Channel Islands

CALIFORNIA STATE UNIVERSITY

Combining EEG, EMG and IMU to build a functional Brain Computer Interface

A Thesis Presented to

The Faculty of the Computer Science Department

In (Partial) Fulfillment

of the Requirements for the Degree

Masters of Science in Computer Science

by

Student Name:
Adan Sanchez

Advisor:
Dr. Jason Isaacs

May 2021

APPROVED FOR MS IN COMPUTER SCIENCE

May 27, 2021

Advisor: Dr. Jason Isaacs — Date

May 27, 2021

Dr. Bahareh Abbasi — Date

May 27, 2021

Dr. Scott Feister — Date

APPROVED FOR THE UNIVERSITY

Jill Leafstedt (May 28, 2021 12:34 PDT)

5/28/2021

Dr. Jill Leafstedt — Date

Non-Exclusive Distribution License

In order for California State University Channel Islands (CSUCI) to reproduce, translate and distribute your submission worldwide through the CSUCI Institutional Repository, your agreement to the following terms is necessary. The author(s) retain any copyright currently on the item as well as the ability to submit the item to publishers or other repositories.

By signing and submitting this license, you (the author(s) or copyright owner) grants to CSUCI the nonexclusive right to reproduce, translate (as defined below), and/or distribute your submission (including the abstract) worldwide in print and electronic format and in any medium, including but not limited to audio or video.

You agree that CSUCI may, without changing the content, translate the submission to any medium or format for the purpose of preservation.

You also agree that CSUCI may keep more than one copy of this submission for purposes of security, backup and preservation.

You represent that the submission is your original work, and that you have the right to grant the rights contained in this license. You also represent that your submission does not, to the best of your knowledge, infringe upon anyone's copyright. You also represent and warrant that the submission contains no libelous or other unlawful matter and makes no improper invasion of the privacy of any other person.

If the submission contains material for which you do not hold copyright, you represent that you have obtained the unrestricted permission of the copyright owner to grant CSUCI the rights required by this license, and that such third party owned material is clearly identified and acknowledged within the text or content of the submission. You take full responsibility to obtain permission to use any material that is not your own. This permission must be granted to you before you sign this form.

IF THE SUBMISSION IS BASED UPON WORK THAT HAS BEEN SPONSORED OR SUPPORTED BY AN AGENCY OR ORGANIZATION OTHER THAN CSUCI, YOU REPRESENT THAT YOU HAVE FULFILLED ANY RIGHT OF REVIEW OR OTHER OBLIGATIONS REQUIRED BY SUCH CONTRACT OR AGREEMENT.

The CSUCI Institutional Repository will clearly identify your name(s) as the author(s) or owner(s) of the submission, and will not make any alteration, other than as allowed by this license, to your submission.

Combining EEG, EMG and IMU to build a functional Brain Computer Interface

Title of Item

Electroencephalogram; Electromyogram; IMU; Machine learning; Brain-Computer Interface

3 to 5 keywords or phrases to describe the item

Adan M. Sanchez

Author(s) Name (Print)

5/26/2021

Author(s) Signature Date

Combining EEG, EMG and IMU to build a functional Brain Computer Interface

Adan Sanchez

May 24, 2021

Abstract

A Brain-Computer Interface (BCI) is a direct connection between a computer and the brain - in this case, the human brain. Invasive BCIs tend to perform better as the amount of noise decreases when electrodes are placed under the scalp. However, placing the electrodes inside the skull increases the complexity of the device and risk to the user. Non-invasive BCIs are a viable alternative to this. Furthermore, the noise encountered when placing the device outside of the skull can be filtered out by modern machine learning algorithms. In this study, the EMOTIV Epoc+ headset, equipped with 14 surface electrodes and 2-axis motion sensors, was used to gather EEG, EMG and IMU data from a user. The machine learning algorithms tested were Logistic Regression, Decision Trees, Adaptive Boosting, Random Forests, Gradient Boosting Trees, Support Vector Machines and a Multilayer Perceptron Neural Network. Their performance is presented and directly compared along with different combinations of EEG, EMG, and IMU data being fed into them. Several viable combinations were discovered, some of which yielded accuracy of up to 95%. These results imply the implementation of a versatile BCI is feasible using machine learning algorithms and a consumer grade EGG headset.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction to BCI

Human-computer interaction is concerned with the design and use of novel interfaces between people and computers. One specific instance of this is a Brain-Computer Interface (BCI), sometimes referred to as a neural-control interface (NCI) or mind-machine interface (MMI). This technology provides a direct link between an external device and the brain. Often, BCIs aim to provide biotechnological solutions for physical ailments, such as paralysis, by integrating the brain with assistive devices [1].

BCIs can be mostly categorized as invasive, partially invasive, and non-invasive. Invasive and partially invasive methods tend to require surgery in order to implant a device or part of a device into the user. This can create a great risk to the user and increase the complexity of working with the BCI. Presently, non-invasive BCIs tend to rely on electroencephalography (EEG) to acquire data from the user through wearable devices such as headbands or earbuds.

This paper seeks to test and implement a BCI that uses other sensors, such as inertial measurement units (IMU) or electromyograms (EMG), in conjunction with EEG in order to produce a more efficient BCI. An advantage of such an interface would be that it would be able to operate in multiple paradigms which would make it very versatile [2]. For example, this device could be easily adapted to meet the needs of someone who has limited use of their limbs but still has full control over the muscles around the neck and head. Machine learning algorithms can also be used in to minimize noise and normalize the process of data acquisition methods for different individuals [3] [4] [5]. It is for this purpose that the data collected in this experiment will be used to train a number of different machine learning algorithms.

Ideally, every machine learning algorithm should aim for 100% precision. Realistically, this is not possible even in the best of cases. Comparing proposed BCIs to each other can be challenging as well, as they do not all use the same data set or operate under the same conditions. Thus, in order to measure the feasibility of the BCI proposed in this paper we must first establish what it would take for it to be feasible.

In this context, feasibility means that the BCI would be precise, robust and able to be implemented under various circumstances. A precise BCI would achieve significantly higher-

than-random precision (>70%) precision. A robust BCI would mean an equally high recall as we want to minimize the number of misclassified states. Being able to be implemented under various circumstances would mean using the EEG sensor in conjunction with either EMG and/or IMU in order to achieve the desired results.

1.2 Challenges and Contributions

Obtaining meaningful data from the brain, noninvasively, can be challenging as different factors, such as skull thickness, hair density, different brains or even different eye movements, can produce noise which hinders the ability to produce clear, concise commands that a computer can execute [6]. Additionally, many proposed BCIs target severely disabled individuals while using abled-body subjects to develop them [7] which can hinder their real-world application. Even so, BCIs that purely rely on EEG have already been produced with limited results. In [8], we see how a number of existing BCIs rely on only one type of suitable brain signals. This can greatly reduce their usability as some users cannot produce the specific brain activity patterns required for these systems. Thus, the need for a BCI with greater usability in different paradigms arises.

The contribution of this paper is to provide methodologies for a BCI that is able to operate within different paradigms through the use of different sensors in addition to EEG in an effort to reduce the reliability on a single type of brain signal. Different machine learning algorithms and techniques are also evaluated in order to increase the efficacy of the BCI.

1.3 Paper Outline

The remainder of this paper is structured as follows: In Chapter 2, a brief overview of the different data collection techniques is given. Also provided in this chapter is an overview of the machine learning techniques used in this paper. Chapter 3 revolves around the data used in this paper. In this chapter, the data collection methods are discussed, an overview of the data set acquired is given and the pre-processing techniques used are presented. In Chapter 4, the implementation of the machine learning algorithms is shown as well as the results in different paradigms with each algorithm. The overall results from the tests are discussed in Chapter 5 and the meaning of these results is explored in Chapter 6. A conclusion and a discussion of future work is given in Chapter 7.

2 Background

2.1 Electroencephalogram (EEG)

Ionic current within the neurons of our brain result in voltage fluctuations. Although subtle, these can be measured through the use of the electrophysiological monitoring method known as electroencephalography [9]. Hans Berger, a German psychiatrist in the 20th century, was the first to record and analyze EEG signals from a human and the first to describe abnormal EEGs in neurological diseases [10]. He identified the oscillatory activity of these brain waves with devices for detecting small electric currents, first a string galvanometer and later a Siemens double-coil galvanometer [11]. Today's EEG detectors are more advanced than the galvanometers used by Berger one hundred years ago. So much so in fact, that it is a leading method used in BCIs [12].

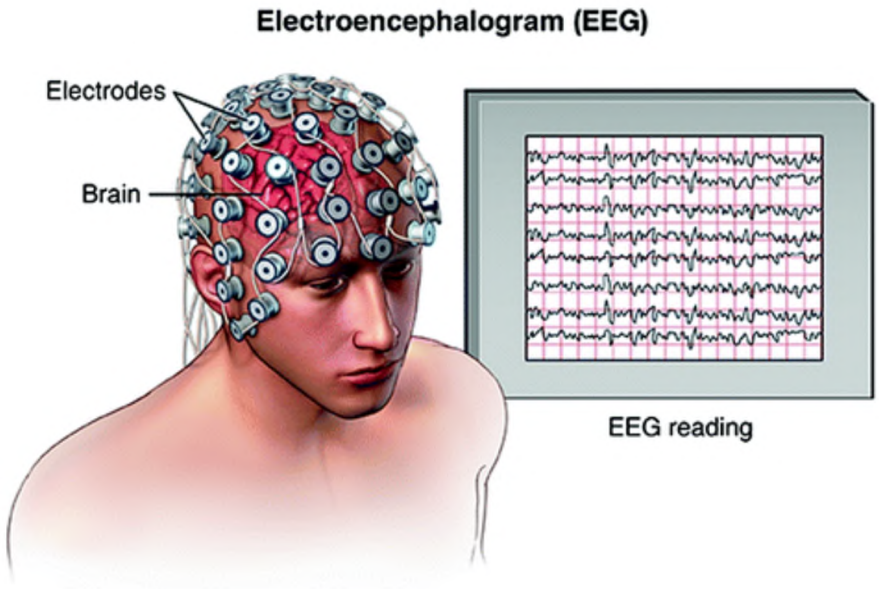


Figure 1: Example of EEG Headset [13]

2.2 Electromyograms (EMG)

Electromyography could most aptly be characterized as an electrodiagnostic medicine technique that measures electrical activity in response to a nerve's stimulation of a muscle [14]. The term electromyography was first introduced by French scientist Étienne-Jules Marey in 1890 [15]. However, experiments using EMG date back to 1666 when the first documented

experiments on electric eels by Italian scientist Francesco Redi were conducted. In present day, specifically in Computer Science, we have used EMG as form of human-computer interaction and has even been successfully used in some BCIs. In [16], we see that EMG can be viable for detecting movement intention, with 22/30 of their patients showing sufficient surface EMG in their finger/wrist extensor muscles. Although in this paper they developed an EMG signal detector without the use of machine learning, it proves that it can be successfully used to detect movement intention in conjunction with EEG. In Figure 2, we can see an EMG device capturing muscle activity of superficial masseter and anterior temporalis bilaterally.

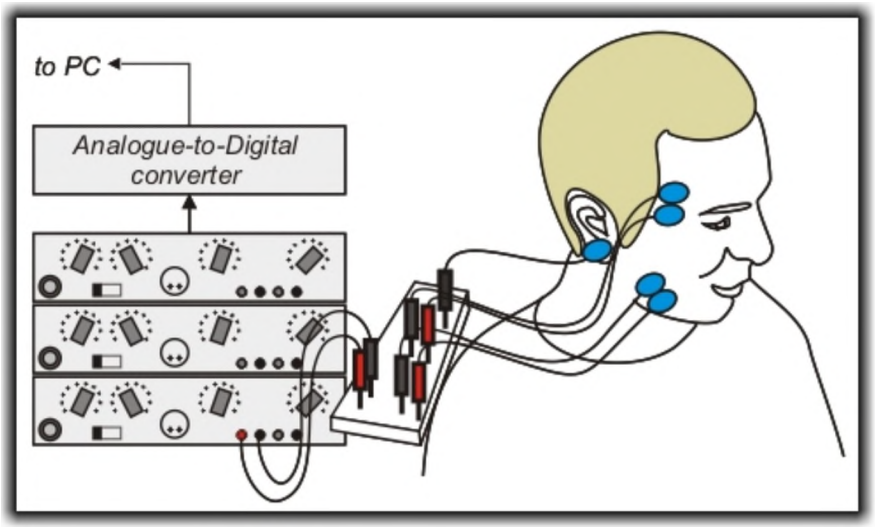


Figure 2: EMG Device [17]

2.3 Inertial Measurement Unit (IMU)

An inertial measurement unit is a device that can measure the specific force, angular rate and/or orientation of a body through the use accelerometers or gyroscopes [18]. What a specific IMU can measure and report varies by model but in general, they all serve to electronically provide a sense of spacial awareness. While often incorporated into navigational systems to calculate attitude, angular rates, linear velocity or even the position relative to a global reference frame, they also serve as orientation sensors in many other products. One of these use cases is in BCIs, often in conjunction with EEG. In [19], we see several successful uses of IMUs, mostly with Kalman filter based algorithms, in order to classify/predict motions of a user. An example of one of an IMU device can be seen in Figure 3.

