

Brain-Computer Interfaces for Virtual Quadcopters
based on a spiking-neural network architecture -
NeuCube

Akshay Raj Gollahalli

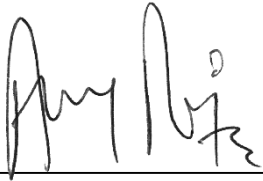
A thesis submitted to
Auckland University of Technology
in partial fulfilment of the requirement for the degree of
Master of Computer and Information Sciences (MCIS)

2015

Faculty of Design & Creative Technologies
School of Computer and Mathematical Sciences

ATTESTATION OF AUTHORSHIP

“I hereby declare that this submission is my own work and that, to my best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined or acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.”



Akshay Raj Gollahalli

April, 2015

Table of Contents

LIST OF FIGURES.....	V
LIST OF TABLES.....	VIII
LIST OF ACRONYMS AND ABBREVIATIONS	IX
LIST OF SYMBOLS.....	XII
ATTESTATION OF AUTHORSHIP	XIII
ACKNOWLEDGMENT	XIV
ABSTRACT	XV
CHAPTER 1: INTRODUCTION.....	1
1.1 Motivation	2
1.2 Research Scope and Focus	3
1.3 Research Questions	3
1.4 Overview of this study	3
1.5 Structure of the thesis	4
CHAPTER 2: BRAIN FUNCTIONS AND BRAIN DATA.....	6
2.1 Functionality of the Brain.....	6
2.1.1 Introduction to brain areas.....	6
2.1.2 Functions of the brain.....	7
2.2 Neuron-to-neuron Communication	9
2.2.1 Introduction to Neurons.....	9
2.2.2 Communication between neurons.....	11
2.3 Different types of brain imaging techniques:	13
2.3.1 Functional Magnetic Resonance Imaging (fMRI).....	14
2.3.2 Positron Emission Tomography (PET).....	16
2.3.3 Computed Tomography (CT).....	18
2.3.4 Electrocorticography.....	19
2.3.5 Electroencephalography (EEG).....	19
2.4 Placement of EEG Electrodes	21
2.5 Conclusion.....	23
CHAPTER 3: INTRODUCTION TO BRAIN COMPUTER INTERFACES	24
3.1 Introduction to BCI	24
3.2 Components of BCI.....	24
3.3 Functionality of BCI.....	25
3.4 Data Acquisition.....	26

3.4.1	<i>Invasive:</i>	26
3.4.2	<i>Partially Invasive:</i>	26
3.4.3	<i>Non-Invasive:</i>	27
3.5	Noise Reduction and Feature Extraction	27
3.6	Feature Translation and Model Creation	28
3.7	Feedback Generation.....	29
3.8	Biofeedback.....	30
3.9	Developments in BCI	30
3.9.1	<i>Control Signals</i>	30
3.9.2	<i>Different types of BCI</i>	31
3.9.3	<i>Current applications of BCI systems</i>	32
3.9.4	<i>BCI in communication</i>	33
3.9.5	<i>BCI in medical research</i>	33
3.9.6	<i>BCI in Entertainment</i>	34
3.10	An introduction to Quadcopters.....	37
3.11	Current limitation of BCI	39
3.12	Overview of Methods used in BCI	40
3.12.1	<i>EEG data acquisition for BCI</i>	40
3.12.2	<i>Different types of BCI platforms</i>	42
3.13	Conclusion	43

CHAPTER 4: SOME METHODS FOR DATA TRANSFORMATION PATTERN RECOGNITION AND MACHINE LEARNING FOR BCI..... 44

4.1	Some pattern recognition algorithms.....	44
4.2	Introduction to Neural Networks	44
4.2.1	<i>Perceptron Model</i>	46
4.2.2	<i>Multilayer Perceptron</i>	47
4.3	Spiking Neural Networks	48
4.3.1	<i>Hodgkin-Huxley model</i>	48
4.3.2	<i>Leaky Integrate and Fire</i>	49
4.4	Evolving Spiking Neural Network (eSNN).....	51
4.4.1	<i>Rank order population encoding</i>	51
4.4.2	<i>One pass learning algorithm</i>	52
4.5	Introduction to Spike-Timing Dependent Plasticity (STDP).....	53
4.6	Dynamic Evolving Spiking Neural Network (DeSNN)	54
4.7	Introduction to Fourier Transform.....	54
4.7.1	<i>Discrete Fourier Transform</i>	55
4.7.2	<i>Fast-Fourier Transform</i>	56
4.8	Introduction to Butterworth algorithm	56
4.9	Introduction to Threshold Based Encoding algorithm.....	57
4.10	Conclusion	57

CHAPTER 5: THE PROPOSED METHODOLOGY AND BCI SYSTEM DESIGN	58
5.1 Overview of the NeuCube Architecture	58
5.1.1 Modules in NeuCube.....	59
5.1.2 Classification of EEG data	60
5.2 Introduction to the concept of EEGRotor.....	60
5.3 Experiments with EEGRotor.....	62
5.3.1 Subjects used for this study.....	62
5.3.2 Experimental Procedure	62
5.3.3 Framework.....	64
5.4 EEGRotor for Brain-Computer Interface	66
5.4.1 EEGRotor Training Module	71
5.4.2 EEGRotor Testing Module.....	73
5.4.3 EEGRotor State of Mind Module.....	74
5.4.4 EEGRotor Activation Plot Module	75
5.4.5 Dynamic EEG Data Module	76
5.4.6 Total EEG Data Module.....	76
5.4.7 EEG Data Control Panel Module.....	77
5.5 Conclusion.....	78
CHAPTER 6: EXPERIMENTAL RESULTS AND ANALYSIS OF THE EEGROTOR	79
6.1 Working of the Complete Software Suite.....	79
6.1.1 Training NeuCube for BCI	79
6.1.2 Testing NeuCube for BCI.....	80
6.2 Analysis of Results	81
6.2.1 Results in tabular form	82
6.2.2 Visualisation	84
6.3 Discussion	86
6.3.1 Confusion matrix for Self-generated mapping.....	87
6.3.2 Confusion matrix for Talairach coordinates	89
6.4 Conclusion.....	91
CHAPTER 7: CONCLUSIONS	92
7.1 A Review of the Study	92
7.2 Limitations	93
7.3 Contribution of this Study	93
7.4 Future Work	94
7.4.1 Future Work on the EEGRotor	94
7.4.2 Hardware Implementation of EEGRotor	95
REFERENCES	98
APPENDIX A: CODE FOR EEGROTOR	111

testing.m.....	111
training.m.....	112
emotion.m	116
eeg_live.m.....	118
neucube.m.....	119
Plotting Option.....	120
EEGRotor controller code.....	121
Function for starting EEG data	122
Stopping EEG	123
Disconnecting EEG.....	123
APPENDIX B: SCREENSHOTS	124
EEGRotor Virtual Environment	124
NeuCube for BCI.....	125
EEGRotor Testing Module	126
EEGRotor Training Module	127
EEG Data Control Panel	127
EEG Data Activation Plot.....	128
Emotional State.....	128
Dynamic EEG Data	129
Total EEG Data.....	130
Blue Print of EEGRotor Hardware (Side View)	131
Blue Print of EEGRotor Hardware (Top View).....	131
Rendered Image of EEGRotor Hardware (Perspective View)	132
APPENDIX C: VIDEOS AND SOFTWARE LINKS.....	133
NeuCube for BCI running EEGRotor	133
EEGRotor VE Simulation.....	133
EEGRotor VE Code and Models	133

List of Figures

Figure 1.1: Trends in BCI for 2007-2013 (Ahn, Lee, Choi, & Jun, 2014).	2
Figure 1.2: An overview of this study	5
Figure 2.1: Structure of the Brain	6
Figure 2.2: Left & Right Hemisphere	7
Figure 2.3: Divisions of central nervous system (Eric R Kandel, 1985)	7
Figure 2.4: Examples of neurons (Haken, 2007)	9
Figure 2.5: (a) Cross section of a neuron cell body. (b) A synaptic neurotransmission. (c) A full neuron structure showing axon, dendrites and synaptic connection to other neuron. (Freeman, 2000) (BruceBlaus, 2013)	10
Figure 2.6: Synaptic Neurotransmission overview (Romano, 1995)	12
Figure 2.7: Electro-neurotransmission overview ("Potential, Electric," 2014)	13
Figure 2.8: Spike pattern of neuron-to-neuron communication (Greenfield, 1997).	13
Figure 2.9: Functioning of MRI	15
Figure 2.10: MRI Working.	15
Figure 2.11: Opening and closing of the eye can be measured by fMRI	16
Figure 2.12: Diagram of PET system	17
Figure 2.13: PET scanned image. Brighter regions show higher absorption of glucose (Madakasira et al., 2002).	18
Figure 2.14: Diagram of CAT scanner	18
Figure 2.15: CAT scan images	19
Figure 2.16: Delta wave	20
Figure 2.17: Theta wave	20
Figure 2.18: Alpha wave	20
Figure 2.19: Beta wave	21
Figure 2.20: Gamma wave	21
Figure 2.21: International 10-20 system outline	22
Figure 2.22: Emotiv EEG 10-20 system map	23
Figure 3.1: BCI overview	25
Figure 3.2: BCI Cycle.	25
Figure 3.3: Invasive electrodes	26
Figure 3.4: Partially invasive electrodes	26
Figure 3.5: Non-Invasive EEG data collection	27
Figure 3.6: Feature extraction (J. Wolpaw & Wolpaw, 2011)	27
Figure 3.7: Line equation	29
Figure 3.8: Data classification; the dotted lines represents non-classified classes.	29
Figure 3.9: Applications of BCI	33

Figure 3.10: BCI in Entertainment	34
Figure 3.11: (left to right) (a) A BCI based game where the subject plays Tetris using EEG signals. (b) A BCI based foosball which uses motor to control the shooting of the ball (Blankertz et al., 2010).	35
Figure 3.12: (a) A robot hand movement based on fMRI-BCI. (b) Quadriplegic women lifting a bottle with Invasive-BCI. (c) An EEG based BCI system which is capable of moving the prosthetic hand ..	36
Figure 3.13: GUI for the Media Communication Centre application based on P300 ERP	37
Figure 3.14: Model of Quadcopter	38
Figure 3.15: 256 and 4 channel EEG devices (“Wireless EEG system/4-ch MPEG-4 compression/ambulatory”).....	40
Figure 3.16: Emotiv EPOC and Neurosky MindWave	41
Figure 4.1: Two state neuron	45
Figure 4.2: Model of a neuron (Turner, 2015)	46
Figure 4.3: Perceptron model	47
Figure 4.4: Multi-layer perceptron	48
Figure 4.5: Hodgkin-Huxley model	49
Figure 4.6: Spike pattern	49
Figure 4.7: I&F model.....	50
Figure 4.8: LIF spikes	50
Figure 4.9: A population encoding based on Gaussian receptive fields. (a) For input value $\vartheta = 0.75$ (vertical thick line) and intersection points of each Gaussian is computed (inverted triangles). (b) Points are converted into spike time delays (N. Kasabov, 2013)	51
Figure 4.10: Algorithm for training eSNN	52
Figure 4.11: STDP with 60 spike pairing between pre- and post-synaptic potential	53
Figure 4.12: $x(n)$ is a frequency domain and $X(\omega)$ is a time domain. When $x(n)$ is sent through the Fourier it is changed into $X(\omega)$ and vice versa.	55
Figure 5.1: NeuCube architecture (N. K. Kasabov, 2014).....	58
Figure 5.2: EEG data classification flow in NeuCube (N. K. Kasabov, 2014)	60
Figure 5.3: EEGRotor Virtual Environment	61
Figure 5.4: End of the VE	62
Figure 5.5: Training the NeuCube for BCI to run EEGRotor	64
Figure 5.6: Testing the NeuCube for BCI to run EEGRotor	66
Figure 5.7: Functional diagram of NeuCube for BCI to control EEGRotor	68
Figure 5.8: NeuCube for Brain Computer Interface (BCI)	70
Figure 5.9: EEG data collection model to collect four class EEG data with its class label and create a final_eeg.mat file	71
Figure 5.10: 3D Matrix.....	72
Figure 5.11: Folder structure for Training phase.....	73
Figure 5.12: Interface of the testing module.....	73
Figure 5.13: State of mind module with raw EEG data, delta waves, theta waves, alpha waves and beta waves	74

Figure 5.14: Activation Plot for EEG data collected from Emotive headset	75
Figure 5.15: Dynamic EEG data plotted over time. Y-axis shows the amplitude and X-axis shows the time	76
Figure 5.16: Final EEG	77
Figure 5.17: Interface for the EEG data control panel	77
Figure 6.1: Training	80
Figure 6.2: (Left) The cube before initialising and showing the input features. (Right) Showing the trained cube after the training is done, thicker lines indicate stronger connections.	84
Figure 6.3: This cube shows the activation level for each feature; the brightness of dots indicates the level of activation.	84
Figure 6.4: (Left) The cube showing the Talairach coordinates before initialisation. (Right) The cube showing the connections between neurons, thicker lines indicate stronger connections.	85
Figure 6.5: The cube shows the activation level; the brightness of dots indicates the level of activation. ..	85
Figure 6.6: Comparison of actual class to classified class in self-generated mapping.	86
Figure 6.7: Comparison of actual class to classified class in manual input of mapping.	87
Figure 6.8: Test1 using self-generated mapping	88
Figure 6.9: Test2 using self-generated mapping	89
Figure 6.10: Test2 using self-generated mapping	89
Figure 6.11: Test1 using Talairach coordinates	90
Figure 6.12: Test2 using Talairach coordinates	90
Figure 6.13: Test3 using Talairach coordinates	91
Figure 7.1: NeuCube configuration with a new Module M10 for online learning, developed by the author.	95
Figure 7.2: Side view of quadcopter	96
Figure 7.3: Top view quadcopter.....	96
Figure 7.4: Rendered perspective view quadcopter.....	97
Figure B.1: EEGRotor virtual environment	124
Figure B.2: EEGRotor virtual environment ending	125
Figure B.3: NeuCube for BCI.....	125
Figure B.4: EEGRotor testing module.....	126
Figure B.5: EEGRotor training module	127
Figure B.6: EEG data control panel.....	127
Figure B.7: EEG data activation plot.....	128
Figure B.8: Emotional state	128
Figure B.9: Dynamic EEG data.....	129
Figure B.10: Total EEG data	130
Figure B.11: Blue print of EEGRotor hardware (side view)	131
Figure B.12: Blue print of EEGRotor hardware (top view).....	131
Figure B.13: Rendered image of EEGRotor hardware (perspective view).....	132

List of Tables

Table 4.1: Logical OR	45
Table 5.1: Four classes assigned for four facial movements	63
Table 6.1: Results using self-generated mapping – Test 1.....	82
Table 6.2: Results using self-generated mapping – Test 2.....	82
Table 6.3: Results using self-generated mapping – Test 3.....	83
Table 6.4: Results using manual input of mapping – Test 1	83
Table 6.5: Results using manual input of mapping – Test 2	83
Table 6.6: Results using manual input of mapping – Test 3	84
Table 6.7: Accuracy results.....	91
Table 7.1: EEGRotor parts.....	95

List of Acronyms and Abbreviations

ADHD – Attention Deficit Hyperactivity Disorder
AER – Address-Event Representation
ALS – Amyotrophic Lateral Sclerosis
AWT – Abstract Window Toolkit
BCI – Brain Computer Interface
BLDC – Brushless Direct Current
BOLD – Blood-Oxygen-Level Dependent
CAT – Computed Axial Tomography
CCA – Canonical Correlation Analysis
CCW – Counter Clock Wise
CT – Computed Tomography
CW – Clock Wise
DARPA – Defense Advanced Research Projects Agency
DeSNN – Dynamic Evolving Spiking Neural Network
dMRI – diffusion MRI
DNA – Deoxyribonucleic Acid
ECoG – Electrocorticography
ECP – EEG Control Panel
EDCP – EEG Data Control Panel
EEG – Electroencephalography
EROS – Event-Related Optical Signal
ERP – Event Related Potential
FC – Flight Controller
FDG – Fluorodeoxyglucose
FFT – Fast Fourier Transformation
fMRI – Functional Magnetic Resonance Imaging
GUI – Graphical User Interface
HCI – Human Computer Interaction

HCI – Human Computer Interaction
IFG – Inferior Frontal Gyrus
IP – Internet Protocol
ITR – Information Transfer Rate
LDA – Linear Discriminant Analysis
MEP – Motor Evoked Potential
MLP – Multilayer Perceptron
MR – Magnetic Resonance
MRI – Magnetic Resonance Imaging
MS – Multiple Sclerosis
NIRS – Near-Infrared Spectroscopy
NSF – National Science Foundation
PET – Positron Emission Tomography
REM – Rapid Eye Movement
RNA – Ribonucleic Acid
RO – Rank Order
ROI – Region of Interest
RPM – Rotations per Minute
SCP – Slow Cortical Potentials
SDK – Software Development Kit
SI – Système International d'Unités (International System of Units)
SNR – Signal to Noise Ratio
SSVEP – Steady-State Visual Evoked Potential
STBD – Spatio-Temporal Brain Data
STDP – Spike-Timing Dependent Plasticity
STG – Superior Temporal Gyrus
SVM – Support Vector Machine
TCP – Transfer Control Protocol
TLU – Threshold Logic Unit
UCLA – University of California Los Angeles
VE – Virtual Environment
VEP – Visual Evoked Potential

VTOL – Vertical Take-Off and Landing

List of Symbols

τ – Torque

Υ – Gamma rays (or Photos)

μV – Microvolt

w_i *or* $w[i]$ – Input weights

x_i *or* $x[i]$ – Inputs

ATTESTATION OF AUTHORSHIP

“I hereby declare that this submission is my own work and that, to my best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined or acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.”

Akshay Raj Gollahalli

April, 2015

ACKNOWLEDGMENT

I would like to thank my supervisor Professor Nikola Kasabov, for accepting me as a master student and for allowing me to be a part of an esteemed institute. Professor Kasabov's support, teaching and comments on my work were very important to my study. He steered me towards the right research path.

I would like to further extend my sincere thanks to Joyce D' Mello, manager of Knowledge Engineering and Discovery Research Institute (KEDRI). Her encouragement and kind words kept me working hard to successfully complete my study.

I would like to thank all PhD students and other staff at KEDRI who have supported me throughout my studies.

Last but not the least, I would like to thank my parents for all their support and encouragement in my good and bad times.

ABSTRACT

A novel framework is proposed in this study that uses a spiking neural network for learning spatio-temporal and spectro-temporal data called NeuCube. It is capable of learning and classifying such data in real time (online).

NeuCube-based methodology is proposed, tested and implemented for a control of Quadcopter using brain signals. A Quadcopter is a flying drone, which is used for its stability and ability to carry heavy loads for practical applications. There are three types of movement in flying drones/flight: Pitch, Yaw and Roll. These movements are controlled by a small controller called “Flight Controller”, which has its sensors such as 3-axis gyros, accelerometer, barometer and many more, which ensure a stabilised flying drone. Usually these are controlled by a radio control kit. In this study the path chosen is to control a Quadcopter with brain data.

In this study, a 14-channel Emotiv EPOC EEG device was used. In this experiment rather than using a real Quadcopter, the author uses a virtual environment to move a virtual drone in four directions. For each direction a facial movement was used to produce and record EEG data: Left eyebrow up, Right eyebrow up, both eyebrows up and both eyebrows down. NeuCube uses DeSNN (Dynamic Evolving Spiking Neural Network) to classify the data and to send the commands to virtual Quadcopter to move.

This experiment was tested during which the system showed good results.

Chapter 1: Introduction

The brain is a complex organ of the human nervous system. It is estimated that a Cerebral cortex consists of more than 20 billion of neocortical neurons (Drachman, 2005). Cerebral cortex is the largest part of the brain, which is divided into two parts: left and right hemispheres. A cerebrum (both hemisphere together) consists of different lobes which have different function of its own. It is divided into: Occipital Lobe, Temporal Lobe, Parietal Lobe and Frontal Lobe. The main function of Cerebrum is to control voluntary actions (E. A. Martin, 2010).

Di Sessa (1985) expressed the view that the future of using computers is not limited to scientists alone, but is for all. In this era we can see that everyone, including students, home users and various others use computers for their daily needs. The technology has advanced so much today that a thumbnail track pad was invented by an MIT researcher (Hardesty, 2015).

As the technology advances, its importance for medical and scientific use has also increased exponentially. Controlling machines by using brain signals has also become possible. Technologies such as Electroencephalography (EEG) and Functional Magnetic Resonance Imaging (fMRI), which can record brain signals have been developed. These signals from certain patterns that can be learnt by applying machine learning and pattern recognition methods and can be used to control a moving object. The branch of Human-Computer interface that deals with the interaction between the brain and the computer is called Brain Computer Interface (BCI). According to Vallabhaneni, Wang, and He (2005), BCI can be defined as a means of human communication with computers based on the neural activity among neurons which is independent of any muscular or peripheral activities.

When there is a physical activity (or performing tasks), there are neuron-to-neuron interactions which give rise to Neural Oscillation (Llinas, 1990), which was first discovered by Hans Berger that led to the discovery of EEG. The brain signals can record when a person does some task and then fed to a classifier which recognises the pattern associated to that task (Al-ani & Trad, 2010). This thesis is based on DeSNN (Dynamic Evolving Spiking Neural Network) algorithm which classifies four class problem and moves a virtual Quadcopter in a space.

This chapter consists of sections discussing motivation, background research, research questions, the author's contribution to this study and finally the study structure itself.

1.1 Motivation

BCI would help people who are immobile to control devices through thoughts and to communicate with other people. There are many examples of prominent politicians, scientists and actors, who suffered various diseases that made them immobile, some of them participating in the development of new BCI technologies.

Amyotrophic Lateral Sclerosis (ALS) is also known as Lou Gehrig's disease and is a neurodegenerative disease which affects nerve cells in the brain and spinal cord. Motor from the brain neurons are connected to the spinal cord, then from the spinal cord these neurons are distributed to muscles; since this is a degenerative disease, slowly these neurons die which makes the person immobile (*What is ALS?*). Doctor Cathy Wolf who is in her 60's, was diagnosed with ALS at the age of 18. She is an expert in Human Computer Interaction (HCI) and is a lead contributor for Wadsworth Center's development of BCIs. She has participated in different experiments including Test-to Speech (using BCI), P300 letter prediction (also known as "a-ha" response), word prediction and other (Wolf, 2013).

The proposed framework in this study aims at demonstrating that it is possible to develop BCI, based on EEG data to control an object.

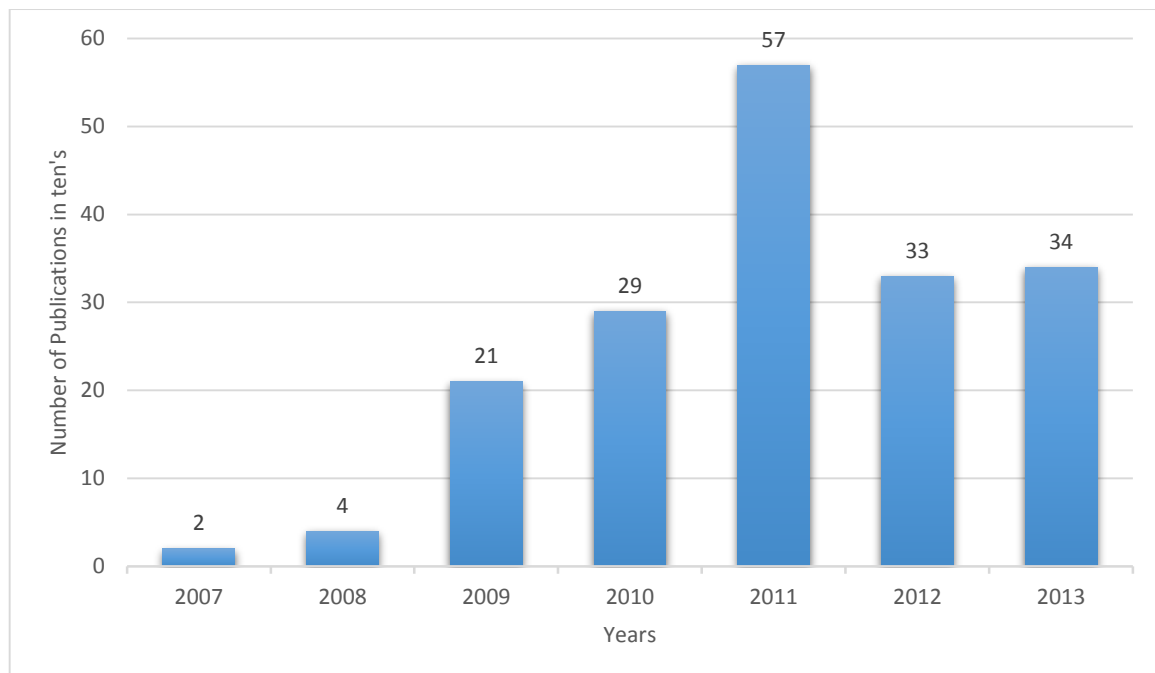


Figure 1.1: Trends in BCI for 2007-2013 (Ahn, Lee, Choi, & Jun, 2014).

In Figure 1.1, we can see that in year 2011 there were the highest number of publications related to BCI. The authors believed that portable and wireless EEG devices like Emotiv EPOC and Neurosky were factors which influenced the high number of publications. They also believed that BCI had more influence on gaming development rather than being used in clinical methods (Ahn et al., 2014).

1.2 Research Scope and Focus

The research scope for this study is to define if a human is capable of moving an object in a given space. We would be focusing more on DeSNN architecture rather than using any conventional method. More about this is Chapter 5:

1.3 Research Questions

The main objective for this research is to create a framework for BCI, making it easier for others to use compared to traditional implementation. Keeping this in mind the following research questions are formulated:

1. What is Brain Computer Interface (BCI)? What are the challenges?
2. How good Emotiv EPOC is compared to other EEG devices (medical devices) in collecting brain data.
3. How does NeuCube based architecture improve the performance of object control when compared to other traditional methods?
4. Can NeuCube-based BCI help people with paralysis, Multiple Sclerosis (MS) or Amyotrophic Lateral Sclerosis (ALS)?
5. How accurate is NeuCube framework in terms of controlling of a drone?
6. What are the limitations of this approach?

1.4 Overview of this study

- Literature review related to the study of brain data is presented;
- Literature review on BCI and its various implementations is also presented;
- The motivation for using Quadcopter for this study is explained;
- A new framework for BCI has been proposed based on the NeuCube architecture.
- An online/real time data processing system based on the NeuCube is implemented.
- An easily affordable EEG system is used, where a user can record data directly from the NeuCube software.
- For every recording (with input and output data), a person can see their emotional levels (alpha, beta, gamma and delta waves).

- No third party EEG data was used; all experiments conducted in this study was done on the author's data.
- The data collected was recorded in controlled conditions and only movement of eyelids were used.
- A virtual environment was developed for a multi-rotor (Quadcopter) flight to be controlled in a given space, where the user is given a feedback of score.
- Poster presentation was given on the same framework using NeuCube at NCEI 2015 workshop.
- An abstract was published in the NCEI 2015 proceedings.

1.5 Structure of the thesis

To answer all the research questions, this thesis has been structured as follows:

Chapter 2: This chapter focuses on how the brain functions. A detailed review of how neuron-to-neuron interaction takes place is also presented.

Chapter 3: This chapter presents BCI technologies and some advanced brain mapping systems.

Chapter 4: An introduction to a basic pattern recognition algorithm is presented; a few algorithms that are used in this study are explained as well.

Chapter 5: This chapter proposes a framework of a Brain Computer Interface (BCI) based on NeuCube architecture. The functionality of the module developed by the author for online/offline training/testing is explained.

Chapter 6: A pilot implementation of the new framework is described and tested on the author's data and experimental results are presented in this chapter.

Chapter 7: The study is concluded in this chapter.

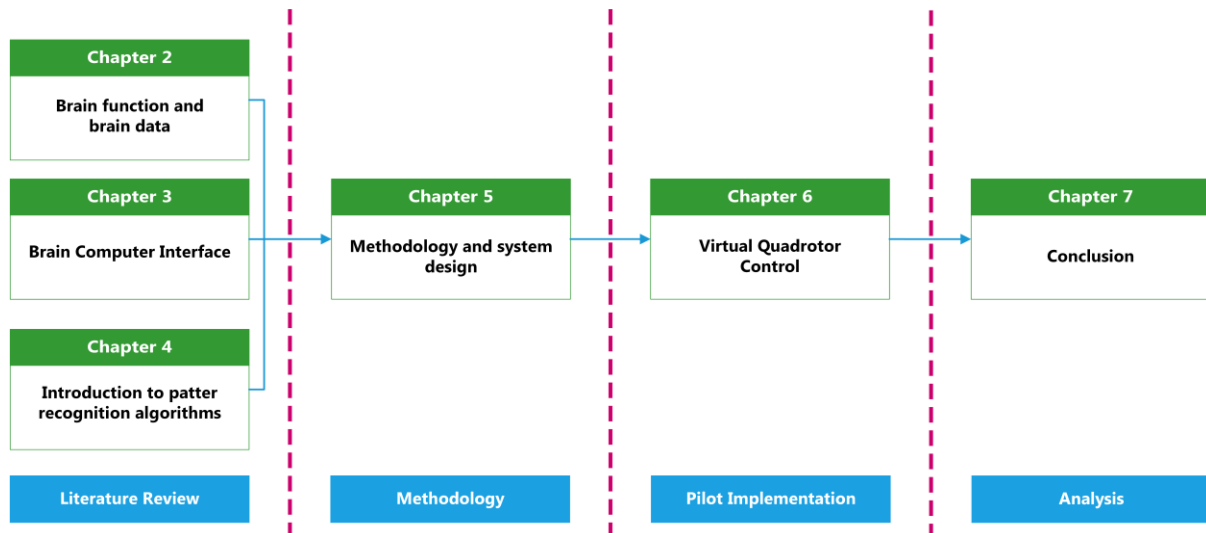


Figure 1.2: An overview of this study

Chapter 2: Brain functions and brain data

Before discussing the principles of BCI, the author would like to briefly describe the brain functions and different types of brain data. A detailed explanation on neuron-to-neuron communication is shown and also different types of brain imaging techniques are explained.

The chapter provides a literature review and discusses the working of the brain, communication between neurons and different types of brain data acquisition hardware.

2.1 Functionality of the Brain

The brain is considered to be a collection centre for storing information and behaviour (Freeman, 2000). It's a complex organ that has more than 100 billion neurons connected to each other in a most complex manner (Haken, 2006).

2.1.1 Introduction to brain areas

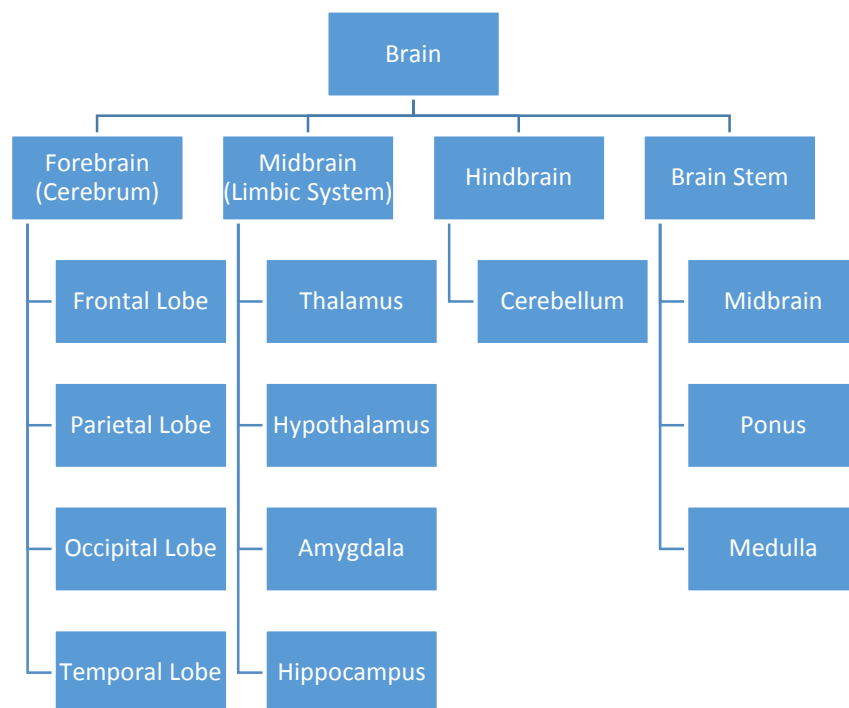


Figure 2.1: Structure of the Brain

Our focus for this study is on Forebrain (part of the Cerebrum), which is divided into two parts: right hemisphere and left hemisphere. Put together it's called cerebrum (Griggs, 2010) that can be seen in **Figure 2.2**.

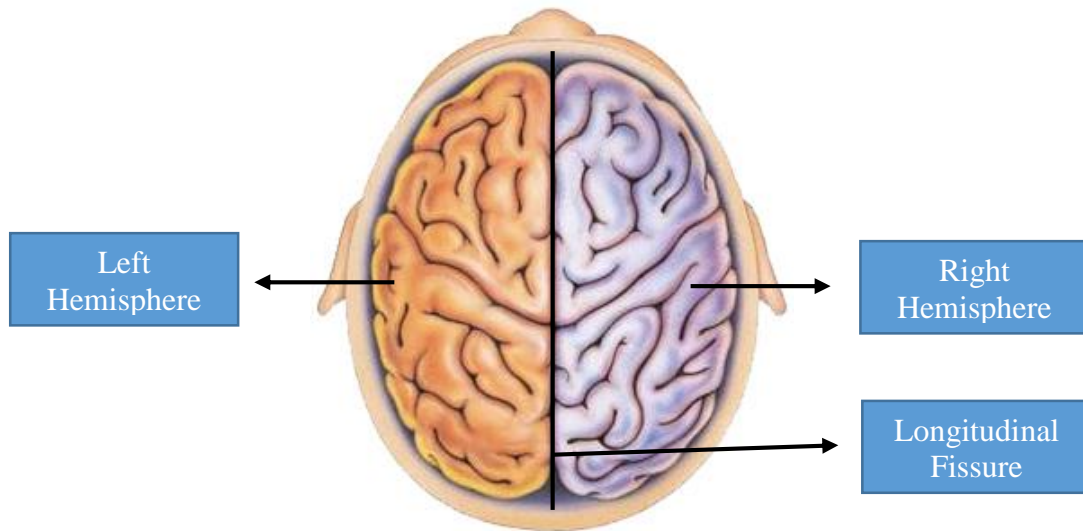


Figure 2.2: Left & Right Hemisphere

The brain is divided into two sections longitudinally by the longitudinal fissure (Colman, 2014e).

2.1.2 Functions of the brain

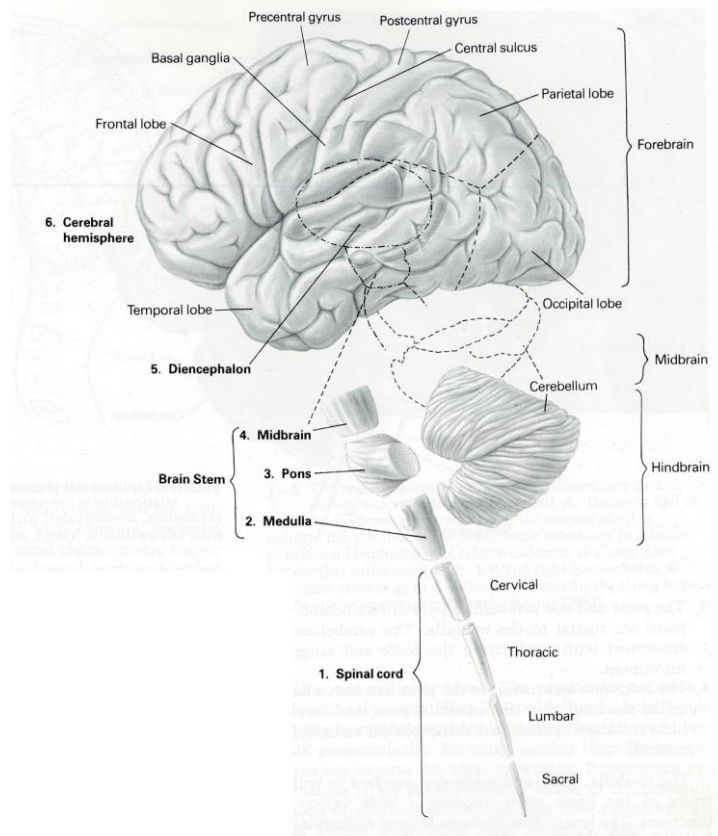


Figure 2.3: Divisions of central nervous system (Eric R Kandel, 1985)

Each division of the brain has its own functions and is part of the nervous system. The nervous system is divided into six parts (Eric R Kandel, 1985).

2.1.2.1 Spinal cord

It is a collection of neuron fibres which extend to the Medulla (Allaby). Its function is to carry motor and sensory signals throughout the body. It relates also to involuntary response to any stimuli, also known as “Reflex” actions (or Reflex Arc) ("Spinal cord," 2014)

2.1.2.2 Medulla

Formally Medulla is known as Medulla Oblongata, which is one of the three main parts of the brain stem. The presence of white matter is highest in this region with minimal amount of grey matter. The function of this region is to control Cardiac, Respiratory and Vasomotor muscles ("Medulla Oblongata," 2012)

2.1.2.3 Pons

Pons is also known as “bridge of Varolius”; it contains much white matter and very little grey matter ("Pons," 2012). Its function is to control force and range of movement (Eric R Kandel, 1985).

2.1.2.4 Midbrain

It acts like a junction between the cerebral cortex and the spinal cord. Its main function is to control the reflex actions of the visual and auditory systems, for example moving the head and eyes ("Midbrain," 2012).

2.1.2.5 Diencephalon

It is located in the anterior part of the cerebrum. It acts like a junction between sensory areas and higher brain centres. The optical nerve starts from this region ("diencephalon," 2014).

2.1.2.6 Cerebral Hemispheres

It is made up of an outer layer of grey matter and the inner layer of white matter ("Cerebral Hemisphere," 2012). This part of the brain is concerned with most of the human functions including perception, cognitive and motor movements (Eric R Kandel, 1985).

Most of the brain works in a cross over manner, that is, all the work you do is processed on the opposite hemisphere of the brain. For example, when you see something with your left eye, your right hemisphere processes it; likewise, when you touch something with your right

hand, the opposite (left) hemisphere associated to that part of the body processes it (except for a few facial nerves). Even the auditory processing is done in the opposite side of the hemisphere. Smell is the one sense organ that is not processed in the opposite hemisphere; that is, if you smell with the left nostril, that smell is processed only in the left hemisphere of the brain and vice versa (Carter & Frith, 1998).

2.2 Neuron-to-neuron Communication

2.2.1 Introduction to Neurons

There are over twenty types of neurons in the brain (Haken, 2007).

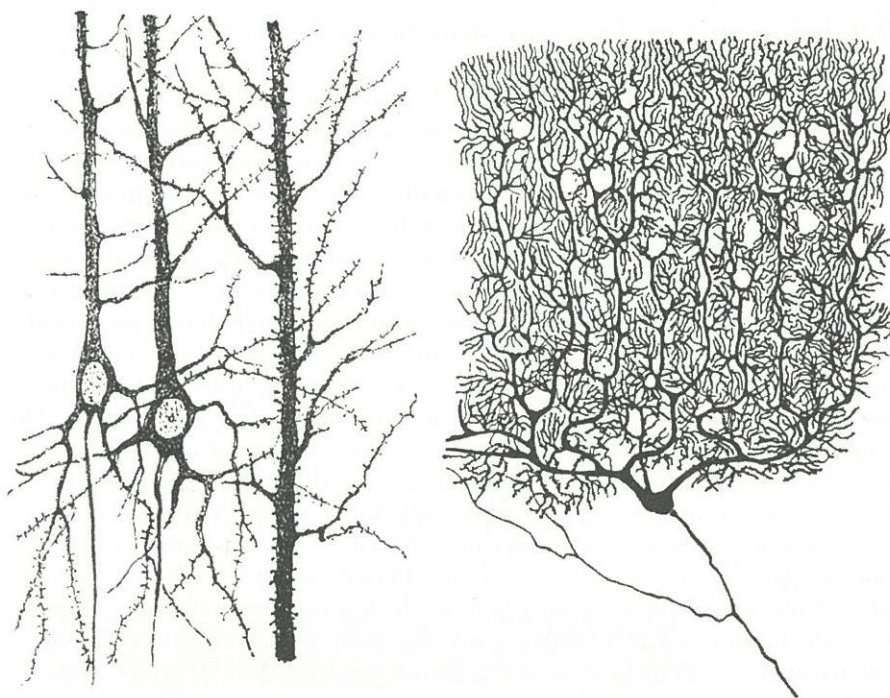


Figure 2.4: Examples of neurons (Haken, 2007)

Out of twenty types of neurons, two types of neurons are shown in **Figure 2.4**. The left part of the figure shows a cluster of neurons called Pyramidal cells and the right one is a Purkinje cell.

