Final Year Project Titled

"P300 SPELLER SYSTEM FOR BRAIN COMPUTER INTERFACE"

Submitted in Partial Fulfilment of the Requirement of the Degree of Bachelor of Technology

By

PRAVEENKUMAR SUTHAR (151070018)

PRAFULL PARMAR (151070025)

YASH BARAPATRE (151070030)

ADITYA SAMANT (151070038)

(GC-18)

Bachelor of Technology (Computer Engineering)

2018-19

Under the Guidance of

Ms. S. S. Suratkar

DEPARTMENT OF COMPUTER ENGINEERING & INFORMATION TECHNOLOGY

VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE

MUMBAI-400019

2018-2019

STATEMENT OF CANDIDATES

We state that work embodied in this Project entitled "P300 SPELLER SYSTEM FOR BRAIN COMPUTER INTERFACE" form our own contribution of work under guidance of Ms. S. S. Suratkar at the Department of Computer Engineering and Information Technology, Veermata Jijabai Technological Institute Mumbai. The report reflects the work done during the period of candidature but may include related preliminary material provided that it has not contributed to an award of previous degree. No part of this work has been used by us for the requirement of another degree except where explicitly stated in the body of the text and the attached statement.

PRAVEENKUMAR SUTHAR

(151070018)

PRAFULL PARMAR

(151070025)

YASH BARAPATRE

(151070030)

ADITYA SAMANT

(151070038)

Date: _____

APPROVAL SHEET

The dissertation entitled “P300 SPELLER SYSTEM FOR BRAIN COMPUTER_INTERFACE” by GC-16 is found to be satisfactory and is approved for the Degree of Bachelor of Technology.

Ms. S. S. Suratkar _____

Project Mentor **Examiner**

Date: _____

CERTIFICATE

This is to certify that work entered is the work of Final Year Project Phase II of Computer Engineering Branch, Academic Year 2018-19 and Bachelor of Technology Program, Group GC-16 has satisfactorily completed the required assignment for the VIII semester of the year 2018-19 as laid down by the Institute.

_____ _____

Ms. S. S. Suratkar **Examiner**

Project Mentor

_____

Dr. M. M. Chandane

College Seal/Stamp

Place: Veermata Jijabai Technological Institute, Mumbai.

Date: / /2019

ACKNOWLEDGEMENT

We would like to thank all those people whose support and cooperation has been an invaluable asset during the course of this Project. We would also like to thank our Guide Ms. S. S. Suratkar for guiding us throughout this project and giving it the present shape. It would have been impossible to complete the project without their support, valuable suggestions, encouragement and guidance.

We convey our gratitude also to Dr. M. M. Chandane, Head of Department for his motivation and providing various facilities, which helped us greatly in the whole process of the project.

We are also grateful for all other teaching and non-teaching staff members of the Computer Technology for directly or indirectly helping us for the completion of this project and the resources provided.

We acknowledge the support of Prof. Rahul Ingle for providing us immense support and platform for our project.

PRAVEENKUMAR SUTHAR **(151070018)**

PRAFULL PARMAR **(151070025)**

YASH BARAPATRE **(151070030)**

ADITYA SAMANT **(151070038)**

Date: _____

Abstract

The P300 Speller System for Brain Computer Interface has an objective of effectively communicating with patients having motor neuron diseases. The system makes use of the electroencephalogram (EEG) signals recorded from the scalp, as the patient responds to some visual stimuli. A rare visual event leads to an event-related potential (ERP), which is detected in the electroencephalogram after approximately 300 ms as per the oddball paradigm. The recorded data is then used to train a classifier for predicting the sequence of characters that the patient wants to convey. Such a system would act as a potent manner of expression for the suffering patients.

Contents

List of Figures

List of Abbreviations

Amyotrophic lateral sclerosis (ALS)

Artificial Neural Network (ANN)

Brain activity pattern (BAP)

Brain computer interface (BCI)

Electrocardiography (ECG)

Electrocorticography (ECoG)

Electroencephalography (EEG)

Electromyography (EMG)

Magnetoencephalography (MEG)

Near infrared spectroscopy (NIRS)

Superconducting quantum interface device (SQUID)

Support Vector Machine (SVM)

Visual Evoked Potentials (VEPs)

Functional magnetic resonance imaging (fMRI)

Chapter 1

Synopsis

1.1 Problem Statement

The aim of the research is to design an efficient P300 based speller system which, in real-time predicts a sequence of characters that the user has in their mind. This would involve directing the subject to focus on the target letter in P300 speller GUI while an Electroencephalogram machine records their brain waves. Implementation of real-time data transfer would be taken care of by the LabStreamingLayer (LSL). Hypothesis is that a Machine Learning model trained on this collected data will generate parameters on the basis of which a real-time letter prediction system would work. For this purpose, both single and ensemble variants of a Support Vector Machine classifier will be used, taking into consideration its properties of non-linearity, high accuracy and outlier handling.

1.2 Objectives

- To design a P300 speller GUI serving the functionality of character intensifications and mode selections.
- To record brain waves with the help of an EEG machine (Enobio 3G) to capture impulses generated in the occipital lobe due to visual disturbances.
- To establish a real-time connection between P300 speller GUI implemented in python code and NIC software using the LabStreamingLayer (LSL) implemented using the python library - pylsl.
- To send markers from python program to the NIC software and simultaneously send the data recorded by EEG machine to the python script.
- To train a Machine Learning model from the collected data and predict character sequence in real-time.

1.3 Statement of Scope

To develop a GUI based on the P300 speller with additional features such as multiple modes for training and testing and which is compatible with Enobio EEG device.
To train a machine learning model using SVMs and Neural Networks to detect VEPs in EEG signal recordings.
To predict the character which the user is focusing on in the P300 GUI grid using the aforementioned model.

1.4 Major Constraints

The Enobio EEG device used is an 8 channel device having 8 electrodes. Devices with a higher number of electrodes can give recordings from more regions of the brain and in more detail.

A GUI for the P300 speller had to be developed from scratch since the EEG device being used wasn't compatible with already existing P300 software.
The programming language being used is Python which offers fewer libraries than other platforms like Matlab for EEG signal processing.
Since direct communication between the NIC software and the python program was not possible, a separate python library LSL (Lab Streaming Layer) had to be used in order to send data to and from these two system components.

1.5 Hardware Resources Required

- Enobio 3G EEG device
- Computer with atleast 8GB RAM and Intel i5 CPU

1.6 Software Resources Required

- Neuroelectric Instrument Controller (NIC) software
- LabStreamingLayer library
- Visual Studio Code

Chapter 2

Introduction

Peripheral nerves and muscles are required to control any natural form of communication. It gets started with the user's intent. This intent triggers a process by which certain areas of the brain are activated. Signals generated by this trigger are sent through the peripheral nervous system (specifically, the motor pathways) to corresponding muscles. These muscles in turn perform the required movements which are necessary for performing communication or control task. Motor output is the result of this process. An alternative way to perform natural communication and control is by using a brain computer interface (BCI). The neuromuscular output channels i.e, the body's normal efferent pathways are bypassed by a BCI. A BCI directly measures brain activity without depending on peripheral nerves and muscles. It then transforms the recorded brain signals into corresponding control signals for different applications. This transformation requires signal processing and pattern recognition. These two works are done by a computer.

A direct correlation between the mental tasks, cognitive functionality, and the brain activities was identified many years ago by the scientific community. That sparked the curiosity about the multidisciplinary field of neuroscience, culminating with a new non-muscular channel to communicate with the external world (Cecotti et al., 2011). BCI research groups have increased over the last couple of decades, stimulated and inspired by the new advances of neurophysiology, by the advance of computer mechanism, and by the increasing awareness of the needs of disabled people.

2.1 Introduction to the Human Brain

In order to understand the background of BCI technology and to find out about the source of the P300 control signals, the fundamental principles of the human brain structure and functions are briefly introduced in this section. Humanity's insatiable curiosity has explored every part of the human body. Particularly great attention has been paid to discovering the anatomical structure of the brain and its functionality. The first experiments were performed on animals and humans with serious illnesses as indicated by Canolty et al. (2012). Over time, a substantial knowledge of brain physiology has been acquired, leading to an even greater desire for understanding the psychological and physiological operations of the human brain.

The human brain is "a dynamic, evolving information-processing system and the most complex one". One of the most interesting and lasting fields of research is the study of the human brain. The brain evolves initially from stem cells, and then grows and develops by evolving its structure and functionality to reach the adult brain state. This contrasts with the cognitive process which evolves and develops throughout life time in a continuous way to enable the brain to learn and progress.

The human brain made up of two hemispheres: right and left. The right hemisphere is in charge of the left side of the body and the left hemisphere has the responsibility for the right side of the body. Each hemisphere consists of four lobes: the frontal, temporal, parietal, and occipital as illustrated in Figure 1.1.

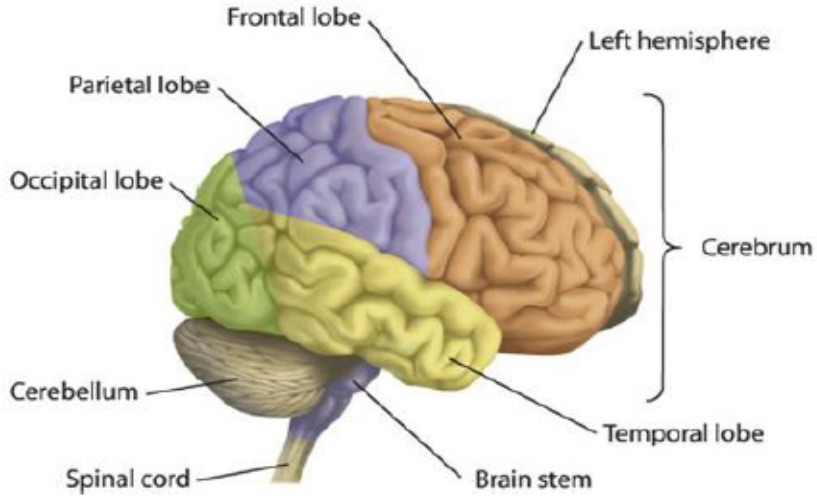


Figure 2.1: The structure of the cerebral cortex: frontal, temporal, parietal and occipital lobes

It is constructed of 100 billion neurons (Azevedo et al., 2009). As shown in Figure 1.2, the structure of single neurons involves the cell body, the axon, and the dendrites.

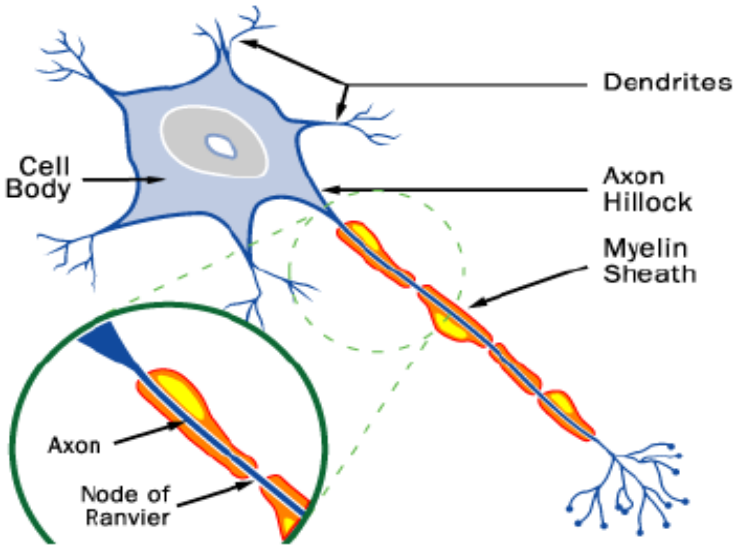


Figure 2.2: the structure of single neurons involves the cell body, the axon, and the dendrites

There are different types of neurons resulting in the emergence of functional compartments. As reported by Benuskova and Kasabov (2007), each functional system of the human brain has a different spatial region and is in charge of processing special sorts of information. The cognitive functions occur mainly in the cerebral cortex which is a thin outer layer of the human brain with thickness of 2-4 mm. With the assistance of brain imaging technologies, specifically the functional magnetic resonance imaging (fMRI), the brain functions has been precisely localized as shown in Figure 1.3. For example, the primary motor cortex is in charge of the initiation of voluntary movements. Since the P300 Speller depends on visual stimuli, this study focuses on the visual cortex which is responsible for processing visual information.

The brain activity patterns can be acquired by recording the electrical, metabolic or magnetic measurements of the neurons, forming what is called brain data.

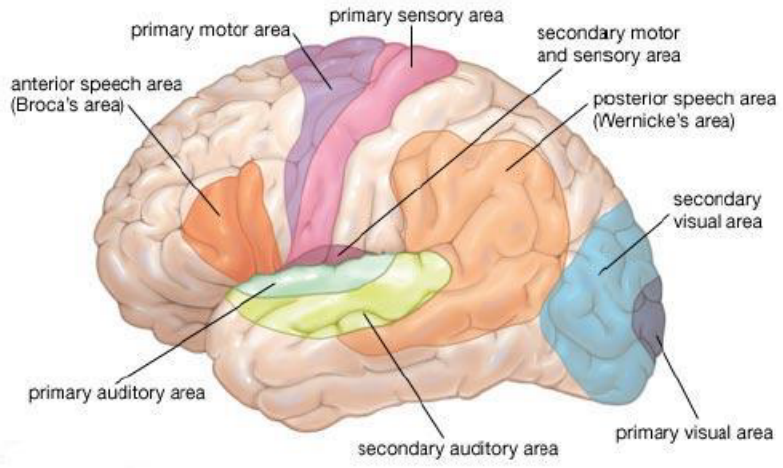


Figure 2.3: Brain functions localisation on the cortical cortex

2.2 Brain Computer Interface

Brain-computer interface (BCI) is basically a collaboration between a brain and a device. This device enables signals produced from the brain to direct some kind of an external activity, such as controlling a cursor or a prosthetic limb. This interface creates a direct communications pathway between the brain and the object which is to be controlled by the brain. For example, in cursor control, the signals are directly transmitted from the brain to the mechanism which controls the cursor. It doesn't take the normal route i.e, using our finger to control a mouse, after the neuromuscular part of our body has been directed by the brain signals to act correspondingly.

BCIs are often used for assisting, augmenting, or repairing human cognitive or sensory-motor functions. The basic working principle of any BCI system is that, it detects P300 signals and converts neurophysiologic signals into basic actions. These actions are then displayed through a computer screen. For example in the case of P300 speller the basic purpose of the BCI system is to map the P300 signals into the right character to spell. Even when we are sitting idle the brain generates lots and lots of signals. This is because the brain is doing some kind of work like controlling heart rate, respiration, etc. These signals produced by the brain in response to vital functions are called as Non-P300 signals whereas the signals produced by the brain in response to some kind of thinking or reasoning skills are known as P300 signals . So first these two kinds of signals need to be separated so that we can combine the P300 signals to generate the right character. Finally this character can be displayed on a screen.

In basic terms, a BCI depends on monitoring the brain signals, which can be done via different imaging techniques. This is done in order to detect specific distinguishable brain wave alterations which can be controlled by the user. Different waves are associated with different commands representing a new communication mechanism.

2.2.1 Applications of Brain Computer Interface

Brain Computer Interface (BCI) is one of the latest developments in the field of science and technology. This technology has a huge number of applications. Some of these are as follows:

- A BCI provides disabled people with an opportunity of communication, a chance to control environment and movement restoration

- It assists the control of devices such as wheelchairs, vehicles or assistance robots for disabled people
- It provides additional channel of control in computer games
- It can be used to monitor the attention of long distance drivers like pilots
- Intelligent relaxation devices can be developed using BCI

2.3 Electroencephalogram (EEG)

In EEG technology electrodes are placed directly on the scalp to measure weak (5-100 μV) electrical potentials generated by different activities in the brain. Electrodes are separated from the actual electrical activity because of the fluids, bone and skin of the scalp. So, the signals tend to be smoothed and are rather noisy. Hence, it is clear that EEG measurements have good temporal resolution with delays being less than tens of milliseconds but spatial resolution tends to be really poor. Spatial resolution has accuracy of about 2-3 cm at best, but usually it is worse than that. Two centimeters on the cerebral cortex can produce catastrophic results. We could infer that the user is reading when actually he is watching TV.

2.4 Event Related Potential

Small voltages that are generated in the brain structure in response to a specific event or stimuli are called as event related potential (ERPs) (Blackwood and Muir, 1990). They are basically EEG changes which are time locked to measure sensory, motor or cognitive events. Event-related potentials can be generated by varieties of sensory, cognitive or motor events. They are assumed to reflect the summed activity of postsynaptic potentials. These postsynaptic signals are produced when a huge number of similarly oriented cortical pyramidal neurons produce synchronously while processing information (Peterson et al., 1995). ERPs can be basically divided into 2 categories. One is termed as 'sensory' or 'exogenous' as the components peak roughly around 100 milliseconds after the stimulus is produced. The other is termed as 'cognitive' or 'endogenous' as they examine information processing in which the subject evaluates various stimulus.

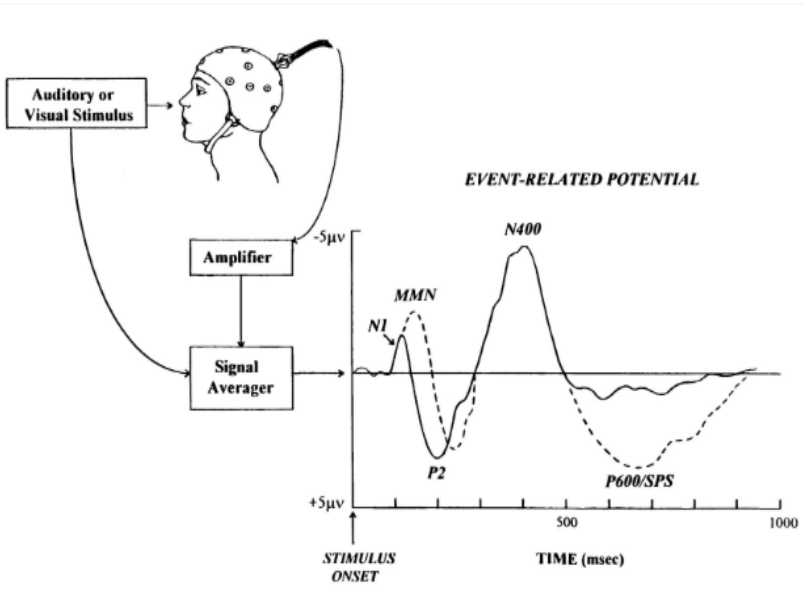


Figure 2.4: An ERP Signal

2.5 P300 Signal

Sutton et al. in 1965 discovered the P300 signal. Since then researchers in the field of ERP have analyzed it thoroughly. P300 signal is an endogenous ERP. For most adults in between the age of 20 and 70 the latency range is 250-400 ms for auditory stimuli. The latency is defined as the speed of stimulus classification that results from discrimination of one event from another. Shorter latencies indicate superior mental performance as compared to longer latencies. P300 amplitude reflects stimulus information. It indicates that greater attention produces larger P300 waves. Different paradigms have been used to generate the P300, of which the âœoddballâ paradigm is the most common. In this method different stimuli are presented in order to make one of them occur relatively infrequently. Reduced P300 amplitude is an indication of the broad neurobiological vulnerability that determines disorders within the externalizing spectrum.