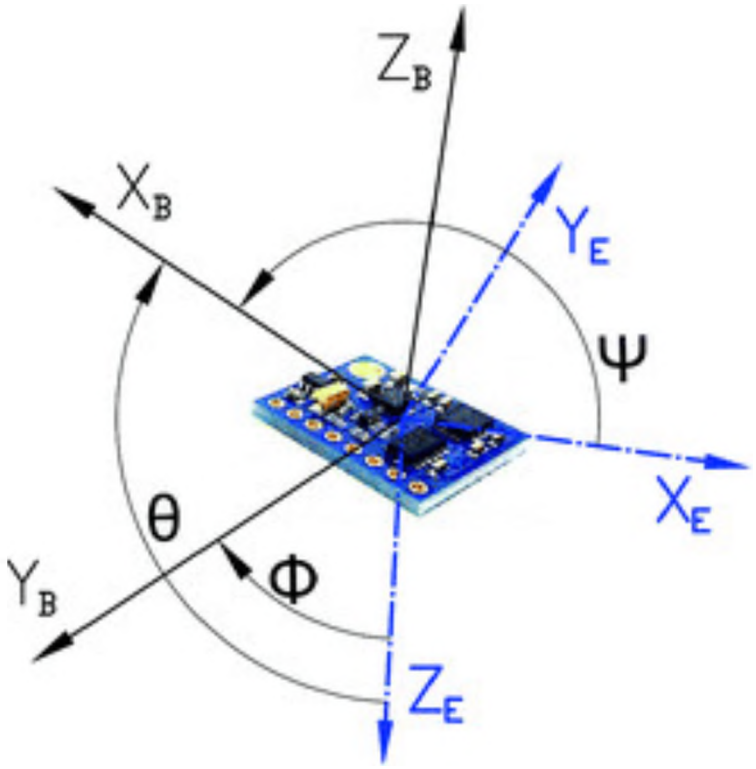


Figure 3: IMU Device [20]

2.4 Machine Learning Techniques

Although relatively young, the machine learning field is rapidly growing. A great number of machine learning algorithms and their variations now exists. Far too many to test individually. Thus, a limited number had to be chosen. The algorithms seen here were chosen due to their robustness and widely available documentation.

2.4.1 Logistic Regression

Logistic regression, in its simplest form, is a statistical method that models a binary dependent variable using a logistic function. This is often used to model the probability of a certain event or class existing, e.g. pass/fail, win/loose. Models with increased complexity can be extended to model multiple classes or events at a time [21]. In image recognition, such a method could be used to model the probability, between 0 and 1, of a certain object being in a given picture.

An example of the logistic function,

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}}, \tag{1}$$

where L, the curve's maximum value; k, the logistic growth rate or steepness of the curve; x_0, the x value of the sigmoid's midpoint, can be seen in Figure 4.

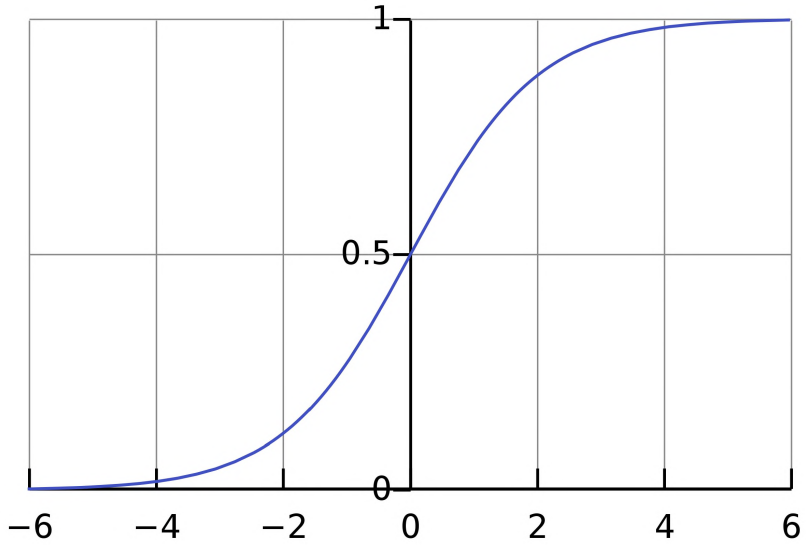


Figure 4: Standard logistic sigmoid function where $L = 1$, $k = 1$, $x_0 = 0$

However, multi-class classification means that there are more than two classes to classify. Thus, a One-vs-Rest (OvR) approach is used when implementing logistic regression. This approach splits the data set into multiple binary classification problems which allows a binary classifier to be trained for each problem created.

2.4.2 Decision Trees

A decision tree is a tree-like model which aims to predict the value of a target variable by learning simple decision rules inferred from the data features. In this case, a decision tree is used to classify different classes. A tree model will usually consist of internal nodes, which represent an attribute of the model, branches, which represent a test outcome and a leaf node which represents a class label. In decision tree learning, branches represent feature conjunctions which lead to class labels known as leaves. Most decision tree learners, such as the one used in this paper, are deterministic which means that given a fixed data set,

they produce a tree with the same structure. An outline of a decision tree model can be seen in Figure 5. These models are quite popular in machine learning due to their relative simplicity and ease of implementation [22]. However, a disadvantage of decision trees is they could create over-complex trees that do not generalize data as well.

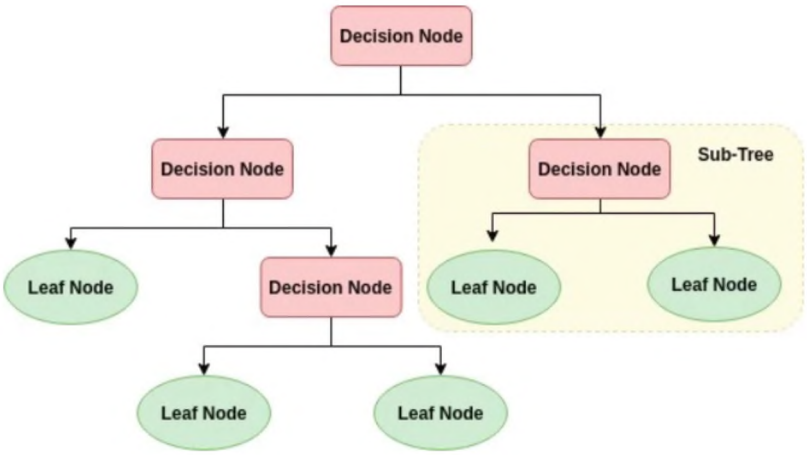


Figure 5: Decision Tree model outline [23]

2.4.3 AdaBoost

AdaBoost, or Adaptive Boosting, is what is known as a meta-estimator. It basically combines multiple weaker classifiers into a stronger one. A weak model is one which may perform better than random guessing but not by a lot. A feature of AdaBoost is that it can be applied on top of almost any given classifier in order to learn from the mistakes of the weaker model and provide a stronger model in the end. In this specific case, AdaBoost is paired with a model that specifically targets multi-class classification referred to as Stagewise Additive Modeling using a Multi-class exponential loss function or AdaBoost-SAMME for short. This model is outlined in section 1.2 of [24]. An overview of the AdaBoost method can be seen in Figure 6.

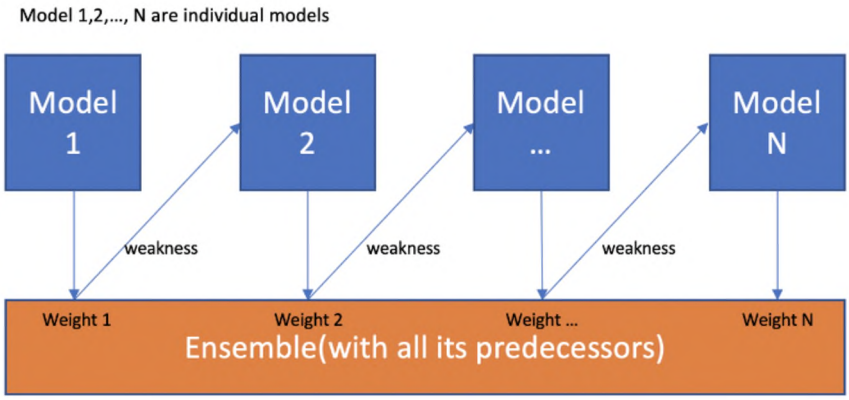


Figure 6: AdaBoost method outline [25]

2.4.4 Random Forests

Random forests are an ensemble learning classification method. As the title implies, this method consists of a large number of individual decision trees. Each tree model, generally all with equal weight, produces a class prediction and the class with the most votes becomes the model used for a future prediction with each model. We can see an example of this in Figure 7. One of the main advantages of this method versus just using Decision Trees is that Random Forests can correct for overfitting which tends to occur in Decision Trees. An advantage of this method is that weaker models can be built concurrently and independently of others, thus speeding up training time.

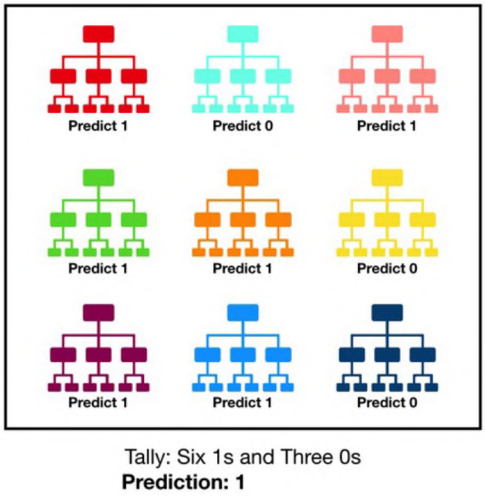


Figure 7: Random Forests making a prediction [26]

2.4.5 Gradient Boosting Trees

Gradient Boosting Trees or Gradient Boosting for short is another ensemble classification technique. This technique is essentially a version of Adaptive Boosting (2.4.3) where the weak classifier is a Decision Tree, hence the *trees* in its name. Boosting refers back to the idea of using a weaker classifier to build a stronger one [27]. This technique differs from Random Forests in two main areas: the way trees are built and the way results are combined. Gradient Boosting builds trees one at a time giving it the ability to correct errors made by previous trees. A weakness of this algorithm is that it can be easier to overfit than regular Random Forests. However, through careful tuning of parameters this can be avoided.

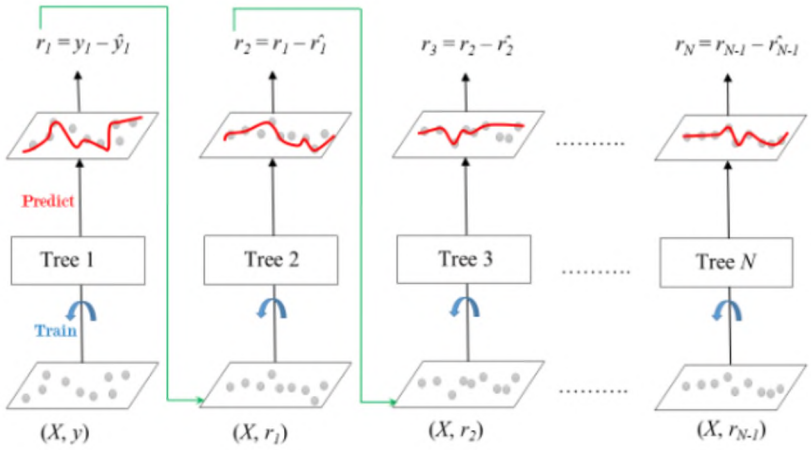


Figure 8: Gradient Boosting [28]

2.4.6 Support Vector Machines

Support Vector Machines (SVM) aim to distinctly classify N classes, or features, by finding a hyperplane in the N-dimensional plane [29]. Hyperplanes can be simply thought of as boundaries between classes as seen in Figure 10. In 2-dimensions, a hyperplane would simply be a line and in 3-dimensions it could be thought of as a plane. However, SVM allows us to extrapolate this concept to N-dimensions but it becomes difficult to visualize the hyperplane when $N > 3$. Points closer to the hyperplane are referred to as the support vectors which help position and orient the hyperplane. As the margin between these points increases, we maximise the margin of our model.

In Figure 9, this can be conceptualized for 2D and 3D dimensions.

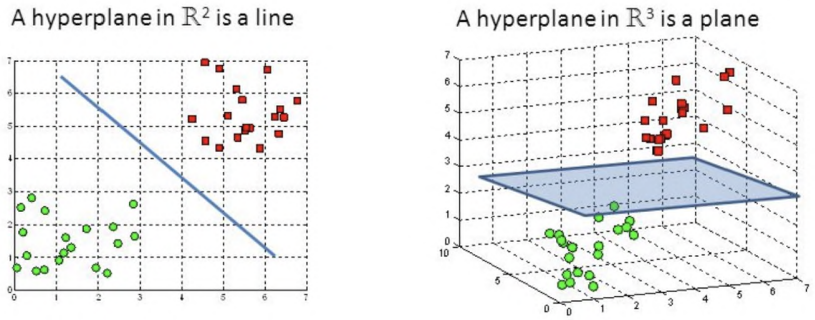


Figure 9: Hyperplane visualized [30]

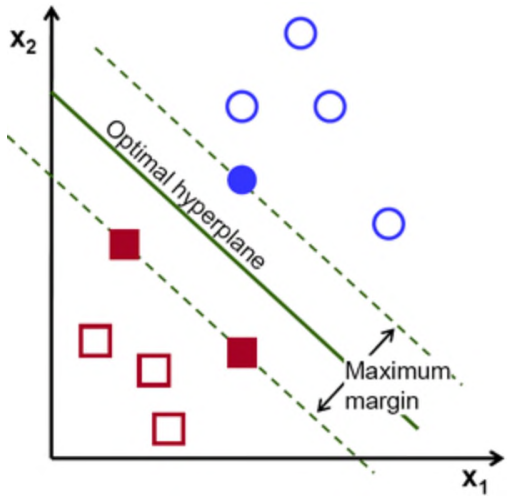


Figure 10: Hyperplane margin in SVM model [30]

2.4.7 Neural Networks

Neural Networks, formally known as Artificial Neural Networks, are sets of interconnected artificial neurons inspired by neurons in a brain. This term is an umbrella term that encompasses many types of neural networks. In this case, we will be using a Multilayer Perceptron (MLP) neural network which consists of at least three layers of nodes. These include an input layer, a hidden layer and an output layer, although more complex models can exist. MLPs generally achieve good generalisation of unseen data, specifically in cases where full theoretical models cannot be constructed [31]. This makes an MLP neural network a good candidate for this application.

