analog stick has not been moved is stop. Upon receiving an instruction to move the car in a certain direction, the left analog stick based on that direction with different values representing different actions. This input is replicated in the terminal to run the python script file, indicating that the input from the

Figure 6.1: XBOX 1 controller with left analog stick movement

controller is sent to the Car and transmitted through its movements as seen in Fig 6.2. This is achieved from the function provided in the code excerpt below where the joystick values for the left analog stick is calculated and sent to the UDP server and transferred to the Robot Car respectively.

```
  for joystick in joysticks.values():
   axis_x = joystick.get_axis(0)
   axis_y = joystick.get_axis(1)
   if(axis_x > 0.5):
    direction = 4
    print("right")
   elif(axis_x < -0.5):
    direction = 3
    print("left")
   elif(axis_y > 0.5):
    direction = 2
    print("backward")
   elif(axis_y < -0.5):
    direction = 1
    print("forward")
    if(axis_x>= -0.9 and axis_y>= -0.9 and axis_x<= 0.9 and axis_y<= 0.9)
      direction = 0
def udp_command_sender(command):
    UDPServerSocket.sendto(str.encode(command),('192.168.117.117',
    localPort)
while True:
    controller_data()
    udp_command_sender(numbers_to_command(direction))
    time.sleep(0.01)
```

The movement of the 4WD Robot Car through the output of the model is achieved by collecting the output of the model based on the prediction of the model with the real-time data collected from the participants. The predicted value is achieved with the use of the **getKey** function in **model_connect_debug.py** as seen below. The function collects the real time data based on the settings of the model and transmits it for itss reduction in the number of channels to cater to the requirement of 16 in the function **process_key**, corresponding to the number of channels in our EEG cap used to collect data.

Once changes are made to cater to the way data is represented based on the configurations to the EEG Cap, the real-time data is then fed to the model. The data is then

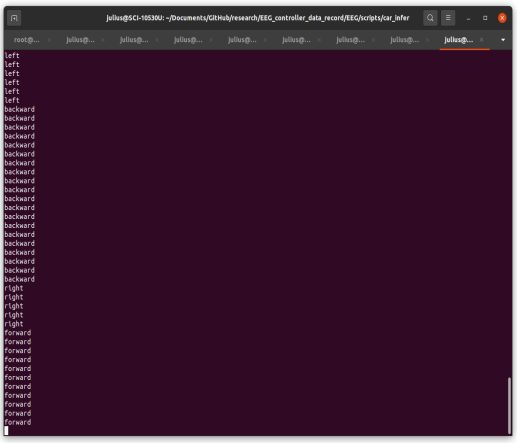


Figure 6.2: Tracking of controller movement on console

preprocessed within the same class as to where the model is defined in the model's file. The data's shape is further altered with the shape of data being altered to 384 data points before preprocessing to maintain consistency with other data points the model receives throughout the duration of the real-time experiment.

```
def process_key(key):
    key = key[1:17,:]
    if key.shape[1] != 384:
        diff = 384 - key.shape[1]
        new_arr = np.repeat(key[:,-1:], diff, axis = 1)
        key = np.concatenate((key, new_arr), axis = 1)

    return key

def getKey(queue):
     while True:
        time.sleep(0.1)
        while True:
            key = np.array(board.get_board_data())
            if key.size > 0:
                break
            time.sleep(0.1)
        new_key = process_key(key)
        data_queue.put(new_key)
```

The predicted value is then generated through the function of predict which is also defined in **new_model.py**. 5 possible values are generated on the basis of each value representing a movement imagined by a participant which translate to a type of movement from the car. Each predicted label value represents a movement of the car, similar to that of the XBOX controller.(**0 - Left Hand - w - Forward, 1 - Right Hand - s - Backward, 2 - Legs - a - Left Turn, 3 - Tongue - d - Right Turn, 4 - No Thought - S - No Movement**) These values are then translated into **ascii** based commands which are used to move the car foward as shown in the excerpt of **model_connect_debug.py** in the function **predict_key** below.

```
def predict_key(key):
    prediction = eeg_predictor.predict(key)
    #print('prediction:', prediction)
    if prediction  == 0:
```

```
        return 'w'
    elif prediction == 1:
        return 's'
    elif prediction == 2:
        return 'a'
    elif prediction == 3:
        return 'd'
    elif prediction == 4:
        return 'z'
```

To ensure that the messages from the prediction function are getting transmitted to the main script that handles the movement of the car based on **ascii** based commands, the output of **getKey** is checked and its output is encoded and the predicted value is sent to **car_control_pipe.py** using the **curses** library **send_data** function.

```
try:
    stdscr = curses.initscr()
    curses.cbreak()

    def send_data():
        while True:
            start_time = time.time()
            key = data_queue.get()
            if key is not None:
                if key == 'q':
                    print('exit')
                    break

                print(key.shape)
                prediction = predict_key(key)
                print(prediction)
                msg = str(prediction).encode('ascii')
                len_send = os.write(wf, msg)
            elapsed_time = time.time() - start_time
            delay = max(0,15 - elapsed_time)
            time.sleep(delay)


    key_thread = threading.Thread(target = getKey, args = (queue,))


    key_thread.start()
    send_data()

finally:
    curses.echo()
    curses.nocbreak()
    os.write(wf, 'exit'.encode('ascii'))
    os.close(wf)
```

Upon receiving the value using **pipe_data**, the movement instruction for the Robot Car is sent using a UDP server to the car using the excerpt from **car_control_pipe.py** below:

```
while True:
    pipe_data()
    udp_command_sender(to_command(key))
```

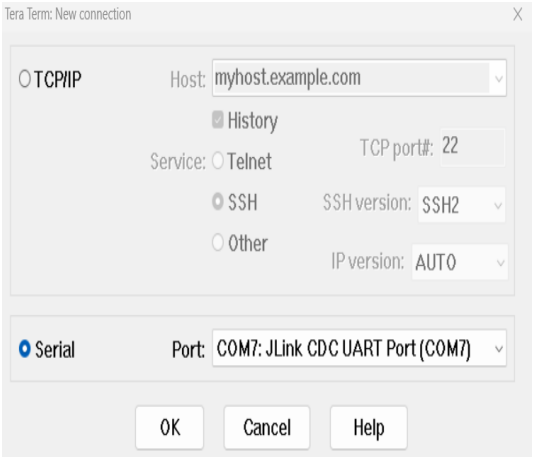


Figure 6.3: Tera Term Connection Setup

6.1.2 Tracking

The trajectories of the car and its movement within the boundary area is collected from the connection of the tag RTLS unit which is placed with the robot car and its changing coordinates are collected based on the position of the anchor units. The change in its coordinates are tracked using a PC terminal program Tera Term. The method of tracking trajectories of the Robot Car from the tag unit is the same for the controller and the output of the model which are both implemented separately in the evaluation of the trajectory tracking of the Robot Car. The passive anchor tag which is used to only collect the coordinates of the main tag unit which is placed alongside the Robot Car. The passive RTLS unit is connected with the OS which is used to run Tera Term using a micro USB cable. The unit is recognised from upon the successful download and installation of the JLink software from Segger as they provide a stable connection between an OS and the passive RTLS unit used to collect the changing coordinates of the tag unit and hence the Robot Car itself.