Every neuron cell has the following four main parts;

2.2.1.1 Synapse

It is the small gap between two neurons where chemical transmission takes place (neurotransmission) (Allaby, 2014).

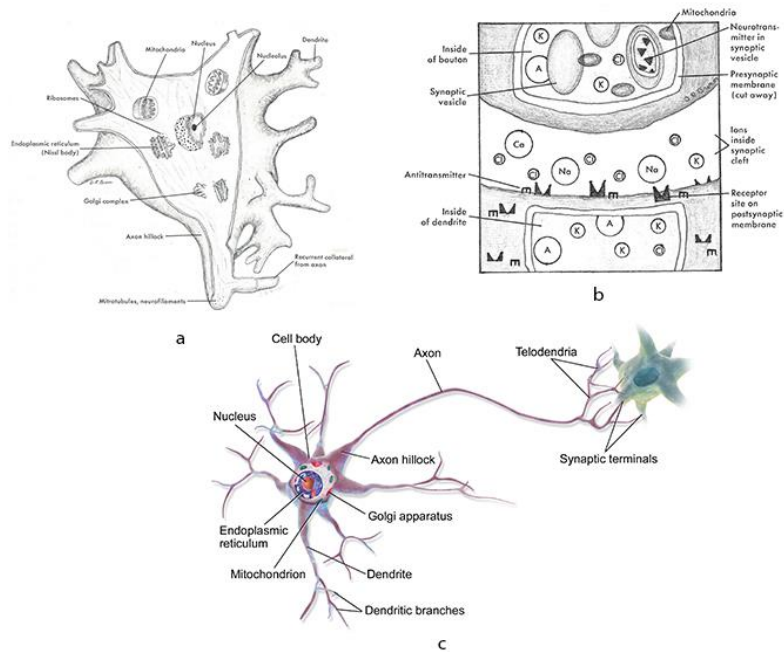


Figure 2.5: (a) Cross section of a neuron cell body. (b) A synaptic neurotransmission. (c) A full neuron structure showing axon, dendrites and synaptic connection to other neuron. (Freeman, 2000) (BruceBlaus, 2013)

2.2.1.2 Dendrites

They are small spine like structures that are short and connect to the cell body. Usually a neuron has two or more dendrites (D. R. Brown, 1980). They not only receive chemicals from the axon, but in a few sensory neurons, due to their external sensation input they exhibit receptor potentials (Jan & Jan, 2001).

2.2.1.3 Cell body

Neurons are built similarly to any other cell body. **Figure 2.5** shows the following: Nucleus (where RND and DNA is stored), mitochondria (the power house of the cell), golgi complex (storage place for RNA), endoplasmic reticulum (transport agent for the cell), nissl body (available usually in neurons and glandular cells, this produces all the proteins and enzymes for neurotransmissions) and microtubules (helps in flow of proteins and enzymes from neuron-to-neuro) (D. R. Brown, 1980).

2.2.1.4 Axon

This is one long fibre, which starts from soma that takes the action potential generated by the cell body and transfers it to another cell. The length of axon varies from neuron

to neuron but some are quite long. For example, the axons of some motor cells start from the spinal cord and end in the leg (D. R. Brown, 1980).

2.2.2 Communication between neurons

Before discussing about how neurons communicate (action potential), we should first know the basic principles of electricity. Ions with similar (positive-to-positive and negative-to-negative) charges repel each other and with different charges attract (D. R. Brown, 1980). A potential energy is required to move ions from one place to another. This potential difference is called voltage ("Potential, Electric," 2014). A flow of electrically charged particles moving from one point to another point, in the presence of a potential difference is called current. Its international system of unit is Amperes ("Electric Current," 2014).

There are two types of neurotransmission in a neuron: chemical transmission and electrical transmission. Chemical transmission happens in between dendrites and axons whereas electrical transmission happens between two dendrites (Stufflebeam, 2008).

2.2.2.1 Chemical Neurotransmission

In chemical neurotransmission two neurons are chemically connected, which means that there is no physical connection but there is a gap called synaptic cleft ("synaptic cleft," 2014). The neuron which sends chemical impulses is called presynaptic neuron and the one which receives it is called postsynaptic neuron. An action potential cannot cross a presynaptic neuron due to the presence of extracellular fluid in the synaptic cleft, but chemical transmission can take place. There are five types of ion exchanges in neurotransmissions (or neurotransmitters) that can happen between two neurons:

- Potassium ions (K^+) : Voltage of -90 mV (D. R. Brown, 1980)
- Chloride ions (Cl^-): Voltage of -70 mV (D. R. Brown, 1980)
- Protein molecules (A^-)
- Calcium ions (Ca^{2+})
- Sodium (Na^+): Voltage of +65 mV (D. R. Brown, 1980)

These ions can be sent through the synaptic membrane opening called ion channel, which acts like a valve between two pipes (axons and dendrites). These ion channels open and close whenever a neurotransmission takes place. As the ions are charged particles, when they are sent from a presynaptic neuron to a postsynaptic neuron (the ions enter the postsynaptic neuron only when the receptor site binds them together), a small electrical charge occurs. This electrical pulse is called synaptic potential. Now

the postsynaptic neuron fires action potential if and only when the sum of all the neurotransmitters (charged ions) exceeds its threshold. The transmission between neurons ends when the synaptic cleft is cleaned (removal of neurotransmitters). Most of the neurotransmitters are allowed to enter a presynaptic neuron to reuse the ions by a process called reuptake (Stufflebeam, 2008).

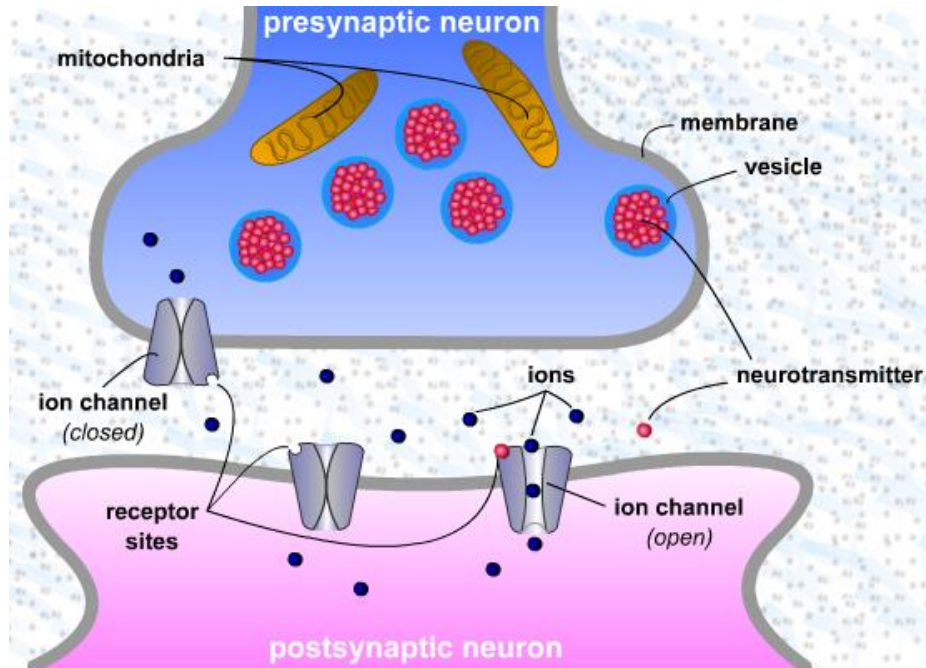


Figure 2.6: Synaptic Neurotransmission overview (Romano, 1995)

2.2.2.2 *Electrical Neurotransmission*

Electrical impulses occur when there is a physical contact between two neurons. As the channel opens and closes other chemicals also enter through it. As charged ions enter and exit the cells, a synaptic potential in one cell produces a synchronous synaptic potential in another neuron ("Electric Current," 2014).

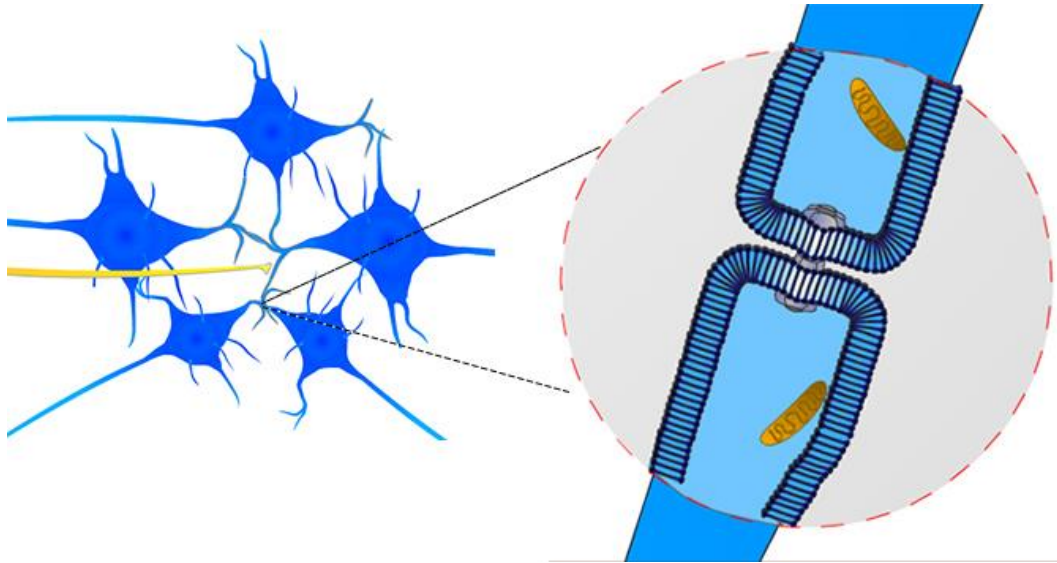


Figure 2.7: Electro-neurotransmission overview ("Potential, Electric," 2014)

Electrical impulses, as explained above, produce charged ions (current). The charge of each ion lasts for about one millisecond. This can be seen in **Figure 2.8**, where the asterisk is a stimulation to produce a spike.

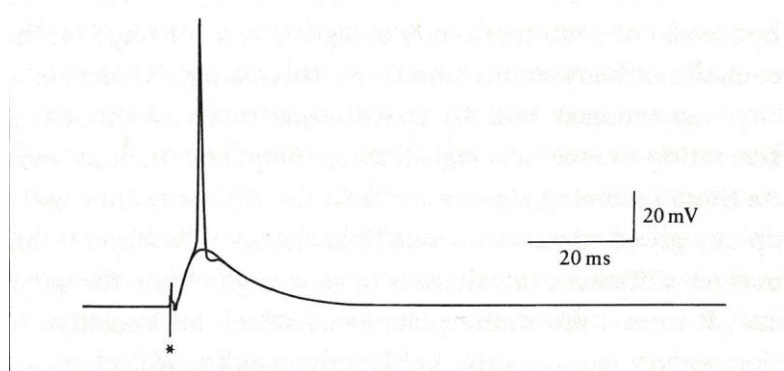


Figure 2.8: Spike pattern of neuron-to-neuron communication (Greenfield, 1997).

The whole process takes about 20 ms. Luigi Galvani was the first person who discovered that the neurons in the spinal cord could emit/generate electricity (Greenfield, 1997).

2.3 Different types of brain imaging techniques:

Brain imaging (also known as “neuroimaging”) is a type of non-invasive method of taking images of the brain ("Neuroimaging," 2012). The first tracking of blood flow was seen in the year 1890 by two neuroscientist (Roy & Sherrington, 1890). Later after the discovery of X-rays in 1895, a neurosurgeon in 1918 used X-rays for taking the first brain image (Fox, 1984);

this was the starting point for brain imaging. The following are some of the most commonly used neuroimaging techniques:

- Functional Magnetic Resonance Imaging (fMRI);
- Electroencephalography (EEG);
- Positron Emission Tomography (PET);
- Computed Tomography (CT);
- Magnetoencephalography;
- Single-Photon Emission Computed Tomography (SPECT);
- Event-related optical signal (EROS);
- Near-infrared spectroscopy (NIRS);
- diffusion MRI (dMRI);
- Electrocorticography (ECoG).

2.3.1 Functional Magnetic Resonance Imaging (fMRI)

Before going into the fMRI first let us look at what Magnetic Resonance Imaging (MRI) is. It was first discovered in 1973 (R. W. Brown, Cheng, Haacke, Thompson, & Venkatesan, 2014). MRI functions on a principle called Nuclear Magnetic Resonance (NMR) from which its name was derived. MR produces powerful magnetic fields that make the protons in the hydrogen atom align to the magnetic field applied. Once the magnetic field is applied, the proton spins and comes to rest (known as the equilibrium). This change in the alignment of the proton produces radio waves which are then measured by the MRI coils. When there is an equilibrium there is no emission of radio waves, so an alternate magnetic field is added perpendicular to the now magnetised proton. This added magnetic field makes the proton spin again. After reaching the required spin, the new magnetic field is removed; then changes in the radio waves are measured by the coils. These changes in the magnetic field of the atom can be mapped and turned in an image of brain which has (x, y, z) coordinates. **Figure 2.9** illustrates the functioning of MRI. This cycle is repeated for each anatomical slice of the brain, which is known as layer (*How MRI Works*).

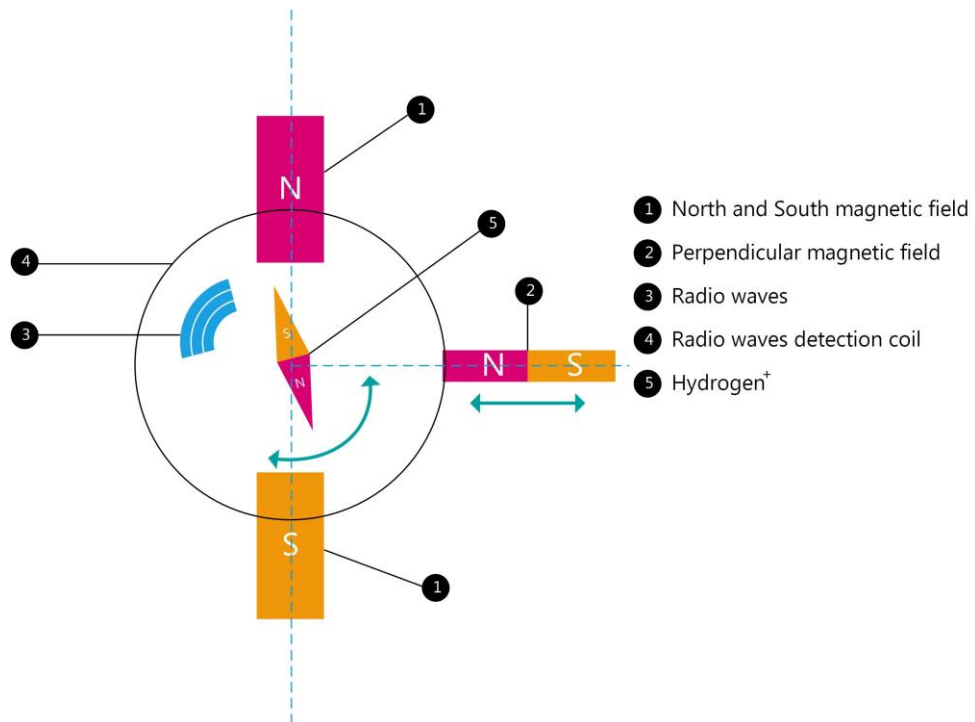


Figure 2.9: Functioning of MRI

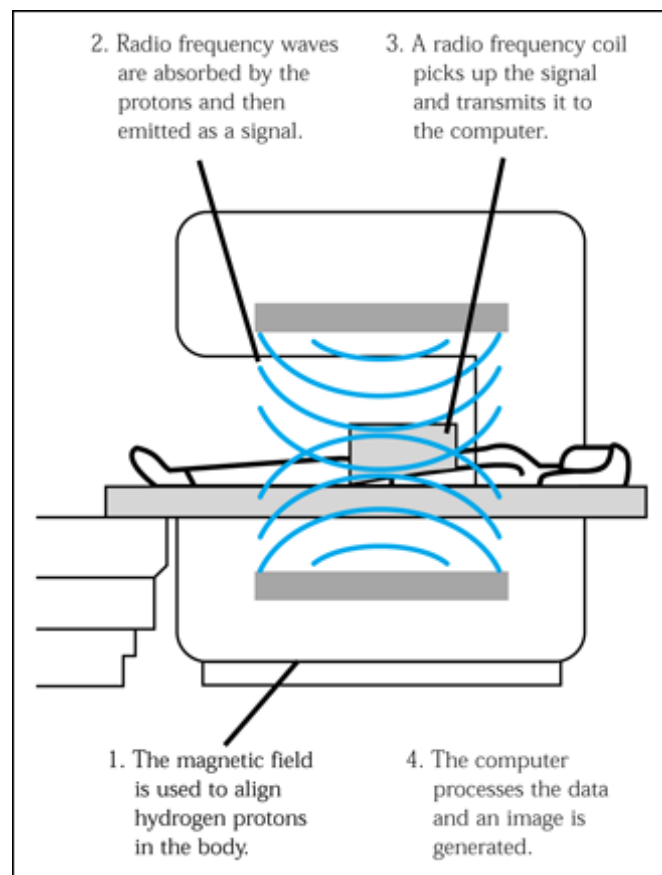


Figure 2.10: MRI Working.

The resolution of each layer is always high because of the flow of blood in the nerves. The brain has many connected nerves that supply blood & oxygen. Blood contains more than 92% of water (O'Neil, 2014) which is made up of hydrogen & oxygen (H_2O); since there are about 4-6 billion red blood cells per cubic millimetre (L, 2005), this ensures high resolution images of each layer.

fMRI works on the same principle as MRI, but the main difference is that fMRI detects Blood-oxygen-level dependent (BOLD) changes in the brain. When a human performs a task, the neurons tend to take more amount of oxygen. This increase in oxygen intake changes the magnetic field of the red blood cell. The changes are then detected by the fMRI system (*What is fMRI?*). This is known as BOLD. **Figure 2.11** shows an experiment of 250 seconds, where the subject is shown an image on the screen and is asked to open and close his /her eyes; the change in BOLD levels is observed in the graph.

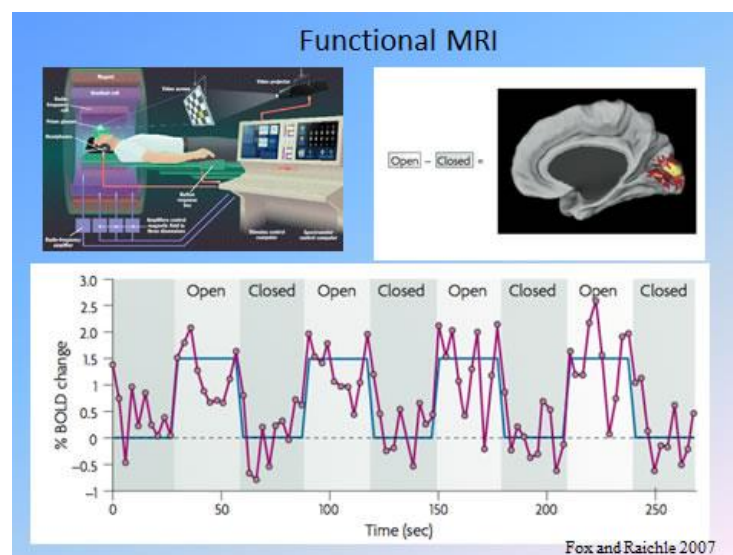


Figure 2.11: Opening and closing of the eye can be measured by fMRI

2.3.2 Positron Emission Tomography (PET)

PET scans are a type of nuclear medicine that uses radioactive isotope to produce gamma rays from the available electrons inside the body. A radioactive carrier which is specially made glucose molecules called Fluorodeoxyglucose (FDG) is injected into the body of the patient/subject. The subject is kept under observation for a few hours (1 to 2 hours) so that no reactions to the injected substance are found. It is known that the tumours absorb higher amount of sugars than normal tissues. Once the sugar is absorbed the radioactive molecule is decomposed with in the tumour and releases

positrons. These positrons are then bombarded with electrons in the body to form photons (or also known as gamma rays). These newly formed photons are bounce in the opposite side of the bombardment. They are then flushed out of the body. This is detected by the photon (or gamma) cameras that are attached to the scanner. This scanner computes the exact two dimensional coordinates of the photons emitted. The bigger the tumour, the more is the absorption of FDG. When a large amount of FDG is absorbed, higher amounts of positrons are emitted which are then bombarded with electrons rapidly to form multiple photons (or gamma rays) per image. This would show the size of the tumour ("Positron Emission Tomography (PET Scan),"). **Figure 2.12** presents the two stages of process. When positron (e^+) and electron (e^-) collide with each other it provides gamma (γ) rays, which are then read by gamma cameras. In the same figure the blue dotted lines signify imaginary coordinates. The image on the right side of the figure shows gamma rays deflecting each other. These deflected positrons produce gamma rays that are detected by the cameras which show their coordinates. In this case it is (2,6). There is a hazard using PET scan. **Figure 2.13** shows PET scanned image.

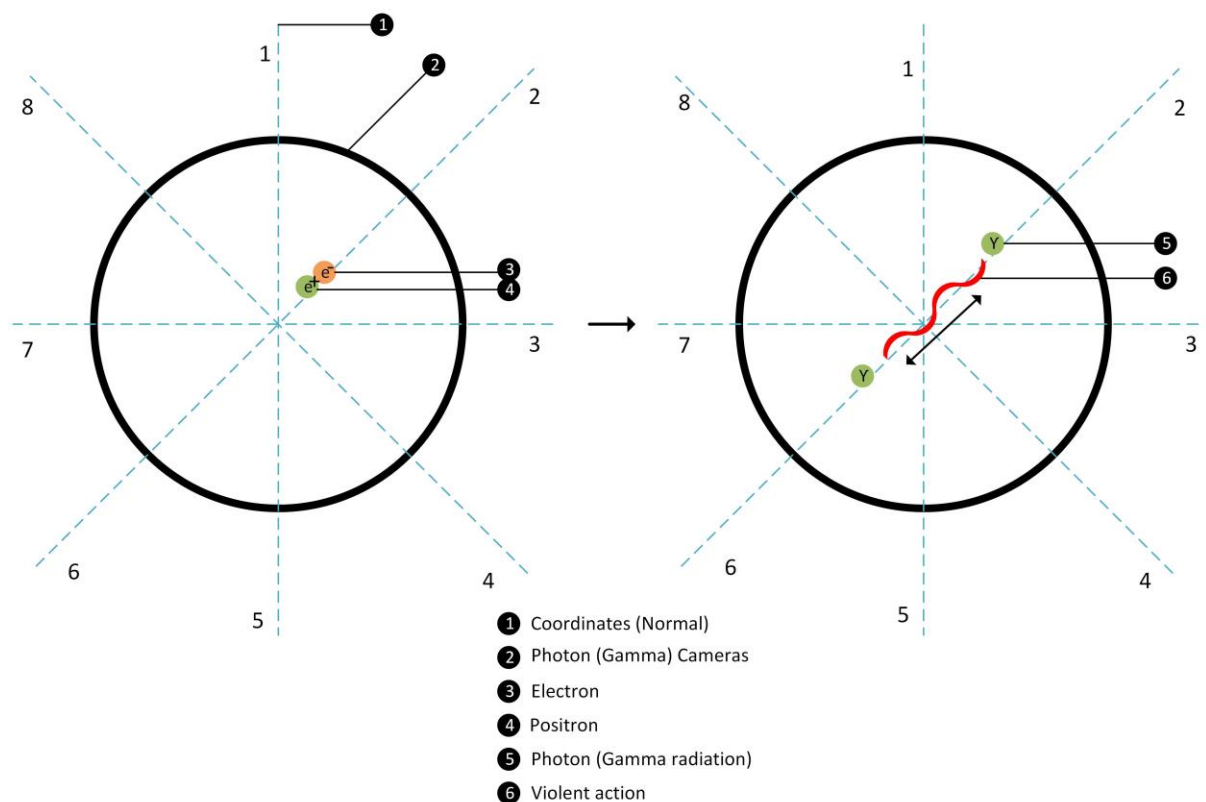


Figure 2.12: Diagram of PET system

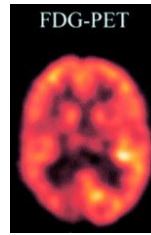


Figure 2.13: PET scanned image. Brighter regions show higher absorption of glucose (Madakasira et al., 2002).

2.3.3 Computed Tomography (CT)

Similarly to PET scanner, Computed Tomography (CT) also known as CAT (Computed Axial Tomography), takes images in the form of slices. These slices are 2D images. When these slices are stacked together using sophisticated software techniques, it forms a 3D image. The main difference is the way the images are produced. Unlike MRI (which uses magnetic field) and PET (which uses nuclear medicine techniques), CAT uses X-ray as its main source of imaging. The X-ray photographer rotates 360° around the subject (or a place of concentration) (Kalender, 2011).

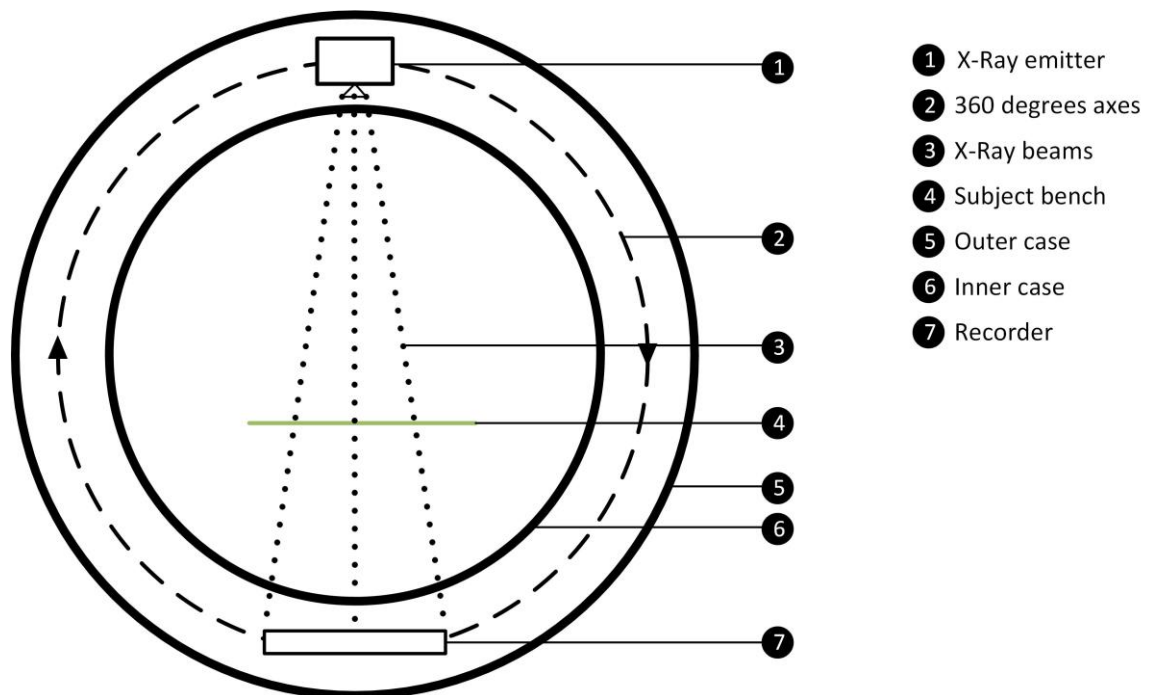


Figure 2.14: Diagram of CAT scanner

Figure 2.14 shows the internal structure of a CAT scanner, where the subject is placed on a bench and the X-ray emitter takes multiple images as it rotates around the subject.

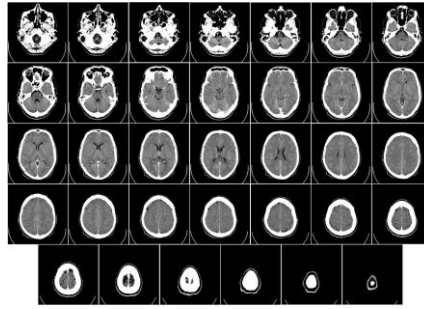


Figure 2.15: CAT scan images

Figure 2.15 shows multiple scans of a brain taken by CAT scanner.

2.3.4 Electrocorticography

Electrocorticography (ECoG) is a partially invasive EEG device which is used to acquire EEG data. Like any other data acquisition tool this also has its pros and cons.

2.3.5 Electroencephalography (EEG)

In the year 1875, a neuroscientist named Richard Caton, was the first to discover that there are electrical activities between neurons (Finger, 2001). Later, in the year 1929, the German psychiatrist Hans Berger was the first person who recorded brain data of a human and found the first type of wave called the alpha wave ("Electroencephalography," 2015). Since then the number of research studies has increased exponentially. EEG is a device which is used to measure the electrical activity of the neurons from the scalp without any invasive methods. In the medical field, EEG devices are used to diagnose epilepsy and sleep disorders (Martin & McFerran). This type of brain signals acquisition system is similar to ECoG systems in terms of collecting data, but physically EEG is placed on top of the scalp whereas ECoG is placed in the subdural region.

Over the years new and improved systems of EEG have appeared that can acquire data easily.

After the discovery of the brain waves, scientists have used EEG devices to acquire other types of brain patterns. These brain patterns show the excited or relaxed state of mind. There are six major types of brain frequency activities that can be measured with the use of EEG device ("Electroencephalography," 2015) (Colman, 2014f) (Colman, 2014d). They are briefly described below.

2.3.5.1 Delta wave

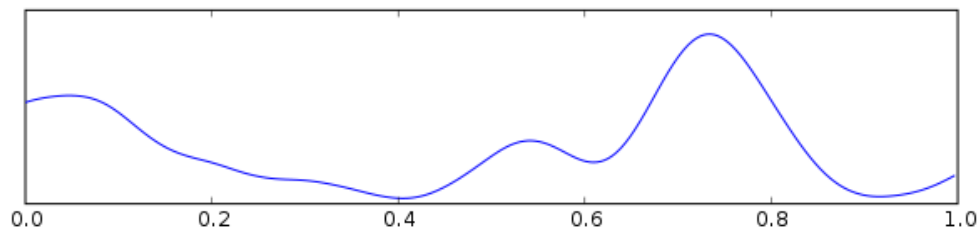


Figure 2.16: Delta wave

Figure 2.16 shows a delta wave that has low frequency and higher amplitude, ranging from 1 to 3 Hz and a voltage of around 150 μV . This type of wave is also called slow-wave sleep and occurs when the subject is in deep sleep without dreams (Colman, 2014c).

2.3.5.2 Theta wave

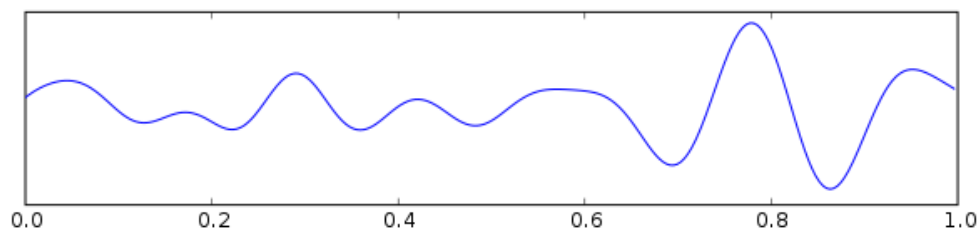


Figure 2.17: Theta wave

Figure 2.17 shows a different type of brain signal where the frequency has increased when compared to delta wave. The frequency ranges from 5 to 8 Hz. This is also called Theta rhythm, which usually occurs in the hippocampus region. This occurs when the subject is in Rapid Eye Movement (REM) sleep or semi-deep sleep (Colman, 2014h).

2.3.5.3 Alpha wave

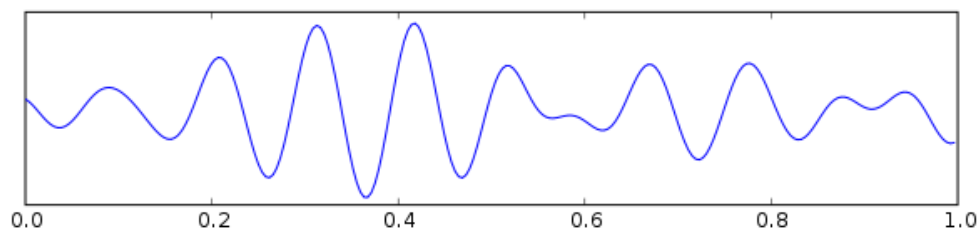


Figure 2.18: Alpha wave

Figure 2.18 shows the Alpha wave which has relatively higher amplitude and medium frequency of 8 to 12 Hz. This wave usually occurs when a person is relaxed (awake) and eyes are closed (Colman, 2014a) but not sleeping.

2.3.5.4 Beta wave

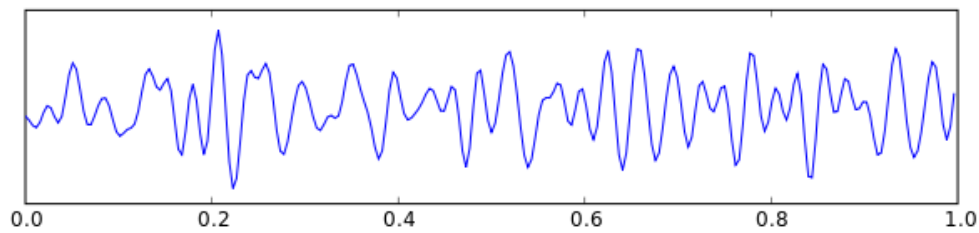


Figure 2.19: Beta wave

Figure 2.19 shows a low voltage wave, also called Beta rhythm. The frequency ranges from 10 to 30 Hz and usually occurs when the person is in aroused state (Colman, 2014b).

2.3.5.5 Gamma wave

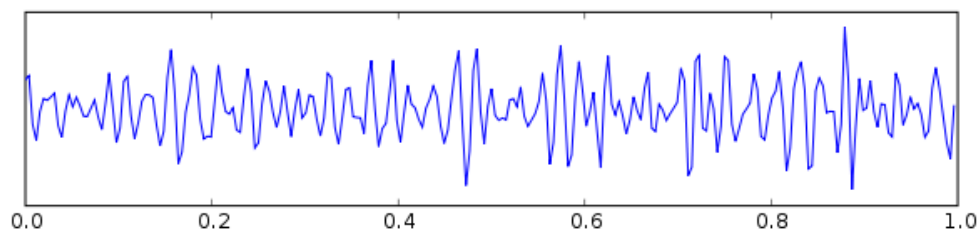


Figure 2.20: Gamma wave

The Gamma wave shown in Figure 2.20 has relatively high frequency (when compared to other waves) that ranges from 35 to 75 Hz. This type of wave is also called 40 Hz oscillation and occurs when the person is thinking about something interesting (Colman, 2014d).

2.3.5.6 Mu wave

Also called the wicket rhythm that has a frequency ranging from 8 to 13 Hz. This occurs in the motor cortex of the human brain. This wave is suppressed when the person makes any physical movements of his/her muscles and also tends to get suppressed when he/she even thinks of doing such actions. This wave is also suppressed when the person sees someone else making any muscle movement (Colman, 2014g). This wave overlaps with alpha waves as this frequencies tend to be almost the same which forms the same wave patterns as shown in **Figure 2.18**.

2.4 Placement of EEG Electrodes

The placement of EEG electrodes is instructed by an international system called “10-20 systems”. The number 10 refers to the mean distance between Nasion to first electrode and

Inion to last electrode. The number 20 refers to the mean distance between electrodes to the adjacent electrodes. 10% mean distance between Nasion to first electrode (front of the brain, above nasal cavity) and Inion (back of the brain, above the neck) with respect to the last electrode (Herwig, Satrapi, & Schönfeldt-Lecuona, 2003).

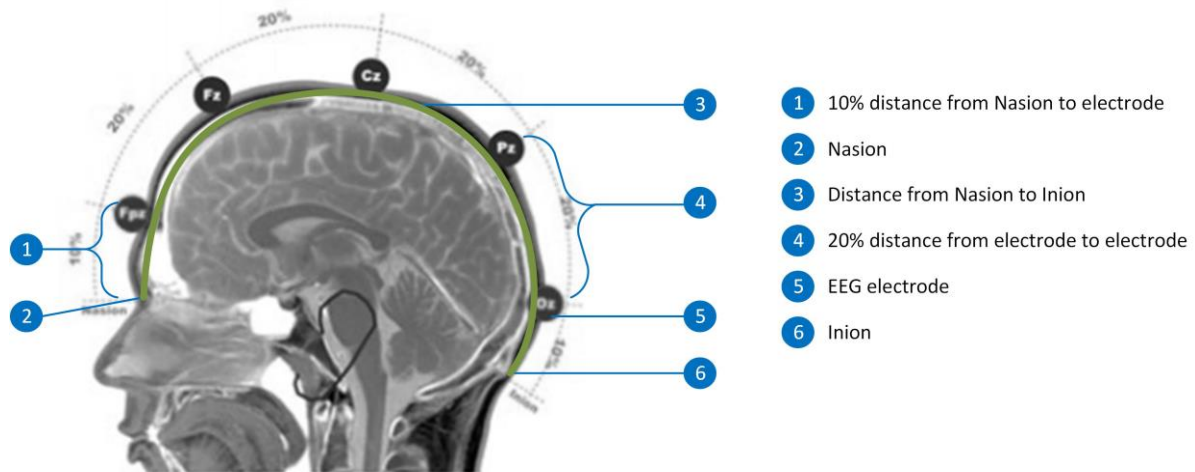


Figure 2.21: International 10-20 system outline

From the above Figure 2.21, first the distance between Nasion and Inion is taken (the distance of 50% of the total distance is taken from one side of the ear to the centre). 10% of the total measurement is given to Nasion & Inion and marked as one electrode. From the first marked electrode 20% distance is taken towards the centre and marked as another electrode. This cycle is repeated till the last electrode placed near Inion is reached.

Our study uses a commercial EEG recording system Emotive EPOC (*Epoc*) which records 14 channels of brain signals as shown in **Figure 2.22**

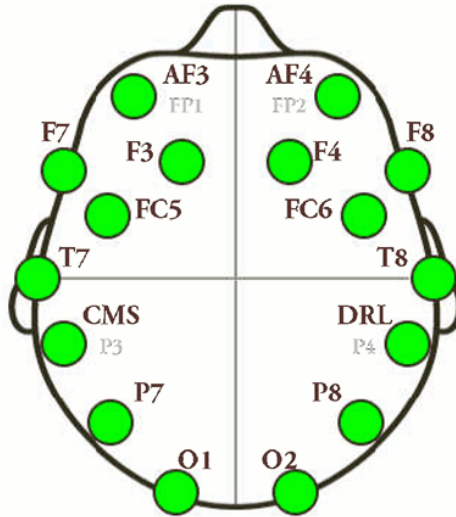


Figure 2.22: Emotiv EEG 10-20 system map

2.5 Conclusion

The chapter describes briefly how the brain functions, shows a detailed view of neuron-to-neuron communication and finally reviews a few brain imaging techniques. The EEG technique will be used in the following chapters for the creation of a new BCI framework based on the NeuCube spiking neural network architecture (N. K. Kasabov, 2014).

Chapter 3: Introduction to Brain Computer Interfaces

This chapter introduces Brain Computer Interfaces (BCI), their functioning and types of data acquisitions. Furthermore, current developments and limitations are shown and an introduction to quad copter (also known as drone) is given.