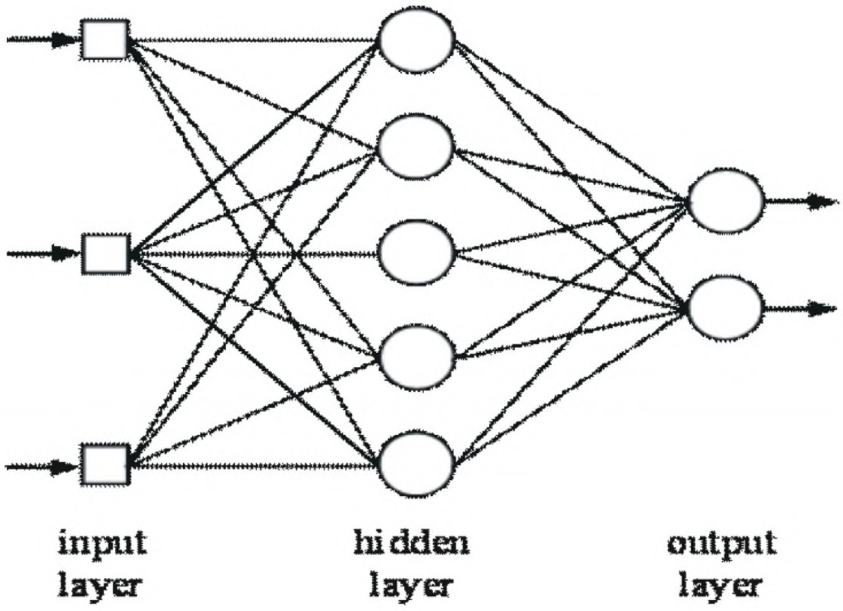


Figure 11: Outline of neural network [32]

3 Experiment Set Up

3.1 Subjects

While conducting this research, a worldwide pandemic took place. This pandemic was caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV2). Given the nature of the virus, close proximity to others was extremely limited. As with almost everything at the time, this experiment was ultimately affected. Social distancing measures made the feasibility of obtaining data from a large pool of subjects impossible. Therefore, all data used in this experiment was collected from its author as that proved to be the safest and most ethical way of completing this research. The data was collected over a period of two months and consisted of 100 sessions. Each session lasted around 3 minutes.

3.2 Data Collection

The headset used in this experiment is the Emotiv Epoc+ as seen in Figure 12. This headset was chosen due to its relative low cost and level of accuracy at its price range [33]. The headset is equipped with 14 electrodes as well as a gyroscope. The location of which can be seen in Figure 13 and Figure 14. EEG electrode location can be placed on the following places: pre-frontal (Fp), frontal (F), temporal (T), parietal (P), occipital (O), central (C), and between Fp and F (AF). Their lateralized location is marked as follows: odd numbers (1,3,5,7) refer to electrodes placed on the left hemisphere, even numbers (2,4,6,8) refer to those on the right hemisphere.

Using these electrodes, the headset is able to capture 14-channel electroencephalography (EEG) data with a rate of up to 128 Hz. Additionally, it is also able to capture gyroscopic data in two axes (X and Y or left/right and up/down from the perspective of the user) at the same resolution. The location of this gyroscope can be found behind the Front Power/Charging Indicator as seen in Figure 12. Using triangulation, the headset is also able to generate electromyography (EMG) data at a resolution of 32 Hz. It is also able to capture alpha, low beta, high beta, gamma and theta bands at a resolution of 8 Hz using proprietary algorithms. For the purposes of this experiment, only raw EEG, raw gyroscopic data and EMG data will be used.

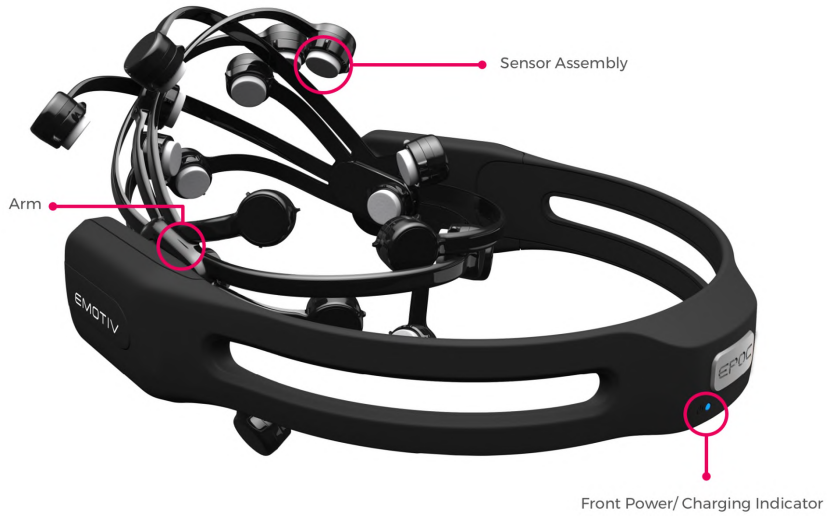


Figure 12: Emotiv EPOC Headset [34]

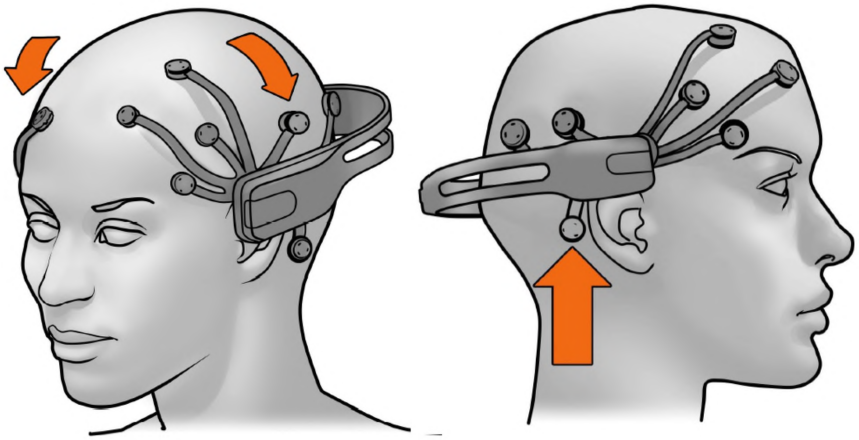


Figure 13: EEG headset placement [35]

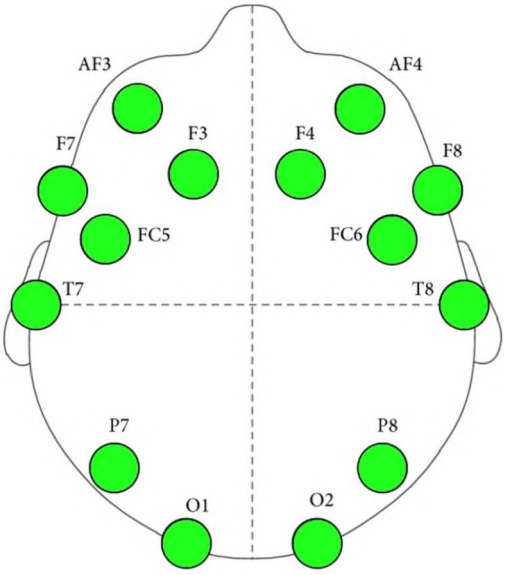
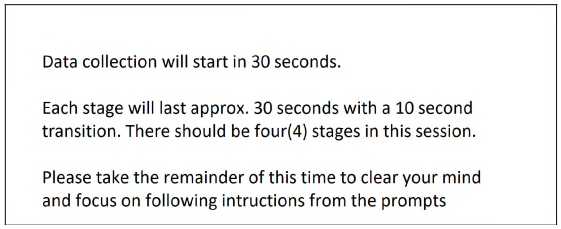


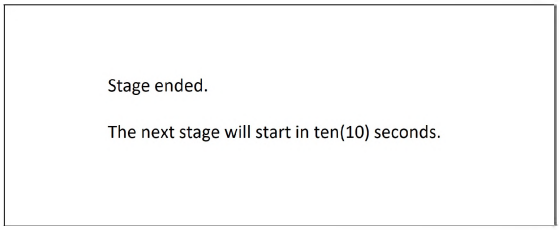
Figure 14: Headset Sensor Mapping [36]

3.3 Methods

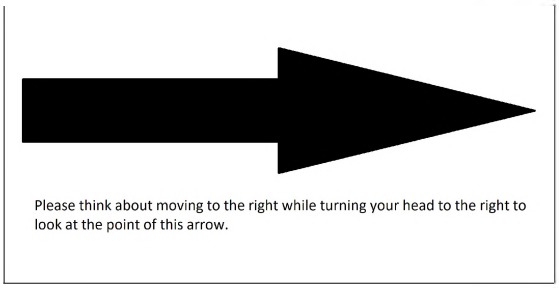
This experiment aims to test whether a discrete number of pre-determined actions can be predicted from the data collected from a user. In this case, there are 4 pre-determined actions that the user is instructed to perform. The actions consist of looking in a given direction (up, down, left, right) while simultaneously thinking of moving in that direction. The user is told to follow these instructions for given period of time in order to allow the headset to collect enough data to obtain an accurate state of mind (see Figure 15). Thirty seconds was chosen as the given time interval to collect data for each direction as a user could become distracted if they are not stimulated for long periods of time.



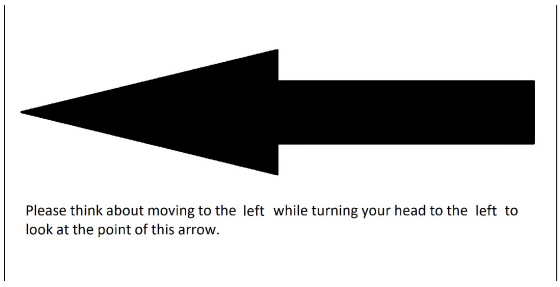
(a) Start



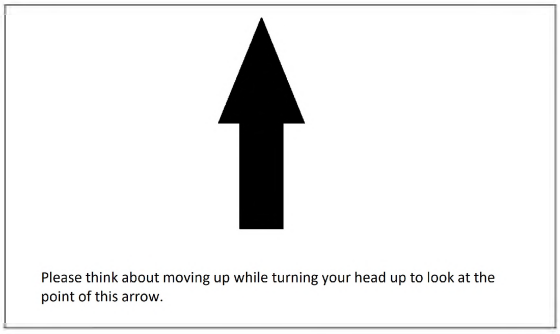
(b) Transition



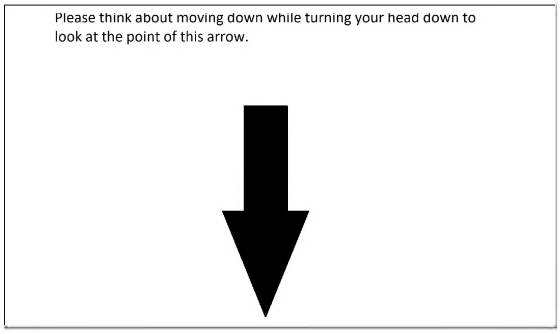
(c) Right



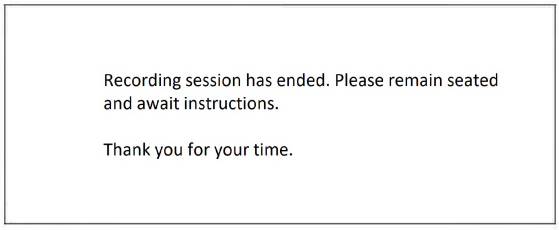
(d) Left



(e) Up



(f) Down



(g) End

Figure 15: Prompts displayed while capturing data.

While there are only 5 different actions we would like to segregate from the data collected, there are in fact 6 total states the user can find themselves in during the data collection process. These include a baseline state, a transition state as well as the four different action states. The transition state is only there to ease the user's transition from one state to another. It is discarded when processing the data for training. The process of collecting data begins when the user is sat in front of a computer screen while wearing the headset. After the user makes themselves comfortable in front of the screen, the process starts when a prompt displayed on the screen tells them to try and clear their minds for the next 30 seconds.

After this, the first action prompt appears on screen telling them to look in the given direction while tilting their head so that they directly face the tip of the arrow displayed on screen. This is so that we may obtain the greatest amount of head movement so that the gyroscope sensor in the headset will be able to collect data. In order to keep the amount of rotation constant between runs and subjects, the size of screen should remain constant while collecting data. Before the next action is given, a 10 second transition slide is displayed on screen so that the user may be prepared for the next action. This is repeated 3 more times to total 180 seconds of data. We obtain approximately 128 data points of EEG and gyroscopic data per second and 32 points of EMG data per second. Each session, the order of the action prompts is randomized so that the user may not memorize a pattern and loose focus.

3.4 Data Analysis

3.4.1 EEG Data

We can visualize the data of a given session by plotting it over time. In Figure 16, we can see a time series representation for all 14 EEG channels where time on the X-axis is measured in seconds and the Y-axis is measured in microvolts. Here, 6 different states are labeled and separated by a dotted vertical line. Each line is labeled according to the location of the electrode in the head of the user as seen in Figure 14. This allows us to see certain patterns that occur within our data. For example, we tend to see relative big spikes whenever we move from one state to another.

