EEG and Machine Learning

In Brain-Computer Interface

By

CHER WA VANG

Bachelor of Science in Electrical Engineering

Oklahoma State University

Stillwater, Oklahoma

2014

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for the
Degree of
Masters of Science
July, 2016

EEG and Machine Learning

In Brain-Computer interface

Thesis Approved:

Hu, Jingtong, Ph.D.

Thesis Adviser

Stine, James E., Jr., Ph.D.

Rajamani, Vignesh, Ph.D.

Acknowledgements

I would like to thank my advisor, Dr. Hu Jingtong, for his support and guidance during this thesis work. I am grateful to Dr. James Stine, and Dr. Rajamani Vignesh for serving on my committee. I would also like to thank the School of Electrical and Computer Engineering at Oklahoma State University for providing me with the opportunity for my education. Lastly I would like to thank my friends and family, whose influence on my life has made me who I am today.

Name: CHER WA VANG

Date of Degree: MAY 2016

Title of Study: EEG and Machine Learning in Brain-Computer interface

Major Field: ELECTRICAL ENGINEERING

Abstract: When a neuron within the brain fires, small traces of electrical activity can be measured. Electroencephalography (EEG) is one such method of measuring that electrical activity. With the emergence of inexpensive, and portable so called “Wearable EEG” devices, such as the Emotiv EPOC+, what is traditionally used for clinical diagnosis and cognitive neuroscience is now more readably available for the consumer.

The growth of computing power has grown exponentially since the implementation of the first semiconductor in 1947. The average household computer has more computing power than the computer used to take Apollo 11 to the moon. Computers have grown powerful enough that they can run a machine learning algorithm to see patterns that to the human perception, may appear to be random.

One of the first expressions of human art and culture was first expressed as paintings on cavern walls, then through language, writing, the radio, the television, the internet, and soon to be virtual reality (VR). The human race is at the dawn of the age of VR. With the explosive success of commercial VR such as the Oculus Rift, and the HTC VIVE, VR is here to stay.

The purpose of this research is to go over the practicability of EEG technology and machine learning in brain-computer interface to allow a person to play video games with their mind. By reading EEG brain signals during video gaming activities, a machine learning algorithm will attempt to produce a model to predict future video game actions. This research also offers a brief future potential capabilities, and further improvements to the system.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Using your mind to interface directly with software is an often used theme in science fiction, such as the 1999 Wachowski Brothers blockbuster film “The Matrix,” and more recently the light novel series “Sword Art Online” by Reki Kawahara.

In The Matrix, the movie depicts a dystopian future in which most humans live in a virtual reality simulation called the Matrix. In this film the human mind interface with the Matrix through a wire in the back of the head.

In Sword Art Online, a Virtual Reality Massive Multiplayer Online Role-Playing-Game (VRMMORPG) is created. Users would connect to this game through a fantasy device called Nerve Gear, a helmet that wirelessly interfaces with the brain, stimulating the user’s five senses.

In this research, a machine learning algorithm is used to model the human brain when playing video games. To interface with the game, the Emotiv EPOC+ is used. The EPOC+ is a 14 channel Bluetooth wireless EEG, with a 14-bit Analog to Digital Converter (ADC) resolution, and a 128 samples per second (SPS) sampling rate. [10]

1.1 Machine Learning

Machine learning is defined by the Encyclopedia Britannica [1] as “an artificial intelligence discipline concerned with the implementation of computer software that can be learned autonomously.” Machine learning is the study and development of algorithms that can learn and make predictions form a set of data. Instead of making predictions based of a set of rules, a Machine learning algorithm makes data driven prediction based on a model, a structure and corresponding interpretation that summarizes or partially summarizes a set of input and output data. [2]

Machine learning tasks can be classified into three broad categories depending on the nature of the learning feedback available to the system: Supervised learning, unsupervised learning, and reinforcement learning. [3]

In supervised learning, the computer is given a set of independent attributes (inputs) and a designated dependent attribute (outputs) by the “teacher” with the goal of learning a rule to map the inputs to the outputs [2]. This type of learning is used to learn a problem that may be simple for humans learn, but too complex for a set of rules to code. For example, being presented with a lineup of fruits, and being told to identify each fruit. It’s simple to humans solve this problem, but difficult to describe into code for a computer to solve. Types of supervise learning are classification and regression. In classification, inputs are divided into two or more classes, or a group set. The model then places incoming data in one or more of these categories. Based on which categories the data is placed in, it makes a prediction. Regression is similar to classification, but is used with continuous data to predict future data points [4]. The difference between classification and regression is shown in **Figures 1 and 2**. In **Figure 1**, the red line divides the two classes (or groups). When new data comes to get predicted, the predicted outcome is determined by where the data lies on the graph. In **Figure 2**, the red line shows the trend of the data. Using this line, future predictions can be made.

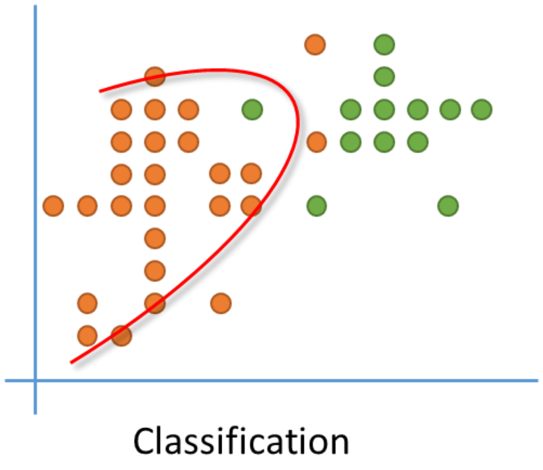


Figure 1: Example of classification learning

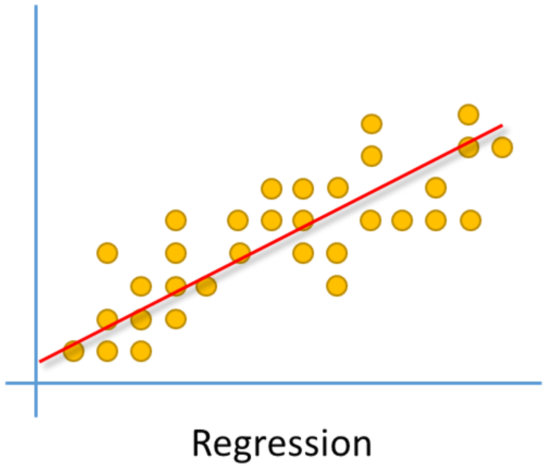


Figure 2: Example of regression learning

Unsupervised learning is like supervised learning, but with no designated dependent attribute. The input is known, but the output isn't. This type of learning is used when trying to find a pattern not known, or obvious from a set of data. For example, trying to detect a radio signal below the noise floor. The radio signal would have a pattern that the algorithm will try to find in the noise. [2]

Reinforcement learning is inspired by behaviorist psychology. The computer takes actions in an environment to maximize some notion of reward. Correct actions are rewarded (reinforced) while incorrect actions are not. For example, a robot navigating a maze. Every correct move is given positive reinforcement, while incorrect ones are not. This repeats until the robot can correctly solve the maze.

This thesis focuses on the use of supervised learning, specifically classification learning. The workflow of supervised classification learning is shown in **Figure 3** for learning and **Figure 4** for predicting. In **Figure 3,** the process starts by loading the independent and dependent data sets. This data contains multiple independent dependent pairs. Then preprocessing the data to extract certain features for learning. Afterwards it is sent to the classification learner to teach. After many cycles through many learners, the best one is selected as the ideal model. In **Figure 4**, the independent data is sent to the preprocessed. This data is only one set of the independent data. This data gets prepossessed the same way it did during training. Using the same model during training the preprocess data gets passed in and the predicted dependent data comes out. In this example, the predicted data is "to move left."

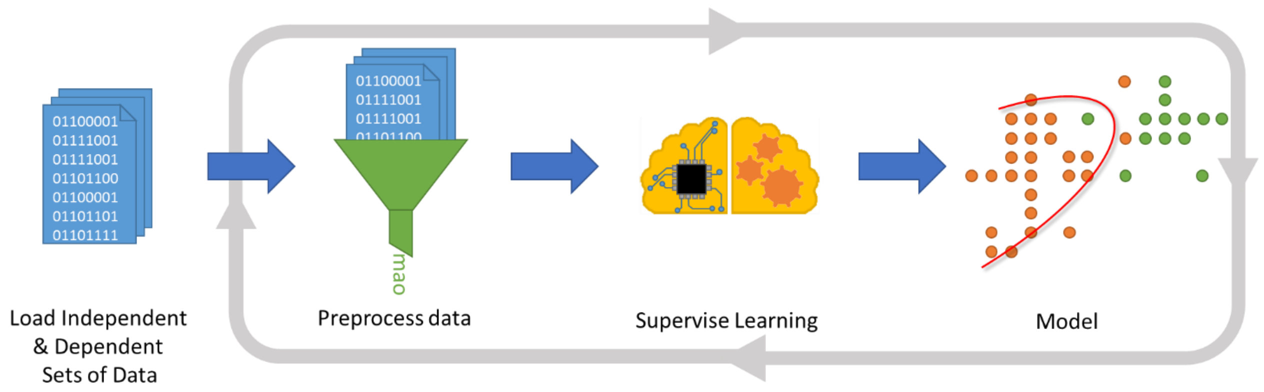


Figure 3: Workflow during training of classification learning

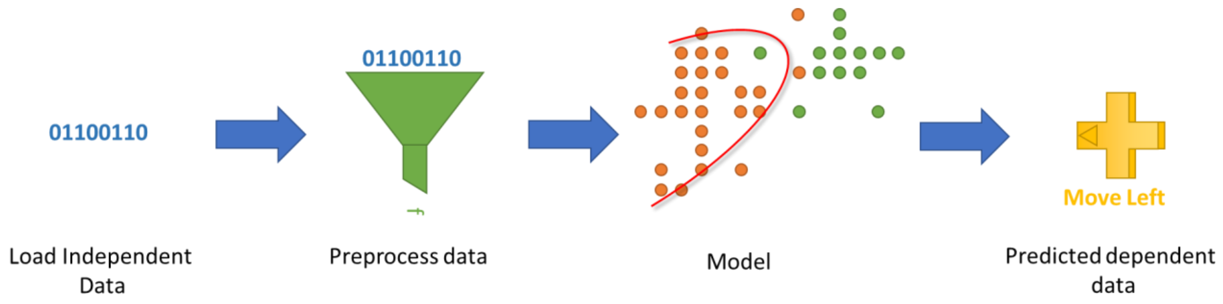


Figure 4: Workflow during prediction of classification learning

This purpose of this research is to test the feasibility of using machine learning to interpret brain signals in a fast and efficient way. The machine learning platform that will be utilize in this research is the MATLAB classification learner.

1.2 Electroencephalography

EEG is an electrophysiological monitoring technique for recording and interpreting electrical activity in the brain. This phenomenon was first observed in 1875 by Richard Caton, a physician practicing in Liverpool, who presented his findings of electrical activity of rabbits and monkeys in the British Medical Journal. The first recording of human EEG was done in 1924 by a German physiologist and psychiatrist Hans Berger. Berger also invented the first electroencephalogram [5].

The nerve cells of the brain generate electrical impulses that fluctuate in distinct rhythmic patterns. EEG waves are measured with typically with 8 to 16 pairs of electrodes, placed on the scalp. The difference in voltage between the pairs is recorded as the signal. Typical interpretations of the signal are done by taking spectral analysis of it. A spectral analysis of the signal shows the brain pattern in frequency domain [6]. When looking at the frequency domain, various frequency bands are associated with different rhythmic activities. Five commonly bandwidths known as alpha, beta, theta, delta, and gamma are shown in **Table 1**. An Example of each brain wave is shown in **Figure 5**. This example came from the EEG waves captured in this experiment filtered through MATLAB. Note the “Original EEG Wave” has its DC offset of about 4200 uV removed.

Band	Frequency (Hz)	Associated Activity [17]
Delta	1-4	Deep meditation, sleep, and source of empathy
Theta	4-7	Learning, and memory
Alpha	8-15	Mental coordination, calmness, and alertness
Beta	16-31	Problem solving, judgment, decision making, focused mental activity
Gamma	31+	Love, high altruism, and higher virtues

Table 1: EEG Bands and Frequencies

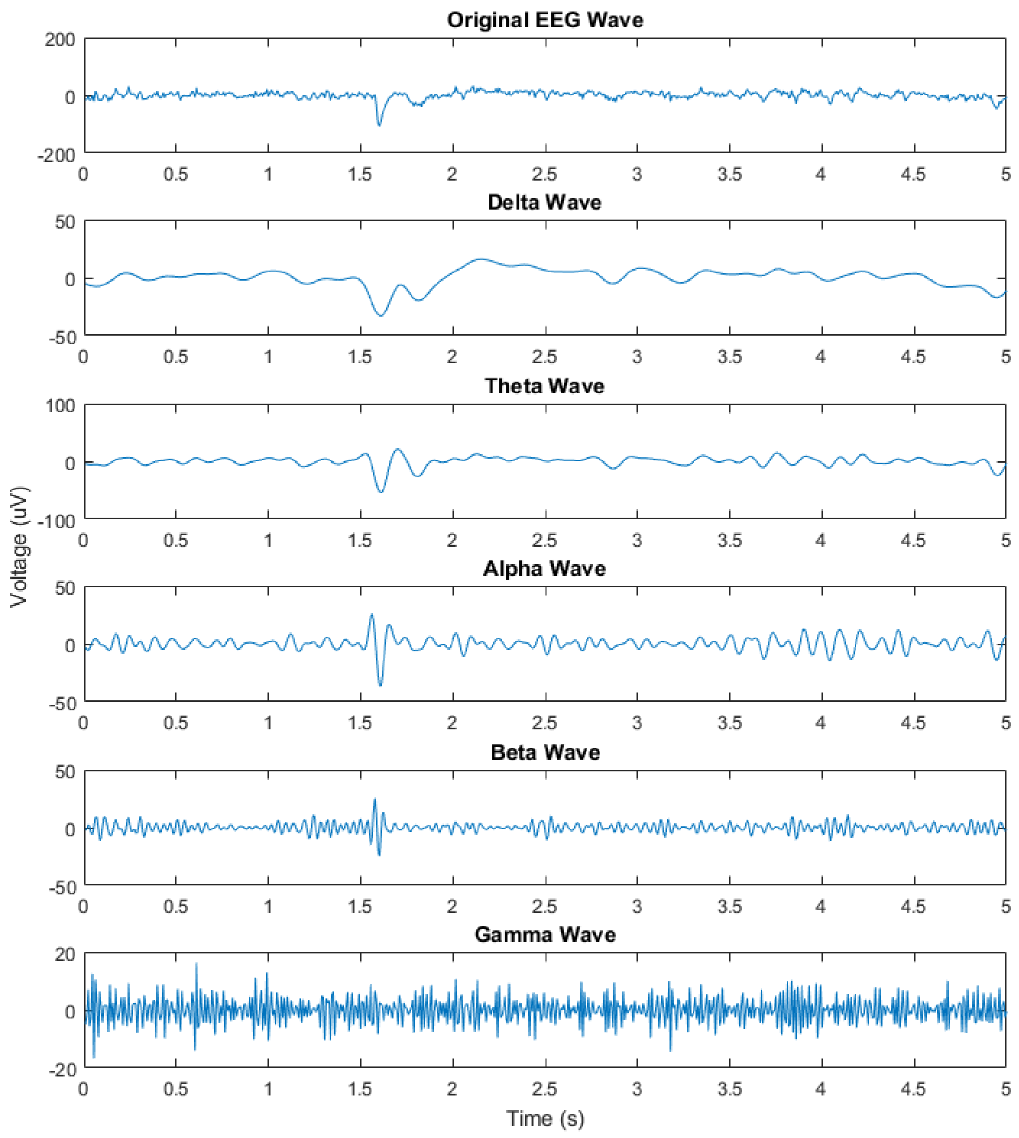


Figure 5: Comparison of EEG waves

Low frequencies (typically 0.1 to 1 Hz), and high frequencies (typically 60 to 70 Hz) are filtered out to remove any artifacts that may occur from eye, cardiac, or muscle activity [7].