3.1 Introduction to BCI

As the name suggests, BCI is a method used for communication between humans and computer based on brain activity. The research of BCI was first started in University of California Los Angeles (UCLA). The research funding was granted by National Science Foundation (NSF), then later followed by a contract from the Defense Advanced Research Projects Agency (DARPA) (Martin Weil & Randolph, 1994).

3.2 Components of BCI

A BCI is not a single machine, but a collection of multiple components. Following are the main components of a complete BCI system:

- **EEG device:** Electroencephalograph (EEG) is a device which is used to collect signals generated by the electrical activity of neurons in the brain (as described in section 2.3.5)
- **Amplifier:** An amplifier boosts the signals collected by the EEG so that the computer can detect difference between signals;
- **Pattern Recognition software:** This is the heart of the BCI systems. Its main use is to recognise the patterns in the incoming signals and classify them accordingly.
- **Computer:** A computer is needed to connect all the components together and do the computation for the pattern recognition.

Figure 3.1 shows the overview of BCI system.

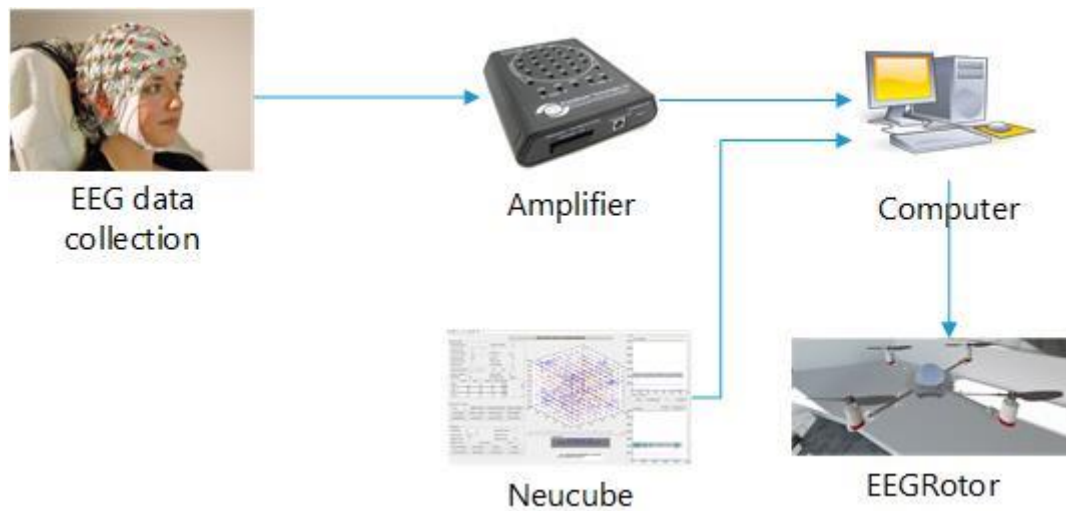


Figure 3.1: BCI overview

3.3 Functionality of BCI

BCI techniques involve recording EEG brain data and classifying it. The following steps are performed (J. R. Wolpaw, 2014):

- Human subject preparation;
- Data acquisition;
- Feature extraction;
- Model activation;
- Feedback.

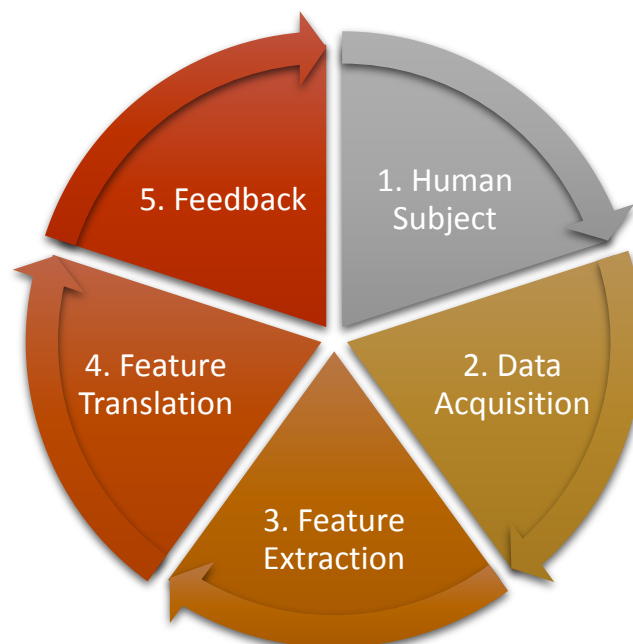


Figure 3.2: BCI Cycle.

3.4 Data Acquisition

A human brain data can be collected in three ways (Naik, 2014): invasive, partially invasive and non-invasive.

3.4.1 Invasive:

Invasive approach toward BCI involves neurosurgeons and computer scientists to work together. This is because the electrodes have to be implanted inside the subdural space (Niels Birbaumer et al., 2006) (Naik, 2014). This makes the EEG acquisition more clear and reduces the noise levels. Figure 3.3 shows an example of invasive technique.

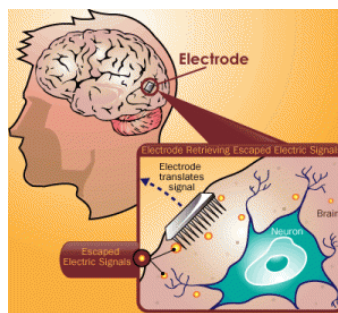


Figure 3.3: Invasive electrodes

3.4.2 Partially Invasive:

In this type of data collection, like the Invasive technique there would be a surgical procedure to implant the electrodes, but unlike the invasive technique the electrodes are placed on top of the brain just underneath the skull. This placement would give clearer signals and the advantage of this is that there would be no scarred tissue, unlike Invasive techniques (Roebuck, 2012). Figure 3.4 shows an example of partially invasive technique.

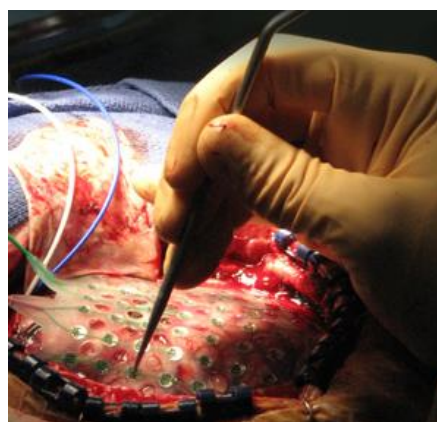


Figure 3.4: Partially invasive electrodes

3.4.3 Non-Invasive:

Non-invasive techniques are most commonly used nowadays. This type of method consists of different types of data acquisition techniques. The most common way of acquiring data is to use Electroencephalography (EEG), which uses contacts on top of the scalp rather than having any surgical procedure to place them (Tsihrintzis & Virvou, 2010). Figure 3.5 shows non-invasive EEG technique.



Figure 3.5: Non-Invasive EEG data collection

3.5 Noise Reduction and Feature Extraction

Feature extraction can be defined as a filtering process where the difference of relevant content can be extracted from irrelevant content and then representing the data in a more meaningful manner. The Noise occurs usually in the non-invasive form of EEG data acquisition method, which involves the electrode to be placed on the scalp. Any slight movement of the electrodes can cause noise. High-pass and low-pass filters are examples of noise filter, which are capable of removing noise in a raw EEG data. The measurement of signal to noise is done by Signal to Noise ratio. The higher the SNR ratio, the lower the signals with noise.

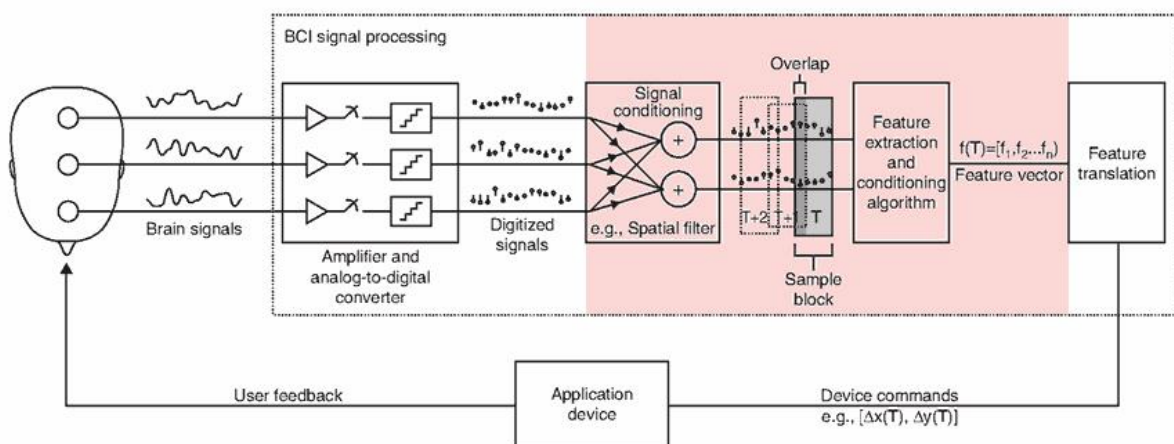


Figure 3.6: Feature extraction (J. Wolpaw & Wolpaw, 2011)

In Figure 3.6, we can see the shaded part where the feature extraction happens. The process first starts with the analogue signals from the EEG device which are sent to the amplifier where the signals are amplified so that the processing of the signals is much easier to handle. Then these signals are converted to digital signals which are passed on to “signal condition”, where the signals are enhanced and/or the unwanted signals are removed. Finally the filtered signals are sent to “feature extraction and conditioning algorithm” where the features are extracted from one or more channels to produce feature vectors which are then sent to feature translation (about this section 3.6) (J. Wolpaw & Wolpaw, 2011).

There are three steps for feature extraction:

- Making sure the input signals are without any noise;
- Extracting the features from the filtered signals;
- Finally, preparing the feature vector for the final stage of feature translation.

3.6 Feature Translation and Model Creation

The feature extraction stage is successful in extracting the features, but those features are not readable by the user. They must be translated to a different scale that would be easily interpreted. This is achieved with a translation algorithm. Feature translation consists of mathematical equations which convert the input feature vector into an appropriate command for the application to read. In some cases, there might be a situation where the input might be a different level, where it might have been converted to a higher dimension (for example, a robot hand which has multiple axes of division in a three dimensional space). The input data may have few observations with few features or many observations with many features. In any of these cases, the final goal is to show the difference between the features and users intent. A simple equation as (3.1) can be used as a feature translation method.

$$Y = mX + c \tag{3.1}$$

The difference between X and Y can be defined by the above linear equation, where m is the slope and c is the Y intercept. If the above equation is used as a BCI model, X

would be the features and Y would be the output command (J. Wolpaw & Wolpaw, 2011).

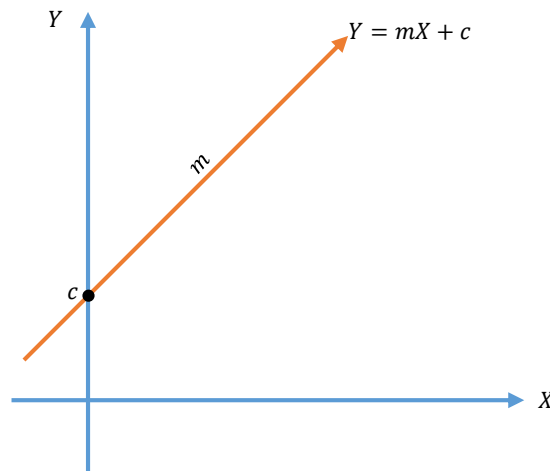


Figure 3.7: Line equation

3.7 Feedback Generation

This can be achieved through data classification. The system is developed to classify which input data belongs to which class with the help of pattern recognition algorithm (Kittler, 1997). Figure 3.8 shows a simple example of classification problem where the data from EEG device is collected and the classifier classifies as to where that particular input data belongs. To get the correct classification we have to follow a few steps. The first step is to train the data, for example if an EEG data; contains three movements (wrist up, down and rest) then for each movement we have class labels one, two and three respectively.

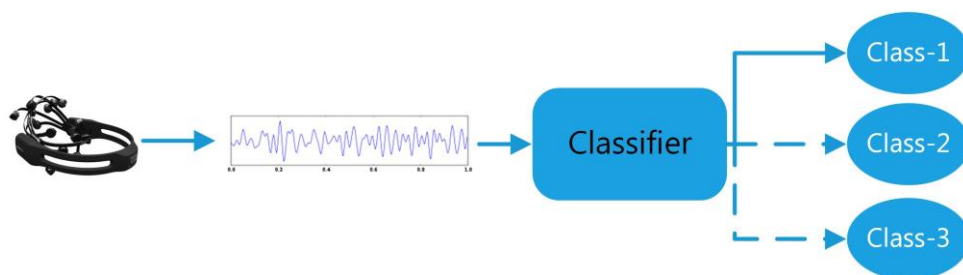


Figure 3.8: Data classification; the dotted lines represents non-classified classes.

These collected data with their class labels are sent to the classifier for training, where the pattern recognition software learns the pattern flow. The next step is to validate the classifier with new data (external data) that follows the same process as in Figure 3.8. The classifier algorithm (pattern recognition algorithm) checks the pattern to see if it

matches any data which it was already trained on. If there is a match, it would assign the data to the matched class. Otherwise, if there is no matched pattern it would give the nearest matched class as the output.

There are many types of classification algorithms, but the popular ones used are the following (Lotte, Congedo, Lécuyer, & Lamarche, 2007):

- Regression techniques;
- Support Vector Machine (SVM);
- Linear Discriminant Analysis (LDA);
- Multilayer Perceptron (MLP);
- Dynamic Evolving Spiking Neural Network (DeSNN) (N. Kasabov, Dhoble, Nuntalid, & Indiveri, 2013a) ;

3.8 Biofeedback

In the 1960's neurophysiologist Barbara Brown and neuropsychiatrist Joseph Kamiya, developed a electrophysiological device that was used by physiologist to get feedback from the patient on their mental state ("biofeedback," 2014). This device helps in monitoring the stress level, hypertension and other health related problems. This also helps the patients to overcome these problems (Martin & McFerran). According to Norris (1995) it was seen that Neurofeedback (A device for brain feedback) was able to help children with ADHD to control their behaviour.

3.9 Developments in BCI

3.9.1 Control Signals

These are also known as evoked potential or Event Related Potential (ERP), and can be defined as a controlled stimulus of electrical activity when the subject performs a stimulating task ("evoked potential," 2009). There are a few different types of ERP signals (Schalk & Leuthardt, 2011):

- Visual Evoked Potential (VEP);
- Steady-State Visual Evoked Potential (SSVEP);
- P300 Response;
- Motor Evoked Potential (MEP).

3.9.1.1 Visual Evoked Potential (VEP):

This is a type of electrical potential that happens when a subject performs task which involves visual performance. The electrodes are placed on the visual cortex, as it is the place where visual processing takes place. It was seen that there was P100 response when the subject performed a task (Creel, 2012). An example of this is to control television (TV) with these signals (Y. Wang, Gao, Hong, Jia, & Gao, 2008).

3.9.1.2 Steady-State Visual Evoked Potential (SSVEP):

SSVEP is a type of VEP, when the subject is subjected to certain levels of light frequency. A good amount of positive spikes were observed when the subjects were subjected to 8 – 30 Hz of flickering light. The event potential was observed at 80 – 100 ms (Morgan, Hansen, & Hillyard, 1996). An example of this is to control a robot (Prueckl & Guger, 2009).

3.9.1.3 P300 Response:

When a user does any visual task, the visual stimulus triggers brain signal after 300 ms (M. Wang et al., 2015). The best example of this is the P300 spelling software which was developed by Farwell and Donchin (1988).

3.9.1.4 Motor Evoked Potential (MEP):

To evoke this potential there must be an external stimulation to the central nervous system or the muscles, so that a positive spike can be observed ("Motor Evoked Potential (MEP)," 2009).

3.9.2 Different types of BCI

According to Graimann, Allison, and Pfurtscheller (2010) BCI could operate in one of the following two modes:

- **Synchronous or cue-paced BCI Systems:** This type of BCI systems are used when the subject has only fixed time response. For example, let us consider a BCI system where the system tells the subject when to do a task with a beep sound. After this sound is heard the subject has about 2 to 6 seconds for the task to be completed. When the task is completed the BCI classifies the data after a limited time frame and gives the recognised command to the application. These types of systems are easy to set up as the instruction on when to start the task is predetermined by the system.

- **Asynchronous or self-paced BCI Systems:** These types of BCI systems have more technically demanding implementation as they require continuous processing of data received from the subject. The subjects do not have to listen to what the system tells them to do; rather the subjects may or may not have to send signals to the system. This is more technically demanding because of its continuous computational process of the incoming signals and classifying them in real time.

Furthermore, the BCI can also be classified into two types in accordance with its input type (Roberto Hornero, 2012):

- **Exogenous:** These types of systems depend of external stimulation to the subject. The best examples of these types of systems are P300 based spelling applications or SSVEP systems. These types of systems don't need any training as the systems takes care of stimulating the subject either visually or by a device which can stimulate the brain directly.
- **Endogenous:** This type is the opposite of exogenous systems. The user should be able to control his/her ability to produce good quality signal, for example frequencies like Sensorimotor Rhythms or Slow Cortical Potentials (SCP). These types of systems need vigorous training by the user so that the task can be performed better.

3.9.3 Current applications of BCI systems

BCI can be invasive, partially invasive or non-invasive. Each of them have their own usage. These systems are used in three main fields (Figure 3.9):

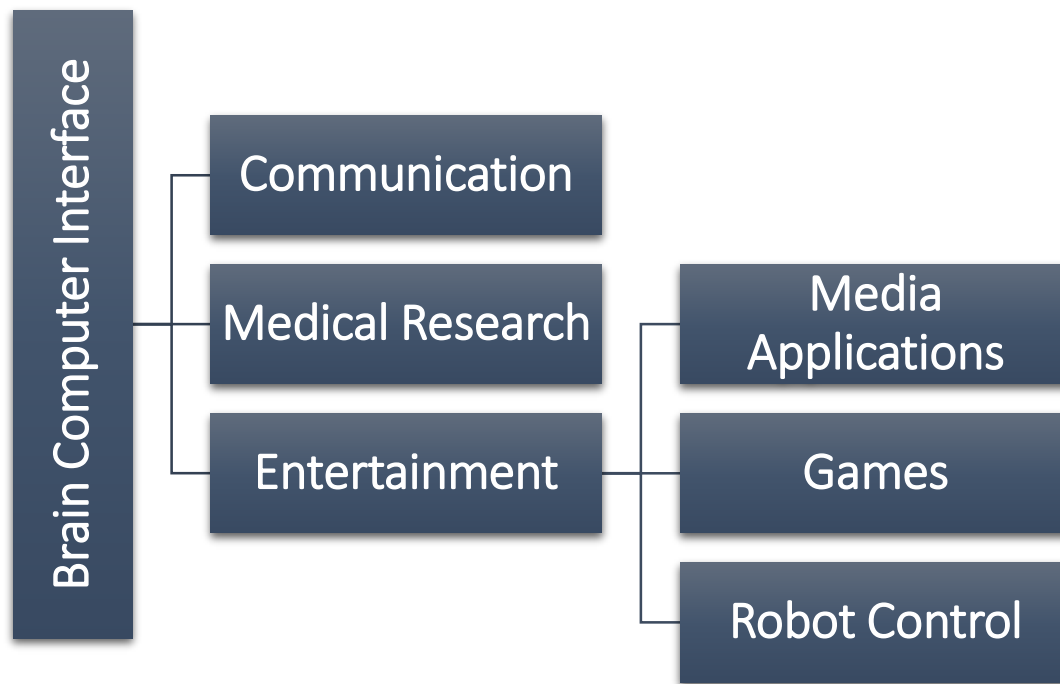


Figure 3.9: Applications of BCI

3.9.4 BCI in communication

The main usage for BCI communication is to help paralysed patients. As the technology has evolved, the increase in the BCI technology has also evolved. In 1990, a group of researchers developed a software which was able to take signals from the brain and control a mouse cursor. The subject on whom this experiment was carried out was given intense training (J. R. Wolpaw, McFarland, Neat, & Forneris, 1991). A spelling software based on BCI was developed by N. Birbaumer et al. (1999) who were able to implement a better way of applying the BCI techniques using slow cortical potentials. Guenther et al. (2009) developed an invasive BCI speech synthesis device, which was able to take signals from the brain and use them to control a speech synthesiser. The subject on whom this was tested on was affected by locked-in syndrome.

3.9.5 BCI in medical research

Ruiz et al. (2011) used fMRI-BCI based system and were able to enhance the neuron-to-neuron connectivity and behaviour of a group of participants who were divided into three groups. Their age ranged from 18 to 30. All participants were trained to control their region of interest (ROI) by using visual feedback. The first group were trained to control the coupling between Inferior Frontal Gyrus (IFG) and Superior Temporal Gyrus (STG). The second group were given feedback on Blood-Oxygen-Level

Dependent (BOLD) using only IFG and finally, the third group were used as a control group, who received sham feedback (making something to believe it was true; but in reality its false). The participants were tested on giving a word which they have to describe as real or abstract and the stimulation was Stimulus-onset asynchrony. It was seen that the first and second group were able to learn and control their IFG, STG and BOLD. But the third group of participants failed to control IFG and IFG.

3.9.6 BCI in Entertainment

BCI can be used for different purposes as shown in Figure 3.9. All applications are based on serious research-oriented experiments that are meant to help people who have mental and/or physical disability.

BCI for entertainment can be divided into different categorises shown in Figure 3.10:



Figure 3.10: BCI in Entertainment

3.9.6.1 BCI in Games

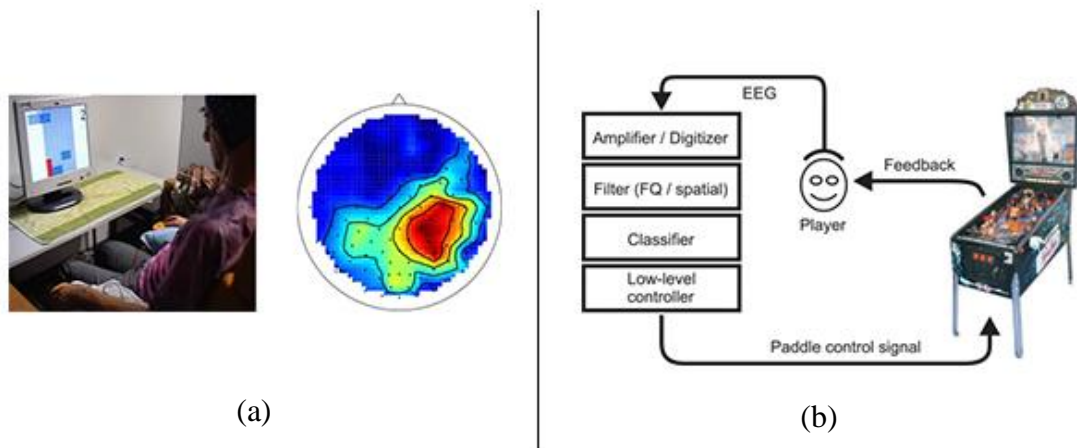


Figure 3.11: (left to right) (a) A BCI based game where the subject plays Tetris using EEG signals. (b) A BCI based foosball which uses motor to control the shooting of the ball (Blankertz et al., 2010).

In **Figure 3.11(a)**; the subject is seen playing a game using brain signals with the help of non-invasive EEG device. The game has four classes, representing is the four direction of movement and rotation of a Tetris game object. The experiment is based on three MEP values which were used to move the game object left, right & down and used mental rotation to rotate a game object. Likewise, another game was developed, which controls the motor in a foosball game; it has two classes, for MEP values that are used to control the movement of the handle and hitting it. There were other games, for example the BrainBall game which was developed to monitor the level of relaxation using alpha waves from the occipital lobe. Another interesting BCI game is BCI-PacMan, which uses motor cortex signals to move the object in the gaming environment (Blankertz et al., 2010).

3.9.6.2 BCI in Robot Control

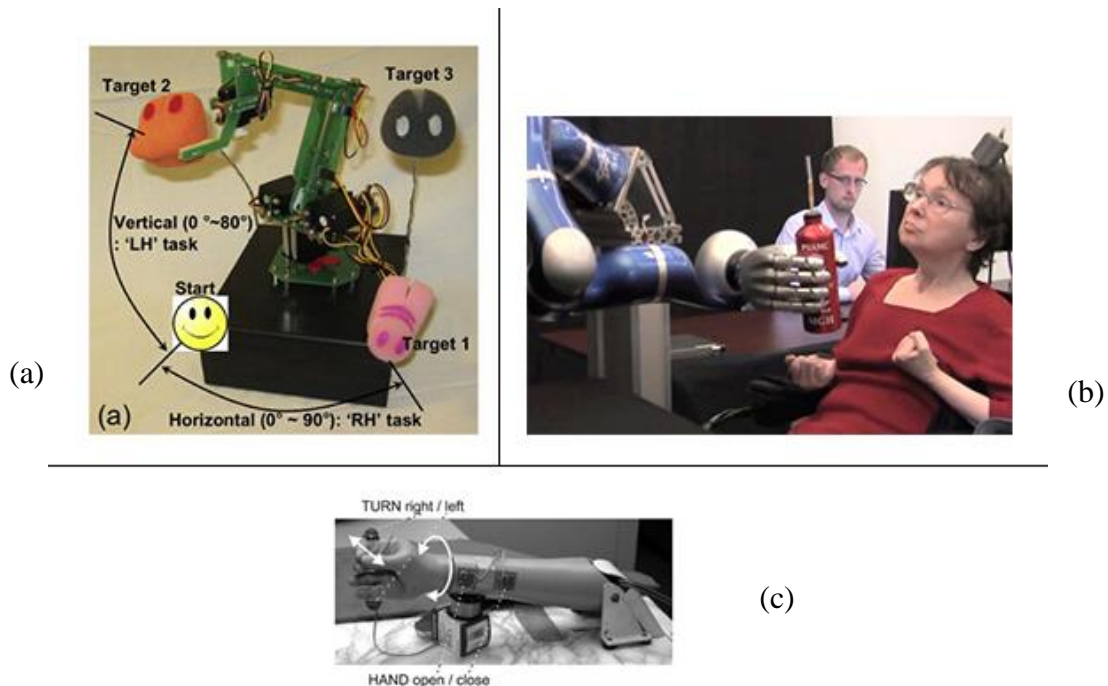


Figure 3.12: (a) A robot hand movement based on fMRI-BCI. (b) Quadriplegic women lifting a bottle with Invasive-BCI. (c) An EEG based BCI system which is capable of moving the prosthetic hand

Figure 3.12(a) is a robotic machine that was developed by Lee, Ryu, Jolesz, Cho, and Yoo (2009) and is based on real-time fMRI BCI system. They used this system to track the BOLD in the motor cortex during the hand-motor imaginary task. There were three right handed participants. The participants were told to imagine clenching their hands as realistically as possible. The task for the participants was to move the robotic arm to touch the three objects; the robot had two degrees of freedom which would move horizontally and vertically (Lee et al., 2009). **Figure 3.12(b)** shows a woman aged in her early 60's who was affected by quadriplegia and had been paralysed for 15 years. She was able to move a robotic arm. By using BCI the researchers collected signals from her motor cortex by using a partially invasive method as the means for data collection (Duncan, 2012). Finally, **Figure 3.12(c)** shows a prosthetic hand using BCI methods and is based on steady-state visual evoked potentials (SSVEP).

3.9.6.3 BCI in Media Application

Media applications such as photo gallery, music applications, browsing internet and many other forms of applications are an essential part of daily lives. Paralysed people too would like to have access to the same kind media as any other person has; so, authors Teo et al. (2006) developed a media centre for paralysed people based on P300 ERP.

This application has four modules: GUI Module – which provides the interface with the user; Control Panel Module – exchange the application controls between external sources; Media Modules – this includes MP3 players, photo gallery & other media applications; and P300 Module – This is where the classification of the input data takes place. **Figure 3.13** shows the front end GUI.

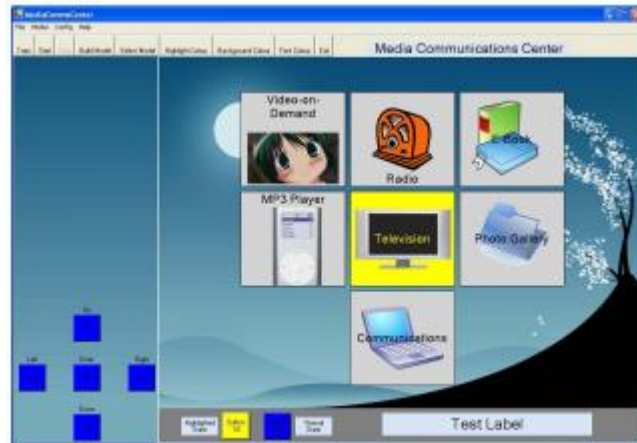


Figure 3.13: GUI for the Media Communication Centre application based on P300 ERP

3.10 An introduction to Quadcopters

Quadcopters, also called Quadrotor, is a radio controlled unmanned flying object that are like helicopters that can vertical take-off and land (VTOL). In 1921 the first full-scaled manned quadcopter was developed by De Bothezat (Altug, Ostrowski, & Mahony, 2002). Due to the unstable flying and expensive maintenance of helicopters, Quadcopters have been developed to increase stability (Sa & Corke, 2011).

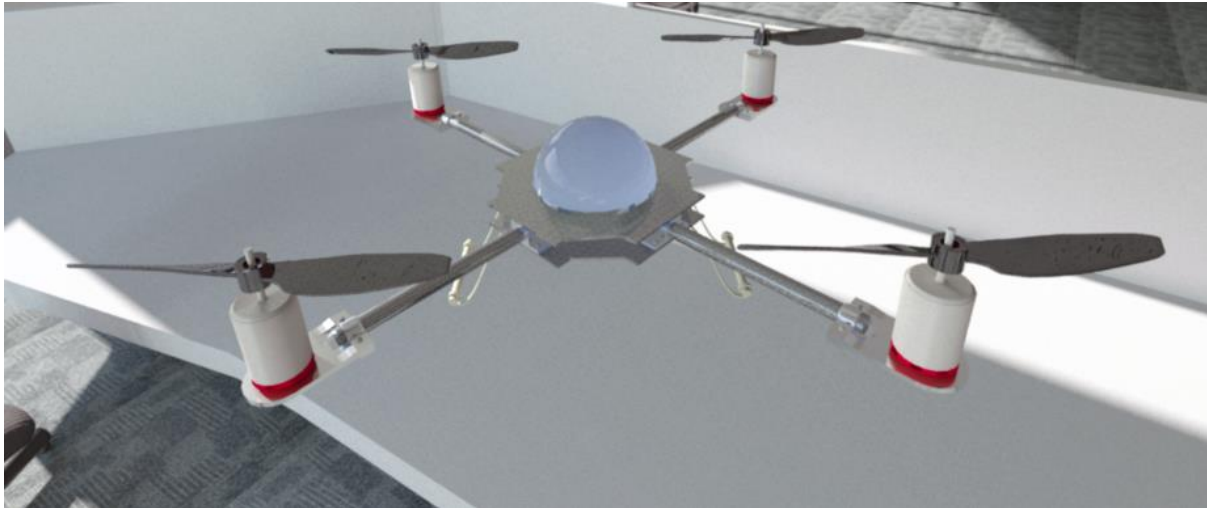


Figure 3.14: Model of Quadcopter

As seen in **Figure 3.14**, a Quadcopter has four fins, all of which turn at equal speed when hovering; in each pair of opposite rotors they rotate in the same direction clockwise (CW) and the other two rotate in counter clockwise (CCW) (Allen, 2014). If we want the Quadcopter to go forward, both motors in the back would increase their rotation speed and the front two motors would decrease their speed; this increase and decrease in speed would make the object go forward, backward, left and right. When it comes to VTOL, all the four motors rotate at the same rotations per minute (RPM) (Pounds, Mahony, & Corke, 2010).

Every Quadcopter has something called as Flight Controller (FC), to stabilize the Quadcopter while flying. Every flight controller has the following main components ("HobbyKing HKPilot Mega Mini Combo Flight Controller GPS and Power Module,")

- *3-Axis Gyro*: It is a micro-electro mechanical system (MEMS) that is used to measure the orientation (Turner, 2015).
- *Accelerometer*: It measures the acceleration or deceleration of the flight digitally ("Accelerometer," 2014).
- *Magnetometer*: It is an instrument used for measuring the direction or the intensity in the magnetic field. Example digital compass (Bankman, 2000).
- *Barometer*: It measures the atmospheric pressure ("barometer," 2013).

All components together with the remote control for manoeuvring the quadcopter, makes sure that the flight is stable and does not go out of control. Optionally, the user may also be able to attach a GPS device to the Quadcopter, so that the user can have its exact geographical position of it. This would help the user to use autopilot and use “Go Home” feature.

Another main important feature in quadcopters is their Brushless Direct Current (BLDC) type of motor. The reason for using these motors is that they have high RPM and torque (τ), which can be used to lift heavy loads; the RPM of these motors are controlled by Pulse Width Modulator (PWM) (Moseler & Isermann, 1998).

3.11 Current limitation of BCI

This section discusses the problems involved in the current implementation of BCI techniques such as the cost of equipment, data acquisition, human trials and the speed of processing.

Commercial use of EEG devices is limited to small number of channels. For example to collect EEG data and filter the noise *Mynd play* is a company which uses two to three channels EEG headsets to control a media player; another company, *Focus Band* uses three dry electrodes to monitor the state of mind when asleep.

However, researchers suggest that commercial equipment like Emotiv shows degraded results when compared to a medical systems (Duvinage et al., 2012). BCI computation requires powerful systems to process continuous incoming brain signals, so that the command sent from the signals do not overlap with previous commands.

Information Transfer Rate (ITR) is used to measure the amount of bits transferred per session or per task (Obermaier, Neuper, Guger, & Pfurtscheller, 2001). A P300 BCI speller application has an ITR of 2.3 characters/minute but in a control condition the number could go up to 12.75 character/minute (P. T. Wang, King, Do, & Nenadic, 2012). This is really slow for using it in a real world implementation.

To get higher resolution of brain signals it is considered that invasive-BCI techniques are better. As discussed earlier, it would involve a surgical procedure and the patient might react to the electrodes and develop scar tissues, which is not recommend. Another way of extracting higher resolution images or signals is using fMRI, but the problem is that the devices to acquire this level of resolution are very expensive. If someone did find the means to acquire such a system, the data acquisition can only be done by an expert. Due to this problem the data acquisition is done in one place and testing and validation is done in another place. This would mean that only static data can be collected but not continuous (or online) data.

Finally, the last problem is the training of subjects. For example, SCP based BCI would need extensive training sessions before the subject is capable of using the BCI application

(Niels Birbaumer, 2006). This training makes sure that the subject is capable of using the machine and the application; else there are chances of getting false positive errors.

Although there are many commercial and cheap EEG devices, they are not suitable for doing intensive research, as they are limited to three to 14 (Emotiv EPOC) channels. This equipment is used for small applications like gaming (Nijholt, Bos, & Reuderink, 2009).

3.12 Overview of Methods used in BCI

As discussed earlier, there are different types of EEG data acquisition tools which can be medical equipment or commercial equipment. This chapter focuses on such equipment where the data acquisition and the BCI classification algorithm is considered. For any BCI to work correctly, the researchers need to have all the necessary equipment, the skills to perform and train the person with the task that he/she has to perform. The following are the equipment and computational knowledge (algorithms) that an expert needs to obtain before he/she starts the examination (or testing a subject on BCI).

- EEG data acquisition device;
- BCI platforms;
- Feature extraction;
- Pattern recognition.

3.12.1 EEG data acquisition for BCI

There are different types of EEG data acquisition hardware ranging from a small consumer device to a professional medical EEG unit. Medical EEG devices have from four to 256 channels (a channel signifies one electrode).



Figure 3.15: 256 and 4 channel EEG devices (“Wireless EEG system/4-ch MPEG-4 compression/ambulatory”)

This equipment is expensive due to its unique functionalities such as dry electrodes and brain mapping algorithm with proprietary software. These types of equipment are usually used by experts who take detailed readings of patients who need medical attention, but these devices are not for regular and daily use.

As the technologies evolve the cost to develop and commercialise EEG devices increases. There are many devices which can be used by commercial users and researchers. Two companies well known for commercial EEG devices are Emotiv and Neurosky.



Figure 3.16: Emotiv EPOC and Neurosky MindWave

Emotiv EPOC is a 14 channel EEG device running at 128 Hz SPS (Samples per Second) (internal sampling at 2048 Hz) and costs around USD \$399. According to Ranky and Adamovich (2010) an experiment was conducted using a robotic arm called iARM that has seven degrees of freedom. They found out that Emotiv was able to outperform in the terms of portability, user customisation and robustness.

The next most popular type of EEG device is Neurosky MindWave that has three channels runs at 512 Hz SPS and costs around USD \$80 just for the headset. According to Yoh, Kwon, and Kim (2010) a game “Hansel and Gretel” was developed that used Neurosky MindWave. In this game the gamer was able to achieve the final result with ease. This demonstrated that even having the disadvantage of three electrodes they were able to play the game without any problem.

So the pros of using commercial equipment rather than using medical EEG devices are as follows:

- Commercial grade EEG equipment are affordable when compared to medical grade EEG equipment;
- These systems follow the International 10-20 system, which is standardised to all medical equipment as well;

- No expert handling is required to use these systems;
- Quality of collected EEG data can be compared to data collected with medical EEG devices.

3.12.2 Different types of BCI platforms

A BCI platform is a software median between the EEG hardware (and software) & the command you want to convert it into. It is also capable of giving neuro feedback to the user. There are different kinds of BCI applications. In this study, the author will be using “NeuCube architecture for Brain Computer Interface”, more on this in Chapter 5:. That being said, BCI application are of two main types; one which can run online data (dynamic streams of data) and the second one is offline data (static data); keeping that in mind there are different types of BCI applications are capable of working with both online and offline data; the most popular are OpenVibe, BrainBay, BCI2000, TOBI and BCILAB.

- **OpenVibe:** One of the most widely used open source BCI application software currently available. This software is used for its multiple platform support and different types of research can be conducted with it, including BCI, virtual reality and many more. OpenVibe is easy to use, modular and portable (Renard et al., 2010). Though this is one of the widely used BCI application, the main problem is that it is mostly used for VEP. But in this study we need to deal with signals from the motor cortex, not from the visual cortex.
- **BCI2000:** This is the second most widely used EEG acquisition and signal processing software. The communication on this application is based on TCP/IP unlike OpenVibe which uses serial port or USB ports (though this can be converted to TCP/IP communication by implementing your own module). BCI2000 runs four modules: operator (visualisations and EEG control), source storage, signal processing and user application (commands to application). This software also supports real time and offline data analysis (Schalk, McFarland, Hinterberger, Birbaumer, & Wolpaw, 2004).
- **BrainBay:** It’s a bio and neuro-feedback application which is designed to work with OpenEEG hardware. This software also supports HCI and NeuroServer capabilities that can be used to transmit live/online data via TCP/IP. The main difference between this and other BCI applications is that this software was mainly developed for OpenEEG hardware like ModularEEG devices. It can handle its software implementation easily (Veigl). Though this is a free application, when trying to implement it with the existing

hardware such as Emotiv and Neurosky we have to develop plugins to make sure this software interacts with the BrainBay toolbox.

- **BCILAB:** It is primarily a Matlab toolbox for BCI. This was designed to conduct field tests of general medical BCI systems. It is capable of visualising EEG data and can extract the state of mind (*BCILAB*, 2013). Though this is free software, implementing this in our study is not possible because of its limitation in classifying data and there are very few visualising modules associated with it.

The sole purpose of all BCI platforms is to take the EEG data, classify it and visualize it. The discussed software/toolbox do not have feature extraction and data classification. So keeping these problems in mind, a new framework for BCI is proposed NeuCube: For Brain Computer Interface (BCI). NeuCube can be said as an all-in-one application. More on this in Chapter 5:.

3.13 Conclusion

This chapter discusses different types of BCI and how important and useful they can be and also their limitations. The main principles of BCI are used in a further chapter to develop new BCI framework and a working system for the control of a Quadcopter.

