12-2021

Determining States of Movement in Humans Using Minimally Processed EEG Signals and Various Classification Methods

Maurice Barnett
mauricb@g.clemson.edu

Determining States of Movement in Humans Using Minimally Processed EEG Signals and Various Classification Methods

---

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Engineering

---

by
Maurice Barnett
December 2021

Accepted by:
Dr. Yongkai Wu, Committee Chair
Dr. Eric Patterson, Committee Co-chair
Dr. Richard Brooks
Dr. Nathan Rowland

Abstract

Electroencephalography (EEG) is a non-invasive technique used in both clinical and research settings to record neuronal signaling in the brain. The location of an EEG signal as well as the frequencies at which its neuronal constituents fire correlate with behavioral tasks, including discrete states of motor activity. Due to the number of channels and fine temporal resolution of EEG, a dense, high-dimensional dataset is collected. Transcranial direct current stimulation (tDCS) is a treatment that has been suggested to improve motor functions of Parkinson's disease and chronic stroke patients when stimulation occurs during a motor task. tDCS is commonly administered without taking biofeedback such as brain state into account. Additionally, the administration of tDCS by a technician during motor tasks is a tiresome process. Machine learning and deep learning algorithms are often used to perform classification tasks on high-dimensional data, and have been successfully used to classify movement states based on EEG features. In this thesis, a program capable of performing live classification of motor state using machine learning and EEG as biofeedback is proposed. This program would allow for the development of a device that optimally administers tDCS dosage during motor tasks. This is achieved by surveying the literature for motor classification techniques based on EEG signals, recreating the methods in the surveyed literature, measuring their accuracy, and creating an application to perform online capturing and analysis of EEG recordings using the classifier with the highest accuracy to demonstrate the feasibility of real-time classification. The highest accuracy of motor classification is achieved by training a random forest on binned spectral decomposition from a normalized signal. While live classification was successfully performed, accuracy was limited by external changes to the recording environment, skewing the input to the trained model.

Dedication

This thesis is dedicated to my parents and my partner Danika for their unconditional love and support.

Acknowledgments

I would like to thank Dr. Rowland and everyone in his lab for sharing their expertise with me, contributing the dataset, and providing all resources necessary. I would like to thank Dr. Wu for chairing my committee and Dr. Brooks for being a part of my committee. I would like to thank Dr. Russell for first encouraging me to join the graduate program. I would like to thank Dr. Zordan for introducing me to research during my undergraduate studies. Lastly, I would like to thank Dr. Patterson for continuing my undergraduate research, providing me the opportunity for this project, and being an invaluable mentor throughout my time at Clemson University.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Approximately 4% of the U.S. population is affected by movement and neurodegenerative disorders such as chronic stroke and Parkinson's disease. It is projected that by 2030 there will be 1.2 million people living with Parkinson's disease in the United States [37]. Additionally, there are approximately 4 million people in the United States that are affected by chronic stroke [11]. These conditions considerably affect patients' ability to move. Transcranial direct current stimulation (tDCS) is a treatment that has been suggested to improve symptoms of Parkinson's disease and chronic stroke patients when stimulation occurs during a movement task [35]. Consistently determining when administration should occur is a manual and tiresome process. Additionally, biofeedback is often not accounted for when determining when tDCS administration should occur. Therefore, a program capable of discerning movement state from biofeedback is desired. Accurately discerning states of movement from states of rest using biofeedback would lead to the development of a device that administers tDCS at optimal periods of time.

One potential form of biofeedback is electroencephalography (EEG). EEG is a non-invasive technique used for recording neuronal signals in real time from the brain. EEG is commonly used to diagnose and monitor epilepsy due to the fact that seizure activity has different frequency characteristics than non-seizure activity. Due to its portability, behavioral activities such as various movement states may be performed by a subject while recording EEG signals, which can then be analyzed

to determine the sources and frequencies driving EEG changes during various behaviors. EEG is often used in research to develop brain machine interfaces that perform tasks such as 2D and 3D cursor control [52] [38]. Its historical use in brain machine interfaces makes it a promising avenue to perform motor classification on. However, EEG data acquisition is comprised of several channels at fine temporal resolution, which leads to a dense, high-dimensional dataset, making it difficult to perform analysis on.

While EEG provides a method of receiving biofeedback, there still must be a method of classifying movement state based on EEG observations. Determining the movement state of an individual can be simplified to a binary classification problem: states of movement compared to states of rest. Machine learning is a vast field of study comprised of algorithms that perform regression and classification tasks well. Additionally these algorithms are computationally effective against high-dimensional datasets, and often have well supported implementations in several programming languages. While there are several methods of performing classification, these factors make machine learning a well-suited starting point for problem.

In order to develop a program capable of discerning movement state using biofeedback, a survey of established literature is conducted to identify different methods of motor classification using EEG signals. Then, the classification algorithms explored in the literature are applied to a prerecorded dataset of EEG recordings of subjects (healthy control, Parkinson's disease, and chronic stroke) performing a virtual reality-guided movement task comprised of periods of both movement and rest. The resulting classification algorithm accuracies are compared to those found in the literature. Then, the most accurate classification method is used to perform live classification on a healthy subject wearing an EEG recording apparatus.

Chapter 2

Background

2.1 Electroencephalography

Electroencephalography (EEG) uses an array of electrodes placed on a subject's scalp to record signals of voltage potential changes from within the brain. The continuous signals are filtered, amplified, and in modern systems converted to digital signals for further processing and storage. Hans Berger, a psychiatrist, first published this technique in 1929 [5]. Berger used EEG to identify two regular wave patterns: one large and one small. These wave patterns would later be termed alpha and beta waves, respectively [6]. The peak-to-peak distance in these waves represents a potential change of 150-200μV. These deflections in potential, initially termed "waves," represent the averaged, coordinated activity of neurons firing synchronously in bursts. These waves are also commonly called "oscillations." Berger classified alpha and beta waves to have respective frequency ranges of 11-15 Hz and 20-32 Hz. Subsequently, in 1936, delta (0 to 3.5 Hz) and theta (4 to 7.5 Hz) waves were discovered by Walter [50]. In 1938, the gamma wave was first described by Jaspers and Andrews with a frequency of 50 Hz found over the sensorimotor cortex [29]. Over the next several decades, the 10 Hz alpha wave was observed to be the most dominant frequency in the conscious human brain. It was also established that the other waves are based on harmonics (a multiple of some base frequency) of the alpha wave [32]. A table depicting center frequencies and frequency ranges of each band is shown in Table 2.1.

Frequency Band	Center (Hz)	Range (Hz)
Delta	2.5	1-3
Theta	5	4-7
Alpha	10	8-12
Beta	20	13-30
Gamma	40	31-50

Table 2.1: Table depicting the ranges of EEG brain wave frequencies [47]. Each brain wave center frequency is a harmonic of the alpha wave (10 Hz) [32].

Since the initial description of each frequency band, the bands have been studied in association with various behaviors or functions within brain signaling. For example, delta oscillations occur most prominently during slow wave sleep [23]. They are also detected when a subject is attempting to detect a target or go-stimulus in a set of distractors or no-go stimuli, or during other rapid, decision-making processes. They are also associated with motivation and reward mechanisms. Theta oscillations appear when a subject performs memory tasks, and are most dominant in the hippocampus. They have also been associated with emotional arousal and fear conditioning [8]. Alpha oscillations are generally associated with attentional processes, sensory stimulation, and knowledge-system access [23]. Beta oscillations are generally associated with activation in the sensorimotor cortex, and they are hypothesized to help maintain steady states [14]. Gamma oscillations are synchronized during broad cortical activation, attentive processing of information, active maintenance of memory contents, and conscious perception [23]. Gamma oscillations can exhibit high frequencies (up to 200 Hz) and are often split into multiple frequency bands (e.g., low, mid, high, or a combined broadband gamma) due to the large frequency band width [46].

Each electrode in an EEG montage is termed a "channel," and each channel represents the averaged signal of 10^5 - 10^6 neurons [34]. In the frequency domain, each frequency band represents a synchronization of clustered neurons. An increase or decrease of neuronal synchronization in a frequency band in response to an event or stimulus is known as either an event-related synchronization (ERS) or event-related desynchronization (ERD), respectively [41]. ERS and ERD are determined by observing an amplitude enhancement or attenuation in a specified frequency range.

Alpha and beta are unique frequency bands due to their ERS and ERD response [23]. For alpha oscillations, ERD can be found in regions of the brain relevant to a task while ERS can be found in parts of the brain that are not relevant to that task. This behavior of attenuating active regions of

the brain while enhancing inactive regions of the brain with the respective frequency band is called the antagonistic ERS/ERD response [41]. This behavior can be seen in the beta band, particularly in the motor cortex region. For example, when a subject moves their left arm, the beta band power of the region of the sensorimotor cortex responsible for the left arm movement will decrease temporarily, while the region responsible for right arm movement will increase temporarily. Both region's beta band power will then return to normal when the arm movement is finished. This antagonistic ERS/ERD response is believed to inhibit irrelevant regions of the brain to a task to decrease the internal signal-to-noise ratio of neuronal activity.

In addition to frequency bands, the signal's source electrode must be taken into consideration. The most common method of electrode placement is detailed in the 10-20 International EEG Electrode Placement System shown in Figure 2.1 [1] [28]. The naming convention of each electrode is generally based on its respective lobe (the letter label), the laterality of the electrode (the parity of the numerical label), and the distance (the magnitude of the numerical label) of the electrode from the antero-posterior line which connects the nose to the back of the head. Starting on the antero-posterior line, the electrodes are identified with a "Z" instead of a number, indicating zero. The further an electrode is to the left of this line, the higher its assigned odd number (i.e., 1, 3, 5, 7). Similarly, the further an electrode is to the right of this line, the higher its assigned even number (i.e., 2, 4, 6, 8). The lobes of the brain used in the 10-20 Placement System are frontal (F), occipital (O), temporal (T), and parietal (P). While the letter preceding the number generally corresponds to the lobe, there are three exceptions: electrodes with "C" preceding the number correspond to the central area of the EEG map, electrodes with "Fp" preceding the number correspond to pre-frontal, and electrodes with "A" preceding the number correspond to the ear. For example, electrodes "A1" and "A2" correspond to the left and right ear, respectively.

Between the frontal lobe and the parietal lobe lies the sensorimotor cortex, which is responsible for movement and tactile perception. A color-coded display of the motor cortex is shown in Figure 2.2. Figure 2.3 shows a functional map of the sensorimotor cortex. By pairing information from these two figures and the 10-20 International EEG Electrode Placement System, it is possible to determine which electrodes may be useful for determining different states of motor activity. The

[1] https://commons.wikimedia.org/wiki/File:21_electrodes_of_International_10-20_system_for_EEG.svg
[2] https://commons.wikimedia.org/wiki/File:21_electrodes_of_International_10-20_system_for_EEG.svg

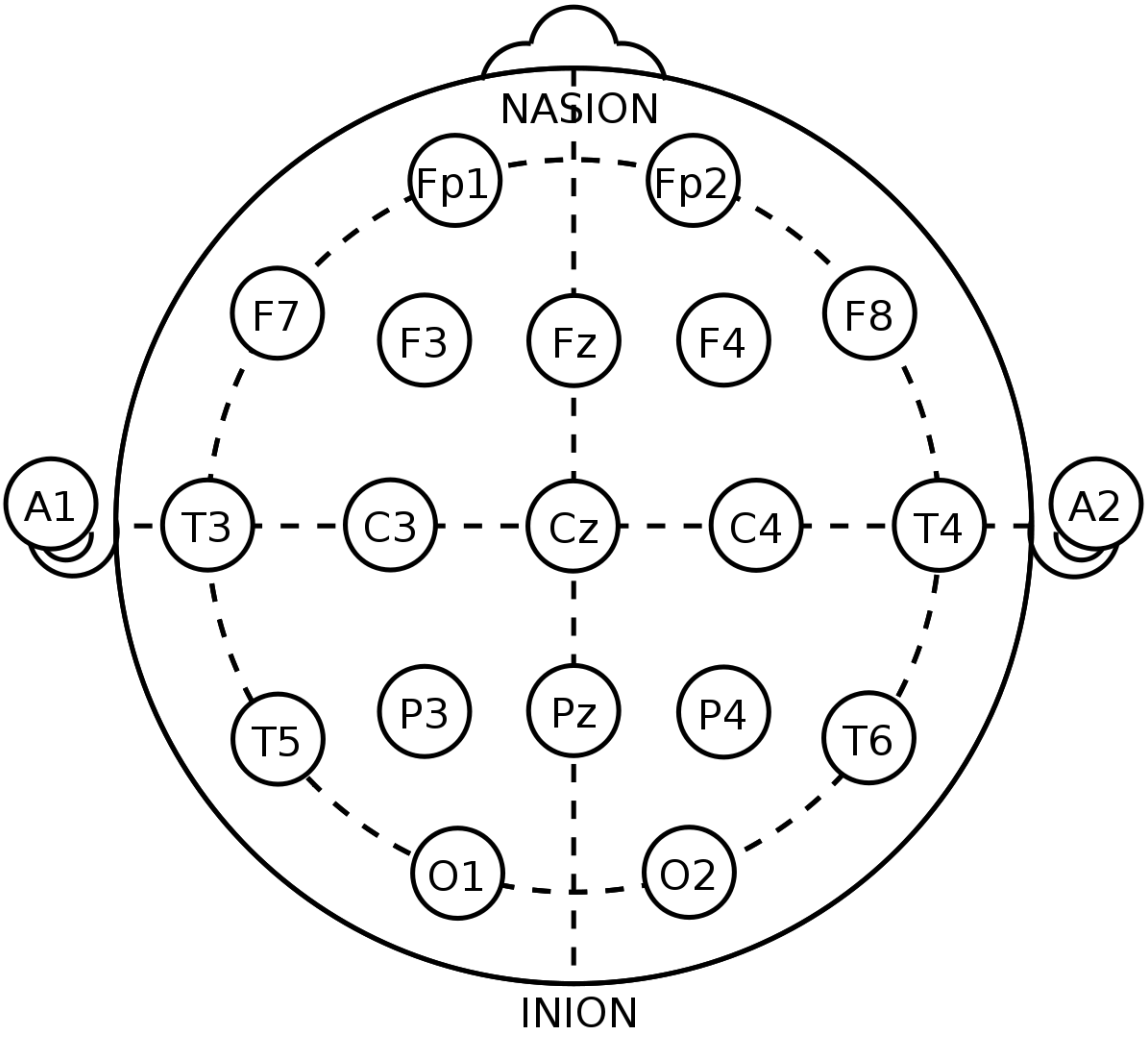

Figure 2.1: The 10-20 International EEG Electrode Placement System [2].

activity recorded by the C3 and C4 electrodes correspond to upper limb movements due to their placement over the sensorimotor cortex. C3 relates to the right upper limb and C4 relates to the left upper limb.

EEG is not the only form of neuronal activity recording. Electrocorticography (ECoG) and magnetoencephalography (MEG) are two other widely used methods [21]. EEG is useful for this application due to its relatively low cost and and portability (at the cost of signal clarity and spatial resolution). ECoG and MEG have a higher signal-to-noise ratio than EEG, however, ECoG utilizes electrodes placed directly on the surface of the brain, which requires an invasive surgical procedure, and MEG machinery is cumbersome. In addition, MEG measures magnetic fields resulting from neuronal current where ECoG and EEG measure voltage potential. The sample rates for all three of these methods are commonly between 125 and 1,024 Hz per channel. Figure 2.4 demonstrates the temporal and spatial resolutions of different neuronal recording techniques.