Where the electrodes, or pads, are placed are also important in EEG. The placement effects where on the surface on the brain the brain activity is being recorded. This is important since functionality of the cerebrum, the largest part of the human brain, is sectioned. The four main sections are shown in **Figure 6**.

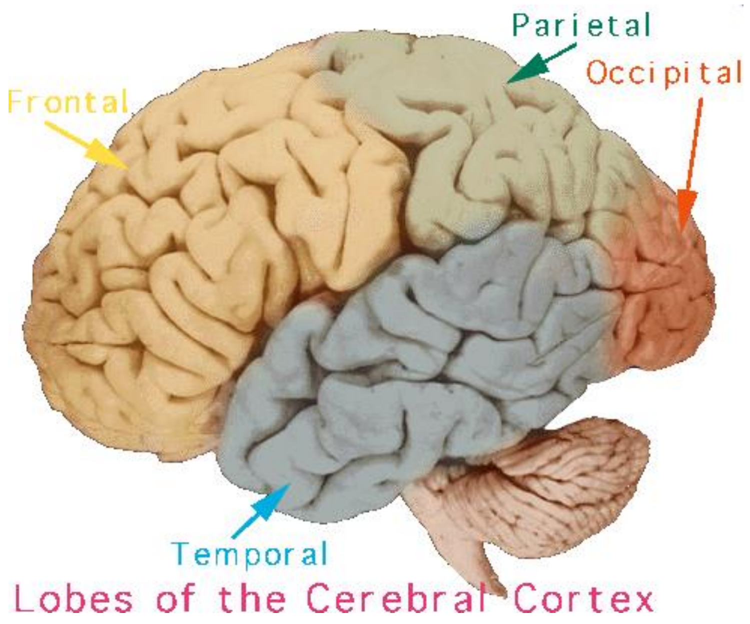


Figure 6: Four sections of the Cerebral Cortex. Image has been edited to remove the blue background from [8]

The Cerebrum or cortex is associated with higher brain function such as though and action. The four sections are the frontal lobe, parietal lobe, occipital lobe, and temporal lobe. Each lobe is associated with different functions. The frontal lobe is associated with reasoning, planning, parts of speech, movement, emotions, and problem solving. The parietal lobe is associated with movement, orientation, recognition and perception of stimuli. The occipital lobe is associated with visual processing. The temporal lobe is associated with perception and recognition of auditory stimuli, memory, and speech [8].

Scalp EEG locations can be defined using the modified combinatorial nomenclature (MCN), which is a higher resolution version of the international 10-20 system. This new system is illustrated in **Figure 7**, where the green markers are the EEG pad locations on the Emotiv EPOC+, and the orange markers are pad locations for the common mode sense (CMS) and driven right leg (DRL). The CMS and DRL are used to form a feedback loop to remove any electrical interference since the EEG electrodes are measuring in the ranges of micro volts.

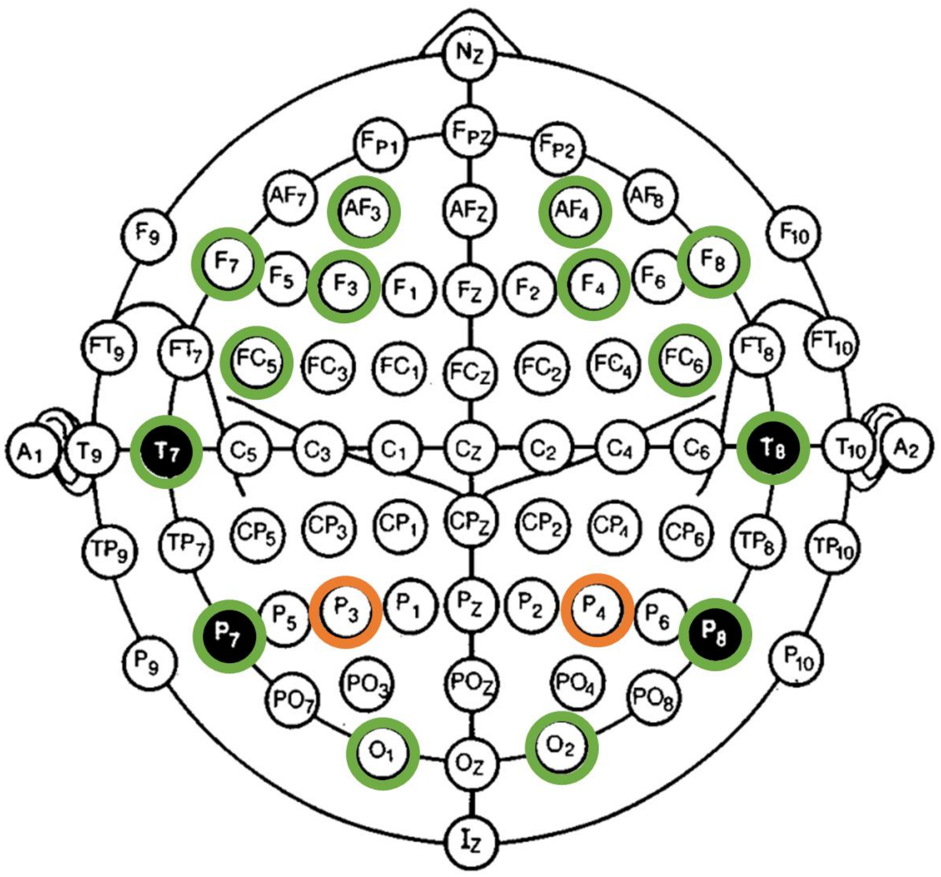


Figure 7: MCN locations edited to show Emotiv EPOC+ locations highlighted [9]

Each EEG sight has a letter and number to identify it. The letters F, C, P, O, and T stand for frontal, central, parental, occipital, and temporal lobes respectively (note central is not a lobe). Along with these are F_p, A, N, I, which stand for front polar, earlobes, nasion, and inion respectively. These letters represent the original 10-20 system. In the higher resolution MCN system, extra markers are added. The double letters refer to positions in-between two areas from the 10-20 system. These double letters AF, FC, CP, PO, FT, and TP stand for the areas in between the aforementioned letters, with AF standing for the area between F and F_p. The number subscripts represent the distance away from the center, with the subscript "Z" representing zero. Odd and even numbers represent the left and right side respectively. The larger the number, the further away that location is away from zero [9].

1.3 Brain-Computer Interface

Human-computer interfaces (HCIs) such as mouse, keyboards, and touch screens have become ubiquitous while interfacing with computers. There is a growing need for direct brain-computer interfaces (BCI) in situations where HCIs are not viable, such as individuals who are not capable of producing muscular movement to control HCIs. BCI can offer an extra dimension of control not available in HCIs.

BCI is a direct communication pathway based on neural activity generated by the brain, independent of peripheral nerves and muscles, to an external device. The purpose of BCI is often used for researching, mapping, assisting, enhancing, augmenting, or repairing human cognitive or sensory-motor functions, providing a new channel of control that requires voluntary adaptive control by the user [11].

BCI's can be categorized into two types: invasive, and noninvasive. Invasive or partially invasive BCIs involve some form of brain implant, or some form of direct communication with the cortex of the brain or grey matter. Noninvasive BCIs do not directly interface with the brain, or break the skin. Noninvasive techniques reduce the risk for users, since surgery or permanent attachment of a device is not necessary. Noninvasive techniques include computerize tomography (CT), position electron tomography (PET), Magnetic resonance imaging (MRI), functional magnetic resonance imaging (FMRI), EEG, and more [11]. This thesis focuses on the use noninvasive methods of BCI with EEG.

The BCI system used in this thesis is shown in **Figure 8**. The BCI System start with the Emotiv EPOC+ sampling EEG data from the user. Then sending that EEG data through proprietary 2.4 GHz wireless to the PC. Once in the PC the data can go anywhere such as a videogame.

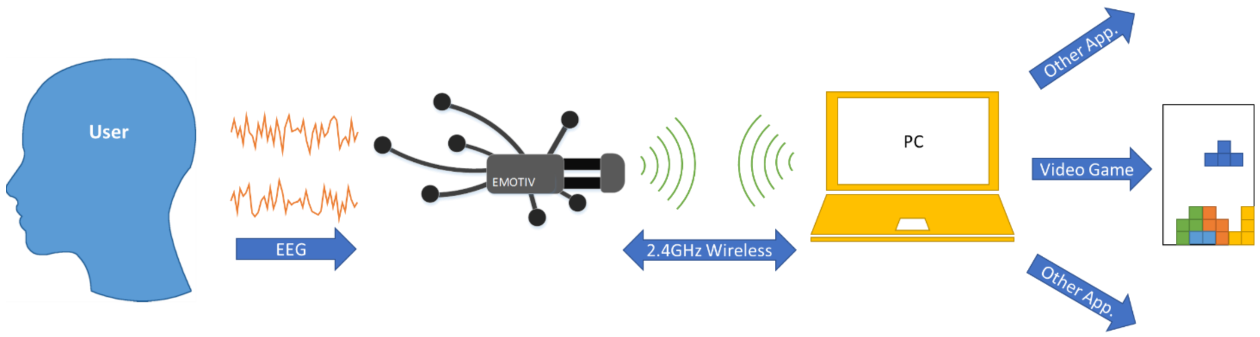


Figure 8: Brain-Computer Interface System

Chapter 2: Background and Related Work

Previous BCI work

BCI related work started in the 1970s, funded by the Pentagon's Advance Research Projects Agency (DARPA). This research, managed by George Lawrence and coworkers, focus of this research was to develop techniques to help improve performance of soldiers with high mental loads. Although his research produced a lot of insight on methods of cognitive biofeedback, he did not produce any usable devices [11].

DARPA later expanded its focus on biocybernetics. The goal was exploring the potential of controlling devices with biological signals in real time through computer processing. In 1977 Jaques Vidal, with funding from DARPA, was able to produce one of the first BCI. The term BCI was first coined by Vidal from the University of California Los Angeles (UCLA) Brain-Computer Interface Laboratory. Vidal proved that electrical brain activity could be used to effectively communicate the user's intent. His experiment showed the possibility of moving a cursor through a 2 dimensional maze using only visual evoked potentials. Vidal's work along with others, proved that electrical brain activity can be interpreted. Future work expanded on EEG and other brain imaging devices into BCI [11].

BCI in Video Games

BCI work in video games has always been an idea thrown a lot works of art, and entertainment area of the consumer market. This concept of BCI in video games is becoming more of a reality now with concept of EEG wearables, as well as the price computing power that has gone down significantly since the 1970s. EEG wearables are made with the intent on being economical yet powerful, and portable. The Emotiv EPOC+ is an example of EEG wearable, with an MSRP of US$799 [10], and light enough to be work comfortably. Although BCI in video games are not common now, a niche that is starting to adopt them is VR. The rendering power of consumer video graphics card (VGC) is approaching the visual realism is cinematic quality, but in real time. The first implementing of VR systems could cost over US$200,000 10-15 years ago [12]. Now a VR ready hardware, as defined by Oculus Rift [13] can cost as low as US$1000.

Now that the hardware has been made more readily available, the only thing left is software. In an IEEE journal on Software Architecture [14], the authors discuss the popularity of BCI. More specifically “the computer’s ability to recognize human emotional states given physiological signals.” Despite its popularity, the journal continues quoting, “there are few frameworks, libraries, architectures, or software tools, which allow systems developers to easily integrate emotion recognition into their software projects.” This absence of a framework for emotional recognition and other BCI is what’s making it difficult for BCI to exist in videogames. The journal proposes its own framework on how to architect a computer base emotion recognition framework, using video games to test the user’s frustration. Companies like Emotiv are also providing their own SDKs to developers to help with the software side of BCIs.

A thesis submitted by M. Moazzami to Michigan State University in 2012 proposes the use of machine learning in BCI with the Emotiv EPOC [15]. In the thesis Moazzami uses the built in machine learning tool provided by the Emotiv Environment to perform the training. In his experiment, Moazzami trains the program by performing a task, recording the EEG data, and sending that data to the Emotiv engine to do the learning. After two months of training, Moazzami was able to interface with a keypad on screen. Although the experiment work, the training took two months before Moazzami was able to interface with the gamepad. Two hours is too much for an average gamer to train and not play a game. In Moazzami’s experiment, the machine learning was done using the Emotiv Engine. Although it works, it doesn’t allow for the most flexibility and control of the machine learning process. What this thesis hopes to improve upon Moazammi’s is to shorten the machine training from two months to under an hour, to use machine learning engine that allows for more control, and to not take away from the user’s game time. This thesis plans to achieve this by training while gaming, and using a more open machine learning platform to hopefully achieve faster learning.