Chapter 4: Some Methods for Data Transformation Pattern Recognition and Machine Learning for BCI

This chapter provides some introduction to the working of different algorithms that are used in this study.

NeuCube uses different types of algorithms to encode, learn and classify the data. The family of Artificial Neural Network are broadly made up of the following networks

- Neural Network;
- Spiking Neural Network;
- Evolving Spiking Neural Network;
- Dynamic Evolving Spiking Neural Network.

The above types of networks are discussed in more detail further below.

The following algorithms are also discussed:

- Butterworth algorithm;
- Fast Fourier Transformation.

And finally the chapter describes how Threshold based encoding works.

4.1 Some pattern recognition algorithms

There are different types of algorithms that can be used as classifiers for EEG data. Such as C4.5 and K-Nearest Neighbours. All these algorithms are based on supervised learning.

- **K-Nearest Neighbours:**

When a new data is given to this algorithm, it searches for k nearest elements available in the training set that is similar to the new one.

4.2 Introduction to Neural Networks

In 1945, two neuroscientists namely Warren McCulloch and Walter Pitts developed a simple neuron known as artificial neuron. This type of neuron depends on threshold. The input or output from artificial neuron can be true/false or 1/0, this is known as binary function. (James

A Anderson & Rosenfeld, 1993). For example, a truth table of Inclusive OR function is considered, where the threshold $\theta = 1$. Then we get from the **Figure 4.1**

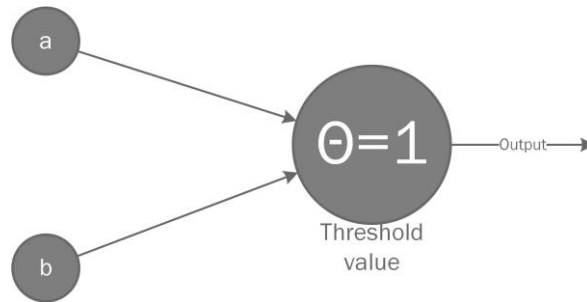


Figure 4.1: Two state neuron

a	b	Output
0	0	0
1	0	1
0	1	1
1	1	1

Table 4.1: Logical OR

Let a and b represent synapses and indicate active state and inactive state at the time $(i + 1)$. At $(i + 1)^{th}$ time, if a is active and b is not active then the neuron output state is active, since active synapsis plus inactive synapsis is always active. Hence, it follows the logical OR truth Table 4.1. The same can be applied to logical AND at the $(i + 2)^{th}$ time quantum (James A Anderson & Rosenfeld, 1993).

Based on the findings of McCulloch-Pitts neuron, Hebb (2005) stated the following

“When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.”

This means that the connection between pre-synaptic and post-synaptic potential depends on the influence (firing) of pre-synaptic neuron that will fire the post-synaptic neuron; the more the firing, stronger are the connection weight (strength in connection between neuron to neuron). This theory is known as Hebb's Law, which is a learning rule that is given as follows (Hebb, 2005):

Let there be a k -dimension column vectors:

$$\vec{x} = [x_1, x_2, \dots, x_k]^T$$

Neurons with random weight between -1 to 1:

$$\vec{w} = [w_1, w_2, \dots, w_k]^T$$

And the output for the neuron is given by:

$$y = \vec{w}^t \vec{x} = \sum_{i=1}^k w_i x_i$$

After the input pattern is given Hebb's rule is applied, which is given by:

$$\Delta \vec{w} = \eta \vec{x} y \quad (4.1)$$

(4.1) shows the change in synaptic weight $\Delta \vec{w}$ is equal to the learning rate η times the input \vec{x} times the postsynaptic neuron y .

Where η is the fixed learning rate

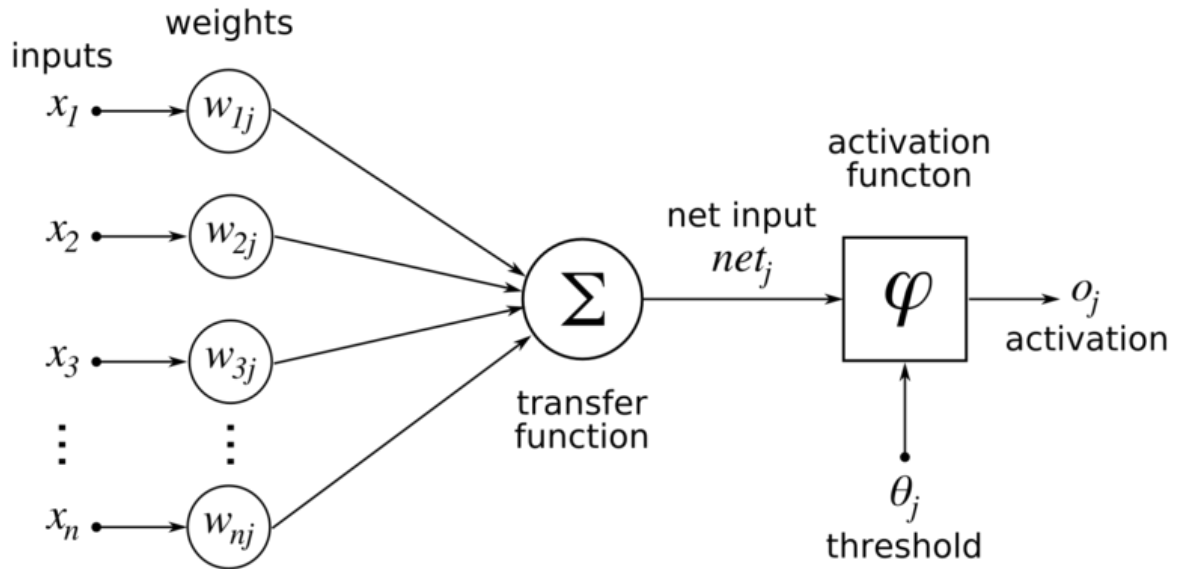


Figure 4.2: Model of a neuron (Turner, 2015)

4.2.1 Perceptron Model

In 1958, Frank Rosenblatt who was a psychologist, proposed a model called the Perceptron. This was one of the first pattern recognition algorithm ever created. Based on this perceptron model are some more advanced models, which are able to learn data easily and fast. The main component of perceptron model is a component called Threshold Logic Unit (TLU). All previous neural networks used simple linear elements whereas the perceptron model uses non-linear elements (James A. Anderson, 1995).

The perceptron model is illustrated in **Figure 4.3**. The threshold function is expressed as follows:

$$TLU\ output = +1\ if\ \sum w[i]x[i] > \theta,$$

$$TLU\ output = -1\ if\ \sum w[i]x[i] \leq \theta.$$

The difference between the previous algorithms and the perceptron model is the introduction of the so-called as *bias*. Which can be expressed as follows

$$y = w_0 + \sum_{i=1}^n w_i x_i$$

Also, the perceptron model defines a hyper plane in $n + 1$ dimensions:

$$y = w_n x_n + w_{n-1} x_{n-1} + \dots + w_1 x_1 + w_0$$

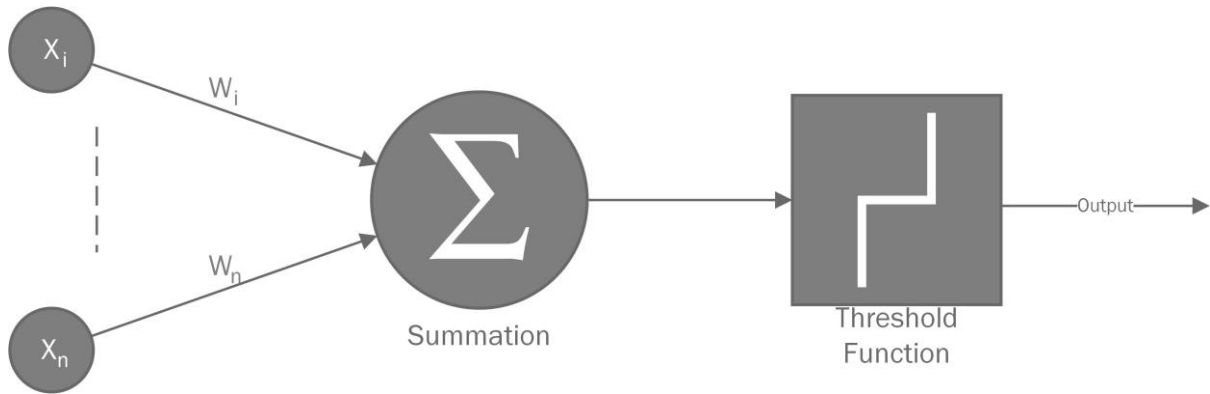


Figure 4.3: Perceptron model

4.2.2 Multilayer Perceptron

As the name suggests, this model has few layers of neurons: input layer, hidden layer and output layer. The input layer and the output layer has no direct connection to each other. Similar to perceptron model, the data flow in this network is feed-forward. Unlike the perceptron model, the hidden layer and the output layer do not have a step function, rather they have sigmoid functions. The non-linear transformation functions and the hidden layers, in this model can solve nonlinear problems, whereas the perceptron model cannot. Figure 4.4 illustrates a model of multi-layer perceptron.

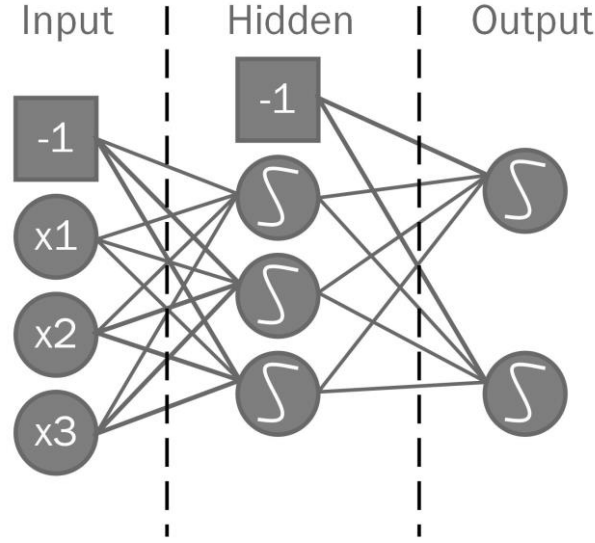


Figure 4.4: Multi-layer perceptron

The output of the multilayer perceptron is always between 0 and 1, unlike the step function where the output is either 1 or 0.

4.3 Spiking Neural Networks

As we have discussed earlier, the brain is a complex organ that is capable of making decisions. We are what our brain is. Brain is a collection of neurons, which send signals to each other in form of spikes. Using the principle of biological neuron, a 3rd generation of neural networks was developed. This artificial spiking neuron model is known as the Spiking Neural Networks (SNN).

4.3.1 Hodgkin-Huxley model

Hodgkin & Huxley (1952) are known as the fathers of spiking neurons. They developed a model that correlated with the electro-chemical information transmission of a neuron. The model used electrical circuits that had capacitors and resistors (Figure 4.5). In the figure, C is the capacitance of the membrane and gNa , gK and gL is the ion exchanges in the live neurons with equilibrium potentials E_{Na} , E_k and E_L ; h , m and n are the opening and closing of the voltage. $I(t)$ is the input current at time $t < 0$ (Paugam-Moisy & Bohte, 2012)

$$C \frac{du}{dt} = -gNa m^3 h (u - E_{Na}) - gk n^4 (u - E_k) - gL (u - E_L) + I(t)$$

$$\tau_n \frac{dn}{dt} = -[n - n_0(u)], \tau_m \frac{dm}{dt} = -[m - m_0(u)], \tau_h \frac{dh}{dt} = -[h - h_0(u)]$$

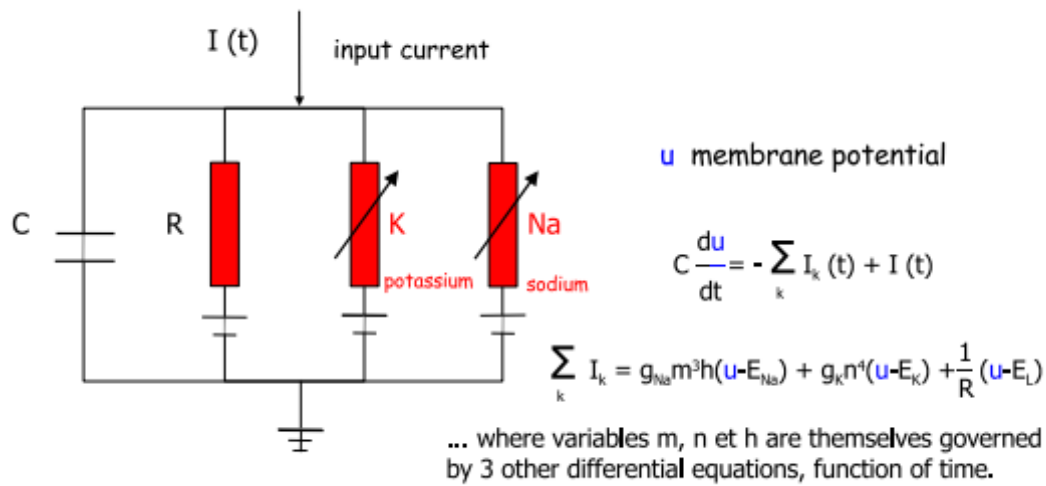
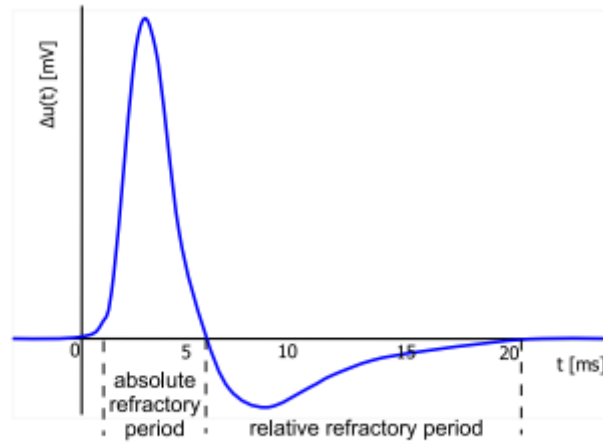


Figure 4.5: Hodgkin-Huxley model



Dynamics of spike firing

Figure 4.6: Spike pattern

A well calibrated Hodgkin-Huxley (HH) model could compare to a biological model; furthermore, the researchers were able to get a neuron like physical property; this was achieved by having a sudden increase in firing time then followed by a short period of inactiveness, as can be seen in **Figure 4.6**. However, the HH model is too complex to setup and simulate SNN.

4.3.2 Leaky Integrate and Fire

Before Leaky Integrate and File (LIF) was developed, Integrate and Fire (I&F) neurons were proposed. They were known to be computationally faster. The equation for L&I is given as follows

$$C \frac{du}{dt} = -\frac{1}{R}(u(t) - u_{rest}) + I(t)$$

Where, u is the membrane potential.

The spike firing time $t^{(f)}$ is given by

$$u(t^{(f)}) = \vartheta \text{ with } u'(t^{(f)}) > 0 \quad (4.2)$$

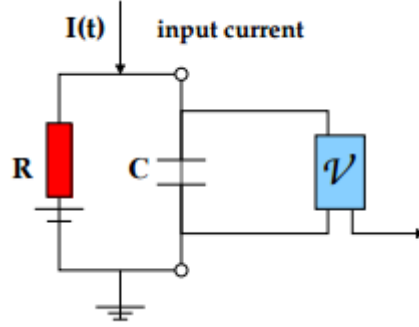


Figure 4.7: I&F model

The most used type of I&F neuron is called the Leaky Integrate and Fire (LIF) neuron model. The main difference between LIF and HH model is that in the LIF neuron every spike is a uniform event, which is defined only by the time when it appears and the shape of the action potential is ignored (Paugam-Moisy & Bohte, 2012). In this model a Resistor R is connected in parallel to Capacitor C as shown in the Figure 4.7.

The membrane potential for LIF is described with first-order differential equation:

$$\tau_m \frac{du}{dt} = u_{rest} - u(t) + RI(t)$$

Where, $\tau_m = RC$ is the time constant for neuron membrane, which gives the voltage leakage, using the threshold value, see (4.2). Once a spike has reached its maximum threshold, u_{rest} is set to 0. This is shown in Figure 4.8.

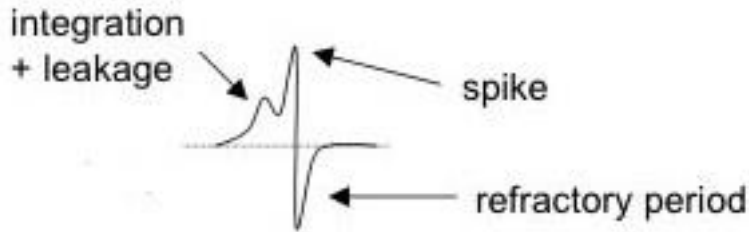


Figure 4.8: LIF spikes

4.4 Evolving Spiking Neural Network (eSNN)

In this type of network, the classifier learns and evolves at every iteration. The eSNN architecture was first proposed in a paper written by Wysoski, Benuskova, and Kasabov (2006) where it was used as a pattern recognition for visual systems. This model has been evolved from Thorpe model that stresses on the importance of early spikes, which boost and affect the post-synaptic potentials. The eSNN uses supervised one-pass learning algorithm to classify the output. Like any other spiking neural network, this algorithm also uses encoding methods to convert the input data into spike trains. Rank order population encoding is used in eSNN. Also, eSNN uses feed forward as its main topology (N. Kasabov, 2013)

4.4.1 Rank order population encoding

Rank order population is an extended version of rank order encoding algorithm; like any other encoding methodology this algorithm's main feature is to take the input data and convert them into spikes (N. Kasabov, 2013).

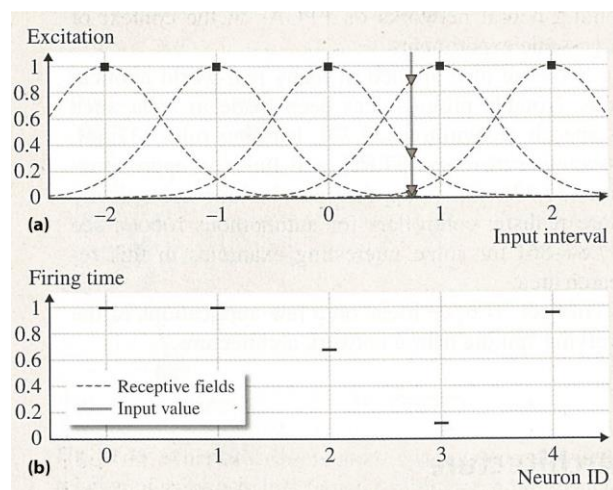


Figure 4.9: A population encoding based on Gaussian receptive fields. (a) For input value $\vartheta = 0.75$ (vertical thick line) and intersection points of each Gaussian is computed (inverted triangles). (b) Points are converted into spike time delays (N. Kasabov, 2013)

A receptive field is an area where stimulation leads to response of a neuron (Krantz, 1999). Figure 4.9 shows an implementation of receptive field using Gaussian function. Variable n of intervals as $[I_{min}^n, I_{max}^n]$ is defined. Receptive field of neuron i with respect to Gaussian function is given by its centre μ_i as

$$\mu_i = I_{min}^n + \frac{2i - 3}{2} \cdot \frac{I_{min}^n - I_{max}^n}{M - 2}$$

And the width input intervals σ is given by

$$\sigma = \frac{1}{\beta} \cdot \frac{I_{min}^n - I_{max}^n}{M - 2}$$

Where $1 \leq \beta \leq 2$, β controls the width of the receptive field. For Figure 4.9, $\beta = 2$, input intervals $[I_{min}^n, I_{max}^n]$ was set to $[-1.5, 1.5]$ and $M = 5$ (N. Kasabov, 2013).

4.4.2 One pass learning algorithm

Learning methods main objective is to produce output neurons with certain class label l , where $l \in L$. Where L is the number of class labels in a given data set. When set of input data is given to the network, spike trains are produced through SNN that may fire output neurons. If no neurons are fired, the classification of the input data is unknown. If there are neurons that spike, the earliest spike time of the neuron is determined. The label of this neuron is the classification output (N. Kasabov, 2013).

Require: m_l, s_l, c_l for a class label $l \in L$

```

1:   Initialize neuron repository  $R_l = \{ \}$ 
2:   for all samples  $X^{(i)}$  belongs to class  $l$  do
3:        $w_j^{(i)} \leftarrow (m_l)^{order(j)}, \forall j \mid j \text{ pre-synaptic}$ 
           neuron  $i$ 
4:        $u_{max}^{(i)} \leftarrow \sum_j w_j^{(i)} (m_l)^{order(j)}$ 
5:        $\vartheta^{(i)} \leftarrow c_l u_{max}^{(i)}$ 
6:       if  $\min(d(w^{(i)}, w^{(k)})) < s_l, w^{(k)} \in R_l$ 
           then
7:            $w^{(k)} \leftarrow \text{merge } w^{(i)} \text{ and } w^{(k)}$ 
8:            $\vartheta^{(k)} \leftarrow \text{merge } \vartheta^{(i)} \text{ and } \vartheta^{(k)}$ 
9:       else
10:           $R_l \leftarrow R_l \cup \{w^{(i)}\}$ 
11:       end if
12:   end for

```

Figure 4.10: Algorithm for training eSNN

Where, i is the training sample for class label $l \in L$. $w^{(i)}$ are the real-valued weight vector, having $w_j^{(i)} \in \mathbb{R}$ that means the connection between pre-synaptic neuron j and the newly created neuron i . Next the input spikes are sent through the network and weight $w_j^{(i)}$ is calculated according to the order of spike transmission through a synapse j . m_l is the modulation factor of Thorpe neuron model, $\vartheta^{(i)}$ is the firing threshold of the created neuron i , $u_{max}^{(i)}$ is the maximum potential of the created neuron i . c_l is the

fraction parameter of the module. Existing neuron is denoted by k and s_t is the similarity threshold.

Both firing threshold and the weight vectors are merged as

$$w_j^{(w)} \leftarrow \frac{w_j^{(i)} + Nw_j^{(k)}}{1 + N}$$

$\forall j \mid j \text{ pre-synaptic neuron of } i$

$$\vartheta_j^{(w)} \leftarrow \frac{\vartheta_j^{(i)} + N\vartheta_j^{(k)}}{1 + N}$$

Where, N denotes the number of samples previously used to update neuron k . Figure 4.10 shows the algorithm for training eSNN neuron.

4.5 Introduction to Spike-Timing Dependent Plasticity (STDP)

STDP is an unbalanced temporal form of Hebbian rule (explained in section 4.2), which has a temporal interrelationship between a pre- and a post-synaptic neuron. It is widely believed that synaptic plasticity is the cause for learning and storing information in brain. In STDP, we have two types of results for a synapsis: Long-Term Potentiation (LTP) and Long-Term Depression (LTD). LTP is said when a spike in a pre-synaptic occurs few milliseconds before a post-synaptic potential. LTD is said when a spike in a pre-synaptic occurs after a post-synaptic potential. So STDP can be defined as the relative time between pre-synaptic spike to post-synaptic potential (Sjöström & Gerstner, 2010).

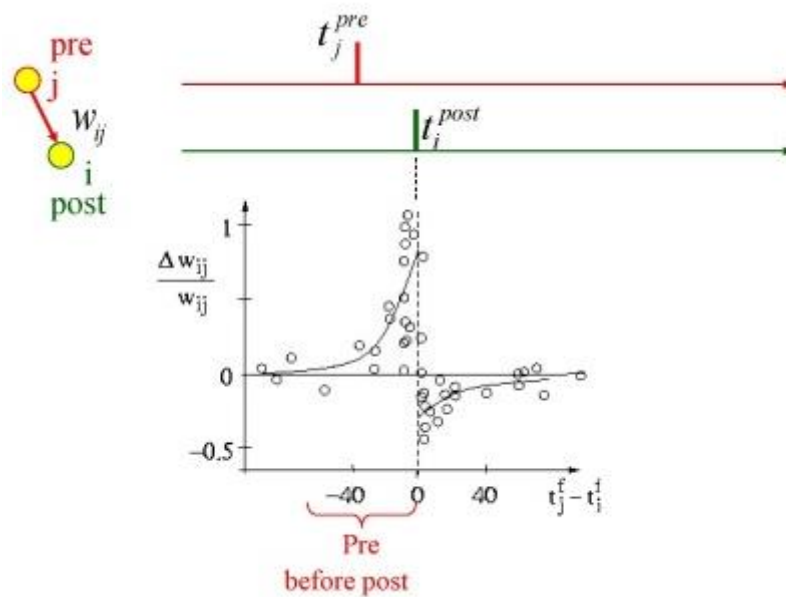


Figure 4.11: STDP with 60 spike pairing between pre- and post-synaptic potential

Let's consider, Δw_j as the change in weight of synapsis from pre-synaptic neuron j ; and lets consider the presynaptic spike arrival time for neuron j as t_j^f where $f = 1, 2, 3, \dots N$ that are the counts to pre-synaptic spikes. And let's consider the firing time for post-synaptic neuron as t^n , where $n = 1, 2, 3, \dots N$. So the weight change in the network Δw_j with a stimulation protocol (that stimulates action potential in a neuron) is given by (Sjöström & Gerstner, 2010)

$$\Delta w_j = \sum_{f=1}^N \sum_{n=1}^N W(t^n - t_j^f) \quad (4.3)$$

From (4.3) $W(x)$ (where $x = t^n - t_j^f$) is one type of STDP function, which is also known as learning window, shown in the Figure 4.11. Popular choice for SDTP function are

$$W(x) = A_+ \exp(-x/\tau_+) \text{ for } x > 0 \quad (4.4)$$

$$W(x) = -A_- \exp(x/\tau_-) \text{ for } x < 0$$

Here A_+ and A_- are the dependent current value of the weights w_j ; τ_+ and τ_- are the time constraints that are in the order of 10ms each.

4.6 Dynamic Evolving Spiking Neural Network (DeSNN)

DeSNN is an evolved version of eSNN. The problem with eSNN is that it uses rank order encoding method. The weight adjustments in this method is done only once, which is only capable of classifying static data but is not that efficient for Spatio- and Spectro- Temporal Data (SSTD). Due to this problem the weights must be tuned for each training over time and for this reason STDP was used in addition to the Rank Order (RO) learning based on the order of the incoming spikes. (see section 4.5. and N. Kasabov, Dhoble, Nuntalid, and Indiveri (2013b) for more details).

4.7 Introduction to Fourier Transform

Joseph Fourier is known as the father of modern mathematics. In 1822, he wrote the book "The Analytical Theory of Heat". In the same book he described a transformation function which is now known as "Fourier Transformations". It decomposes the time-domain function into its component frequencies (Adiutori, 2005) and is described by the following equations (Strang & Nguyen, 1996)

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{in\omega} \quad (4.5)$$

The inverse function is

$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega) e^{-in\omega} d\omega \quad (4.6)$$

Where $X(\omega)$ is the frequency domain (also known as the Fourier function) and $x(n)$ is the time domain. Where the time n is discrete (separated individually and unique) and ω is continuous.

For example, let us consider the graph in Figure 4.12

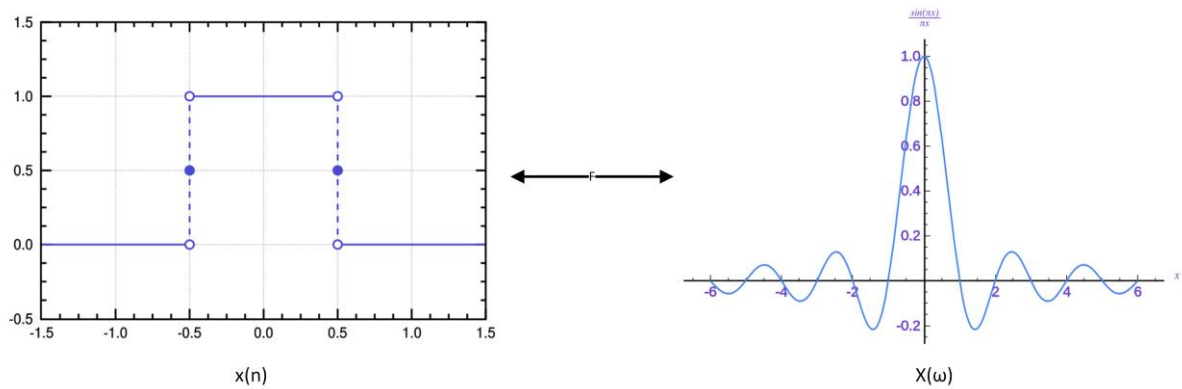


Figure 4.12: $x(n)$ is a frequency domain and $X(\omega)$ is a time domain. When $x(n)$ is sent through the Fourier it is changed into $X(\omega)$ and vice versa.

On the left side of the above figure, we have a rectangle function which is in the form of time, when applied Fourier transform on it, we get frequency pattern that is on the right side of the above figure.

4.7.1 Discrete Fourier Transform

Discrete Fourier Transform (DFT) represents regular data points in any given data and gives its strengths in periodic components. However, DFT of real numbers will be a complex number of same length; this causes two types of errors: aliasing and leakage that can be addressed by using **Fast-Fourier Transform** (Weissstein).

Let us consider a continuous Fourier transform as

$$f(v) = \mathcal{F}[f(t)](v) = \int_{-\infty}^{\infty} f(t) e^{-2\pi i v t} dt$$

Now, generalise to discrete function $f(t) \rightarrow f(t_k)$ by $f_k \equiv f(t_k)$ where $t_k \equiv k\Delta$ and where $\Delta = 0, \dots, N-1$. This will give the DFT as $F_n = \mathcal{F}_k[\{f_k\}_{k=0}^{N-1}](n)$

$$F_n \equiv \sum_{k=0}^{N-1} f_k e^{-2\pi i n k / N} \quad (4.7)$$

And the inverse of DFT is

$$f_k \equiv \frac{1}{N} \sum_{n=0}^{N-1} F_n e^{2\pi i n k / N} \quad (4.8)$$

4.7.2 Fast-Fourier Transform

Fast-Fourier Transform (FFT) is a type of DFT, which was first introduced by James W. Cooley and John W. Tukey in the year 1965. FFT is used to reduce the computation work for N from $2N^2$ to $2N \log_2 N$. DFT can be computed using FFT by Danielson-Lanczos lemma, if N is power of 2 (Weisstein).

FFT can be differentiated into two classes – decimation in time and frequency. The method Cooley and Tukey first reverses the input data in bit wise order, that is they break up the transform length N into two $N/2$ (Weisstein).

The FFT used in this study is written in Matlab. The equation used in Matlab can be seen in the (4.9):

$$X(k) = \sum_{j=1}^N x(j) \omega_N^{(j-1)(k-1)}, \quad (4.9)$$

$$\omega_N = e^{(-2\pi i)/N}$$

4.8 Introduction to Butterworth algorithm

In 1930 physicist and engineer Stephen Butterworth developed a signal processing filter, now known as Butterworth filter. It was capable of flat frequencies responses for passband (Butterworth, 1930). For a low pass filter, the equation is given as (Wilcock):

$$|G_{lowpass}(f)| = \sqrt{\frac{1}{1 + \left(|f|/f_c\right)^{2n}}} \quad (4.10)$$

And for the high pass filter the equation is given as

$$|G_{highpass}(f)| = 1 - |G_{lowpass}(f)|$$

$$|G_{highpass}(f)| = 1 - \sqrt{\frac{1}{1 + \left(|f|/f_c\right)^{2n}}} \quad (4.11)$$

In the (4.10) and (4.11), f_c is the cut-off frequency and n is the order. The sharpness of the output frequency increases with the increase in n .

The passband (frequencies which can pass through) for the frequency is given by the following expression:

$$|G_{passband}(f)| = \sqrt{\frac{1}{1 + \left[(|f| - f_b)/f_c \right]^{2n}}}$$

The stopband (frequencies which cannot pass through) is given by the following expression:

$$\begin{aligned} |G_{stopband}(f)| &= 1 - |G_{passband}(f)| \\ &= 1 - \sqrt{\frac{1}{1 + \left[(|f| - f_b)/f_c \right]^{2n}}} \end{aligned}$$

4.9 Introduction to Threshold Based Encoding algorithm

The threshold based encoding algorithm converts the given input data (in this study it is EEG data) to trains of spikes. Let's consider an EEG channel X and its data points over time as x_i where $i = 1, 2, 3, \dots$; the mean of this channel is given by

$$X = \frac{\sum_{i=1}^{n-1} (x_i - x_{i+1})}{n - 1}$$

So the threshold value can be calculated as

$$x_n = \begin{cases} \text{if } (x_{n-1} - x_n) > X \text{ then } 1 \\ \text{else } -1 \end{cases}$$

Where $n = 1, 2, 3, \dots$

4.10 Conclusion

This chapter provides a brief description of the main types of neural networks and some pattern recognition algorithm, along with data transformation algorithms such as Fourier Transform, Butterworth algorithm. Threshold based encoding is also briefly explained.

Chapter 5: The Proposed Methodology and BCI System Design

Once we understand the functioning of the brain, BCI and various types of algorithms, we now introduce the functioning of NeuCube architecture for our BCI implementation. This chapter introduces the NeuCube architecture and the modules developed specifically to make it online BCI (dynamic) application. The test conducted in this study was to try and move the EEGRotor VE using EEG signals from the author brain. Screenshots of NeuCube for BCI can be seen Screenshots and video demo of NeuCube for BCI (video links) can be seen in Appendix C:.

5.1 Overview of the NeuCube Architecture

NeuCube is a spiking neural network architecture for learning, mapping and understanding of spatio-temporal and spectro-temporal data. It can be used for different kinds of such data, including brain data.

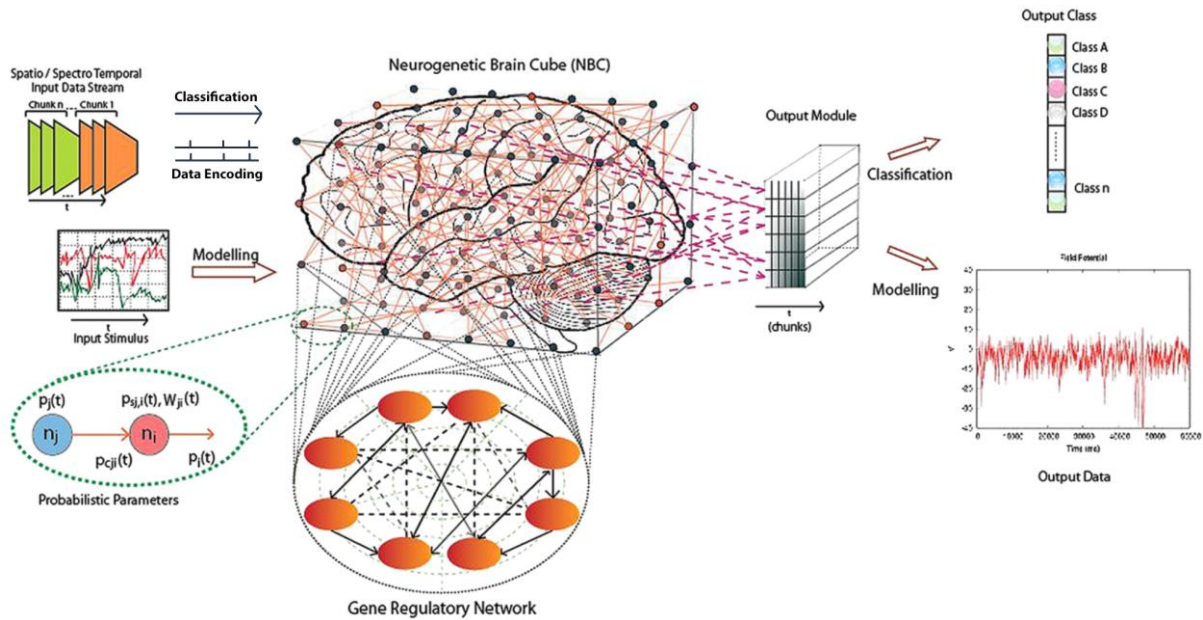


Figure 5.1: NeuCube architecture (N. K. Kasabov, 2014)

In our application, input data is a $[14 \times 128]$ matrix, where 14 is the number of channels (features) and 128 is the time points. One such data is called a sample. For each action (a task), multiple samples are taken. A set of tasks is labelled as one class. For example, a

person performs two tasks; one with eyes open and the other with eyes closed, each of these tasks are done 20 times (20 samples). So this means that we have a two class data with twenty samples each. These pair of samples are converted into 3D matrix (more on this in 5.4.1).

The 3D matrix is fed into the NeuCube, which then converts the samples into spikes using a Introduction to Threshold Based Encoding algorithm. These spikes are entered into the reservoir which uses Introduction to Spike-Timing Dependent Plasticity (STDP) to make the connections between the neurons and learn the pattern, also known as the unsupervised learning. Once the patterns are learnt, a classifier is trained (in this case Dynamic Evolving Spiking Neural Network (DeSNN) to classify a new data.

5.1.1 Modules in NeuCube

Figure 5.1 depicts the NeuCube architecture, which is made up of multiple modules. These are modules from the originally proposed architecture (N. K. Kasabov, 2014):

- Data encoding module;
- 3D Spiking Neural Network Reservoir;
- Classifier (output function);
- Gene regulatory network (GNR).

The NeuCube architecture is used for classifying data. The input data could be EEG or other Spatio-Temporal Brain Data (STBD) data.

The method for classifying STBD with NeuCube can be briefly described as following:

- I. Encode data into trains of spikes from STDB, i.e. EEG or FMRI data;
- II. By training the cube in unsupervised learning the 3D SNNr is evolved;
- III. Using the deSNN to train a classifier.
- IV. Optimise the model by iterating from I to III for different parameter values until maximum accuracy is achieved;
- V. Recall the model for new data.

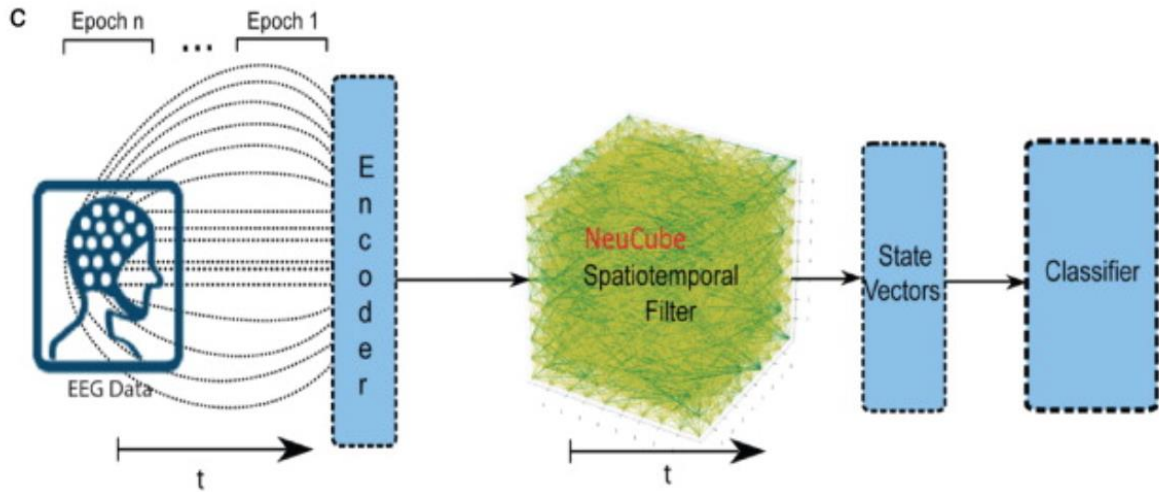


Figure 5.2: EEG data classification flow in NeuCube (N. K. Kasabov, 2014)

5.1.2 Classification of EEG data

In this study we have used the same method, but in much more advanced way to classify the given new input data after the training of the cube is done. Figure 5.2 presents the flow of the classification process.

The following steps occur when NeuCube learns and classifies the input data.

- EEG data in the form of 128x14 matrix is prepared and label names for the output classes are determined.
- Once the data is setup correctly, the input data is encoded into spike trains using AER encoding and fed to NeuCube.
- Unsupervised training is conducted in NeuCube using STDP algorithm.
- Once the training is done the connections between the neurons can be visualised.
- A classifier is trained as a second phase.
- Test classification result visualised for the sake of understanding the data.