Upon the successful connection of the JLink software and the passive RTLS unit, the unit is then connected to the Tera Term software. This is achieved from a serial connection to the JLink port that is established in the OS and Tera Term itself as displayed in Fig 6.3. A serial port is then generated with a baud rate of **115200** and other default values.

Upon the successful generation of a serial port, Enter is pressed twice, resulting in the command interface of TeraTerm with the passive unit to occur as shown in Fig 4.7. Before starting the movement script for the car for both the controller and the output of TensorBasic, the command of **lep** is used to track the real-time coordinates of the tag RTLS unit. The changing coordinates are tracked once the changes are logged through **File then Log** which allows to store the changing coordinates of the tag unit based on the position of the other anchor tag units with are active and not passive as the Robot Car is moving as seen in Fig 6.4.

After generating the log file with all coordinates of the tag unit in comparison to the anchor tags, the log file is converted to a **.txt** file, as displayed in Fig 6.5. which is then used directly for visualisation. Once a secure connection is made between the Robot Car and its movement is conducted based on the appropriate inputs, the movements are tracked through the RTLS units that were used previously to create a boundary area and to keep track of the coordinate of the robot car within that boundary area. As the car starts moving from its starting position, Terra Term is used to log the change

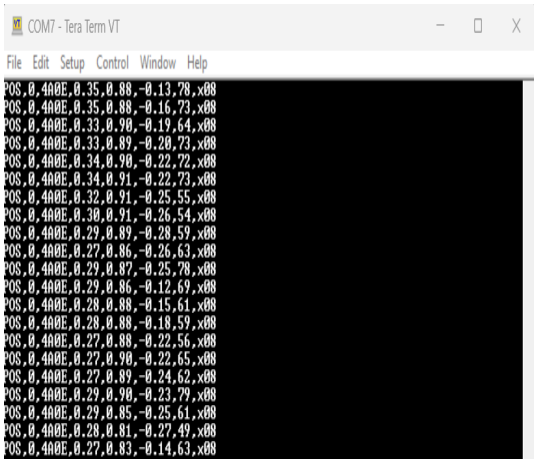


Figure 6.4: Tera Term Trajectory tracking

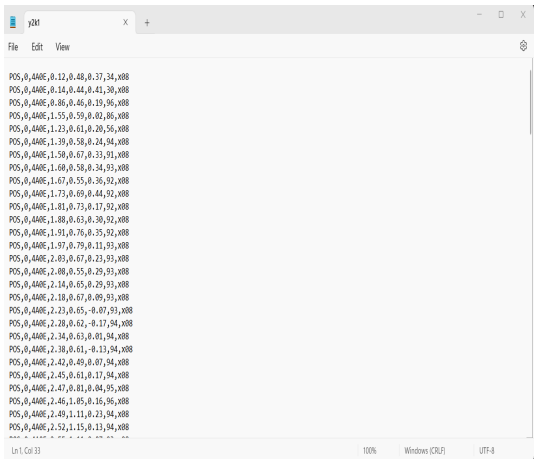


Figure 6.5: .txt file consisting of tag positions

in coordinates of the tag anchor within that boundary area as seen in Fig 4.8 with its coordinates and Fig 4. as further evidence. The logging of the coordinates continues even if the car is not moving afterwards and is only stopped after logging is stopped from the Terra Term program. Once logging is stopped, a **.txt** file containing all the coordinates is generated and is used as the input to a python script which visualises the coordinates from this text file. This methodology is applied when the Robot Car is moved using the controller and the values generated from TensorBasic. Upon completing a single session of tracking, the passive unit must change types in the mobile application before starting the process of tracking trajectories again.

6.1.3 Visualisation

The visualisation script used to plot the changing trajectories of the Robot Car that are collected from the **.txt** file consists of a 2D graph which tracks the movement of the car. This is achieved from the code excerpt below which opens the saved **.txt** file and plots the changing trajectories. Similar with trajectory tracking, the method of visualing the changing trajectories is similar for the controller and the output of TensorBasic, unlike the movement and the experimental setup connections established for both of them.

The collected file's data is then parsed for each new line and X and Y axis values of the tag for every parsed area of the data are collected, resulting in plots like the one

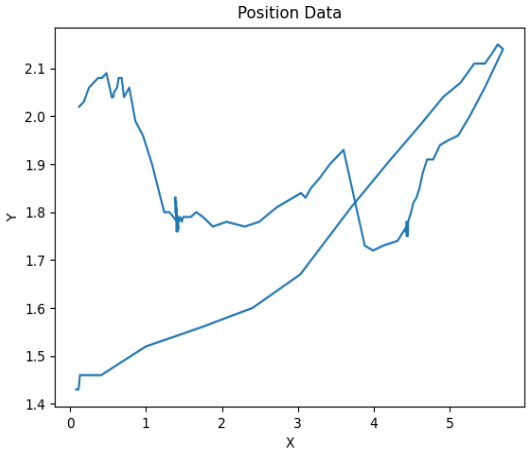


Figure 6.6: Trajectory graph for a controller going forward and backward

below in Fig 6.6 which was used as a plotted graph of the changing trajectory of the car using the controller for a forward and backward movement till the boundary area of the respective anchor RTLS units.

The script used to generate the plot is the same when it comes with tracking the movements of the robot car and the tag unit using the output of our model or the controller. These visualisations are then used as a comparative measure of the accuracy of the model in being able to process and model EEG signals based on motor imagery related tasks of people that is evoked through visual stimuli.

```
import numpy as np
import matplotlib.pyplot as plt

with open('.txt_file_path', 'r') as file:
  data = file.read()

lines = data.strip.split('\n')
pos_data = np.genfromtxt(lines, delimiter = ',', usecols = (3,4))

xs = pos_data[:, 0]
ys = pos_data[:, 1]

plt.plot(xs,ys)
plt.title('Plotting␣Changing␣Trajectories')
plt.xlabel('X_axis_value')
plt.ylabel('Y_axis_value')
plt.show()
```

6.2 Model Result

After the training and testing of TensorBasic, TensorBasic, the highest performance generated from our model is **68.7%** with the testing set.