Two common types of movement analyzed using EEG are imagined movement and actual movement. Imagined movement, also referred to as motor imagery, is the act of mentally performing a movement task without activating any of the relevant muscles (e.g., thinking about walking instead of walking). The counterpart, actual movement, is when the movement is both mentally and physically performed. It has been shown in the literature that motor imagery elicits a similar neuronal response to that of actual movement [40]. If class of movement intent can be inferred from EEG signals and motor imagery, it would be possible to create brain machine interfaces for those that are physically handicapped. It is for this reason that motor imagery is popular for developing brain machine interfaces and is often explored in the literature.

2.2 Processing the Data

One common method of storing and accessing recorded EEG data is through the European Data Format (EDF) [30]. This format is comprised of metadata describing the dataset and a set of channels with corresponding data streams. Each channel corresponds to an EEG electrode's detected potential. In addition to these EEG electrode channels, there are channels reserved for electrocardiography (ECG), oxygen saturation, and general auxiliary inputs. This datatype is accessed using the MNE Python library [18].

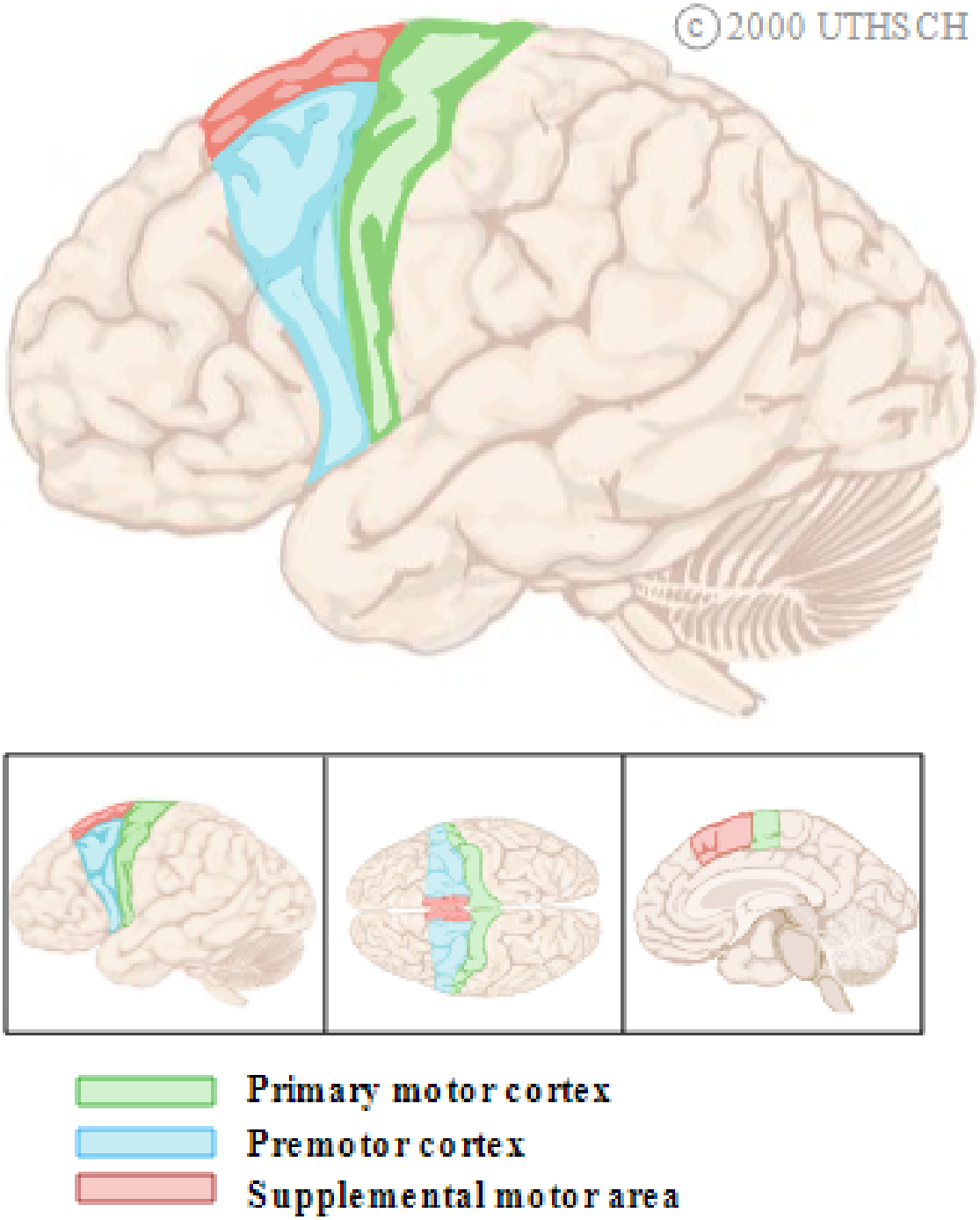

Figure 2.2: The motor cortex of the brain, with highlighted regions for the primary motor cortex, premotor cortex, and supplemental motor area [33].

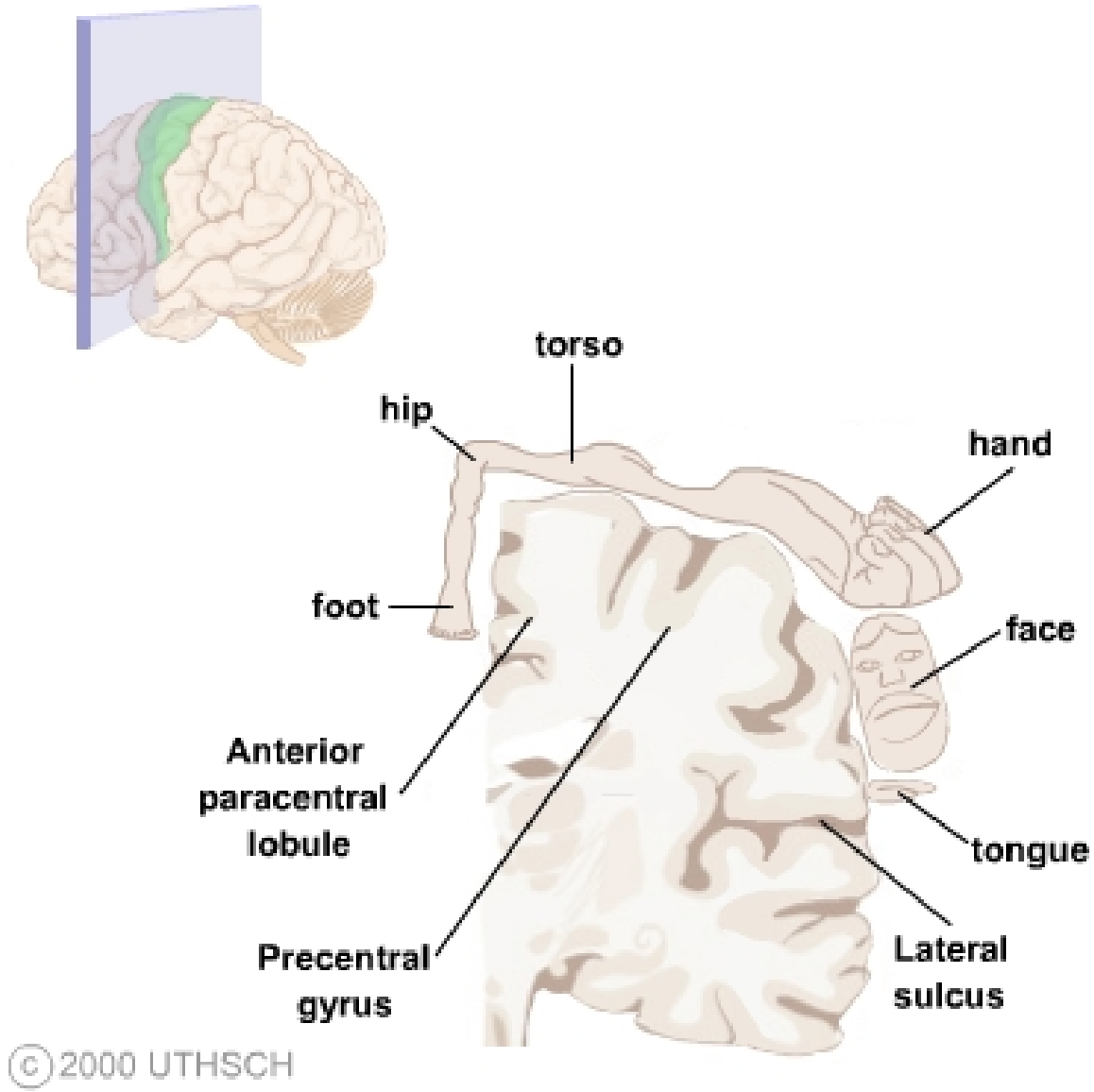

Figure 2.3: A functional labelling of the primary motor cortex [33].

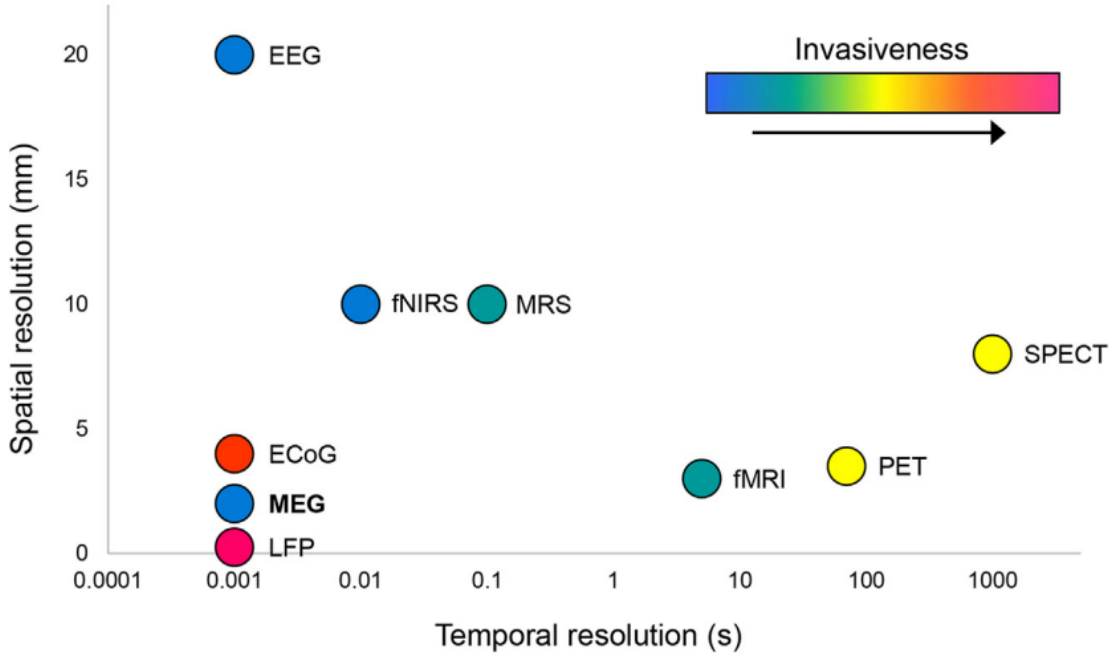

Figure 2.4: A chart contrasting different techniques of recording neuronal activity [21].

The Butterworth filter is a type of signal processing filter used to implement low-pass, high-pass, band-stop, and band-pass frequency filters. The order of the filter defines the sharpness of the attenuation slope at the cutoff frequency. The band-pass and band-stop implementations of this filter are created with an order of two for feature extraction and live analysis. Figure 2.5 shows the attenuation of a low-pass Butterworth filter at varying orders [3]. They are often used in tandem with EEG analysis to remove various types of noise, such as noise from the power system or voltage drift from the recording apparatus by using a notch filter and band-pass filter respectively.

The Fast Fourier Transform (FFT) is an algorithm designed to perform the Discrete Fourier Transform (DFT, shown in Equation 2.1) quickly [51]. DFTs allow for the approximation of amplitude in the frequency domain from a time series. The size of the FFT relative to the sample rate of the data defines the spectral resolution of the output. An FFT continuously taken over a sliding window of data with respect to time is referred to as a Short Time Fourier Transform (STFT). Since there is strong evidence that the frequency of brain activity correlates to events, STFTs are useful for interpreting EEG signals. Additionally, the result of the STFT may be binned to ensure

[3] https://commons.wikimedia.org/wiki/File:Butterworth_Filter_Orders.svg
[5] https://commons.wikimedia.org/wiki/File:Butterworth_Filter_Orders.svg

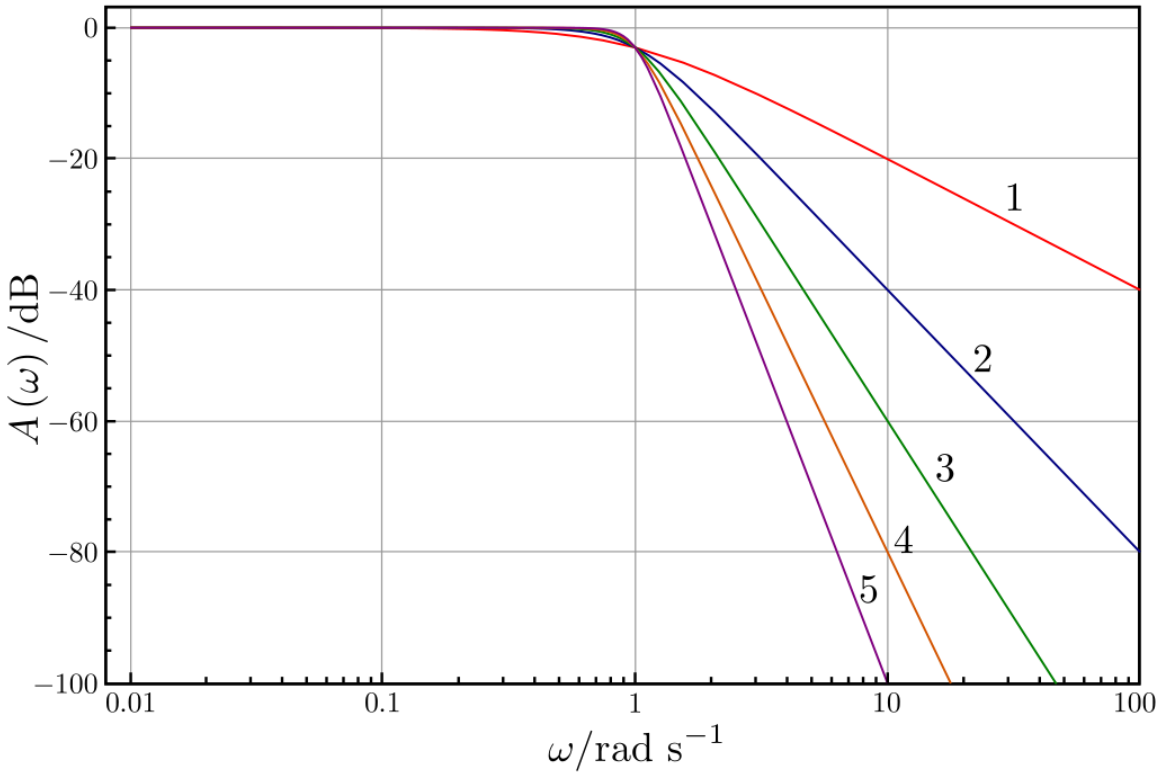

Figure 2.5: Attenuation of a low-pass Butterworth filter at the cutoff frequency at varying orders [5].

information is not lost as different subjects may exhibit different frequencies for the same actions [38].

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N}kn} \tag{2.1}$$

Power spectral density (PSD) is a metric which seeks to determine the power of a signal in the frequency domain over some average of time. Welch's method (derived by Peter Welch) applies the FFT algorithm to several contiguous subsets of the original signal [51]. These FFT results are then averaged together to generate an array of numbers representing the power of each frequency over the original signal. This algorithm is commonly used in analyzing EEG signals for spectral power [9].