Another study on video games and BCI was done in January 2012 by a team at Laboratorium voor Neuro en Psychofysiologie [16]. In the conference paper, the team attempts their first try to allow a user to play a tactical video game using steady-state visual evoked potential (SSVEP) classifier with a commercial grade EEG device such as the Emotiv EPOC. SSVEP is visual stimuli to an observation. The researches built a tower defense style game, where the user can interface with the video game by staring at certain visual cues that flash at different frequencies. These cues when in the peripheral vision doesn't generate an SSVEP, but when the users would focus on these cues, the frequency at which the cues would flash would be picked up the Emotiv EPOC. The tests were successful with 7 out of 8 users. The conclusions of the paper were that healthy users of all ages could use BCI, users could control and play a game using BCI, a consumer grade EEG device was sufficient for BCI, and the user can control the game in real time. This research has the advantage of not having to train a response in order to play a game. The use of SSVEP, and developing their own game aided in this. However, this requires the video game to have a purpose controls for BCI. This greatly restricts the section of video games availability to only ones design for BCI. SSVEP also has other limitations. The controls for SSVEP are all visual, and appear on screen. Meaning the user will have to take their attention away from the action on the screen to control their game. This limits the variety of games to slow pace, or turn base games. This thesis hopes to improve on this by allowing the user to play a larger set of games, as this experiment doesn't rely on a game to be designed for BCI. Although the paper doesn't require any training, this thesis hopes to limit the training by masking it within playing the video game.

Chapter 3: Methodology

3.1 Introduction

The algorithm can be broken down into three main systems. These three systems are: data acquisition, machine learn, and machine predict.

Figure 9 shows the system in data acquisition mode. In data acquisition mode, both EEG data from the Emotiv EPOC+ and physical gamepad data are recorded and stored in a text file. Gamepad controller data is also fed into the virtual gamepad so the user can still play the game. The text file is then used to train the learner in in machine learning system.

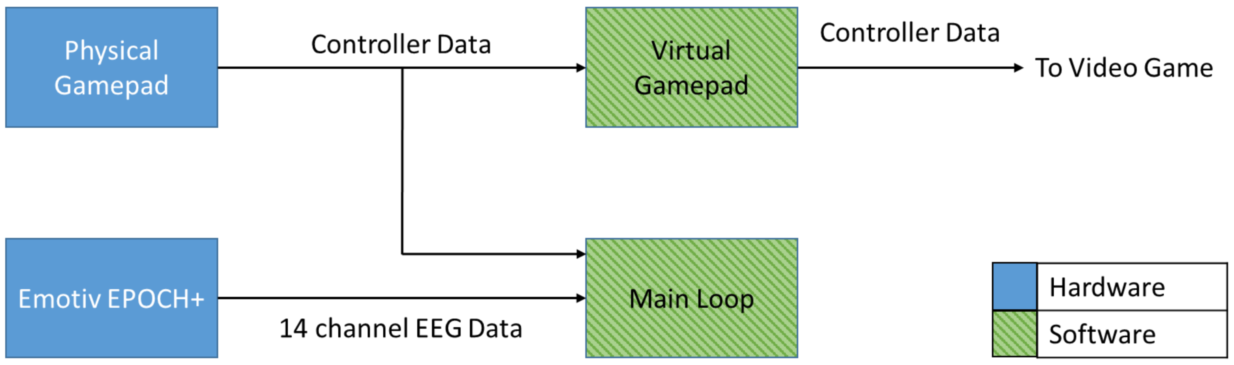


Figure 9: Block diagram of system in data acquisition mode

Machine learning is done using MATLAB classification learner. In machine learning mode, the saved EEG and controller data are used as independent and dependent data respectively. The EEG data is further featured extracted to get more valuable data for the learner. Several classification learners are trained and the best model is selected for machine predict mode.

Figure 10 shows the system in machine predict mode. In machine predict mode, the physical gamepad is used only to record the intended button press of the user, since the game should now be played using only the EEG data. Within the main loop the EEG data is being fed into the MATLAB machine learning model to predict the outcome.

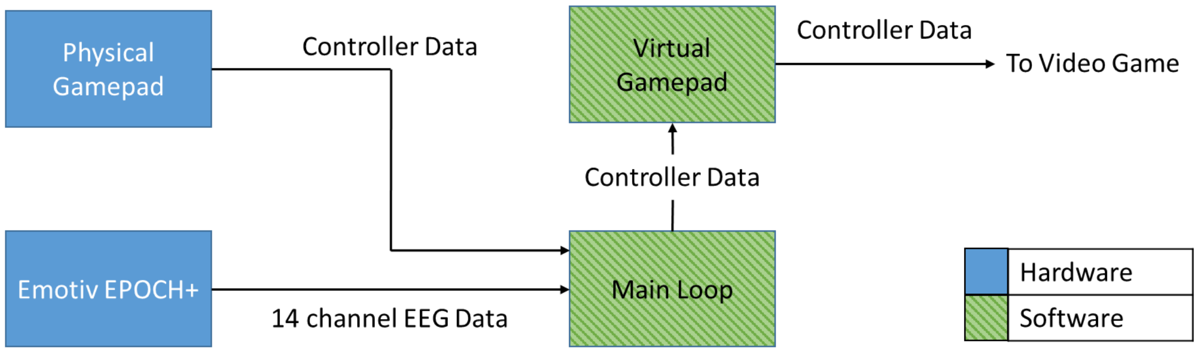


Figure 10: Block diagram of system in machine predict mode

3.2 Tools

Many different software and hardware tools were used in this research. This subchapter goes over the various tools used, and the justification for the use of that tool.

3.2.1 Programming Language

The main algorithm was written in Visual Studio 2015 using C++. C++ is a widely used programming language, and is often used by developers when providing software development kits (SDK). Of the SDK options available by software used in this research, C++ was a common language that all the developers provided. See **Table 2**.

SDK	C/C++	.Net	Java	Python	MATLAB
Emotiv EPOC+	Yes	Yes	Yes	Yes	Yes
Matlab	Yes	Yes	Yes	Yes	Yes
Direct Input	Yes	No	No	No	No
vJoy	Yes	No	No	No	No

Table 2: List of SDKs with supported software

3.2.2 Emotiv EPOC+

The Emotiv EPOC+, shown in **Figure 11**, is a wearable EEG device developed for BCI. This device was chosen for its high sampling rate, portability, and ease of use. It has 14 EEG channels, with a sampling rate of 128 sps, 14-bit ADC resolution (0.51 uV per bit), with Bluetooth Smart, and proprietary 2.4GHz wireless. [10] With its 128-bit sampling rate, it can capture brain waves signals up to 64 Hz. Which is enough to capture Gamma brain waves while naturally filtering out high frequency artifacts.

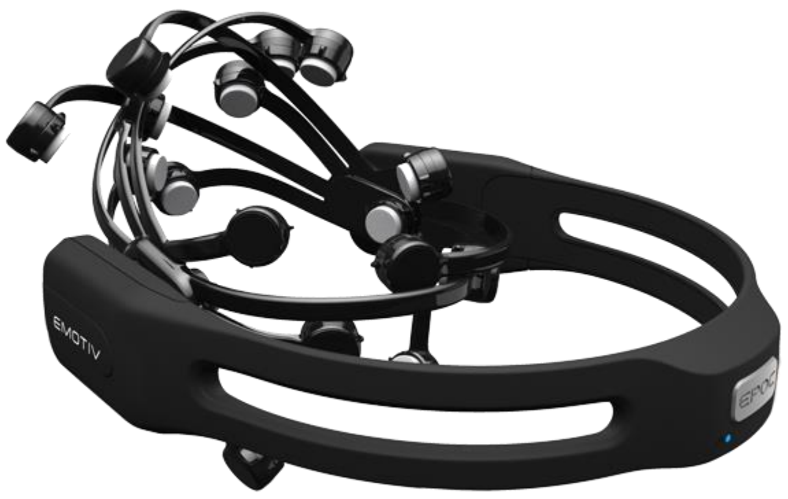


Figure 11: Image of Emotiv EPOC+ [10]

The Emotiv EPOC+ comes with two programs Emotiv Xavier Control Panel, and Emotiv Xavier Testbench. These two programs are great to get the user to get acquainted with EEG and brain monitoring, but do not do much for developing BCI applications. The Emotiv EPOC+ SDK, on the other hand, is used for BCI development. The SDK comes with the necessary drivers and code to allow the software to interface with hardware. Although the SDK comes in many languages, C++ was chosen since it has the highest support among the Emotiv community.

3.2.3 MATLAB

MATLAB is a multi-paradigm numerical computing environment [18]. It is used in multiple applications, such as: computer vision, image processing, data visualization, and machine learning. This research uses MATLAB to develop its machine learning model. Built into MATLAB is a simple to use Classification Learner application. This application has multiple learners built in that the user can train, and export their model. However, this model is locked in the MATLAB environment. In order to use the model, the C++ code would have to launch the MATLAB engine using the MATLAB C++ SDK.

3.2.4 Direct Input

To take in controller inputs, the Microsoft standard Direct Input SDK was used. This universal controller input [19] allows for a variety of controllers to work with the algorithm, but restricts the program to only work in a windows environment. A Gamecube controller was used in this research, although any controller will work.

3.2.5 vJoy

To emulate a virtual controller, the vJoy SDK was used. vJoy is a device driver that bridges the gap between and device that Is not a joystick and an application that requires a joystick [20]. This SDK allows the programmer to feed in controller inputs to virtual controller. This virtual controller is seen as a physical controller to the computer, and video games. The vJoy SDK was the deciding factor in which to write the algorithm in, since it is the only open source virtual controller with a programmable feeder available at the time of this research.

3.3 Algorithm

As stated before, the algorithm can be broken down into three main systems: data acquisition, machine learn, and machine predict. A full cycle of operation would start with the algorithm in data acquisition mode in a C++ program. This program would terminate once enough EEG and controller samples are taken, and saved to a text file (typically >1000). Afterwards in MATLAB, the samples would be used to teach a machine learning model. Once the best model is found, the same C++ program would start again, this time in machine predict mode. In this mode instead of samples being saved to a text file, EEG samples are sent to the MATLAB engine running in parallel, and controller samples are ignored. The MATLAB engine would then return a predicted value base of the model built earlier.

The algorithm for data acquisition and machine predict is detailed in the flowchart **Figures 12, 13, 14, 15, and 17**. The systems data acquisition, and machine predict are shown in **Figures 12, 13, 14, and 15**, while machine learn is shown in **Figure 17**.

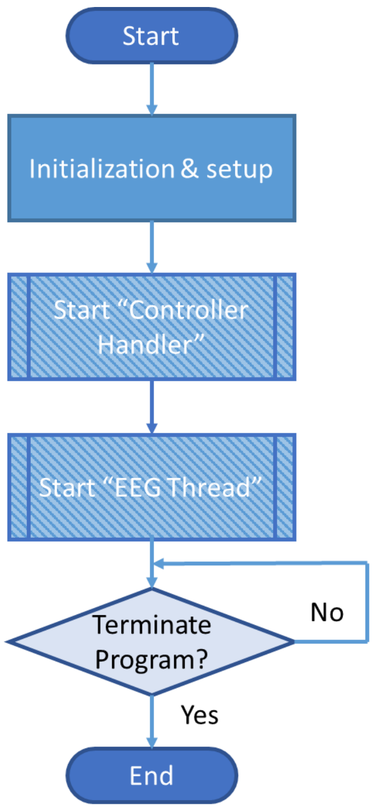


Figure 12: Initialization and setup flow chart

The flow chart in Figure describes the main loop. The main loop initializes the necessary setup for the vJoy virtual controller, and the Emotiv EPOC+ EEG device. Along with initialization, it also starts the physical controller handler, and the EEG thread.

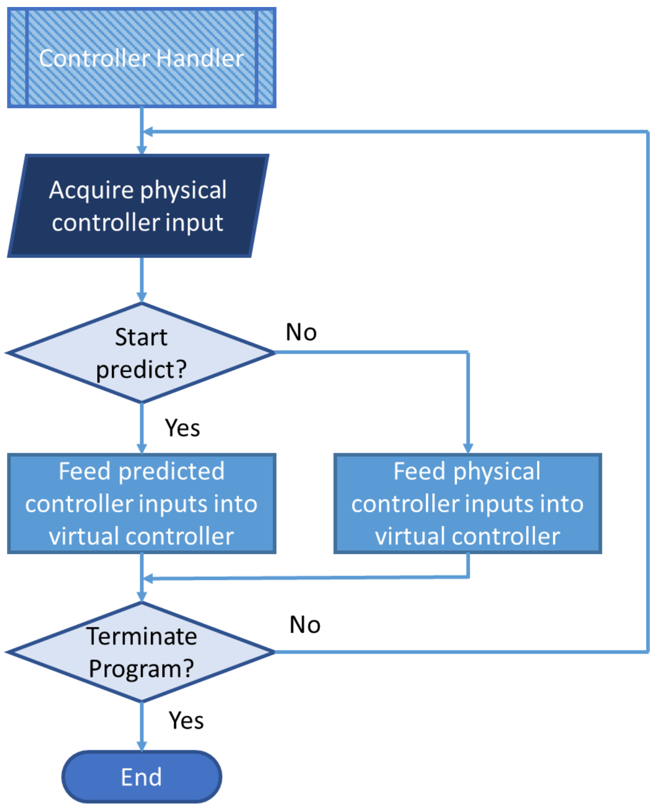

Controller Handler
Acquire physical controller input
Start predict?
No
Yes
Feed predicted controller inputs into virtual controller
Feed physical controller inputs into virtual controller
Terminate Program?
No
Yes
End

Figure 13: Controller handler subroutine flowchart

The flow chart in **Figure 13** describes how the Controller handler subroutine works. It starts by acquiring the physical controller input using Microsoft's Direct Input API. If the program is in data acquisition mode, it would feed the physical controller inputs to the vJoy virtual controller. If the program is in machine predict mode, it would feed the predicted controller inputs to the virtual controller. This predicted controller input is a global struct that is updated in the EEG thread subroutine.