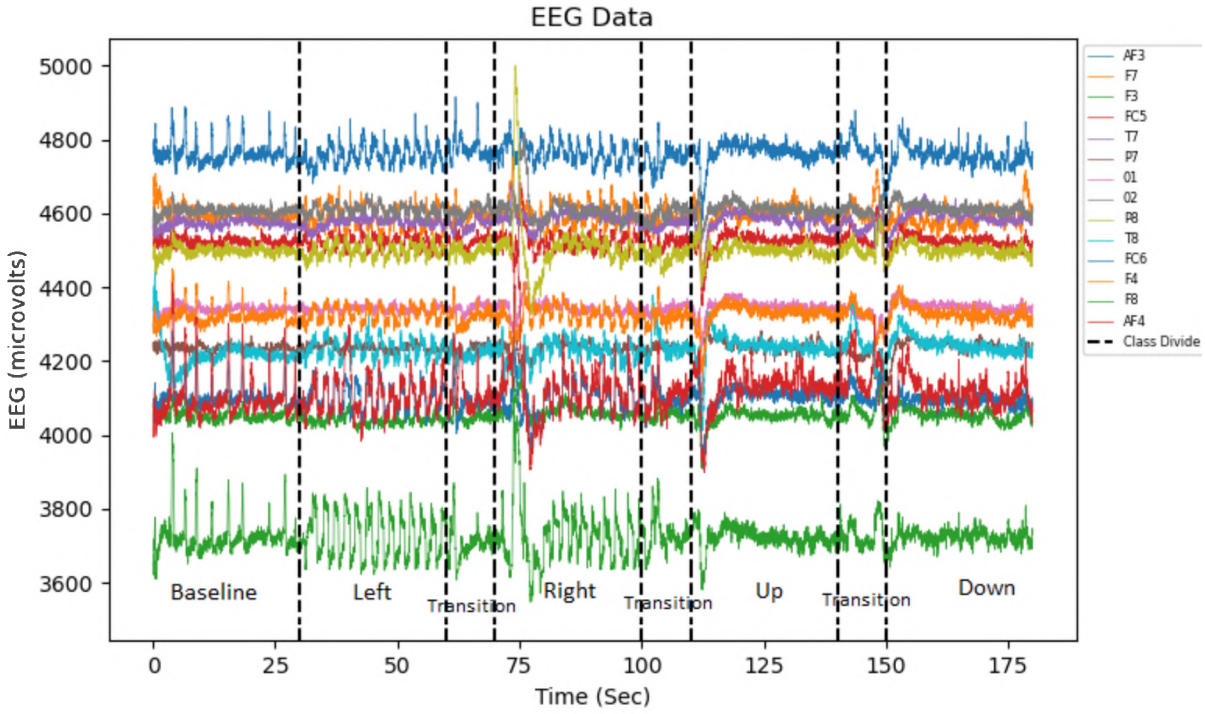


Figure 16: EEG vs Time

3.4.2 IMU Data

After graphing our 2 channels of IMU data (X and Y axis position), we see similar results in Figure 17. We also see big spikes when moving from one state to another.

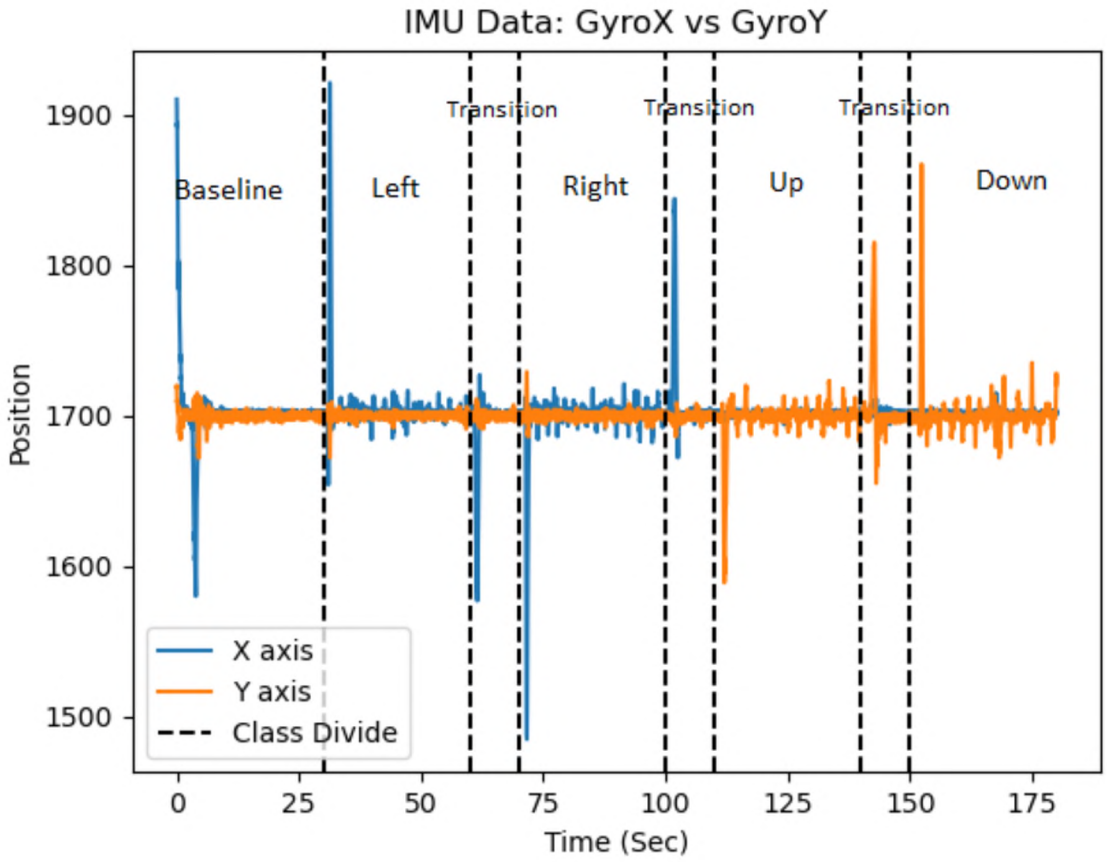


Figure 17: IMU vs Time

3.4.3 EMG Data

When graphing our EMG data, our graph looks different from the previous graphs. This is due to the fact that this data is a digital output of a triangulation algorithm rather than an analog reading of a sensor. Figure 18 shows the different types of outputs available. The readings of each output can be interpreted as follows:

1. BlinkWink

    0 - Neutral state

    1 - Blink

    2 - Left Wink

    3 - Right Wink

2. Horizontal Eyes

    -1 - Eyes look left

    0 - Eyes look forward

    1 - Eyes look right

3. Upper Face

    0 - Neutral state

    1 - Action on left side of face

    2 - Action on right side of face

4. Upper Face Power: Intensity of action taken normalized between 0 and 1.

5. Lower Face

    0 - Neutral state

    1 - Action on left side of face

    2 - Action on right side of face

6. Lower Face Power: Intensity of action taken normalized between 0 and 1.

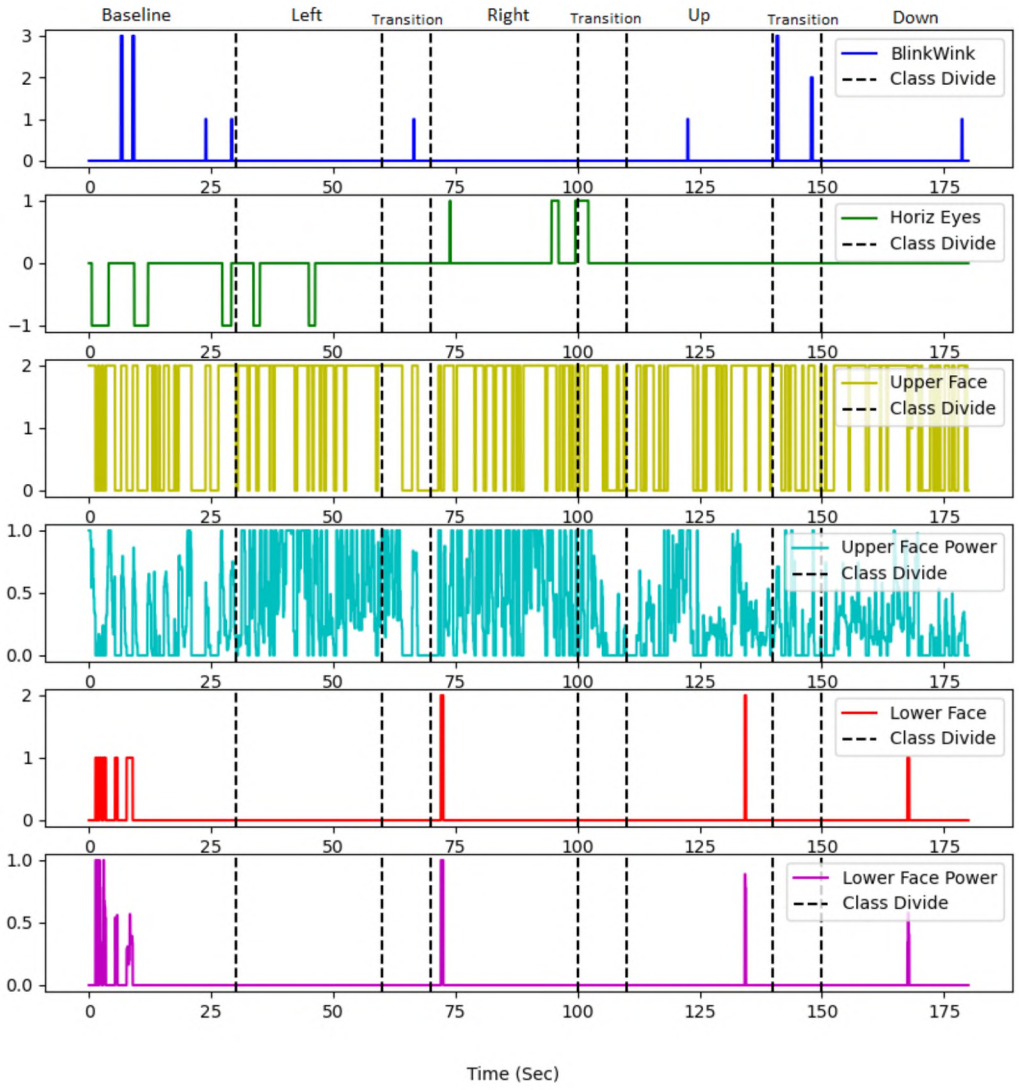


Figure 18: EMG vs Time

3.5 Data Pre-processing

The headset used in this experiment can, at its highest settings, produce 128 different samples of EEG data per second per electrode with a total of 14 electrodes. Additionally, each sample may additionally contain the IMU data and EMG data. We can easily see how a lot of data may be collected in a short amount of time. If we were to train a classifier with one set of samples at a time, the classifier would be making a decision every 1/128th of a second. Even with relative high accuracy, this many decisions in such a short amount of time could be problematic for real-world applications. Therefore, in an effort to reduce the computing power needed for pre-procession of the data, a batch learning method is used in this experiment as a form of dimensionality reduction. Instead of training with one sample at a time, a number of samples for a given time interval is given to the classifiers in a batch form. In our case, the given time interval is 30 seconds as that is how long the user is asked to concentrate on a specific direction.

4 Classification

The tuning of hyperparameters is often be used to maximize the efficiency of a given classifier. This is the case for this experiment. For each classifier tested, their hyperparameters were tuned using a grid-search method. Although time intensive, this process has been proven to improve the performance of a classifier [37].

Cross validation is a group of statistical techniques that helps assess how the results of a statistical model will generalize. In this instance, K-fold cross-validation is used in order to help prevent overfitting in the trained models. In K-fold cross-validation, the dataset is split into K parts or "folds." The model is then trained using K-1 subsets, leaving one remaining set as validation data. The accuracy of the model is then calculated by taking the average accuracy of all K subsets on the validation data [38]. Stratified random sampling is also often used with K-fold cross-validation as is the case here as well. This means that the sampling of the data is proportioned in such a way that the subsets reflect the proportion of the training set.

Some machine learning algorithms use the Euclidean distance to calculate distance between points. In these cases, it is important to normalize all features so as to not let one particular feature disproportionately affect calculated distances. The support vector machine and the multilayer perceptron neural network are affected by this phenomena, thus appropriate scaling techniques were used before training [39].

In machine learning, regularization is a technique used to better fit a function to a given training set and thus reducing error. It does this by constraining coefficient estimates towards zero, much like a form of regression [40]. This technique is sometimes used when trying to reduce or avoid overfitting. Regularization is used along with the support vector machine and neural network classifier presented here as it has shown it can improve efficiency [41].

To implement these methods, the scikit-learn [42] machine learning library in Python is extensively used.

4.1 Evaluating Classifiers

Each of the 100 sessions recorded was split into 5 classes (Baseline, Left, Right, Up, Down). This meant the data set used for this experiment consisted of 500 total samples. This was split into 80% testing data and 20% validation data. Thus, making the input size for our classifiers 400 samples, leaving the remainder 100 samples for testing.

In order to evaluate each classifier properly, all combinations of sensor data were used to find the best possible performance. Data from the EEG sensor consists of 14 features as each electrode in the headset has an individual output. Data from the IMU sensor consists of 2 features, X-axis and Y-axis. Data from the EMG sensor consists of 6 different features. When combining any two or more of these sensors, the total input features is equal to the sum of their individual features. Using the batch pre-processing method described previously, the feature space for EEG data is 3835 x 14. Since the IMU collects data at the same rate, its input space is 3835 x 2 as we only have 2 features. Finally, EMG has an input space of 958 x 6 as this data is collected at a slower rate.

In addition to K-fold cross validation, the standard deviation of the KFold average is also used in order to take into account the variability in the data when comparing the means. The classifiers are also evaluated by calculating precision, recall and F-Score. In order to calculate this, we must first define the following terms:

True Positives (TP) - Correctly predicted positive values where the value of the actual class and predicted class are the same.
False Positives (FP) – Incorrectly predicted positive values.
False Negatives (FN) – Incorrectly predicted negative values.

With these defined, we can then calculate the following:
Precision - The ratio of correctly predicted positive observations to the total predicted positive observations.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall (Sensitivity) - The ratio of correctly predicted positive observations to the all observations in actual class.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F-Score - The weighted average of Precision and Recall.