STFTs and PSD may be used to create a two-dimensional plot, or spectrogram, that transforms the frequency analysis of a signal into a grid, traditionally shown as an image, where one axis corresponds to frequency and the other axis corresponds to time. Each element in the grid corresponds to the amplitude of its respective frequency at the time of the signal.

Entropy is a calculation that quantifies the amount of disorder or uncertainty in a system. Differential entropy is a subset of entropy that is used to measure the complexity of continuous, Gaussian, random variables. Differential entropy has also been commonly used for EEG signal analysis [12]. It has been shown that while EEG signals are not explicitly Gaussian, when the EEG signal is divided into several epochs, the individual epochs can be compared to a Gaussian distribution [10]. The equation to calculate differential entropy is shown in Equation 2.2, but with Gaussian distributed random variables, Equation 2.3 can be used.

$$h(X) = -\int_X f(x) log(f(x)) dx \tag{2.2}$$

$$h(X) = \frac{1}{2} log(2\pi e\sigma^2) \tag{2.3}$$

2.3 Traditional Classification Methods

There are several machine learning algorithms that can be used to perform classification by training a model capable of discerning patterns more effectively than human analysis. This thesis explores the following classification methods based on a survey of related literature: logistic regression [53], linear discriminant analysis [53] [45] [31] [17], decision trees [31], naïve Bayes [31], K-nearest neighbors [31], support vector machines [39] [53] [4] [31] [17], random forests [4], voting classifiers [31], and AdaBoost [17]. Several of these classification methods utilize hyperparameters which fine-tune the performance of a classifier. A grid search K-fold validation is performed to survey parameter values and determine the most accurate model for the given dataset.

Logistic regression is performed by applying a logistic function to a trained linear combination of weights and features. This creates an output mapping from 0 to 1 that describes the probability of something being true or false dependent on the input and the weights. An example of this function is shown in Equations 2.4 and 2.5, where p represents probability, w represents a set of weights, and x represents input for i features. Logistic regression has one hyperparameter called C which corresponds to regularization. Regularization is the penalization of large-value weights in the model in order to prevent overfitting [19]. Yong and Menon utilized logistic regression to classify rest, simple arm movement, goal oriented arm movement, and hand clenching using motor imagery [53].

$$p = \frac{1}{1 + e^{-k}} \tag{2.4}$$

$$k = w_0 + w_1 x_1 + w_2 x_2 + ... + w_i x_i \tag{2.5}$$

Linear discriminant analysis (LDA) uses linear algebra to change the data's basis to reduce the data's dimensionality [3]. When input data change their basis through LDA, the resulting data often clusters by class, allowing for classification. LDA has one hyperparameter associated with it: the solving function. Three solving functions are tested: singular value decomposition, least squares, and eigen solvers. Yong and Menon utilized LDA to classify rest, simple arm movement,

goal oriented arm movement, and hand clenching using motor imagery [53]. Rodrigo et. al. utilized LDA to classify rest, preparation, and movement in actual movement tasks [45]. Khrishna et. al. used LDA as part of their voting classifier to classify motor imagery states of left arm, right arm, left leg, and right leg movement [31]. Gao et. al. utilized LDA with AdaBoost to classify motor imagery for left hand, right hand, and rest [17].

Decision trees use a starting point and several conditional branching points comprised of a feature and a threshold to perform classification [19]. Depending on whether the conditional is true, the algorithm will traverse a designated path. At the bottom of each branch's path is a class which represents the tree's classification of the input data. One difficulty of this model is choosing the correct feature and threshold at each branching point to optimize the information gain. This optimization is often implemented through either the Gini Impurity (Equation 2.6) or the entropy (Equation 2.7) where $p_{i,k}$ is defined as the fraction of samples of class k at node i. These two algorithms attempt to quantify the impurity of a set of samples. If all the classes of one set of samples are the same, the score will be 0. Gini Impurity is often quicker to calculate than entropy since entropy requires a logarithmic calculation. For the sake of this thesis, the Gini Impurity is used to avoid this logarithmic calculation. The cost function that is optimized using the Gini Impurity or entropy is defined in Equation 2.8. In this equation, m is defined as the number of samples represented and G represents either the Gini Impurity or entropy, depending on what is chosen. This model is prone to overfitting since a tree could potentially have as many conditional branches as necessary to perfectly classify the training dataset, demonstrating the need for hyperparameters. The hyperparameters considered for decision trees are `min_weight_fraction_leaf`, `min_samples_split`, and `max_depth`. `min_weight_fraction_leaf` is defined as the minimum sum of weighted samples out of the total training set weight at a node for it to be a leaf. A leaf is defined as a node with no children. `min_samples_split` is defined as the minimum number of samples required at a leaf to transform the leaf into a node with branching paths. `max_depth` is defined as the maximum depth (the distance from the root node to the furthest leaf) of the decision tree. Khrishna et. al. use decision trees as part of their voting classifier to classify motor imagery states of left arm, right arm, left leg, and right leg movement [31].

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2 \tag{2.6}$$

$$G_e = -\sum_{k} p_{i,k} log(p_{i,k}) \tag{2.7}$$

$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right} \tag{2.8}$$

Naïve Bayes (NB) classification is an expansion of Bayes' theorem, shown in Equation 2.9, and the naïve assumption that all input features are independent of each other. While this assumption (in most cases) is not true and will hinder the model's accuracy, it simplifies the calculation. Bayes' theorem is used to answer the question "What is the probability of class A given that B is true?" Since B represents the input data, $P(B|A)$ is substituted with $\prod_i P(b_i|A)$ and $P(B)$ is substituted with $\prod_i P(b_i)$. When each feature is Gaussian in nature, $P(b_i|A)$ is defined by Equation 2.10. The training set is used to define these probabilities. A classification result is achieved by testing the probability of each class ($P(a_j|B)$) and returning the class with the highest probability. Naïve Bayes classification utilizes one hyperparameter called variance smoothing (stylized `var_smoothing`) which is a coefficient multiplied by the variance of the features used to smooth the Gaussian distribution curve. The Gaussian distribution curve is used to apply probabilities to continuous valued features (as opposed to categorized or discretized features). Khrishna et. al. use naïve Bayes classification as part of their voting classifier to classify motor imagery states of left arm, right arm, left leg, and right leg movement [31].

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.9}$$

$$P(b_i|A) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}) \tag{2.10}$$

K-Nearest Neighbors (KNN) spatially plots the samples of the training set, then classifies the given input by determining the most frequency class of the closest K neighbors to that input [1]. The distance between samples is often defined using Euclidean distance, shown in Equation 2.11. The hyperparameter associated with KNN is the number of neighbors (stylized `n_neighbors`). Khrishna et. al. use KNN as part of their voting classifier to classify motor imagery states of left arm, right arm, left leg, and right leg movement [31].

$$d(A, B) = \sqrt{\sum_{i=1}^{n}(A_i - B_i)^2} \tag{2.11}$$

Support vector machines (SVMs) perform classification by optimizing a hyperplane (a plane in an arbitrary number of dimensions) that separates the classes of the training set [26]. The optimization maximizes the distance of the samples to the hyperplane. The radial basis function (RBF) kernel (Equation 2.12) increases the SVM's efficiency by creating a new set of features based on similarity. A kernel is defined as a function that is passed into an algorithm to perform an abstract task (e.g., distance functions). Due to the prominence of RBF's use in the literature, it is used as the kernel function for SVM in this thesis. The two hyperparameters trained for SVM are C for the regularization, and γ for the RBF kernel. Mebarkia and Reffad utilized SVMs with RBF to classify imagined left hand movement against imagined right hand movement [39]. Yong and Menon utilized SVMs to classify rest, simple arm movement, goal oriented arm movement, and hand clenching using motor imagery [53]. Bentlemsan et. al. utilized SVMs to classify motor imagery in left foot, right foot, both feet, and tongue movement [4]. Khrishna et. al. utilized SVMs as part of their voting classifier to classify motor imagery states of left arm, right arm, left leg, and right leg movement [31]. Gao et. al. utilized SVMs with AdaBoost to classify motor imagery for left hand, right hand, and rest [17].

$$K(x_i, x_j) = exp(-\gamma||x_i - x_j||^2), \gamma > 0 \tag{2.12}$$

2.4 Ensemble Methods

Ensemble learning is a method of training several models and using them in tandem to achieve higher accuracy. Voting classifiers, random forests, and AdaBoost are the three ensemble methods that are explored.

A voting classifier combines predictions from several different trained classifiers until a consensus is reached [19]. Hard voting classification and soft voting classification are two methods of combining each model's prediction. Hard voting counts the total number of predictions in each class and chooses the class with the highest number of votes. Soft voting assigns each model a weight and determines a probability for each class based on a linear combination of each model's probability and weight. The class with the highest probability is chosen. Khrishna et. al. utilized a voting classifier comprised of KNN, SVM, LDA, naïve Bayes, and decision trees to classify motor imagery states of left arm, right arm, left leg, and right leg movement [31].

Random forests use an ensemble of different decision trees in parallel [24]. Hard voting is used to determine the classification results from a random forest. If all decision trees in a random forest are similar, the result of the random forest classification is similar to using one of the decision trees. The hyperparameters considered for random forests are `min_samples_leaf`, `max_features`, and `max_depth`. `min_samples_leaf` is defined as the minimum number of samples required to create a leaf node. `max_features` is defined as the maximum number of considered features when looking for the most optimal branch conditional. `max_features` is either a specified integer, the square root of the number of features, or the logarithm-base-2 of the number of features. Bentlemsan et. al. utilized random forests to classify motor imagery in left foot, right foot, both feet, and tongue movement [4].

Boosting combines weak models (models that classify only slightly better than chance) to create a classifier with high accuracy. AdaBoost is a popular boosting algorithm [15]. The algorithm starts with a single trained classifier. The weights of misclassified samples are increased, and a second classifier is then trained on the newly weighted training set. This process is iterated for a user-specified number of classifiers (stylized `n_estimators`). When predicting new values, the result is a soft vote of all classifiers weighted by overall training set accuracy. In this thesis, the type of classifier

is varied over SVM with RBF, naïve Bayes, and decision trees. Gao et. al. utilized AdaBoost in conjunction with SVM and LDA to classify motor imagery for left hand, right hand, and rest [17]. While Gao et. al. were able to use LDA as the classifier base of AdaBoost, the library used for training AdaBoost does not support LDA with AdaBoost because LDA is unable to take in a sample weight parameter.

2.5 Deep Learning Classification Methods

Convolutional Neural Networks (CNNs) use convolutional layers and pooling layers to find patterns in data [19]. The convolutional layers train kernels to perform the convolution operation on the input function. The pooling layers select the dominant features identified by a preceding convolutional layer, and excludes the non-dominant features, reducing the dimensionality of the data as it passes through the model. At the end, fully connected or "dense" layers are used to identify non-linear patterns in the training process. This model is loosely based on the structure of the visual cortex. The early convolutional layers in a model react to smaller learned patterns while the later convolutional layers react to larger learned patterns. These multiple layers are necessary for the model to learn a hierarchy of features from the input data. To give an example, imagine a CNN trained to identify humans in photos. Early convolutional layers may have learned features to identify an eye or a nose, and a middle convolutional layer might see that two eyes and a nose were identified and infers a face from that combination, and a final convolutional layer may see that there is a face, two arms, and two legs and predict that it is a human. CNNs are largely used for image classification. Lun et. al. utilized CNNs on raw EEG data to perform motor imagery classification between clenching left hand, clenching right hand, clenching both hands, and clenching both feet [36]. Tayeb et. al. utilized CNNs to create three CNN models to classify left and right hand motor imagery from spectrograms [48].

Recurrent Neural Networks (RNNs) are deep learning models that perform predictions on time series data [19]. In a recurrent layer, each neuron is connected to the input to the layer, and the neuron preceding it, thus giving the model a sense of internal memory. Inside the layer there are two sets of weights that are trained: one for the input to the layer, and one for the preceding neuron. Since the length of memory is defined by the number of neurons in a layer, the amount that can be remembered

by an RNN is limited. Long short-term memory (LSTM) mitigate this issue by training gates that learn when to store, use, and forget features for an extended amount of time. These networks are often used for natural language processing and audio processing (a type of signal analysis). Since RNNs are useful in time series data and signal analysis, it is expected that RNNs can be useful for analyzing EEG data. Tayeb et. al. utilized an LSTM network and a Recurrent Convolutional Neural Network to classify left and right hand motor imagery [48].

Chapter 3

Related Work

Yong and Menon utilized logistic regression, linear discriminant analysis (LDA), and support vector machines (SVMs) with the radial basis function (RBF) kernel to classify four states of motor imagery: rest, simple arm movement (e.g., repeated arm flexions), goal oriented arm movement (e.g., reaching towards a glass of water), and grasping [53]. There were 21 EEG electrodes considered for analysis in tandem. The EEG signals were downsampled from 1,000 Hz to 250 Hz and band-passed for 6-35 Hz. The signal was epoched from 1-3 seconds. The researchers used common spatial patterns (CSP), filter-bank common spatial patterns (FBCSP), and logarithmic band power (LBP) for feature extraction. When using FBCSP, the signal was used to generate three new signals, band-passed by 7-15 Hz, 15-25 Hz, and 25-30 Hz, respectively. Binary classification between rest and motor imagery had a range of 75-81% accuracy, where binary classification between the motor imagery states had a range of 61-67% accuracy. The accuracies reported are with the highest performing classifier per subject. The researchers found that LBP for features and SVM for classification yielded the highest accuracy.

Dr. Rowland's lab at the Medical University of South Carolina has demonstrated LDA's effectiveness in classifying Parkinson's disease patients from healthy controls, determining Parkinson's disease severity (mild vs. severe), and predicting motor improvement with deep brain stimulation with accuracies of 84%, 93%, and 99% respectively using magnetoelectroencephalography (MEG) [20].

These results are shown in Figure 3.1.

Rodrigo et. al. utilized LDA to classify rest, preparation, and movement in individuals performing actual movement [45]. The researchers used 28 electrodes to record EEG data at 256 Hz that was downsampled to 60 Hz. To perform feature extraction, a Laplacian spatial filter and electrode re-referenced filter was applied to the original set of signals to get three sets of signals. An r^2 test was applied on the FFT values for the C3 signal from all the filtered sets of data between the following sets: rest and preparation, rest and movement, and preparation and movement. The type of filter on the C3 electrode with the highest r^2 value was used and electrodes with an r^2 value within some threshold were added to the feature set along with C3. Additionally, the frequency band considered was selected via r^2 test. Performing binary classification between the three classes had an accuracy range of 64-68%, whereas performing multi-class classification yielded an accuracy of 55%.

Mebarkia and Reffad used SVMs with the RBF kernel to classify imagined left hand movement against imagined right hand movement using C3 and C4 EEG electrodes [39]. These signals were band-passed from 0.5-30 Hz and then analyzed to extract 16 features in total using power spectral density (PSD), kurtosis, cumulative sum of signals, and continuous wavelet transforms. The set of features extracted and the length of EEG signal with respect to time were varied. These parameters were selected by using a genetic algorithm. Using one SVM and an ensemble of three SVMs were explored. Utilizing one SVM achieved an accuracy of 90%, compared to utilizing three SVMs which achieved an accuracy of 94%. Consensus was reached in the ensemble of SVMs by using a hard voting method.

Bentlemsan et. al. utilized random forests and SVMs to classify motor imagery for left foot, right foot, both feet, and tongue movement [4]. C3, C4, and Cz EEG electrodes were recorded at 250 Hz, and then band-passed through 9 different filters to cover 4-40 Hz. Features were extracted from these signals using FBCSP for 18 total features. The random forest yielded 80% accuracy on average whereas the SVM yielded 66%.