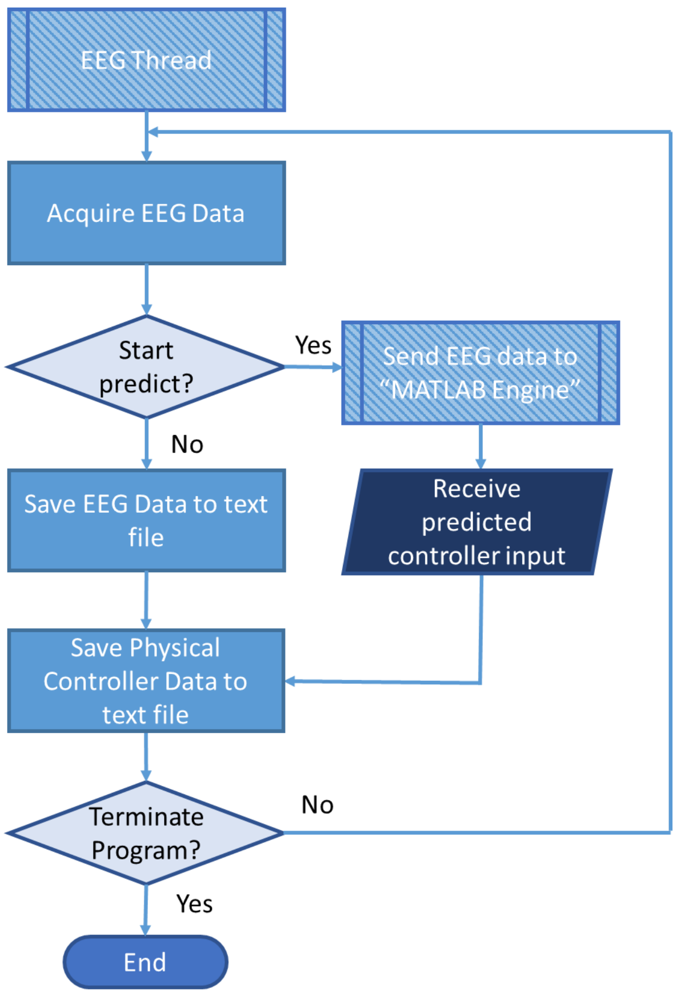

EEG Thread
Acquire EEG Data
Start predict?
Yes
Send EEG data to "MATLAB Engine"
No
Save EEG Data to text file
Receive predicted controller input
Save Physical Controller Data to text file
Terminate Program?
No
Yes
End

Figure 14: EEG thread subroutine flowchart

Figure 14 shows the flow chart of the EEG Thread sub routine. It starts by acquiring the EEG data from the Emotiv EPOC+. They are a total of 22 channels that are read from the EPOC+. These are an index counter, 14 EEG channels, 2 gyroscope axis, time stamp, function ID, function value, markers, and sync signal. Only the 14 EEG channels are utilized. If the program is in data acquisition mode, it reads the 14 EEG channels, and saves them to separate text files every 32 cycles. Each cycle it reads one sample of data from each channel. With a sampling rate of 128 sps, saving data every 32 cycle, is equivalent to saving a snapshot of the previous quarter second of EEG data. At the end of every 32 cycle the controller data is also saved. This is done in the EEG thread subroutine instead of the controller handler subroutine because this will better synchronize the two sets of data.

During machine predict mode, the EEG data is sent to the “MATLAB engine” sub routine instead. The MATLAB sub routine would then return the predicted controller button. Finally, it would save the physical button data to a text file as a means of validating that the predicted button matches the intended button.

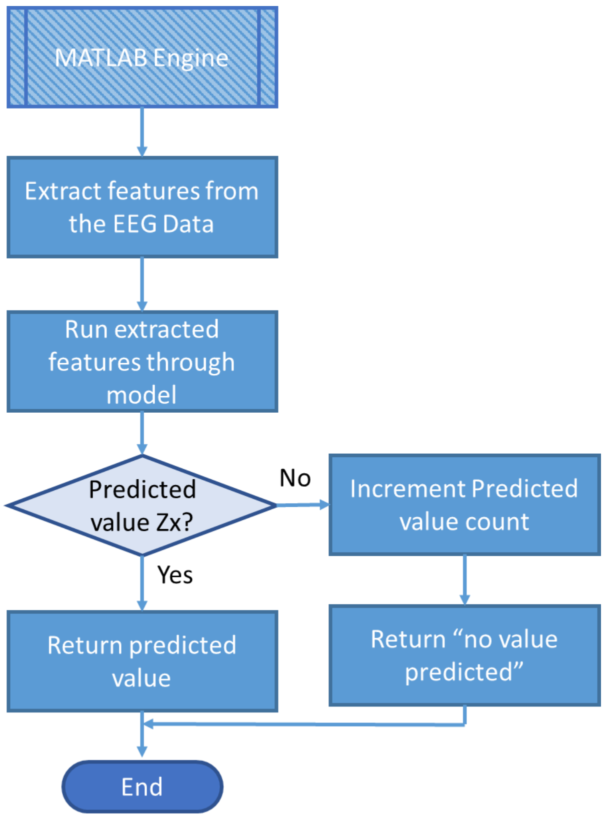

MATLAB Engine
Extract features from the EEG Data
Run extracted features through model
Predicted value Zx?
No
Yes
Increment Predicted value count
Return predicted value
Return “no value predicted”
End

Figure 15: MATLAB engine subroutine Flow chart

Figure 15 shows the MATLAB engine sub routine. This subroutine is only present during machine predict mode. In this mode the EEG data is sent to the MATLAB engine. Once in the MATLAB engine, features are extracted from the EEG data. The process of feature extraction starts filtering the data with a high pass 0.5 Hz finite impulse response (FIR) filter. Then by converting the EEG time domain signals to frequency domain signals. Then certain band widths are assigned to their respective brain wave. See **Figure 5**. The mean and standard deviation is then calculated, and sent to the model that was trained during machine learn. This then returns a predicted value to be fed to the vJoy virtual controller.

A condition then checks to see if that particular prediction has been predicted Z amount times, where Z is a value large enough to return a probable correct prediction. This calculation is based off **Equation 1**, where P_{cn} is the probability of a correct prediction for button n (not the probity that button n will be predicted), P_{FNRn} is the probability of false negative rate (or percentage of a false negative) for button n in the classification learner model where the predicted class did not match the true class (in other words, how often the classification learner was wrong). The P_{FNRn} is the sum of all the false negative for that button n. False negatives occur when a prediction incorrectly indicates that a class is absent. The number of false negative that make up the sum P_{FNRn} is n - 1. Based on this equation, the lower the P_{FNRn} the greater the P_{cn}.

$$P_{cn} = 1 - (P_{FNR})^{Z}$$

Equation 1: Probability of guessing correctly equation

The first button to be predicted Z amount of times becomes the final prediction. Until then it returns no value, or no button predicted. In theory this should predict the correct value most of the time since it's the first to Z predicted values before it finalizes on a prediction, and from **Equation 1** the larger Z is the more probable it is a correct prediction. This is true for P_{FNR} probabilities that are high, but are still less than P_{TPRn}, or probability of true positive rate for button n (probability that the predicted class matched the true class). P_{TPRn} and P_{FNRn} are mutually exclusive, and a complete subset. True positives occur when a prediction correct indicates that a class is present. In other words, as long as $P_{TPRn} > P_{FPRn}$, and with a large enough Z, P_{cn} will have a high probability of correct predictability. Calculations of P_c with various values of P_{NFRn} and Z are shown in **Table 3**.

Z = 1

P_c	P_{NFRn}
0.55	0.45
0.65	0.35
0.75	0.25
0.85	0.15
0.95	0.05

Z = 2

P_c	P_{NFRn}
0.7975	0.45
0.8775	0.35
0.9375	0.25
0.9775	0.15
0.9975	0.05

Z = 3

P_c	P_{NFRn}
0.9089	0.45
0.9571	0.35
0.9844	0.25
0.9966	0.15
0.9999	0.05

Z = 4

P_c	P_{NFRn}
0.959	0.45
0.985	0.35
0.9961	0.25
0.9995	0.15
1	0.05

Z = 5

P_c	P_{NFRn}
0.9815	0.45
0.9947	0.35
0.999	0.25
0.9999	0.15
1	0.05

Table 3: Calculated values of P_c with various values of P_{NFRn} and Z

Even with P_{FNRn} values close to but still less than P_{TPRn}, **Equation 1**, still holds, since P_{FNRn} is the sum of all false negatives, and as long as no individual false negative is close to the P_{TPRn}. In summary if the true positive probability is higher than any individual false negative probability, and checking to see if that prediction has been predicted multiple time, is a highly probable predictor. An example of false negative and true positives is shown in **Figure 16**.

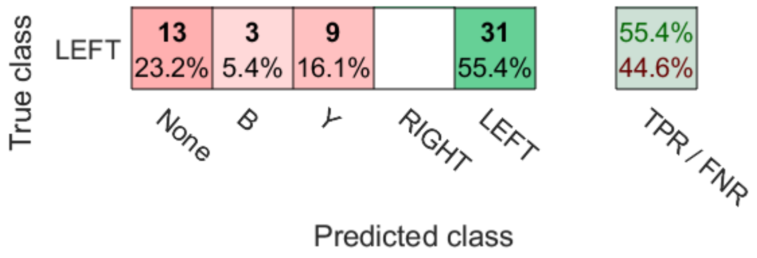


Figure 16: False negative and true positive probabilities

In **Figure 16** the sum of the false negatives is 44.6%, but since the prediction of "None" doesn't hurt the predictor, the adjusted $P_{FNRLeft}$ is 12/43, or 27.9%, which works out to a P_{cLeft} of 97.8% with Z = 3. In this example, the probability of picking left is much higher than the probability of picking B, Y, or Right, therefore there is a high probability of it true positive and satisfy the condition of predicting the value Z times in the flow chart in **Figure 15**.

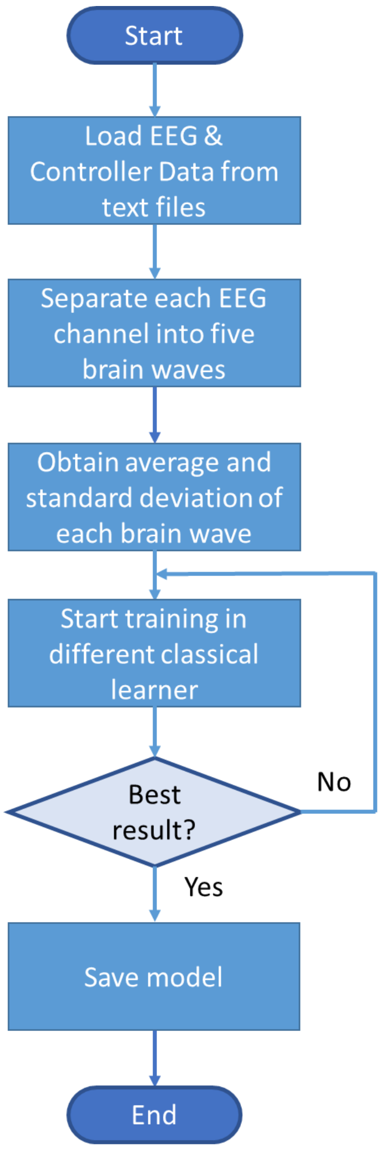

Start
Load EEG & Controller Data from text files
Separate each EEG channel into five brain waves
Obtain average and standard deviation of each brain wave
Start training in different classical learner
Best result?
No
Yes
Save model
End

Figure 17: MATLAB machine learn system

The entire machine learning process, shown in **Figure 17**, is done in the MATLAB environment. It starts by loading the EEG and controller data text files, and converting these into MATLAB matrixes. The EEG matrixes are 32 x Z, and the controller matrix is 1 x Z, where "Z" is the length that the data acquisition has ran for in units of quarter seconds. Each EEG signal is then filtered through a 0.5 Hz high pass FIR filter, and converted to a frequency domain signal. This converts the fourteen 32 x Z matrix into fourteen 512 x Z matrix, where the first element of m (where 512 x Z is an m x n matrix) represents the magnitude at 0 Hz, and the 512th element representing the magnitude at 64 Hz (from Nyquist Theorem, and a sampling rate of 128 sps).

The frequency domain signal is then separated into different band widths representing the brain waves. These divisions are shown in **Table 4**.

Brain Wave	Frequency (Hz)	Elements in matrix
Delta	1-4	1 to 24
Theta	4-7	25 to 56
Alpha	8-15	57 to 120
Beta	16-31	121 to 244
Gamma	31+	245 to 512

Table 4: Division of frequencies in frequency domain matrix

Separating the signals converts the fourteen 512 x Z matrix into seventy W x Z matrixes (where W represents the different length for each brain wave band width). The mean and standard deviation of each signal is then calculated, converting seventy W x Z matrixes into 140 1 x Z matrixes. These matrixes represent the independent data that will be fed into the classification learner.

The controller matrix is converted from its numerical representation, to its functional representation shown in **Table 5**.

Numerical Button Representation	Functional Representation
0	None
1	X
2	A
4	B
8	Y
16	L TRIGGER
32	R TRIGGER
64	INVALID 8
128	Z
256	INVALID 10
512	START
1024	INVALID 11
2048	INVALID 12
4096	UP
8192	RIGHT
16384	DOWN
32768	LEFT

Table 5: Numerical to Functional button mapping

After this conversation, both the EEG brain wave data, and the controller data are combined into a 141 x Z table. This table is then used as the independent and dependent data in MATLAB's classification learner. The learner has multiple different classification learners built in, shown in **Figure 18**.