Figure 6.7: Ground Trajectory Line In Real World Experiment

6.2.1 Evaluation Setting

The evaluation setting for the trajectories for the Robot Car and the RTLS tag unit involves the creation of a ground truth trajectory which in this case is a diagonal line from the point of origin of the square boundary area to the top right boundary of the enclosed area as seen in Fig 6.7 in real world view. When moving the Robot Car, the car is meant to go forward halfway through and is meant to go back two steps and go back forward again. After moving the Robot Car using the controller and model, the differences between the ground and moving trajectories are recorded to give a strong evaluation on the error range from both the controller and the model's output in moving the car on a certain path.

6.2.2 Comparison with controller

The first representation of the evaluation setting in the plotting and tracking of trajectories is from the XBOX controller. The controller was used to move the Robot Car forward and backward from the starting point of the graph to the top right corner twice. Upon the successful completion of this action it would be able to project a plot with the diagonal trajectory as seen in the real-world implementation of the trajectory in Fig 6.8.

The results of the projections of the Robot Car from the controller are shown in Fig 6.9 and show that the movement of the car from the controller is somewhat accurate with its movements mostly diverting more left of the trajectory at the beginning and a little more to the right halfway through till the end. Although it moves away from the projected range of the ground trajectory, it manages to reach the endpoint of the top right corner at the end of each iteration.

The results from the model's output, in similarity show the same trajectory but with plot points that are further away from the ground truth trajectory than expected. This can be further shown in Fig 6.9 where although the car is moving forward backward and forward, it is not performing those actions at the correct area which is the ground truth trajectory which is established in the graph and the real world experiment area. In

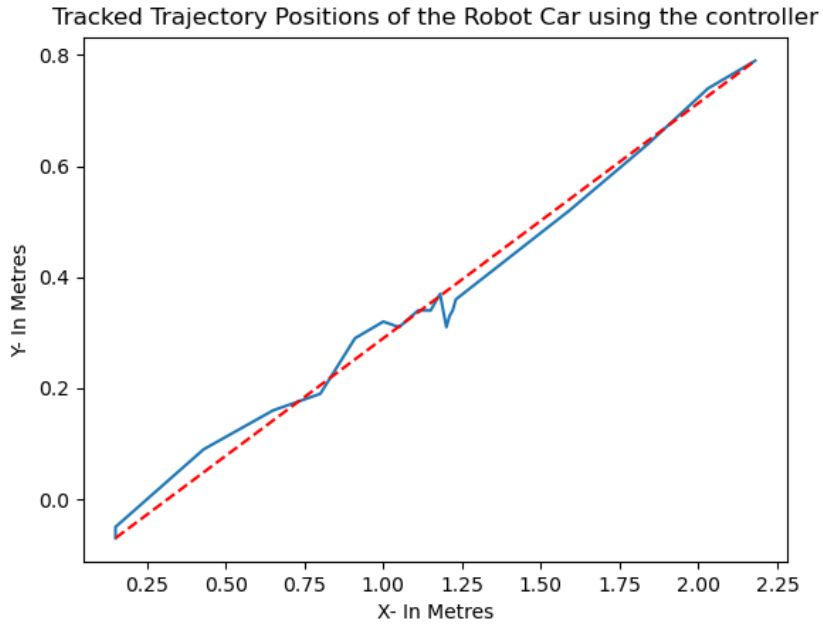


Figure 6.8: Robot Car Trajectory Graph Using XBOX Controller

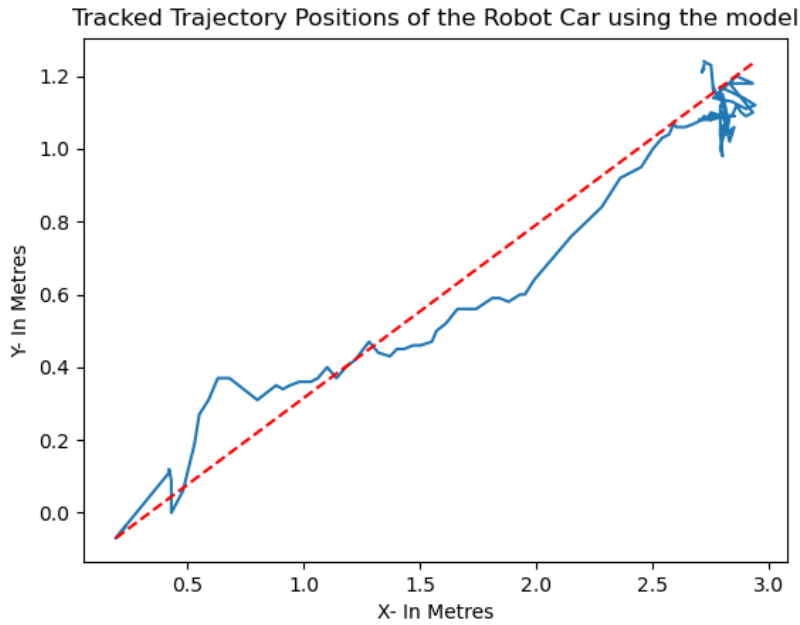


Figure 6.9: Robot Car Trajectory Graph Using XBOX Controller

addition, these movements are not performed at the correct time as well.

6.3 Discussion

In this discussion, we focus on the results and speak about the effectiveness generated by the model and their use in deploying the Robot Car to replicate a fixed trajectory point and movements that have been set beforehand. The main area of focus which encapsulates other areas of implementation is the accuracy of the model's predictions in

the movement of the car with the controller as the comparative area of focus. As shown earlier from their respective plots, it can be seen that the model's accuracy is not too far off with the accuracy of the controller when analysing the path the car takes based on their instructions. This shows the capability of non-euclidean models' with their potential in being able to receive raw EEG signals and using them in the creation of valid prediction values. Moreover, it also shows their proficiency in applying their towards successful BNI tasks like movement of external objects as demonstrated in this research paper.