Khrishna et. al. utilized a voting classifier to classify between left arm, right arm, left leg, and right leg motor imagery [31]. The EEG signals were acquired using 24 electrodes at 128 Hz, and then band-passed for 8-30 Hz. The researchers used cross-correlation to extract features to receive

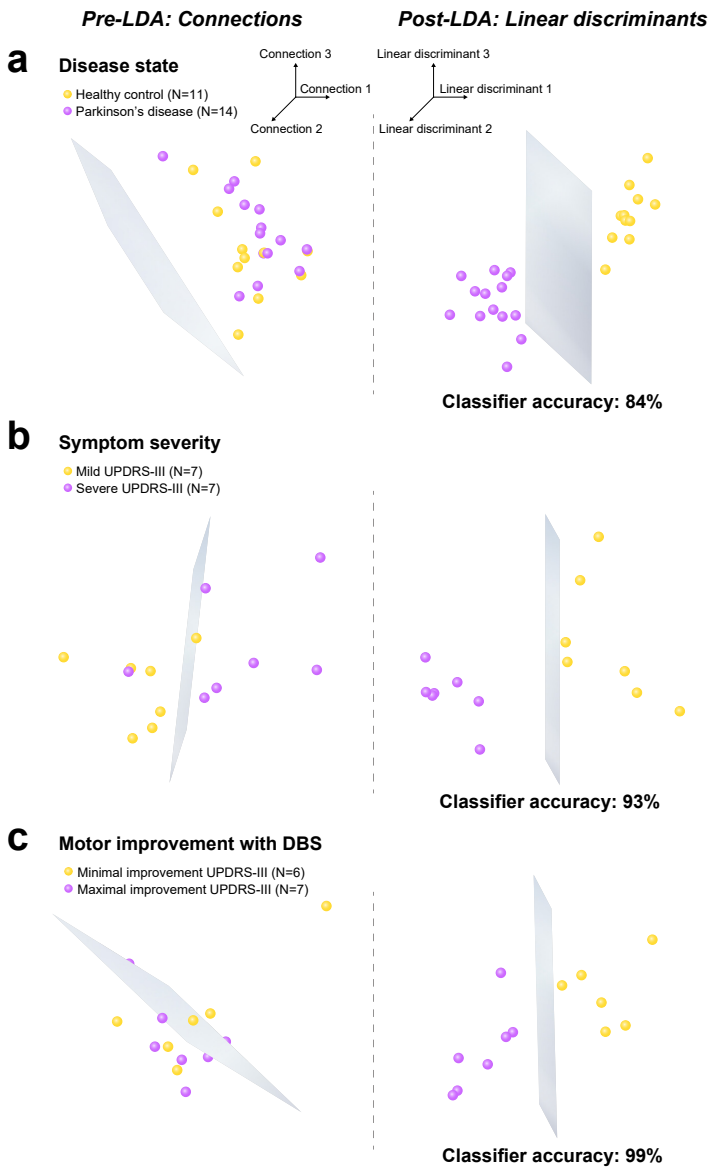

Figure 3.1: Visualizing the use of LDA with MEG data [20].

one resultant cross-correlation array for each channel. Five classifiers were trained on each channel's data: k-nearest neighbors (KNN), SVM, LDA, naı"ve Bayes, and a decision tree. The researchers were able to achieve an average accuracy of 86%.

Gao et. al. utilized AdaBoost with SVM and LDA to classify motor imagery for left hand, right hand, and rest [17]. The C3 and C4 electrodes were recorded at 1,000 Hz and then band-passed for 8-30 Hz. Feature extraction was performed by calculating the Kolmogorov complexity. The researchers determined that using AdaBoost with a base of SVM and LDA achieved a classification accuracy of 74% and 72%, respectively.

Lun et. al. used a Convolutional Neural Network (CNN) to perform a motor imagery classification task [36]. The motor imagery classes were as follows: clenching left hand, clenching right hand, clenching both hands, and clenching both feet. The CNN consisted of five convolutional layers, four pooling layers, and one fully connected layer. Four seconds of raw, unprocessed EEG signal sampled at 160 Hz for two electrodes were used as input data. The pair of electrodes chosen was based off of cross-validation across nine different symmetrical electrode pairings. The C3 and C4 electrode pair was observed to yield the highest classification accuracy. Using a 90/10 training-to-test ratio, researchers were able to classify the four states of imagined movement at 98% accuracy.

Tayeb et. al. sought to classify left and right hand motor imagery while comparing results between CNNs, Long-Short Term Memory (LSTM) networks, and Recurrent Convolutional Neural Networks (RCNNs) [48]. Researchers chose to observe C3, C4, and Cz electrodes and epoched the signals by four seconds with a 125-millisecond stride. The signals were passed through a notch filter at 50 Hz, a high-pass filter at 0.5 Hz, and a band-pass filter for 2-60 Hz. This signal was passed directly to the LSTM network as input and used to derive a spectrogram for the CNNs and RCNN. In total, one LSTM, one RCNN, and three CNNs were created. The CNN architectures were observed to have accuracies of 84%, 67%, and 92%, the LSTM architecture was observed to have an accuracy of 66%, and the RCNN architecture was observed to have an accuracy of 78%.

Chapter 4

Methods

All analysis was conducted in Python 3.9 [44] using the NumPy [22], SciPy [49], and Scikit-Learn [43] libraries. Visualizations were performed using Matplotlib [27] and Graphviz [13].

4.1 The Prerecorded Dataset

The prerecorded dataset used for analysis was collected by Dr. Nathan Rowland's research lab at the Medical University of South Carolina over the course of 2019 to 2021. The dataset is comprised of EEG data for different subjects. Each subject is either a healthy control subject, Parkinson's disease patient, or chronic stroke patient. The dataset was collected to determine the effect of transcranial direct current stimulation (tDCS) when administered to the sensorimotor cortex. As such, some patients have tDCS administered to them throughout the data collection process. While the subjects' EEG data are recorded, they are to perform a motor task in a virtual reality environment using an Oculus Rift headset and handheld controllers. The subjects are instructed to only use one arm, and to use the arm most affected by their Parkinson's disease or stroke, if applicable. Each task in the trial starts out with their arm in a common holding position near their body. At some period, a vibration is felt on the controller, and then at a subsequent period, a sphere will virtually appear in front of them at a random location. The subject is instructed to reach out and touch the sphere upon appearance with their designated arm, hold the position, and return to the home position. The

Subject Number	Condition	tDCS Administration	# Trials Completed
Subject 28	healthy control	sham	6 trials
Subject 29	healthy control	stim	6 trials
Subject 40	Parkinson's disease	sham	6 trials
Subject 41	Parkinson's disease	sham	6 trials
Subject 42	chronic stroke	stim	5 trials
Subject 43	chronic stroke	stim	4 trials

Table 4.1: List of subjects' data used with their condition, tDCS administration type, and number of trials completed.

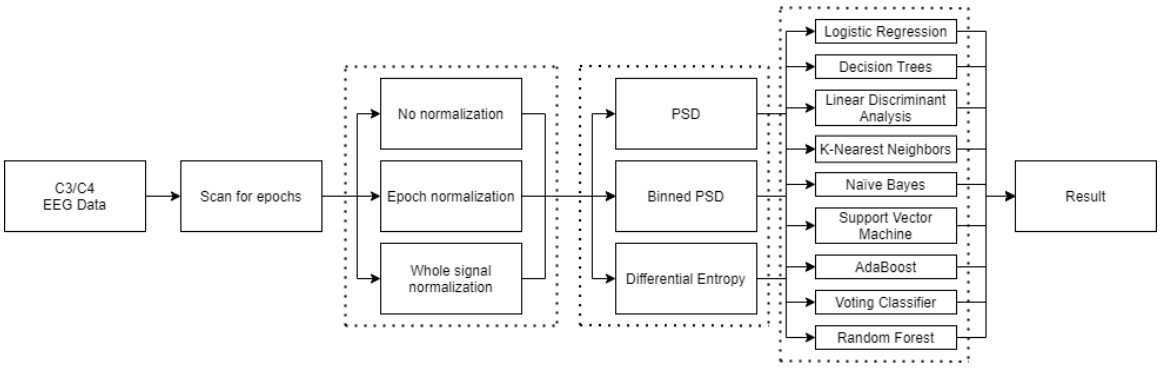

Figure 4.1: The workflow used for analysis on the prerecorded dataset.

period in which the subject stays in the home position is referred to as a "hold epoch." The period in between the controller vibration and the target sphere appearing is referred to as a "prep epoch." Lastly, the period from which the sphere appears to the subject's successful target reach is referred to as the "reach epoch." Each trial is comprised of 12 of these reaches. Each subject performs up to six of these of trials separated by trials of self-paced left arm and right arm flexions. If a subject experiences severe fatigue, not all trials will be completed. Only the hold and reach epochs are considered for analysis as part of this thesis. Table 4.1 shows each subject, their condition, and the number of recorded trials. In the tDCS column, "stim" refers to tDCS administration, whereas "sham" refers to the absence of tDCS administration.

The workflow of the analysis is shown in Figure 4.1. Reading from left to right, the first dotted box indicates the selection of normalization method, the second dotted box indicates the selection of feature extraction, and the third dotted box indicates the selection of the trained model.

4.2 Preprocessing

Due to the low signal-to-noise ratio of EEG signals, several preprocessing techniques are often utilized when analyzing them. Some common preprocessing techniques include high-pass filtering starting at 1 or 2 Hz to remove voltage drift of the recording apparatus, notch filtering at 50 or 60 Hz to remove electrical power system noise, or performing independent component analysis and principal component analysis to remove muscle and eye movement artifacts. In this thesis, a specific design decision is made to perform minimal preprocessing to lessen the burden of processing on the hardware running the code, which allows for smaller form computing devices to run this same workflow to achieve live classification. The absence of these preprocessing techniques will likely lower the overall accuracy demonstrated by the trained models. Despite attempting to minimize the amount of preprocessing, there is one preprocessing technique utilized: normalization. Three types of normalization techniques are explored: the lack of normalization, normalization per event epoch, and normalization of the entire signal. Normalization is performed via the Z-Score method shown in Equation 4.1. An example of a normalized signal epoch is shown in Figure 4.2.

$$x_{norm} = \frac{x - \bar{x}}{\sigma} \tag{4.1}$$

4.3 Feature Extraction

Due to the nature of EEG signals, it is possible to extract a bevy of features for further processing. Feature extraction is necessary in order to quantify patterns of activity in the signal. Based on a survey of the literature, there are three feature extraction techniques explored, two of which based on power spectral density (PSD), and one based on differential entropy.

PSD is employed in two different ways: binned by 1 Hz and binned by wave band. The SciPy implementation of Welch's method is used. By using the sample rate value as the number of segments for the FFT calculation, the result of Welch's method has a spectral resolution of 1 Hz. The subset of this result representing the 0-50 Hz frequency range is stored for each epoch. Therefore, for each electrode considered, 51 features are outputted. Figure 4.3 shows a Welch's method output of the EEG signal shown in 4.2. To extract the binned PSD features, all the respective PSD values for each

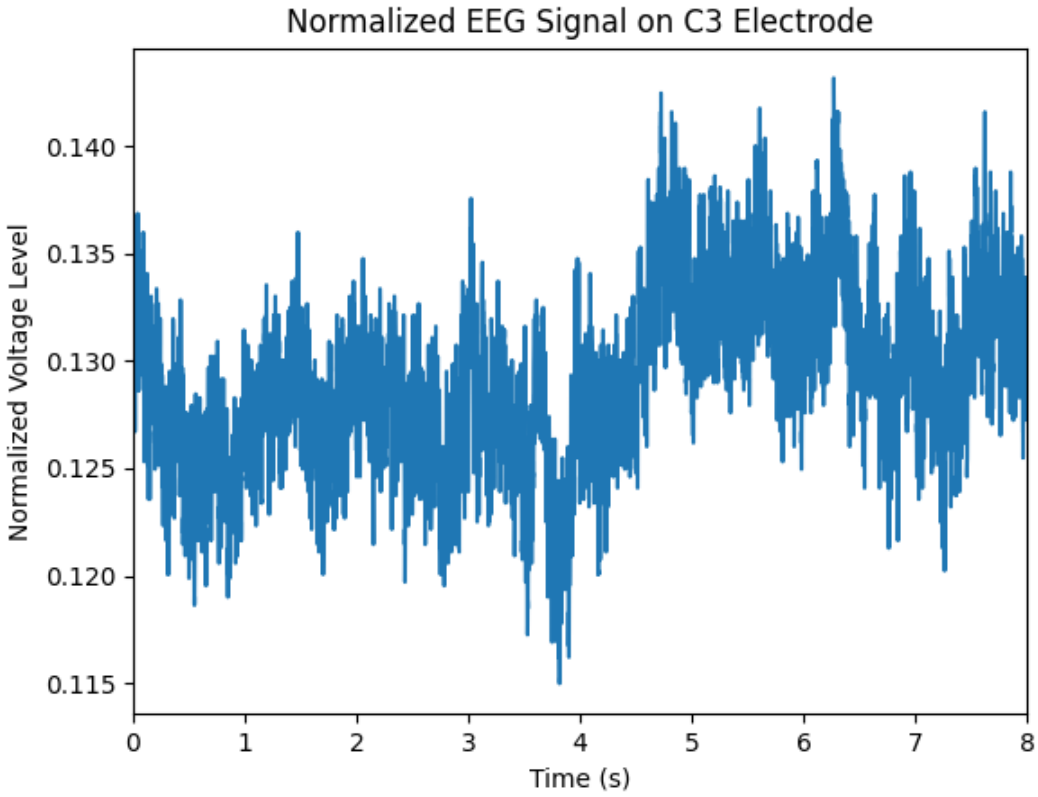

Figure 4.2: An 8 second epoch of an EEG signal on the C3 electrode with Z-score normalization.

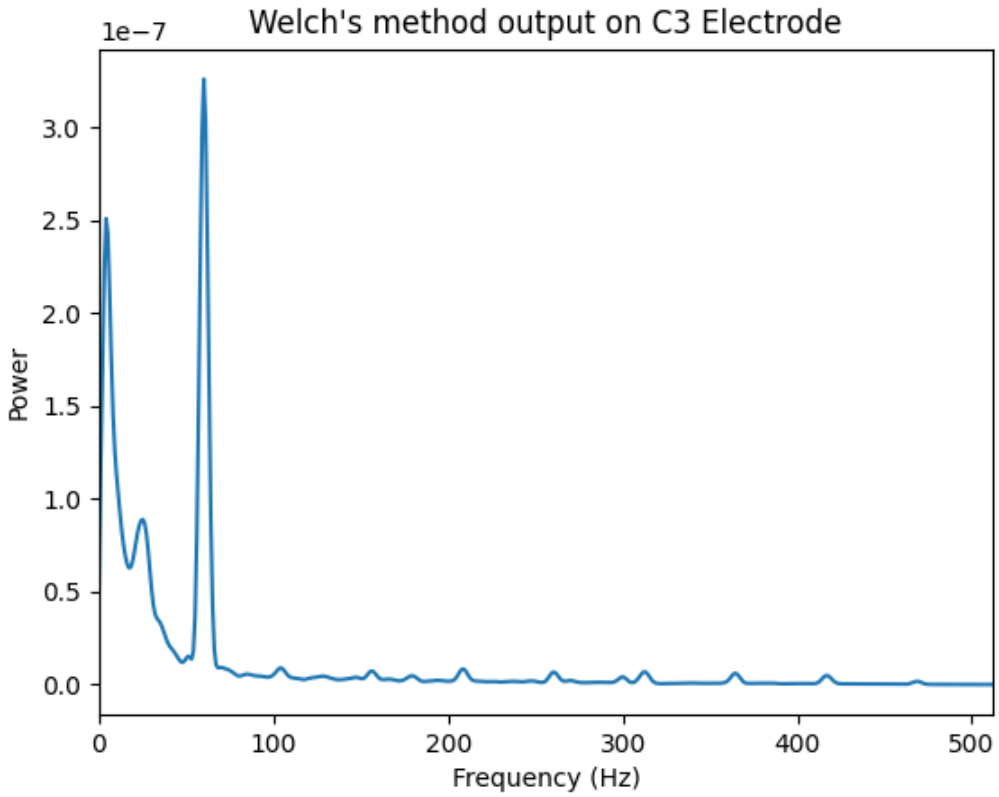

Figure 4.3: An example of the output from Welch's method. The large, 60 Hz spike demonstrates the importance of implementing a 60-Hz notch filter to reduce environment noise.

frequency band are summed according to Table 2.1 to output five total features for each electrode, one for each frequency band.