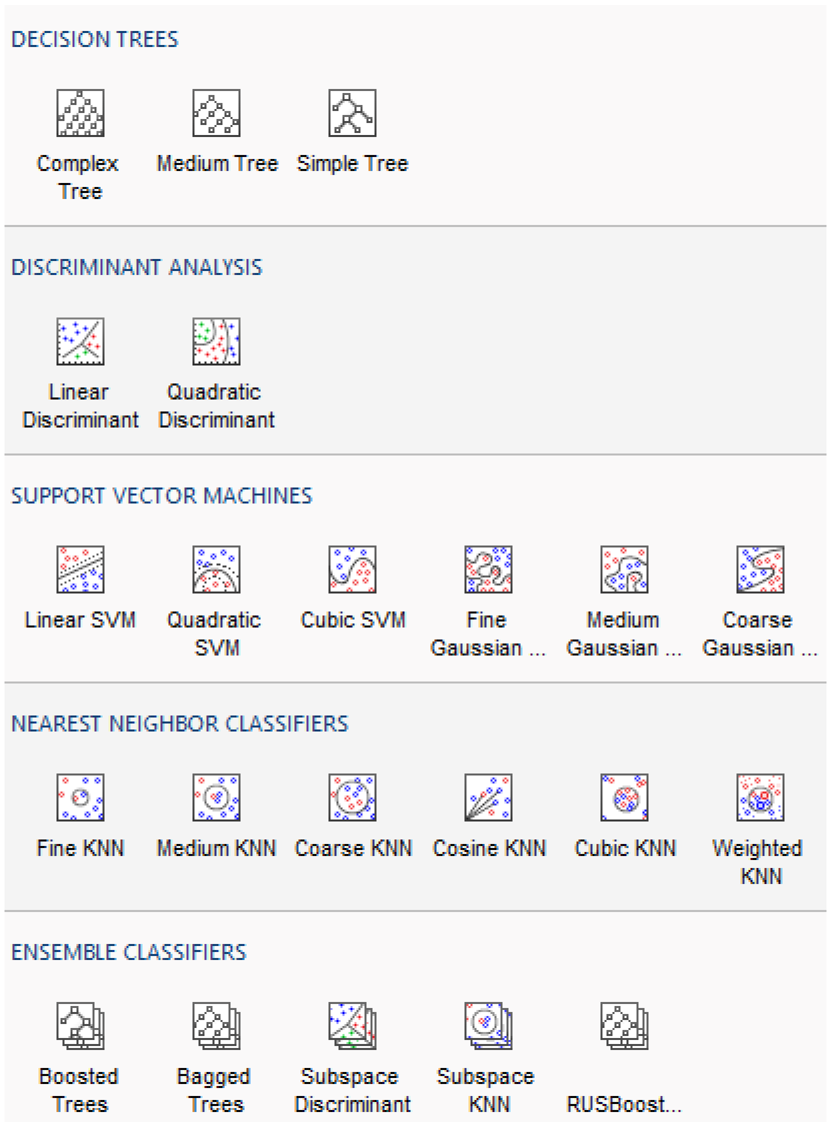
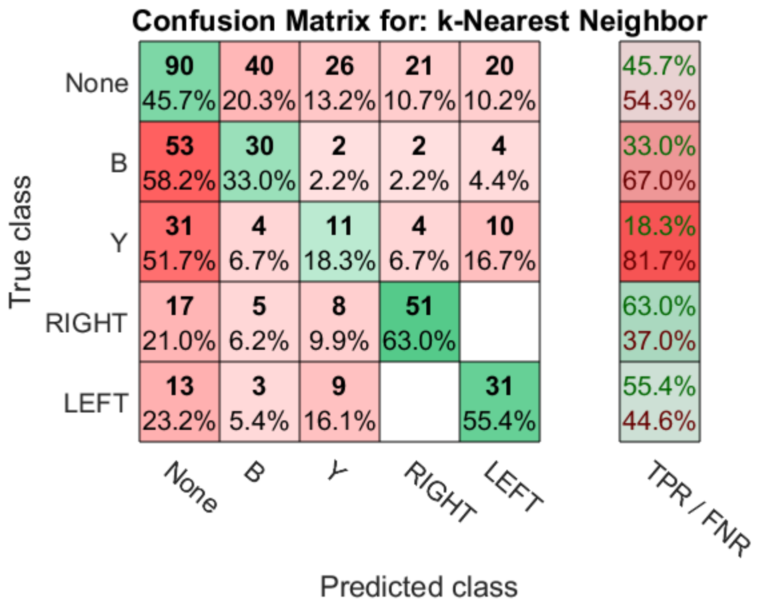


Figure 18: List of classification learner built into MATLAB

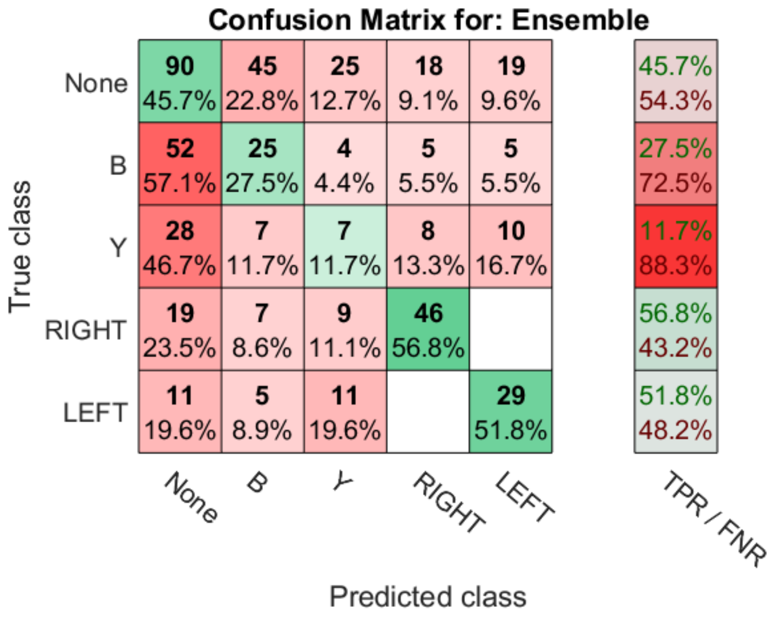
These learners have a default setting, but may be further tuned in the advance tab. After selecting a trainer and training, the confusion matrix could be brought up to see the classifier performance in each class for the selected model. This is shown for Fine K-Nearest Neighbor (KNN) in **Figure 19**, and Subspace Discriminant in **Figure 20**.



Confusion Matrix for: k-Nearest Neighbor

True class \ Predicted class	None	B	Y	RIGHT	LEFT	TPR / FNR
None	90 45.7%	40 20.3%	26 13.2%	21 10.7%	20 10.2%	45.7% 54.3%
B	53 58.2%	30 33.0%	2 2.2%	2 2.2%	4 4.4%	33.0% 67.0%
Y	31 51.7%	4 6.7%	11 18.3%	4 6.7%	10 16.7%	18.3% 81.7%
RIGHT	17 21.0%	5 6.2%	8 9.9%	51 63.0%		63.0% 37.0%
LEFT	13 23.2%	3 5.4%	9 16.1%		31 55.4%	55.4% 44.6%

Figure 19: Confusion matrix for classification learner Fine KNN

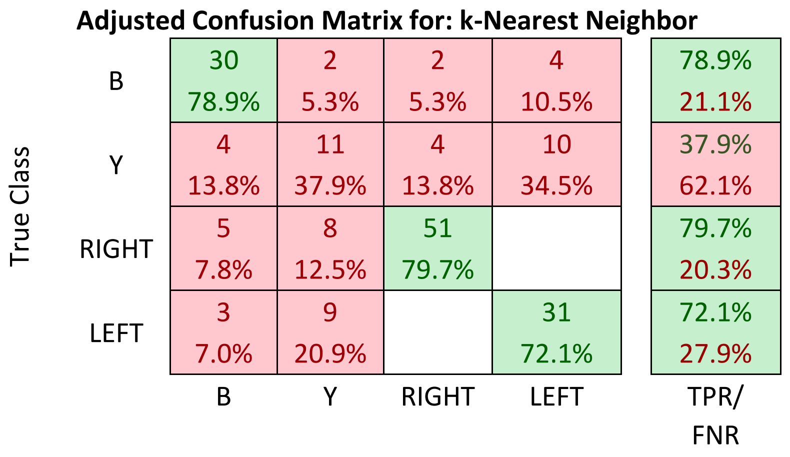


Confusion Matrix for: Ensemble

True class \ Predicted class	None	B	Y	RIGHT	LEFT	TPR / FNR
None	90 45.7%	45 22.8%	25 12.7%	18 9.1%	19 9.6%	45.7% 54.3%
B	52 57.1%	25 27.5%	4 4.4%	5 5.5%	5 5.5%	27.5% 72.5%
Y	28 46.7%	7 11.7%	7 11.7%	8 13.3%	10 16.7%	11.7% 88.3%
RIGHT	19 23.5%	7 8.6%	9 11.1%	46 56.8%		56.8% 43.2%
LEFT	11 19.6%	5 8.9%	11 19.6%		29 51.8%	51.8% 48.2%

Figure 20: Confusion matrix for classification learner Subspace Discriminant

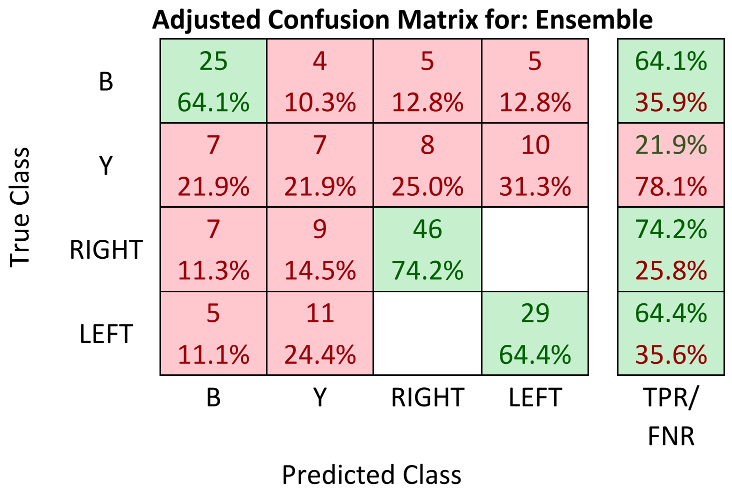
These two are examples of two different learners. In order to evaluate which learner is better, the adjusted TPR/FNR must be calculated. This adjustment is to account for the fact that “None” guesses don’t hurt how well it predicts, only how quick it predicts. The adjusted values for both are shown in **Figures 21 and 22** for KNN and subspace discriminant respectively.



Adjusted Confusion Matrix for: k-Nearest Neighbor

True Class	B	Y	RIGHT	LEFT	TPR/ FNR
B	30 78.9%	2 5.3%	2 5.3%	4 10.5%	78.9% 21.1%
Y	4 13.8%	11 37.9%	4 13.8%	10 34.5%	37.9% 62.1%
RIGHT	5 7.8%	8 12.5%	51 79.7%		79.7% 20.3%
LEFT	3 7.0%	9 20.9%		31 72.1%	72.1% 27.9%

Figure 21: Adjusted Confusion matrix for classification learner Fine KNN



Adjusted Confusion Matrix for: Ensemble

True Class	B	Y	RIGHT	LEFT	TPR/ FNR
B	25 64.1%	4 10.3%	5 12.8%	5 12.8%	64.1% 35.9%
Y	7 21.9%	7 21.9%	8 25.0%	10 31.3%	21.9% 78.1%
RIGHT	7 11.3%	9 14.5%	46 74.2%		74.2% 25.8%
LEFT	5 11.1%	11 24.4%		29 64.4%	64.4% 35.6%

Predicted Class

Figure 22: Adjusted Confusion matrix for classification learner Subspace Discriminant

After the adjustment it is shown that the fine KNN learner performed better than the Subspace Discriminant. Looking at the fine KNN learner though, it seems the FNR is much higher than the TPR for button Y. This leads to a higher P_{FNRY} which makes the predictor for Y to be unreliable according to **Equation 1**, with a P_{cY} of 76.1% with Z = 3. For this particular Learner, Y is not a possible button for prediction, and is removed from the subset of predicted values. However, buttons B, Right and Left perform very well, with P_{cn} of 99.1%, 99.2%, and 97.8%

respectively. Selecting classification learner fine KNN as the model to be used in machine predict mode.

Chapter 4: Findings

4.1 Intro

The following experiment was done using a computer desktop with the following specs: Intel core i5 4690k at 3.5 GHz, Nvidia GTX 970, 8 Gb 16MHz DDR3 RAM. The video game under test is Tetris Worlds running on the Wii emulator Dolphin version 4.0.

A baseline procedure for testing was develop for consistency in the experiment. During data acquisition two sets of data would be taken. Each would acquire data for about 5 minutes each, providing about 2000 data points, with a couple minutes in between sets to reset. During machine learn, the best overall adjusted TNR would be selected as the predicting model. Any buttons that were not viable to test would be thrown out and not predicted. During machine predict the user would use their mind to play the game with the virtual controller, and use the physical controller to record the true result. This is done to verify predicted results are correct with what the user intended. The physical control data during machine predict would not be passed on to the virtual controller. Z would be set to 3. Before running the experiment, it is vital to test the contact quality of the EEG pads, and make sure they are green during data acquisition and machine predict as shown in **Figure 23**.