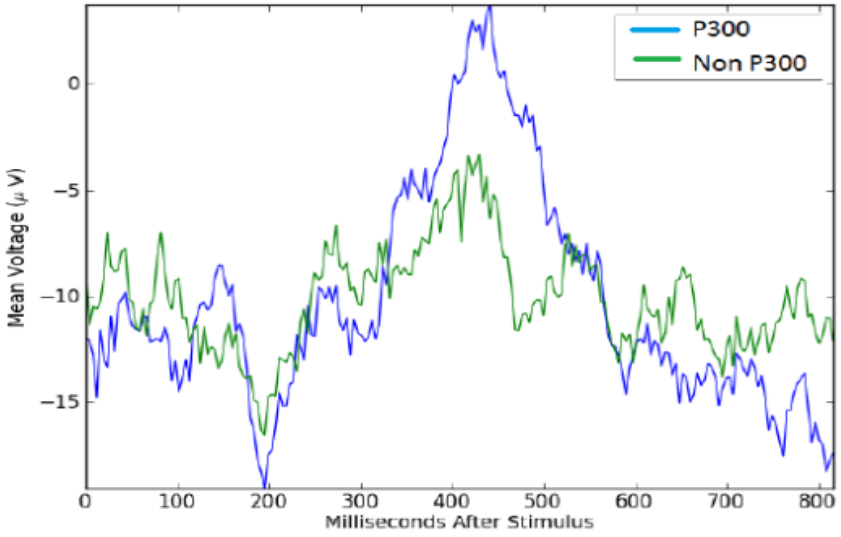


Figure 2.5: P300 Signal Vs Non P300 Signal

2.6 Motivation

One of the most significant characteristics of human beings is their capability to communicate. The richness and complexity of communication between people play an important role in relationships. However, direct conveyance of emotions, concepts and thoughts from one brain to another brain is still impossible. They have to be converted into verbal/written messages, drawings, gestures or other distinguishable expressions.

Typically, written and verbal communications are sent using the throat, mouth and hands, although the expressions are generated earlier in the human brain. However, severely disabled people are unable to use the typical output channels for communication. So they need effective tools like Brain Computer Interface(BCI) for effective communication using eye blinks, concentrating on a particular command, etc.

The number of potential users for BCIs is high and is increasing, since there are many people worldwide suffering from long-term or life-long disability. People are suffering from diseases like Amyotrophic Lateral Sclerosis(ALS) which attacks motor neurons in the brain resulting in complete and permanent paralysis.Still, these patients are fully conscious, and have needs, feelings and a deep desire to communicate with others. These factors motivates to focus on brain-computer interfaces in this study, and particularly on the P300 spelling paradigm, which has been investigated in several studies due to its importance.

2.7 Literature Survey

2.7.1 Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials

This paper describes the development and testing of a system whereby one can communicate through a computer by using the P300 component of the Event-Related Potentials (ERP). There is a display of screen containing 26 alphabets together with some extra symbols and commands which serves as the keyboard or prosthetic device. It studied 4 healthy volunteers who used the system to communicate a 5 letter word to a computer and the primary purpose was to determine the number of trials and the rate of event presentation that are required to achieve a specific level of accuracy in communication.

Four different algorithms were used to compute the scores :

1. Stepwise linear discriminant analysis (SWDA)
2. Peak Picking
3. Area
4. Covariance

The values obtained from above analyses were then used to determine the letter upon which the subject was focusing attention.With this approach the characters can be communicated with reliability at the rate of 1 character every 26 sec , or 2.3 char/min. This is, of course, a rather low rate for a communication channel. They also used data from only 5 subjects, so it didn't perform well across subjects and across sessions.

2.7.2 BCI competition III: dataset II- ensemble of SVMs for BCI P300 speller

This paper presents the algorithm that has provided the best classification performance on the dataset produced by a P300 speller matrix during the BCI 3 competition.

Within the context of P300 based BCI , several classification methods, like Support Vector Machine (SVM) and Linear Discriminant Analysis (LDA) have been used. Several BCI competitions have been organized in order to promote the development of BCI and the underlying data mining techniques and these competitions allow the community to benchmark several classification techniques in an unbiased way. Indeed,the development and test data are provided by BCI laboratories but the truth about test data are not known to competitors. This system uses P300 speller for recording brain signals and it is based on the oddball paradigm which states that rare unexpected stimuli produce a positive deflection in the EEG after about 300ms P300 speller has been introduced by Farwell Donchin which was published in 1988.

In this, to reduce the influence of signal variability in the classification problem it tries to use an ensemble of classifiers approach which helps to reduce signal to noise ratio and therefore the classifier variability. A multiple classifier system is developed for each subject and each of the single classifier is a linear support vector machine. They achieved a correct classifier performance of 73.5% and 96.5% for respectively 5 and 15 character sequences and this performance has been evaluated on a test set composed of 200 spelling characters.

Drawback of this paper is that it provides only offline analysis of the classification algorithms and online capacity has to be verified . Also BCI competition 3 has only provided datasets from 2 different subjects although from different acquisition sessions. The algorithm used is not able to handle inter-subject variability since they have only used signals from the same subject for training and testing. This issue of inter-subject learning is important in order to make this BCI speller efficient with a new patient and without the need of a training session.

2.7.3 BCI competition 2003-data set IIb: support vector machines for the P300 speller paradigm

The inference from this paper is that by training SVM for binary classification on the given training set they obtained the optimal parameter values which led to better accuracy on the given BCI dataset in the testing phase.

In this paper they propose to use SVM(Support Vector Machine) to analyse data from the P300 speller paradigm . They performed a P300 speller paradigm in which a subject is presented a 6x6 matrix, containing 36 symbols. Each row and each column is highlighted once within one trial. If the symbol, to which the subject attends, gets highlighted, a P300 component occurs in the EEG. Therefore, from 12 EEG epochs , a classifier should identify one row and one column containing a P300 to infer the attended symbol.

Consequently, from each set of six rows and six columns, there is need to find one row and one column, which is most likely to be associated with a P300. For this purpose, they trained an SVM algorithm for binary classification in a training set labeled with "1" and "-1" for P300 presence/absence, and computed the value of its discriminant function within a test set, with a high score indicating presence of a P300. To infer the symbols from the unlabeled test set, the SVM was trained on the whole training set using the optimal parameter values from five fold cross-validation.They achieved an accuracy of 84.5% for separation of P300 from non-P300 signals on training set and after five repetitions on the test set the error rates decreased from 35.5% to 0.0%. But the algorithm was trained and tested on the dataset provided by BCI laboratories and there is future scope of practical realization with an online scenario.

2.7.4 Score normalization of ensemble SVMs for brain-computer interface P300 speller

The inference from this paper is that better results were obtained when normalization techniques were used along with Ensemble Support Machine (ESVM) on EEG data.
Used the same ensemble technique[2] for reducing classifier variability. Since in multi classifier system the averaged score can be affected by one classifier as the score of different classifiers are not in the same level so they used normalization methods to normalized the scores of each of classifiers.

Different Score Normalization Techniques for ESVM are used ,i.e, :

1. Min-Max Normalization
2. Z-Score Normalization
3. Median Absolute Deviation (MAD

Used the BCI Competition 3- Dataset 2 for training and testing and also it contained data about only 2 subjects.
The proposed Min-Max normalization achieves 76% on 5 epoch and 97% on 15 epoch compared to initial method of 73.5% and 96.5% and in case of Z-score the accuracy is 75.5% and 97.5% and for MAD it is 76% and 98%.
Drawbacks are that algorithms are tested on the dataset provided by BCI and their performance is not checked on real-time data and also since the dataset contains data about only 2 subjects therefore they perform badly for inter-subject testing.

2.7.5 The BCI Competition 2003: Progress and Perspectives in Detection and Discrimination of EEG Single Trials

The inference from this paper is that by training the algorithm on the given BCI Competition III dataset II for only 5 epochs, a result similar to training the dataset on 15 epochs is obtained with

frequency of intensification of 5.7Hz.

The aim of brain computer interface (BCI) research is to establish a new augmentative communication system that translates human intentions reflected by suitable brain signals into a control signal for an output device such as a computer application or a neuroprosthesis. According to the definition put forth at the first international meeting for BCI technology in 1999, a BCI "must not depend on the brain's normal output pathways of peripheral nerves and muscles". The design of the BCI Competition 2003 encompasses the first two issues. The third issue, feedback, is largely an empirical matter that must be addressed at least in part in online experiments.

This data set, i.e., BCI Competition III Dataset 2, represents a complete record of P300 evoked potentials (three sessions from one subject) recorded with the Wadsworth BCI2000 software using the paradigm described in and originally by Farwell and Donchin.

The objective in the contest was to use the data from two sessions (i.e., 42 characters) to train a classifier and to then predict the 31 characters in the one remaining session.

The user was presented with a 6 by 6 matrix of characters. The user's task was to focus attention on characters in a word that was prescribed by the investigator (i.e., one character at a time). The six rows and six columns of this matrix were successively and randomly intensified at a rate of 5.7 Hz. Two out of 12 intensifications of rows or columns highlighted the desired character (i.e., one particular row and one particular column).

Signals were collected from one subject in three sessions and digitized at 240 Hz. Each session consisted of a number of runs. In each run, the subject focused attention on a series of characters.

They received seven submissions to this data set. Five of them predicted all 31 characters correctly, i.e., 100% accuracy. (By comparison, the accuracy expected by chance was 2.8%.) In addition, submissions needed only as few as five sequences (out of the 15 in the data file) to produce the same result.

2.8 Literature Survey Conclusion

- Since all these proposed system are based on the offline data that are provided by BCI there is little knowledge about the performance of these same algorithms on real-time EEG data.
- Also the dataset provided only consist of 2 or 5 subjects which leads to less accurate results for person not present in the training and testing phase.
- So, we aim to work on getting the performance of SVM algorithm on the real-time EEG data and provide good accuracy in the classification results of the target character.

Chapter 3

Methods Used for Brain Computer Interface

3.1 Brain Imaging Techniques

Older studies of the human brain and cognition were qualitative in nature and with a limited applicability (Posner, 1990). Despite that, cognitive science has made remarkable progress driven by by the studies that investigated both qualitative and quantitative measurements of the brain (Smith et al., 2002). It is well-known that BCIs are controlled by the quantitative measurements of the brain activity patterns (BAPs), but what do these measurements mean? Depending on the methodology used to measure these patterns, BAPs have different interpretations. There are several imaging techniques employed to detect a particular brain activity pattern. Relevant imaging technique for BCI applications and their characteristics are explained in this section.

3.1.1 Electrocorticography (ECoG) and Microarray Electrodes

ECoG is an invasive procedure to measure the brain electrical activities. Billions of neurons are embodied in the human brain. When the nerve cells are activated, small electrical signals called action potentials are generated. ECoG practice involves a surgical operation to implement a grid of biocompatible electrodes on the cortex surface as described by Wolpaw J. and Wolpaw E. (2012). Figure 3.1 shows the electrodes grid implemented on a human brain.

A similar approach uses microarray electrodes to improve the data quality by integrating analogue circuits, allowing the recording of the activity of a single neuron and reflecting a higher spatial resolution and leading to outstanding signal-to-noise ratio (Figure 2.1).

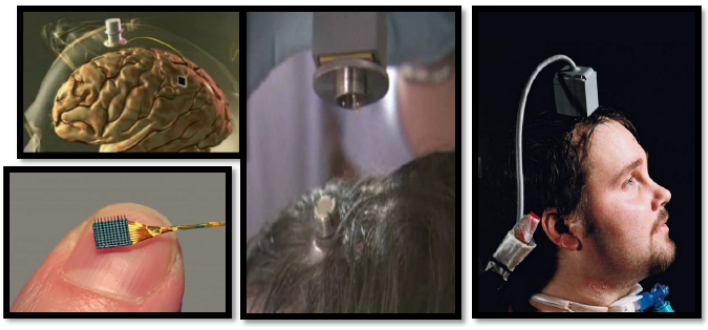


Figure 3.1: Microarray electrodes implementation in the human brain (Neurogadget, 2011).

Waves are less influenced by the conductivity of the skull and the muscular artifacts comparing to other external brain imaging techniques, which makes ECoG an appropriate strategy to be used in BCI application.

In spite of the quality of the data acquired by microarray electrodes and ECoG, there are critical handicaps as reported in Smith (2004). This includes the invasive nature of these systems, and the potential inconsistency between the neurons and the electrodes, as well as the possible infections which may result in blocking the data transmission.

3.1.2 Magnetoencephalography (MEG)

According to Smith (2004), MEG is a noninvasive scheme for measuring the brain activity. This method works by measuring the magnetic field generated by the electrical flow in the cortex. The superconducting quantum interface device (SQUID) is used in this model, as shown in Figure 2.2.

Figure 3.2: SQUI device used to collect MEG data.

Although MEG systems collect data with a remarkably high spatial resolution data (up to 3 mm), and can significantly improve the speed of BCIs, its usage in BCIs is limited to few studies, e.g. Lal et al. (2005), and Kauhanen et al. (2006). The reason behind that is the size of the SQUID instrumentation, the extremely high costs, and the non-portable style of the device. Thus, MEG is not applicable for real-world BCIs' applications as it is not practical for real-time analysis.

3.1.3 Hemodynamic Activity of the Brain: Near Infrared Spectroscopy (NIRS) and Functional Magnetic Resonance Imaging (fMRI)

Both NIRS and fMRI are used to monitor the oxygen levels of the blood passing through the brain, since the consumption of oxygen increases in active neurons. Devices used to collect the data are shown in Figure 2.3. The advantage of these technologies is the impressive spatial resolution. Actually, neurons' functionality can be distinguished from other parts of the brain and not only from the cortex as other imaging techniques. However, the acquisition equipments are large and extremely expensive. Added to that, the temporal resolution is poor (responds within a few seconds), therefore, studies such as (Weiskopf et al., 2004; Hong, Coyle, Ward, Markham McDarby, 2004), reject the usability of NIRS and FMRI after they tested them in BCIs' applications.

Figure 3.3: Devices used to collect fMRI data on the left, and NIRS data on the right (Waisman Lab, 2007).

3.1.4 Electroencephalograph (EEG)

This technique is very similar to ECoG technique, as it depends on measuring the electrical activity of the cortex using a number of electrodes. However, EEG is generally a noninvasive procedure. Instead of implementing the electrodes on the cortex surface by a surgical operation, EEG electrodes are simply placed on the patient's scalp. Despite the poor spatial resolution, EEG is the main technique used in current studies, and it has been investigated by numerous researchers. It has excellent temporal resolution of less than a millisecond. It is also relatively inexpensive and simple to acquire making it the only practical non-invasive brain imaging modality for repeated real-time brain behavioural analysis.
There are various systems available for recording EEG data. Comparing to the discussed imaging techniques, EEG acquisition systems are cheaper. They are also smaller and more portable.

It is clear that EEG imaging technique for the human brain is the best candidate to be employed in this investigation as it covers all the requirements specified earlier. A short review of the main concepts of EEG is presented below.

The existence of electrical signals in the human brain was discovered by Richard Caton, a British surgeon, in 1875. However, it was 1924 when the first EEG data was recorded by Hans Berger, a German neuropsychiatrist. He evidenced that weak electrical brain signals can be recorded and presented on a piece of paper without involving invasive surgical procedure, using his standard radio to amplify the electrical signals. When humans perform any activity such as move, smile or even think, some nerve cells are activated and generate short electrical signal called action potentials (Wolpaw et al., 2012). These potentials are transferable between cells through synapses (Figure 2.4).

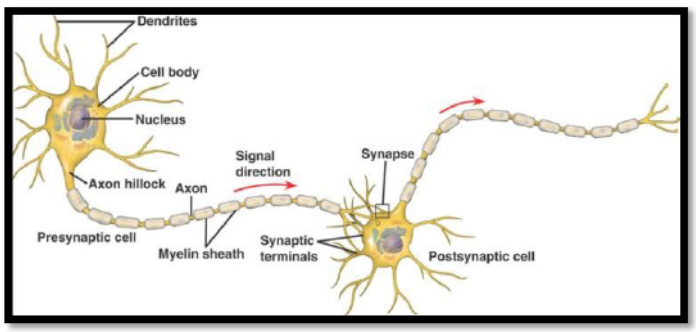


Figure 3.4: Electrical brain signals called action potentials are transferred between cells through synapse

Abhang, Rao, Gawali and Rokade (2011) define the EEG as a methodology to illustrate the electrical activity patterns of the brain's surface, the cortex, or more precisely as a way to signify the reflection of the summed synaptic potentials of the nerve cells. The frequency of these potentials measure between 1 Hz to over 30 Hz, and they are divided into six bands depending on the frequency. BCI applications utilize the band frequencies of 1-30 Hz, while potentials measures of less than 1 Hz or higher than 30 Hz are only used for limited clinical purposes (Abhang et al., 2011). Fig 2.5 presents the characteristics of these six bands including their names, frequency range, state and example.

EEG data is typically recorded by small electrodes. There are different types of electrodes available: disposable electrodes, metal cup electrodes, needle electrodes (invasive), and gelled electrodes cups. The metal and gelled electrodes are currently used for BCI applications (Stieglitz et al., 2009). Despite that, some researchers are attempting to develop user-friendly dry electrodes to minimize the preparation time required (Liao et al., 2012).

Although studies use different numbers of electrodes according to the mental tasks analysed, EEG electrodes are generally placed at particular locations on the scalp. The International Federation of Societies for EEG and clinical physiology made the first step to standardise the placement methodology allowing researchers to compare their outcomes in a better and practical way as reported by Koessler et al. (2009). The 10-20 international standard EEG placement system consists of 21 electrodes. However, it was extended over the time reaching the number of 512 electrodes for some medical application.