To perform differential entropy, the raw EEG signal is fed into five different 2^{nd}-order Butterworth band-pass filters, each corresponding to a frequency band defined in Table 2.1. The output of each of these filters is fed into the differential entropy function to generate five features for each EEG electrode, one for each frequency band. A visualization of this workflow for differential entropy feature extraction is presented in Figure 4.4. Both the Butterworth filter and differential entropy function are implemented by SciPy.

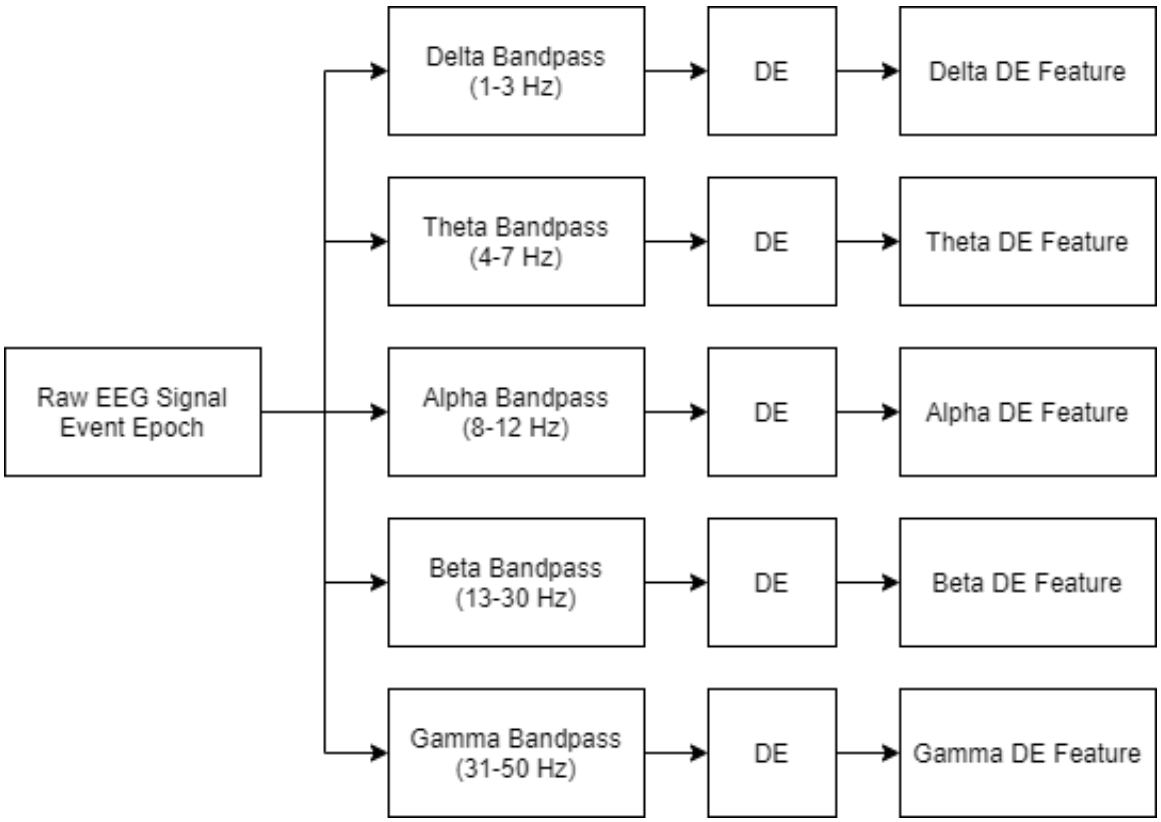

Figure 4.4: A visualization of differential entropy implementation workflow.

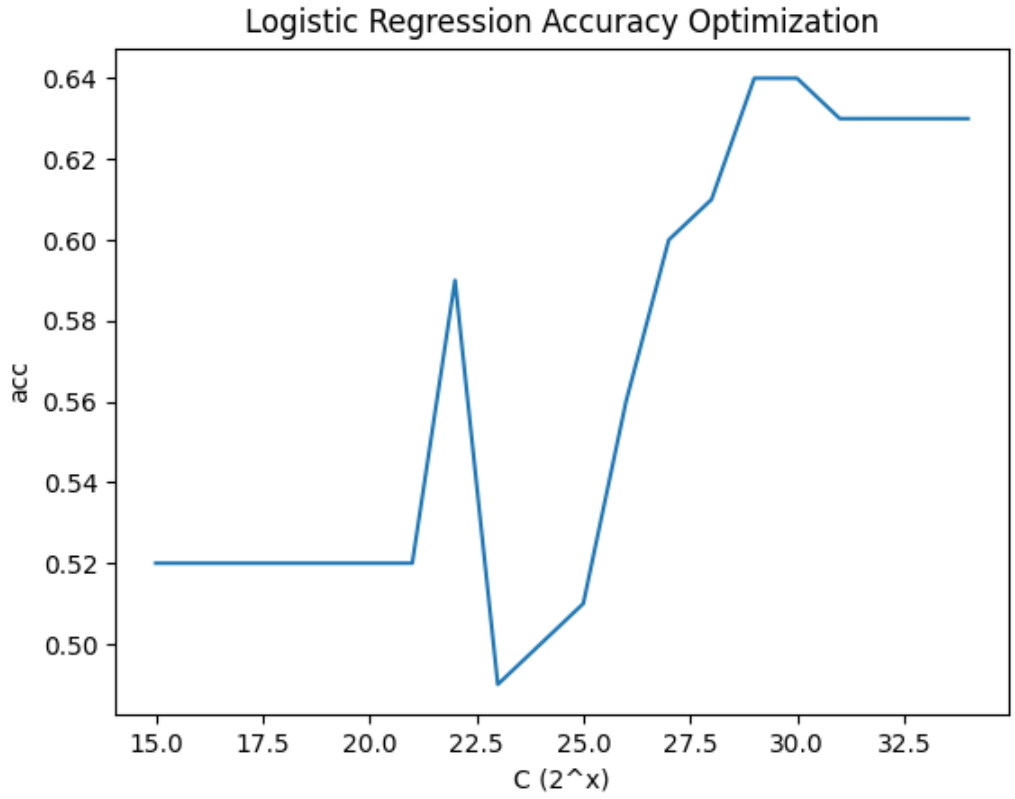

Figure 4.5: A visualization of k-fold validation on logistic regression optimizing C.

4.4 Machine Learning

Now that normalization and feature extraction has been discussed, machine learning is employed to perform the classification task of distinguishing states of movement from states of rest using the extracted features. As part of training each explored machine learning algorithm, 70% of the EEG epochs are used for training and 30% of the EEG epochs are used for validation such that no epoch is in both the training and validation set. This allows the created models to be tested for efficacy on unseen data. The following paragraphs discuss the hyperparameter searching process for each classifier used.

For logistic regression, a univariate grid search was performed on the parameter C for values 2^x for $15 \leq x < 35$. A visualization of this process is shown in Figure 4.5.

Comparisons are conducted for accuracies achieved by singular value decomposition (SVD), least

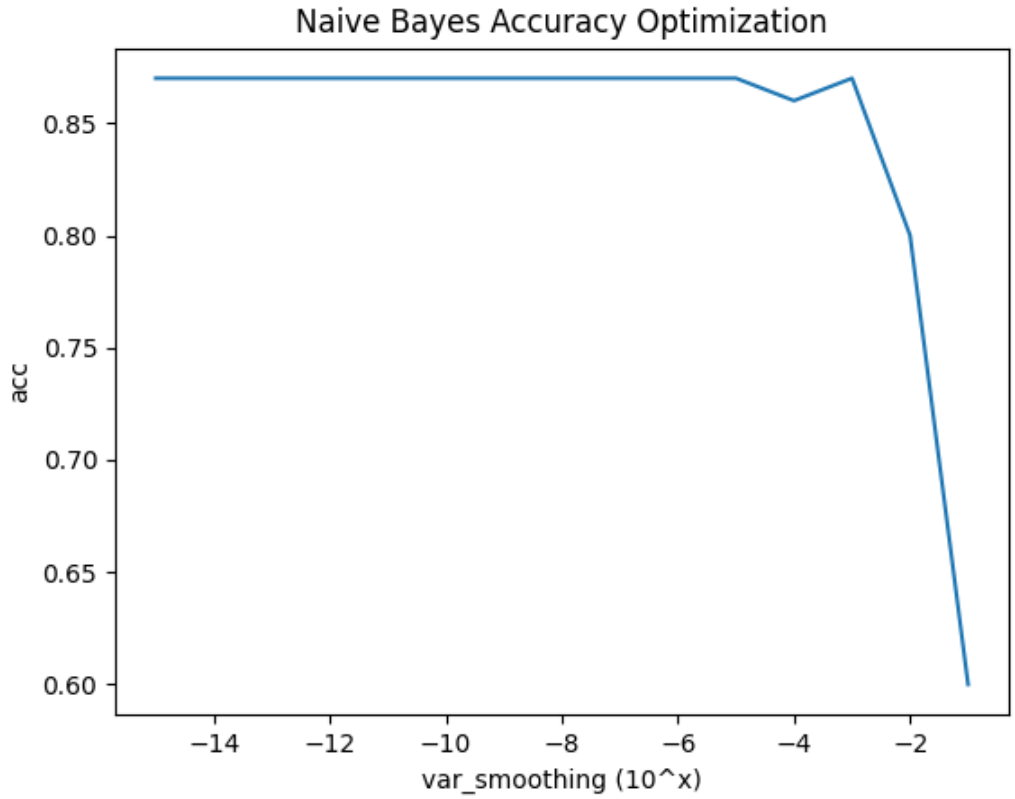

Figure 4.6: A visualization of k-fold validation on naïve Bayes classification optimizing `var_smoothing`

.

squares (LSQR), and eigen solvers for LDA. Although all models demonstrated similar accuracies, SVD is chosen since it does not compute the covariance matrix, and therefore takes less time to run.

For naïve Bayes classification, a Gaussian implementation was used due to the Gaussian nature of epoched EEG data [10]. A parameter representing the variance (termed `var_smoothing`) is varied by 10^x for $-15 \leq x < 0$. A visual representation of search results is shown in Figure 4.6.

For KNN, the number of neighbors that the classifier considered is varied for $3 \leq$ `n_neighbors` < 10. A visual representation of search results is shown in Figure 4.7.

For decision trees, the parameter `min_weight_fraction_leaf` is empirically determined to be 0. A

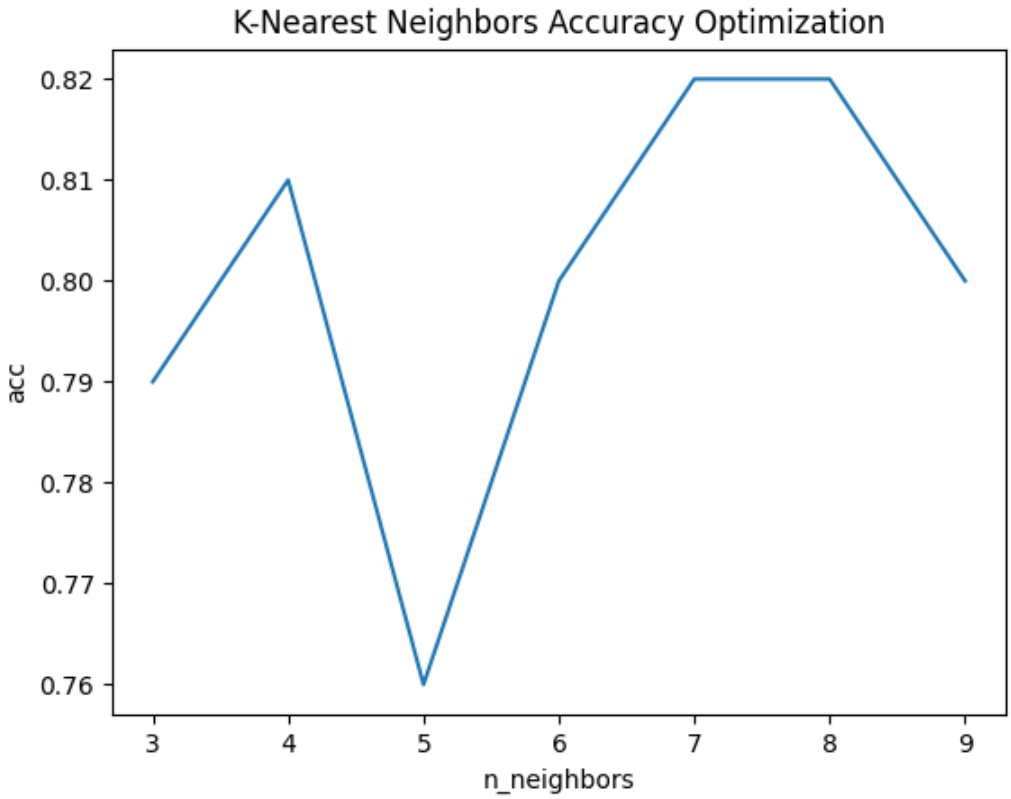

Figure 4.7: A visualization of k-fold validation on KNN optimizing `n_neighbors`.

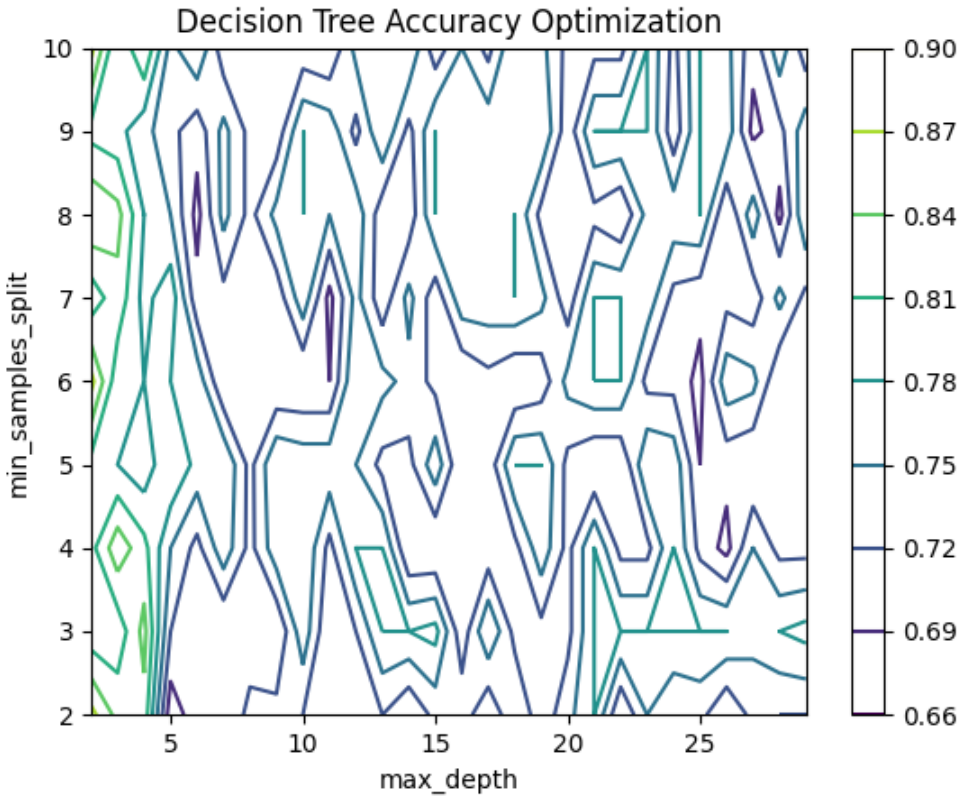

Figure 4.8: A visualization of k-fold validation on decision trees optimizing `min_samples_split` and `max_depth`.

grid search is performed to optimize the parameters `min_samples_split` and `max_depth`. `min_samples_split` is varied for $2 \leq x < 11$ and `max_depth` is varied for $2 \leq x < 30$. A visualization of this result is shown in Figure 4.8. An example of a trained decision tree is shown in Figure 4.9.