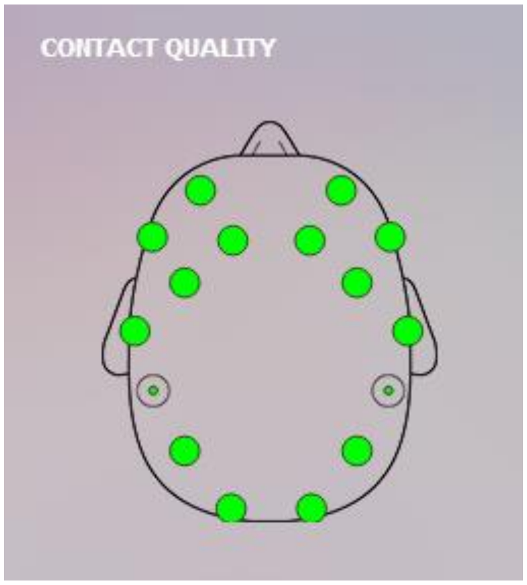
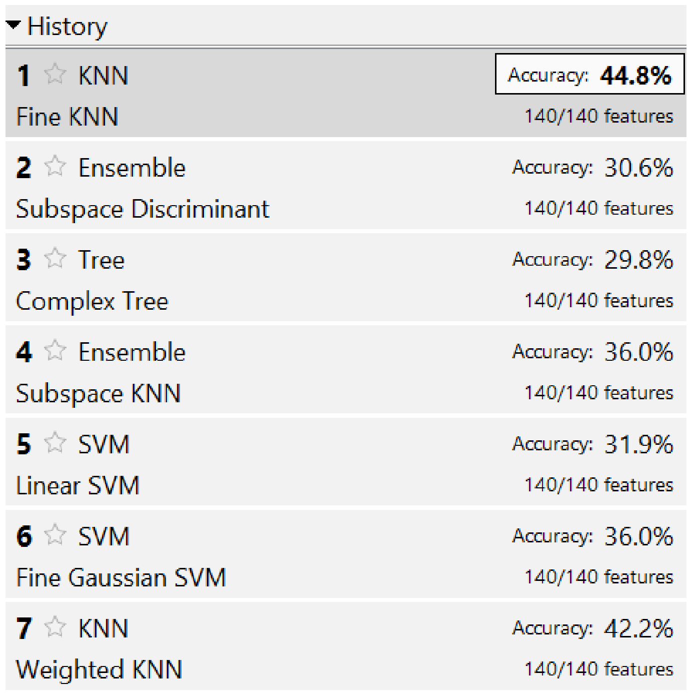


Figure 23: Contact quality check in Emotiv Xavier software

A total of four buttons are used for this experiment: “Right” to move the Tetris piece right, “Left” to move left, “B” to rotate, and “Y” to drop.

4.2 Experiment

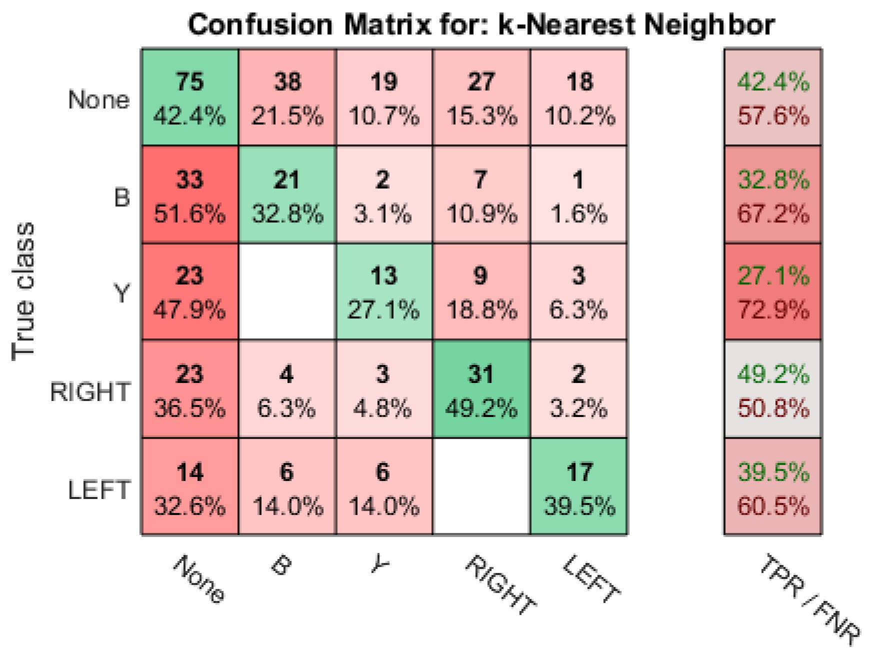
After two games of Tetris, both sets of EEG and controller data were saved during data acquisition. After being imported into MATLAB, and featured extracted, they were used to teach the classification learner. The technique of feature extraction is demonstrated in Chapter 3. From a series of teaching with various classification learners, the one with the best accuracy is selected for both sets of data. In the case of both, Fine KNN was the best result. This is shown for data set 2 in **Figure 24**.



1 ☆ KNN	Accuracy: **44.8%**
Fine KNN	140/140 features
2 ☆ Ensemble	Accuracy: 30.6%
Subspace Discriminant	140/140 features
3 ☆ Tree	Accuracy: 29.8%
Complex Tree	140/140 features
4 ☆ Ensemble	Accuracy: 36.0%
Subspace KNN	140/140 features
5 ☆ SVM	Accuracy: 31.9%
Linear SVM	140/140 features
6 ☆ SVM	Accuracy: 36.0%
Fine Gaussian SVM	140/140 features
7 ☆ KNN	Accuracy: 42.2%
Weighted KNN	140/140 features

Figure 24: List of classification learners taught for data set 2

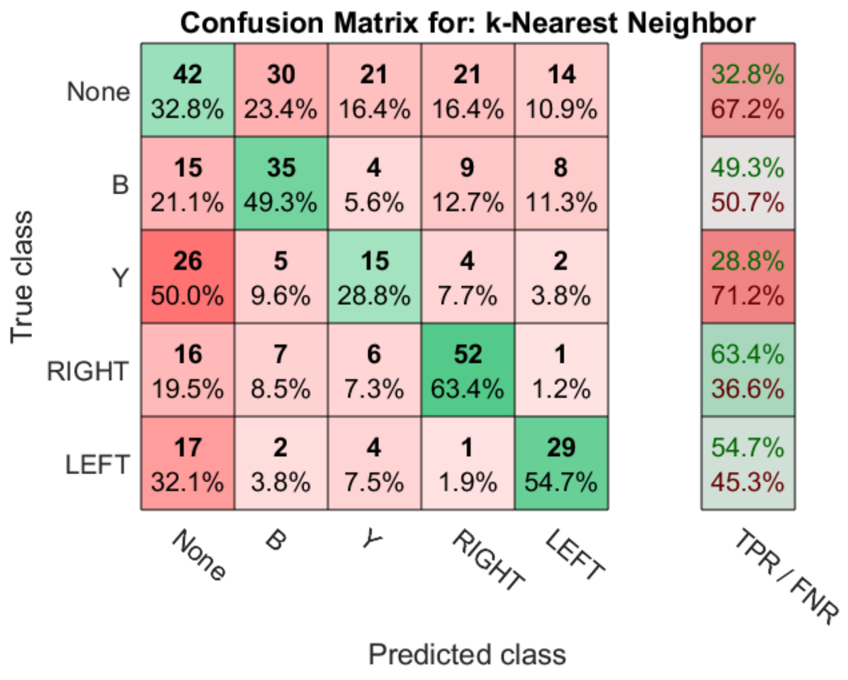
The confusion matrix for data set 1, and data set 2 are shown in **Figures 25 and 26**. The adjusted matrix for data set 1 and data set 2 are shown in **Figures 27 and 28**.



Confusion Matrix for: k-Nearest Neighbor

True class \ Predicted class	None	B	Y	RIGHT	LEFT		TPR / FNR
None	75 42.4%	38 21.5%	19 10.7%	27 15.3%	18 10.2%		42.4% 57.6%
B	33 51.6%	21 32.8%	2 3.1%	7 10.9%	1 1.6%		32.8% 67.2%
Y	23 47.9%		13 27.1%	9 18.8%	3 6.3%		27.1% 72.9%
RIGHT	23 36.5%	4 6.3%	3 4.8%	31 49.2%	2 3.2%		49.2% 50.8%
LEFT	14 32.6%	6 14.0%	6 14.0%		17 39.5%		39.5% 60.5%

Figure 25: Confusion Matrix for Fine KNN for data set 1



Confusion Matrix for: k-Nearest Neighbor

True class \ Predicted class	None	B	Y	RIGHT	LEFT		TPR / FNR
None	42 32.8%	30 23.4%	21 16.4%	21 16.4%	14 10.9%		32.8% 67.2%
B	15 21.1%	35 49.3%	4 5.6%	9 12.7%	8 11.3%		49.3% 50.7%
Y	26 50.0%	5 9.6%	15 28.8%	4 7.7%	2 3.8%		28.8% 71.2%
RIGHT	16 19.5%	7 8.5%	6 7.3%	52 63.4%	1 1.2%		63.4% 36.6%
LEFT	17 32.1%	2 3.8%	4 7.5%	1 1.9%	29 54.7%		54.7% 45.3%

Figure 26: Confusion Matrix for Fine KNN for data set 2

Adjusted Confusion Matrix for: K-Nearest Neighbor

True Class \ Predicted Class	B	Y	RIGHT	LEFT	TPR/ FNR
B	21 67.7%	2 6.5%	7 22.6%	1 3.2%	67.7% 32.3%
Y		13 52.0%	9 36.0%	3 12.0%	52.0% 48.0%
RIGHT	4 10.0%	3 7.5%	31 77.5%	2 5.0%	77.5% 22.5%
LEFT	6 20.7%	6 20.7%		17 58.6%	58.6% 41.4%

Figure 27: Adjusted Confusion Matrix for Fine KNN for data set 1

Adjusted Confusion Matrix for: K-Nearest Neighbor

True Class \ Predicted Class	B	Y	RIGHT	LEFT	TPR/ FNR
B	35 62.5%	4 7.1%	9 16.1%	8 14.3%	62.5% 37.5%
Y	5 19.2%	15 57.7%	4 15.4%	2 7.7%	57.7% 42.3%
RIGHT	7 10.6%	6 9.1%	52 78.8%	1 1.5%	78.8% 21.2%
LEFT	2 5.6%	4 11.1%	1 2.8%	29 80.6%	80.6% 19.4%

Figure 28: Adjusted Confusion Matrix for Fine KNN for data set 2

Observing **Figures 27 and 28** shows a great P_{FNRn} for all cases. The corresponding P_{cn} is calculated in **Table 6**. Of the four buttons recorded, it appears the button Y is the weakest, with the lowest FNR percentage. From previous test, see **Figures 21 and 22**, button Y continues to have a poorest performance of the four.

P_c Calculation for Data set 1

button	P_c calc
B	96.6%
Y	85.9%
Right	98.86%
Left	92.91%

P_c Calculation for Data set 2

button	P_c calc
B	94.7%
Y	80.8%
Right	99.05%
Left	99.26%

Table 6: P_c button calculation for both sets of data

4.3 Results

Figures 29 and 30 show the plot comparison between the predicted button presses (top), and the intended button presses (bottom) using the model from data set 1. True positives occur when the colors of the top and bottom circles match. False negatives occur when they don't match. A calculation of the FNR and PNR has been calculated and compared to the FNR/PNR from **Figure 27**, in **Figure 31**.

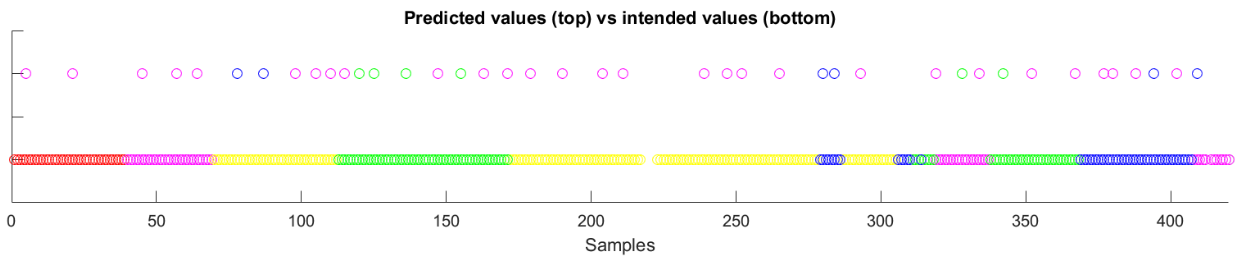


Figure 29: Comparison of Predicted to intended button presses, for first set of data
Red = "do nothing," magenta = "Right," yellow = "left," green = "Y," blue = "B"

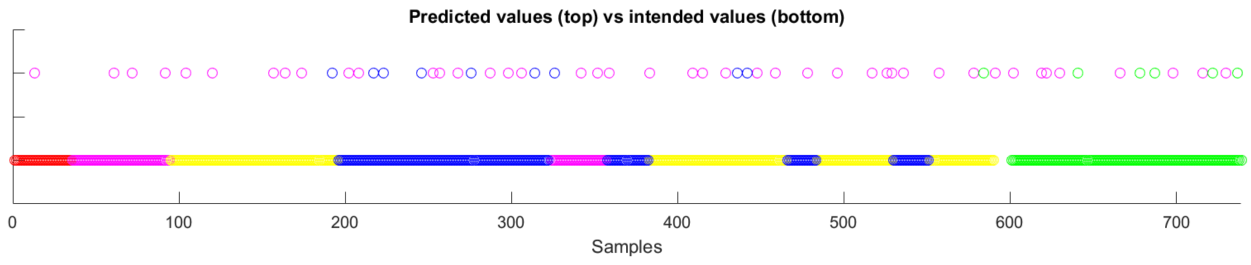


Figure 30: Comparison of Predicted to intended button presses for second iteration of first set of data
Red = "do nothing," magenta = "Right," yellow = "left," green = "Y," blue = "B"

	Original Machine Learning		First Pass		Second Pass	
B	21	67.7%	5	83.33%	5	83.33%
	9	32.3%	1	16.67%	1	16.67%
Y	13	52.0%	3	50.00%	5	55.56%
	12	48.0%	3	50.00%	4	44.44%
RIGHT	31	77.5%	5	17.24%	5	11.63%
	9	22.5%	24	82.76%	38	88.37%
LEFT	17	58.6%	0	0.00%	0	0.00%
	12	41.4%	0	0.00%	0	0.00%
	TP/ FN	TPR/ FNR	TP/ FN	TPR/ FNR	TP/ FN	TPR/ FNR

Figure 31: comparison of true positives and false negatives for the original machine learner, first pass prediction, and second pass prediction

According **Figure 31**, buttons “B” and “Y” perform on par or better than the original machined learning model. Meanwhile, the button “Right” was over predicted throughout the experiment, and the button “Left” didn’t predict at all.