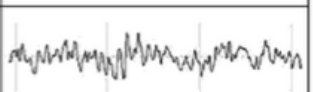
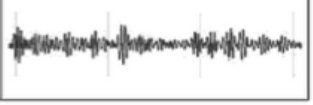
Frequency Band Name	Frequency Bandwidth	State Associated with Bandwidth	Example of Filtered Bandwidth
Raw EEG	0–45 Hz	Awake	
Delta	0.5–3.5 Hz	Deep Sleep	
Theta	4–7.5 Hz	Drowsy	
Alpha	8–12 Hz	Relaxed	
Beta	13–35 Hz	Engaged	

Figure 3.5: EEG Bands

Normally, BCI applications use a small numbers of electrodes to reduce the long preparation time and because the real-time processing of large amount of EEG data by current technologies is inadequate. Figure 2.6 shows the electrodes' positions and the channelsâŸ names in the 10-20 placement system.

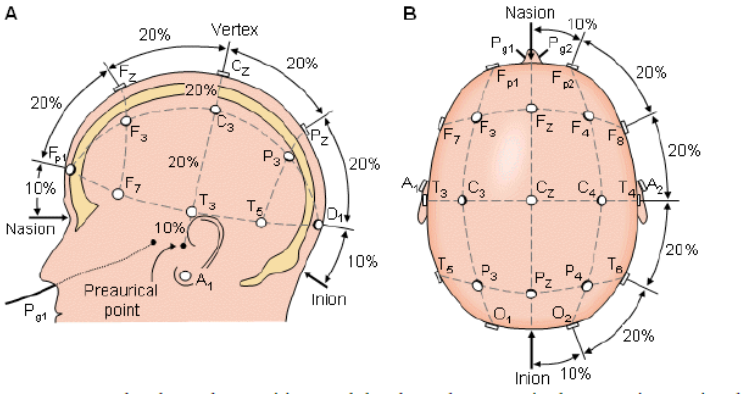


Figure 3.6: The electrodes' positions and the channels' names in the 10-20 international EEG replacement system

Noting that hand-operated placement of EEG electrode is a challenging and time consuming task, Electro Caps and EEG headsets were introduced to save time and efforts.

Chapter 4

Methods

In the P300 speller paradigm, we present a 6x6 matrix containing 36 characters. Each row and column of the matrix is highlighted in a single trial called epoch. Whenever the character that user focuses on is highlighted in a trial, a P300 spike is seen in the electroencephalogram signals. Out of the 12 intensifications in a single epoch, only two correspond to the row and column of the character that was focused by the subject. After highlighting the rows and columns for a fixed number of epochs, all the epochs are averaged out and then a specific row and column from this set of rows and columns which is mostly likely to be associated with the P300 spike is determined.

A. Overall System Flow
EEG signal data from the User (Subject) is collected by the Enobio EEG machine module which then communicates the signal data to the NIC module which stores the data. The other modules involved are the Machine Learning (ML) module and P300 Graphical User Interface (GUI) module, which are sub-modules of the P300 module. The GUI module sends the markers to the NIC module during the stimulation of the rows and columns. The NIC sends the collected EEG signal data containing markers and timestamp to the ML module. ML module analyses the data and predicts the character which is then communicated to the GUI module, which highlights the respective character on the P300 speller grid. All the modules and the relationships among them are shown in the overall system diagram shown in Fig. 1.

B. Data Collection
The data has been collected using an Enobio-8 3G device. Enobio-8 3G is a wireless and portable electrophysiology sensor system for the recording of the electrical activity of the human brain having 8 channels. Refer Fig. 2a and Fig. 2b for snapshots of the Enobio device used for collection of EEG data. The device is connected to Neuroelectrics Instrument Controller (NIC) software.

4.1 Framework of a BCI System

The fundamentals of BCIs need to be clearly understood in order to achieve the study objectives. The general framework of a BCI is presented in Figure 4.1. BCI framework comprises data acquistion, pre-processing, classification and biofeedback.

4.1.1 Data Acquisition

There are different types of data collection methods resulting in different kinds of data. The brain signals are acquired and relayed to the computer while the user is performing the appropriate mental task for the used BCI, or while paying attention to a specific stimulus. For example, suitable mental tasks used for moving a wheelchair might be imagining moving the right/left hand and the right/left foot. Other scenarios are possible as well. The user may focus his/her attention on a visual stimulus to spell a letter, which is then translated by the interface into a command.

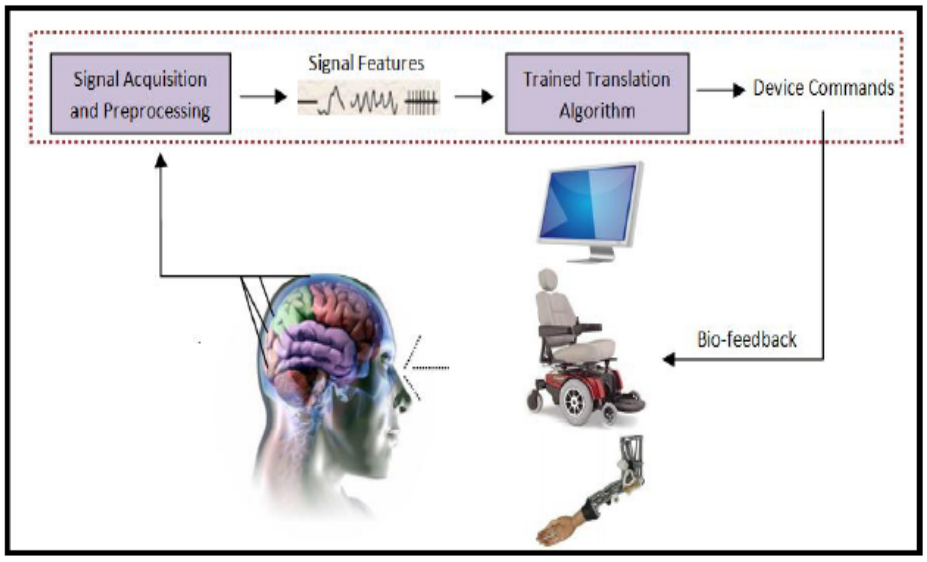


Figure 4.1: The general framework of a BCI system.

The data is modified before being transferred to the computer as shown in Figure 4.2. The signals are amplified and then passed through an analog-to-digital converter before they are transferred to the data acquisition unit and then to the acquisition software in the computer for processing. There are different methods to collect these signals from the brain.

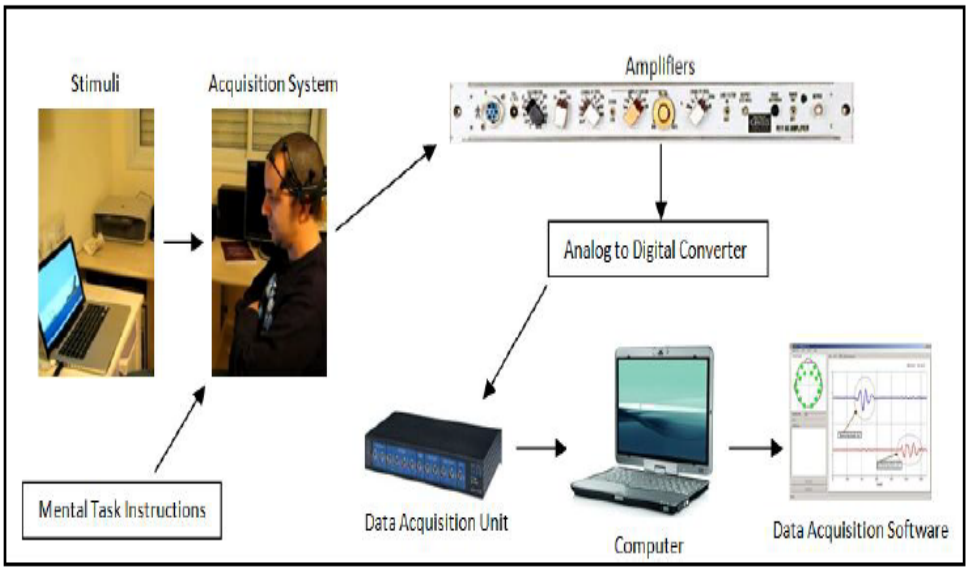


Figure 4.2: General framework for acquiring brain signals presents the steps to input the data into a BCI in the appropriate format.

4.1.2 Signal Pre-processing

Pre-processing is required for the brain data due to the fact that the acquired data could be affected by artifacts which are generated by non-cerebral origins. There are two types of artifacts: biological and environmental. Examples of the biological artifacts include: eye-induced artifacts such as eye movements, muscle-activation-induced artifacts (also referred to as electromyography (EMG) which are electrical signals recorded to detect the skeletal muscles activities), and cardiac artifacts identified as electrocardiography (ECG) which are heart's electrical signals. In contrast, environmental artifacts are generated outside the human body, and can be produced by electrode

movement, or electronic devices causing rhythmic bursts. The mentioned artifacts may produce lower and higher frequencies out of the normal signals of the human brain, resulting in poor signal-to-noise-ratio and lower classification accuracy. Thus, the brain data should be filtered to remove the undesired noise. Effective artifact removals result in significant improvements on the interface performance.

Noting that the brain signals are demonstrated in a high-density spatio-temporal format that contains a considerable amount of redundant data, temporal and spatial filters are required. Attention should be focused on the channels that are located in the top of the responsible cortex loop for the performed mental task. Moreover, temporal filters are required to locate the time frame of the intended samples of the data, noting that the brain data is measured in milliseconds. Reducing the density of the brain data in both the temporal and the spatial axes is reported to produce some remarkable effects on BCIs performance as reflected in the experimental studies.

4.1.3 Data Classification

BCIs success depends very much on classification. A classification algorithm is trained firstly using a categorized dataset or a number of datasets. These categories (class labels) are associated with mental tasks and commands for communication. For example, if the class label is 'right', the mental task performed by the user is imagine moving the right hand, and the associated command is to move a wheelchair to the right side by the interface. Principally, classification algorithms are based on analogical reasoning and similarity measurements between the characteristic/patterns of the training samples and the new samples in order to predict the intended command. Figure 4.3 represents a conceptual demonstration of a BCI classification task. Generally, the datasets involve the acquisition time of each sample and its class in addition to the brain activity measurements. The data needs to be adequate and accurate, with a good number of samples, but not excessive as this will lead to confusion rather than clarification, resulting in a low speed of processing and synthesis. After training, the classifier might be tested in order to obtain a predicted accuracy of the real-time BCI. Finally, users may start using the interface in a real-time mode.

The main classifiers used in the BCI related research involve: K-nearest neighbor, Support Vector Machine, Linear Discriminate Analysis and Neural Networks.

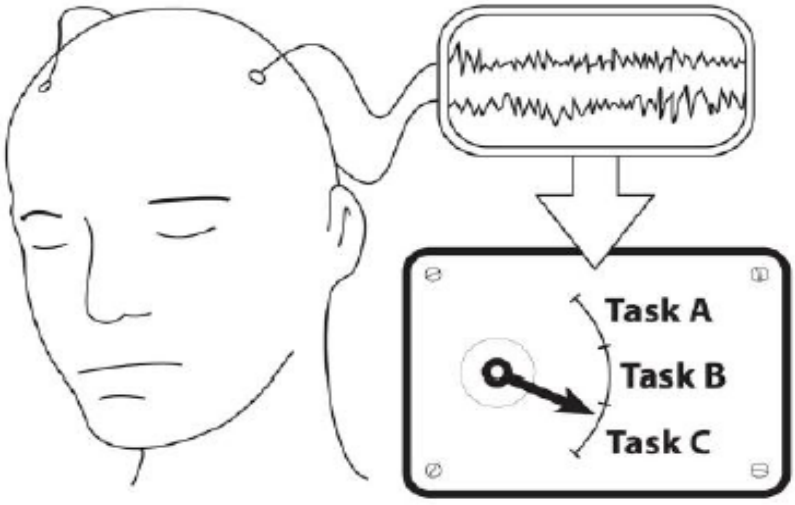


Figure 4.3: A conceptual demonstration of a BCI classification task.

4.1.4 Biofeedback

Biofeedback is the procedure in which a human obtains knowledge about his/her physiological state. This could happen repeatedly in a loop allowing the subject to monitor one physiological state or more in order to assist in a task performance.
Neurofeedback was firstly used by Hair in the early 1970s to aid in meditation and relaxation. Although some researchers are skeptical about it, others such as Trejo, Rosipal, and Matthews (2006) and Congedo, Lubar, and Joffe (2004) consider it a powerful therapeutic tool that can be used to learn self-regulation of the body systems, to stabilize mood, to normalize behaviour and to improve the mental performance. The effect of biofeedback on BCIs was tested on children (Ali-Nazari Berquin, 2010). In agreement with their hypothesis, the childrenâŸs performance was improved when feedback was provided.

Chapter 5

Software Requirements Specification

5.1 Functional Model and Description

5.1.1 System Diagram

The system diagram provides a top-level overview of the system and its components. The system components include the NIC module, Enobio EEG machine module and P300 module which is composed of the GUI and the machine learning modules. The diagram also shows how the various modules interact and collaborate with each other and with the user.

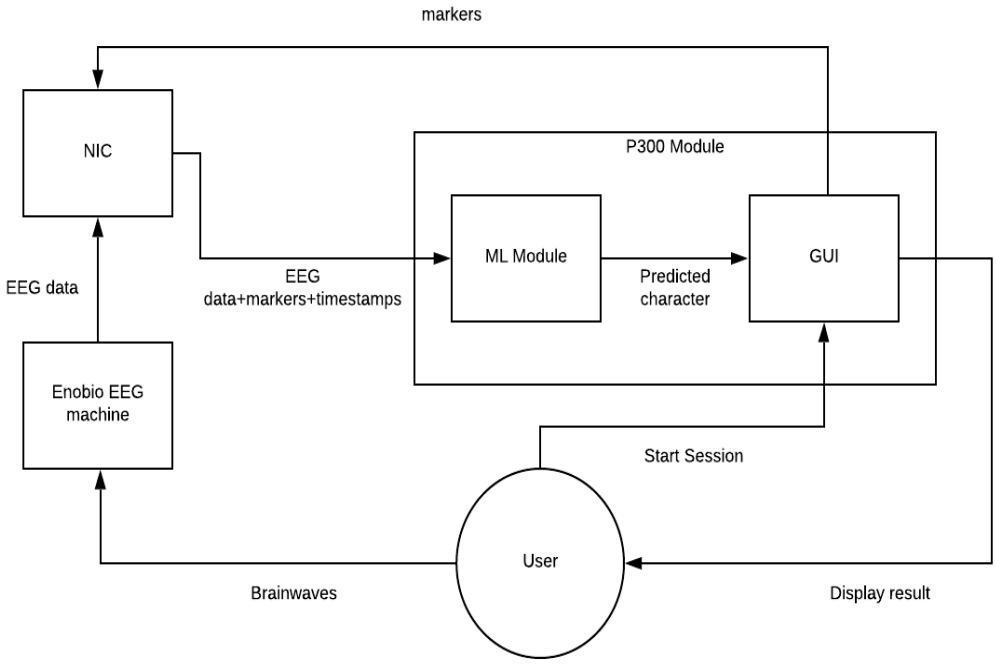


Figure 5.1: System Diagram of P300 Speller System.

5.1.2 Marker Diagram

The marker is an important field in the EEG data that contains a lot of information about the recordings such as the index when that data tuple was recorded, the target row, the target column as well as the epoch number. The marker diagram illustrates how markers are sent between the system modules in different system modes, namely training and testing modes. It explains how the marker is created, what it is composed of and the system elements that it is sent to.

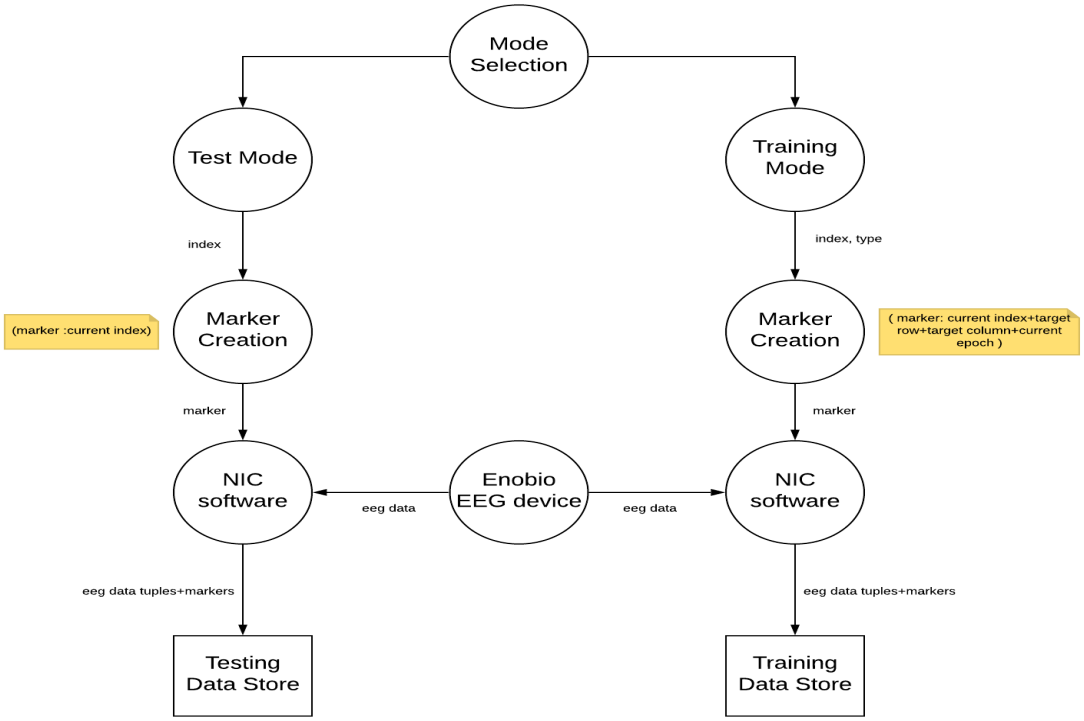


Figure 5.2: Marker Diagram of P300 Speller System.

5.1.3 Architectural Diagram

The architectural diagram explains how the user can choose the classifier type and the data to be used for training and model creation. The classifiers can be 0 (already saved classifier), 1(Neural Networks), 2(Support Vector Machines). The data for training can be from the current session, a combination of all stored data or data from all sessions for the current subject.