6.4 Acknowledgements

This study follows the guidelines of Macquarie University and is approved with the ethical approval number: 5201800483. The implementation of the model provided was conducted with the following students: Jianchao Lu **(TensorBasic model creation and preprocessing of collected data)**, Yang Zhang **(Paper Structure)**, Yuzhe Tian **(Connection to network server and car)**, Yihao Zhang **(Software Connection to OpenBCI)** The code for the GUI of this project is available here : https://github.com/Prithivi2001/EEG-Signal-Capture-Program-GUI

Chapter 7

Future Work

The future work involved with this research study on processing EEG signals for Brain-Computer Interaction (BCI) through a software engineering approach cover a lot of areas that will require further analysis in the future with the aim of further enhancing the performance and effectiveness of EEG signal processing and classification for BCI task based state-of-the-art models.

One major aspect would be to further enhance the current end-to-end deep learning model which uses Riemmanian Geometry [5, 10, 18]. This can be further evident from the visualisations that show that although the model's output is closely accurate to the controller, it still requires further processing, particularly in the real time processing of signals to improve in the future and finding hidden spatiotemporal features from the raw data [10, 15, 26]. This can be done from the trained model being able to process raw EEG signals that are longer than specified for the current experiment parameter, which will allow for more data to be processed. Moreover, this will also allow the model to evaluate its capabilities in terms of how long a data should be to correctly predict a value based on the collection of that data.

Another area of significant focus would be on the collection of data, specifically the increase in the number of participants and a few alterations to the current EEG cap that is used to collect data. This is mainly due to the size of the cap being too small for many of the volunteering participants whose data could not be collected for training [1]. For those who had been able to connect to the cap, the channel electrodes were not connected properly and brought discomfort. After performing a revision in the hardware components used to collect the EEG signal data, the model's performance could further improve with an increase in participants whose data can be used to train the model. Moreover, by being able to have a data collection process where participant's data can be collected more easily, it can allow for the model to be tested on diverse age groups of participants, giving a better outlook on areas of implementation within the model which can improve, ranging from the data collection time period, the stimuli presented to the participants as well as the kinesthetic movements for participants to imagine while looking at a particular stimuli.

The last area of further focus would be the improvement of the performance of the deployment of the model in the real-world experiment. As seen from Fig 4.12, it can still be improved through the inclusion of another function which performs further analysis before being sent to the model for processing there. Moreover, the inclusion of hardware components like sensors and cameras can be able to receive and collect real-time differences between the planned trajectory path and the trajectory of the cars in the movement of the external objects which the model can use for a more optimal solution in the future.

Chapter 8

Conclusion

This research paper's main attempt was to better evaluate EEG signals to use in the performance of BCI tasks. These can range from emotion regulation [6, 9, 26] to sleep stage recognition [15] to Motor Imagery [17, 18]. This research paper also highlighted the significance of end-to-end non-euclidean deep learning models that capture features from the spatiotemporal frequency patterns of EEG signals. This in turn, allows for the subject of EEG signal processing to be analysed with more effectiveness so that it can be used with a framework that can get the most out of it, leading to an effective performance of the model. Based on the number of participants that was used and the application of the model to perform its required task in its real-world experiment implementation, it further shows how important graph neural networks are due to their ability to form findings based on channels that are the most important in implementation rather than generating a feature based on the whole model [18].

Chapter 9

Abbreviations

EEG	Electroencephalogram
KNN	K Nearest Neighbor
SVM	Support Vector Machines
DBN	Deep Belief Network
CNN	Convolutional Neural Network(s)
RNN	Recurrent Neural Network(s)
LSTM	Long Short Term Memory
GRU	Gated Relational Unit
ReLU	Rectified Linear Unit
GUI	Graphical User Interface
DE	Differential Entropy
FFT	Fast Fourier Transform
ACRNN	Asymmetric Convolutional Recurrent Neural Network
RF	Random Forest
PCNN	Parallel Convolutional Neural Network
MASS	Montreal Archive of Sleep Studies

Bibliography

[1] A. Abdulrahman and M. Baykara, "A comprehensive review for emotion detection based on eeg signals: Challenges, applications, and open issues." *Traitement du Signal*, vol. 38, no. 4, 2021.

[2] K. K. Ang, C. Guan, K. S. G. Chua, B. T. Ang, C. Kuah, C. Wang, K. S. Phua, Z. Y. Chin, and H. Zhang, "Clinical study of neurorehabilitation in stroke using eeg-based motor imagery brain-computer interface with robotic feedback," in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*. IEEE, 2010, pp. 5549–5552.

[3] F. Bahari and A. Janghorbani, "Eeg-based emotion recognition using recurrence plot analysis and k nearest neighbor classifier," in *2013 20th Iranian Conference on Biomedical Engineering (ICBME)*. IEEE, 2013, pp. 228–233.

[4] H. Chao and Y. Liu, "Emotion recognition from multi-channel eeg signals by exploiting the deep belief-conditional random field framework," *IEEE Access*, vol. 8, pp. 33 002–33 012, 2020.

[5] M. Congedo, A. Barachant, and R. Bhatia, "Riemannian geometry for eeg-based brain-computer interfaces; a primer and a review," *Brain-Computer Interfaces*, vol. 4, no. 3, pp. 155–174, 2017.

[6] H. Cui, A. Liu, X. Zhang, X. Chen, J. Liu, and X. Chen, "Eeg-based subject-independent emotion recognition using gated recurrent unit and minimum class confusion," *IEEE Transactions on Affective Computing*, 2022.

[7] H. Cui, A. Liu, X. Zhang, X. Chen, K. Wang, and X. Chen, "Eeg-based emotion recognition using an end-to-end regional-asymmetric convolutional neural network," *Knowledge-Based Systems*, vol. 205, p. 106243, 2020.

[8] A. Demir, T. Koike-Akino, Y. Wang, M. Haruna, and D. Erdogmus, "Eeg-gnn: Graph neural networks for classification of electroencephalogram (eeg) signals," in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, 2021, pp. 1061–1067.

[9] X. Du, C. Ma, G. Zhang, J. Li, Y.-K. Lai, G. Zhao, X. Deng, Y.-J. Liu, and H. Wang, "An efficient lstm network for emotion recognition from multichannel eeg signals," *IEEE Transactions on Affective Computing*, vol. 13, no. 3, pp. 1528–1540, 2020.

[10] X. Geng, D. Li, H. Chen, P. Yu, H. Yan, and M. Yue, "An improved feature extraction algorithms of eeg signals based on motor imagery brain-computer interface," *Alexandria Engineering Journal*, vol. 61, no. 6, pp. 4807–4820, 2022.