From previous confusion matrixes, it seems obvious that right and left should have the highest true positive rate. This could be due to inconsistency of the conductivity of the EEG hydrator pads. As time goes by, the conductivity changes, typically going down. This change in conductivity changes the model of the machine learning algorithm. This can be seen between data set 1 and 2. Both are approximately taken ~5-7 minutes apart. As observed in the scatter plots of **Figures 32, 33, 34, and 35**, it can be observed that the plots do not appear to resemble each other. Some show high similarty such as alpha channels 11 and 12 in **Figure 35**. These are only 7 comparrisoins of a possible 9,870. With 140 sets of data, 9,870 is the total number of parings that could be compared without duplicates.

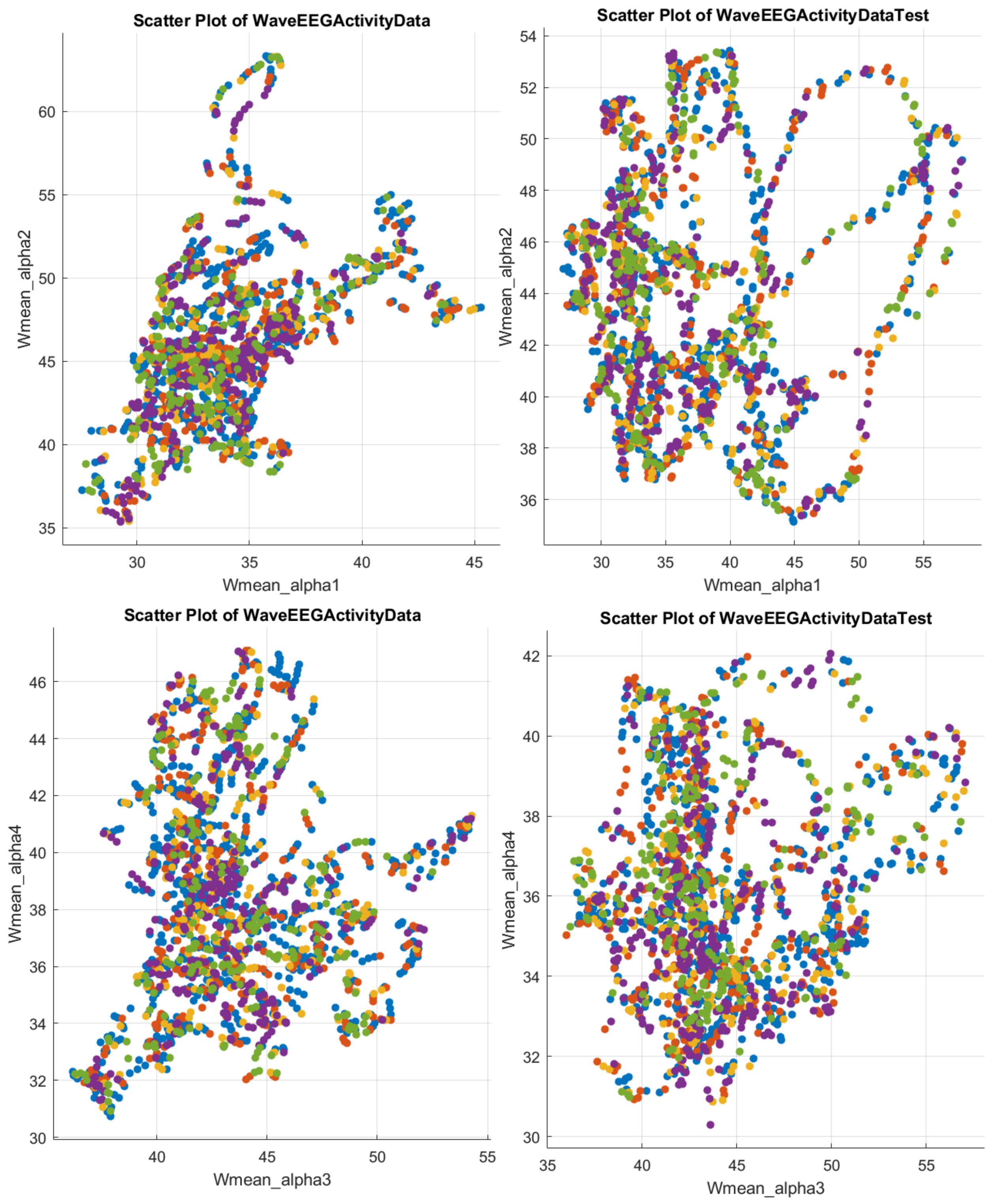


Figure 32: Comparison of the means of Alpha channels 1 - 4 (left-data set 1) (right-data set 2)

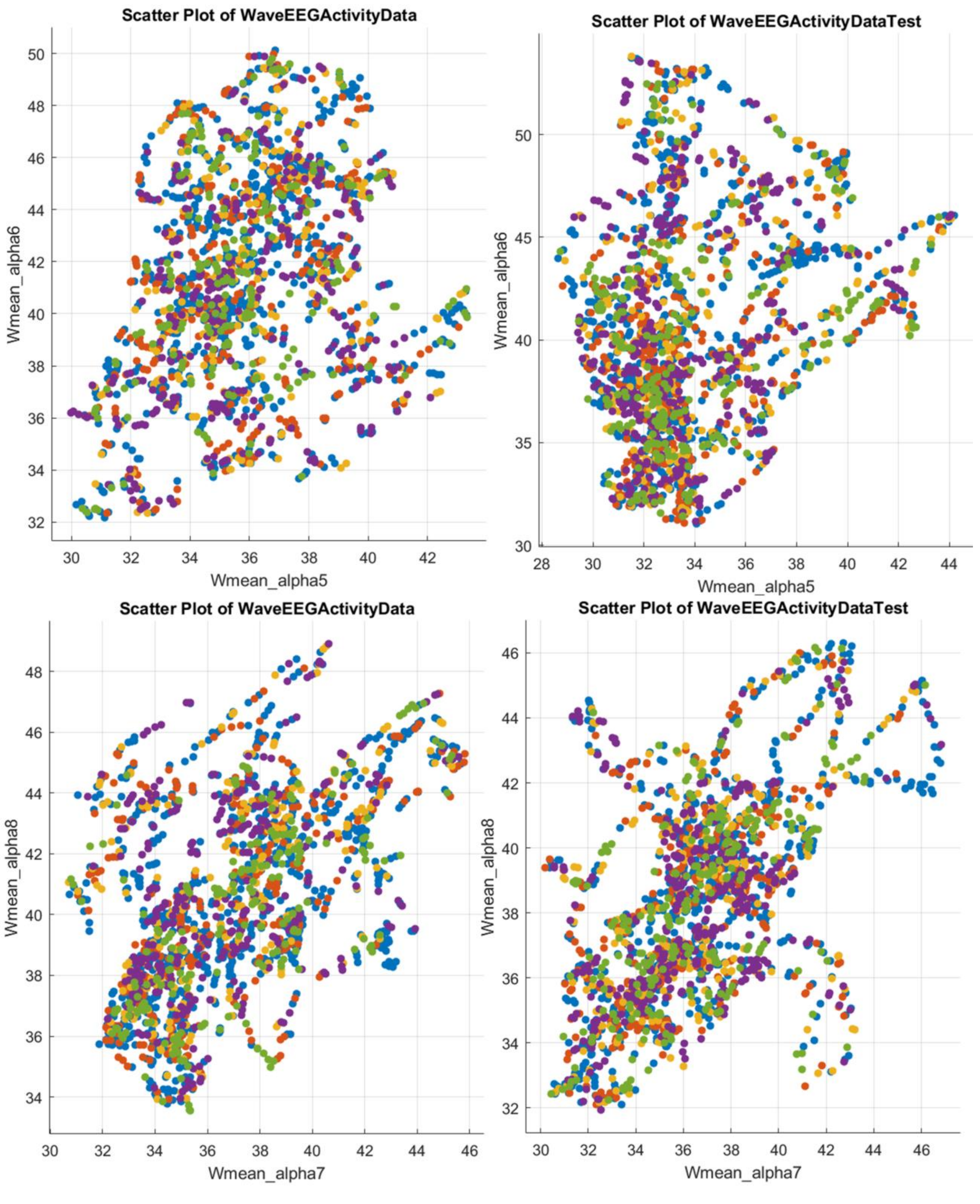


Figure 33: Comparison of the means of Alpha channels 5 - 8 (left-data set 1) (right-data set 2)

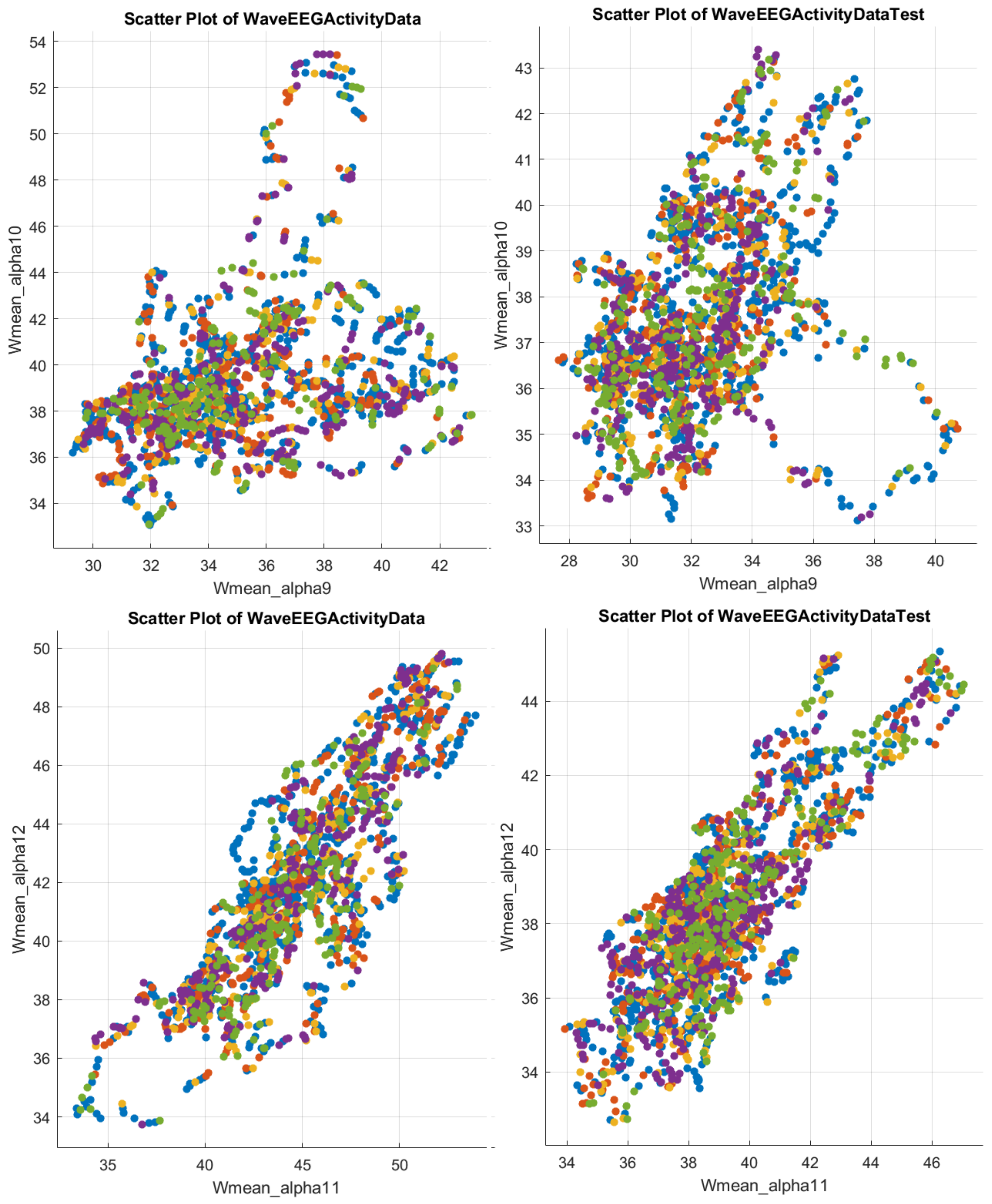


Figure 34: Comparison of the means of Alpha channels 8 - 12 (left-data set 1) (right-data set 2)

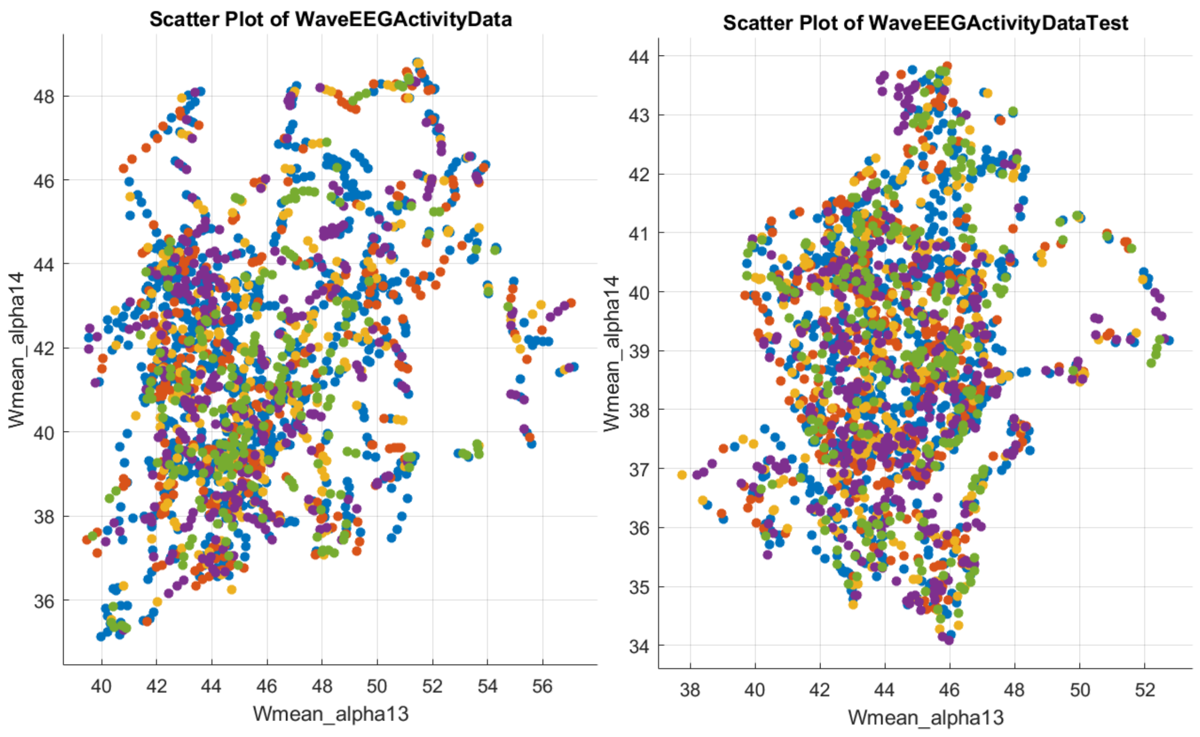


Figure 35: Comparison of the means of Alpha channels 13 -1 4 (left-data set 1) (right-data set 2)