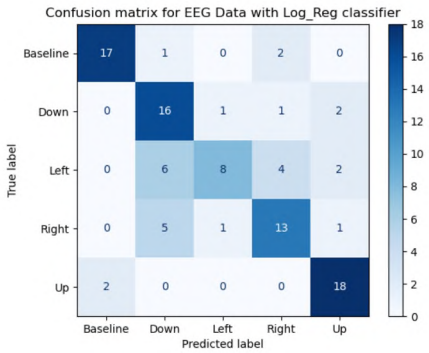
$$\text{FScore} = 2 \times \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$$

4.1.1 Logistic Regression

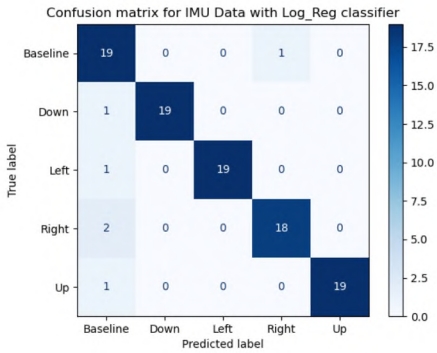
After careful hyperparameter tuning, the code for training the Logistic Regression classifier can be seen in Figure 26 found in Appendix A. An advantage of this classifier is that it can be trained in parallel mode which vastly speeds up training times. We can see the results in Table 1. From these we see that in the best of cases, it can outperform random using IMU data. In cases without IMU data, EEG by itself still performs well with a 73.98% precision. However, the model does lag behind in its KFold average precision with a 66.8%. This suggests this model might not be the best candidate for unseen data. In Figure 19a, we can see that it labeled a significant portion of Left and Right states as Down states.

Sensor/s	Precision	Recall	FScore	KFold	KFold SD
EEG	73.98	72.00	71.18	66.8	0.101
IMU	94.78	94.00	94.20	94.4	0.040
EMG	36.42	36.00	35.90	34.2	0.056
EEG/IMU	83.85	83.00	83.09	77.8	0.104
EEG/EMG	51.10	51.00	50.93	47.4	0.076
IMU/EMG	48.97	48.00	47.33	48.8	0.059
EEG/EMG/IMU	48.99	49.00	48.77	49.6	0.079

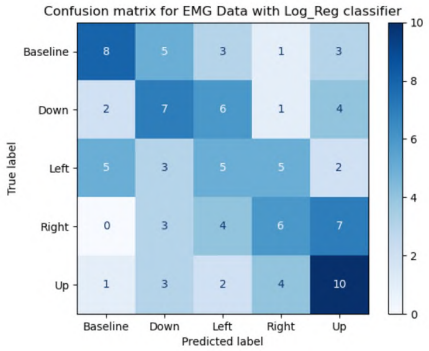
Table 1: Logistic Regression Results (in percentage)



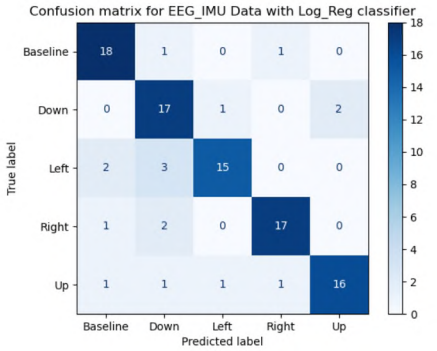
(a) EEG Data



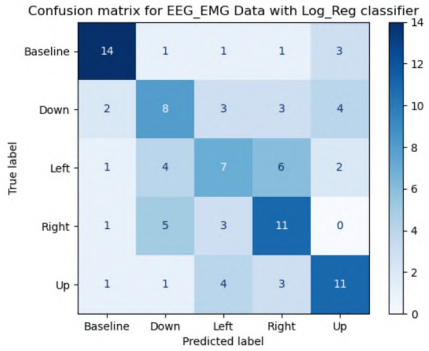
(b) IMU Data



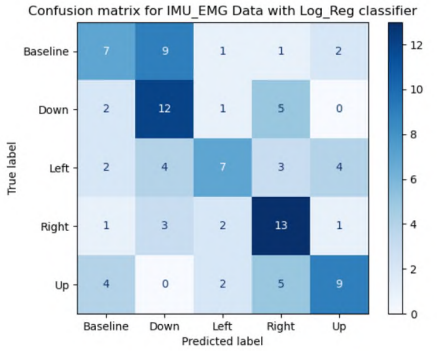
(c) EMG Data



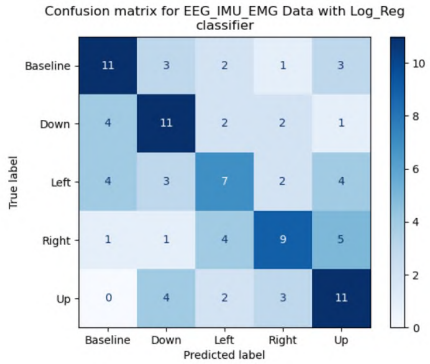
(d) EEG, IMU Data



(e) EEG, EMG Data



(f) IMU, EMG Data



(g) EEG, IMU, EMG Data

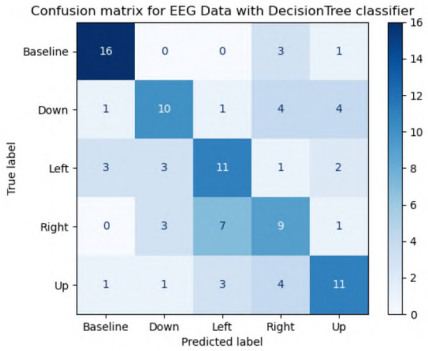
Figure 19: Logistic Regression Confusion Matrices.

4.1.2 Decision Trees

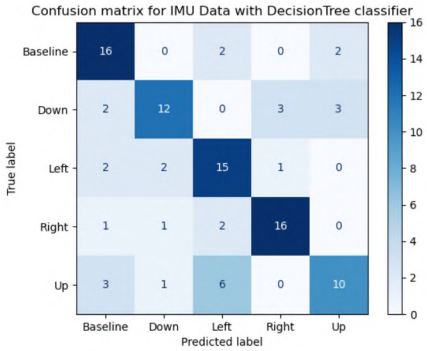
The code for training the Decision Trees classifier can be seen in Figure 27. Referencing Table 2 found in Appendix A, we see that the Decisions Trees classifier, although better than random, was not the best classifier of the pack. The best precision was IMU once again but this time we only see it reach 69.67%. However, the KFold average precision remain consistent with single training sessions which means our model will generalize just as well. In Figure 20, we see that for the most part the incorrectly predicted classes are distributed somewhat evenly.

Sensor/s	Precision	Recall	FScore	KFold	KFold SD
EEG	41.53	42.00	41.48	42.8	0.083
IMU	69.67	69.00	68.64	63.4	0.052
EMG	34.96	35.00	34.89	37.2	0.088
EEG/IMU	61.37	62.00	60.71	46.8	0.076
EEG/EMG	57.27	56.00	55.88	43.6	0.098
IMU/EMG	71.53	71.00	70.53	41.4	0.06
EEG/EMG/IMU	64.65	64.00	63.92	41.8	0.096

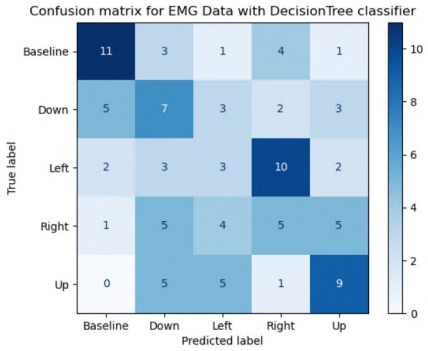
Table 2: Decision Trees Results (in percentage)



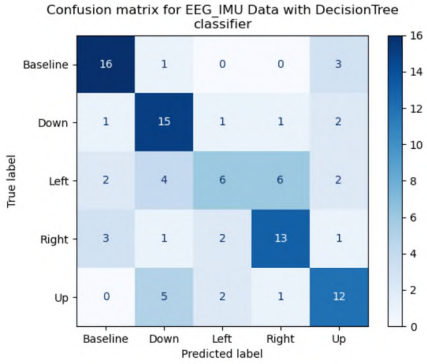
(a) EEG Data



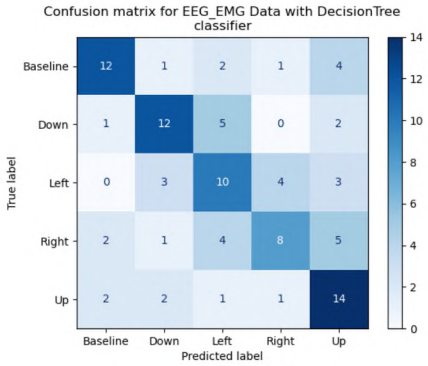
(b) IMU Data



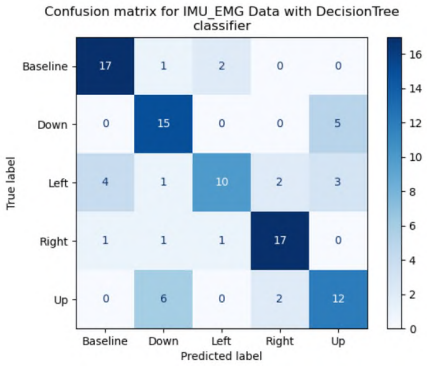
(c) EMG Data



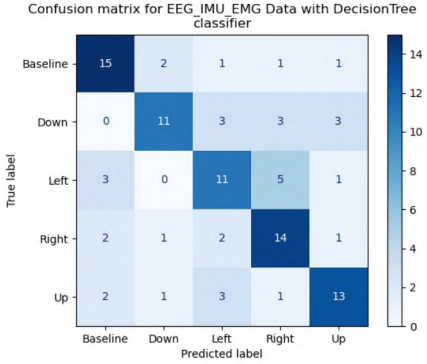
(d) EEG, IMU Data



(e) EEG, EMG Data



(f) IMU, EMG Data



(g) EEG, IMU, EMG Data

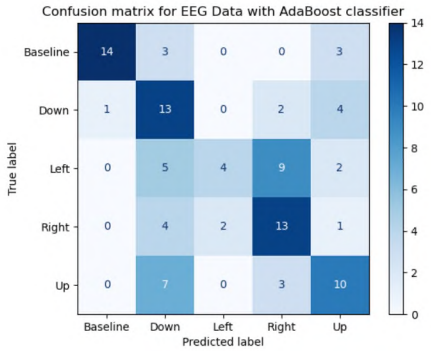
Figure 20: Decision Trees Confusion Matrices.

4.1.3 AdaBoost

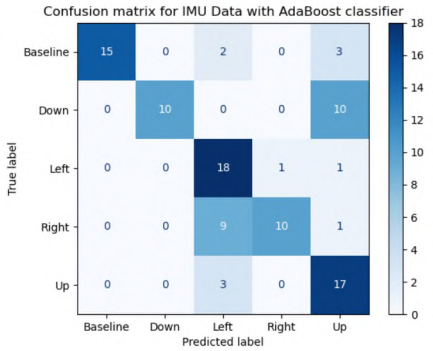
For the most part, the only hyperparameter that seems to make the most difference in training the AdaBoost classifier is the number of estimators. As seen in Figure 28 found in Appendix A, an optimal number was chosen after using the GridSearch method. Overall, we see that this classifier shows significant improvements over that previous classifier as seen in Table 3. This classifier being an ensemble method could be a big contributor in the improvement of the results. We see that once again IMU data seems to provide the best precision, however, recall is not so great in this case. We can see why in Figure 21b as half of the Down states were incorrectly classified as Up states.

Sensor/s	Precision	Recall	FScore	KFold	KFold SD
EEG	65.29	64.00	64.01	72.00	0.067
IMU	80.06	70.00	70.3	74.80	0.106
EMG	47.46	41.00	41.33	45.60	0.119
EEG/IMU	81.58	79.00	78.83	80.60	0.056
EEG/EMG	70.17	67.00	66.71	68.40	0.074
IMU/EMG	80.42	72.00	73.00	73.80	0.073
EEG/EMG/IMU	80.37	79.00	79.37	81.80	0.043

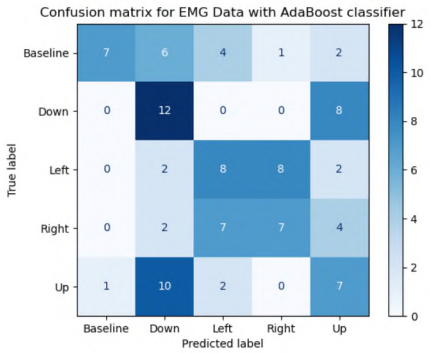
Table 3: AdaBoost Results (in percentage)



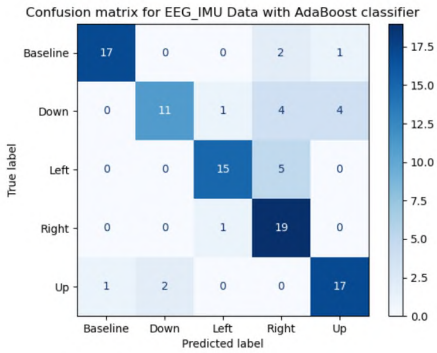
(a) EEG Data



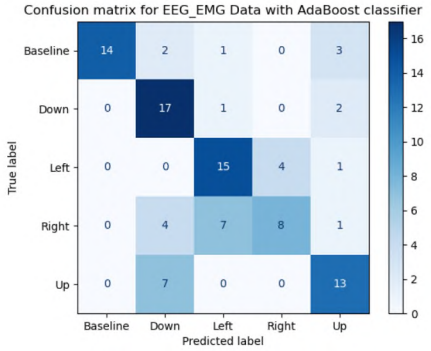
(b) IMU Data



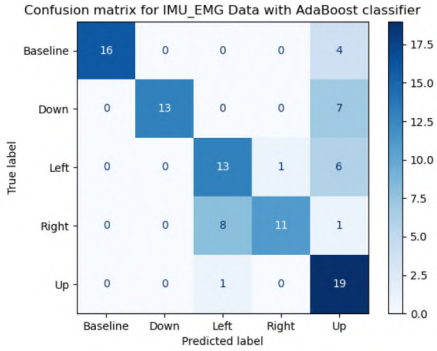
(c) EMG Data



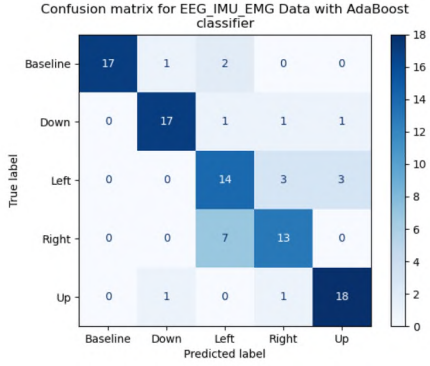
(d) EEG, IMU Data



(e) EEG, EMG Data



(f) IMU, EMG Data



(g) EEG, IMU, EMG Data

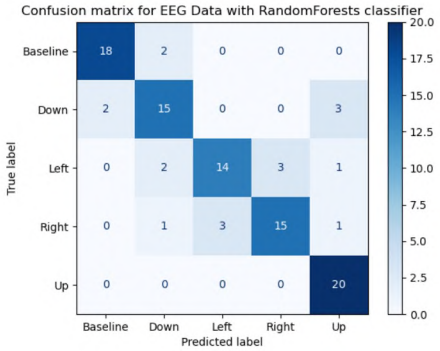
Figure 21: AdaBoost Confusion Matrices.