For random forests, `min_samples_leaf` is empirically determined to be 1, `max_features` to be `sqrt`, and `max_depth` to be 30. A grid search is performed for `min_samples_split` and `n_estimators`. `min_samples_split` is varied for $2 \leq x < 5$ and `n_estimators` is varied over the set $5n$ for $5 \leq n < 21$. A visualization of this result is shown in Figure 4.10. The average feature importances of the trained random forests for each subject are shown in Table 4.2. Feature importance is calculated by taking each feature's average depth of use and weighing the average out of one relative to the other features' depths. The earlier a feature is used in a tree, the more important it is considered.

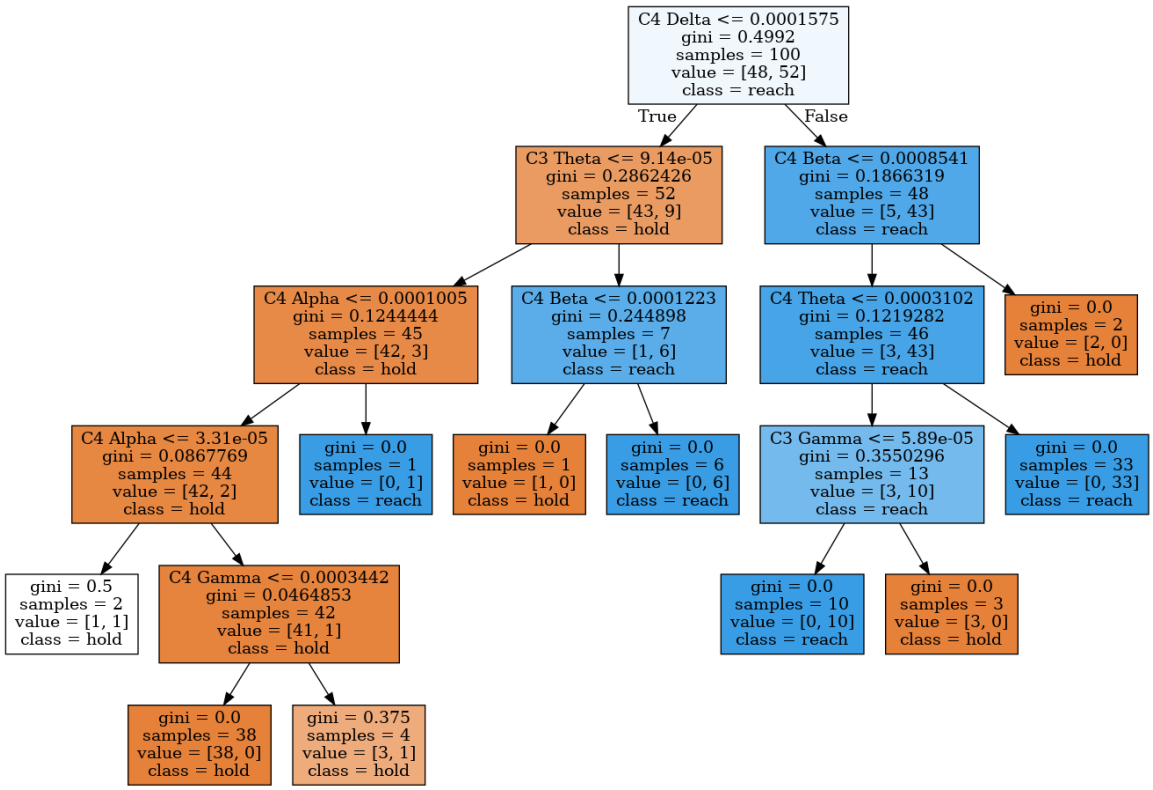

Figure 4.9: A decision tree created from training Subject 0028.

	C3					C4				
Subject #	Delta	Theta	Alpha	Beta	Gamma	Delta	Theta	Alpha	Beta	Gamma
28	0.111	0.110	0.065	0.039	0.029	0.234	0.168	0.136	0.039	0.069
29	0.103	0.051	0.043	0.030	0.032	0.242	0.272	0.120	0.039	0.066
40	0.021	0.057	0.099	0.259	0.113	0.003	0.027	0.080	0.217	0.124
41	0.130	0.146	0.084	0.068	0.044	0.120	0.083	0.066	0.187	0.071
42	0.161	0.118	0.113	0.068	0.076	0.097	0.093	0.136	0.060	0.080
43	0.121	0.183	0.082	0.048	0.086	0.138	0.088	0.093	0.084	0.077

Table 4.2: The average feature importance out of one for each feature over each subject.

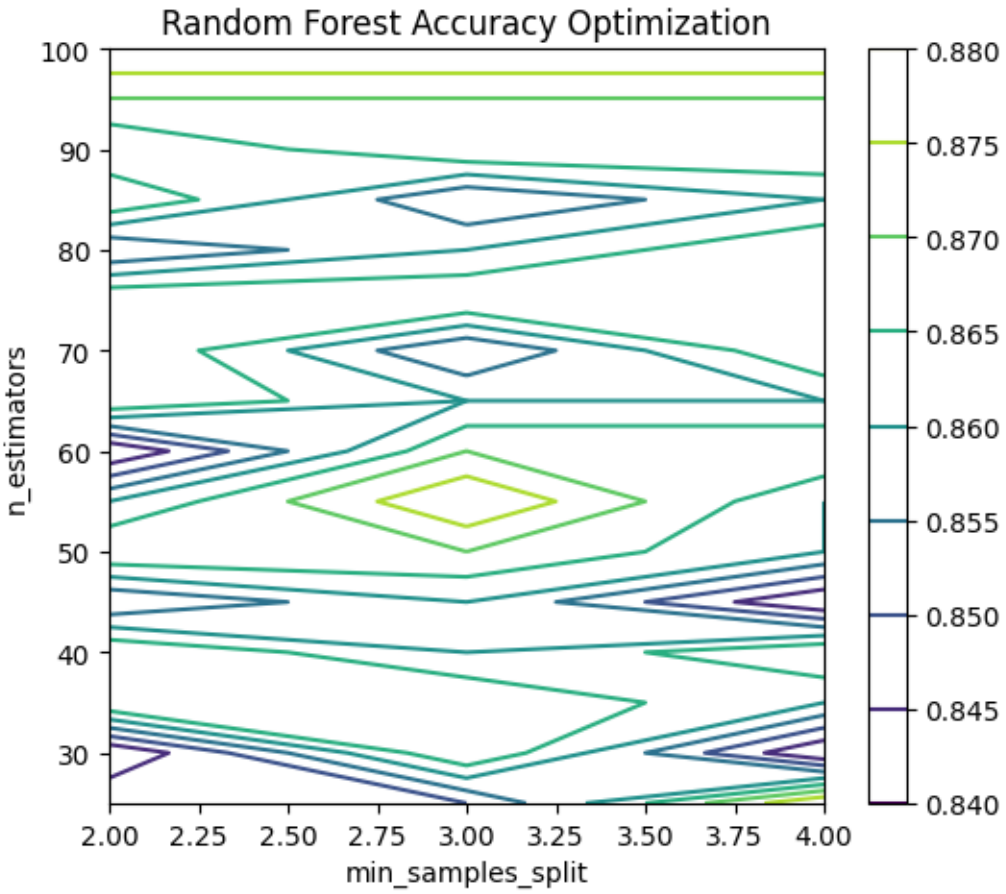

Figure 4.10: A visualization of k-fold validation on random forests optimizing `min_samples_split` and `n_estimators`.

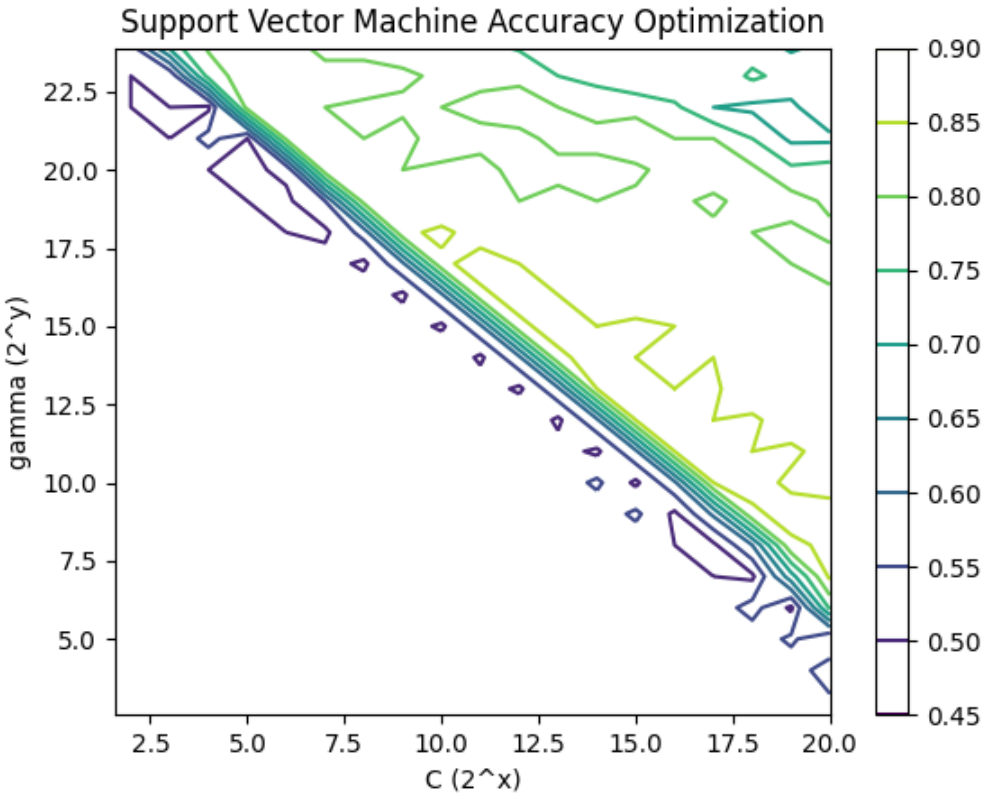

Figure 4.11: A visualization of k-fold validation on SVM optimizing C and γ.

For SVM, the radial basis function kernel is used with a grid search performed on C and γ, where C is varied by 2^C for $0 \leq C < 21$ and γ is varied by 2^γ for $-12 \leq \gamma < 25$. A visualization of this result is shown in Figure 4.11.

For AdaBoost, `n_estimators` is varied by $25x$ for $8 \leq x \leq 16$, and using SVM, naïve Bayes, and decision trees are contrasted as the base estimator. A visual representation of this search result is shown in Figure 4.12.

For voting classifiers, an ensemble of pre-trained models for SVM, logistic regression, LDA, decision trees, random forests, naïve Bayes, and KNN is created. One hard voting classifier and four soft voting classifiers are utilized. Soft voting classifiers used the following weight methods: uniform weights, weights determined by the individual models' training set accuracy (termed Training Set Weights), weights predetermined based upon predicted performance (termed Discrete Weights), and

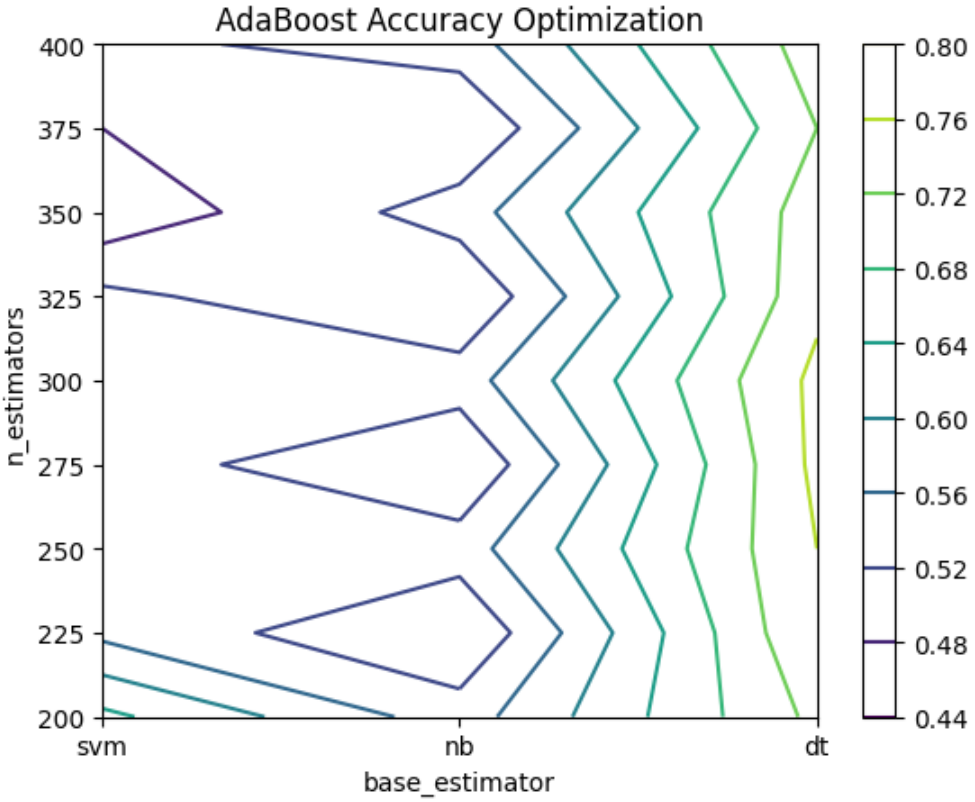

Figure 4.12: A visualization of k-fold validation on AdaBoost optimizing `n_estimators` and contrasting the base estimator.

Model	Discrete	Global
Support Vector Machine	3	0.727
Logistic Regression	1	0.598
Linear Discriminant Analysis	2	0.711
Decision Trees	1	0.729
Random Forests	3	0.791
Naïve Bayes	2	0.675
K-Nearest Neighbors	2	0.722

Table 4.3: The weights used for Discrete and Global Accuracy voting classifiers.

weights predetermined based upon empirical global accuracy of all models within the ensemble (termed Global Accuracy Weights). The Discrete and Global Accuracy weights are shown in Table 4.3.

4.5 Live Analysis

To perform live analysis of EEG data, an OpenBCI EEG headset (Cyton+Daisy 16 Channel data acquisition with Ultracortex Mark IV headset, pictured in Figure 4.13) [42] is procured, and a program is created to display and analyze the live data using Python, Matplotlib, and OpenBCI's API called Brainflow [7].

The OpenBCI headset workflow is comprised of three components: the USB dongle, the on-headset transmitter, and the on-headset data acquisition chip. The USB dongle and on-headset transmitter communicate via Bluetooth connection. The data acquisition chip samples the full set of 16 EEG channels at 125 Hz, or eight EEG channels at 250 Hz.