NeuCube uses STDP for unsupervised learning, DeSNN for supervised learning and threshold based encoding to convert an input sample into a spike train.

Further discussions on NeuCube for BCI is provided in section 6.1.

5.2 Introduction to the concept of EEGRotor

As discussed previously (section 3.10), due to safety concerns Quadcopter was not used in this study. A virtual environment was developed that has a model of Quadcopter in it. Codes for the EEGRotor VE implementation can be found in Appendix C:.

EEGRotor is a virtual environment, which was developed using Maya and Unity 4. 3D modelling was developed in Maya, which is a 3D modelling software and the gaming engine used was Unity. This environment is a replica of Knowledge Engineering and Discovery Research Institute (KEDRI) at Auckland University of Technology (AUT) in WT building.

The main challenge in this is that the subject operating this by NeuCube BCI has to make sure that the Quadcopter model should pass through all the rings. These rings act like a feedback to the user. Each ring that he/she passes through with the model gives him/her 10 points, if missed no points are given. The classified output from the NeuCube is directly sent to the virtual environment using Java's Abstract Window Toolkit (AWT). **Figure 5.3** shows the environment in action, the last ring in this environment finishes the game and **Figure 5.4** shows the results to the subject.

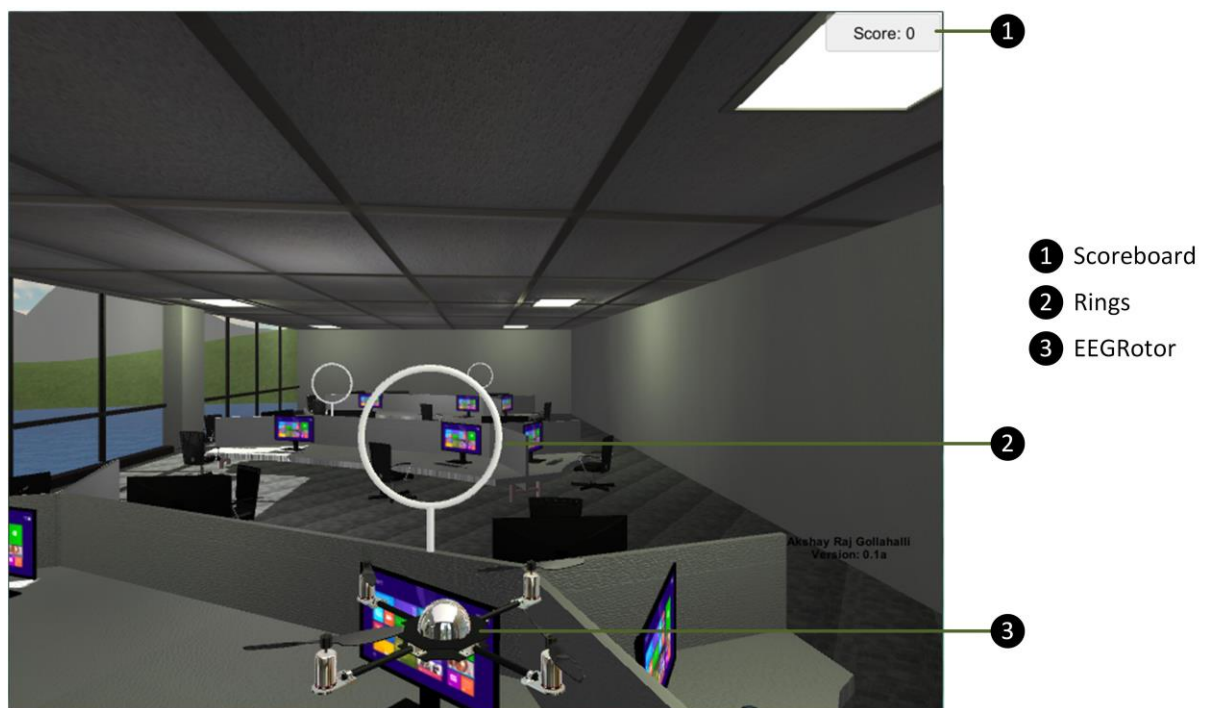


Figure 5.3: EEGRotor Virtual Environment

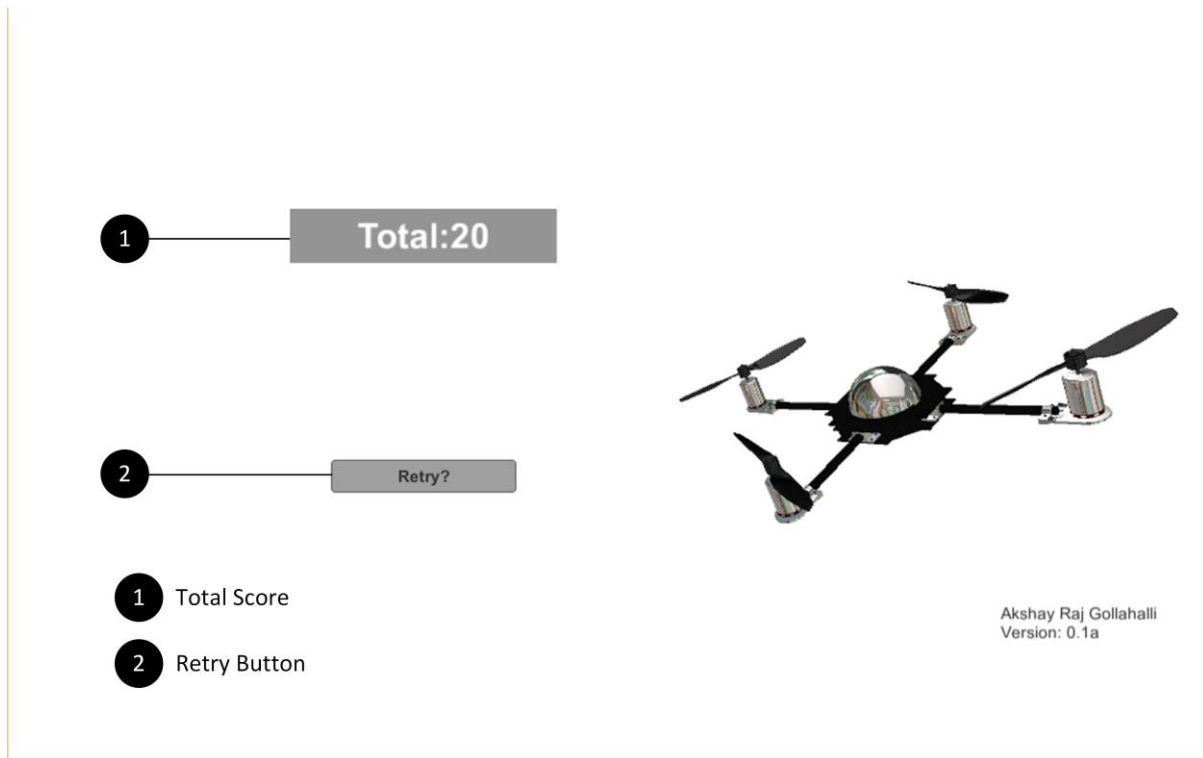


Figure 5.4: End of the VE

5.3 Experiments with EEGRotor

Resource for the experiments include the number of subjects used, experimental procedure and study framework.

5.3.1 Subjects used for this study

Due to time limitation for this study no external subjects were used for the experiments and the pilot testing was carried out on the author. The system was trained and tested using the author's EEG signals.

5.3.2 Experimental Procedure

The Emotive EPOC system is a headset made up of 14 saline based electrodes (wet electrodes) and has its own proprietary software. The author made sure that the electrodes were wet enough and were making a perfect contact to his scalp; this was monitored on the Emotiv control panel where green light for every electrode indicates the connection is good. Before putting on the head set the author had a healthy meal so that weakness would not affect the data recording process. Once the connection is perfect, recording of the EEG data was started.

The guidelines for data collection was by Ferree, Luu, Russell, and Tucker (2001) and Pivik et al. (1993) were followed. The experiment was conducted in KEDRI, AUT in the NeuLab. The environment was quiet and there was no disturbance.

Class number	Direction	Facial (mental and physical) Movement
1	Front	Both eyebrows up
2	Right	Right eyebrow up
3	Back	Both eyebrows down
4	Left	Left eyebrow up

Table 5.1: Four classes assigned for four facial movements

Table 5.1 shows four classes assigned to four facial movements. A class represents the label number for a facial movement. Twenty samples for each class were taken using the software developed in this study. The software functionality is described in the section 5.4. Two stages of EEG data were recorded: the author used his thoughts to mimic his facial movements, these thought signals were recorded and in the second stage, EEG data was collected while facial muscles were moved. While recording the data, it was made sure that the author was calm, sitting comfortably and there were no other movements. The system which was used to collect the data is explained in section 5.4.1.

In Emotiv EPOC headset, each second of data is represented as 128 time points (128 Hz = 128 time points or milliseconds). Twenty EEG data samples were collected for each class. Each sample has 2 seconds of EEG data. Total time recorded was nearly equal to 2.7 minutes for each stage. The reason for collecting thought process and physical muscle movement EEG data is to minimise the error rate and other external disturbances. These two stages where combined together to form a set of EEG data.

This set is sent through an encoder that converts the EEG data into trains of spikes. These spike trains are sent through unsupervised learning (section 4.5); finally the data is sent through supervised learning (section 4.6) to classify.

As explained earlier (section 5.1.2), user can either save the trained NeuCube parameters (neuron weights, location and coordinates) or split the sample. In this study the author saved the NeuCube parameters for classifying with new data. NeuCube for BCI has an option to load previously saved parameters, so that classification with new data could be done easily. The author loaded back the trained NeuCube. EEGRotor virtual environment is opened. At this point the stream of EEG data is passed through

the NeuCube which recognises the data coming through and send the signal to the virtual environment.

5.3.3 Framework

The proposed framework is divided into two procedures: training NeuCube for BCI with recorded EEG data (Figure 5.5); testing (recall) EEGRotor with new EEG data.

5.3.3.1 Training NeuCube for BCI to run EEGRotor

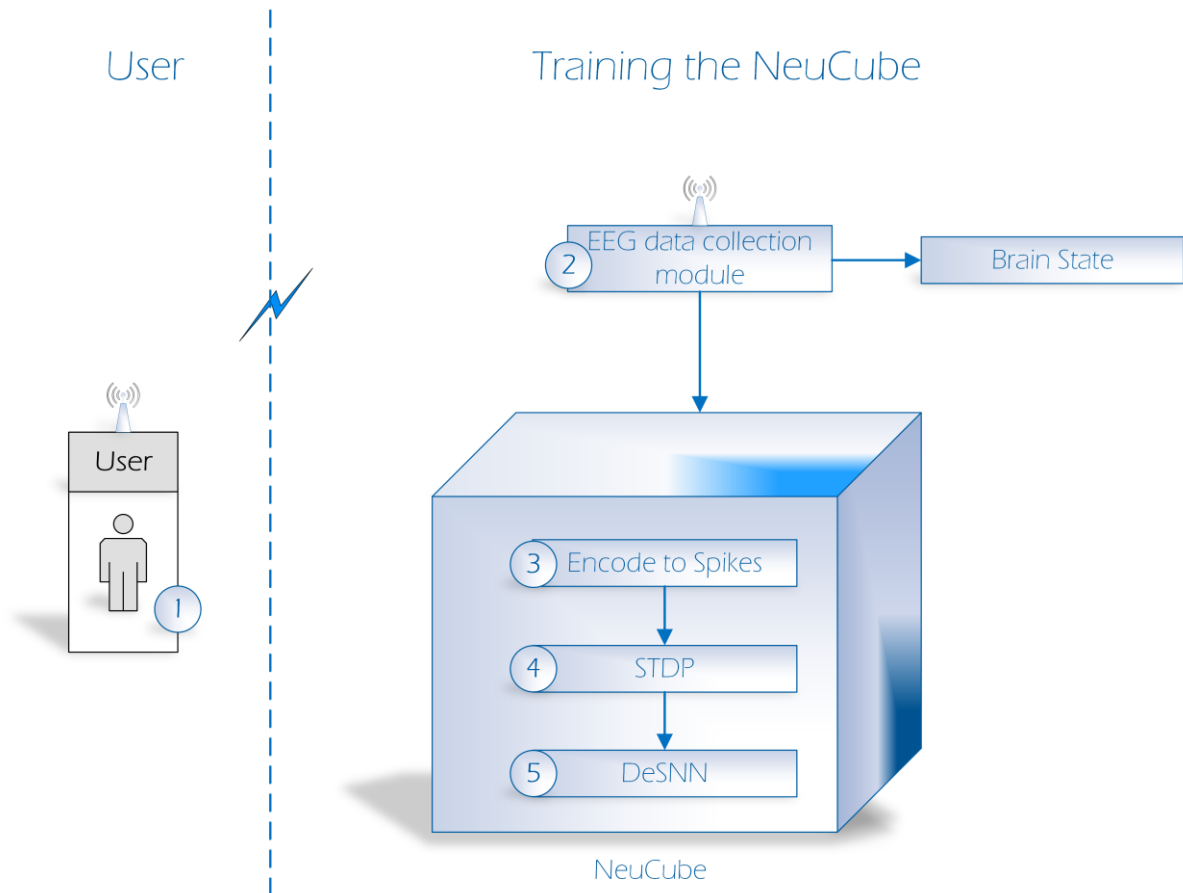


Figure 5.5: Training the NeuCube for BCI to run EEGRotor

The arrows in Figure 5.5 shows the flow of training an EEG data in the NeuCube.

1. The user wearing the Emotiv EPOC headset with 14 wet electrodes can transmit data via Bluetooth at the rate of 128 Hz.
2. The signals transmitted from the Emotiv headset can be recorded for each class using **EEGRotor Training Module**; the module requires specification of how many samples per class are needed and the time frame for each class. Once the recording is done the module creates a *.mat file which is required to train the data; the outline of the folder structure and the file can be seen in Figure 5.11.

3. Using **EEGRotor State of Mind** Module is an optional module, where the state of mind can be checked. State of mind plays important role in this study, the user can see if he/she is calm or excited in this module. If the user feels that the state of mind is fluctuating, the EEG data can be rerecorded using the training module. This module uses Fast Fourier Transformation (FFT) and Butterworth algorithm to split the given data into Alpha, Beta, Gamma and Theta waves with time points.
4. Encoder (section 4.9) is a part of the NeuCube architecture. It converts the EEG data into trains of spikes. These spikes are binary data, which is required by STDP and DeSNN algorithms.
5. STDP algorithm makes sure that the weight between two neurons change while the learning takes place. This is explained in section 4.5.
6. While training the data for 100% the DeSNN classifier is not used. The main use for this algorithm is to classify the data and also if the label values are present it would give you the accuracy value.
7. As discussed earlier (section 5.3.2) the user has the ability to train the data in two ways: complete or split the data. Splitting the data will use the first half for training and the second half for testing/validation. Once the training of the NeuCube is finished, DeSNN classifier can be used to classify the second half of the data with the trained data. This procedure is optional for BCI in the training stage.

Once the training of NeuCube is finished, the user can save a complete copy of the neuron weights and its coordinates. The usability of this cube is explained in 5.3.3.2.

5.3.3.2 Testing the NeuCube for BCI to run EEGRotor

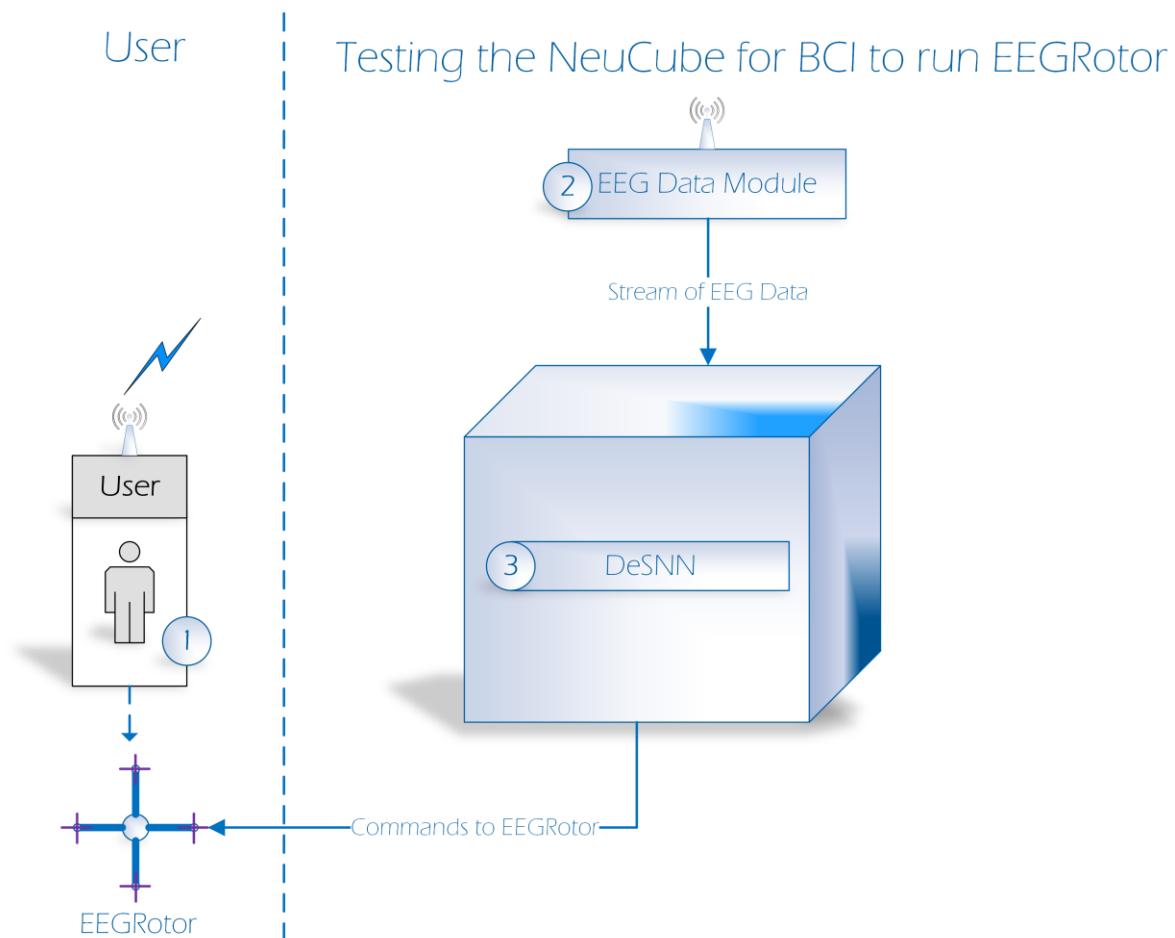


Figure 5.6: Testing the NeuCube for BCI to run EEGRotor

The dotted arrow in the Figure 5.6 shows a hypothetical flow of controlling the EEGRotor and the bold arrows show the flows of testing EEG data.

1. As explained in section 5.3.3.1, Emotiv EPOC headset with 14 wet electrodes can transmit data via Bluetooth at the rate of 128 Hz.
2. Dynamic EEG data module (section 5.4.5) is inter-linked to EEG data control panel module (section 5.4.7), which enables the flow of EEG data from Emotiv headset to the NeuCube.
3. Finally, the DeSNN classifier takes in the stream of EEG data and classifies it. Based on the classified class, the NeuCube sends the appropriate commands to EEGRotor. The commands are discussed in Table 5.1.

5.4 EEGRotor for Brain-Computer Interface

This section describes how NeuCube and EEGRotor for BCI work and the functions of the modules. As discussed earlier, NeuCube is a pattern recognition framework for spatio-

temporal data (N. K. Kasabov, 2014) and is capable of classifying input data without its label values (only after its trained with data and class values). The basic workflow of the model can be seen in **Figure 5.7**.

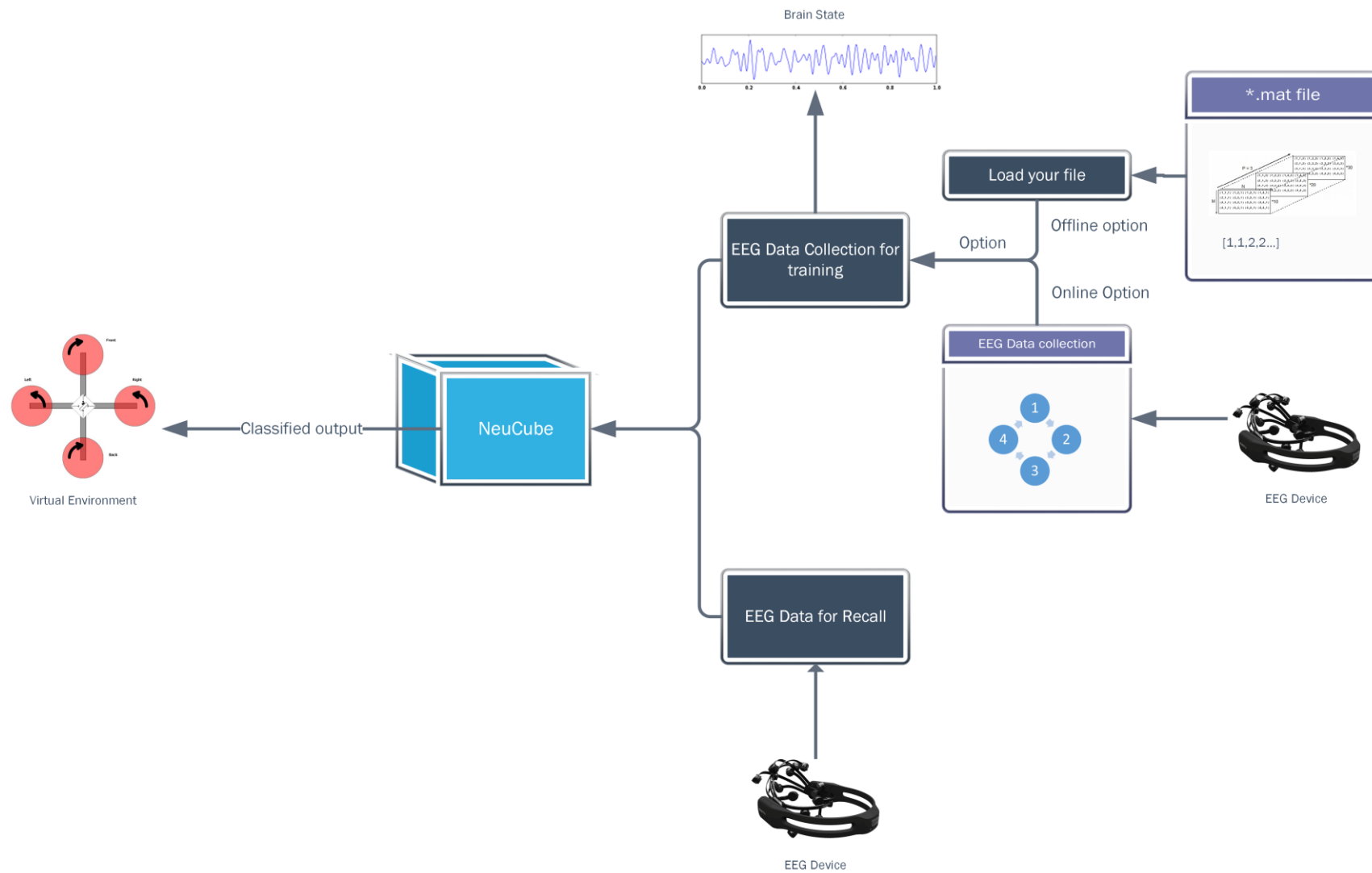


Figure 5.7: Functional diagram of NeuCube for BCI to control EEGRotor

As explained in section 5.3.2, the training module in Figure 5.6 shows that the classification of the input data can be continuous stream of EEG data or static EEG data. An input can be given in two formats, in .mat format or in a stream of 128×14 matrices. The process starts with inputting data, then the state of mind response of the subject can be observed (optional); from there it goes to unsupervised training and finally supervised training which classifies the new input data and sends the recognised command to the virtual environment. For training, the training data is taken from the subject and the training is done in the same fashion as the testing (more on this in Chapter 6:). **Figure 5.8** shows how the software looks like while reading and classifying the data.

Figure 5.8 shows the functions, and how to optimise each of the modules for better performance. In Chapter 6:, the author will discuss on how the data is collected using the training module. Further in section 6.1, the author will discuss on how the complete modules when come together work. Finally, the author would discuss on how we could increase the performance of the software.

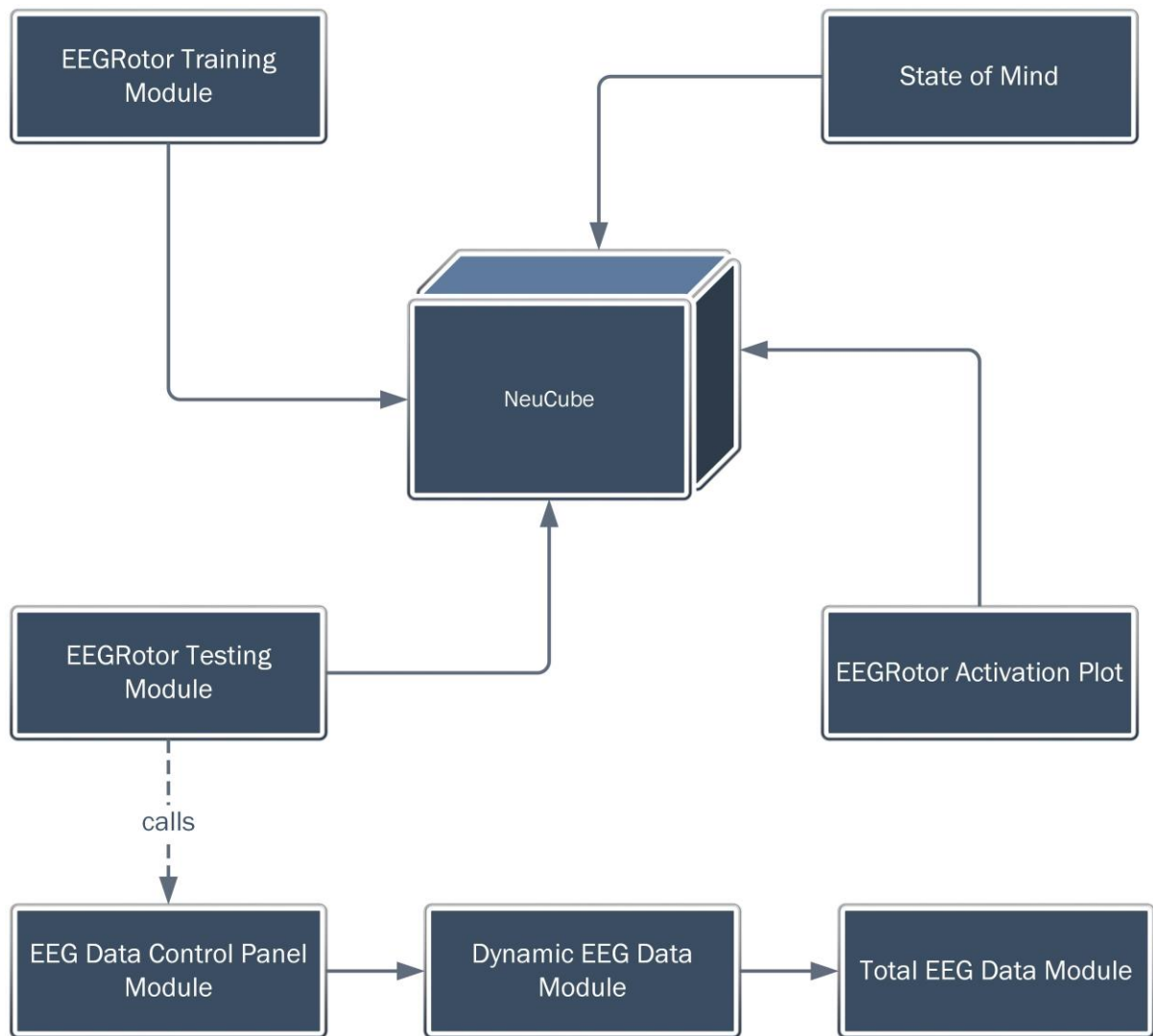


Figure 5.8: NeuCube for Brain Computer Interface (BCI)

There are seven modules associated with NeuCube for BCI to run EEGRotor:

- EEGRotor Training Module
- EEGRotor Testing Module
- EEGRotor State of Mind Module
- EEGRotor Activation Plot Module
- Dynamic EEG Data Module
- Total EEG Data Module
- EEG Data Control Panel Module

A detailed description of each module is given below:

5.4.1 EEGRotor Training Module

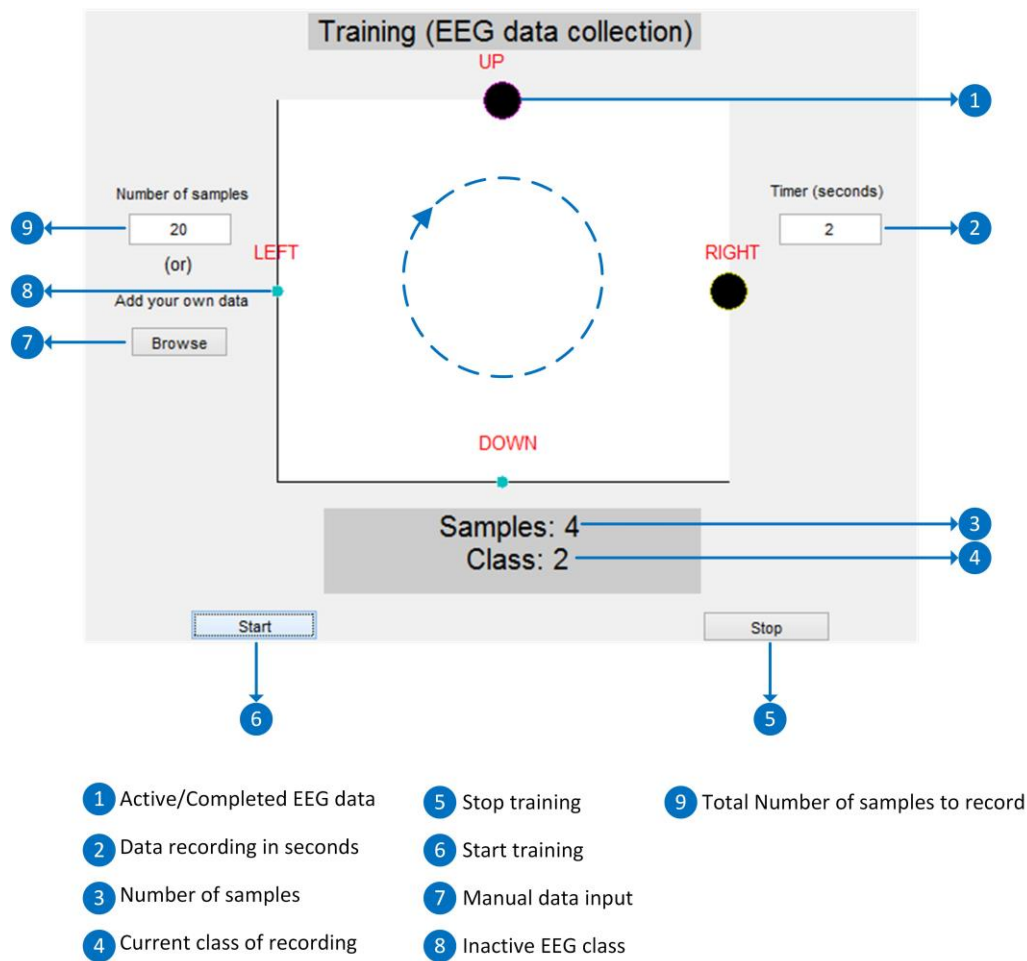


Figure 5.9: EEG data collection model to collect four class EEG data with its class label and create a `final_eeg.mat` file

Figure 5.9 shows the feature of the EEG Training phase, there are two buttons, start and stop; start button starts the flow of dynamic data from the Emotiv EPOC device that would produce 128×14 two dimensional matrix in the Matlab; where 14 is the number of EEG channels and 128 is the time points (in EPOC head set 128 is equal to one second). The stop button pauses the data streaming.

The training module does the following:

- The data can be collected in two ways: One way is to provide your own data in the format of a 3D matrix and a label as a 2D matrix. The other way is that this module takes the data directly from the Emotiv EEG headset and created the 3D and 2D matrix automatically. If you choose the second way to collect data, you need to first select the number of samples you want to record for each class. If

you choose to provide the data manually you need to follow the format shown in **Figure 5.10**.

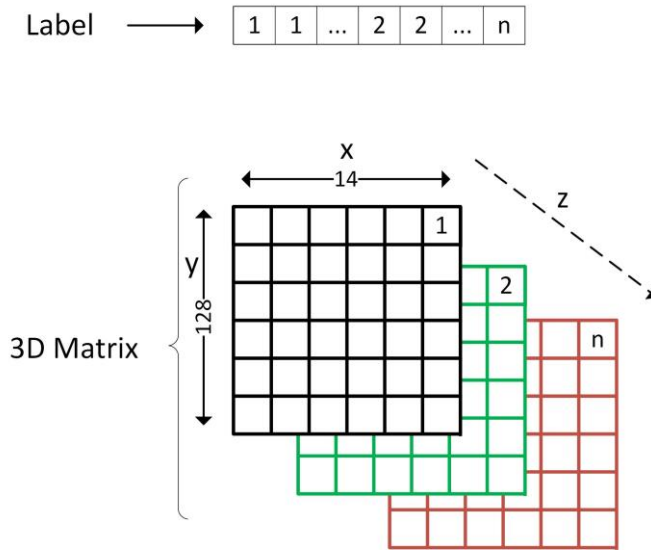


Figure 5.10: 3D Matrix

A three dimensional matrix consist of (x, y, z) data elements, where x is the number of EEG channels, y is the number of time points and z is the number of samples.

- Next the user has to select the timer (data recorded in seconds) input; for example in the **Figure 5.10** the timer (C) shows 2, which means the user can take two seconds of data for each class, $y = 128 \times 2$ and x will be 14 because these are 14 channels of EEG device; so the final output would be $(x \times y) = (256 \times 14)$ matrix. The higher the timer, the higher the number of data points are and the more time for the software to compute.
- Once this is done and the EEG headset is placed on the subject's head, the user can start the process by clicking on the start button. The process starts in clockwise direction; from Table 5.1 the class starts from class 1 – UP, and ends at class 4 – LEFT.
- EEG data collection stops automatically once the number of samples specified is reached. However, the researcher can stop the process anytime by clicking on the stop button.
- Once the process is completed, the software creates a folder structure which can be seen in **Figure 5.11**; each class (EEG data) is stored separately in its corresponding folder with its label (class label) file. It also creates a file called

eeg_final.mat, who's structure is in the form of a 3D matrix (Figure 5.10) is then used in the NeuCube for training.

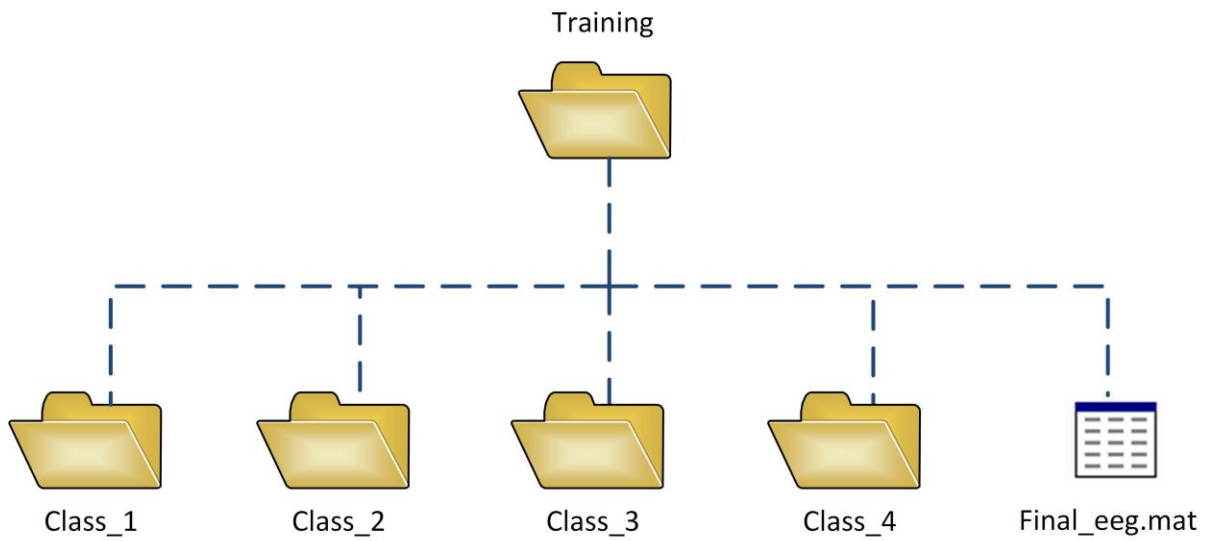


Figure 5.11: Folder structure for Training phase

5.4.2 EEGRotor Testing Module

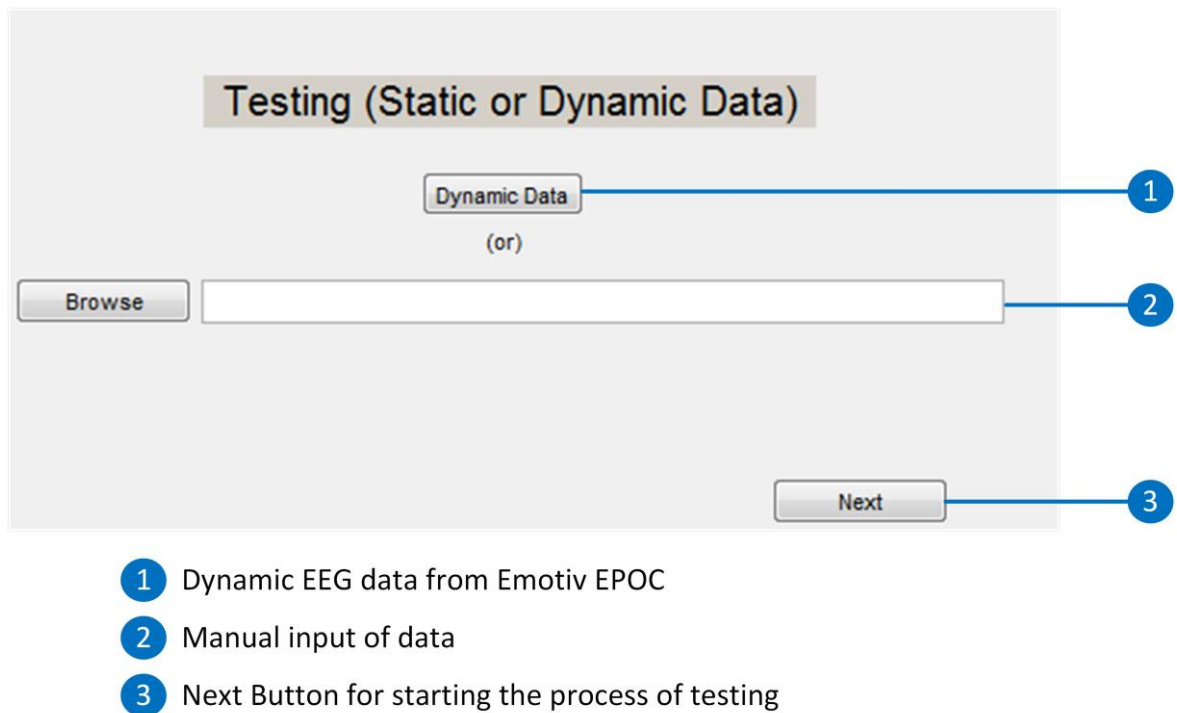


Figure 5.12: Interface of the testing module

This module includes the trained NeuCube which is tested by using a new data that could be directly taken from the user or it could be a manual input. Once this is done, the user can then click on “Next” which will take him/her to the next process of classifying the data. After the classification the user could use the output to give the appropriate

command which to EEGRotor VE. In this module the user need not give any label values for the sample data, as it is not important. When no label values are supplied NeuCube predicts the class label for the data.