5.1.4 Activity Diagram

The activity diagram shows the steps through which the interaction between the subject and the speller system goes. It contains branches for the two training modes, i.e word training and subject training mode, and the testing mode. It explains what activities the subject and system engage in and the sequence of these activities.

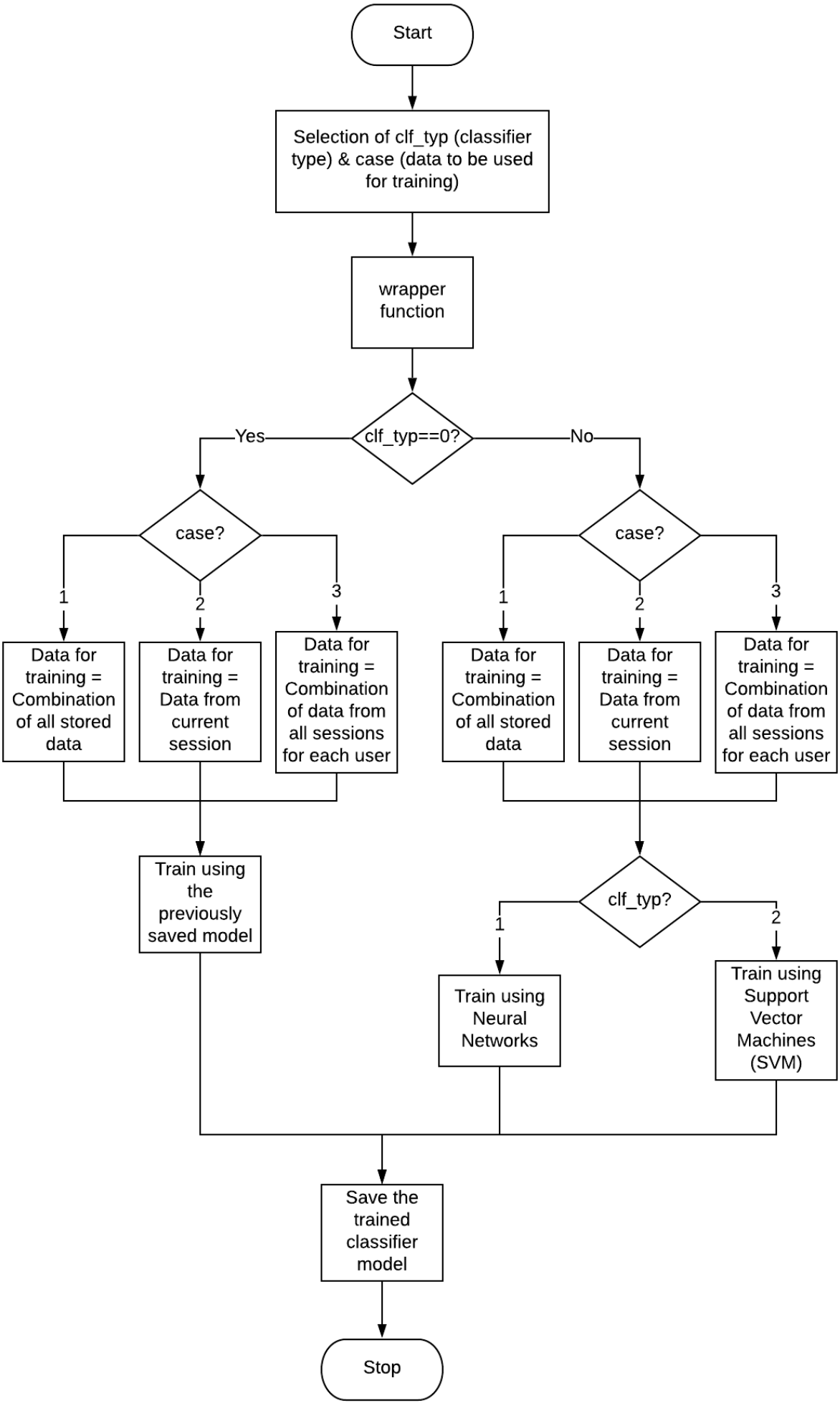
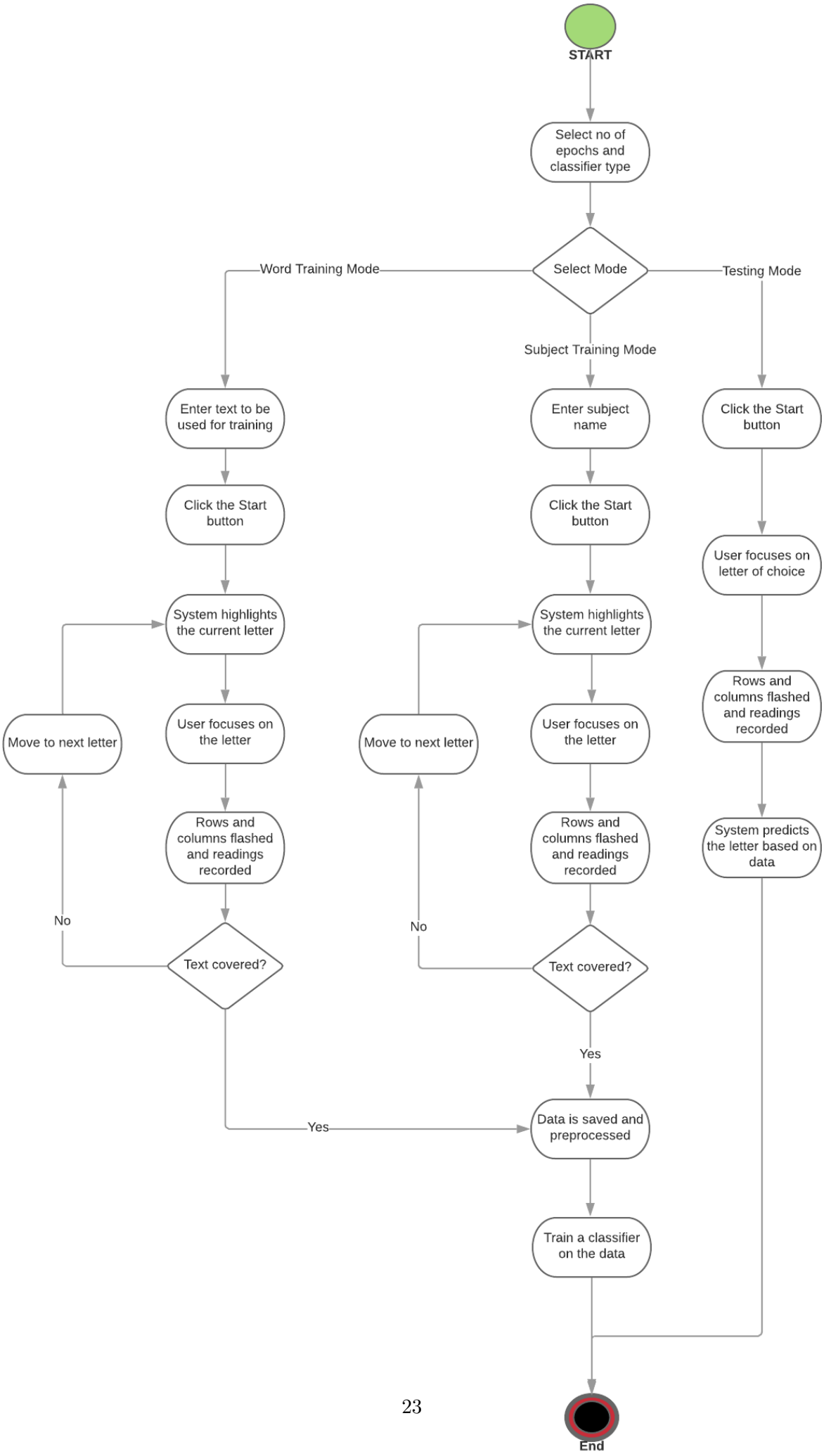


Figure 5.3: Architectural Diagram for P300 Speller System.


START
Select no of epochs and classifier type
Select Mode
Word Training Mode
Subject Training Mode
Testing Mode
Enter text to be used for training
Click the Start button
System highlights the current letter
User focuses on the letter
Rows and columns flashed and readings recorded
Text covered?
No
Move to next letter
Enter subject name
Click the Start button
System highlights the current letter
User focuses on the letter
Rows and columns flashed and readings recorded
Text covered?
No
Move to next letter
Yes
Click the Start button
User focuses on letter of choice
Rows and columns flashed and readings recorded
System predicts the letter based on data
Yes
Data is saved and preprocessed
Train a classifier on the data
End

5.2 Component Design

5.2.1 Class Diagram

The class diagram shows the classes, their relationships and cardinalities. It also describes the attributes of the classes and the functions that they can perform.

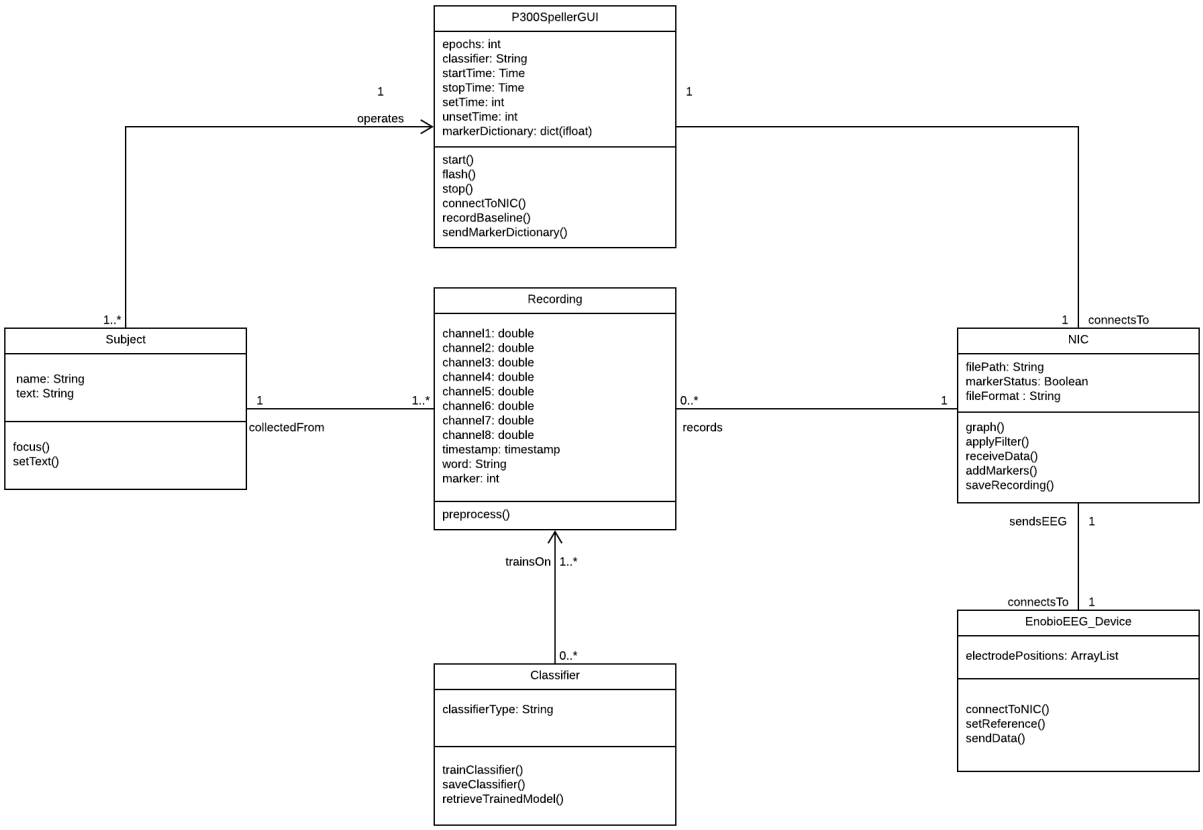


Figure 5.5: Class Diagram for P300 Speller System.

5.3 Data Description

Data generated from the EEG headset is communicated to the NIC software on the host machine via Bluetooth technology. It is collected in the form of a .easy file which stores the generated data stream. This stored data is in a tabulated format, containing eight columns of EEG signals and one column which stores the timestamp. The LabStreaming layer, which is responsible for transportation and synchronization of data, allows to send a marker field from the program running on the host machine to the NIC software. This is then appended to the nine columns. Thus, in total, ten columns of data are stored in the raw data file. For the ease of use, this raw data file in .easy format is converted into the .csv format.

The first eight columns containing the EEG data store values ranging from -4 x 10^9 to 4 x 10^9 nano electronvolts (neV). For the ease of purpose, these values are later normalized to reduce the load on the system while training the machine learning algorithm, preserving the accuracy of results at the same time. The timestamp field, which differs from the system time, stores the instants at which the data is collected, i.e., every 2 milliseconds, since its operating frequency is 500Hz. The marker column contains values ranging from 1 to 12. The significance of these values is that each of them stands for a particular row or column (six rows and six columns adding to 12) in the GUI of the P300 speller. Hence, based on the row or column being flashed at that instant, the marker field stores an integer.

Sample of raw EEG data collected using the device is shown Figure 5.6 below.

Channel#1	Channel#2	Channel#3	Channel#4	Channel#5	Channel#6	Channel#7	Channel#8	Markers	Timestamp
-109883525	-112856297	-74587216	-304887361	-110331011	-109582660	-125725803	-108782566	0	1.54401E+12
-109883968	-112859009	-74588472	-304892453	-110331363	-109584926	-125728502	-108782127	0	1.54401E+12
-109883227	-112858410	-74587781	-304893637	-110330114	-109582414	-125727448	-108779319	0	1.54401E+12
-109874103	-112850067	-74578653	-304885113	-110319239	-109572234	-125715594	-108770359	0	1.54401E+12
-109872648	-112851029	-74580232	-304889825	-110320801	-109571874	-125716854	-108771873	0	1.54401E+12

Figure 5.6: Raw EEG data file.

Chapter 6

Algorithms

6.1 Machine Learning

Machine learning is currently one of the most influential subfields of Computer Science. With applications in numerous fields, including information technologies, medicine, physics, and finances, Machine Learning has an ever growing influence on science and society.

6.1.1 Machine Learning in Medicine

Machine learning in medicine has recently made headlines. Google has developed a machine learning algorithm to help identify cancerous tumors on mammograms. Stanford is using a deep learning algorithm to identify skin cancer. A recent JAMA article reported the results of a deep machine-learning algorithm that was able to diagnose diabetic retinopathy in retinal images. It's clear that machine learning puts another arrow in the quiver of clinical decision making.

Still, machine learning lends itself to some processes better than others. Algorithms can provide immediate benefit to disciplines with processes that are reproducible or standardized. Also, those with large image datasets, such as radiology, cardiology, and pathology, are strong candidates. Machine learning can be trained to look at images, identify abnormalities, and point to areas that need attention, thus improving the accuracy of all these processes. Long term, machine learning will benefit the family practitioner or internist at the bedside. Machine learning can offer an objective opinion to improve efficiency, reliability, and accuracy.

It's been said before that the best machine learning tool in healthcare is the doctor's brain. At one point, autoworkers feared that robotics would eliminate their jobs. Similarly, there may be physicians who fear that machine learning is the beginning of a process that could render them obsolete. But it's the art of medicine that can never be replaced. Patients will always need the human touch, and the caring and compassionate relationship with the people who deliver care. Neither machine learning, nor any other future technologies in medicine, will eliminate this, but will become tools that clinicians use to improve ongoing care.

The focus should be on how to use machine learning to augment patient care. A machine learning algorithm that can review the pathology slides and assist the pathologist with a diagnosis, is valuable. If results can be generated in a fraction of the time with an identical degree of accuracy, then, ultimately, this is going to improve patient care and satisfaction.

6.2 Neural Networks

6.2.1 Introduction

An Artificial Neural Network (ANN) is a computational model that is inspired by the way biological neural networks in the human brain process information. The basic unit of computation in a neural

network is the neuron, often called a node or unit. It receives input from some other nodes, or from an external source and computes an output. Each input has an associated weight (w), which is assigned on the basis of its relative importance to other inputs. The node applies a function f (defined below) to the weighted sum of its inputs as shown in figure below:

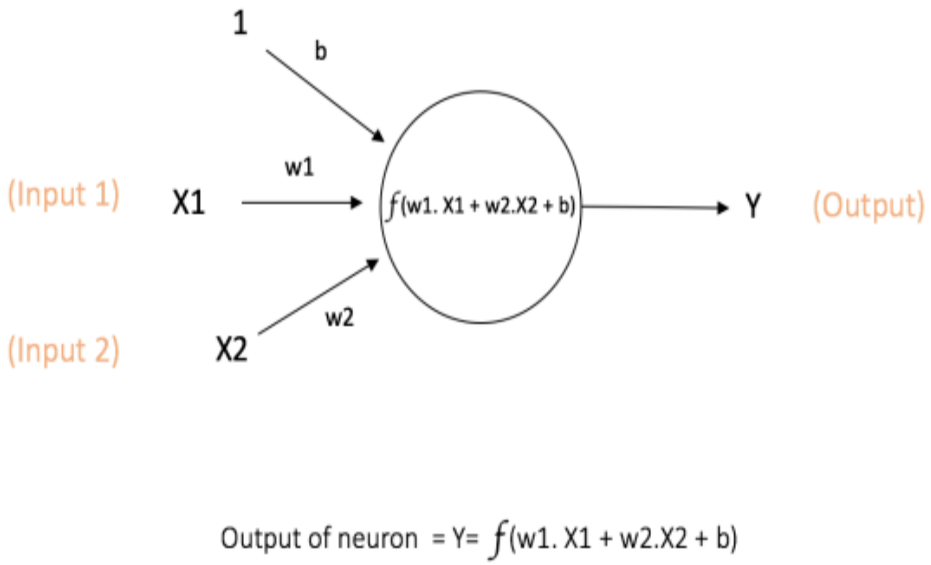


Figure 6.1: Representation of a neuron with two inputs, a bias and an activation function f.

The above network takes numerical inputs X1 and X2 and has weights w1 and w2 associated with those inputs. Additionally, there is another input 1 with weight b (called the Bias) associated with it. The main function of Bias is to provide every node with a trainable constant value (in addition to the normal inputs that the node receives). The output Y from the neuron is computed as shown in the figure. The function f is non-linear and is called the Activation Function. The purpose of the activation function is to introduce non-linearity into the output of a neuron. This is important because most real world data is non linear and we want neurons to learn these non-linear representations. Every activation function (or non-linearity) takes a single number and performs a certain fixed mathematical operation on it. Some examples of activation functions are Sigmoid, tanh and ReLu (Rectified Linear Unit).