The program is a multi-processed application: one process to update raw, normalized, and filtered data buffers, one process for each EEG channel to perform FFTs on the respective electrode signal, one process to perform classification using a saved model as well as the data stored in the raw buffer, and one process to update a visualization with the newly procured data. The program presents a plot illustrating the filtered C3 and C4 electrode signals, two spectrograms (one for each electrode), and a bar chart showing the probability of the current state of movement. A picture of the program showing live data is pictured in Figure 4.14.

The data acquisition process actively prompts the OpenBCI headset for any newly acquired data.

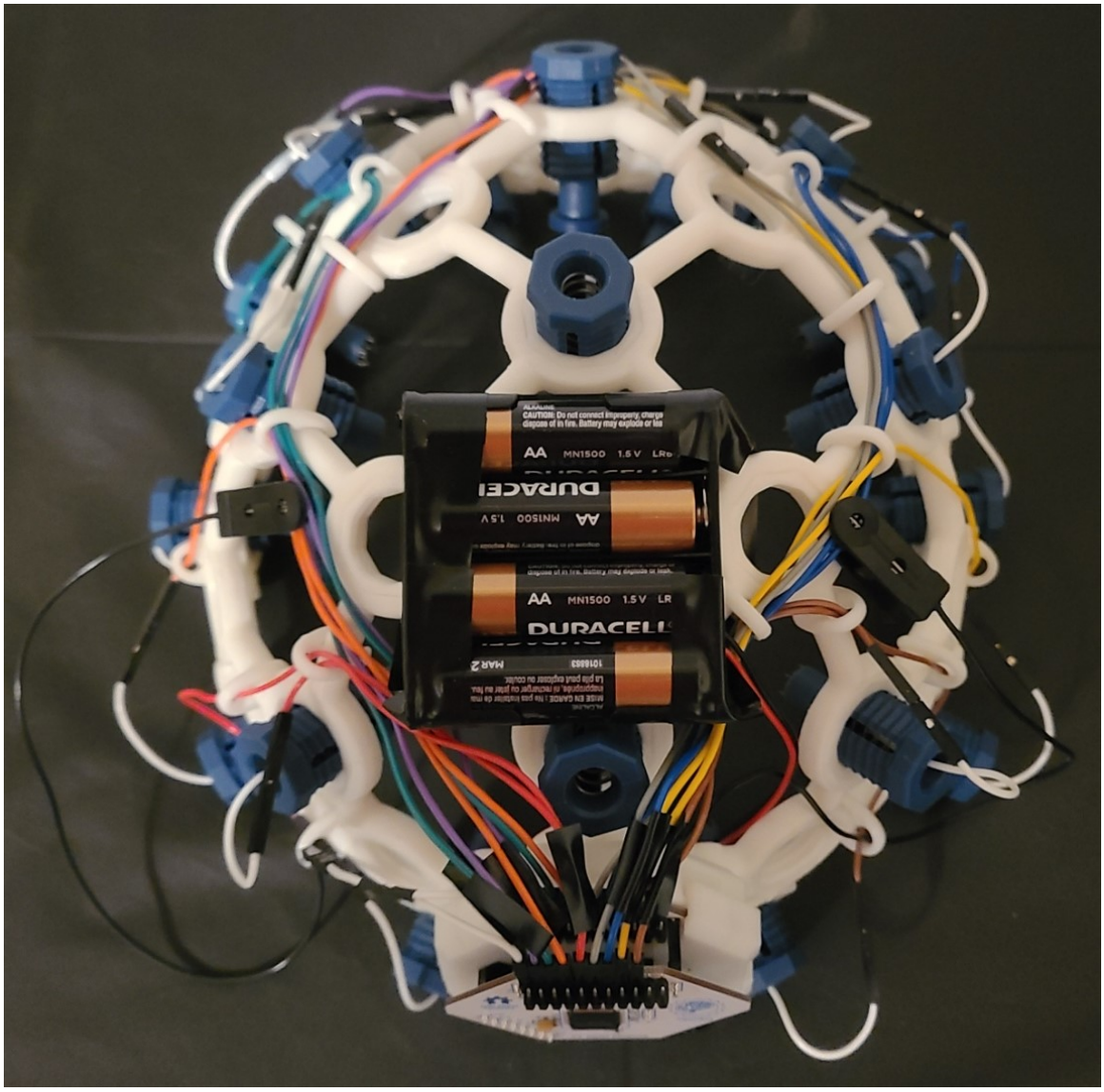

Figure 4.13: A picture of the assembled Cyton+Daisy Board with Ultracortex Mark IV headset from OpenBCI.

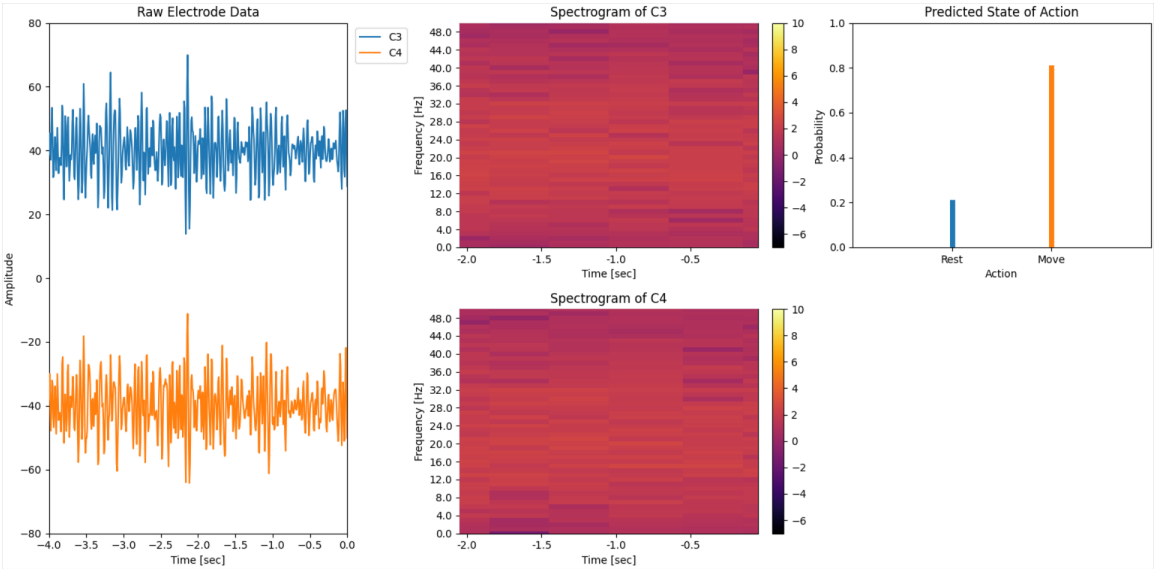

Figure 4.14: A picture of the GUI created in Python to assist in live data analysis.

For a packet of data to be sent to the USB dongle, the circular buffer on the data acquisition chip must contain at least 60 samples. When operating at 125 Hz, packet acquisition and transmission is limited to 0.48 seconds, which in turn limits the update rate of the live data plot. The data buffers are set to hold 16 seconds of data (4,000 samples for 2 channels). The raw data are passed through a 2^{nd}-order Butterworth band-stop filter centered at 60 Hz with a 4 Hz width to reduce influences from the United States power system, then are passed through a 2^{nd}-order Butterworth band-pass filter with a 22.5 Hz center and 17.5 Hz width. This band-pass filter covers 5-50 Hz. This result is stored in the filtered data buffer. This filtered data buffer is what is displayed on the program. While the data buffers contain 16 seconds of data, only four seconds of data are displayed at a single interval. This difference is due to the fact that normalization over the entire signal instead of an epoch of movement achieved higher accuracy on average. While this is not normalization of the entire signal, it acts as an approximation of one.

The spectrogram calculation processes are the same for the C3 and C4 electrodes. The last second of samples on the electrode is acquired, and an FFT is calculated using FFTW 3.0 [16] using either 125 or 250 samples (depending on whether the OpenBCI headset is operating under 8 channel mode or 16 channel mode). The result of this is sent to another buffer storing the last 20 FFT results in a grid in order to generate the spectrogram.

Live classification is performed by first loading a binary file that contains a model that has been pre-trained on the subject wearing the OpenBCI headset using a preliminary training data collection script. The script prompts the user to perform a series of left arm flexions, right arm flexions, and rests over several iterations using the OpenBCI headset to collect annotated data. This annotated data is then processed for normalization, feature-extracted using binned PSD output, then fed to a random forest classifier. While the program is running, normalization occurs over the raw data buffer, and the last four seconds are used to extract binned PSD features, which are then fed into the pre-trained model. The calculated probabilities for each class are saved to a buffer that the animation process reads and displays via a bar chart.

Chapter 5

Results

For all the following tables, SVM corresponds to support vector machines, LR corresponds to logistic regression, LDA corresponds to linear discriminant analysis, DT corresponds to decision trees, RF corresponds to random forests, NB corresponds to naı"ve Bayes, KNN corresponds to k-nearest neighbors. Table 5.1 shows the average accuracy percentages of the different normalization methods across all classifiers. Please note that all accuracies depicted are the classification accuracies on the validation set and no samples in the validation set are used to train any of the models. On average, performing normalization over the entire signal outperforms the lack of normalization and normalization per event epoch. A table of standard deviations is shown in Table 5.2. Due to the higher accuracy performance of normalization over the entire signal, this normalization method is selected.

Table 5.3 depicts the accuracy percentages of different feature extraction methods against the classifier methods with normalization over the entire signal. PSD binned (PSDBIN) by the frequency

Norm	SVM	LR	LDA	DT	RF	NB	KNN	Average
None	45.910	45.455	84.089	45.455	45.455	83.181	75.682	60.747
Epoch	43.408	40.872	84.773	44.772	45.680	82.044	77.274	59.832
All	87.954	54.091	88.410	87.045	89.091	84.773	84.547	82.273

Table 5.1: The average accuracy percentages of varying normalization methods and classifiers in 10 trials on Subject 0029.

Norm	SVM	LR	LDA	DT	RF	NB	KNN	Average
None	7.859	6.062	5.868	6.062	6.062	4.177	5.254	5.906
Epoch	5.298	13.828	3.039	5.031	5.911	4.210	5.463	6.111
All	4.675	19.605	5.815	5.362	3.354	3.869	5.947	6.947

Table 5.2: The standard deviations of varying normalization methods and classifiers in 10 trials on Subject 0029.

Feature	SVM	LR	LDA	DT	RF	NB	KNN	Average
PSD	85.455	44.771	75.911	80.455	87.272	83.181	82.501	77.078
PSDBIN	87.954	54.091	88.410	87.045	89.091	84.773	84.547	82.273
DE	69.590	66.363	66.364	62.728	68.636	66.817	67.499	66.857

Table 5.3: The average accuracy percentages of varying feature extraction methods and classifiers in 10 trials on Subject 0029.

band (delta, theta, alpha, beta, and gamma) outperforms PSD binned by 1 Hz and differential entropy. A table of standard deviations for each average is shown in Table 5.4. Binning the PSD output by frequency band is selected as the primary feature extraction method due to its high performance.

Table 5.5 depicts the average accuracy percentages with normalization over the entire signal and binning PSD by frequency band for feature extraction using traditional classification methods. Decision trees, SVM, and KNN achieve the highest accuracy, followed by LDA. Logistic regression demonstrates the lowest accuracy. The standard deviations for these averages are shown in Table 5.6. SVM, Naïve Bayes, and KNN demonstrate the lowest standard deviation of all these methods, illustrating their high consistency.

Table 5.7 depicts the average accuracy percentages for the ensemble classification methods. Random forests demonstrate the highest classification accuracy, followed by all the voting classifiers, with AdaBoost classifying at the lowest accuracy, though not as poorly as logistic regression. The standard deviations for these averages are shown in Table 5.8. Each ensemble classifier demonstrates similar standard deviations, with the voting classifiers exhibiting the lowest standard deviations, and

Feature	SVM	LR	LDA	DT	RF	NB	KNN	Average
PSD	3.892	4.552	6.620	5.689	3.069	2.874	5.362	4.580
PSDBIN	4.675	19.605	5.815	5.362	3.354	3.869	5.947	6.947
DE	11.907	4.391	5.226	7.816	7.091	3.740	4.916	6.441

Table 5.4: The standard deviations of varying feature extraction methods and classifiers in 10 trials on Subject 0029.

Subject	SVM	LR	LDA	DT	NB	KNN
0028	86.364	83.636	80.910	80.453	83.863	87.047
0029	87.954	54.091	88.410	87.045	84.773	84.547
0040	77.501	43.635	78.410	69.770	65.000	71.817
0041	71.590	71.363	75.000	73.862	69.772	69.999
0042	52.223	55.001	50.278	60.557	51.112	58.052
0043	60.344	51.033	53.792	65.863	50.690	61.724
Average	72.663	59.793	71.133	72.925	67.535	72.198

Table 5.5: The average accuracy percentages of varying subjects and traditional classifiers in 10 trials.

Subject	SVM	LR	LDA	DT	NB	KNN
0028	4.908	3.520	3.891	4.443	3.293	3.562
0029	4.675	19.605	5.815	5.362	3.869	5.947
0040	4.963	4.121	6.449	7.111	8.173	3.894
0041	6.269	7.437	8.902	5.593	6.608	8.559
0042	8.959	6.778	11.672	6.903	4.935	7.235
0043	6.349	9.022	6.338	10.338	8.454	6.594
Average	6.021	8.414	7.178	6.625	5.889	5.965

Table 5.6: The standard deviations of varying subjects and traditional classifiers in 10 trials.

AdaBoost exhibiting the highest standard deviation.

Table 5.9 shows a ranking of all classifiers sorted by their average accuracy.

A table of training time for all the classification methods is shown in 5.10. Random forests demonstrate the highest accuracy as well as a relatively low standard deviation, but take the longest time to train. The act of performing classification was instantaneous.

Training a random forest classifier using the preliminary training data collection script with the OpenBCI headset achieves an 85% accuracy. While this accuracy is promising, external factors

Subject	RF	Vote Trained	Vote Discrete	Vote Global	Vote Uniform	Vote Hard	AdaBoost
0028	85.909	85.455	85.000	85.455	85.456	84.545	83.182
0029	89.091	88.638	89.546	88.865	88.865	88.864	82.047
0040	75.908	80.683	80.000	80.456	80.001	81.137	63.864
0041	81.138	82.046	83.183	82.274	82.501	82.955	78.408
0042	67.779	60.001	60.833	60.834	60.277	56.655	68.055
0043	74.483	60.690	63.796	61.725	62.415	62.757	65.943
Average	79.051	76.252	77.060	76.601	76.586	76.152	73.583

Table 5.7: The average accuracy percentages of varying subjects and ensemble classifiers in 10 trials.

Subject	RF	Vote Trained	Vote Discrete	Vote Global	Vote Uniform	Vote Hard	AdaBoost
0028	2.407	2.668	2.874	2.668	3.069	2.088	4.571
0029	5.295	4.008	3.585	3.625	3.625	4.072	6.814
0040	4.545	3.901	3.354	4.178	4.121	4.552	7.533
0041	5.670	4.726	6.445	4.768	4.916	5.277	4.578
0042	3.940	7.314	9.923	10.430	10.231	7.188	6.834
0043	13.084	8.160	6.348	8.038	8.361	9.451	7.662
Average	5.823	5.130	5.422	5.618	5.720	5.438	6.332

Table 5.8: Standard deviations of varying subjects and ensemble classifiers in 10 trials.

Rank	Classifier	Accuracy
1	Random Forest	79.051
2	Voting - Discrete Weights	77.060
3	Voting - Global Accuracy Weights	76.602
4	Voting - Uniform Weights	76.586
5	Voting - Training Set Weights	76.252
6	Voting - Hard	76.152
7	AdaBoost	73.583
8	Decision Trees	72.925
9	Support Vector Machines	72.663
10	K-Nearest Neighbors	72.198
11	Linear Discriminant Analysis	71.133
12	Naïve Bayes	67.535
13	Logistic Regression	59.793

Table 5.9: A ranking of all tested methods with their averages over all the subjects' data.