[11] F. P. George, I. M. Shaikat, P. S. Ferdawoos, M. Z. Parvez, and J. Uddin, "Recognition of emotional states using eeg signals based on time-frequency analysis and svm classifier." *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 9, no. 2, 2019.

[12] K.-W. Ha and J.-W. Jeong, "Motor imagery eeg classification using capsule networks," *Sensors*, vol. 19, no. 13, p. 2854, 2019.

[13] M. M. Hassan, M. G. R. Alam, M. Z. Uddin, S. Huda, A. Almogren, and G. Fortino, "Human emotion recognition using deep belief network architecture," *Information Fusion*, vol. 51, pp. 10–18, 2019.

[14] N. Jatupaiboon, S. Pan-ngum, and P. Israsena, "Emotion classification using minimal eeg channels and frequency bands," in *The 2013 10th international joint conference on Computer Science and Software Engineering (JCSSE)*. IEEE, 2013, pp. 21–24.

[15] Z. Jia, Y. Lin, J. Wang, R. Zhou, X. Ning, Y. He, and Y. Zhao, "Graphsleepnet: Adaptive spatial-temporal graph convolutional networks for sleep stage classification." in *IJCAI*, 2020, pp. 1324–1330.

[16] H. Jiang, R. Jiao, Z. Wang, T. Zhang, and L. Wu, "Construction and analysis of emotion computing model based on lstm," *Complexity*, vol. 2021, 2021.

[17] C. Ju and C. Guan, "Graph neural networks on spd manifolds for motor imagery classification: A perspective from the time-frequency analysis," *arXiv preprint arXiv:2211.02641*, 2022.

[18] ——, "Tensor-cspnet: A novel geometric deep learning framework for motor imagery classification," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[19] V. L. Kaundanya, A. Patil, and A. Panat, "Performance of k-nn classifier for emotion detection using eeg signals," in *2015 International Conference on Communications and Signal Processing (ICCSP)*. IEEE, 2015, pp. 1160–1164.

[20] D. Li, L. Xie, B. Chai, Z. Wang, and H. Yang, "Spatial-frequency convolutional self-attention network for eeg emotion recognition," *Applied Soft Computing*, vol. 122, p. 108740, 2022.

[21] G. Li, C. H. Lee, J. J. Jung, Y. C. Youn, and D. Camacho, "Deep learning for eeg data analytics: A survey," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 18, p. e5199, 2020.

[22] M. Li, H. Xu, X. Liu, and S. Lu, "Emotion recognition from multichannel eeg signals using k-nearest neighbor classification," *Technology and health care*, vol. 26, no. S1, pp. 509–519, 2018.

[23] T.-D.-T. Phan, S.-H. Kim, H.-J. Yang, and G.-S. Lee, "Eeg-based emotion recognition by convolutional neural network with multi-scale kernels," *Sensors*, vol. 21, no. 15, p. 5092, 2021.

[24] E. Salama, R. El-Khoribi, M. Shoman, and M. Wahby Shalaby, "A 3d-convolutional neural network framework with ensemble learning techniques for multi-modal emotion recognition. egypt. inform. j.(2020)."

[25] L. Shen, W. Zhao, Y. Shi, T. Qin, and B. Liu, "Parallel sequence-channel projection convolutional neural network for eeg-based emotion recognition," *IEEE Access*, vol. 8, pp. 222 966–222 976, 2020.

[26] W. Tao, C. Li, R. Song, J. Cheng, Y. Liu, F. Wan, and X. Chen, "Eeg-based emotion recognition via channel-wise attention and self attention," *IEEE Transactions on Affective Computing*, 2020.

[27] I. Wichakam and P. Vateekul, "An evaluation of feature extraction in eeg-based emotion prediction with support vector machines," in *2014 11th international joint conference on computer science and software engineering (JCSSE)*. IEEE, 2014, pp. 106–110.

[28] P. Wierzgała, D. Zapała, G. M. Wojcik, and J. Masiak, "Most popular signal processing methods in motor-imagery bci: a review and meta-analysis," *Frontiers in neuroinformatics*, vol. 12, p. 78, 2018.

[29] X. Xie, Z. L. Yu, H. Lu, Z. Gu, and Y. Li, "Motor imagery classification based on bilinear sub-manifold learning of symmetric positive-definite matrices," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 6, pp. 504–516, 2016.

[30] T. Zhang, W. Zheng, Z. Cui, Y. Zong, and Y. Li, "Spatial–temporal recurrent neural network for emotion recognition," *IEEE transactions on cybernetics*, vol. 49, no. 3, pp. 839–847, 2018.

[31] Y. Zhang, C. S. Nam, G. Zhou, J. Jin, X. Wang, and A. Cichocki, "Temporally constrained sparse group spatial patterns for motor imagery bci," *IEEE transactions on cybernetics*, vol. 49, no. 9, pp. 3322–3332, 2018.

[32] W.-L. Zheng and B.-L. Lu, "Investigating critical frequency bands and channels for eeg-based emotion recognition with deep neural networks," *IEEE Transactions on autonomous mental development*, vol. 7, no. 3, pp. 162–175, 2015.

Chapter 10

Appendix

10.1 Self Attention Models

Self attention Models is another state-of-the-art artificial neural network where its application focuses on language understanding. This is mainly done through its ability to find connections within different sections of input by allowing them to highlight the sections of input sequence that contain the most amount of information. Its recent implementation on image processing alongside other state-of-the art models shows its implementation is capable of performing just as well in capturing information from EEG signals and its different mechanisms during preprocessing and feature extraction can lead to a model which is very accurate and easy to maintain in the future [15].

10.1.1 Signal Collection

Attention-Based Convolutional Recurrent Neural Network(ACRNN)

For the ACRNN model, experiments were done using the DEAP and DREAMER like []. For the DEAP dataset that holds EOG and EEG signals. The signals with the DEAP dataset were collected with 32 participants who watched 40 emotional music videos.[] Other channels within that signal like the 8 channel peripheral psychological signals and EOG artifacts were disregarded during collection, the latter of which was removed using the blind source separation technique. Upon collection, participants had recorded their level of arousal, valence, liking and dominance for each video they watched from 1 to 9. This allowed the experiment faculty to be able to recognise which videos were influential in evoking certain which would be useful in the recognition of emotion and their classification of valence and arousal later in the model. The signals from EEG artifacts were sampled at 512 Hz and downsampled to 128 Hz.