6.2.2 Feed-forward Neural Networks

The feedforward neural network was the first and simplest type of artificial neural network devised. It contains multiple neurons (nodes) arranged in layers. Nodes from adjacent layers have connections or edges between them. All these connections have weights associated with them. A feedforward neural network can consist of three types of nodes:

1. **Input Nodes** - The Input nodes provide information from the outside world to the network and are together referred to as the "Input Layer". No computation is performed in any of the Input nodes - they just pass on the information to the hidden nodes.

2. **Hidden Nodes** - The Hidden nodes have no direct connection with the outside world (hence the name "hidden"). They perform computations and transfer information from the input nodes to the output nodes. A collection of hidden nodes forms a "Hidden Layer". While a feedforward network will only have a single input layer and a single output layer, it can have zero or multiple Hidden Layers.

3. **Output Nodes** - The Output nodes are collectively referred to as the "Output Layer" and are responsible for computations and transferring information from the network to the outside world.

In a feedforward network, the information moves in only one direction - forward - from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network (this property of feed forward networks is different from Recurrent Neural Networks in which the connections between the nodes form a cycle). Two examples of feedforward networks are given below:

1. Single Layer Perceptron - This is the simplest feedforward neural network and does not contain any hidden layer.

2. Multi Layer Perceptron- A Multi Layer Perceptron has one or more hidden layers.

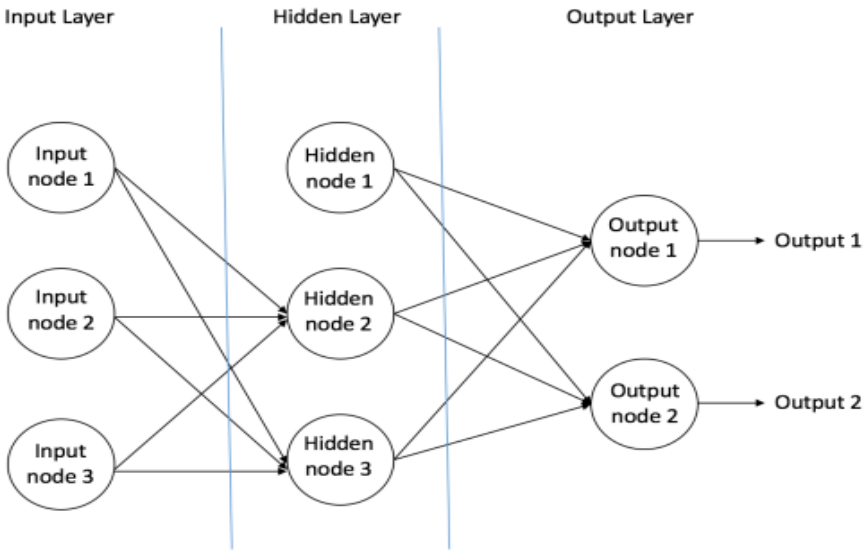


Figure 6.2: Layers of a Neural Network.

6.2.3 Multi Layer Perceptron

A Multi Layer Perceptron (MLP) contains one or more hidden layers (apart from one input and one output layer). While a single layer perceptron can only learn linear functions, a multi layer perceptron can also learn non - linear functions. Based on Figure 6.2, we have

1. **Input Layer:** The Input layer has three nodes. The Bias node has a value of 1. The other two nodes take external inputs (which are numerical values depending upon the input dataset), which we can call X1 and X2. As discussed above, no computation is performed in the Input layer, so the outputs from nodes in the Input layer are 1, X1 and X2 respectively, which are fed into the Hidden Layer.

2. **Hidden Layer:** The Hidden layer also has three nodes with the Bias node having an output of 1. The output of the other two nodes in the Hidden layer depends on the outputs from the Input layer (1, X1, X2) as well as the weights associated with the connections (edges). Remember that f refers to the activation function. These outputs are then fed to the nodes in the Output layer.

3. **Output Layer:** The Output layer has two nodes which take inputs from the Hidden layer and perform similar computations. The values calculated (Y1 and Y2) as a result of these computations act as outputs of the Multi Layer Perceptron.

Given a set of features X = (x1, x2, â) and a target y, a Multi Layer Perceptron can learn the relationship between the features and the target, for either classification or regression.

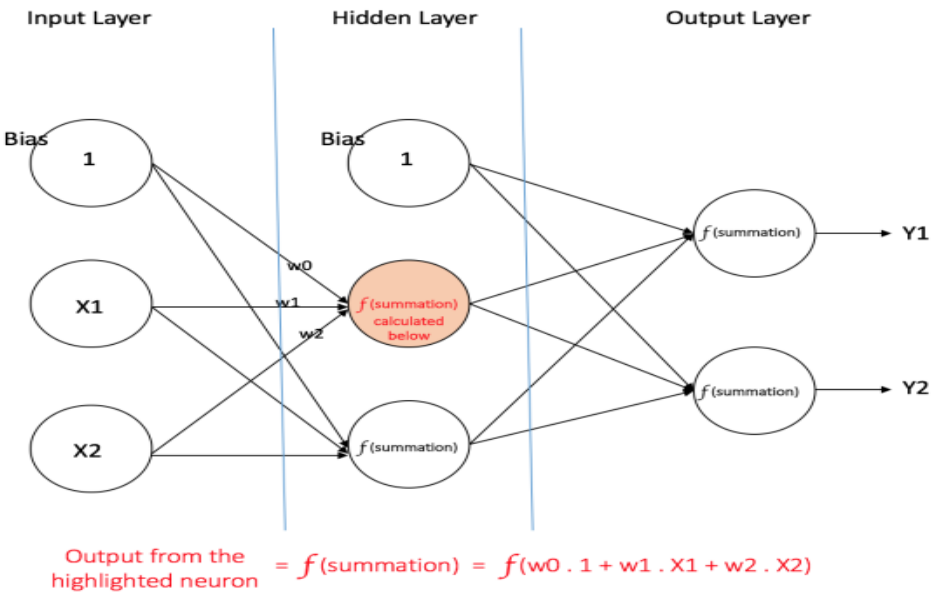


Figure 6.3: Example for calculation of output.

6.2.4 Training our MLP: The Back-Propagation Algorithm

The process by which a Multi Layer Perceptron learns is called the Backpropagation algorithm. Backpropagation, short for "backward propagation of errors", is an algorithm for supervised learning of artificial neural networks using gradient descent. Given an artificial neural network and an error function, the method calculates the gradient of the error function with respect to the neural network's weights. It is a generalization of the delta rule for perceptrons to multilayer feedforward neural networks. The "backwards" part of the name stems from the fact that calculation of the gradient proceeds backwards through the network, with the gradient of the final layer of weights being calculated first and the gradient of the first layer of weights being calculated last. Partial computations of the gradient from one layer are reused in the computation of the gradient for the previous layer. This backwards flow of the error information allows for efficient computation of the gradient at each layer versus the naive approach of calculating the gradient of each layer separately.

At the heart of backpropagation is an expression for the partial derivative dC/dw of the cost function C with respect to any weight w (or bias b) in the network. The expression tells us how quickly the cost changes when we change the weights and biases. And while the expression is somewhat complex, it also has a beauty to it, with each element having a natural, intuitive interpretation. And so backpropagation isn't just a fast algorithm for learning. It actually gives us detailed insights into how changing the weights and biases changes the overall behaviour of the network. In backpropagation, a loss function is a method of evaluating how well your algorithm models your dataset. If your predictions are totally off, your loss function will output a higher number. If they're pretty good, it'll output a lower number. As you change pieces of your algorithm to try and improve your model, your loss function will tell you if you're getting anywhere. Some examples of loss functions used are Mean squared error, likelihood loss and cross entropy loss (log loss).

An optimizer ties together the loss function and model parameters by updating the model in response to the output of the loss function. In simpler terms, optimizers shape and mold your model into its most accurate possible form by futzing with the weights. The loss function is the guide to the terrain, telling the optimizer when it's moving in the right or wrong direction. Some examples are rmsprop, adam, adagrad and sgd.

Changing our weights too fast by adding or subtracting too much (i.e. taking steps that are too large) can hinder our ability to minimize the loss function. We don't want to make a jump so large

that we skip over the optimal value for a given weight. To make sure that this doesn't happen, we use a variable called the "learning rate". This thing is just a very small number, usually something in the range of 0.001, that we multiply the gradients by to scale them. This ensures that any changes we make to our weights are pretty small. In math talk, taking steps that are too large can mean that the algorithm will never converge to an optimum.

6.2.5 Notations used

w^l_{jk} can be used to denote the weight for the connection from the k^{th} neuron in the (l-1)th layer to the j^{th} neuron in the l^{th} layer. In the example shown below, $w^3{}_{24}$ is used to denote the weight for the connection between 4^{th} neuron in the 2^{nd} layer to the 2^{nd} neuron in the 3^{rd} layer.

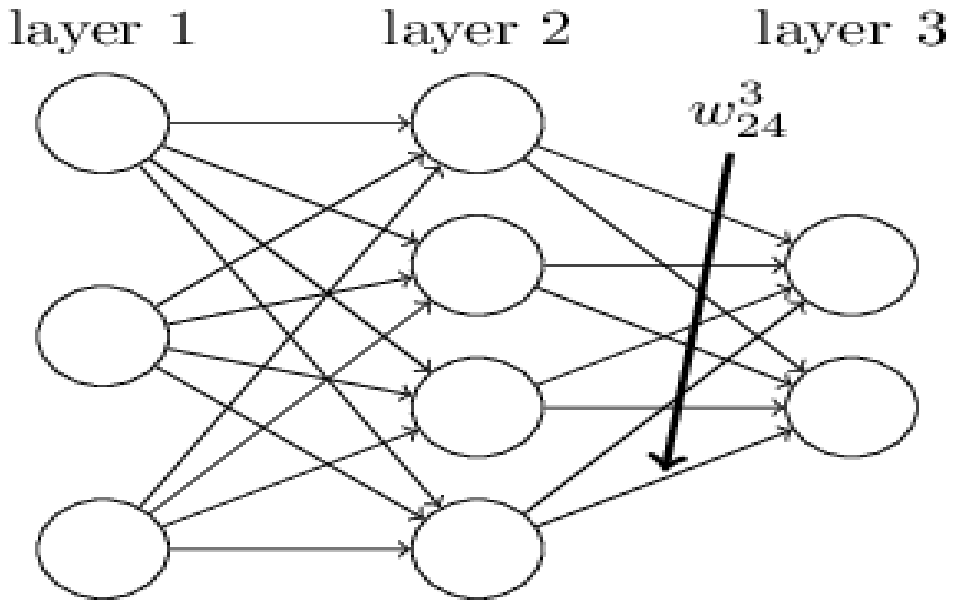


Figure 6.4: Notations in a Neural Network.

Similarly, b^l_j is used for the bias and a^l_j for the activation of the j^{th} neuron in the l^{th} layer. With these notations, the activation a^l_j of the j^{th} neuron in the l^{th} layer is related to the activations in the (l-1)th layer by the equation.

$$a^l_j = f\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right) \tag{6.1}$$

where the sum is over all neurons k in the (l-1)th layer. To rewrite this expression in a matrix form we define a weight matrix w^l for each layer, l. The entries of the weight matrix w^l are just the weights connecting to the l^{th} layer of neurons, that is, the entry in the j^{th} row and k^{th} column is w^l_{jk}. Similarly, for each layer l we define a bias vector, b^l. The components of the bias vector are just the values b^l_j, one component for each neuron in the l^{th} layer. And finally, we define an activation vector a^l whose components are the activations a^l_j. f is the vectorizing or the activation function.

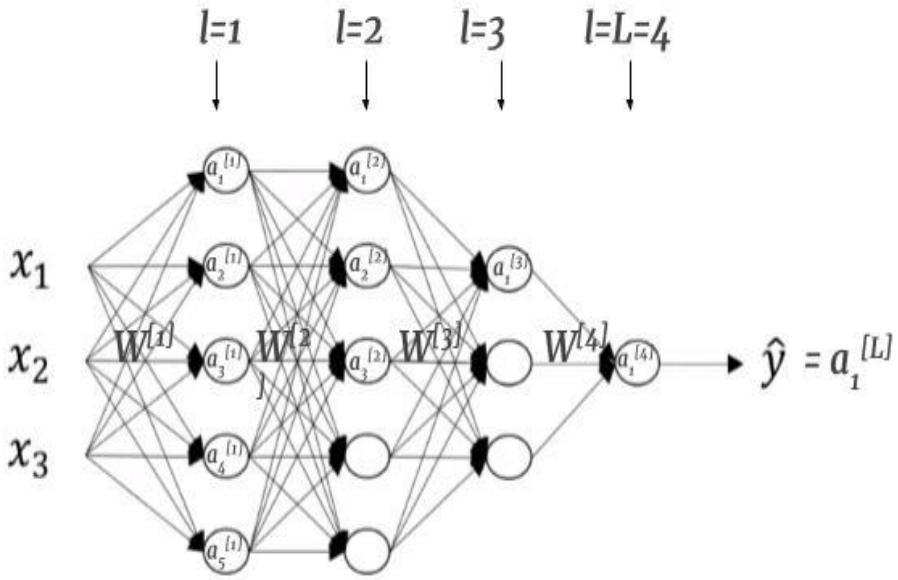


Figure 6.5: Notations in a Neural Network.

A compact version of the above equation can now be written as

$$a^l = f(w^l a^{l-1} + b^1) \tag{6.2}$$

We can use z^l to denote $w^l a^{l-1} + b^1$. So, now we have

$$a^l = f(z^l) \tag{6.3}$$

6.3 Support Vector Machine

6.3.1 Introduction

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. It is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In this algorithm, each data item as a point in n-dimensional space is plotted (where n is number of features), with the value of each feature being the value of a particular coordinate. Then, classification is performed by finding the hyper-plane that differentiate the two classes very well.

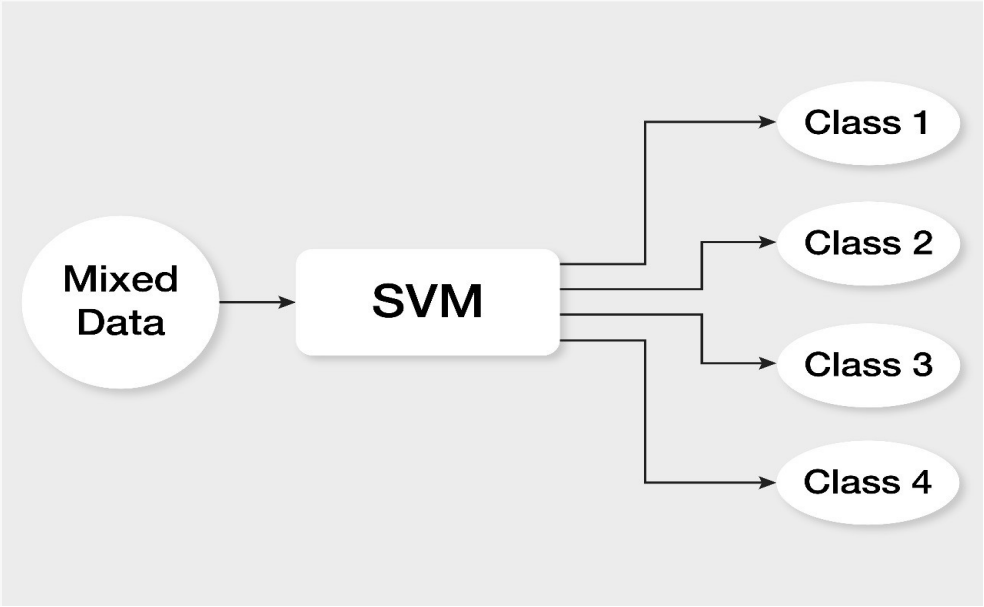


Figure 6.6: SVM is widely used for classification of mixed data.

In two dimentional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side. For example, the two images shown below are the two dimensional versions of the data before and after classification.

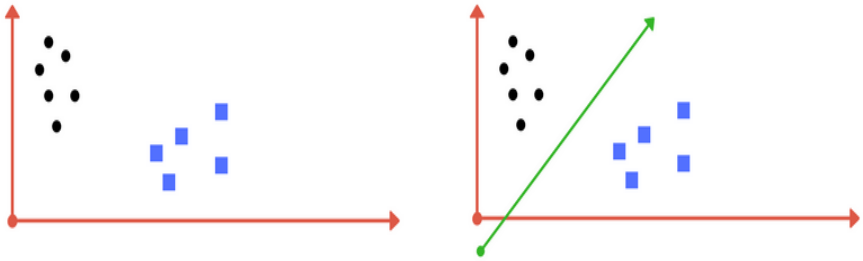


Figure 6.7: Hyperplane separates the original data such that it can be organized into classes.

Hyperplane: A hyperplane in an n-dimensional Euclidean space is a flat, n-1 dimensional subset of that space that divides the space into two disconnected parts.
SVM performs separation of classes by finding out a line/ hyper-plane (in multidimensional space that separate outs classes). It can solve linear and non-linear problems and work well for many practical problems. According to the SVM algorithm, the points closest to the line from both the classes are found. These points are called support vectors. Now, the distance between the line and the support vectors is computed. This distance is called the margin. The goal is to maximize the margin. The hyperplane for which the margin is maximum is the optimal hyperplane. Thus SVM tries to make a decision boundary in such a way that the separation between the two classes is as wide as possible.

There can be cases where there is no line that can separate the two classes in the x-y plane. In this case, a transformation is applied and one more dimension is added as called the z-axis. Assuming value of points on z plane, $w = x^2 + y^2$. In this case, it can be manipulated as distance of point from z-origin. Now if plotting is done in z-axis, a clear separation is visible and a line can be drawn.

When this line is transformed back the to original plane, it maps to circular boundary as shown in the image below. These transformations are called **kernels**.

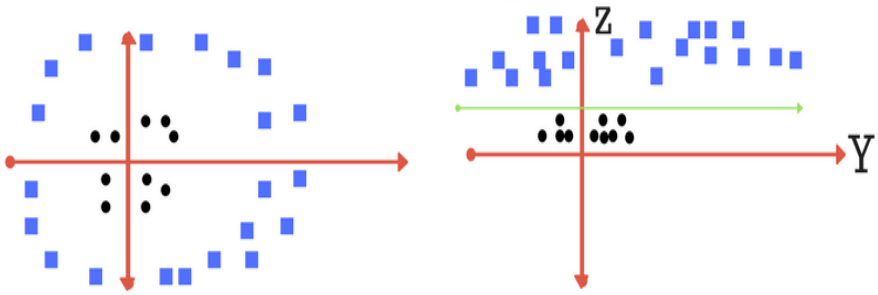


Figure 6.8: Analyzing the data in a higher dimension.