Classifier	Seconds
Support Vector Machines	4.5025
Logistic Regression	0.254
Linear Discriminant Analysis	0.0011
Decision Trees	1.0251
Random Forests	12.9011
Naïve Bayes	0.0008
K-Nearest Neighbors	0.0573
AdaBoost	9.289

Table 5.10: The average training time of 100 samples in 10 trials. Timings were performed on an 11th Gen Intel(R) Core(TM) i7-1165G7 processor at 2.80GHz with 16 GB RAM in serial processing. The voting classifier is not listed as it uses pre-trained models.

between the training and live GUI use lead to inconsistent live performance. At rest, the model rapidly oscillates between predicting movement and rest. At movement, the model still oscillates between the two predicted states, but has higher confidence in movement on average.

Chapter 6

Conclusions and Discussion

6.1 Conclusions

Extracting data via PSD binned by frequency bands with a fully normalized signal achieves the highest classification performance. Additionally, the random forest classifier demonstrates the highest accuracy with 79.1% over all the subjects' data used, where logistic regression demonstrates the lowest accuracy at 59.8%. The random forest classifier also demonstrates one of the lowest standard deviations at 5.961, which is comparable to naïve Bayes, the classifier with the lowest score (5.889). Random forests performed well because the several decision trees used will focus on varying features, allowing all of the features to be fully considered. As many trees are developed, certain features will naturally be considered more important than others, which effectively weights the features. Logistic regression performed poorly due to the high dimensionality of the input data and undersampling.

Comparing to the results in the literature, logistic regression performed poor in comparison to the other classifiers (59.8% accuracy), which was expected due to the results from the implementation by Yong and Menon [53]. The researchers do not attribute a specific accuracy to logistic regression, but state that in only 9.5% of the time, logistic regression outperforms linear discriminant analysis (LDA) and support vector machines (SVMs). The performance of LDA (at 71.1% accuracy) was within

reason from the accuracies reported by Yong and Menon (75-81% accuracy) [53] and Rodrigo et. al. (64-68% accuracy). SVM performed within reason (at 72.7% accuracy) compared the accuracies depicted by Yong and Menon (75-81% accuracy) [53] and Bentlemsan et. al. (66% accuracy), but not as well as the implementation done by Mebarkia and Reffad, in which a single SVM and an ensemble of three SVMs achieved 90% and 94% accuracy, respectively [39]. It is believed that this large improvement in Mebarkia and Reffad's classifier is due to the fact that the researchers sought to classify left hand motor imagery from right hand motor imagery compared to the movement and rest classification performed in this thesis. It is hypothesized that left hand motor imagery and right hand motor imagery have a far larger difference in neuronal behavior than either left hand motor imagery from rest or right hand motor imagery from rest. The random forest trained in this thesis achieved a 79.1% accuracy, which is comparable to the implementation by Bentlemsan et. al. which achieved an accuracy of 80% [4]. The voting classifiers trained in this thesis achieved an accuracy range of 76.2-77.1% which is not as strong as the voting classifier created by Khrishna et. al. (86% accuracy) [31]. Similar to the dataset used by Mebarkia and Reffad [39], the dataset used by Khrishna et. al. classifies motor imagery in right arm, left arm, right foot, and left foot without any consideration of rest, which may explain the stronger performance. The AdaBoost classifier trained as part of this thesis achieved an accuracy of 73.6% using decision trees as the classifier base, which is very comparable to the AdaBoost classifiers trained by Gao et. al. which for an SVM base and LDA base achieved an accuracy of 74% and 72% respectively [17].

It is demonstrated that live classification can be achieved although with varying results. Even though the training script achieves high training set accuracy, live classification tends to favor movement rather than rest. When movement occurs, there is an increase in probability of the movement state, but movement is already the predicted class. It is hypothesized that between training the model and executing the live GUI, there are external factors that change, skewing the results (e.g., the electrode orientation on the head and the electrode contact quality).

6.2 Future Work

A more robust analysis of using Convolutional Neural Networks (CNNs) and Long-Short Term Memory (LSTM) models would be useful since they were implemented with poor results during this

project. The model described by Lun et. al. was recreated and trained on the Physionet database, the same as in the paper [36]. In order to overfit the model on a subset of 10 subjects in that dataset, two convolutional and pooling layers had to be removed, otherwise the training accuracy was always limited to at random. It is believed that there are hyperparameters that were not discussed in the paper that would lead to higher training and validation performance.

Due to the ability to perform live classification on EEG data, albeit unrefined, the developed program and feature extraction workflow could be used to contribute to a transcranial direct current stimulation (tDCS) device that factors in current movement state of a Parkinson's disease or chronic stroke patient to administer proper tDCS dosage. Since the proposed device would need to be portable, using a small form factor computing device such as a Raspberry Pi would be preferred. All of the classifiers were able to perform classification instantaneously on the testing computer and all of the code is written in Python (a cross-platform language), therefore it is possible to perform these algorithms on a Raspberry Pi. The main difficulty would be to perform training quickly, as random forests and AdaBoost took 12.9 and 9.3 seconds to train on the testing computer, respectively. If deep learning were to be used, it may be difficult to perform this workflow on the Raspberry Pi due to the absence of a general-purpose graphics processing unit (GPGPU). GPGPUs allow for high throughput linear algebra computation which is necessary for training and using deep learning algorithms in a timely manner. While the Raspberry Pi lacks this technology at the time of writing, there are several mobile devices that have this architecture, such as the NVIDIA Jetson and new smartphones. Using an NVIDIA Jetson or Raspberry Pi as a wearable device for the proposed device is bulky, but due to the fact that the OpenBCI headset uses Bluetooth transmission, an app on a smartphone may suffice for an external processing unit.

Since several physical factors may be an issue for the live classification of movement state, simplification of the headset may be useful to reduce environment variability. For instance, the OpenBCI headset has support for 16 channels, however only two were considered for this research. Simplifying the number of electrodes would allow for a less cumbersome headset apparatus that makes better contact with the scalp and may better fit the skull.

Bibliography

[1] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175, 1992.

[2] Elon Musk and. An integrated brain-machine interface platform with thousands of channels. *Journal of Medical Internet Research*, 21(10):e16194, October 2019.

[3] S Balakrishnama and A Ganapathiraju. Linear discriminant analysis - a brief tutorial. *Institute for Signal and Information Processing*, 1998.

[4] Maouia Bentlemsan, ET-Tahir Zemouri, Djamel Bouchaffra, Bahia Yahya-Zoubir, and Karim Ferroudji. Random forest and filter bank common spatial patterns for eeg-based motor imagery classification. In *2014 5th International Conference on Intelligent Systems, Modelling and Simulation*, pages 235–238, 2014.

[5] Hans Berger. Über das elektrenkephalogramm des menschen. *Archiv für Psychiatrie und Nervenkrankheiten*, 87(1):527–570, December 1929.

[6] Hans Berger. Über das elektrenkephalogramm des menschen. zweite mitteilung. *Archiv für Psychiatrie und Nervenkrankheiten*, 40(1):160–179, 1930.

[7] BrainFlow. Brainflow.

[8] György Buzsáki and Edvard I. Moser. Memory, navigation and theta rhythm in the hippocampal-entorhinal system. *Nature Neuroscience*, 16(2):130–138, 2013.

[9] Rupal Chaudhari and Hiren J Galiyawala. A review on motor imagery signal classification for bci. *Signal Processing: An International Journal (SPIJ)*, 11(2):16, 2017.

[10] Dong Wei Chen, Rui Miao, Wei Qi Yang, Yong Liang, Hao Heng Chen, Lan Huang, Chun Jian Deng, and Na Han. A feature extraction method based on differential entropy and linear discriminant analysis for emotion recognition. *Sensors (Switzerland)*, 19(7), 2019.

[11] Eric S. Donkor. Stroke in the 21st century: A snapshot of the burden, epidemiology, and quality of life. *Stroke Research and Treatment*, 2018:1–10, November 2018.

[12] Ruo Nan Duan, Jia Yi Zhu, and Bao Liang Lu. Differential entropy feature for EEG-based emotion classification. *International IEEE/EMBS Conference on Neural Engineering, NER*, pages 81–84, 2013.

[13] John Ellson, Emden Gansner, Lefteris Koutsofios, Stephen North, Gordon Woodhull, Short Description, and Lucent Technologies. Graphviz — open source graph drawing tools. In *Lecture Notes in Computer Science*, pages 483–484. Springer-Verlag, 2001.

[14] Andreas K Engel and Pascal Fries. Beta-band oscillations—signalling the status quo? *Current Opinion in Neurobiology*, 20(2):156–165, April 2010.

[15] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[16] M. Frigo and S.G. Johnson. The design and implementation of fftw3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

[17] Lin Gao, Wei Cheng, Jinhua Zhang, and Jue Wang. EEG classification for motor imagery and resting state in BCI applications using multi-class adaboost extreme learning machine. *Review of Scientific Instruments*, 87(8):085110, August 2016.

[18] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A. Engemann, Daniel Strohmeier, Christian Brodbeck, Roman Goj, Mainak Jas, Teon Brooks, Lauri Parkkonen, and Matti S. Hämäläinen. MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, 7(267):1–13, 2013.

[19] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 1st edition, 2017.

[20] IE Harmsen, L Milosevic, L Garcia-Dominguez, C Decke, A Fomenko, A Fasano, SK Kalia, M Hodaie, AM Lozano, and NC Rowland. Interhemispheric cortical and cerebellar coupling patterns predict deep brain stimulation efficacy in parkinson's disease. *Paper is unpublished at time of writing but has been accepted for publication.*, 2021.

[21] Irene E. Harmsen, Nathan C. Rowland, Richard A. Wennberg, and Andres M. Lozano. Characterizing the effects of deep brain stimulation with magnetoencephalography: A review. *Brain Stimulation*, 11(3):481–491, May 2018.

[22] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[23] Christoph S. Herrmann, Daniel Strüber, Randolph F. Helfrich, and Andreas K. Engel. EEG oscillations: From correlation to causality. *International Journal of Psychophysiology*, 103:12–21, 2016.

[24] Tin Kam Ho. Random decision forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR '95, pages 278–282, M, 1995. IEEE Computer Society.

[25] Leigh R. Hochberg, Daniel Bacher, Beata Jarosiewicz, Nicolas Y. Masse, John D. Simeral, Joern Vogel, Sami Haddadin, Jie Liu, Sydney S. Cash, Patrick Van Der Smagt, and John P.

Donoghue. Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, 2012.

[26] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. *Department of Computer Science, National Taiwan University*, 2016.

[27] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90, 2007.

[28] H H Jasper. The ten-twenty electrode system of the international federation. *Electroencephalography and clinical Neurophysiology*, 10:371–375, 1958.

[29] HERBERT H. JASPER. ELECTRO-ENCEPHALOGRAPHY. *Archives of Neurology & Psychiatry*, 39(1):96, January 1938.

[30] Bob Kemp and Jesus Olivan. European data format 'plus' (edf+), an edf alike standard format for the exchange of physiological data. *Clinical Neurophysiology*, 114(9):1755–1761, 2003.

[31] D. Hari Khrishna, I. A. Pasha, and T. Satya Savithri. Multi-level voting method to classify motor imagery eeg signals. *ARPN Journal of Engineering and Applied Sciences*, 13(11):3815–3819, June 2018.

[32] Wolfgang Klimesch. Alpha-band oscillations, attention, and controlled access to stored information, 2012.

[33] James Knierim. Motor cortex (section 3, chapter 3) neuroscience online: An electronic textbook for the neurosciences: Department of neurobiology and anatomy - the university of texas medical school at houston, Oct 2020.

[34] Mikhail A. Lebedev and Miguel A. L. Nicolelis. Brain-Machine Interfaces: From Basic Science to Neuroprostheses and Neurorehabilitation. *Physiological Reviews*, 97(2):767–837, mar 2017.

[35] Jorge Leite, Leon Morales-Quezada, Sandra Carvalho, Aurore Thibaut, Deniz Doruk, Chiun-Fan Chen, Steven C Schachter, Alexander Rotenberg, and Felipe Fregni. Surface EEG-Transcranial Direct Current Stimulation (tDCS) Closed-Loop System. *International journal of neural systems*, 27(6):1750026, sep 2017.

[36] Xiangmin Lun, Zhenglin Yu, Tao Chen, Fang Wang, and Yimin Hou. A Simplified CNN Classification Method for MI-EEG via the Electrode Pairs Signals. *Frontiers in Human Neuroscience*, 14(September):1–14, 2020.

[37] C. Marras, J. C. Beck, J. H. Bower, E. Roberts, B. Ritz, G. W. Ross, R. D. Abbott, R. Savica, S. K. Van Den Eeden, A. W. Willis, and Cm Tanner. Prevalence of Parkinson's disease across North America. *npj Parkinson's Disease*, 4(1):21, dec 2018.

[38] Dennis J. McFarland, William A. Sarnacki, and Jonathan R. Wolpaw. Electroencephalographic (EEG) control of three-dimensional movement. *Journal of Neural Engineering*, 7(3), 2010.

[39] Kamel Mebarkia and Aicha Reffad. Multi optimized SVM classifiers for motor imagery left and right hand movement identification. *Australasian Physical and Engineering Sciences in Medicine*, 42(4):949–958, 2019.

[40] Th. Mulder. Motor imagery and action observation: cognitive tools for rehabilitation. *Journal of Neural Transmission*, 114(10):1265–1278, June 2007.

[41] C. Neuper and G. Pfurtscheller. Event-related dynamics of cortical rhythms: Frequency-specific features and functional correlates. *International Journal of Psychophysiology*, 43(1):41–58, dec 2001.

[42] OpenBCI. Openbci all-in-one biosensing r&d bundle.

[43] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[44] Python Core Team. *Python: A dynamic, open source programming language*. Python Software Foundation, 2019.

[45] Miguel Rodrigo, Luis Montesano, and Javier Minguez. Classification of resting, anticipation and movement states in self-initiated arm movements for EEG brain computer interfaces. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, pages 6285–6288, 2011.

[46] N.C. Rowland, C. De Hemptinne, N.C. Swann, S. Qasim, S. Miocinovic, J.L. Ostrem, R.T. Knight, and P.A. Starr. Task-related activity in sensorimotor cortex in Parkinson's disease and essential tremor: Changes in beta and gamma bands. *Frontiers in Human Neuroscience*, 9(September), 2015.

[47] Joni N. Saby and Peter J. Marshall. The utility of eeg band power analysis in the study of infancy and early childhood. *Developmental Neuropsychology*, 37(3):253–273, 2012. PMID: 22545661.

[48] Zied Tayeb, Juri Fedjaev, Nejla Ghaboosi, Christoph Richter, Lukas Everding, Xingwei Qu, Yingyu Wu, Gordon Cheng, and Jörg Conradt. Validating deep neural networks for online decoding of motor imagery movements from eeg signals. *Sensors (Switzerland)*, 19(1), 2019.

[49] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[50] W.Grey Walter. The location of cerebral tumours by electro-encephalography. *The Lancet*, 228(5893):305–308, 1936. Originally published as Volume 2, Issue 5893.

[51] P Welch. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE trans. audio electroacoust.*, 15(2):70–73, June 1967.

[52] Jonathan R Wolpaw and Dennis J McFarland. Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans. *Proceedings of the National Academy of Sciences of the United States of America*, 101(51):17849–17854, dec 2004.

[53] Xinyi Yong and Carlo Menon. EEG classification of different imaginary movements within the same limb. *PLoS ONE*, 10(4):1–24, 2015.