4.1.4 Random Forests

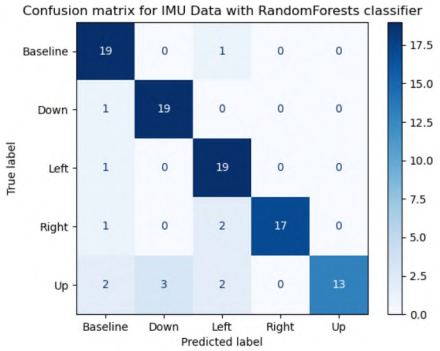
Random forests is another ensemble algorithm and the code used for training it can be seen in Figure 29 found in Appendix A. As the results in Table 4 indicate, this is one the best classifiers thus far. The lowest precision reported is with EMG data at 54.01%. This is still over double that of random. IMU data by itself still provides a respectable precision of 88.94%. Surprisingly, the best precision is reported to be the one with EEG and IMU data combined at 91.21%. Overall, precision and recall were pretty even along with the KFold average precision which means these models are very robust and well generalized. This is further reflected in Figure 22.

Sensor/s	Precision	Recall	FScore	KFold	KFold SD
EEG	82.14	82.00	81.70	79.20	0.066
IMU	88.94	87.00	86.78	88.60	0.049
EMG	54.01	53.00	52.26	53.00	0.103
EEG/IMU	91.21	91.00	90.99	87.20	0.056
EEG/EMG	77.82	78.00	77.46	81.20	0.095
IMU/EMG	85.28	85.00	84.79	88.00	0.039
EEG/EMG/IMU	84.44	84.00	83.97	87.00	0.043

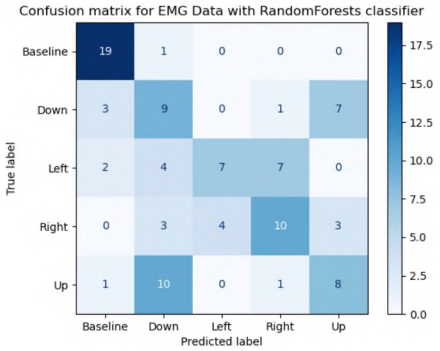
Table 4: Random Forests Results (in percentage)



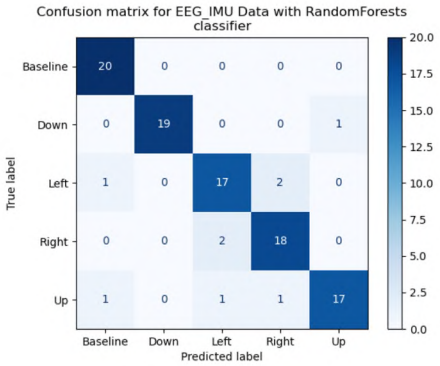
(a) EEG Data



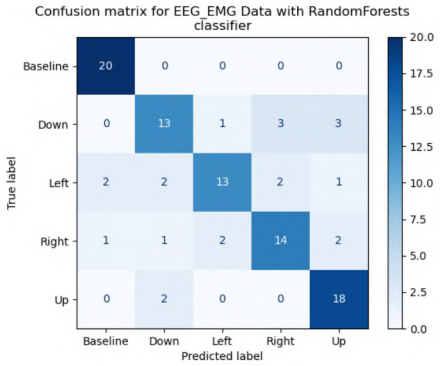
(b) IMU Data



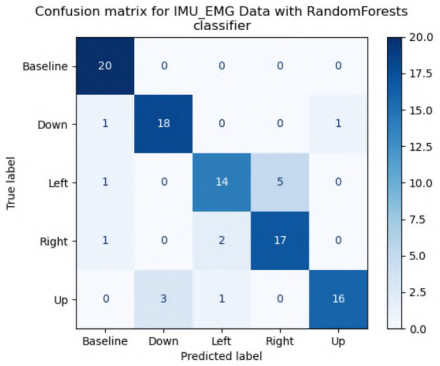
(c) EMG Data



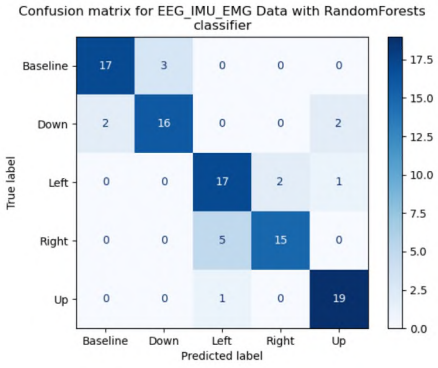
(d) EEG, IMU Data



(e) EEG, EMG Data



(f) IMU, EMG Data



(g) EEG, IMU, EMG Data

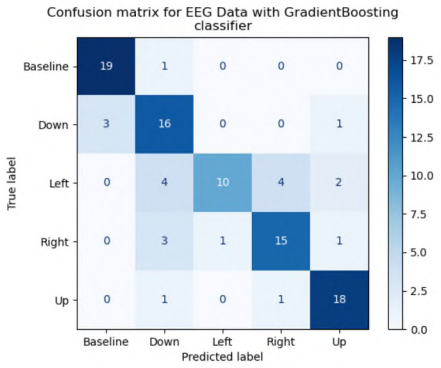
Figure 22: Random Forests Confusion Matrices.

4.1.5 Gradient Boosting Trees

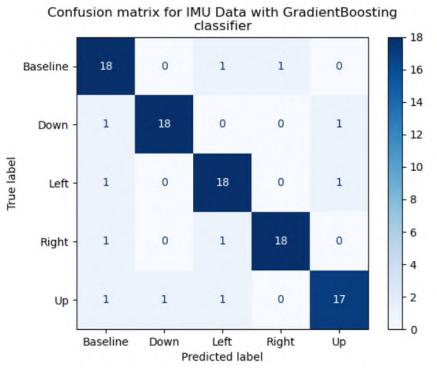
The code used for training Gradient Boosting Trees, our last ensemble classifier, can be seen in Figure 30 found in Appendix A. With this classifier, the highest scores are achieved when combining either EEG, EMG, IMU or all of them. In fact, this classifier seems to yield the best results thus far. In Table 5, we can see that we obtain a result higher than 70% in most cases. KFold tells us once more that these models will do relatively well with unseen data. The confusion matrices seen in Figure 23 tell a similar story.

Sensor/s	Precision	Recall	FScore	KFold	KFold SD
EEG	79.96	79.00	79.04	78.19	0.089
IMU	89.30	89.00	89.06	94.80	0.016
EMG	63.45	63.00	62.83	59.00	0.057
EEG/IMU	93.34	93.00	93.01	93.80	0.052
EEG/EMG	84.46	84.00	83.68	78.20	0.080
IMU/EMG	95.12	95.00	94.97	95.20	0.026
EEG/EMG/IMU	94.41	94.00	94.06	95.60	0.035

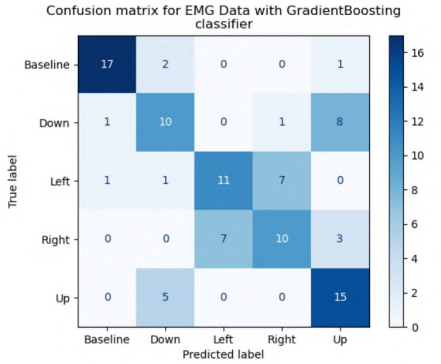
Table 5: Gradient Boosting Results (in percentage)



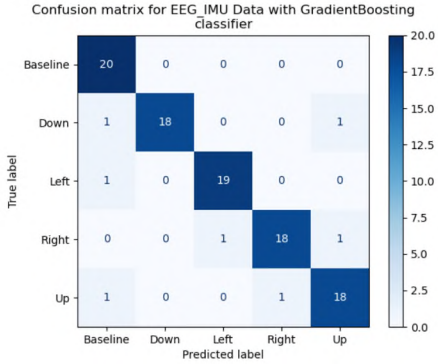
(a) EEG Data



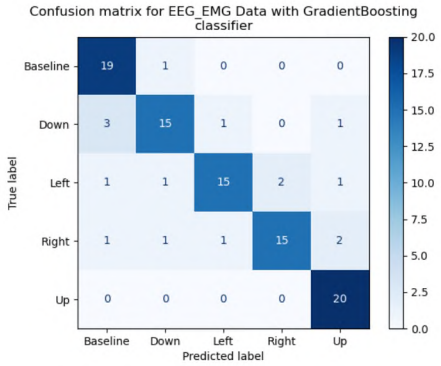
(b) IMU Data



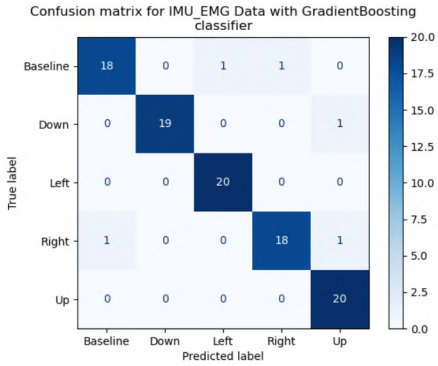
(c) EMG Data



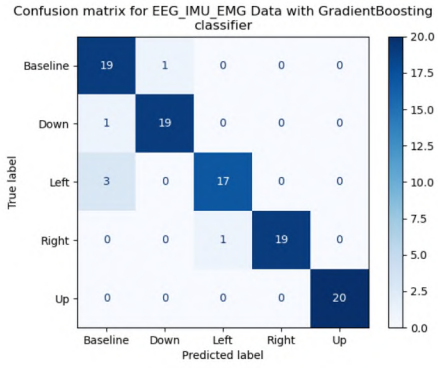
(d) EEG, IMU Data



(e) EEG, EMG Data



(f) IMU, EMG Data



(g) EEG, IMU, EMG Data

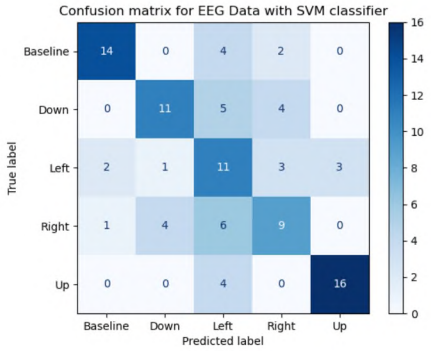
Figure 23: Gradient Boosting Confusion Matrices.

4.1.6 Support Vector Machines

The code for training the Support Vector Machines classifier can be seen in Figure 31 found in Appendix A. Here we can also see data being normalized using the StandardScalar() from sklearn package in python. Additionally, we can also see regularization being used with the C parameter. In Table 6 we see that in some cases, such as that of IMU data, SVM performed much better than a large portion of the classifiers with a precision of 95.10%. However, it did not perform as well in other cases - specifically those without IMU data. SVM operates by using the distance between points, it makes sense that it tends to do better in cases where IMU data is present since IMU data is a map of where the user has been. Overall, it still performed reasonably well as seen in Figure 24.