With the multimodal dataset of DREAMER, data from 23 participants, 14 of which were male and 9 of which were female.[] Each participant watched 18 clips where upon viewing each stimuli, participants recorded their levels of valence, arousal and dominance, similar to the data collected from the DEAP dataset. Baseline signals for each EEG signal were 4 seconds before each film clip. Before filtering the data from EEG signals, ECG signals were discarded along with ocular artifacts which were removed using linear phase FIR filters. To avoid instances of multiple emotions for participants and due to the length of each video, the last 180 seconds of each clip for each participant in each signal were examined further. By being able to do so, the data from the EEG signals from both DEAP and DREAMER could be able to integrate upon preprocessing since the number of channels looked at in these models were different. This in turn, would let new

discoveries of emotions be found more easily that compliment the EEG signals from each dataset.

GraphNet Spatial Temporal Graph Convolution Network (ST-GCN)

The signal collection framework involves the use of data from the Monreal Archive of Sleep Studies (MASS)-SS3 dataset. The data type that is recorded are of polysomnography (PSG) type from 62 participants with 28 male and 34 female discrepancy. The data that is collected from each recording for each participant comprises of 20 EEG channels, 2 EOG channels, 3 EMG channels and 1 ECG channel. Unlike other models presented before and due to the main objective of the GraphSleepNet model, the signal collection framework does not contain any form of self-assessment as seen in [].

Spatial-Frequency Convolutional Self- Attention Network(SFCSAN)

The signal collection framework of the Spatial frequency convolutional self-attention network (SFCSAN) model uses both DEAP and DREAMER dataset. The EEG signals that are collected from the DEAP dataset comprise of 32 participants watching 40 music videos that are a minute long. Once a video is finished, a self-assessment is done by participants for each video where they evaluate their current state based on valence, arousal, dominance and liking.

The signal collection framework of the DREAMER dataset, on the other hand has 23 participants who interact with 18 audio visual stimuli with a sampling rate of 128 Hz.[] Participants initially watch neutral movie clips in order to be able to identify their neutral emotion and act as baseline signals. The duration of each visual audio stimulus are 65 to 393 seconds long with the intention to evoke a single emotion. Similar to the DEAP dataset, each participant self-assesses their current emotion based on valence, arousal and dominance from 1 to 5.

10.1.2 Filtering and Feature Extraction

ACRNN

The filtering process of the ACRNN centres around the removal of the baseline signals and sliding windows. This can be seen in (47) where EEG signals are recorded as:

$$X_R = [X_B, X_T] \quad \epsilon \quad \mathbb{R}^{\mathbb{M}\times\mathbb{N}} \tag{47}$$

with $H\ Hz$ sampling frequency rate and T_1 representing the duration of the signal and M and N are the number of electrode notes and number of sampling points respectively. Moreover, (48) is used to indicate a baseline signal with the duration time of T_2 within the number of sampling points L and X_i is used to house all of the baseline signals in the second i. This results in the mean value of baseline signal to be made as seen in (49):

$$X_B \quad \epsilon \quad \mathbb{R}^{M\times L} \tag{48}$$

$$\overline{X}_B = \frac{\sum_{i=1}^{T_2} X_i}{T_2} \tag{49}$$

with X_B acting as the mean value of baseline signal per second. The trial EEG signals with the duration T_3 is done using (50) where J represents the number of sampling points within that trial EEG signal. The baseline of the trial EEG signals are removed with the

addition of X_T into the slices of X_j with a one-second non overlapping slicing window, resulting in (51):

$$X_T \epsilon \mathbb{R}^{M \times J} \tag{50}$$

$$X_j^{'} = X_j - \overline{X}_B \tag{51}$$

The baseline signals that are removed are appended into a new matrix which is then used to segment into several temporal slices by the sliding window of 3 seconds as it achieves high classification accuracy as it allows for channel-wise attention to be used more easily[].

The feature extraction methods used with the ACRNN model are modeled around the collection of spatial and temporal features while also finding relationships amongst other channels based on the findings. This is achieved through the employment of channel-wise attention, CNN, RNN and a self-attention mechanism as shown in Fig 2..

Spatial Feature Extractor The purpose of the spatial feature extractor module is to differentiate important channels with the unimportant channels from the collected and preprocessed EEG signals. The first mechanism used with the feature extraction methods for this includes is the channel-wise attention which is applied to find the most important channels from the signal through the use of mean pooling to each channel and collecting channel-wise statistics as seen in (52):

$$s^{-} = [s_1^{-}, s_2^{-}, ..., s_m^{-}] \tag{52}$$

Upon completion, a gating mechanism of the channel-wise attention model is used along with 2 fully-connected layers to find channels that have the highest amount of importance when it comes to EEG signals and improve generalisation of the model using probability distribution with a softmax function in (53).

$$v = softmax(W_2 \times (tanh(W_1 \times s^{-} + b_1) + b_2)) \tag{53}$$

This results in a vector, highlighting channels based on their importance which is used to recode the channels that were fed at the beginning in (54), leading to the channel-wise attentive features being generated[].

$$c_j = v_j \times s_j \tag{54}$$

CNN is then used with the channel feature vector to explore and collect spatial information. Convolutional kernels with the same height as the number of channels are made with enough amount of width to capture temporal data as well [21]. Exponential Linear Unit (ELU) is used as the activation functions for operations instead of ReLu [5, 11]. This allows for a particular feature to be selected from a particular after convolution and activation operations are executed. Once completed, MaxPool is used for the reduction of the number of parameters and extract more features. The findings from each channel is then fed to the temporal feature extraction module for further processing.

Temporal Feature Extractor The temporal feature extractor then uses a two layer LSTM and an extended self-attention model to learn temporal data based on the channel-wise features and the corresponding spatial features from the channel-wise features. This is achieved with combining the features with the hidden states with an activation function to learn spatial and temporal information. This is demonstrated in Fig 2.

Once spatial and temporal features are put together, an extended self-attention mechanism is implemented with the aim to find more discriminative temporal information by finding the natural importance of each EEG sample. This is achieved by understanding the the similarity that every sample has from different areas using the findings from the hidden state $h_i^{'}$ being examined as seen in (55):

$$z_i' = f(h_i', q_i) = W^T \sigma(W_1 h_i' + W_2 q_i + b_i) + b \tag{55}$$

The probabilities of all EEG samples are then generated with each sample's probability being defined as:

$$p_i = \frac{exp(z_i'^T) \cdot h_i'}{\sum_{i=1}^{n} exp(z_i'^T \cdot h_i')} \tag{56}$$

The final extended spatiotemporal attentive features for each sample is generated with the multiplication value of the probability of the sample and its respective hidden state which is then passed to the classifier. This can be seen from (57):

$$A_i = p_i \times h_i^{'} \tag{57}$$

The ACRNN model's filtering and feature extractor methods contain many areas of positive impact in relation to processing EEG signals. Its usage of extended self-attention and channel-wise attention allow for it to extract discriminative features within spatial and temporal domains[]. Moreover, its ability to be able to apply channel-wise attention based concepts and self-attention allow for important channels from EEG signals to be identified. This can open the door to create channel pairs and to use them to not only extract new feature-based information but also find hidden states among other channels as well[]. This in turn, results to a model which has the strongest of capabilities in processing EEG signal data.