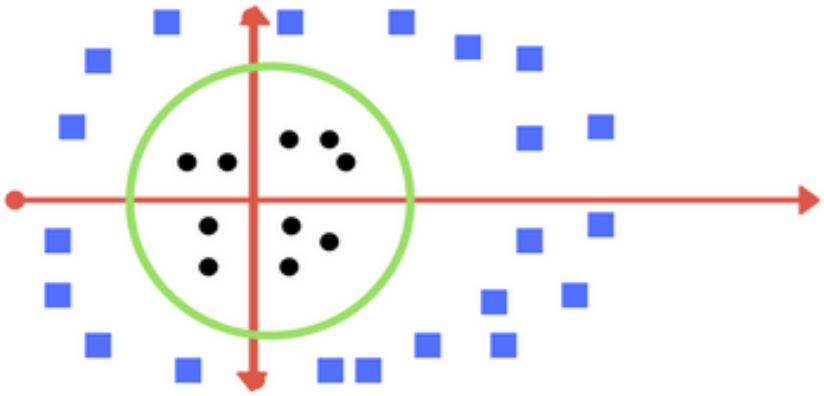


Figure 6.9: Classifying non-linear data with SVM.

For cases where the the data points of a class interfere with the space of data points of other classes, i.e., overlapping takes place, a linear classifier or an easily definable function like that of a circle, cannot separate the different classes. Separating the data space into individual parts, applying SVM algorithm individually to each of these can lead to a phenomenon called 'overfitting'. To solve this problem of classification without having to undergo overfitting, SVM has some parameters which can be tuned to achieve a high accuracy and also generalize over the data. Varying these parameters, considerable non linear classification line with more accuracy in reasonable amount of time can be achieved. Some of the important parameters that help in attaining the above mentioned goals are the regularizaton parameter, gamma parameter and the kernel parameter.

Kernel The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role. For linear kernel the equation for prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows: f(x) = B(0) + sum(ai * (x,xi)) This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients B0 and ai (for each input) must be estimated from the training data by the learning algorithm.

Regularization The Regularization parameter (often termed as C parameter in python's sklearn library) tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. It controls the trade off between smooth decision boundary and classifying training points correctly. A large value of c means you will get more training points correctly.

Gamma The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. In other words, with low gamma, points far away from plausible separation line are considered in calculation for the separation line. Even the far away points get considerable weight and we get a more linear curve.. Where as high gamma means the points close to plausible line are considered in calculation. This is because the closer points get more weight and it results in a wiggly curve as shown in previous graph.

Margin And finally last but very important characteristic of SVM classifier. SVM to core tries to achieve a good margin. A margin is a separation of line to the closest class points. A good margin is one where this separation is larger for both the classes. Images below gives to visual example of good and bad margin. A good margin allows the points to be in their respective classes without crossing to other class.

Pros and cons of the SVM algorithm:-

Pros:

1. It works really well with a clear margin of separation.
2. It is effective in high dimensional spaces.
3. It is effective in cases where number of dimensions is greater than the number of samples.
4. It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

Cons:

1. It doesn't perform well, when we have large data set because the required training time is higher.
2. It also doesn't perform very well, when the data set has more noise i.e. target classes are overlapping.
3. SVM doesn't directly provide probability estimates, these are calculated using an expensive five-fold cross-validation. It is related SVC method of Python scikit-learn library.

Chapter 7

Project Implementation

7.1 Introduction

7.2 Tools and Technologies Used

7.2.1 Enobio 8 3G

Enobio is a wearable, wireless electrophysiology sensor system for the recording of EEG. Using the Neuroelectrics Cap, Enobio 8 is ideal for out-of-the-lab applications. It comes integrated with an intuitive, powerful user interface for easy configuration, recording and visualization of 24 bit EEG data at 500 S/s, including Spectrogram and 3D visualization in real time of spectral features.The Enobio package contains all the components required to perform an EEG monitoring session, and some additional items that may be useful during your experiments. It consists of 8 electrodes.

Figure 7.1: NeuroElectrics Enobio 8 3G Headset.

7.2.2 NIC

NIC (the Neuroelectrics Instrument Controller engine) is the software that controls StarStim8/20/32 and Enobio 8/20/32 systems (the NECBOX is the little box hosting the amplifiers, electronics, battery and communications sub-systems). NIC provides access to EEG data (raw, power spectra, filters, scalp maps) as well as the configuration of stimulation session parameters and rendering of the associated electric fields.

7.3 Data Collection

The data has been collected using an Enobio-8 3G device. Enobio-8 3G is a wireless and portable electrophysiology sensor system for the recording of the electrical activity of the human brain having 8 channels. Refer Fig. 7.2 for snapshots of the Enobio device used for collection of EEG data. The device is connected to Neuroelectrics Instrument Controller (NIC) software which collects the recordings from the device and stores them in different file formats namely ".edf" and ".easy". It also stores the configuration settings during the session in the ".info" file. The recorded file contains 8 fields containing EEG data values in nano-volts, a column for marker and timestamp column for additional information about that particular record. Sample of the raw EEG data collected using the device is shown in Fig.5.6.

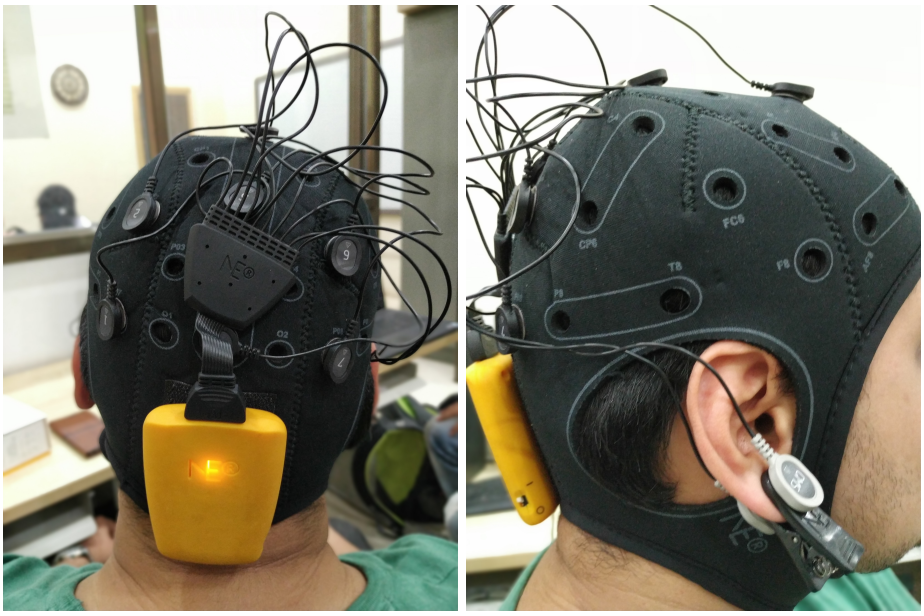


Figure 7.2: Back and Side View of Enobio EEG Device

A P300 speller system has been implemented which generates visual stimulus and has characters organized in a 6x6 grid. A snapshot of the P300 Speller is shown in Fig.7.3. The 6 rows and 6 columns are randomly intensified and de-intensified one by one for 150 ms and 100 ms respectively in a single epoch. An epoch is a sequence of intensification of the rows and columns of the P300 grid. During an epoch each row and column is intensified exactly once in random manner. For each row and column highlighted in a given epoch, we have taken the voltage or amplitude values for the 300 ms after the onset of the stimuli from the electrode positions PO7, P3, Fz, Cz, Pz, P4, PO8, and Oz. Placement of the electrodes has been done using the International 10-20 system. Selected electrode positions are shown in Fig. 7.4. After that, the collected data was band-pass filtered (5-30 Hz) and sent for preprocessing.

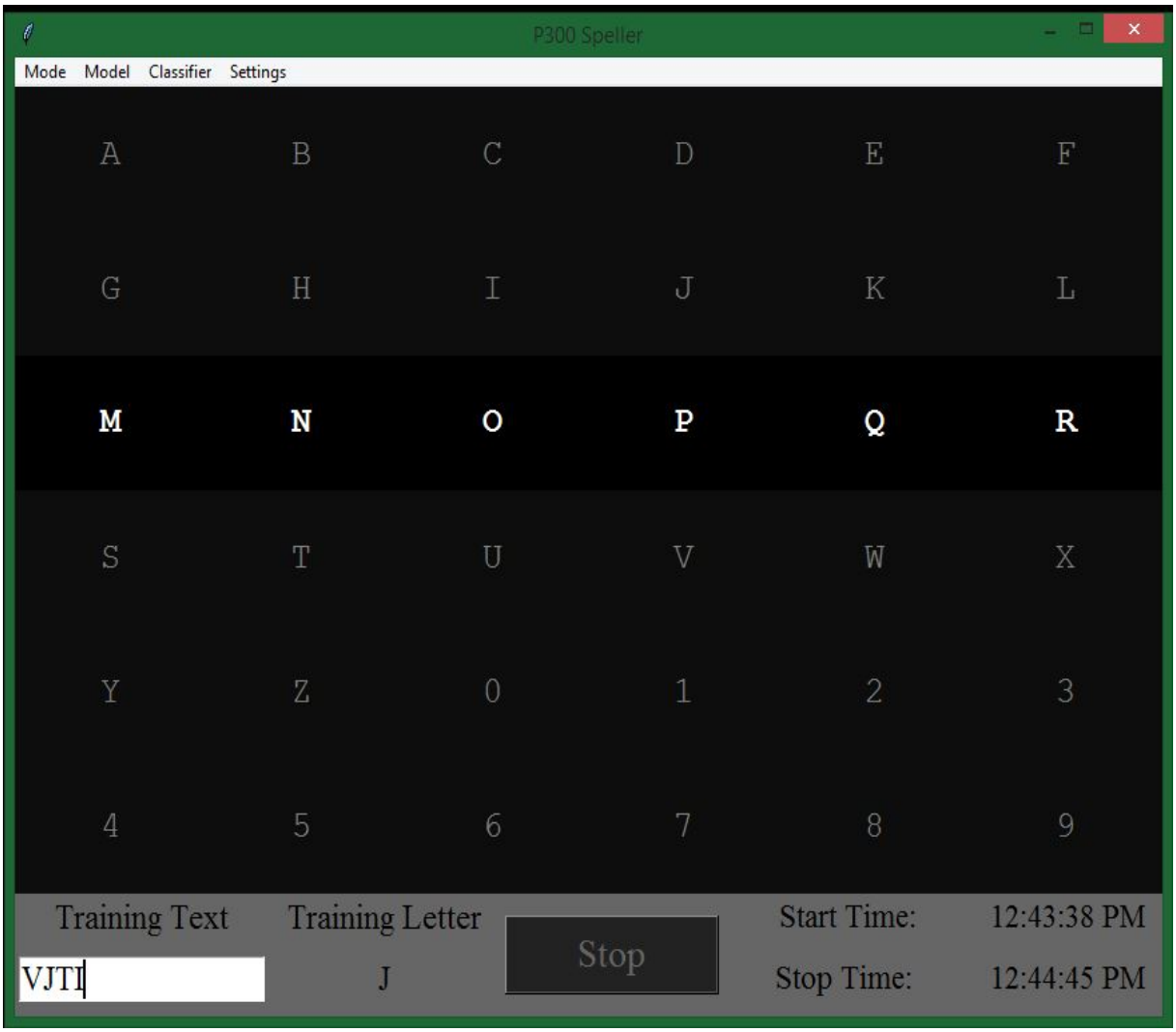


Figure 7.3: P300 Speller System.

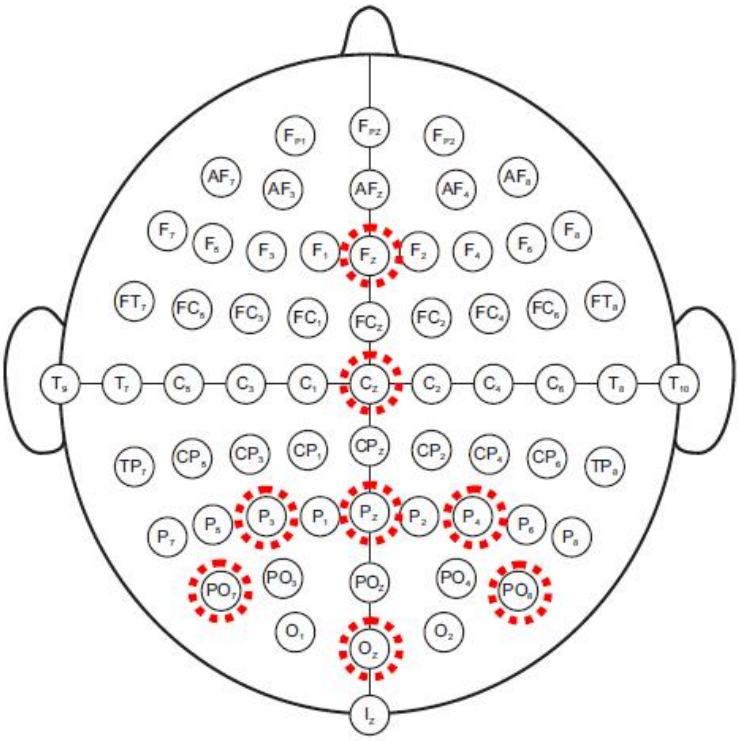


Figure 7.4: International 10-20 System.

7.4 Data Communication

The real time data collected using the Enobio device through NIC software is communicated to the P300 speller system which does the data processing. For this communication *Lab Streaming Layer (LSL)* is used.

Lab streaming layer is a system for synchronizing streaming data for live analysis or recording. LSL is an convenient and reliable way to send EEG stream to applications that can record or manipulate the data, which in our case is the P300 speller system. Marker's are sent to the NIC software using LSL, to mark the EEG recording at the start of an stimulus. Markers are the index of the row or column that was intensified at the corresponding timestamp in the P300 speller system. The rows are indexed from 1 to 6 and the columns are indexed from 7 to 12.

7.5 Data Preprocessing

Initially, the raw file which is in the ".easy" file format is read and data is labelled as positive and negative according to the markers present in the marker field. Data obtained in the range of 250-500ms after the presentation of the stimulus is marked with the marker of row or column that was intensified and presented as stimulus. Unnecessary data which does not pertain to the task is trimmed off. Then ICA is applied to remove irrelevant signal components present in the EEG signals. The EEG signal data is then re-referenced and also baseline correction is applied to the data. Baseline corrected signal data is then bandpass filtered (5-30Hz). Averaging of EEG signal data across epochs is done. An average of all the epochs for a particular character is taken where each epoch consists of 125 * 12 = 1500 records. This is done to remove noise which might be present in a specific epoch and to increase the signal-to-noise ratio, as well as to reduce the size of the data for each character. These preprocessing steps are shown in Fig. 7.5.

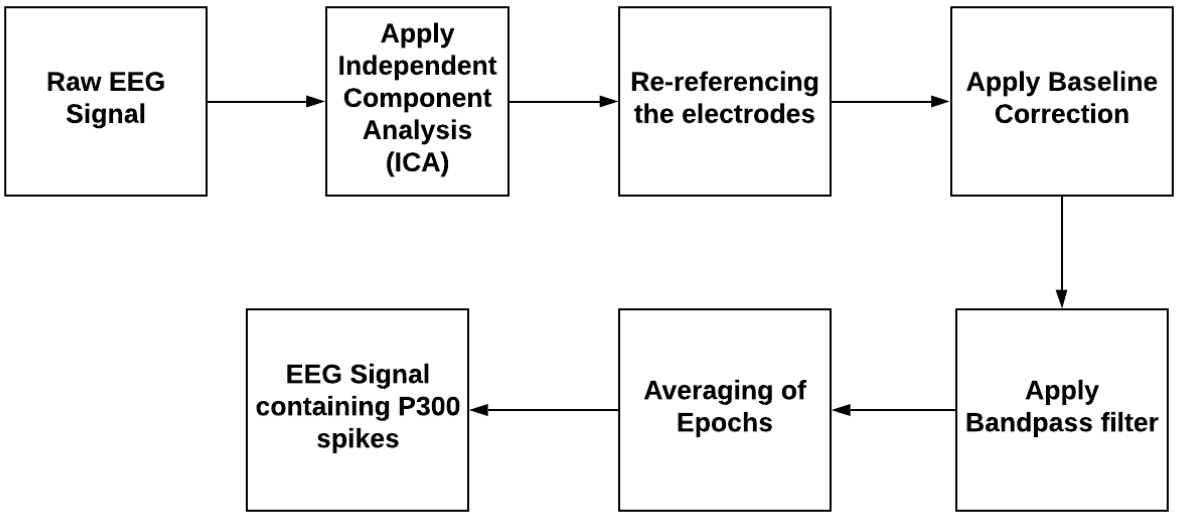


Figure 7.5: EEG data preprocessing steps.

7.6 Data Analysis

After preprocessing, data is analyzed to detect the presence of the P300 waves. A P300 wave is an Event Related Potential (ERP), i.e., a positive deflection in the electroencephalogram (EEG) signal that occurs between 250 ms to 500 ms after stimulus onset. P300 is based on the oddball paradigm, which presents the participant with a sequence of repetitive audio or visual stimuli, infrequently interrupted by an unexpected stimulus.