5.4.3 EEGRotor State of Mind Module

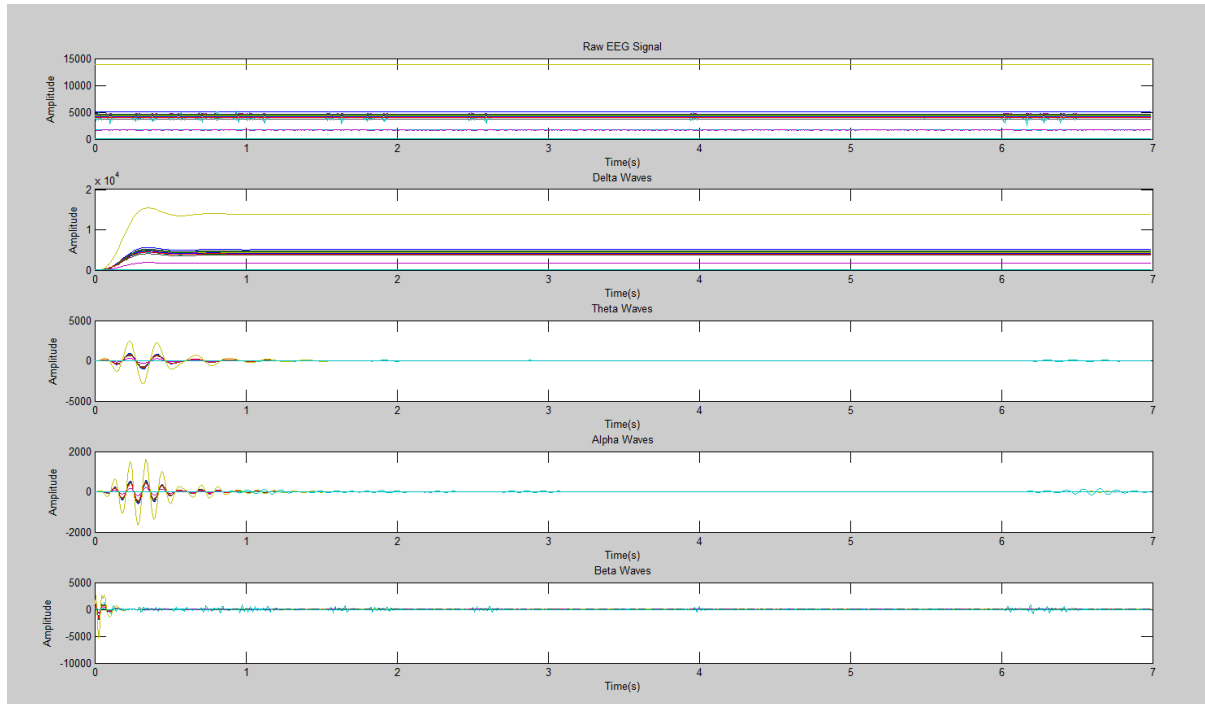


Figure 5.13: State of mind module with raw EEG data, delta waves, theta waves, alpha waves and beta waves

This module is an optional module. It extracts Delta, Theta, Alpha and Beta waves after the EEG acquisition is completed. Due to Matlab's inability to run multiple loops at a time, this particular module cannot run on dynamic data. **Figure 5.13** shows how the EEG wave forms are extracted over time. Y-axes is the amplitude of the wave and the X-axis is the time period in seconds. This module can be enabled by choosing "Emotion State" from the drop down menu of "Plot Options" which is on the right side of the NeuCube interface.

When the EEG testing is done, all the readings are saved to the Matlab workspace as a 2D matrix which consists of n rows and 14 columns; these n rows are the time points for the recorded EEG data. EEG data are stored as $128 * n$. When this is input to the state of mind function the data is extracted using Fast Fourier Transformations and Butterworth algorithm which is available with the Matlab program; after the extraction is done, the extracted alpha, beta, theta and delta waves can be plotted individually.

5.4.4 EEGRotor Activation Plot Module

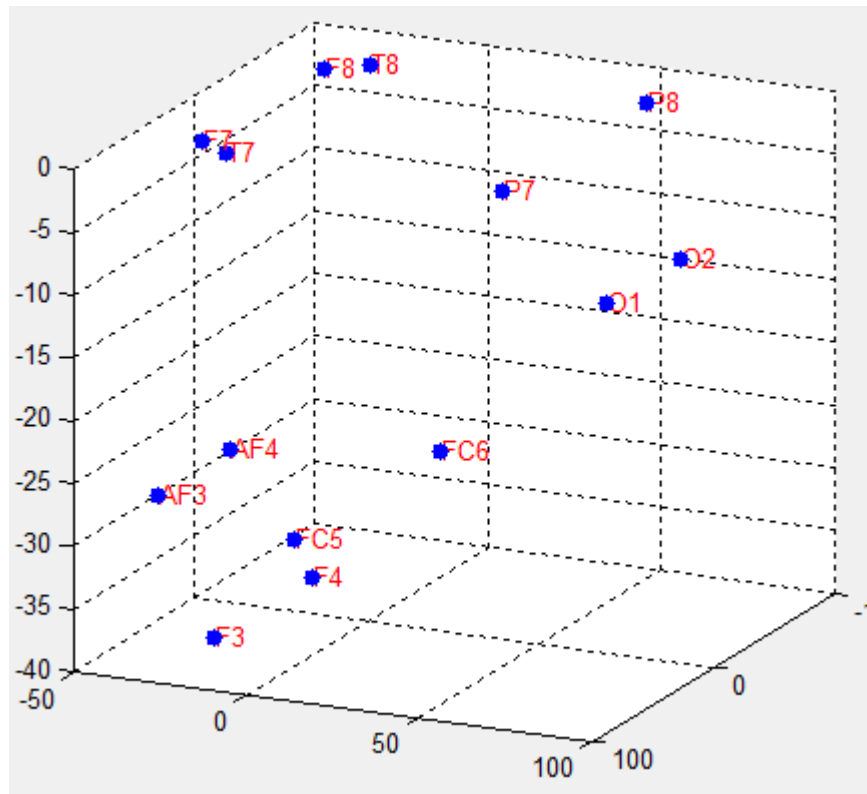


Figure 5.14: Activation Plot for EEG data collected from Emotive headset

Activation plot is an optional plot that uses Talairach coordinates in which the activation state of each electrode directly taken from the Emotiv EPOC EEG device. When a certain threshold is exceeded, the electrode location (blue dots) indicated by the corresponding point in **Figure 5.14** highlights. This shows which part of the brain is more active when compared to other parts of the brain. As with the emotion module, this is an independent module which does not depend on NeuCube.

5.4.5 Dynamic EEG Data Module

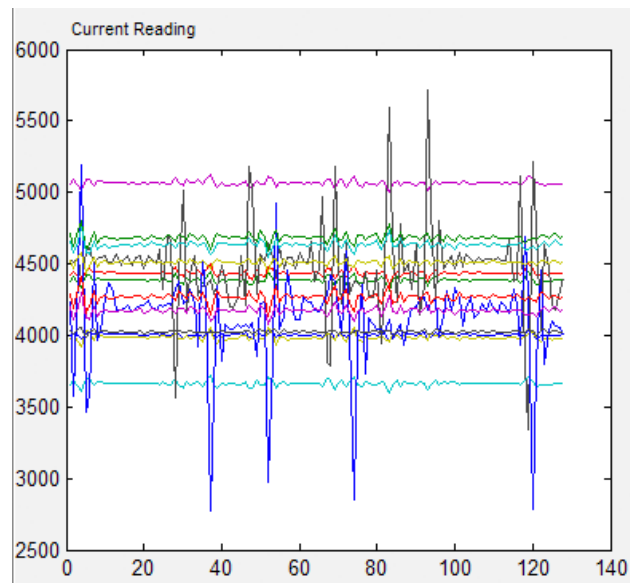


Figure 5.15: Dynamic EEG data plotted over time. Y-axes shows the amplitude and X-axes shows the time

This module provides live visualisation of the incoming EEG data which has certain amplitude and frequency. The Y-axis shows the amplitude of the 14-channel EEG signals separated by their frequencies and the X-axis shows the time points in 128 divisions. As explained earlier 128 time points is equal to one second.

This module uses a timer function for scheduling the data. Old data is replaced by the data.

This module facilitates understanding brain responses to each task. Higher amplitude of brain signals means that the subject is performing visually or mentally intense task. Lower amplitudes occur when the person is in his/her rest position without performing any task.

5.4.6 Total EEG Data Module

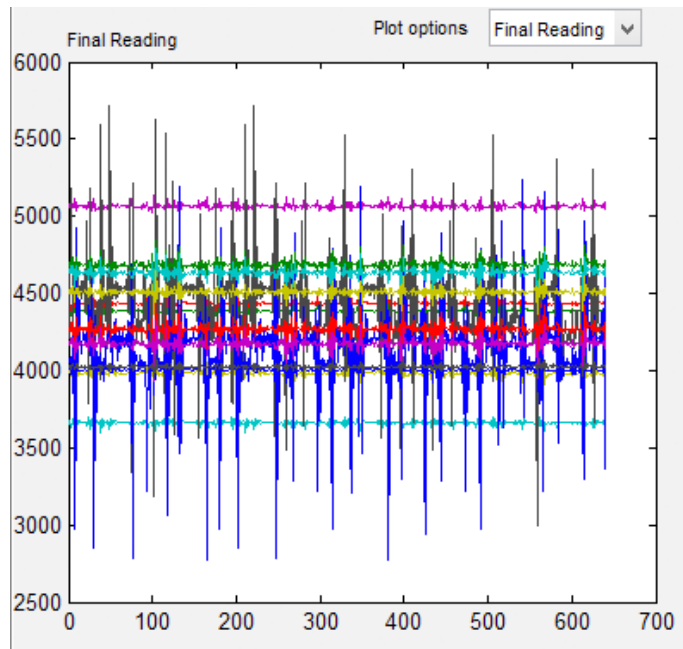


Figure 5.16: Final EEG

This module gives the total output of the EEG data. This module activates when the stop button on the EEG control panel is clicked. Dynamic data that comes in through the Emotiv EPOC EEG device is stored as a 2D matrix in the workspace; this happens if and only if the Stop button on the control panel is clicked. This would give the overall observation of the final EEG data, which can be used to interpret the excitation and rest state of the person (Note that this stage does not give out the emotional state. Refer to **EEGRotor State** of Mind Module). Y-axis shows the amplitude of the 14 EEG channels and the X-axis of the time points in multiples of 128 (one second) divisions. In **Figure 5.16** the number of time points is ≈ 650 which means that the data was recorded for almost 5 seconds ($650 \text{ divisions} / 128 \text{ divisions}$).

Also, this data is cleared when the user starts the reading of EEG data again. This process is connected in classifier of the NeuCube which will be explained in section 6.1.

5.4.7 EEG Data Control Panel Module

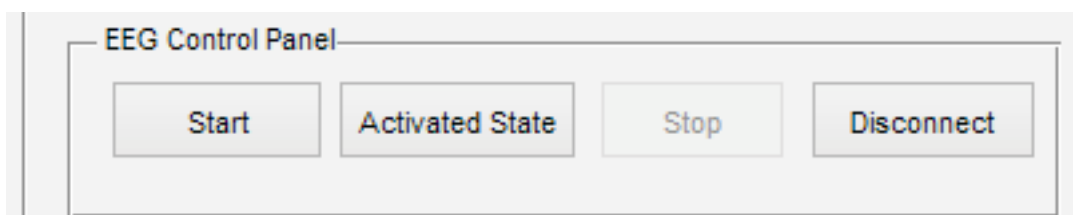


Figure 5.17: Interface for the EEG data control panel

The EEG Data Control Panel (EDCP) is the heart of NeuCube for BCI. Emotiv has proprietary software called “Emotiv Control Panel”, which communicates with Emotiv EPOC EEG headset to transmit EEG data. They also provide Software Development Kit (SDK) developed in C++ that links the EEG data control panel to Emotiv control panel. It can be used as an interface caller to read the EEG data from the EPOC headset. There are two types of connections to the Emotiv EPOC head set: one via TCP/IP and the other one is a direct connection to the hardware through general ports. The main connection from the headset to the computer is done through Bluetooth protocols.

The Start button tells the SDK to connect to modules developed in Matlab and the control panel; once this connection is established the next thing is to start reading the EEG data from the device. Stop button pauses the EEG reading and the Disconnect button will clear the cached data and disconnects the EEG control panel from Emotiv Control Panel (and SDK).

5.5 Conclusion

This chapter describes how the of NeuCube architecture functions. It introduces a virtual environment called EEGRotor and explains the research framework for this study using the virtual environment. All seven modules developed for online application of EEGRotor are also explained.

Chapter 6: Experimental Results and Analysis of the EEGRotor

6.1 Working of the Complete Software Suite

This BCI system is capable of classifying personalised data (data belongs to a single person). **Figure 5.8** shows all modules that use the NeuCube architecture to process continuous stream of brain data. The following software system are required to run the NeuCube based BCI (the EEGRotor):

- A Windows computer running i5 processor or equivalent and 4 GB RAM;
- Matlab R2013a software program;
- Visual Studio 2010;
- Emotiv EPOC Research Version;
- Emotiv Research SDK;
- NeuCube for Brain Computer Interface.

6.1.1 Training NeuCube for BCI

Training is the first stage that needs to be done before the BCI system can classify new data. The following are the steps that need to be followed for the procedure to take place correctly:

- This study is based on four classes (see Table 5.1). EEGRotor Training Module collects the EEG brain data from the subject and creates files accordingly. NeuCube can take more than four class but the user has to create *.mat file that should follow **Figure 5.10**.
- Once the user has recorded the EEG data, the first goal is to train NeuCube; for this the user has to click on the drop down menu at “What do you want to do?” which is to the left of the NeuCube interface that can be seen in Figure 6.1: Training

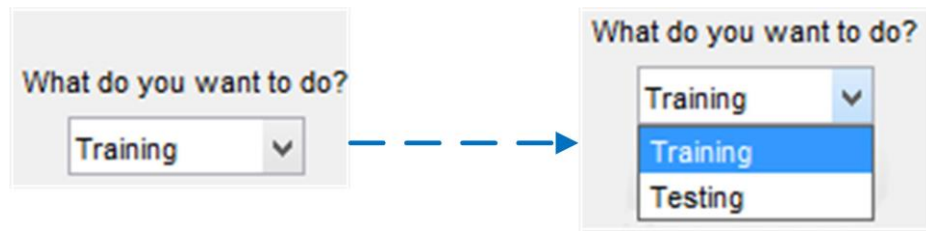


Figure 6.1: Training

- Once the user clicks on it, a pop up window appears; click on the browse button to add *.mat file (making sure the user includes label file within *.mat file) and close the window. This will make the system know that a manual entry was given to it; this will activate the NeuCube system.
- The user has to initialise the system. He/she can also provide neural coordinates (Talairach coordinates for EEG electrodes) and neuron coordinates (neuron coordinates for NeuCube) if needed. Click on “Initialize NeuCube” button which will initialise NeuCube with the appropriate neurons placed with their neural coordinates, this can be seen in the cube (Figure 6.4).
- Before user starts the unsupervised learning, he/she has to train the software with 100% data (no splitting of data); to do that the user has to set Training division to 1.
- Once this is done click on unsupervised learning followed by supervised learning.
- Finally, the user has to save the NeuCube file by clicking on “save NeuCube”. This file contains the connection weights and coordinates for the trained data. This can be then used to classify new EEG data.

6.1.2 Testing NeuCube for BCI

Once NeuCube is trained and it has learnt the provided EEG data, it can classify new data that can be entered either dynamically or manually. The user would have to follow these to get the correct classification.

- The saved NeuCube is loaded back to NeuCube (this is because of the Matlab software program that has to clear cached data).
- There are two options for entering the data that needs classification: either to enter manually the EEG data in 2D format i.e. 128×14 or use continuous EEG by clicking on Dynamic Data in the pop up menu. Click “start” on the EEG data control panel.

- Ensure the subject wears the Emotiv EPOC headset. The SDK connects to Matlab and stream of EEG data (in 128×14 2D matrix) (visualisation of data can be seen in the **Figure 5.15**) is converted to trains of spike and classified using DeSNN classifier (supervised classification), which sends the command to EEGRotor VE

6.2 Analysis of Results

After following the steps given in sections **6.1.1** and **6.1.2** the classified output is presented in the Matlab software command window:

```
----- RESULTS -----
Time:12-Jun-2015 14:25:36
Class label of the given samples:
Sample 0001 -----> class 1
-----
```

In the above result, the user can see the time at which the execution took place; and the sample 0001 indicates the sample number and what class the input data belongs to.

As explained previously, each class indicates one of the four directions in the EEGRotor VE; the class labels and directions of movement can be seen in Table 5.1

NeuCube for BCI was tested on:

- Self-generated neural coordinates with 1000 neurons;
- With Talairach coordinates that has 1485 neurons.

This means that there are two separate NeuCube files which can be used for testing or validating new data.

A self-generated map is produced by the NeuCube using time series correlation, which means that it check the correlation between each feature and check which feature is close together and which is not. The closer features are kept near to each other and the others are kept far away from each other.

Talairach coordinates are 3D coordinate of human brain. It is a map of the brain which shows the structure of each part (Bankman, 2000).

The results from the above tests are presented seen in section **6.2.1**.

6.2.1 Results in tabular form

The results from two types of testing are presented in tabular form.

6.2.1.1 Self-generated Mapping

This test uses self-generated neural coordinates with 1000 neurons and was performed three times.

Sample Number	Classified Class Number	Classified Direction	Actual Class	Actual Direction	Task
1	1	Front	1	Front	Both eyebrows up
2	1	Front	1	Front	Both eyebrows up
3	1	Front	1	Front	Both eyebrows up
4	2	Right	4	Left	Left eyebrow up
5	1	Front	4	Left	Left eyebrow up
6	1	Front	1	Front	Both eyebrows up
7	1	Front	1	Front	Both eyebrows up
8	4	Left	1	Front	Both eyebrows up
9	2	Right	1	Front	Both eyebrows up
10	1	Front	1	Front	Both eyebrows up

Table 6.1: Results using self-generated mapping – Test 1

Sample Number	Classified Class Number	Classified Direction	Actual Class	Actual Direction	Task
1	1	Front	1	Front	Both eyebrows up
2	1	Front	1	Front	Both eyebrows up
3	3	Back	1	Front	Both eyebrows up
4	2	Right	4	Left	Left eyebrow up
5	1	Front	4	Left	Left eyebrow up
6	4	Left	1	Front	Both eyebrows up
7	1	Front	1	Front	Both eyebrows up
8	4	Left	1	Front	Both eyebrows up
9	2	Right	1	Front	Both eyebrows up
10	2	Right	1	Front	Both eyebrows up

Table 6.2: Results using self-generated mapping – Test 2

Sample Number	Classified Class Number	Classified Direction	Actual Class	Actual Direction	Task
1	2	Right	1	Front	Both eyebrows up
2	1	Front	1	Front	Both eyebrows up
3	1	Front	1	Front	Both eyebrows up
4	2	Right	4	Left	Left eyebrow up
5	1	Front	4	Left	Left eyebrow up
6	1	Front	1	Front	Both eyebrows up
7	1	Front	1	Front	Both eyebrows up
8	1	Front	1	Front	Both eyebrows up

9	2	Right	1	Front	Both eyebrows up
10	3	Back	1	Front	Both eyebrows up

Table 6.3: Results using self-generated mapping – Test 3

6.2.1.2 With manual Input Mapping

This test uses Talairach coordinates with 1485 neurons and was performed three times.

Sample Number	Classified Class Number	Classified Direction	Actual class	Actual Direction	Task
1	1	Front	1	Front	Both eyebrows up
2	1	Front	1	Front	Both eyebrows up
3	3	Back	1	Front	Both eyebrows up
4	4	Left	4	Left	Left eyebrow up
5	4	Left	4	Left	Left eyebrow up
6	4	Left	1	Front	Both eyebrows up
7	1	Front	1	Front	Both eyebrows up
8	3	Back	1	Front	Both eyebrows up
9	3	Back	1	Front	Both eyebrows up
10	4	Left	1	Front	Both eyebrows up

Table 6.4: Results using manual input of mapping – Test 1

Sample Number	Classified Class Number	Classified Direction	Actual class	Actual Direction	Task
1	1	Front	1	Front	Both eyebrows up
2	2	Right	1	Front	Both eyebrows up
3	3	Back	1	Front	Both eyebrows up
4	4	Left	4	Left	Left eyebrow up
5	1	Front	4	Left	Left eyebrow up
6	4	Left	1	Front	Both eyebrows up
7	1	Front	1	Front	Both eyebrows up
8	3	Back	1	Front	Both eyebrows up
9	3	Back	1	Front	Both eyebrows up
10	4	Left	1	Front	Both eyebrows up

Table 6.5: Results using manual input of mapping – Test 2

Sample Number	Classified Class Number	Classified Direction	Actual class	Actual Direction	Task
1	1	Front	1	Front	Both eyebrows up
2	1	Front	1	Front	Both eyebrows up
3	1	Front	1	Front	Both eyebrows up
4	4	Left	4	Left	Left eyebrow up
5	2	Right	4	Left	Left eyebrow up
6	4	Left	1	Front	Both eyebrows up
7	1	Front	1	Front	Both eyebrows up
8	1	Back	1	Front	Both eyebrows up
9	1	Back	1	Front	Both eyebrows up

10	1	Left	1	Front	Both eyebrows up
----	---	------	---	-------	------------------

Table 6.6: Results using manual input of mapping – Test 3

6.2.2 Visualisation

This section presents visualisations of neurons connection and activation plots.

6.2.2.1 Self-generated Mapping

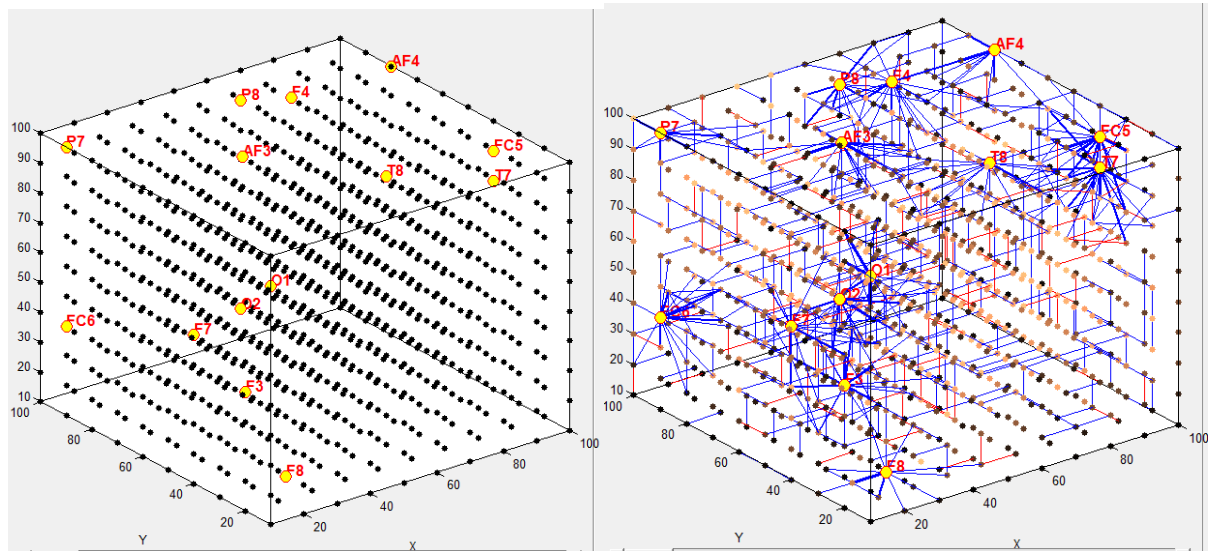


Figure 6.2: (Left) The cube before initialising and showing the input features. (Right) Showing the trained cube after the training is done, thicker lines indicate stronger connections.

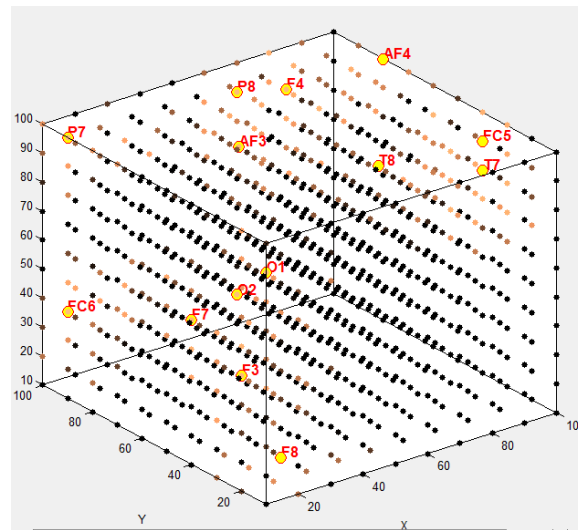


Figure 6.3: This cube shows the activation level for each feature; the brightness of dots indicates the level of activation.

6.2.2.2 With manual Input Mapping

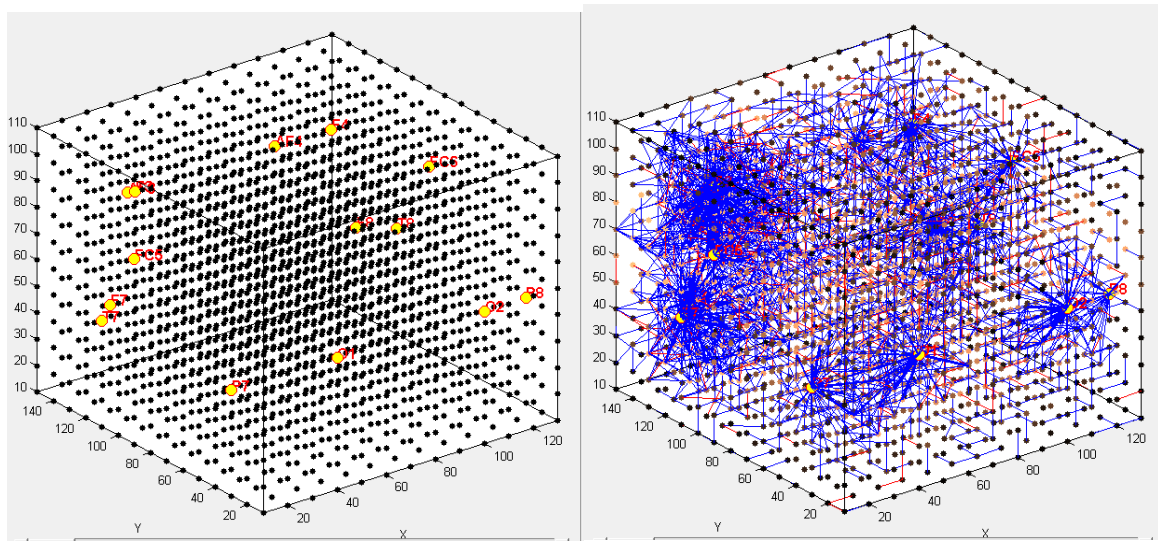


Figure 6.4: (Left) The cube showing the Talairach coordinates before initialisation. (Right) The cube showing the connections between neurons, thicker lines indicate stronger connections.

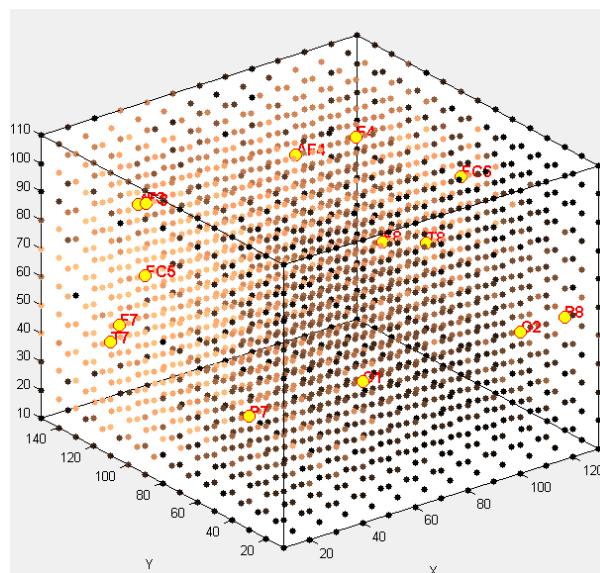


Figure 6.5: The cube shows the activation level; the brightness of dots indicates the level of activation.

NeuCube has the ability to visualise the data dynamically that is it can show the connections between the neurons.

Yellow dots from figures Figure 6.2, Figure 6.3, Figure 6.4 and Figure 6.5 are the input features, each dot corresponds to one electrode of the Emotiv headset. When the training is complete, user can see the connections between the neurons (this visualises how brain would connect neurons). Blue lines signify the connections between neurons, the thicker the line is the stronger is the connection. For example in the **Figure 6.4 (right)** shows a cluster of

connections in the front of the image, which shows that the author had higher amount of activity on his prefrontal cortex.

6.3 Discussion

The NeuCube architecture is a complex software which can be used to classify STBD and SSBD. The architecture has been modified to run online EEG data, which would allow it to issue commands for movements to robots or flying objects. Due to safety concerns, a virtual environment called EEGRotor was specifically developed where a virtual flying object could be navigated.

Two types of tests were run using NeuCube as discussed above: one using the self-generated mapping and the second one using the Talairach coordinates (which is based on 10-20 international system).

Using the tasks and assigned classes shown in the Table 5.1, the data was recorded as discussed in section 5.4.1. Once the training is done, NeuCube was saved and the application was reopened to refresh the cached content. After that NeuCube is loaded back in. Using the EEGRotor Testing Module and EEG Data Control Panel Module the live data is input, which is followed by classifying saving of the data.

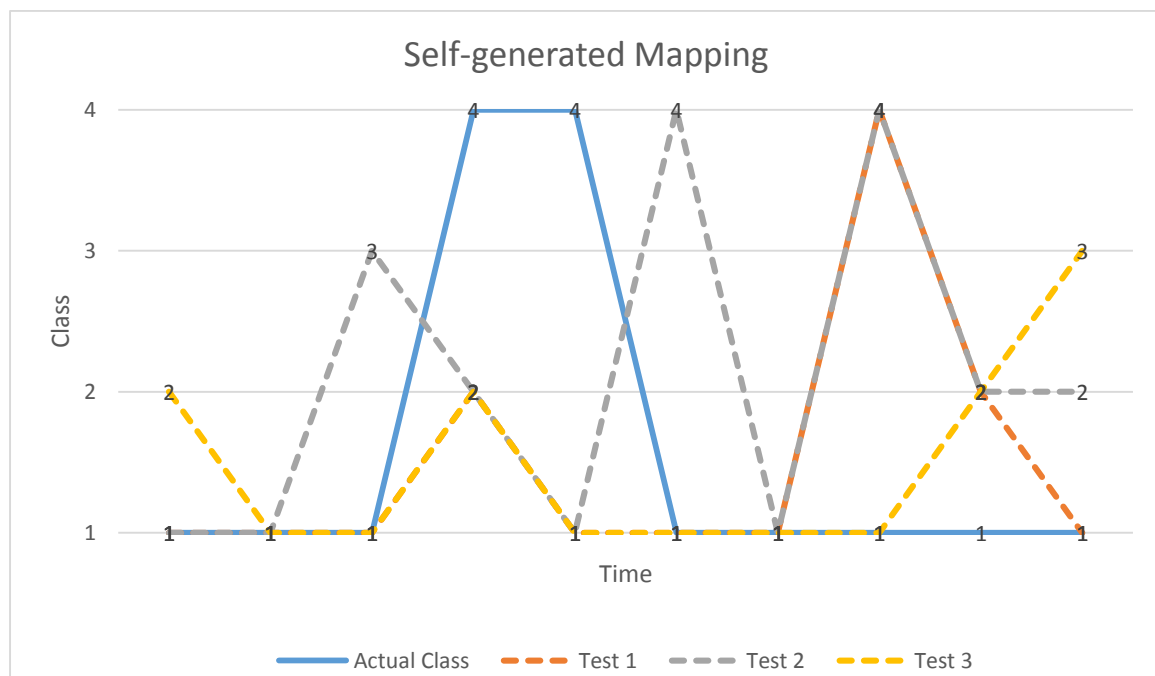


Figure 6.6: Comparison of actual class to classified class in self-generated mapping.

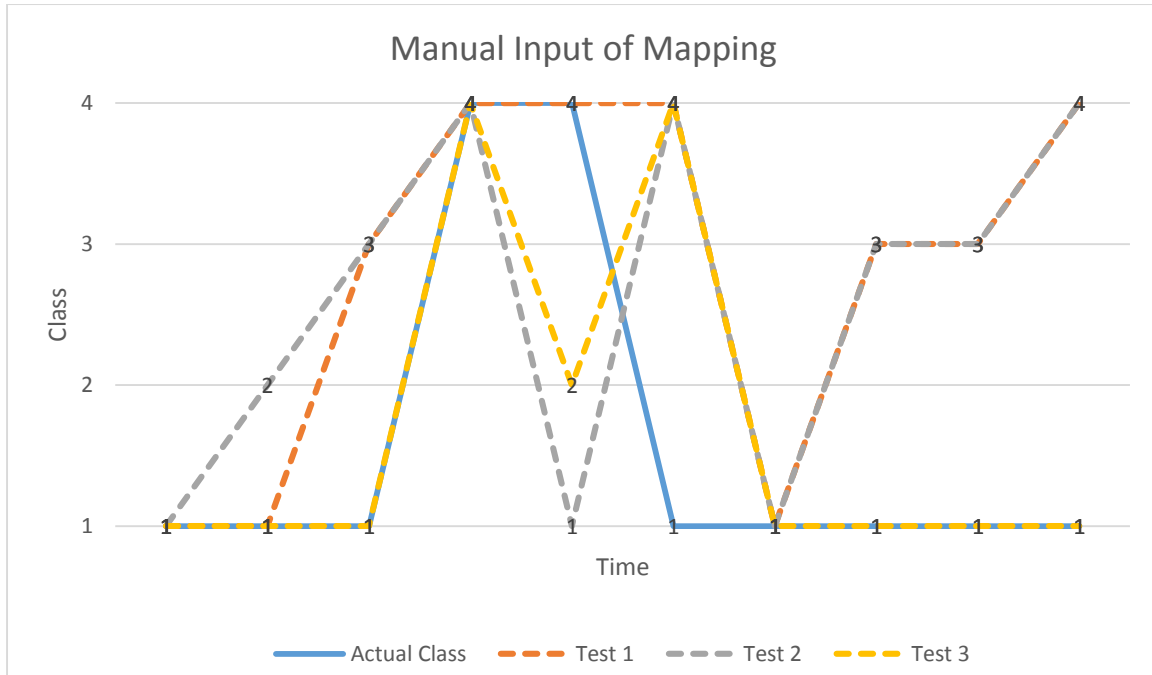


Figure 6.7: Comparison of actual class to classified class in manual input of mapping.

These graphs show promising results for using NeuCube as a BCI system to control an object. However, the author must also be able to test this study on someone else other than himself.

The classified data for self-generated mapping can be seen in Table 6.1, Table 6.2 & Table 6.3 and the classified data for Talairach coordinates can be seen in Table 6.4, Table 6.5 & Table 6.6. To make it more clear, the results have been presented in two graphs (Figure 6.6 and Figure 6.7), in the graphs blue line is the actual class that was used to move the EEGRotor. We can see that there are large differences between those two figures. The confusion matrices for the experiments are provided in 6.3.1 and 6.3.2.

The Left most part of each matrix is the correctly classified order number and top part is the incorrectly classified order.

6.3.1 Confusion matrix for Self-generated mapping

The matrix (**Figure 6.8**) for Test1 shows that for class 1 (Front) was correctly classified six times and incorrectly classified once to class 2 (Right) and class 4 (Left). There were no classifications for class 2 (Right) and class 3 (back). Class 4 (Left) was incorrectly classified once to class 1 (Front) and once to class 2 (Right). The accuracy for this test is 60%

		Predicted Class			
		Front	Right	Back	Left
Actual Class	Front	6	1	0	1
	Right	0	0	0	0
	Back	0	0	0	0
	Left	1	1	0	0

Figure 6.8: Test1 using self-generated mapping

The matrix (**Figure 6.9**) for Test2 shows that for class 1 was correctly classified three times and incorrectly classified twice to class 2 (Right), once to class 3 (Back) and twice to class 4 (Left). There were no classifications for class 2 (Right) and class 3 (Back). Class 4 was incorrectly classified once to class 1 and once to class 2. The accuracy for this test is 30%

		Predicted Class			
		Front	Right	Back	Left
Actual Class	Front	3	2	1	2
	Right	0	0	0	0
	Back	0	0	0	0
	Left	1	1	0	0

Figure 6.9: Test2 using self-generated mapping

The matrix (**Figure 6.10**) for Test3 shows that for class 1 was correctly classified five times and incorrectly classified twice to class 2 and once to class 3. There were no classifications for class 2 and class 3. Class 4 was incorrectly classified once to class 1 and once for class 2. The accuracy for this test is 50%

		Predicted Class			
		Front	Right	Back	Left
Actual Class	Front	5	2	1	0
	Right	0	0	0	0
	Back	0	0	0	0
	Left	1	1	0	0

Figure 6.10: Test2 using self-generated mapping

6.3.2 Confusion matrix for Talairach coordinates

The matrix (**Figure 6.11**) for Test1 shows that for the class 1 correctly classified three times and incorrectly classified thrice to class 3 and twice to class 4. There were no classifications for class 2 and class 3. Class 4 was correctly classified twice. The accuracy for this is 50%

		Predicted Class			
		Front	Right	Back	Left
Actual Class	Front	3	0	3	2
	Right	0	0	0	0
	Back	0	0	0	0
	Left	0	0	0	2

Figure 6.11: Test1 using Talairach coordinates

The matrix (**Figure 6.12**) for Test2 shows that for the class 1 was correctly classified two times and incorrectly classified once to class 2, thrice to class 3 and twice to class 4. There were no classifications for class 2 and class 3. Class 4 was correctly classified once and incorrectly classified once to class 1. The accuracy for this is 30%

		Predicted Class			
		Front	Right	Back	Left
Actual Class	Front	2	1	3	2
	Right	0	0	0	0
	Back	0	0	0	0
	Left	1	0	0	1

Figure 6.12: Test2 using Talairach coordinates

The matrix (**Figure 6.13**) for Test3 shows that the class 1 was correctly classified seven times and incorrectly classified once to class 4. There were no classifications for class 2 and class 3. Class 4 was incorrectly classified once for class 2 and correctly classified once to class 4. The accuracy for this is 80%

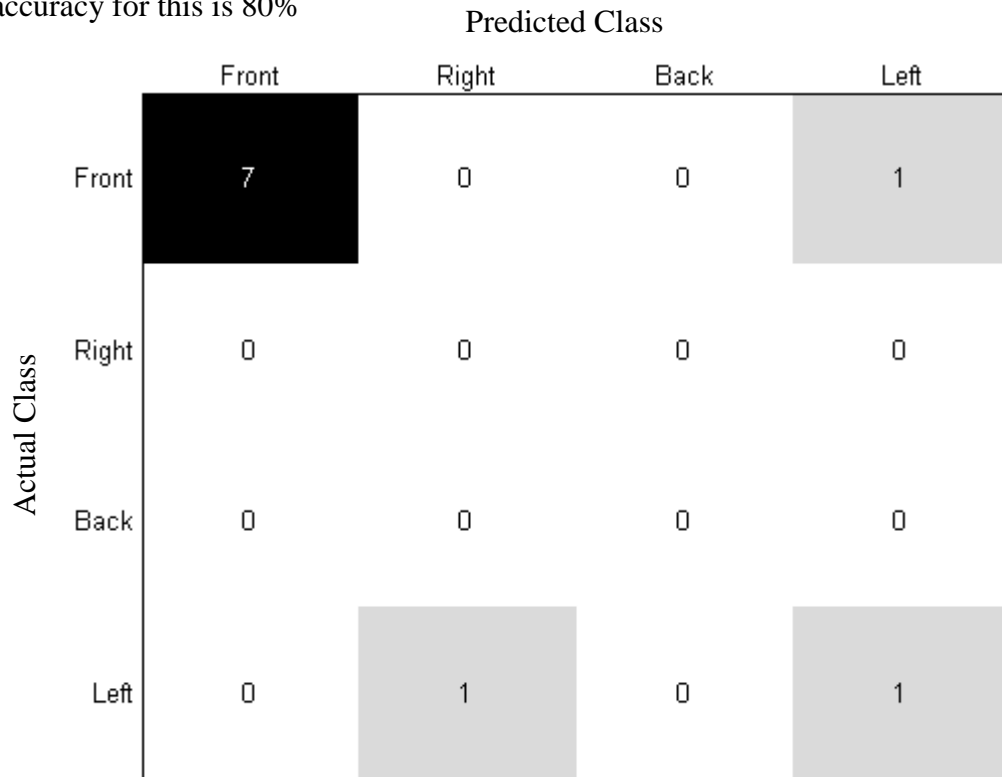


Figure 6.13: Test3 using Talairach coordinates

Self-generated mapping		Talairach coordinates	
Test's	Accuracy	Test's	Accuracy
Test 1	60%	Test 1	50%
Test 2	30%	Test 2	30%
Test 3	50%	Test 3	80%

Table 6.7: Accuracy results

6.4 Conclusion

In this chapter we have discussed how the implemented BCI EEGRotor works when it is trained and tested with EEG data. A detailed analysis of results obtained from using both self-generated mapping and Talairach coordinates mapping is presented. Also, graphical representations of neuron-to-neuron connections are presented.