One issue however that can be seen here which is also consistent with other complex models that tackle the problem of extracting spatial and temporal information include the use of the LSTM methodology in learning temporal based information. Although it has been widely successful in temporal feature extraction, it is subject to taking too long to process enormous amount of data and can lead to overfitting which is very likely and in this context, would require additional resources to ensure the model works efficiently[].

ST-GCN

The filtering framework of the GraphSleepNet model involves the use of the band-pass filters of 0.30-100, 0.10-100 and 10-100 Hz for EEG, EOG, ECG and EMG respectively. This lets EEG signal based channels to be identified much more easily to process and use for classification of sleep states for participants. The final area of filtering which is done before the features are fully extracted, DE features are collected for each channel across 9 frequency band segments of : 0.5-4, 2-6, 4-8, 6-11, 8-14, 11-22, 14-31, 22-40 and 31-50 Hz[]. This allows for the model to identify features and be able build relationships between channels that identify said features.

With the feature extraction framework of the ST-GCN model by *Ziyu Jia et al* involves the understanding of the relationship between the preprocessed channels and the feature matrix. The first important element of the feature extraction module involves adding them into an undirected graph with each separation in the matrix looks at 30 seconds of

the main EEG signal sequence. This function is expressed in (58) showing the learnable adjacency matrix:

$$A_{mn} = g(x_m, xn) = \frac{exp(ReLU(w^T|x_m - xn))}{\sum_{n=1}^{N} exp(ReLU(w^T|x_m - xn))} \tag{58}$$

Spatial Temporal Graph Convolution The adjacency matrix is then passed onto the spatial temporal graph convolution network with the intention to identify a sleep stage for the second part of the feature extraction framework. This is achieved by finding spatial temporal with the use of graph convolution. Chebyshev expansion of the Laplacian matrix is used by taking in the adjacency matrix in the convolutional kernel at (60) to identify information of the neighbors of each node and extract them.

$$g_\theta *_G x = g_\theta(\mathbf{L})x = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})x \tag{60}$$

Temporal features are extracted on the basis of finding enough spatial features that have been extracted from each sleep stage network. This is shown in (60) where the temporal convolution function that takes in the spatial convolution used for a particular stage using a ReLU activation function.

$$X^(l)) = ReLU(\Phi * (ReLU(g_\theta *_G \tilde{X}^{(l-1)}))) \in \mathbb{R}^{NXC_lXT_l} \tag{60}$$

Spatial-Temporal Attention The temporal convolution function output is inputted into the final feature extraction phase of the ST-GCN to capture and normalise attentive spatial dynamics through the generation of the attention matrix as seen in (61) and (62).

$$P = V_p \cdot \sigma((X^{l-1}Z_1)Z_2(Z_3X^{l-1})^T + b_p) \tag{61}$$

$$P'_{m,n} = softmax(P_{m,n}) \tag{62}$$

The temporal attention sector, however, focuses on finding dynamic temporal information within the sleep stage networks. Temporal changes within channels are generated using similar mechanisms to the spatial attention model in (63).

$$Q = V_q \cdot \sigma(((X^{l-1})^T M_1)M_2(M_3X^{l-1}) + b_q) \tag{63}$$

The findings are then used as the main focal point of classification which is applied to in the classifier.

With the filtering and feature extraction structure that is demonstrated with the ST-GCN model by *Ziyu Jia et al*, one thing that is worth mentioning here on the strong points of the model is that it is able to allow the channels that are processed within features to be able to form temporal and spatial based connections due to the adaptive sleep graph learning aspect and spatial and temporal attention mechanism. This allows for the methods to then be used in accordance to the important channels that were defined for the signals to be able to segment and classify all features correctly during the classifier stage of the model.

One key area of contention, however, that is prevalent with this model is the data that is collected for making accurate assumptions and classifications. This can be seen with the account of *Ziyu Jia et al* that proper classifications of sleep states which is done in this model requires longer pieces of data which can also lead to a reduction in

the accuracy of the model and can lead to information redundancy[]. This suggests that the model still contains some area of argument regarding its capability to learn from and apply features that have been extracted in efficiently classifying data based on EEG signal processing[]. This in turn, would require data that is more secure and one that can meet the parameters required for classification to be done well.

SFCSAN

The filtering process involved with the SFCSAN model comprises of different processes based on the datasets used for data collection which in this case are the DEAP and DREAMER dataset.

The DEAP dataset downsamples its original data to 128 Hz. In order to remove EOG artifacts, the input EEG signal is decomposed with the use of a bandpass filter of 4-45 Hz is passed in order to be able to separate the data based on 4 frequencies of Beta, Gamma, Theta and Alpha.[] This allows for the model to be able to collect powerful features of EEG signals for each frequency during feature extraction. Eye artifacts are removed with a blind source separation technique.

With the DREAMER dataset, on the other hand, the process of filtering their EEG data consists of using a butterworth filter with a bandpass type is used to decompose the input signal into the same 4 frequency bands as DEAP. Once applied, a three bandpass Hanning sinc linear phase FIR filters to remove a wide variety of elements within the EEG signals like muscle movements and power line noise.

Once all signals are divided into 4 frequency bands, the EEG signals of each frequency band are segmented through the use of a 3 second non-overlapped Hanning Window. After that, DE features are extracted from each clip, resulting in a feature vector that corresponds to the number of electrodes with all 4 bands. For EEG signals that contain continuous information and follow the Gaussian distribution, the function f(x) = expression is substituted into the DE feature equation for further simplification.

To prevent any remaining artifacts to damage the preprocessed data, the deviation between trial DE features extracted from the baseline before each trial data is taken s final classification features. In order to improve the model's connectivity speed with other feature vectors and remove the effect of EEG features that are disordered, Z score normalisation is applied and passed on for further feature extraction afterwards.