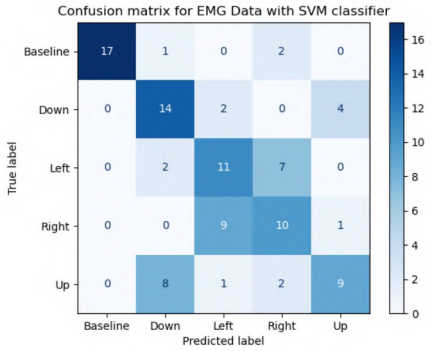
Sensor/s	Precision	Recall	FScore	KFold	KFold SD
EEG	64.40	61.00	62.04	67.40	0.140
IMU	95.10	95.00	95.02	96.00	0.105
EMG	63.15	61.00	61.40	58.40	0.076
EEG/IMU	80.74	78.00	78.25	75.00	0.141
EEG/EMG	77.66	72.00	73.23	70.40	0.153
IMU/EMG	78.35	74.00	74.77	72.60	0.099
EEG/EMG/IMU	73.35	70.00	69.66	74.40	0.116

Table 6: Support Vector Machines Results (in percentage)

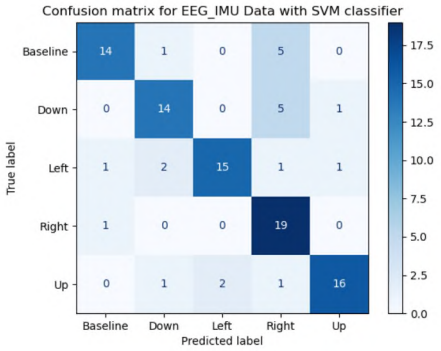


(a) EEG Data

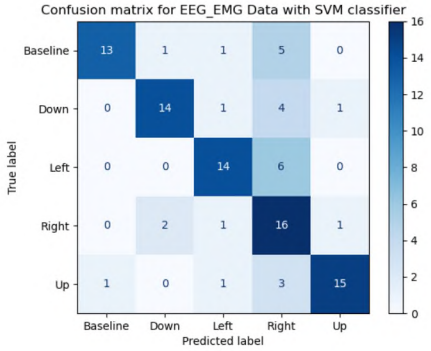
(b) IMU Data



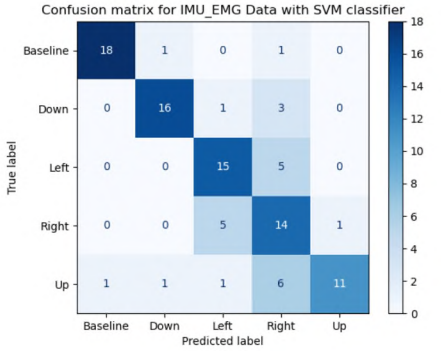
(c) EMG Data



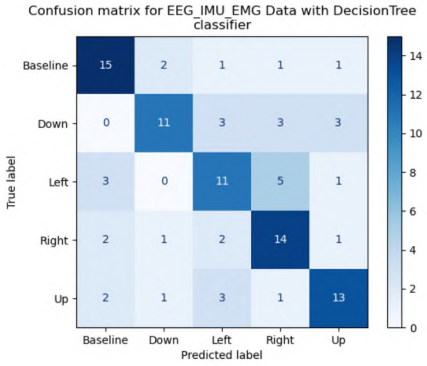
(d) EEG, IMU Data



(e) EEG, EMG Data



(f) IMU, EMG Data



(g) EEG, IMU, EMG Data

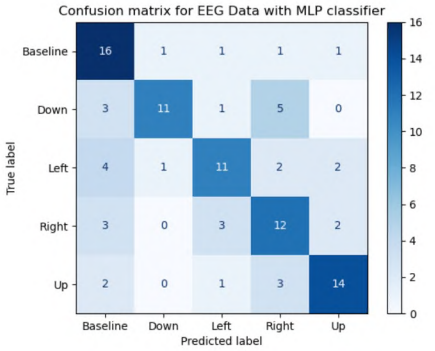
Figure 24: Support Vector Machines Confusion Matrices.

4.1.7 Neural Network

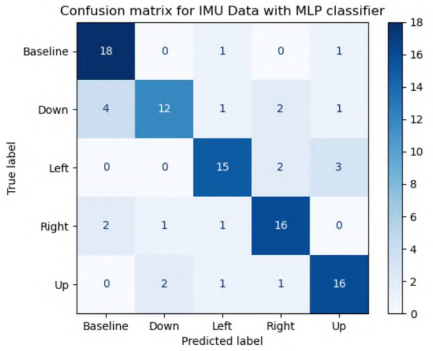
The code for training the MLP Neural Network can be seen in Figure 32 found in Appendix A. After normalizing the data using StandardScalar() from the sklearn package in python, we see that we used two layers of 100 nodes each, used the stochastic gradient-based optimizer "adam" and set the learning rate to "adaptive" which means it will keep the learning rate constant as long as training loss keeps decreasing. The learning rate will only be reduced by 5 each time two consecutive epochs fail to decrease training loss or fail to increase the validation score. In Table 7, we see that, although not the best performance out of all the classifiers, the MLP Neural Network consistently achieves a precision of around 70% in all cases. Overall, the KFold average accuracy is consistent with the single trained model results which means the model is well generalized. We see the same story in Figure 25.

Sensor/s	Precision	Recall	FScore	KFold	KFold SD
EEG	66.46	64.00	64.08	62.40	0.136
IMU	77.27	77.00	76.68	83.60	0.072
EMG	57.72	58.00	57.14	57.00	0.060
EEG/IMU	69.04	67.00	66.72	67.60	0.109
EEG/EMG	77.48	77.00	76.80	74.20	0.090
IMU/EMG	68.85	66.00	64.69	74.20	0.073
EEG/EMG/IMU	77.52	77.00	76.87	73.40	0.074

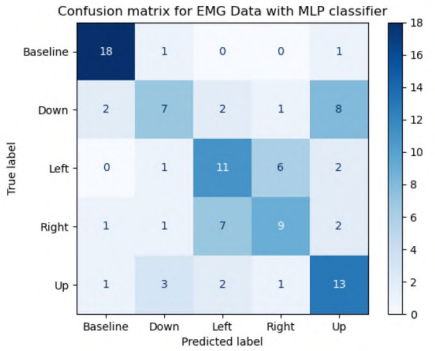
Table 7: Neural Network Results (in percentage)



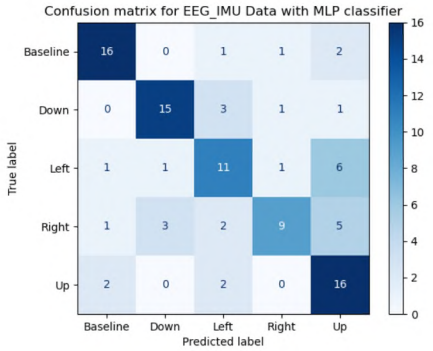
(a) EEG Data



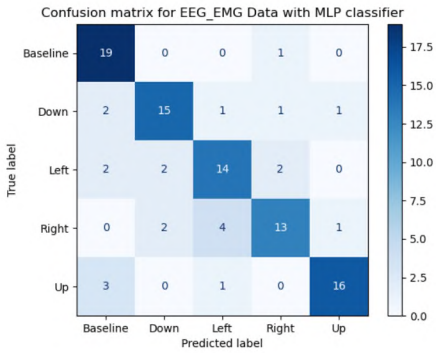
(b) IMU Data



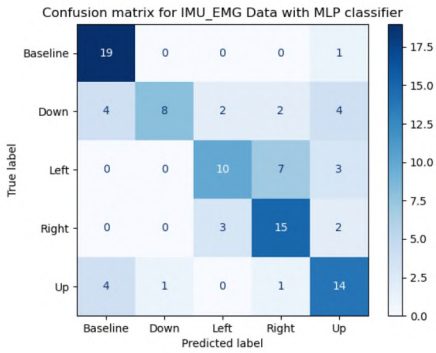
(c) EMG Data



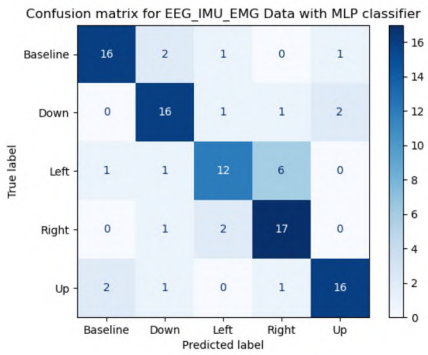
(d) EEG, IMU Data



(e) EEG, EMG Data



(f) IMU, EMG Data



(g) EEG, IMU, EMG Data

Figure 25: Neural Network Confusion Matrices.

5 Results

After evaluating all the classifiers we see that Gradient Boosting Trees performs the best in most paradigms as seen in Table 8. We also see that IMU data by itself, and paired with any other type of data, is a very good predictor.

With 5 states available, random choice should give us a precision of about 20%. The lowest precision recorded was only 34.96% using the EMG sensor with a Decision Tree classifier which is still better than random. In fact, EMG data consistently yielded the lowest precision across the board. However, when paired with the IMU sensor, it achieved one of the highest precisions recorded as seen in Table 8.

Conversely, we see that the best results are obtained with IMU sensor by itself with a precision of around 95.1%. This makes sense as this sensor is essentially mapping the direction the user is looking at. We also see that pairing EEG data with IMU data can marginally increases performance in all ensemble classifiers (AdaBoost, Random Forests, Gradient Boosting).

Classifier	EEG	IMU	EMG	EEG/IMU	EEG/EMG	IMU/EMG	EEG/EMG/IMU
Logistic Regression	73.98	94.78	36.42	83.85	51.1	48.97	48.99
Decision Trees	41.53	69.67	34.96	61.37	57.27	71.53	64.65
AdaBoost	65.29	80.06	47.46	81.58	70.17	80.42	80.37
Random Forests	82.14	88.94	54.01	91.21	77.82	85.28	84.44
Gradient Boosting	79.96	89.3	63.45	93.34	84.46	95.12	94.41
SVM	64.4	95.1	63.15	80.74	77.66	78.35	73.35
Neural Network	66.46	75.00	57.72	69.04	77.48	68.85	77.52

Table 8: Results Overview: Precision (in percentage)

To test whether the increase in performance when pairing IMU data and EEG data is statistically significant, we can calculate the P-Values. Normally this would be done using paired Student's t-test, however KFold Cross Validation was used to evaluate the classifiers which violates a key assumption of the paired Student's t-test since the observations in each sample are not independent. Thus, the non-parametric Wilcoxon signed-rank test [43] is used instead. This is an equivalent of the paired Student T-test, however this test is specifically geared towards comparing data samples which may be paired. An example of paired data samples is when the same algorithm is evaluated on different datasets which is the case here.

Since we want to test for the performance of each classifier using different datasets, we use the individual precision scores obtained from KFold Cross Validation in order to create a distribution of results. Thus, we can compare IMU and EEG/IMU data with by using the Wilcoxon signed-rank test and obtain the respective P-Values for each classifier as seen in Table 9.

Classifier	P-Value
Adaboost	0.115
Random Forests	0.065
Gradient Boosting	0.717

Table 9: P-Values for Ensemble Classifiers

The general accepted level for a distribution to be statistically significant is when the P-Value is below 0.05. Based on these results, it appears that combining EEG and IMU sensors does not yield statistically significant results over using data from the IMU sensor by itself.

Taking the data from the IMU by itself into account, we see how each classifier performed using K-Fold Cross validation in Table 10. Here, we can see the best classifiers are SVM, Gradient Boosting and Logistic Regression with average precisions in the mid-nineties. However, when taking the variance of the data into account, the best classifier for IMU data alone is SVM as the standard deviation in both Logistic Regression and Gradient Boosting could not account for the difference in performance of SVM.

Classifier	KFold	KFold SD
Logistic Regression	94.40	0.040
Decision Trees	88.60	0.049
AdaBoost	74.80	0.106
Random Forests	63.40	0.052
Gradient Boosting	94.80	0.016
SVM	96.00	0.105
Neural Network	81.40	0.072

Table 10: IMU Sensor Only Results (in percentage)

6 Discussion

In this experiment, we were able to successfully build a BCI that achieved significantly higher-than-random precision. Both the Gradient Boosting classifier and Random Forests were able to achieve a precision >70% for most cases. The only case in which this was not consistently achieved was when using EMG data alone. No single classifier was able to obtain a precision higher than 70% in this scenario.

The low precision using EMG data alone could be attributed to the fact that facial expressions are not always consistent even if they do come from the same user. This could also be attributed to the fact that the user was not instructed to follow any patterns with their facial expressions so we were only able to obtain random involuntary face movements that told us very little about the current state the user is in. However, we did see a case in which EMG data enhanced the results when combined with IMU using the Gradient Boosting classifier.

Several more patterns seem to emerge from the results, some which explain the EMG/IMU case. On average, ensemble classifiers tend to do marginally better than the others, even performing relatively well in cases where EMG data was included. This could be attributed to the fact that they are using multiple classifiers to minimize bias and improve precision. However, after obtaining the P-Values when comparing EEG and IMU data, it appears that the small gains in performance are not statistically significant. Thus, it would seem that using the IMU sensor by itself would be the better choice.

When specifically looking at the performance of each classifier using only the data acquired from the IMU sensor, the best classifier seems to be SVM even when taking variances in the data collected into account. Gradient Boosting and Logistic Regression follow closely behind but fail to perform better even after considering variance in the data.

With these results, one could envision this BCI being a good candidate for multiple scenarios. If a user has limited mobility below the chest area, they could still benefit from using the IMU sensor. If the user wanted to use this BCI to enhance their control over a motorized wheelchair, use of the IMU sensor might be limited but EEG sensors by themselves could still prove helpful in such a case.

7 Conclusion and Future Work

7.1 Conclusion

This paper focused on testing and implementing a versatile BCI. Three different modes of collecting data from a user were presented. All three were used by themselves, as well as with different combinations in order to test for the best possible outcome.

Several machine learning classifiers as well as a neural network were selected and evaluated. Their results with the given data were presented and discussed. At first glance, a few patterns surfaced from the results presented. One of these patterns seems to indicate that ensemble classifiers did better than others. However, further analysis of the results reveals that the slight increase in performance using these classifiers occurs only when comparing IMU data to EEG/IMU data.

Furthermore, when analysing the P-Values in these cases, it seems the increase in performance is not statistically significant and data collected from the IMU sensor by itself would suffice in building a functional BCI. In the case of only using data from the IMU sensor, the classifier with the best performance seems to be SVM. Thus, using the data from the IMU sensor paired with the SVM classifier seem yield the best performance.

The results of this paper also suggest that the proposed BCI could be successfully used across different paradigms. Furthermore, the methodology presented would also suggest it could lend itself well to real-time classification due to the limited computing power needed to pre-process the data.

Some of the limitations of this paper include the limited number of subjects available for data acquisition. This could imply the models are heavily overfitted to the small subject pool. However, in real-world applications, a BCI such as the one proposed here would have to be individually calibrated for each user.