Chapter 7: Conclusions

This chapter summarises the study and discusses its limitations. The contribution of this work is outlined and some directions for future work are also provided.

7.1 A Review of the Study

This thesis presents, a case study that makes use of the Emotiv EPOC EEG headset as a real-time EEG recorder, which is simple to use and affordable. Using the NeuCube architecture a new software called EEGRotor was developed that can acquire real-time data and classify it for the purpose of Quadcopter control. A detailed explanation of all developed modules is presented in the thesis.

This thesis starts with a literature review and an introduction to how the brain works. It also explains briefly the functions of the brain, what a neuron is and how neuron-to-neuron interaction occurs.

A literature review on BCI is presented.

Further, a literature review of supervised, unsupervised and filter algorithms are presented along with brief introduction to neural networks. Also, in this chapter all algorithms used in EEGRotor were explained in detail.

Finally, the implementation of a complete NeuCube suite for BCI called EEGRotor is assigned step by step.

Section 1.3 presents the research questions for this study. The first question is related to BCI and is answered in the Chapter 3:. The second question is about working and reliability of the Emotiv EEG device when compared to medical grade EEG devices. The answer to this question can be found in section 3.12.2. The third question is related to the ability of NeuCube to improve the control of a robot. The answer to this question is provided in section 6.3. The fourth question relates to whether NeuCube can be used to help paralysed people. While currently NeuCube is in its early stage of development, it is not quite fast enough due to hardware limitations. In future NeuCube implementation may be used to help a paralytic person. The fifth question is related to the accuracy of performance of the BCI framework; the proposed BCI framework works on dynamic data, however, EEG data collected from the brain with the 14 channel Emotiv device at 128 Hz has much lower resolution than data collected

with sophisticated medical EEG device. Regardless of that though, the results obtained in this study are promising. Finally, the sixth question related to the limitation of this approach is addressed in section 7.2.

7.2 Limitations

The limitations of this study are based on the following findings:

- The main limitation is the hardware speed;
- No parallel processing architecture was used. Which slows down the classification process.
- Classification can only start after the EEG data has been read. This causes a delay of five seconds for the next iteration to start.
- The SDK provided by the Emotiv Company uses old library, which can only be opened if Visual studio 2010 is installed.
- Due to the SDK restriction the NeuCube for BCI framework can only be run on Matlab R2013 family.

7.3 Contribution of this Study

The following are the contributions made in this study:

- Most of the BCI techniques used to control a robot do not have any visual output on how the neurons communicate with each other, but the NeuCube for BCI framework is able to show the real-time connections between the neurons.
- All previous experiments conducted with BCI to control a robot used medical grade EEG devices. In this study Emotiv EMPOC EEG headset with 14 channel was used. It was able to perform well.
- The EEGRotor State of Mind Module developed in this study is able to show state of mind over time. This module separates the raw data into Delta, Theta, Alpha and Beta bands.
- A virtual environment was developed for this study which is an exact replica of the KEDRI institute work space here at AUT. This allows the user (in this case the author) to feel as if the place is already known so they do not need much training.
- The classification is done in almost real time and is significantly faster than achieved by other techniques, and the key to this is the classification algorithm known as DeSNN (section 4.6).

- One of the main important feature is that the classified data can be retrained with the existing NeuCube framework, this would make the framework more reliable.
- The EEGRotor Training Module developed in this study can save the collected EEG data. Since it is independent of NeuCube, it could be used in other implementations as well.
- The user is able to control both number of collected samples as well as their length in seconds. The control panel used for implementing the data to classify is directly connected to the EEG headset so that there are no delays in receiving the data.

7.4 Future Work

The limitations discussed in section 7.2 could be addressed in future as outlined in sections 7.4.1 and 7.4.2 :

7.4.1 Future Work on the EEGRotor

- It is recommended that instead of using an Emotiv device with only 14-channels, future work should consider devices with a higher number of channels as this would increase the signal spatial resolution.
- Processing speed has been one of the main issues for real time classification of new input data. Therefore, it is highly recommended to change the platform from a conventional computer to a faster neuromorphic hardware or to use General-Purpose Computing on Graphics Processing Units (GPGPU). A neuromorphic hardware such as Spiking Neural Network Architecture (SpiNNaker) (*SpiNNaker Home Page*, n.d.).
- General computing processors are slower than GPU processors. When a user runs NeuCube architecture on a computer, computational process is slow due to the processors inability to perform faster. Instead, GPUs could be used that support Compute Unified Device Architecture (CUDA) and Open Computing Language (OpenCL) framework that can use GPU's ability to compute data.
- A new model could be developed to add to the NeuCube framework. The new Module (M10 in Figure 7.1) should be able to learn real time data from the EEG device and train the NeuCube.
- At the time of finalising this thesis, the author has already developed and tested this module.

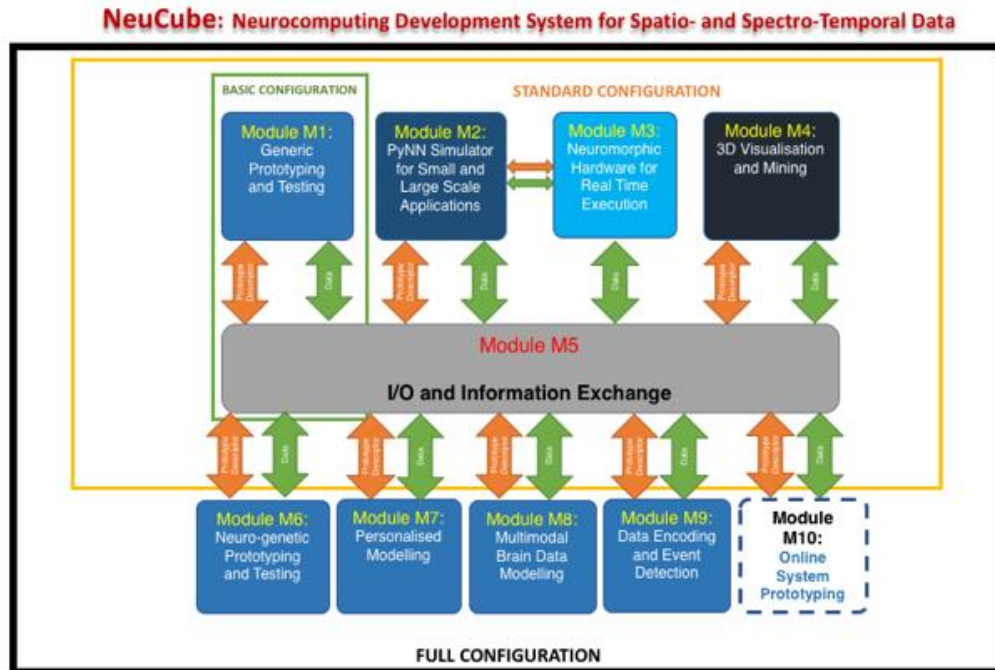


Figure 7.1: NeuCube configuration with a new Module M10 for online learning, developed by the author.

- Improved learning methods for BCI in real time (online) could be implemented.

7.4.2 Hardware Implementation of EEGRotor

A virtual environment is only good to test a software, but what if we use neuromorphic hardware to control a real live quadcopter. A plan that could be implemented in future studies is presented below:

7.4.2.1 Hardware Required

- Raspberry Pi
- Arduino
- IR distance sensor
- Quad rotor kit with flight controller

7.4.2.2 Blueprint

Figure 7.1 shows the dimensions of each parts that would be used to build the actual EEGRotor in future work.

Parts Name	Size in Millimetres
Fin	200 mm
Carbon fibre boom (each)	220 mm
Complete size of the EEGRotor	550 mm

Table 7.1: EEGRotor parts

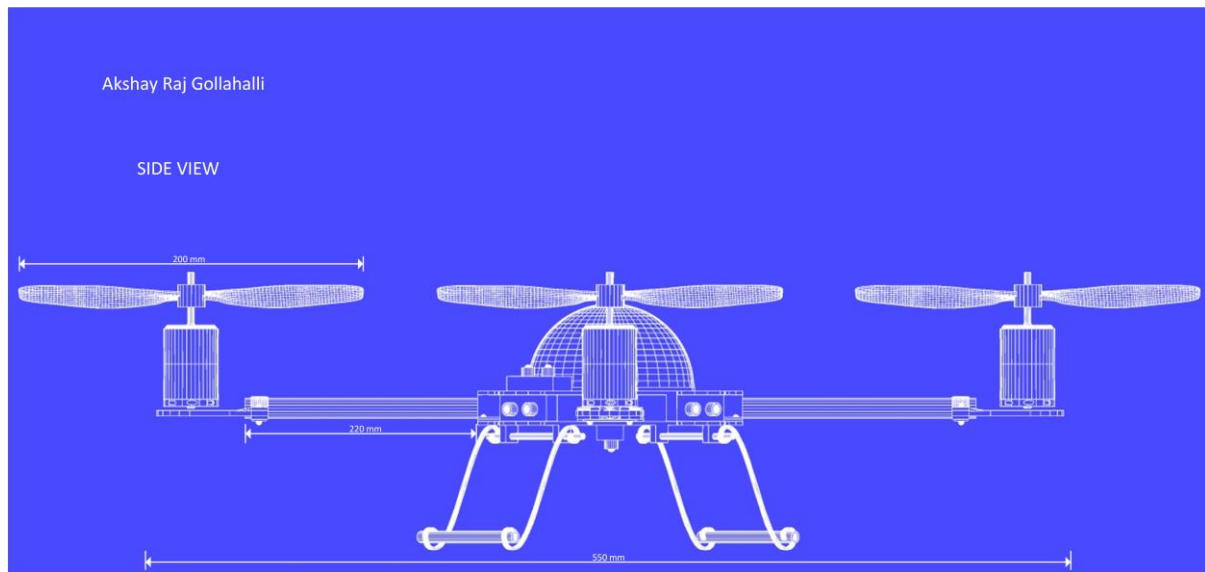


Figure 7.2: Side view of quadcopter

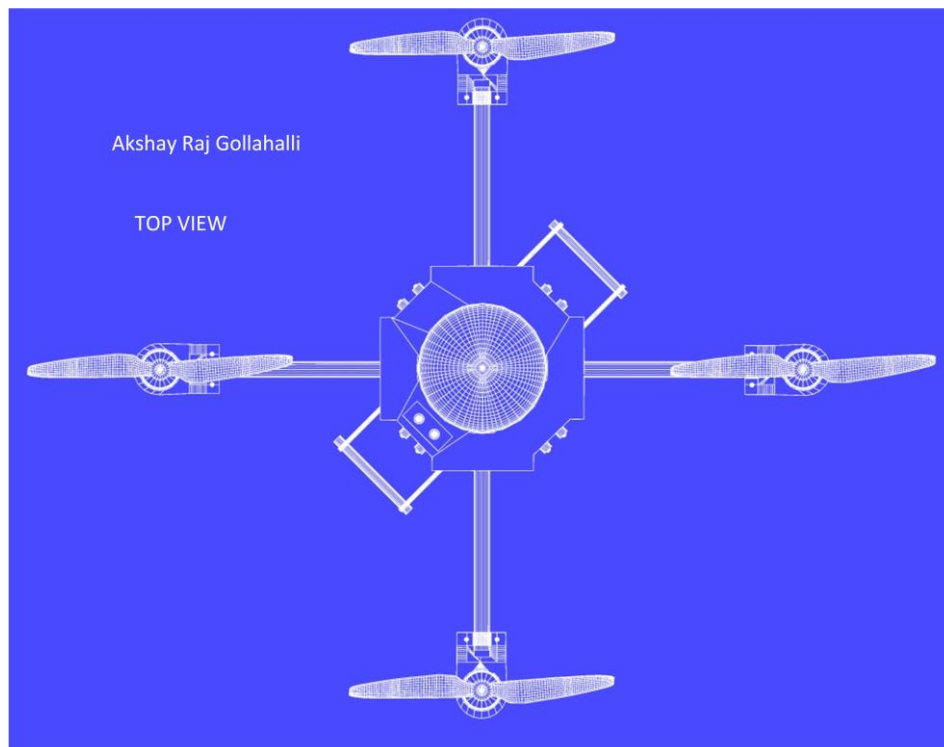
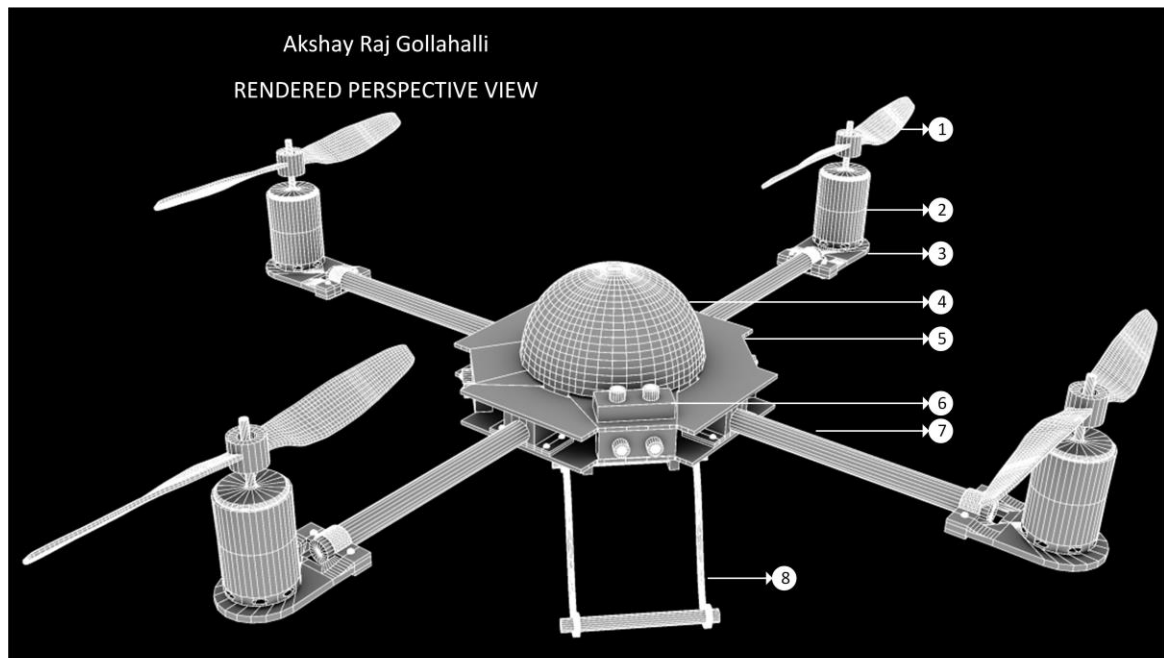


Figure 7.3: Top view quadcopter



- | | |
|---|-------------------------------------|
| ① Carbon Fiber Fin | ⑥ IR Sensor to measure the distance |
| ② High RPM Motor | ⑦ Carbon Fiber Boom |
| ③ Aluminum Motor Holder | ⑧ Landing gear |
| ④ Dome | |
| ⑤ Carbon Fiber Plates to hold electronics | |

Figure 7.4: Rendered perspective view quadcopter

7.4.2.3 Safety

This is always a big concern with Quadcopter. Safety precautions can be taken by not over charging the battery and adding ducts to the fins.

References

- Accelerometer. (2014). In *the Hutchinson unabridged encyclopedia with atlas and weather guide*. Retrieved from <https://networkservices.aut.ac.nz/ezproxy.cgi?url=http%3A%2F%2Fsearch.credoreference.com/ezproxy.aut.ac.nz%2Fcontent%2Fentry%2Fheliconhe%2Faccelerometer%2F0>
- Adiutori, E. F. (2005). FOURIER. *Mechanical Engineering*, 127(8), 30-31.
- Ahn, M., Lee, M., Choi, J., & Jun, S. C. (2014). A Review of Brain-Computer Interface Games and an Opinion Survey from Researchers, Developers and Users. *Sensors*, 14(8), 14601-14633.
- Al-ani, T., & Trad, D. (2010). *Signal Processing and Classification Approaches for Brain-computer Interface*: INTECH Open Access Publisher.
- Allaby. spinal cord: 'Oxford University Press'.
- Allaby. (2014). synapse: 'Oxford University Press'.
- Allen, C. (2014). How a Quadcopter Works.
- Altug, E., Ostrowski, J. P., & Mahony, R. (2002). Control of a quadrotor helicopter using visual feedback *IEEE*. Symposium conducted at the meeting of the Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on
- Anderson, J. A. (1995). *An Introduction to Neural Networks*: BRADFORD BOOK. Retrieved from <https://books.google.co.nz/books?id=AOaYQgAACAAJ>
- Anderson, J. A., & Rosenfeld, E. (1993). *Neurocomputing* (Vol. 2): MIT press.

- Bankman, I. (2000). *Handbook of Medical Imaging: Processing and Analysis Management*: Elsevier Science. Retrieved from <https://books.google.co.nz/books?id=UHkkPBnhT-MC>
- barometer. (2013). In *A Dictionary of Geology and Earth Sciences*. Retrieved from <http://www.oxfordreference.com/10.1093/acref/9780199653065.001.0001/acref-9780199653065-e-744>. doi:10.1093/acref/9780199653065.013.0744
- BCILAB. (2013). Retrieved from <http://scn.ucsd.edu/wiki/BCILAB>
- biofeedback. (2014). In *The Hutchinson Unabridged Encyclopedia with Atlas and Weather Guide*: Helicon. Retrieved from <https://networkservices.aut.ac.nz/ezproxy.cgi?url=http%3A%2F%2Fsearch.oxfordreference.com%2Fcontent%2Fentry%2Fheliconhe%2Fbiofeedback%2F0>
- Birbaumer, N. (2006). Breaking the silence: brain–computer interfaces (BCI) for communication and motor control. *Psychophysiology*, 43(6), 517-532.
- Birbaumer, N., Ghanayim, N., Hinterberger, T., Iversen, I., Kotchoubey, B., Kubler, A., . . . Flor, H. (1999). A spelling device for the paralysed [10.1038/18581]. *Nature*, 398(6725), 297-298.
- Birbaumer, N., Weber, C., Neuper, C., Buch, E., Haapen, K., & Cohen, L. (2006). Physiological regulation of thinking: brain–computer interface (BCI) research. *Progress in brain research*, 159, 369-391.
- Blankertz, B., Tangermann, M., Vidaurre, C., Fazli, S., Sannelli, C., Haufe, S., . . . Curio, G. (2010). The Berlin brain–computer interface: non-medical uses of BCI technology. *Frontiers in neuroscience*, 4.
- Brown, D. R. (1980). *Neurosciences for allied health therapies*: CV Mosby Co.
- Brown, R. W., Cheng, Y.-C. N., Haacke, E. M., Thompson, M. R., & Venkatesan, R. (2014). *Magnetic Resonance Imaging : Physical Principles and Sequence Design* (2 ed.). Retrieved from <http://AUT.ebiblib.com.au/patron/FullRecord.aspx?p=1676112>

- BruceBlaus. (2013). Anatomy of a multipolar neuron.
- Butterworth, S. (1930). On the theory of filter amplifiers. *Wireless Engineer*, 7(6), 536-541.
- Carter, R., & Frith, C. D. (1998). *Mapping the mind*: Univ of California Press.
- Cerebral Hemisphere. (2012). In *Mosby's dictionary of Medicine, Nursing & Health professions*. Retrieved from https://networkservices.aut.ac.nz/ezproxy.cgi?url=http%3A%2F%2Fsearch.credoreference.com/ezproxy.aut.ac.nz%2Fcontent%2Fentry%2Fehsmosbomed%2Fcerebral_hemisphere%2F0
- Colman, A. M. (2014a). alpha wave: Oxford University Press.
- Colman, A. M. (2014b). beta wave: Oxford University Press.
- Colman, A. M. (2014c). delta wave: Oxford University Press.
- Colman, A. M. (2014d). gamma wave: Oxford University Press.
- Colman, A. M. (2014e). longitudinal fissure: Oxford University Press.
- Colman, A. M. (2014f). *mu wave*. Retrieved from <http://www.oxfordreference.com/10.1093/acref/9780199534067.001.0001/acref-9780199534067-e-9357>. doi:10.1093/acref/9780199534067.013.9357
- Colman, A. M. (2014g). mu wave: Oxford University Press.
- Colman, A. M. (2014h). theta wave: Oxford University Press.
- Creel, D. J. (2012). Visually evoked potentials. *Webvision: The Organization of the Retina and Visual System*. Salt Lake City: University of Utah Health Sciences Center, 201-229.
- Di Sessa, A. A. (1985). A principled design for an integrated computational environment. *Human-computer interaction*, 1(1), 1-47.

diencephalon. (2014). In. Retrieved from <http://www.oxfordreference.com/10.1093/acref/9780199684274.001.0001/acref-9780199684274-e-2556>. doi:10.1093/acref/9780199684274.013.2556

Drachman, D. A. (2005). Do we have brain to spare? *Neurology*, 64(12), 2004-2005.

Duncan, D. E. (2012). The Brain-Computer Interface That Let a Quadriplegic Woman Move a Cup. Retrieved from <http://www.theatlantic.com/health/archive/2012/05/the-brain-computer-interface-that-let-a-quadriplegic-woman-move-a-cup/257275/>

Duvinage, M., Castermans, T., Dutoit, T., Petieau, M., Hoellinger, T., Saedeleer, C. D., . . . Cheron, G. (2012). A P300-based quantitative comparison between the Emotiv Epoc headset and a medical EEG device. *Biomedical Engineering*, 765, 2012-2764.

Electric Current. (2014). In *The Hutchinson unabridged encyclopedia with atlas and weather guide*. Retrieved from <https://networkservices.aut.ac.nz/ezproxy.cgi?url=http%3A%2F%2Fsearch.oxfordreference.com/10.1093/acref/9780199684274.001.0001/acref-9780199684274-e-2556>

Electroencephalography. (2015). In *The Columbia Encyclopedia*: Columbia University Press. Retrieved from <https://networkservices.aut.ac.nz/ezproxy.cgi?url=http%3A%2F%2Fsearch.oxfordreference.com/10.1093/acref/9780199684274.001.0001/acref-9780199684274-e-2556>

Epoc. Retrieved from <https://emotiv.com/epoc.php>

Eric R Kandel, J. H. S. (1985). *Principles of Neural Science*: Elsevier.

evoked potential. (2009). In *The Penguin Dictionary of Psychology*: Penguin. Retrieved from <https://networkservices.aut.ac.nz/ezproxy.cgi?url=http%3A%2F%2Fsearch.oxfordreference.com/10.1093/acref/9780199684274.001.0001/acref-9780199684274-e-2556>

- Farwell, L. A., & Donchin, E. (1988). Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and clinical Neurophysiology*, 70(6), 510-523.
- Ferree, T. C., Luu, P., Russell, G. S., & Tucker, D. M. (2001). Scalp electrode impedance, infection risk, and EEG data quality. *Clinical Neurophysiology*, 112(3), 536-544. doi:[http://dx.doi.org/10.1016/S1388-2457\(00\)00533-2](http://dx.doi.org/10.1016/S1388-2457(00)00533-2)
- Finger, S. (2001). *Origins of neuroscience: a history of explorations into brain function*: Oxford University Press.
- Fox, W. L. (1984). *Dandy of Johns Hopkins*: Williams & Wilkins.
- Freeman, W. J. (2000). *Neurodynamics: An Exploration in Mesoscopic Brain Dynamics: An Exploration in Mesoscopic Brain Dynamics*: Springer Science & Business Media.
- Graimann, B., Allison, B., & Pfurtscheller, G. (2010). Brain–computer interfaces: A gentle introduction. In *Brain-Computer Interfaces* (pp. 1-27): Springer.
- Greenfield, S. (1997). The human brain: A guided tour. *London: Weidenfield and Nicolson*.
- Griggs, R. A. (2010). *Psychology: A concise introduction*: Macmillan.
- Guenther, F. H., Brumberg, J. S., Wright, E. J., Nieto-Castanon, A., Tourville, J. A., Panko, M., . . . Andreasen, D. S. (2009). A wireless brain-machine interface for real-time speech synthesis. *PloS one*, 4(12), e8218.
- Haken, H. (2006). *Brain dynamics: synchronization and activity patterns in pulse-coupled neural nets with delays and noise*: Springer Science & Business Media.
- Haken, H. (2007). *Brain dynamics: an introduction to models and simulations*: Springer Science & Business Media.

- Hardesty, L. (2015). Thumbnail track pad. *MIT News Office*. Retrieved from <https://newsoffice.mit.edu/2015/wearable-thumbnail-sensor-controls-digital-devices-0417>
- Hebb, D. O. (2005). *The organization of behavior: A neuropsychological theory*: Psychology Press.
- Herwig, U., Satrapi, P., & Schönfeldt-Lecuona, C. (2003). Using the international 10-20 EEG system for positioning of transcranial magnetic stimulation. *Brain topography*, 16(2), 95-99.
- HobbyKing HKPilot Mega Mini Combo Flight Controller GPS and Power Module. Retrieved from http://www.hobbyking.com/hobbyking/store/_51497_HobbyKing_HKPilot_Mega_Mini_Combo_Flight_Controller_GPS_and_Power_Module.html
- How MRI Works*. Retrieved from <http://www.bitterrootimaging.com/MRI-workings.html>
- Jan, Y.-N., & Jan, L. Y. (2001). Dendrites. *Genes & development*, 15(20), 2627-2641.
- Kalender, W. A. (2011). *Computed tomography: fundamentals, system technology, image quality, applications*: John Wiley & Sons.
- Kasabov, N. (2013). *Springer Handbook of Bio-/neuro-informatics*: Springer Science & Business Media.
- Kasabov, N., Dhoble, K., Nuntalid, N., & Indiveri, G. (2013a). Dynamic evolving spiking neural networks for on-line spatio- and spectro-temporal pattern recognition. *Neural Networks*, 41(0), 188-201. doi:<http://dx.doi.org/10.1016/j.neunet.2012.11.014>
- Kasabov, N., Dhoble, K., Nuntalid, N., & Indiveri, G. (2013b). Dynamic evolving spiking neural networks for on-line spatio-and spectro-temporal pattern recognition. *Neural Networks*, 41, 188-201.

- Kasabov, N. K. (2014). NeuCube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data. *Neural Networks*, 52, 62-76.
- Kittler, J. (1997). Statistical classification. *Vistas in Astronomy*, 41(3), 405-410.
doi:[http://dx.doi.org/10.1016/S0083-6656\(97\)00045-7](http://dx.doi.org/10.1016/S0083-6656(97)00045-7)
- Krantz, J. H. (1999). *Receptive Field Tutorial*. Retrieved from <http://psych.hanover.edu/Krantz/receptive/>
- L, D. (2005). Blood Groups and Red Cell Antigens [Internet]. In. Bethesda (MD): National Center for Biotechnology Information (US). Retrieved from <http://www.ncbi.nlm.nih.gov/books/NBK2263/>
- Lee, J.-H., Ryu, J., Jolesz, F. A., Cho, Z.-H., & Yoo, S.-S. (2009). Brain-machine interface via real-time fMRI: preliminary study on thought-controlled robotic arm. *Neuroscience letters*, 450(1), 1-6.
- Llinas, R. (1990). Intrinsic electrical properties of mammalian neurons and CNS function. *Fidia Research Foundation Neuroscience Award Lectures*, 4, 1-10.
- Lotte, F., Congedo, M., Lécuyer, A., & Lamarche, F. (2007). A review of classification algorithms for EEG-based brain-computer interfaces. *Journal of neural engineering*, 4.
- Madakasira, P. V., Simkins, R., Narayanan, T., Dunigan, K., Poelstra, R. J., & Mantil, J. (2002). Cortical dysplasia localized by [11C] methionine positron emission tomography: case report *American journal of neuroradiology* (Vol. 23, pp. 844-846).
- Martin, & McFerran. biofeedback: 'Oxford University Press'.
- Martin, & McFerran. Electroencephalography: 'Oxford University Press'.
- Martin, E. A. (2010). *Concise medical dictionary*: Oxford University Press.

Martin Weil, & Randolph, E. (1994). Richard M. Nixon, 37th President, Dies. *Washington post*. Retrieved from <http://www.washingtonpost.com/wp-srv/national/longterm/watergate/stories/nixobit.htm>

Medulla Oblongata. (2012). In *Mosby's dictionary of Medicine, Nursing & Health professions*. Retrieved from https://networkservices.aut.ac.nz/ezproxy.cgi?url=http%3A%2F%2Fsearch.credoreference.com.ezproxy.aut.ac.nz%2Fcontent%2Fentry%2Fehsmosbymed%2Fmedulla_oblongata%2F0

Midbrain. (2012). In *Mosby's dictionary of Medicine, Nursing & Health professions*. Retrieved from <https://networkservices.aut.ac.nz/ezproxy.cgi?url=http%3A%2F%2Fsearch.credoreference.com.ezproxy.aut.ac.nz%2Fcontent%2Fentry%2Fehsvetdict%2Fmidbrain%2F0>

Morgan, S., Hansen, J., & Hillyard, S. (1996). Selective attention to stimulus location modulates the steady-state visual evoked potential. *Proceedings of the National Academy of Sciences*, 93(10), 4770-4774.

Moseler, O., & Isermann, R. (1998). Model-based fault detection for a brushless DC motor using parameter estimation *IEEE. Symposium* conducted at the meeting of the Industrial Electronics Society, 1998. IECON'98. Proceedings of the 24th Annual Conference of the IEEE

Motor Evoked Potential (MEP). (2009). In M. Binder, N. Hirokawa, & U. Windhorst (Eds.), *Encyclopedia of Neuroscience* (pp. 2438-2438): Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-540-29678-2_3594. doi:10.1007/978-3-540-29678-2_3594

Naik, G. R. (2014). *Emerging Theory and Practice in Neuroprosthetics*: IGI Global.

Neuroimaging. (2012). In N. Seel (Ed.), *Encyclopedia of the Sciences of Learning* (pp. 2451-2451): Springer US. Retrieved from http://dx.doi.org/10.1007/978-1-4419-1428-6_2291. doi:10.1007/978-1-4419-1428-6_2291

- Nijholt, A., Bos, D. P.-O., & Reuderink, B. (2009). Turning shortcomings into challenges: Brain-computer interfaces for games. *Entertainment Computing*, 1(2), 85-94. doi:<http://dx.doi.org/10.1016/j.entcom.2009.09.007>
- Norris, S. L. (1995). Neurofeedback: One instrument in the orchestra. *Journal of Neurotherapy*, 1(2), 74-76.
- O'Neil, D. (2014). *Blood Components*. Retrieved from http://anthro.palomar.edu/blood/blood_components.htm
- Obermaier, B., Neuper, C., Guger, C., & Pfurtscheller, G. (2001). Information transfer rate in a five-classes brain-computer interface. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 9(3), 283-288.
- Paugam-Moisy, H., & Bohte, S. (2012). Computing with spiking neuron networks. In *Handbook of natural computing* (pp. 335-376): Springer.
- Pivik, R. T., Broughton, R. J., Coppola, R., Davidson, R. J., Fox, N., & Nuwer, M. R. (1993). Guidelines for the recording and quantitative analysis of electroencephalographic activity in research contexts. *Psychophysiology*, 30(6), 547-558.
- Pons. (2012). In *Mosby's dictionary of Medicine, Nursing & Health professions*. Philadelphia. Retrieved from <https://networkservices.aut.ac.nz/ezproxy.cgi?url=http%3A%2F%2Fsearch.credoreference.com.ezproxy.aut.ac.nz%2Fcontent%2Fentry%2Fehsmosbymed%2Fpons%2F0>
- Positron Emission Tomography (PET Scan). Retrieved from http://www.hopkinsmedicine.org/healthlibrary/test_procedures/neurological/positron_emission_tomography_pet_scan_92,p07654/
- Potential, Electric. (2014). In *The Hutchinson unabridged encyclopedia with atlas and weather guide*. Retrieved from https://networkservices.aut.ac.nz/ezproxy.cgi?url=http%3A%2F%2Fsearch.credoreference.com.ezproxy.aut.ac.nz%2Fcontent%2Fentry%2Fheliconhe%2Fpotential_electric%2F0

- Pounds, P., Mahony, R., & Corke, P. (2010). Modelling and control of a large quadrotor robot. *Control Engineering Practice*, 18(7), 691-699.
- Prueckl, R., & Guger, C. (2009). A brain-computer interface based on steady state visual evoked potentials for controlling a robot. In *Bio-Inspired Systems: Computational and Ambient Intelligence* (pp. 690-697): Springer.
- Ranky, G., & Adamovich, S. (2010). Analysis of a commercial EEG device for the control of a robot arm *IEEE*. Symposium conducted at the meeting of the Bioengineering Conference, Proceedings of the 2010 IEEE 36th Annual Northeast
- Renard, Y., Lotte, F., Gibert, G., Congedo, M., Maby, E., Delannoy, V., . . . Lécuyer, A. (2010). OpenViBE: an open-source software platform to design, test, and use brain-computer interfaces in real and virtual environments. *Presence: teleoperators and virtual environments*, 19(1), 35-53.
- Roberto Hornero, R. C. D. Á. (2012). Brain Computer Interface (BCI) systems applied to cognitive training and home automation control to offset the effects of ageing. (08). Retrieved from http://www.fgcsic.es/lychnos/en_en/articles/Brain-Computer-Interface
- Roebuck, K. (2012). *Tangible User Interfaces: High-impact Emerging Technology-What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors*: Emereo Publishing.
- Romano, L. (1995). Riding accident paralyzes actor christopher reeve. *Washington post*.
- Roy, C. S., & Sherrington, C. S. (1890). On the Regulation of the Blood-supply of the Brain. *The Journal of Physiology*, 11(1-2), 85-158.117.
- Ruiz, S., Rana, M., Sass, K., Kircher, T., Birbaumer, N., & Sitaram, R. (2011). Brain network connectivity and behaviour enhancement: a fMRI-BCI study Symposium conducted at the meeting of the 17th Annual Meeting of the Organization for Human Brain Mapping

- Sa, I., & Corke, P. (2011). Estimation and control for an open-source quadcopter Symposium conducted at the meeting of the Proceedings of the Australasian Conference on Robotics and Automation 2011
- Schalk, G., & Leuthardt, E. C. (2011). Brain-computer interfaces using electrocorticographic signals. *Biomedical Engineering, IEEE Reviews in*, 4, 140-154.
- Schalk, G., McFarland, D. J., Hinterberger, T., Birbaumer, N., & Wolpaw, J. R. (2004). BCI2000: a general-purpose brain-computer interface (BCI) system. *Biomedical Engineering, IEEE Transactions on*, 51(6), 1034-1043.
- Sjöström, J., & Gerstner, W. (2010). Spike-timing dependent plasticity. *Scholarpedia*, 5, 1362.
- Spinal cord. (2014). *The Hutchinson unabridged encyclopedia with atlas and weather guide*. Retrieved from https://networkservices.aut.ac.nz/ezproxy.cgi?url=http%3A%2F%2Fsearch.credoreference.com.ezproxy.aut.ac.nz%2Fcontent%2Fentry%2Fheliconhe%2Fspinal_cord%2F0
- SpiNNaker Home Page*. (n.d.). Retrieved from <http://apt.cs.manchester.ac.uk/projects/SpiNNaker/>
- Strang, G., & Nguyen, T. (1996). *Wavelets and filter banks*: SIAM.
- Stufflebeam, R. (2008). Neurons, synapses, action potentials, and neurotransmission. *Consortium on Cognitive Science Instruction*.
- synaptic cleft. (2014). In Allaby, M.(Ed.), *A Dictionary of Zoology*. Retrieved from <http://www.oxfordreference.com/10.1093/acref/9780199684274.001.0001/acref-9780199684274-e-8648>. doi:10.1093/acref/9780199684274.013.8648
- Teo, E., Huang, A., Lian, Y., Guan, C., Li, Y., & Zhang, H. (2006). Media communication center using brain computer interface *IEEE*. Symposium conducted at the meeting of the Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE

Tsihrintzis, G. A., & Virvou, M. (2010). *Multimedia Services in Intelligent Environments: Integrated Systems*: Springer Berlin Heidelberg. Retrieved from <https://books.google.co.nz/books?id=6OIAgf8z8BoC>

Turner, A. (2015). Neuron Model.

Vallabhaneni, A., Wang, T., & He, B. (2005). Brain—computer interface. In *Neural engineering* (pp. 85-121): Springer.

Veigl, C. *BrainBay - an OpenSource Biosignal project*. Retrieved from <http://www.shifz.org/brainbay/>

Wang, M., Daly, I., Allison, B. Z., Jin, J., Zhang, Y., Chen, L., & Wang, X. (2015). A new hybrid BCI paradigm based on P300 and SSVEP. *Journal of Neuroscience Methods*, 244(0), 16-25. doi:<http://dx.doi.org/10.1016/j.jneumeth.2014.06.003>

Wang, P. T., King, C. E., Do, A. H., & Nenadic, Z. (2012). Pushing the communication speed limit of a noninvasive bci speller. *arXiv preprint arXiv:1212.0469*.

Wang, Y., Gao, X., Hong, B., Jia, C., & Gao, S. (2008). Brain-computer interfaces based on visual evoked potentials. *Engineering in Medicine and Biology Magazine, IEEE*, 27(5), 64-71.