Dondong Li et al proposes a different structure on the feature extraction framework for the SFCSAN model. This includes an initial capturing of DE features() amongst preprocessed signals which is done using PDE to learn uninterrupted information with $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ for Gaussian distribution leading to (64):

$$DE = -\int_{-\infty}^{+\infty} (\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}) log(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}) dx$$
$$= \frac{1}{2} log 2\pi e\sigma^2 \qquad (64)$$

The obtained DE features are then fed to the Z-score normalisation method which improves the classification model's connectivity speed with the other vectors for each frequency band through the removal of elements that disrupted the EEG signals upon the use of the DE feature extraction method. Four feature vectors are made for each frequency band. They contain the features that correspond to each band on a certain time period and their dimensions. These vectors are then passed on to the main model which is used to find spatial relationships corresponding to each frequency band vector to the Parallel Convolution Neural Network(PCNN).

PCNN The feature vector for each frequency band is assigned one PCNN layer which is use to traverse through it and find more information within that layer that encapsulates the feature vector. This is achieved using convolutional filtering with 3 PCNN layers with the first layer having 64 kernels that are doubled with all layers that follow. Scaled experimental Linear Units (SELU) are used to ensure there are no instances of exploding and/or vanishing gradient as the sizes of kernels grow larger and feature vector for each frequency is kept intact in order for them to integrate with one another. The feature vectors for each video are further enhanced to be understood on a conceptual level when passed to the intra-frequency band self-attention model to combine the findings for each frequency band matrix to give a final indication of the information specific to that frequency band.

Intra-Frequency band self-attention The intra-frequency band are used as a catalyst to develop a self-attention map to investigate cross-channel information and to provide its unification. This is done from the discriminative features found in the previous PCNN layer using 1 X 1 convolutional kernels being applied into 2 feature spaces. The corresponding feature spaces are used to find cross channel information using matrix multiplication with higher values allowing the model to find areas of the feature space that contain the most information. In order to reduce computational cost for classification, the number of channels used during the concatenation of the 2 feature spaces is reduced. The softmax classifier (38) is applied to generate the importance of each feature within each matrix based on their relationship with other features as seen in (65).

$$\beta_{j,i}^{b} = \frac{exp(f(U_i^b)^T g(U_j^b))}{\sum_i exp(f(U_i^b)^T g(U_j^b))} \tag{65}$$

Once the importance weights are calculated and the attention represented for each frequency band, the discriminative features from the previous PCNN layer for each frequency band are passed onto another feature space to perform matrix multiplication to generate a self-attention map for each frequency band which is then passed for the final section of feature extraction through inter-frequency band mapping.

Inter-frequency band mapping The inter-frequency band mapping model uses different regions in learning different EEG feature representations, allowing it to better represent features on a wider scale as seen in Fig. . This is evident from its application on the SFCSAN which flattens the self-attention map from all frequency bands and concatenates them into a fully-connected layer that maps all features into one vector for classification with the weight matrix of inter-frequency band mapping as shown in (66):

$$Output(f, g, h) = Concat(band^{\theta}, band^{\alpha}, band^{\beta}, band^{\gamma})W \tag{66}$$

The filtering and feature extractor areas of the SFCSAN model have a lot of strong points that contribute to a successful and strong model that can process EEG signals effectively. This can be seen from many of its capabilities such as the self-attention mechanism which captures important spatial and temporal features[], a convolutional self-attention and parallel convolutional neural network that is applied to the respective frequency bands of each signal to learn differences that are presented within each band and generate relationships between each channel. This along with an inter-frequency band mapping element allows for the model to construct dependencies across each frequency band while collection spatial and frequency based information[].

Although this model has the capacity to be able to process and classify EEG signals by finding and implementing spatiotemporal features within those signals, it does possess some elements which have an overall affect on their performance. This can be evident from Fig 2. which shows the training and testing time for each fold for the DEAP and DREAMER dataset, showing how long it takes for training the model in order to be able to classify the signals that are processed in the model. This shows that the model uses too many resources in order to be able to learn parameters of the signals that it is processing. This will only lead to further increase in training time with a larger dataset in the future.

10.1.3 Classification

ACRNN

The classification method that is used in the ACRNN model by *Wei Tao et al* involve the application of the spatiotemporal features coming in the form of a feature vector of the feature extractor with a softmax classifier (67), similar to that of(38). This is done to recognise the emotion of the participants with weight parameters.

$$P = softmax(WA + b) \tag{67}$$

GCE

The classification process by *Ziyu Jia et al* use the same classifier function of softmax like *Wei Tao et al* where the changes that are made with the temporal attention sector based on the findings of the spatial temporal graph convolution. These are then used to determine the sleep stages for each participant based on the signals collected as shown in (68).

$$Q^{'}_{m,n} = softmax(Q_{m,n}) \tag{68}$$

SFCSAN

The classification of the processed signals are done using the output of the final feature vector with concatenated features for each frequency band and a weight matrix. This is achieved with the aforementioned feature vector with the learnable parameters of the output layer with a softmax classifier and an argmax classifier as seen in (69) and (70).

$$p(y'|S) = softmax(W_O O + b_O) \tag{69}$$

$$y = argmax_{y'} p(y'|S) \tag{70}$$

10.1.4 Loss functions

With the self-attention models used to validate models that perform EEG signal processing, *Wei Tao et al* uses a cross entropy loss that is applied across the labeled data of EEG signals across all participants. This is done through (71):

$$\nu = -\sum_{i=1}^{n} \hat{Y}_i log(P_i) \tag{71}$$

Ziyu Jia et al, on the other hand, uses a loss function with minimisation to update the weight vector of the model, which ensures that the smaller the distance is between

the nodes, the larger $A_{m,n}$ is and vice versa(72). This allows for the scatterness of the adjacency matrix to be controlled as $A_{m,n}$ is directly used alongside λ, a regularised parameter that is 0.

$$\lambda_{graph-learning} = ||x_m - xn||_2^2 A_{mn} + \lambda ||A||_F^2 \tag{72}$$

This is then used in the formation of the final loss function (73) which uses the graph learning loss function (72) and cross-entropy (74) as a regularised term in order to ensure that there are no issues with the weight vector which is updated with graph learning loss function.

$$\lambda_{loss} = \lambda_{crossentropy} + \lambda_{graphlearning} \tag{73}$$

$$\lambda_{crossentropy} = -\frac{1}{L} \sum_{i=1}^{L} \sum_{r=1}^{R} y_{i,r} log y_{i,r} \tag{74}$$