Channel#1	Channel#2	Channel#3	Channel#4	Channel#5	Channel#6	Channel#7	Channel#8	Marker	Timestamp	Epoch	Target
-12926.94833	-28548.30537	-1059.092065	10165.04295	138.5802581	2014.485767	-14070.73706	-5637.579067	12	1.55143807e+…	2	1
-12211.7047	-29449.72411	-81.03598624	10424.94753	49.06419868	1692.981698	-12861.74747	-5327.772533	12	1.55143807e+…	2	1
-11383.89737	-30533.25793	532.0560326	10363.7761	-77.60087574	1196.105117	-11572.3105	-4937.967002	12	1.55143807e+…	2	1
-10537.89207	-31595.5115	688.3818084	10020.32596	-94.73179826	583.5451095	-10288.78866	-4554.094444	12	1.55143807e+…	2	1
-9767.558537	-32399.34722	465.2040253	9553.625297	115.9785937	-7.086033817	-9162.993677	-4283.215752	12	1.55143807e+…	2	1
-9131.446005	-32764.01811	81.50342971	9191.756848	603.1190304	-398.1591795	-8367.410115	-4207.447327	12	1.55143807e+…	2	1
-8636.252518	-32653.20248	-192.7257026	9137.584097	1326.165504	-437.5069621	-8033.774042	-4347.004683	12	1.55143807e+…	2	1
-8220.194101	-32209.23036	-137.2782927	9496.472673	2171.705729	-52.09296464	-8181.637137	-4631.032051	12	1.55143807e+…	2	1
-7784.29107	-31733.13402	327.2619719	10221.05285	2987.802801	710.6749025	-8693.356415	-4919.926567	12	1.55143807e+…	2	1
-7222.169524	-31592.64739	1115.7446	11132.86943	3632.865334	1691.759162	-9337.427497	-5050.012961	12	1.55143807e+…	2	1
-6484.138761	-32113.77218	2000.738377	11970.48504	4006.399873	2650.94976	-9850.261488	-4914.890431	12	1.55143807e+…	2	1

Figure 7.6: EEG data after preprocessing.

7.7 Methodologies and Algorithms Details

7.7.1 Snapshots of implemented code

```
242    def createLSL_stream():
243            # Creating an EEG Stream using pylsl
244        global inlet
245        print("Creating EEG stream ....")
246        stream_name = 'NIC'
247        streams = resolve_stream('type', 'EEG')
248        try:
249            for i in range (len(streams)):
250
251                if (streams[i].name() == stream_name):
252                    index = i
253                    print ("NIC stream available")
254
255            print ("Connecting to NIC stream... \n")
256            inlet = StreamInlet(streams[index])
257            print("Connected to EEG Inlet .....")
258        except NameError:
259            print ("Error: NIC stream not available\n\n\n")
```

The above code creates an eeg stream using the pylsl (lab streaming layer) library. It creates an inlet for data flow between the program and NIC software.

```
263    def Baseline_creation(command):
264        global outlet,unset_time,set_time,baseline, baseline_data,inlet
265        sample_count = int(((set_time+unset_time)/2)*12 )
266        if(command == 'start'):
267            createLSL_stream()
268            response = messagebox.askyesno("Success",f"Start Baseline recording?",parent=root)
269            if(response):
270                #send.....Marker
271                baseline_data = np.empty(shape=[0,8])
272                counter = 0
273                while counter < sample_count:
274
275                    arr = np.ones((1,8))
276                    sample, ts = inlet.pull_sample()
277                    arr= arr*sample
278                    sample = arr
279                    baseline_data = np.concatenate((baseline_data,sample),axis=0)
280                    counter+=1
281                inlet = None
282                print(baseline_data)
283                baseline = np.mean(baseline_data,axis = 0)
284                print("Baseline mean : ",baseline)
285                messagebox.showinfo("Success",f"Baseline recording collected.",parent=root)
```

The code given above shows how a baseline is collected for the eeg data. A sample count is set and data is collected until counter reaches the sample count value.

```
422  def set(index,type):
423      ## Sending the marker
424      global set_time ,ascii_value,box
425      font_size = 28
426      font_type = 'Times New Roman'
427      # vec = []
428      if(type=='CC'):
429
430          if(index<7):
431              index1=(index-1)*6+1
432
433              for i in range(index1,index1+6):
434                  #box[i].config(fg="white",font=("Courier", 19,'bold'))
435                  box[i].config(fg="white",font=(font_type, font_size,'bold'))
436
437              print ("row: ",index)
438
439          else:
440              index1=index-6
441              for i in range(index1,index1+31, 6):
442                  #box[i].config(fg="white",font=("Courier", 19,'bold'))
443                  box[i].config(fg="white",font=(font_type, font_size,'bold'))
444              print ("col: ",index1)
445
446      elif(type=="RBox"):
447          ascii_value = ord(str(box[index].cget('text'))[0])
448          box[index].config(background="blue",font=(font_type, font_size,'bold'))
449          print("index received : {}".format(index))
```

The set function has two cases. One is used for highlighting the result character in blue background while the other handles flashing the character during the epochs when data is being gathered.

```
453  def unset(index,type):
454      global unset_time, trainMode,box,List
455      font_size = 20
456      font_type = 'Times New Roman'
457      if(type == "CC"):
458          if(index<7):
459              index1=(index-1)*6+1
460              for i in range(index1,index1+6):
461                  box[i].config(fg="grey",font=(font_type, font_size))
462                  #box[i].config(bg="#0d0d0d",fg="grey",font=("Courier", 19))
463              print ("row ",index," returned")
464          else:
465              index1=index-6
466              for i in range(index1,index1+31, 6):
467                  #box[i].config(bg="#0d0d0d",fg="grey",font=("Courier", 19))
468                  box[i].config(fg="grey",font=(font_type, font_size))
469              print ("col ",index1," returned")
470
471          #intensification delay!
472          root.after(unset_time ,sendMarker,index)
473      elif(type== "RBox"):
474          box[index].config(background="black",font=(font_type, font_size))
475          #box[index].config(background="#0d0d0d",font=("Courier", 19))
476          if trainMode:
477              row = int((index-1)/6)+1
478              col = (index-1)%6+7
479              shuffle(List)
480              while abs(List.index(row)-List.index(col)) not in range(4,8):
481                  shuffle(List)
482              change_color(0)
483          else:
484              shuffle(List)
```

The unset function performs the opposite of the set function and is used to change the background and remove highlighting from a particular character.

```
492  def resultDisplay(row,col):
493      global pred_Text
494      global marker_dict, marker_list
495      ind = (row-1)*6+col-6
496      letter = str(box[ind].cget('text'))
497      print(f'row :{row}, colum: {col} letter: {letter}')
498
499      Text = pred_Text.get(1.0,"end-1c")
500      #Text = pred_Text.get('1.0',END)
501      Text += letter
502      pred_Text.configure(state = "normal")
503      pred_Text.delete('1.0',END)
504      ######kaam baki hai
505      pred_Text.insert(END, Text)
506      pred_Text.configure(state = "disabled")
507      pred_label3.configure(text=letter)
508      print ("Row: {}, Column: {} value: {} ".format(row,col,letter))
509      set(ind,"RBox")
510      StateController()
511      marker_list.clear()
512      marker_dict.clear()
513      root.after(7000,unset,ind,"RBox")
```

The result display function takes the predicted row and column as input parameters and uses them to calculate the index of the box in the grid thus also finding out the character value. This character value is appended to the predicted text box and the index is passed to the set function for highlighting.

```
518  def eeg_stream_controller(cmd):
519
520      global eeg_data,clf,sampler_type,dataset_type,start_simulation
521      if (start_simulation == False):
522          return
523      if cmd is 1 :
524
525          print("Sending Start....")
526          skt.send("Start".encode())
527          while True:
528
529              if skt.recv(1024).decode() =="Ok" :
530                  print("Connection established with the eeg stream reciever.")
531                  break
532              else:
533                  continue
534          change_color(0)
```

The eeg stream controller function controls the functioning of the eeg stream receiver, starting and stopping it when necessary.

```
659   def sendMarker(index):
660       global ascii_value,curr_Epoch
661
662       timestamp = int(time.time()*100)
663       if testMode:
664           marker_list[timestamp]= index
665       else:
666           vec = []
667           if(ascii_value <65):
668               temp = ascii_value - 22
669           else:
670               temp = ascii_value - 65
671
672           col = temp%6 + 7
673           row = fl(temp/6) + 1
674           curr_marker = index*1000000+(row+10)*10000+(col+10)*100+(curr_Epoch+10)
675           vec.append(curr_marker)
676           outlet.push_sample(vec)
677           print("Now sending marker: \t" + str(curr_marker)+"\n")
678
679       root.after(0,change_color,curr_Epoch)
```

The sendMarker function creates a composite marker consisting of the ascii value, row and column values and epoch number and sends it.

```
949   ##=====================Creating the 6x6 Grid of P300 Speller=====================##
950   for r in range(1, 7):
951       for c in range(1, 7):
952           if(6*(r-1)+c<=26):
953               frame[6*(r-1)+c] = Frame(roots,width=width,height=height,bg="white")
954               frame[6*(r-1)+c].pack_propagate(0) # Stops child widgets of label_frame from resizing it
955               box[6*(r-1)+c] = Label(frame[6*(r-1)+c], text=chr(64+6*(r-1)+c), borderwidth=0,
956               background="black", width = width, height = height, fg = "grey", font=("Courier", 19))
957               box[6*(r-1)+c].pack(fill="both", expand=True,side='left')
958               frame[6*(r-1)+c].place(x=(c-1)*width,y=(r-1)*height)
959
960           else:
961               frame[6*(r-1)+c] = Frame(roots,width=width,height=height,bg="white")
962               frame[6*(r-1)+c].pack_propagate(0)
963               box[6*(r-1)+c] = Label(frame[6*(r-1)+c], text=6*(r-1)+c-27, borderwidth=0,
964               background="black", width = width, height = height , fg = "grey", font=("Courier", 19))
965               box[6*(r-1)+c].pack(fill="both", expand=True,side='left')
966               frame[6*(r-1)+c].place(x=(c-1)*width,y=(r-1)*height)
```

This code is used in order to create the P300 alphabet grid for the P300 speller.

```
47    def csv_preprocessing(filename,subjectname,no_records):
48        global default_name, default_filename
49        print("Starting csv_preprocessing:")
50        src_path = os.path.join(os.getcwd(),'Easy')
51        dest_path = os.path.join(os.getcwd(),'Data')
52        dest_path = os.path.join(dest_path,subjectname)
53        for fname in os.listdir(os.path.join(os.getcwd(),'Easy')):
54            if fnmatch.fnmatch(fname,'*Session.easy'):
55                default_filename = fname
56                break
57        def_filename = os.path.splitext(default_filename)[0]
```

The data stored by the NIC software is in .easy format. The above code is used for converting the file format from .easy to .csv. This makes it easier to process the data since it is in a familiar csv format.

```
132 def eeg_filter(min_freq,max_freq,path,filename):
133     file = pd.read_csv(os.path.join(path,filename+".csv"),header=None)
134 
135     data = np.array(file.iloc[:,:8])
136     data = data.transpose()
137     ch_types =['eeg','eeg','eeg','eeg','eeg','eeg','eeg','eeg']
138     ch_names = ['PO7','P3','Fz','Cz','Pz','P4','PO8','Oz']
139     sfreq = 500
140     info = mne.create_info(ch_names=ch_names, sfreq=sfreq, ch_types= ch_types)
```

Filters the eeg data between .5 and 15 Hz.

```
191 def baseline_correction(baseline,path,filename):
192 
```

The baseline correction function performs baseline correction by subtracting the baseline from the gathered data and thus reduces the noise content in the data.

```
286 def erp_trial_averaging(path,filename,epochs,create_file = False):
287 
```

The above code is used for averaging the data over multiple epochs to get a more uniform representation of the data over time and also to minimize noise in it.

```
64  def read_EEG():
65      global marker_dict,eeg_data,inlet,ReadingData,sample_count
66      global records_per_marker
67      ReadingData = True
68 
69 
70      print("In  read_EEG.....")
71      print("Listening for Start Command....")
72 
73      ##   Starting the connection.
74      while True:
75          if cskt.recv(1024).decode() =="Start" :
76              print("Sending Ok....")
77              cskt.send("Ok".encode())
78              break
79 
80          else:
81              continue
```

Receives commands from eeg stream controller and according starts and stops receiving of eeg data from NIC.

```
204     if(sampler == 1):
205         from imblearn.combine import SMOTEENN
206         smote_enn = SMOTEENN(random_state=0)
207         X_resampled, y_resampled = smote_enn.fit_resample(X_train, y_train)
208 
209     elif(sampler == 2):
210         from imblearn.combine import SMOTETomek
211         smote_tomek = SMOTETomek(random_state=0)
212         X_resampled, y_resampled = smote_tomek.fit_resample(X_train, y_train)
213     elif(sampler == 3):
214         from imblearn.under_sampling import RandomUnderSampler
215         rus = RandomUnderSampler(random_state=0)
216         X_resampled, y_resampled = rus.fit_resample(X_train, y_train)
```

Shows the various sampling methods that can be deployed.

```
229         # Initialising the ANN
230         classifier = Sequential()
231         # Adding the input layer and the first hidden layer
232         classifier.add(Dense(output_dim = 9, init = 'uniform', activation = 'relu', input_dim = 8))
233         # Adding the second hidden layer
234         classifier.add(Dense(output_dim = 9, activation = 'relu'))
235         classifier.add(Dense(output_dim = 6, activation = 'relu'))
236         classifier.add(Dense(output_dim = 6, activation = 'relu'))
237         # Adding the output layer
238         classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))
239         # Compiling the ANN
240         classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
241         # Fitting the ANN to the Training set
242         classifier.fit(X_resampled, y_resampled, batch_size = 25, nb_epoch = 150)
```

The Neural Network model implementation uses a four layer structure with ReLu activation in the hidden layers and sigmoid activation in the output layer. Loss is calculated with the help of the binary crossentropy function and optimization is done with the help of 'adam' optimizer. It is trained on the sampled data.

```
246         #Grid search for finding the best hyperparameter
247         Cs =[0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1]
248         gammas =[0.001,0.01,0.1,1]
249         kernel = ['linear','rbf']
250 
251         from sklearn.svm import SVC
252 
253         param_grid={'C':Cs, 'kernel':kernel,'gamma':gammas}
254         gs = GridSearchCV(SVC(),param_grid,cv=5)
255         gs.fit(X_resampled, y_resampled)
256         dic = gs.best_params_
257         best_C= dic['C']
258         best_kernel = dic['kernel']
259         best_gamma = dic['gamma']
260         print("Chosen hyperparameters are:",gs.best_params_)
261 
262         from sklearn.svm import SVC
263         classifier =SVC(C=best_C,kernel = best_kernel, gamma= best_gamma)
264         classifier.fit(X_resampled, y_resampled)
```

The SVM algorithm, which uses the kernel, gamma and C parameters, is fed with the optimum values as calculated by Grid search. It is trained on the sampled data.

```
312     for i in range(len(y_pred)) :
313         #print(y_pred[i])
314 
315         if (y_pred[i] == 1):
316             marker = int(X_test[i, 8])
317             if(marker <7):
318                 row_dict[marker]=row_dict.get(marker,0)+1
319                 #print(row_dict[marker])
320             else:
321                 col_dict[marker]=col_dict.get(marker,0)+1
322                 #print(col_dict[marker])
323     predicted_row = max(row_dict.items(), key=operator.itemgetter(1))[0]
324     predicted_column = max(col_dict.items(), key=operator.itemgetter(1))[0]
325     return predicted_row, predicted_column
```

The output letter is the one which has been predicted maximum number of times by the classifiers over which testing has been done.

Chapter 8

Deployment And Maintenance

8.1 Installation

To work with the fully functional P300 Speller System there are some libraries and tools that need to be installed on a system. These libraries and tools provide various functionalities like communication between devices, etc. that are essential in the experiment evaluation.

Here we list the essential libraries that needs to be installed.

8.1.1 Python

Python is free- and open source software, and runs on most platforms, which makes it attractive for research institutions and substantially lowers the entry barrier for newcomers to the field. We decided to use Python as the programming language of choice because we think it is an excellent general purpose programming language with a large and comprehensive standard library.

8.1.2 PyLSL

This is the Python interface to the Lab Streaming Layer (LSL). LSL is an overlay network for real-time exchange of time series between applications, most often used in research environments. LSL has clients for many other languages and platforms that are compatible with each other.
Pylsl should work with any recent version of the liblsl library (which is also included with this package), on any operating system and with any recent Python version, including 2.7+ and 3.x.

8.1.3 MNE

MNE-Python software is an open-source Python package for exploring, visualizing, and analyzing human neurophysiological data such as MEG, EEG, sEEG, ECoG, and more. It includes modules for data input/output, preprocessing, visualization, source estimation, time-frequency analysis, connectivity analysis, machine learning, and statistics.

8.1.4 Pandas

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.

8.1.5 Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.
Keras is suitable as it :

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Chapter 9

Results

9.1 Outputs

Using artificial neural network, an accuracy of 90% was achieved.The predicted character for each epoch is close to the row and column containing the target character often predicting the row and column correctly. Also the classifier was trained on different subjects data collected either using session wise or subject wise.

Figure 9.1: Prediction of P300 Speller System.

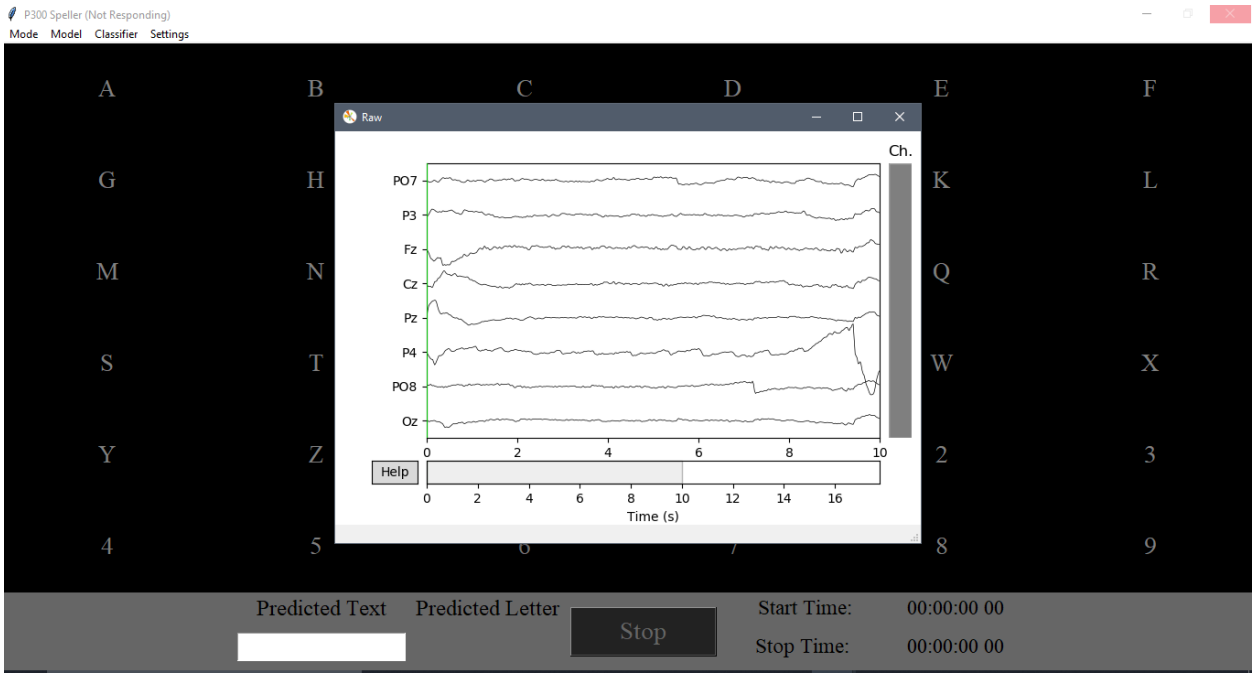


Figure 9.2: MNE Processing of EEG Test Data.