7.2 Future Work

In the future, a larger subject pool would help generalize the method outlined in this paper even more. It would also show how much calibration, either choosing different classifiers or

different parameters, would have to be done per individual in order to achieve the same level of precision. If successful, this could be further implemented into a general purpose BCI [44]. Additionally, reducing the time interval for the batch of data captured would also allow the implementation of a real-time BCI [45].

Appendix A Code

Code implementation of the machine learning techniques used in this paper can be found in Figure 26 - Figure 32 listed below.

```
print("[INFO] evaluating logistic regression classifier...")
clf = LogisticRegression(solver="newton-cg", multi_class='auto', n_jobs
    =-1)
clf.fit(train_data, train_label)
clf_accuracy = clf.score(test_data, test_label)
clf_predictions = clf.predict(test_data)
print("K-Fold Validation for", classifier)
k_splits = 10
kf = StratifiedKFold(n_splits=k_splits, random_state=None)
result = cross_val_score(clf, data, labels, cv=kf, n_jobs=-1)
print("Avg accuracy: {:.2%}".format(result.mean()))
```

Figure 26: Logistic Regression Code

```
print("\n\n[INFO] evaluating Decision Tree classifier...")
clf = DecisionTreeClassifier(random_state=0, criterion="entropy",
    max_features='auto', splitter='random')
clf.fit(train_data, train_label)
clf_accuracy = clf.score(test_data, test_label)
clf_predictions = clf.predict(test_data)
print("K-Fold Validation for", classifier)
k_splits = 10
kf = StratifiedKFold(n_splits=k_splits, random_state=None)
result = cross_val_score(clf, data, labels, cv=kf, n_jobs=-1)
print("Avg accuracy: {:.2%}".format(result.mean()))
```

Figure 27: Decision Trees Code

```
print("\n\n[INFO] evaluating AdaBoost classifier...")
clf = AdaBoostClassifier(n_estimators=200, random_state=0, algorithm='
    SAMME')
clf.fit(train_data, train_label)
clf_accuracy = clf.score(test_data, test_label)
clf_predictions = clf.predict(test_data)
print("K-Fold Validation for", classifier)
k_splits = 10
kf = StratifiedKFold(n_splits=k_splits, random_state=None)
result = cross_val_score(clf, data, labels, cv=kf, n_jobs=-1)
print("Avg accuracy: {:.2%}".format(result.mean()))
```

Figure 28: AdaBoost Code

```
print("\n\n[INFO] evaluating Random Forests classifier...")
clf = RandomForestClassifier(n_estimators=100, bootstrap=True,
    max_features='sqrt', n_jobs=-1)
clf.fit(train_data, train_label)
clf_accuracy = clf.score(test_data, test_label)
clf_predictions = clf.predict(test_data)
print("K-Fold Validation for", classifier)
k_splits = 10
kf = StratifiedKFold(n_splits=k_splits, random_state=None)
result = cross_val_score(clf, data, labels, cv=kf, n_jobs=-1)
print("Avg accuracy: {:.2%}".format(result.mean()))
```

Figure 29: Random Forests Code

```
print("\n\n[INFO] evaluating Gradient Boosting classifier...")
clf = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0,
    max_depth=1, random_state=0)
clf.fit(train_data, train_label)
clf_accuracy = clf.score(test_data, test_label)
clf_predictions = clf.predict(test_data)
print("K-Fold Validation for", classifier)
k_splits = 10
kf = StratifiedKFold(n_splits=k_splits, random_state=None)
result = cross_val_score(clf, data, labels, cv=kf, n_jobs=-1)
print("Avg accuracy: {:.2%}".format(result.mean()))
```

Figure 30: Gradient Boosting Code

```
print("[INFO] evaluating SVM classifier...")
scalar = StandardScaler()
scalar.fit(train_data)
X_train = scalar.transform(train_data)
X_test = scalar.transform(test_data)
clf = svm.SVC(C=4)
clf.fit(X_train, train_label)
clf_accuracy = clf.score(X_test, test_label)
clf_predictions = clf.predict(X_test)
print("K-Fold Validation for", classifier)
scalar.fit(data)
X_data = scalar.transform(data)
k_splits = 10
kf = StratifiedKFold(n_splits=k_splits, random_state=None)
result = cross_val_score(clf, X_data, labels, cv=kf, n_jobs=-1)
print("Avg accuracy: {:.2%}".format(result.mean()))
```

Figure 31: Support Vector Machines Code

```
print("\n\n[INFO] evaluating MLP classifier...")
scalar = StandardScaler()
scalar.fit(train_data)
X_train = scalar.transform(train_data)
X_test = scalar.transform(test_data)
clf = MLPClassifier(activation="relu", hidden_layer_sizes=(100, 100),
    learning_rate="adaptive", solver="adam")
clf.fit(X_train, train_label)
clf_accuracy = clf.score(X_test, test_label)
clf_predictions = clf.predict(X_test)
print("K-Fold Validation for", classifier)
scalar.fit(data)
X_data = scalar.transform(data)
k_splits = 10
kf = StratifiedKFold(n_splits=k_splits, random_state=None)
result = cross_val_score(clf, X_data, labels, cv=kf, n_jobs=-1)
print("Avg accuracy: {:.2%}".format(result.mean()))
```

Figure 32: Neural Network Code

References

[1] Ujwal Chaudhary, Natalie Mrachacz-Kersting, and Niels Birbaumer. Neuropsychological and neurophysiological aspects of brain-computer-interface (BCI) control in paralysis. *The Journal of physiology*, 2020.

[2] Reza Abiri, Soheil Borhani, Eric W Sellers, Yang Jiang, and Xiaopeng Zhao. *A comprehensive review of EEG-based brain–computer interface paradigms*, volume 16 of number 1. IOP Publishing, 2019, page 011001.

[3] J. Del R. Millan, F. Renkens, J. Mourino, and W. Gerstner. *Noninvasive Brain-Actuated Control of a Mobile Robot by Human EEG*, volume 51 of number 6. 2004, pages 1026–1033. DOI: 10.1109/tbme.2004.827086.

[4] Zhe Zhang. *EEG Signal Processing and Analysis Using Efficient Machine Learning Techniques*. Master's thesis, California State University Channel Islands, 2018.

[5] Atena Reyhani Shahrestani. *Brain computer interfaces emotional state detection (EEG pattern recognition)*. Master's thesis, California State University Channel Islands, 2013.

[6] Tonio Ball, Markus Kern, Isabella Mutschler, Ad Aertsen, and Andreas Schulze-Bonhage. Signal quality of simultaneously recorded invasive and non-invasive EEG. *NeuroImage*, 46(3):708–716, 2009. ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2009.02.028. URL: https://www.sciencedirect.com/science/article/pii/S1053811909001827.

[7] Saba Moghimi MASc, Azadeh Kushki MASc PhD, Anne Marie Guerguerian MD FAAP FRCPC, and Tom Chau MASc PhD. A Review of EEG-Based Brain-Computer Interfaces as Access Pathways for Individuals with Severe Disabilities. *Assistive Technology*, 25(2):99–110, 2013. DOI: 10.1080/10400435.2012.723298. eprint: https://doi.org/10.1080/10400435.2012.723298. URL: https://doi.org/10.1080/10400435.2012.723298. PMID: 23923692.

[8] Luzheng Bi, Xin-An Fan, and Yili Liu. EEG-Based Brain-Controlled Mobile Robots: A Survey. *IEEE Transactions on Human-Machine Systems*, 43(2):161–176, 2013. DOI: 10.1109/tsmcc.2012.2219046.

[9] J Craig Henry. Electroencephalography: basic principles, clinical applications, and related fields. *Neurology*, 67(11):2092–2092, 2006.

[10] L F Haas. Hans Berger (1873-1941), Richard Caton (1842-1926), and electroencephalography. *Journal of Neurology, Neurosurgery and Psychiatry*, 74(1):9–9, 2003. DOI: 10.1136/jnnp.74.1.9.

[11] Aatif M. Husain and Saurabh R. Sinha. *Continuous EEG Monitoring Principles and Practice*. Springer International Publishing, 2018.

[12] Swati Vaid, Preeti Singh, and Chamandeep Kaur. EEG Signal Analysis for BCI Interface: A Review. *2015 Fifth International Conference on Advanced Computing Communication Technologies*, pages 143–147, 2015. DOI: 10.1109/ACCT.2015.72.

[13] Siuly Siuly, Yan Li, and Yanchun Zhang. EEG signal analysis and classification. *IEEE Trans Neural Syst Rehabilit Eng*, 11:141–144, 2016.

[14] D Gordon E Robertson, Graham E Caldwell, Joseph Hamill, Gary Kamen, and Saunders Whittlesey. *Research methods in biomechanics*. Human kinetics, 2013.

[15] MBI Reaz, MS Hussain, and F Mohd-Yasin. Techniques of EMG signal analysis: detection, processing, classification and applications (Correction). *Biological procedures online*, 8(1):163–163, 2006.

[16] Sivakumar Balasubramanian, Eliana Garcia-Cossio, Niels Birbaumer, Etienne Burdet, and Ander Ramos-Murguialday. Is EMG a Viable Alternative to BCI for Detecting Movement Intention in Severe Stroke? *IEEE Transactions on Biomedical Engineering*, 65(12):2790–2797, 2018. DOI: 10.1109/TBME.2018.2817688.

[17] Latest News. URL: https://csr.quadram.ac.uk/research/electromyography-of-mastication/ (visited on 05/10/2021).

[18] Marco Iosa, Pietro Picerno, Stefano Paolucci, and Giovanni Morone. Wearable inertial sensors for human movement analysis. *Expert review of medical devices*, 13(7):641–659, 2016.

[19] Ping Li, Ramy Meziane, Martin J.-D. Otis, Hassan Ezzaidi, and Philippe Cardou. A Smart Safety Helmet using IMU and EEG sensors for worker fatigue detection. *2014 IEEE International Symposium on Robotic and Sensors Environments (ROSE) Proceedings*, pages 55–60, 2014. DOI: 10.1109/ROSE.2014.6952983.

[20] M Olinski, A Gronowicz, M Ceccarelli, and D Cafolla. Human Motion Characterization Using Wireless Inertial Sensors, *New Advances in Mechanisms, Mechanical Transmissions and Robotics*, pages 401–408. Springer, 2017.

[21] Raymond E. Wright. Logistic Regression, *Reading and Understanding Multivariate Statistics*, pages 217–244. American Psychological Association, 1995.

[22] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, and S Yu Philip. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.

[23] Arvindpdmn Raam.raam. Decision Trees for Machine Learning, July 2020. URL: `https://devopedia.org/decision-trees-for-machine-learning` (visited on 05/10/2021).

[24] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. BCI adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[25] Nabeel H Al-A'araji, Safaa O Al-Mamory, and Ali H Al-Shakarchi. Classification and Clustering Based Ensemble Techniques for Intrusion Detection Systems: A Survey. *Journal of Physics: Conference Series*, volume 1818 of number 1, page 012106. IOP Publishing, 2021.

[26] Tony Yiu. Understanding Random Forest, August 2019. URL: `https://miro.medium.com/max/1052/1*VHDtVaDPNepRglIAv72BFg.jpeg`.

[27] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Boosting and additive trees, *The elements of statistical learning*, pages 337–387. Springer, 2009.

[28] GeeksforGeeks. ML - Gradient Boosting, September 2020. URL: `https://www.geeksforgeeks.org/ml-gradient-boosting/` (visited on 05/10/2021).

[29] David Meyer and FH Technikum Wien. Support vector machines. *The Interface to libsvm in package e1071*, 28, 2015.

[30] Rohith Gandhi. Support Vector Machine - Introduction to Machine Learning Algorithms, July 2018. URL: `https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47` (visited on 05/10/2021).

[31] Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636, 1998.

[32] Hossein Hakimpoor, Khairil Anuar Bin Arshad, Huam Hon Tat, Naser Khani, and Mohsen Rahmandoust. Artificial neural networks' applications in management. *World Applied Sciences Journal*, 14(7):1008–1019, 2011.

[33] Grant S Taylor and Christina Schmidt. Empirical evaluation of the Emotiv EPOC BCI headset for the detection of mental actions. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 56 of number 1, pages 193–197. SAGE Publications Sage CA: Los Angeles, CA, 2012.

[34] EPOC Headset Details. URL: https://emotiv.gitbook.io/epoc-user-manual/using-headset/epoc%20_headset_details (visited on 05/10/2021).

[35] EPOC User Manual. URL: https://emotiv.gitbook.io/epoc-user-manual/ (visited on 05/10/2021).

[36] Sangeetha Balasubramanian, Shruti Shriya Gullapuram, and Abhinav Shukla. Engagement estimation in advertisement videos with EEG. *arXiv preprint arXiv:1812.03364*, 2018.

[37] Petre Lameski, Eftim Zdravevski, Riste Mingov, and Andrea Kulakov. SVM parameter tuning with grid search and its impact on reduction of model over-fitting, *Rough sets, fuzzy sets, data mining, and granular computing*, pages 464–474. Springer, 2015.

[38] Daniel Berrar. Cross-validation. *Encyclopedia of bioinformatics and computational biology*, 1:542–545, 2019.

[39] Jun-Mo Jo. Effectiveness of normalization pre-processing of big data to the machine learning performance. *The Journal of the Korea institute of electronic communication sciences*, 14(3):547–552, 2019.

[40] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[41] Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5(Oct):1391–1415, 2004.

[42] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[43] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

[44] Gerwin Schalk, Dennis J McFarland, Thilo Hinterberger, Niels Birbaumer, and Jonathan R Wolpaw. BCI2000: a general-purpose brain-computer interface (BCI) system. *IEEE Transactions on biomedical engineering*, 51(6):1034–1043, 2004.

[45] Maria V Ruiz Blondet, Adarsha Badarinath, Chetan Khanna, and Zhanpeng Jin. A wearable real-time BCI system based on mobile cloud computing. *2013 6th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 739–742. IEEE, 2013.