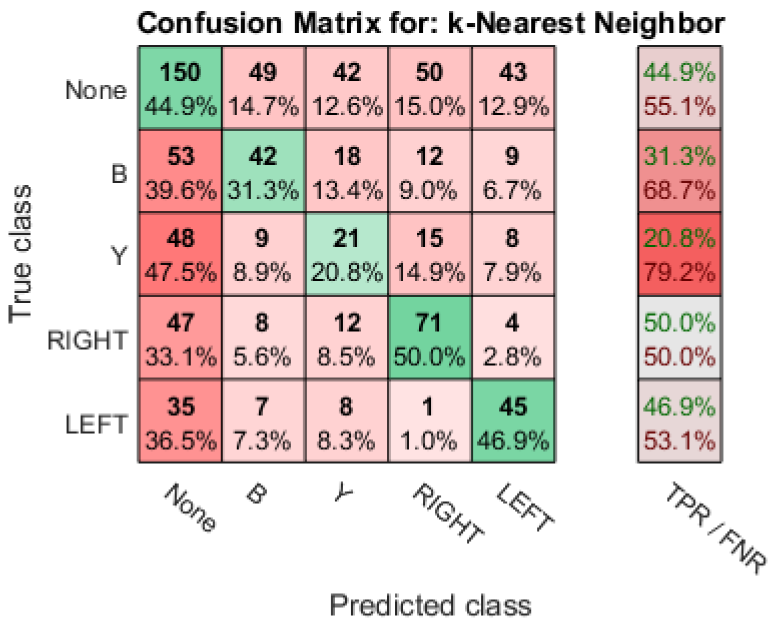
Another reason the two might data sets might not match is due to some cognitive state change during each trial. Though a ~5-7 minuite seperation might not seem long, it might be enough to skew the data. This experiment was repteated multiple times, all results are similar to the ones detailed in this thesis. **Table 7** show TPR and FNR of the each model of the two data sets attemping to valiadate eachothers model. Data set 1's data would be used to cross vailadate data set 2's model, and vise-versa.

Model 1 cross reference with data set 2		
Button	TPR	FNR
None	42.64%	57.36%
B	12.92%	87.08%
Y	13.03%	86.97%
Right	20.19%	79.81%
left	14.81%	85.19%
Total	24.39%	75.61%

Model 2 cross reference with data set 1		
Button	TPR	FNR
None	41.05%	58.95%
B	36.15%	63.85%
Y	34.27%	65.73%
Right	41.11%	58.89%
left	38.13%	61.87%
Total	38.01%	61.99%

Table 7: Cross Comparison of Machine Learning Models

Analysing the cross reference TPR, it can be observed that cross refference each model with eachother's data set will yeild an over all worst prediction than the models and their own data set in **Figures 25 and 26**. To find out if this is a fault of the maching learning process, or the accuracy of the Emotiv EPOC+, a super model was trained by combining both data sets shown in **Figure 36**, with it's adjusted confusion matrix in **Figure 37**. The adjusted Confusion matrix shows that the Combined matrix will still perform relitivey well, despite using both models. The conclusion from this is that the machine learning algorithm will still predict well, but the accuracy of the Emotiv EPOC+ will change over time, even though it is very percise.



Confusion Matrix for: k-Nearest Neighbor

True class \ Predicted class	None	B	Y	RIGHT	LEFT	TPR / FNR
None	150 44.9%	49 14.7%	42 12.6%	50 15.0%	43 12.9%	44.9% 55.1%
B	53 39.6%	42 31.3%	18 13.4%	12 9.0%	9 6.7%	31.3% 68.7%
Y	48 47.5%	9 8.9%	21 20.8%	15 14.9%	8 7.9%	20.8% 79.2%
RIGHT	47 33.1%	8 5.6%	12 8.5%	71 50.0%	4 2.8%	50.0% 50.0%
LEFT	35 36.5%	7 7.3%	8 8.3%	1 1.0%	45 46.9%	46.9% 53.1%

Figure 36: Confusion Matrix of model trained using both data set 1 and 2

Adjusted Confusion Matrix for: K-Nearest Neighbor

True Class		B	Y	RIGHT	LEFT		TPR/ FNR
	B	42 51.9%	18 22.2%	12 14.8%	9 11.1%		51.9% 48.1%
	Y	9 9.0%	21 21.0%	15 15.0%	8 4.0%		39.6% 60.4%
	RIGHT	8 8.4%	12 12.6%	71 74.7%	4 4.2%		74.7% 25.3%
	LEFT	7 11.5%	8 13.1%	1 1.6%	45 73.8%		73.8% 26.2%

Predicted Class

Figure 37: Adjusted Confusion Matrix of model trained using both data set 1 and 2

Chapter 5: Conclusion

5.1 Summary

This thesis presented a brain-computer interface model based off of EEG and machine learning that was able to take EEG data, build a model though machine learning, and control a video game. Although not one hundred percent successful due to some hardware problems, the BCI system show potential.

This thesis also reviewed previous work in BCI in video games and presented areas of improvement. Such improvements are: shorter training period, more flexible Machine learning tools, larger options of video games genre, and capability of playing games not designed for BCI.

The methodology to successfully complete the BCI model was done in three systems. System one, data acquisition. During data acquisition, EEG signals, the independent data, and controller buttons, the dependent data, are sampled every quarter second or every 32 samples and saved to a text file to be used in the next system, machine learn. The controller data is also sent to a virtual controller, vJoy, to interact with the videogame. This is done to give the user feedback on his actions.

Machine learn all takes place in the MATLAB environment. Features are extracted from each EEG channel. These features are the five unique brain waves, and the averages and standard deviation of each. These features are then used to teach the learner which is a classical learner built into MATLAB. After selecting the best learner from the ones available after training, this model can be exported for the next system, machine predict.

In machine predict, the EEG data is captured the same way it was in data acquisition, but this time, the data is sent to the MATLAB engine instead of a text file. Controller data is not used for machine predict, but is instead used as a mean of validating the predicted values. After the EEG signals are sent to the MATLAB engine, a scrip is running to extract features and fed into the model to get a predicted button output. This output is then fed into vJoy to interface with the videogame. Afterwards the physical controller data is compared to the predicted data to ensure that the predicted data is valid.

5.2 Future work

Although not perfect, with careful observation and further test the following are suggestions for improvements to future BCI systems of similar nature. A problem in this experiment was the Emotiv EPOC+ EEG hardware. Although the tool is very precise, it is difficult to maintain accuracy. The data is precise enough that it can teach a machine learning algorithm, but it is not accurate enough to produce a similar model each iteration. This inaccuracy may be due to the nature of wearable EEG. Being portable, and easy to use, also has the side effect of having a lot of leeway. When putting on the EPOC+, it is nearly impossible to put it on the same way every time. They are many factors that reduce the EPOC+'s accuracy:

- The physical placement is different every time it is worn

- The volume of conductive saline solution varies spatially and temporally each time to get enough conductance
- Human factors that might affect the conductance such has hair, or sweat
- Human factors that might affect the EEG readings, such has heartbeat, and eye movement
- Environmental factors such as temperature, or humidity

For future BCI systems it is recommended that these factors are taken into stricter considerations in order to produce more accurate models. Once accurate results can be produce, the next step would be to produce a more robust machine learning model that will be able to produce accurate results with the factors that may have reduce this models accuracy. A library of sub-models all in different human and environmental conditions, with a master model to arbitrate between which sub-models to select based on the environment.

Another improvement that might be made is smarter feature extraction. This researches uses the full set of mean and standard deviation, of each brain wave, for each sensor, in its feature extraction. Smarter feature extractions such as putting a weight to each node in a particular areas of the brain for different applications, since functional parts of the brain can be sectionalizing.

The inherent limits of EEG also hamper the potential of this type of BCI system. EEG only measures the surface of the brain. They are many other noninvasive technologies out there that may further improve this type of BCI systems such as functional magnetic resonance imaging.

This research has showed how to create a BCI system with EEG recording devices, machine learning, and video games. This technology is still very young, and requires further research on machine learning, feature extraction, and noninvasive BCIs that are precise, accurate, informative, and portable. It is the hope of this thesis that it furthers that research.

References

[1] W. L. Hosch. (2016). “*Machine Learning,”* Britannica [Online]. Available: https://www.britannica.com/technology/machine-learning [Accessed: June 6, 2016]

[2] R. Kohavi, F. Provost (1998) “*Glossary of Terms: Special Issue on Applications of Machine Learning and the Knowledge Discovery Process,”* robotics.stanford.edu *[Online].* Available: http://robotics.stanford.edu/~ronnyk/glossary.html [Accessed: June 6, 2016]

[3] S. Russell, P Norvig, “Learning from Observations,” Artificial Intelligence, A Modern Approach, 1st ed. Upper Saddle River, Prentice Hall, 2013, ch. 18, sec 1 pp. 525 - 529

[4] MATLAB. *“Machine Learning with MATLAB,”* Matworks [Online] https://www.mathworks.com/solutions/machine-learning/?requestedDomain=www.mathworks.com#classification [Accessed: June 6, 2016]

[5] L. F. Hass. *“Neurological Stamp,”* National Center for Biotechnology Information. [Online] Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1738204/pdf/v074p00009.pdf [Accessed: June 6, 2016]

[6] Editors of Encyclopedia Britannica. *“Electroencephalography,”* Britannica [Online]. Available: https://www.britannica.com/science/electroencephalography [Accessed: June 6, 2016]

[7] M. R. Nuwer, et al. *“IFCN guidelines for topographic and frequency analysis of EEGs and EPs,”* Electroencephalography and Clinical Neurophysiology/Electromyography and Motor Contro, Vol. 91, no 1, pp. 1-5, July 1994. [Online] Available: http://www.clinph-journal.com/pb/assets/raw/Health%20Advance/journals/clinph/Chapter1-4.pdf]

[8] Serendip. *“Brain Structures and their Functions,”*serendip.brynmawr.edu. [Online]. Available: http://serendip.brynmawr.edu/bb/kinser/Structure1.html [Accessed: June 6, 2016]

[9] F.W. Sharbrough, et al. *“American Electroencephalographic Society Guidelines for Standard Electrode Position Nomenclature,”* Journal of Clinical Neurophysiology, Vol. 8, no. 2, pp. 200-202, December 1990. [Online]. Available: https://www.researchgate.net/publication/203918314_Guideline_thirteen_Guidelines_for_standard_electrode_position_nomenclature_American_Electroencephalographic_Society_J_Clin_Neurophysiol_1994_11_111_113_8195414 [Accessed: June 6, 2016]

[10] Emotiv. *"EPOC+,"* Emotive.com. [Online]. Available: http://emotiv.com/epoc/ [Accessed: June 6, 2016]

[11] A Vallabhaneni, T. Wang, B. He. *"Brain-Computer Interface,"* In *Bioelectric Engineering*, NY: Kluwer Academic / Plenum Publishers, 2005, PP. 85 – 118.

[12] Andrei Sherstyuk, et al. *"Toward Natural Selection in Virtual Reality,"* IEEE Computer Graphics and Applications, vol.30, no. 2, pp. 93-96,C3, March/April 2010, doi:10.1109/MCG.2010.34

[13] Oculus. *"The Rift's Reccommended Spec, PC SDK 0.6 Released, and Mobile VR Jam Voting,"* Oculus [Online]. Available: https://www.oculus.com/en-us/blog/the-rifts-recommended-spec-pc-sdk-0-6-released-and-mobile-vr-jam-voting/ [Accessed: June 19, 2016]

[14] J. Gonzales-Sanchez et al. *"ABE: An Agent-Based Software Architecture for a Multimodal Emotion Recognition Framework,"* in *9th Working IEEE/IFIP Conference on Software Architecture,* CO, 2011, pp. 187-193

[15] M. Maozzami, *"EEG Signal Processing in Brain Computer Interface,"* M.S. Thesis, Dept. Comp. Sci., Michigan State Univ., 2012.

[16] M. Vliet et al. *"Designing a brain-Computer Interface controlled video-game using consumer grade EEG hardware,"* in *ISSNIP Biosignals and Biorobotics Conference: Biosignals and Robotic for Better and Safer Living (BRC)*, Manaus, 2012, pp. 1-6

[17] Brainworks. *"What are brainwaves,"* Brainworks [Online]. Available: http://www.brainworksneurotherapy.com/what-are-brainwaves [Accessed: June 19, 2016]

[18] MATLAB. *"The Language of Technical Computing,"* Matworks [Online] https://www.mathworks.com/products/matlab/ [Accessed: June 19, 2016]

[19] Microsoft. *"Direct Input,"* Microsoft [Online] https://msdn.microsoft.com/en-us/library/windows/desktop/ee416842(v=vs.85).aspx [Accessed: June 19, 2016]

[20] vJoy. *"VJOY,"* vJoy [Online] http://vjoystick.sourceforge.net/site/ [Accessed: June 19, 2016]

VITA

Cherwa Vang

Candidate for the Degree of

Master of Science

Thesis: EEG AND MACHINE LEARNING IN BRAIN-COMPUTER INTERFACE

Major Field: Electrical Engineering

Biographical:

Education:

Completed the requirements for the Master of Science/Arts in your major at Oklahoma State University, Stillwater, Oklahoma in July, 2016.

Completed the requirements for the Bachelor of Science/Arts in your major at Oklahoma State University, Stillwater, Oklahoma 2014.