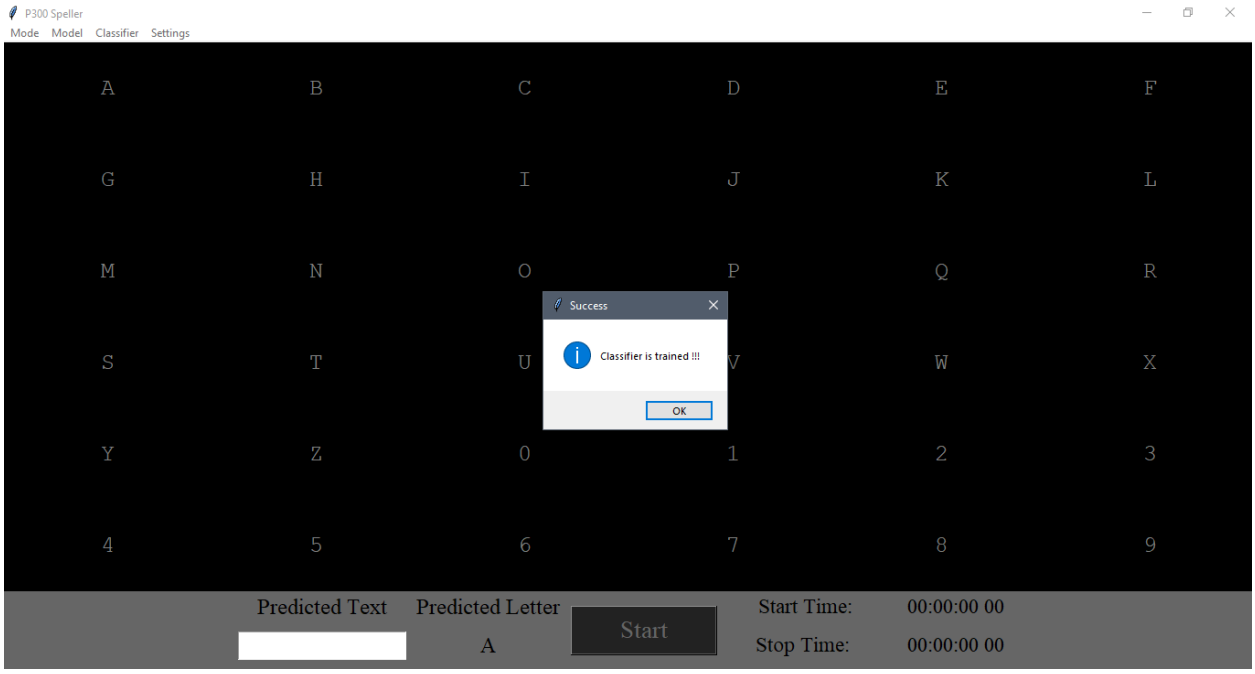


Figure 9.3: Classifier training on train data.

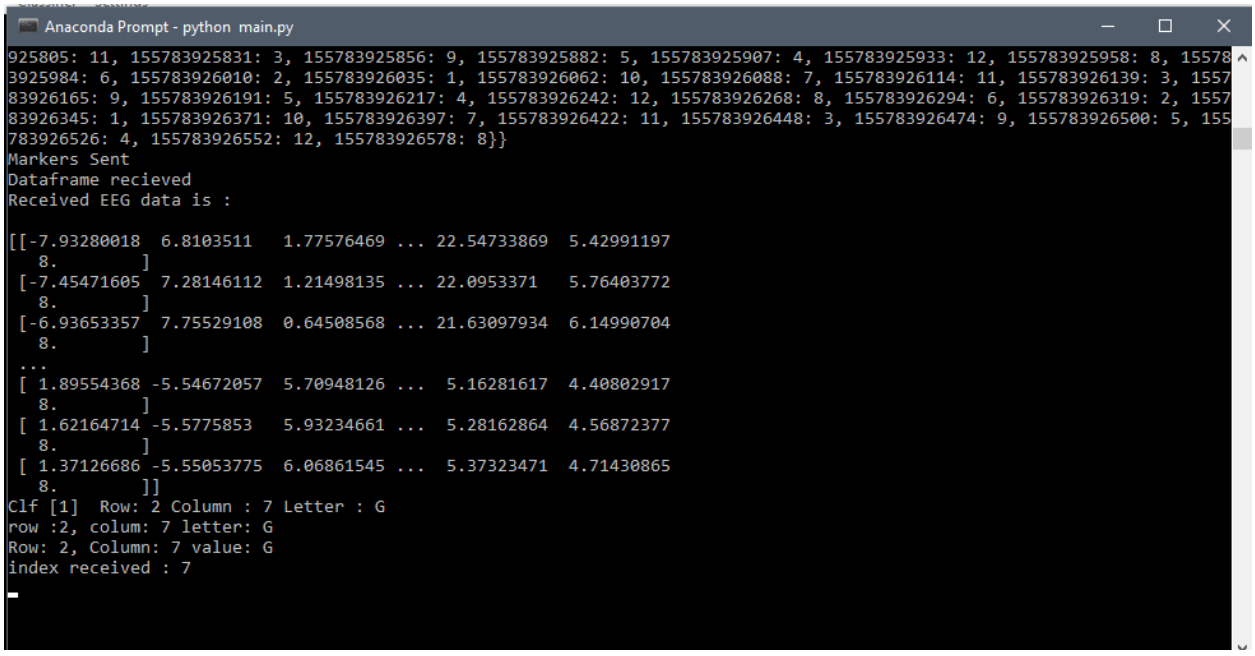


Figure 9.4: Prediction of character in P300 Speller System.

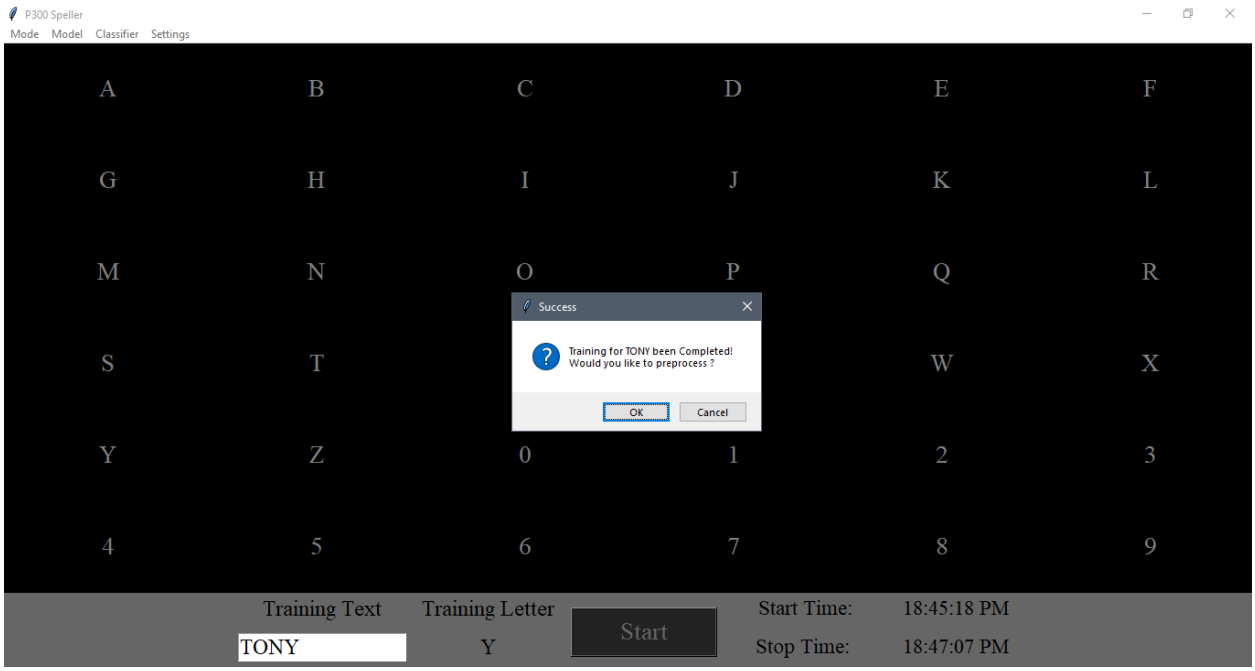


Figure 9.5: Training classifier on a training data.

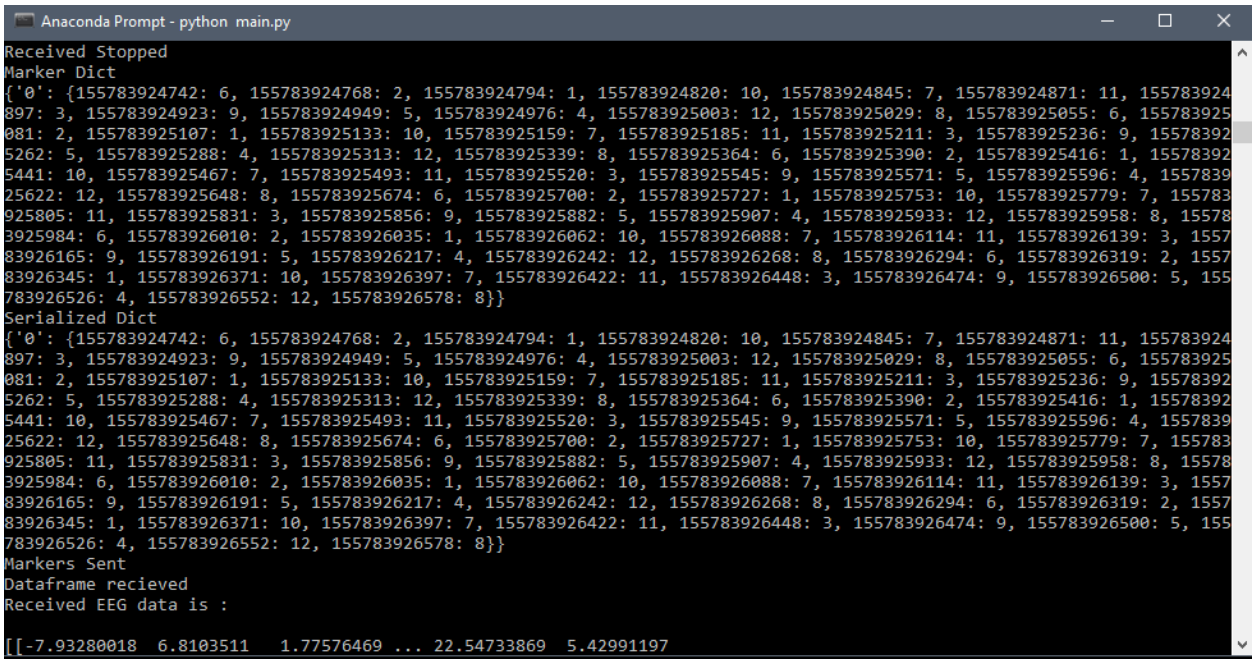


Figure 9.6: Creation of Marker Dictionary for each data sample.

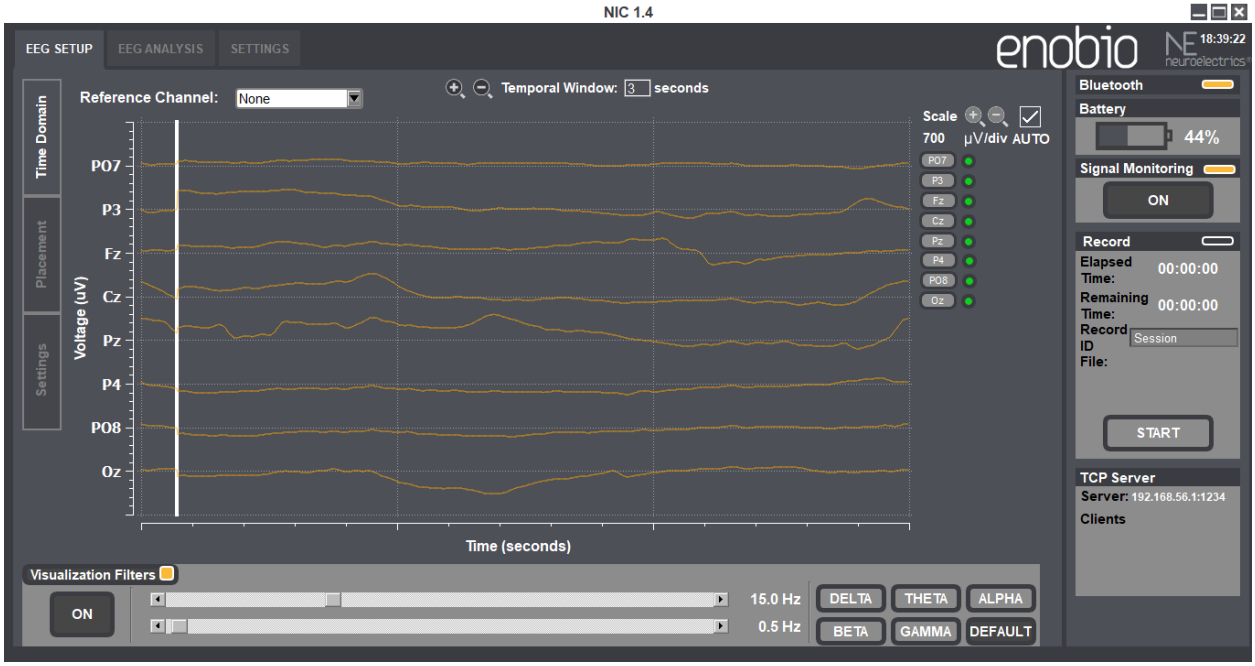


Figure 9.7: Electrodes voltage reading for a subject in NIC Software.

Chapter 10

Conclusions

Brain Computer Interface is a boon for people, which has provided a new hope to people that are suffering from physical disability. Brain Computer Interface is a way of expressing ones thoughts by interfacing brain with the computer and then displaying on the screen what the person is trying to say with help of different desktop interfaces. Using this technology requires successful detection of P300 waves from the EEG signals collected from brain. P300 signals are the ones that correspond to the characters that caused the stimulus and Non P300 signals are just random signals generated in the brain. The aim of this project is to distinguish these signals so that they can be used for further processing. This process is called classification. Detection of these signals require data preprocessing to remove noise that is always present in the collected signal. After reducing the data complexity, Support Vector Machine (SVM) and Artificial Neural Network (ANN) classifiers are used to distinguish the P300 and Non P300 signals. The result obtained is shown and the accuracy is also calculated. Finally using multiclass classifier characterization is performed which identifies the right character to spell. So, this project is about using BCI to help in detecting P300 signals and therefore the right character to spell.

Chapter 11

Future Work

A lot of research can be done in the field of brain computer interface. It is a huge field with a lot of scope for development. Some of the important works that can be carried out in future in this field are as follows

1. P300 detection varies with the data used from subject to subject. So, the neural network can be trained on a larger set of data so that accurate detection of P300 signals can be made which is the first step of any BCI system.
2. Better algorithms can be used to characterize the data set.
3. Development of hardware circuit to implement BCI i.e, interfacing circuits can be designed so that the system becomes a real time system that acquires signal from the brain and directly displays characters on the screen.
4. An overall operating protocol can be designed, so that the user can control how the system works. The how includes, for example, switching the system on or off, controlling what kind of feedback is to be provide and how fast, controlling the speed of implementation of commands, and switching between the device outputs.

Chapter 12

Discussions

In this research project, the attempt is to implement an efficient P300 speller system. Data has been collected from a wide range of subjects. The motive is to achieve an end to end system which can work on real-time data and can also be personalized for each individual subject. The approach is to use SVM and Artificial Neural Network(ANN) for classifying the real-time EEG signals in order to predict the sequence of characters that the subject has in their mind. For ANN different hyperparmeters like epochs, batch size and type of optimizer are tuned to detect the P300 signal in the captured EEG data.

Bibliography

[1] L.A. Farwell and E. Donchin, “Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials,” Electroencephal. Clin. Neurophysiol., vol. 70, no. 6, pp. 510â“523, 1988.

[2] A. Rakotomamonjy and V. Guigue, "BCI competition III: dataset II- ensemble of SVMs for BCI P300 speller," IEEE Trans. Biomed. Eng., vol. 55, pp. 1147, 2008.

[3] M. Kaper, P. Meinicke and T. Lingner, "BCI Competition 2003â”Data Set IIb: Support Vector Machines for the P300 Speller Paradigm,â 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Delhi, 2017.

[4] S. Kundu and S. Ari, "Score normalization of ensemble SVMs for brain-computer interface P300 speller," 2017 8th International Conference on Computing, Communicating and Networking Technologies (ICCCNT), Delhi, 2017.

[5] B. Blankertz, K. R. Muller, G. Curio, T. M. Vaughan, G. Schalk, J. R. Wolpaw, A. Schlogl, C. Neuper, G. Pfurtscheller, T. Hinterberger, M. Schroder, and N. Birbaumer, âœThe BCI Competition 2003: progress and perspectives in detection and discrimination of EEG single trials,â IEEE Trans Biomed Eng, vol. 51, no. 6, pp. 1044- 51, Jun, 2004

[6] L. F. Nicolas-Alonso and J. Gomez-Gill, "Brain computer interfaces, a review,", Sensors, vol. 12, no.2, pp. 1211-1279, 2012.

[7] R. Chaurasiya, N. Londhe and S. Ghosh, "An efficient P300 Speller System for Brain- Computer Interface," 2015 International Conference on Signal Processing, Computing and Control (2015 ISPCC).

[8] S. Xing, R. McCardle, S. Xie, "Reading the mind: The potential of electroencephalography in brain computer interfaces," 19th International Conference on Mechatronics and Machine Vision in Practice (M2VIP), Auckland, 2012.

[9] Cecotti H, Rivet B, Congedo M, Jutten C. A Robust Sensor Selection Method for P300 Brain-Computer Interfaces. Journal of Neural Engineering. 2011;8(1):1â“21.