Weisstein, E. W. Discrete Fourier Transform. Retrieved from <http://mathworld.wolfram.com/DiscreteFourierTransform.html>

Weisstein, E. W. Fast Fourier Transform. Retrieved from <http://mathworld.wolfram.com/FastFourierTransform.html>

. *What is ALS?* : ALS Association. Retrieved from <http://www.alsa.org/about-als/what-is-als.html>

What is fMRI? Retrieved from <http://fmri.ucsd.edu/Research/whatisfmri.html>

- Wilcock, W. 10. Practical Aspects of Filtering. Retrieved from http://www.ocean.washington.edu/courses/ess522/lectures/10_filtering.pdf
- . Wireless EEG system / 4-ch MPEG-4 compression / ambulatory. In wireless-eeg-system-4-ch-mpeg-4-compression-ambulatory-90245-8051595.jpg (Ed.): medicalexpo.
- Wolf, C. G. (2013). Brain-Computer Interfaces Unlock the World for People with Paralysis. Retrieved from <http://alsn.mda.org/article/brain-computer-interfaces-unlock-world-people-paralysis>
- Wolpaw, J., & Wolpaw, E. W. (2011). *Brain-computer interfaces: principles and practice*: Oxford University Press.
- Wolpaw, J. R. (2014). The BCI endeavor and the mission of this new journal. *Brain-Computer Interfaces*, 1(1), 2-4.
- Wolpaw, J. R., McFarland, D. J., Neat, G. W., & Forneris, C. A. (1991). An EEG-based brain-computer interface for cursor control. *Electroencephalography and clinical Neurophysiology*, 78(3), 252-259.
- Wysoski, S. G., Benuskova, L., & Kasabov, N. (2006). Adaptive learning procedure for a network of spiking neurons and visual pattern recognition *Springer*. Symposium conducted at the meeting of the Advanced Concepts for Intelligent Vision Systems, Berlin Heidelberg.
- Yoh, M.-S., Kwon, J., & Kim, S. (2010). *NeuroWander: a BCI game in the form of interactive fairy tale*. presented at the meeting of the Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing - Adjunct, Copenhagen, Denmark. doi:10.1145/1864431.1864450

Appendix A: Code for EEGRotor

testing.m

```
function varargout = testing(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @testing_OpeningFcn, ...
                  'gui_OutputFcn',  @testing_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...
                  'gui_Callback',    []);

if nargin && ischar(varargin)
    gui_State.gui_Callback = str2func(varargin);
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before testing is made visible.
function testing_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for testing
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = testing_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

% --- Executes on button press in browse.
function browse_Callback(hObject, eventdata, handles)

[FileName,PathName] = uigetfile('*.mat','Select the MATLAB data file');

full_name = strcat(PathName,FileName);

setappdata(0,'fileName_testing',FileName);
setappdata(0,'pathName_testing',PathName);
set(handles.rl,'String',full_name);
```



```

function rl_Callback(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in next.
function next_Callback(hObject, eventdata, handles)

uiwait(msgbox('This window will close, click on "Training& Validation"
button','Closing'));

close(handles.testing_fig);

% --- Executes on button press in dynamic.
function dynamic_Callback(hObject, eventdata, handles)

uiwait(msgbox('This window will close now and click on the start button on
the EEG Control Panel','Closing'));

close(handles.testing_fig);

% --- Executes when user attempts to close testing_fig.
function testing_fig_CloseRequestFcn(hObject, eventdata, handles)

delete(hObject);

```

training.m

```

function varargout = training(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @training_OpeningFcn, ...
                  'gui_OutputFcn',  @training_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargin
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function training_OpeningFcn(hObject, eventdata, handles, varargin)

global samples_output time;
%

```

```

samples_output = 20;
set(handles.samples, 'String', num2str(samples_output));
guidata(hObject, handles);

time = 2;
set(handles.timer, 'String', num2str(time));

training_folder = exist('training', 'dir');
if training_folder ~= 7
    mkdir('training');
end

class1_folder = exist('training\class_1', 'dir');
class2_folder = exist('training\class_2', 'dir');
class3_folder = exist('training\class_3', 'dir');
class4_folder = exist('training\class_4', 'dir');

if class1_folder ~= 7
    mkdir('training\class_1');
end
if class2_folder ~= 7
    mkdir('training\class_2');
end
if class3_folder ~= 7
    mkdir('training\class_3');
end
if class4_folder ~= 7
    mkdir('training\class_4');
end

x = [0;1;0;-1];
y = [1;0;-1;0];

location={};
s=cell(1,4);
for a = 1:4
    location{1} = sprintf('UP');
    location{2} = sprintf('RIGHT');
    location{3} = sprintf('DOWN');
    location{4} = sprintf('LEFT');
    n = location{a};
    s(a)=strread(sprintf(n), '%s', 'delimiter', '');
end

set(handles.viewer_training, 'YTick', []);
set(handles.viewer_training, 'XTick', []);
scatter(x,y, 'filled')
text((x-.1), (y+.2), s, 'color', [1,0,0]);
hold on;

set(handles.viewer_training, 'YTick', []);
set(handles.viewer_training, 'XTick', []);

text_input = 'Enter number of samples needed for each class and then click
on start, click on stop to stop recording EEG data. You can also input your
own data by click on browse button.';

```

```

set(handles.current_pos, 'String', text_input);

handles.output = hObject;

guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = training_OutputFcn(hObject, eventdata, handles)
clear global
varargout{1} = handles.output;

function start_training_Callback(hObject, eventdata, handles)
global samples_output h time final_class final_label;
set(handles.viewer_training, 'YTick', []);
set(handles.viewer_training, 'XTick', []);
% Create axes
axes(handles.viewer_training);

x = [0;1;0;-1];
y = [1;0;-1;0];

h = EmotivEEG;
h.Run;

class1 = zeros(128, 25, samples_output);
class1_label = zeros(1, samples_output);
class2 = zeros(128, 25, samples_output);
class2_label = zeros(1, samples_output);
class3 = zeros(128, 25, samples_output);
class3_label = zeros(1, samples_output);
class4 = zeros(128, 25, samples_output);
class4_label = zeros(1, samples_output);

for k = 1:samples_output

    location={};
    s=cell(1,4);
    for a = 1:4
        location{1} = sprintf('UP');
        location{2} = sprintf('RIGHT');
        location{3} = sprintf('DOWN');
        location{4} = sprintf('LEFT');
        n = location{a};
        s(a)=strread(sprintf(n), '%s', 'delimiter', '');
    end

    set(handles.viewer_training, 'YTick', []);
    set(handles.viewer_training, 'XTick', []);
    scatter(x,y, 'filled')
    text((x-.1), (y+.2), s, 'color', [1,0,0]);
    hold on;

    tic
    for s=1:numel(x)

```

```

        samples_class_text = sprintf('Samples: %s \n Class: %s', num2str(k),
num2str(s));
        set(handles.current_pos, 'String', samples_class_text, 'FontSize',
15);

        set(handles.viewer_training, 'YTick', []);
        set(handles.viewer_training, 'XTick', []);
        scatter(x(s), y(s), 400, 'MarkerFaceColor', [0 0 0]);
        drawnow;
        if s == 1
            class1(:, :, k) = h.data;
            class1_label(:, k) = 1;
        elseif s == 2
            class2(:, :, k) = h.data;
            class2_label(:, k) = 2;
        elseif s == 3
            class3(:, :, k) = h.data;
            class3_label(:, k) = 3;
        elseif s == 4
            class4(:, :, k) = h.data;
            class4_label(:, k) = 4;
        end
        pause(time);
    end
    toc
    cla;
    set(handles.current_pos, 'String', 'Done!');
end

save('training\class_1\class1.mat', 'class1', 'class1_label');
save('training\class_2\class2.mat', 'class2', 'class2_label');
save('training\class_3\class3.mat', 'class3', 'class3_label');
save('training\class_4\class4.mat', 'class4', 'class4_label');

final_class = cat(3, class1, class2, class3);
final_label = cat(2, class1_label, class2_label, class3_label);

save('training\final_eeg.mat', 'final_class', 'final_label');

FileName = 'final_eeg.mat';
[stat, struc] = fileattrib('training');
i_PathName = struc.Name;

PathName = strcat(i_PathName, '\');

% b = open('training\');

setappdata(0, 'fileName_training', FileName);
setappdata(0, 'pathName_training', PathName);

h.delete;

function stop_training_Callback(hObject, eventdata, handles)
global h final_class final_label;

h.Stop;

function samples_Callback(hObject, eventdata, handles)

```

```

global samples_output;

samples_output = str2double(get(hObject,'String'));


function samples_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

[FileName,PathName] = uigetfile('*.mat','Select the MATLAB data file');
setappdata(0,'fileName_training',FileName);
setappdata(0,'pathName_training',PathName);


function training_figure_CloseRequestFcn(hObject, eventdata, handles)
global h;

if libisloaded('edk')
    h.delete
end

delete(hObject);


function timer_Callback(hObject, eventdata, handles)

global time;

time = str2double(get(hObject,'String'));


function timer_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

emotion.m

```

function [ ] = state( input_arguments )

eeg_data = input_arguments;

Number=4;
hz = 128;

```

```

time = 0:1/hz:(length(eeg_data)-1)*1/hz;

% delta wave
Wn_delta=5/hz;

[b,a] = butter(Number,Wn_delta);
delta = filter(b,a,eeg_data);

% theta wave

W1 = 8/hz;
W2 = 14/hz;
Wn_theta = [W1 W2];
[c,d] = butter(Number,Wn_theta);
theta = filter(c,d,eeg_data);

% alpha wave

W3 = 16/hz;
W4 = 22/hz;
Wn_alpha = [W3 W4];
[e,f] = butter(Number,Wn_alpha);
alpha = filter(e,f,eeg_data);

% beta wave

W5 = 22/hz;
W6 = 80/hz;
Wn_beta = [W5 W6];
[g,h] = butter(Number,Wn_beta);
beta = filter(g,h,eeg_data);

%% EEG Plotting

set(gcf, 'Position', [1 1 1920 1080])
hold on
title('EEG Wave Patterns')
subplot(5,1,1), plot(time, delta)
    xlabel('Time(s)')
    ylabel('Amplitude')
    title('Delta Waves')
subplot(5,1,2), plot(time, theta)
    xlabel('Time(s)')
    ylabel('Amplitude')
    title('Theta Waves')
subplot(5,1,3), plot(time, alpha)
    xlabel('Time(s)')
    ylabel('Amplitude')
    title('Alpha Waves')
subplot(5,1,4), plot(time, beta)
    xlabel('Time(s)')
    ylabel('Amplitude')
    title('Beta Waves')

```

```
hold off
```

```
end
```

eeg_live.m

```
function varargout = eeg_live(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @eeg_live_OpeningFcn, ...
                  'gui_OutputFcn',  @eeg_live_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function eeg_live_OpeningFcn(hObject, eventdata, handles, varargin)

lmno = getappdata(0,'eegdata');
threshhold = getappdata(0,'threshhold');

spk=diff(lmno,1,1);
spk1=spk;

a = (mean(abs(spk1),1)+std(abs(spk1),0,1))';
disp(a)

x = [-30;-50;-40;-60;-60;-60;-30;30;60;60;60;40;50;30];
y = [50;30;30;0;20;-60;-80;-80;-60;20;0;30;30;50];
z = [30;0;40;30;0;0;10;10;0;0;30;40;0;30];

location={};
s=cell(1,14);
for a = 1:14
    location{1} = sprintf('AF3');
    location{2} = sprintf('F7');
    location{3} = sprintf('F3');
    location{4} = sprintf('FC5');
    location{5} = sprintf('T7');
    location{6} = sprintf('P7');
    location{7} = sprintf('O1');
    location{8} = sprintf('O2');
    location{9} = sprintf('P8');
    location{10} = sprintf('T8');
    location{11} = sprintf('FC6');
    location{12} = sprintf('F4');
    location{13} = sprintf('F8');
```

```

        location{14} = sprintf('AF4');
        n = location{a};
        s(a)=strread(sprintf(n), '%s', 'delimiter', '');
    end
    % the plot
    scatter3(-x,-y,-z, 'filled'); % <- NOT <plot3>
    text(-(x+.3), -(y-.5), -z, s, 'color', [1,0,0]);
    view(115,18)

handles.output = hObject;

guidata(hObject, handles);

function varargout = eeg_live_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function pushbutton1_Callback(hObject, eventdata, handles)
close();

```

neucube.m

```

function start_eeg_dynamic_Callback(hObject, eventdata, handles)
global emotiv_true_false h;
emotiv_true_false = true;

set(handles.disconnect, 'enable', 'on');
set(handles.stop_eeg_dynamic, 'enable', 'on');
set(handles.disconnect, 'enable', 'on');

axes(handles.eeg_dynamic)
h = EmotiveEEG;
h.Run
a=2;
while (emotiv_true_false)
    out = nan([size(h.data),4]);
    for k = 1:a
        out(:, :, k) = h.data + 1;
        plot(out(:, :, k));
    %         handles.newhandel=out(:, :, k); % send it to eeg_live
        setappdata(0, 'eegdata', out(:, :, k));
        pause(0.5);
    end
    a=a+1;
end
h.Stop;
set(handles.eeg_final, 'visible', 'on');
set(handles.activation_plot, 'visible', 'off');
set(handles.emo_plot, 'visible', 'off');
for eeg = 1:size(out(:, :, :), 3)
    eeg_output_1d = permute(out, [1 3 2]);
    eeg_output_1d = reshape(eeg_output_1d, [], size(out, 2), 1);
    plot(eeg_output_1d);
    assignin('base', 'eeg_output_1d', eeg_output_1d)
end

```



```

function activation(hObject, eventdata, handles)
% activation plot

x = [-30;-50;-40;-60;-60;-60;-30;30;60;60;60;40;50;30];
y = [50;30;30;0;20;-60;-80;-80;-60;20;0;30;30;50];
z = [30;0;40;30;0;0;10;10;0;0;30;40;0;30];

location={};
s=cell(1,14);
for a = 1:14
    location{1} = sprintf('AF3');
    location{2} = sprintf('F7');
    location{3} = sprintf('F3');
    location{4} = sprintf('FC5');
    location{5} = sprintf('T7');
    location{6} = sprintf('P7');
    location{7} = sprintf('O1');
    location{8} = sprintf('O2');
    location{9} = sprintf('P8');
    location{10} = sprintf('T8');
    location{11} = sprintf('FC6');
    location{12} = sprintf('F4');
    location{13} = sprintf('F8');
    location{14} = sprintf('AF4');
    n = location{a};
    s(a)=strread(sprintf(n), '%s', 'delimiter', '');
end
% the plot
scatter3(-x,-y,-z, 'filled'); % <- NOT <plot3>
text(-(x+.3), -(y-.5), -z, s, 'color', [1,0,0]);
view(115,18);

```

Plotting Option

```

function plot_options_popup_Callback(hObject, eventdata, handles)
str = get(hObject, 'String');
val = get(hObject, 'Value');

switch str{val};
    case 'Final Reading'
        set(handles.eeg_final, 'visible', 'on');
        set(handles.activation_plot, 'visible', 'off');
        set(handles.emo_plot, 'visible', 'off');
        YesNo = evalin('base', 'exist(''eeg_output_1d'', ''var'')');
        axes(handles.eeg_final)
        if(YesNo == 1)
            change_current_figure(gcf);
            plot(evalin('base', 'eeg_output_1d'));
        else
            return;
        end
    case 'Activation Plot'
        set(handles.activation_plot, 'visible', 'on');
        set(handles.eeg_final, 'visible', 'off');
        set(handles.emo_plot, 'visible', 'off');
        axes(handles.activation_plot)
        change_current_figure(gcf);
        activation
    case 'Emotion State'

```

```

        set(handles.emo_plot,'visible','on');
        set(handles.eeg_final,'visible','off');
        set(handles.activation_plot,'visible','off');
        axes(handles.emo_plot)
        change_current_figure(gcf);
%         emo_Callback
        YesNo = evalin('base','exist(''eeg_output_1d'', 'var')');
        figure;
        if(YesNo == 1)
            change_current_figure(gcf);
            state(evalin('base','eeg_output_1d'));
        else
            return;
        end

end

function training_testing_Callback(hObject, eventdata, handles)
str = get(hObject, 'String');
val = get(hObject, 'Value');

switch str{val};
    case 'Training'
        eeg_rotor(hObject, eventdata, handles);
    case 'Testing'
        testing1(hObject, eventdata, handles);

end

```

EEGRotor controller code

```

% gollahalli
import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;
robot=Robot;

if isempty(class_label_for_validation)
    set(handles.tips,'string','Please find the classification results in
command window');
    fprintf('\n');
    fprintf('----- RESULTS ----- \n');
    str=strcat('Data set:',handles.FileName);
    fprintf(str);
    fprintf('\n');
    fprintf('Time:')
    fprintf(datestr(now))
    fprintf('\nClass label of the given samples:\n')
    for k=1:length(output_neurals_test_class_sn)
        fprintf('Sample %04d -----> class
%d',k,output_neurals_test_class_sn(k));
        if(output_neurals_test_class_sn == 2)
            robot.keyPress(KeyEvent.VK_W);
            robot.delay(8000);
            robot.keyRelease(KeyEvent.VK_W);
        elseif(output_neurals_test_class_sn == 1)
            robot.keyPress(KeyEvent.VK_S);

```

```

        robot.delay(300);
        robot.keyRelease(KeyEvent.VK_S);
    elseif(output_neurals_test_class_sn == 3)
        return;
        robot.keyPress(KeyEvent.VK_A);
        robot.delay(300);
        robot.keyRelease(KeyEvent.VK_A);
    elseif(output_neurals_test_class_sn == 4)
        return;
        robot.keyPress(KeyEvent.VK_D);
        robot.delay(300);
        robot.keyRelease(KeyEvent.VK_D);
    end
    fprintf('\n');
end
fprintf('-----\n');

```

Function for starting EEG data

```

function start_eeg_dynamic_Callback(hObject, eventdata, handles)
% hObject    handle to start_eeg_dynamic (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global emotiv_true_false h;
emotiv_true_false = true;

set(handles.disconnect,'enable','on');
set(handles.stop_eeg_dynamic,'enable','on');
set(handles.disconnect,'enable','on');

axes(handles.eeg_dynamic)
h = EmotivEEG;
h.Run
a=2;
while (emotiv_true_false)
    out = nan([size(h.data),4]);
    for k = 1:a
        out(:, :, k) = h.data + 1;
        plot(out(:, :, k));
%         handles.newhandel=out(:, :, k); % send it to eeg_live
        setappdata(0, 'eegdata', out(:, :, k));
        pause(0.5);
    end
    a=a+1;
end
h.Stop;
set(handles.eeg_final,'visible','on');
set(handles.activation_plot,'visible','off');
set(handles.emo_plot,'visible','off');
for eeg = 1:size(out(:, :, :), 3)
    eeg_output_1d = permute(out, [1 3 2]);
    eeg_output_1d = reshape(eeg_output_1d, [], size(out, 2), 1);
    plot(eeg_output_1d);
    assignin('base', 'eeg_output_1d', eeg_output_1d)
end

```

Stopping EEG

```
function stop_eeg_dynamic_Callback(hObject, eventdata, handles)
% hObject    handle to stop_eeg_dynamic (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global emotiv_true_false;
emotiv_true_false = false;
axes(handles.eeg_final)
set(handles.stop_eeg_dynamic, 'enable', 'off');
```

Disconnecting EEG

```
function disconnect_Callback(hObject, eventdata, handles)
% hObject    handle to disconnect (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global h;

h.delete;
set(handles.disconnect, 'enable', 'off');
```

Appendix B: Screenshots

EEGRotor Virtual Environment

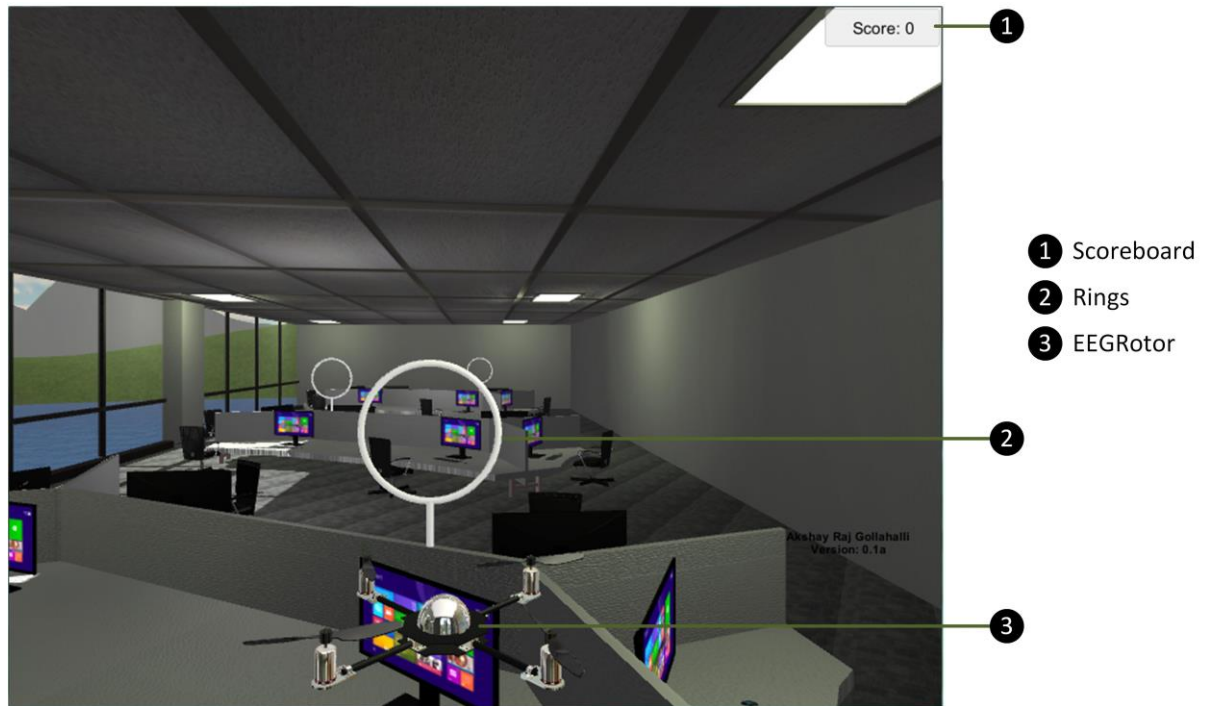


Figure B.1: EEGRotor virtual environment

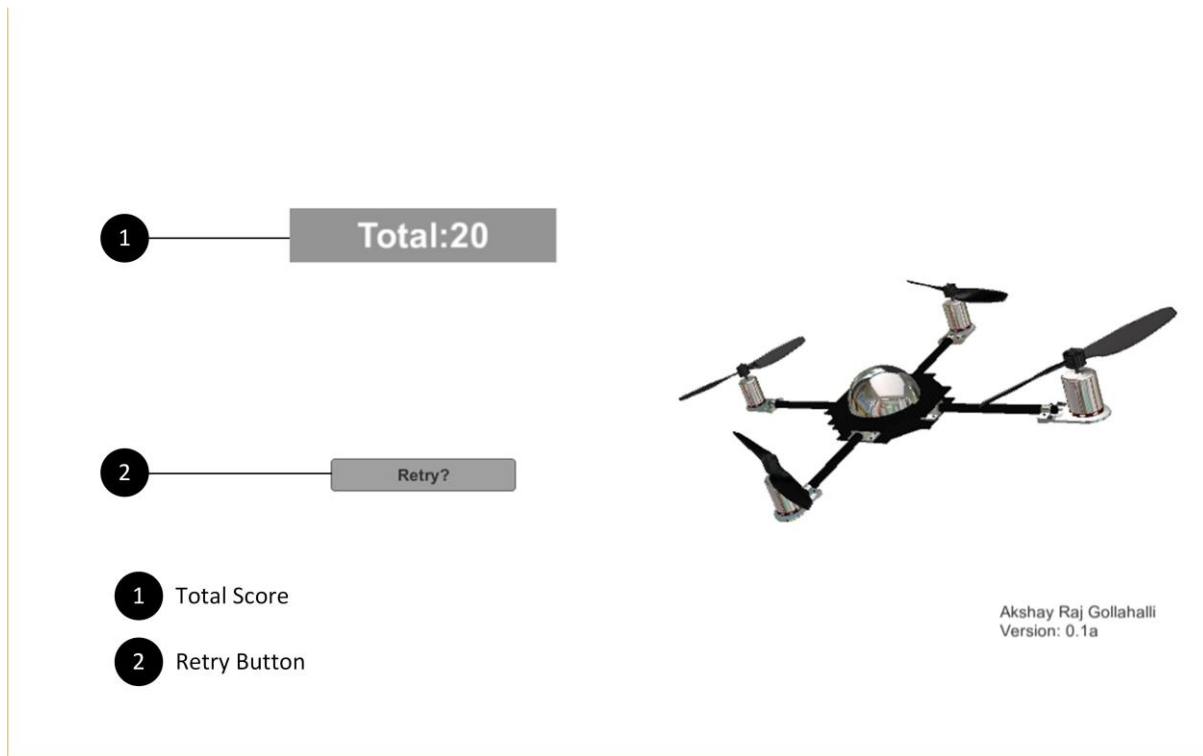


Figure B.2: EEGRotor virtual environment ending

NeuCube for BCI

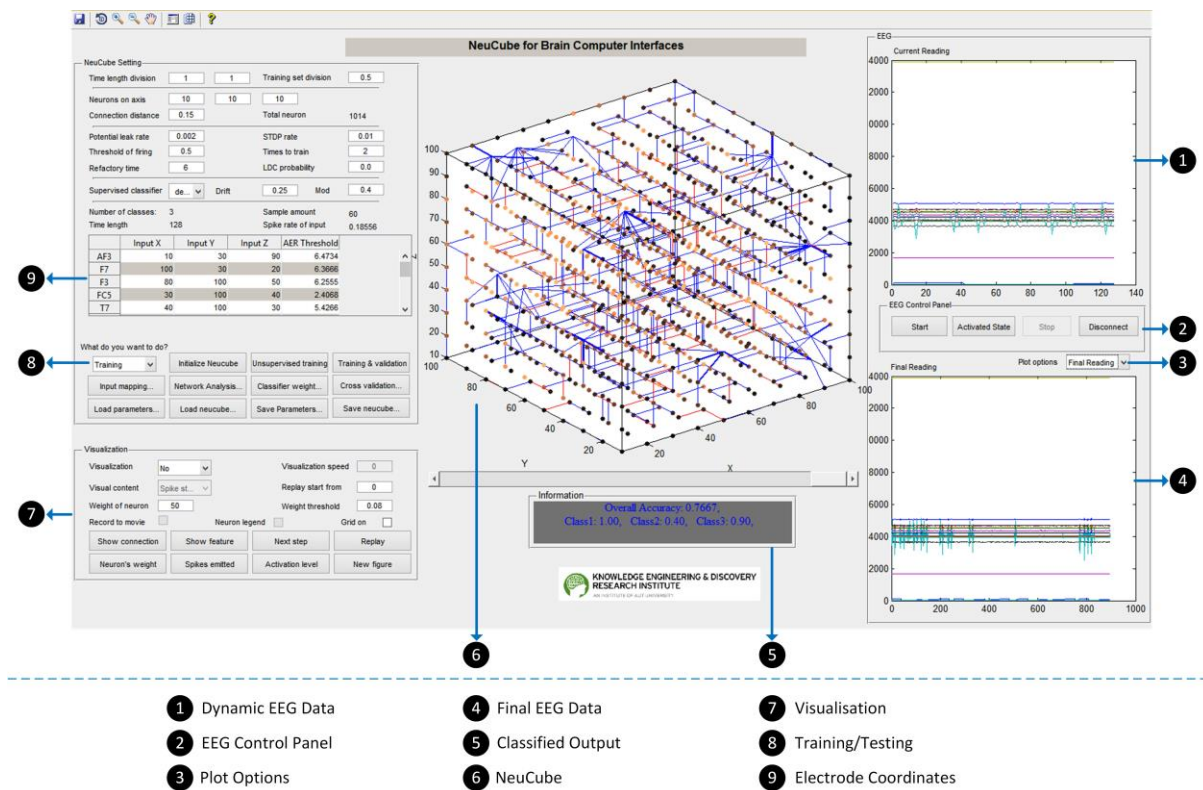


Figure B.3: NeuCube for BCI

EEGRotor Testing Module

The screenshot shows a software window titled "Testing (Static or Dynamic Data)". Inside the window, there is a label "Dynamic Data" with a line pointing to it from a blue circle containing the number "1". Below this label is the text "(or)". To the left of a text input field is a button labeled "Browse". A line points from a blue circle containing the number "2" to the text input field. At the bottom right of the window is a button labeled "Next". A line points from a blue circle containing the number "3" to the "Next" button.

- 1 Dynamic EEG data from Emotiv EPOC
- 2 Manual input of data
- 3 Next for string the process of testing

Figure B.4: EEGRotor testing module

EEGRotor Training Module

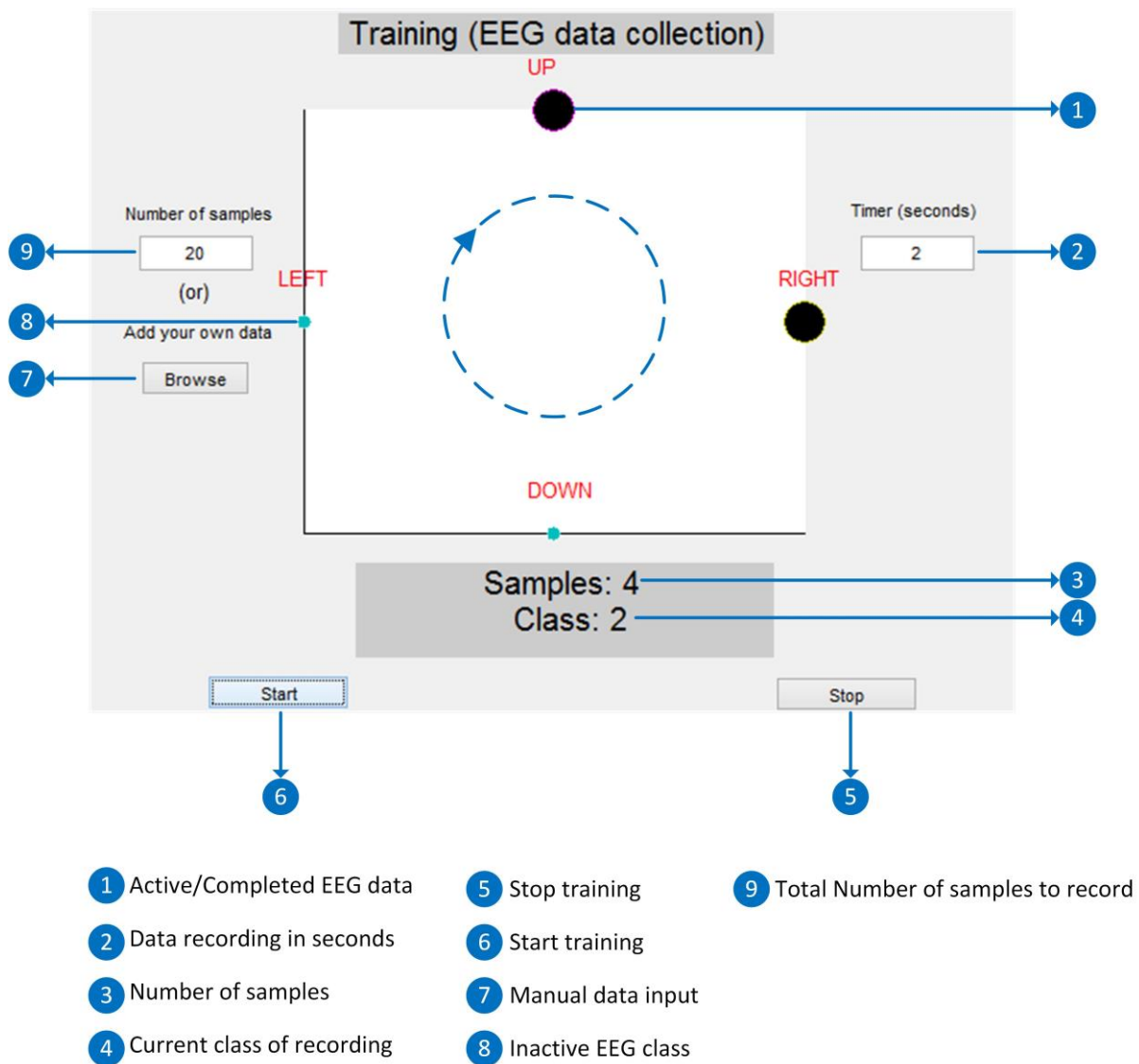


Figure B.5: EEGRotor training module

EEG Data Control Panel

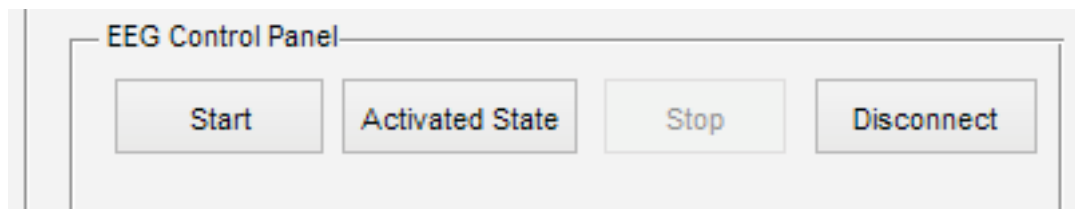


Figure B.6: EEG data control panel

EEG Data Activation Plot

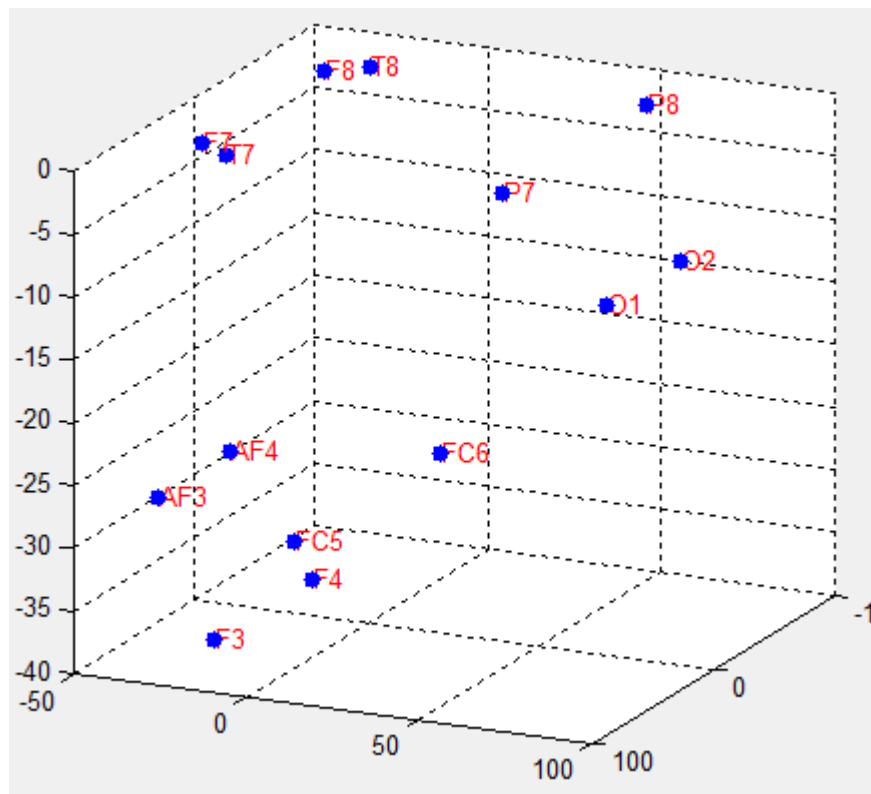


Figure B.7: EEG data activation plot

Emotional State

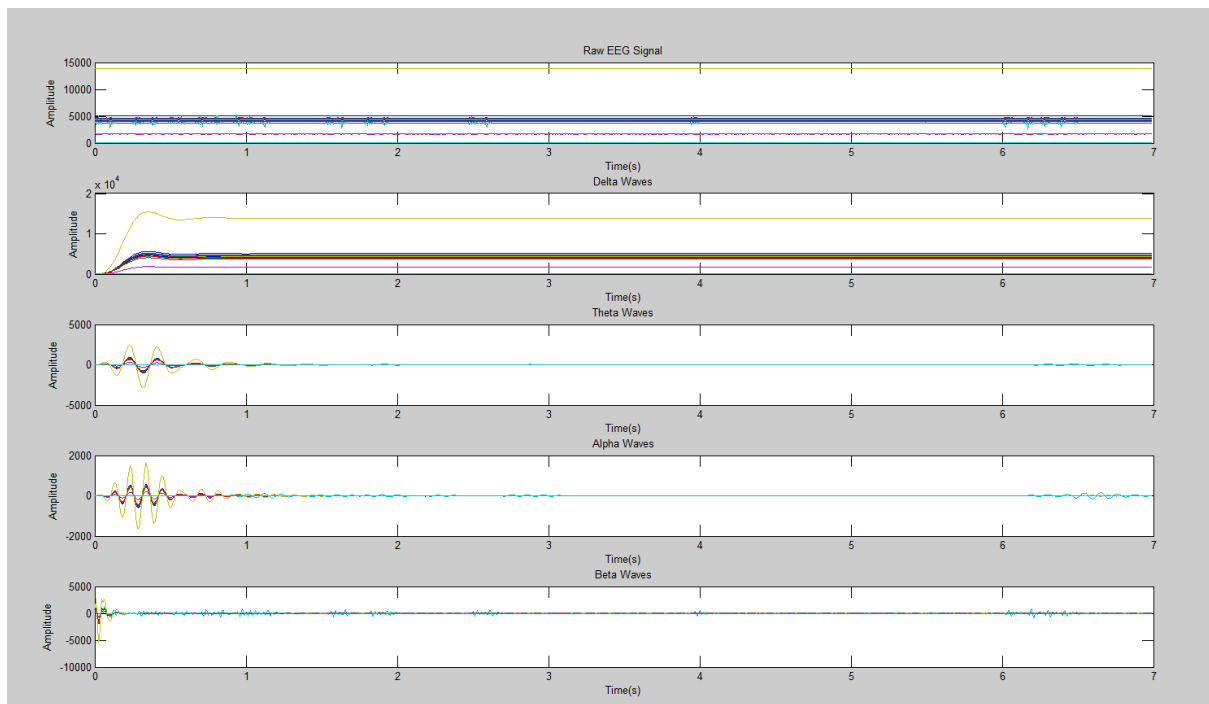


Figure B.8: Emotional state

Dynamic EEG Data

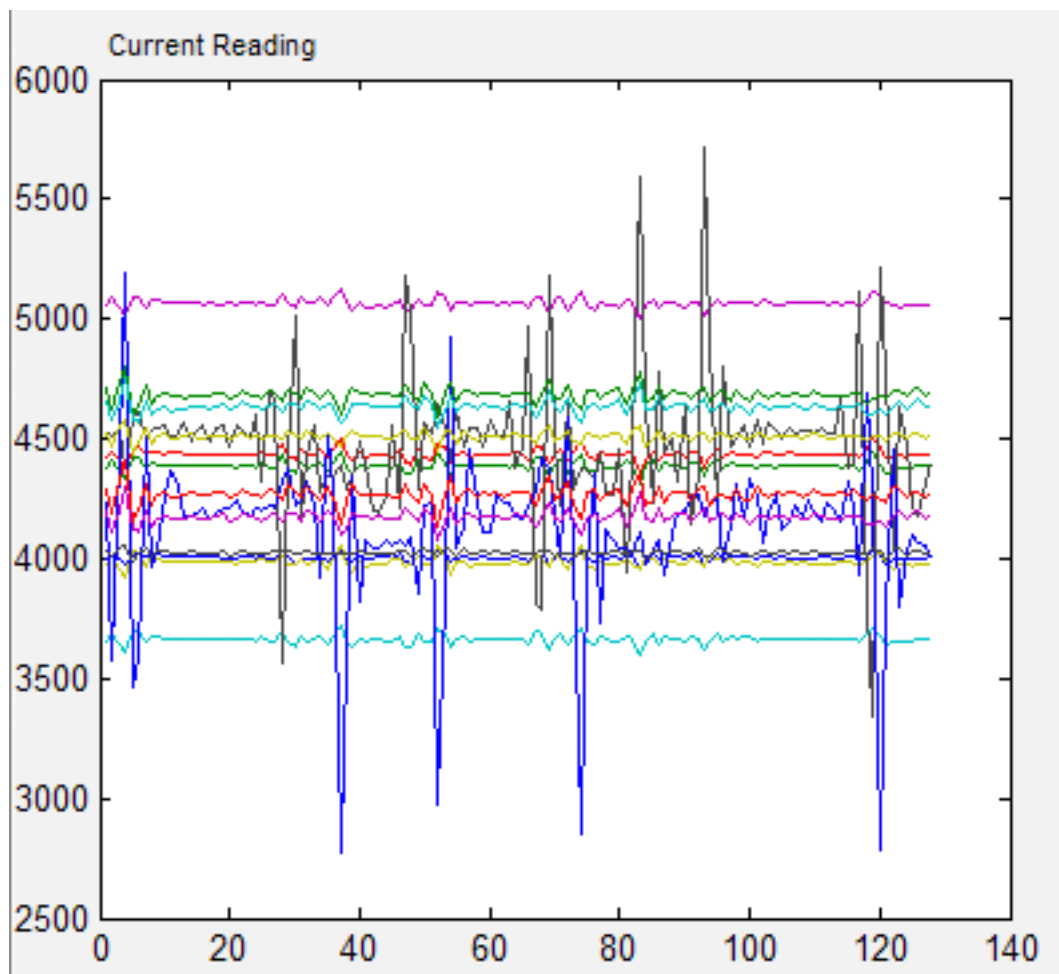


Figure B.9: Dynamic EEG data

Total EEG Data

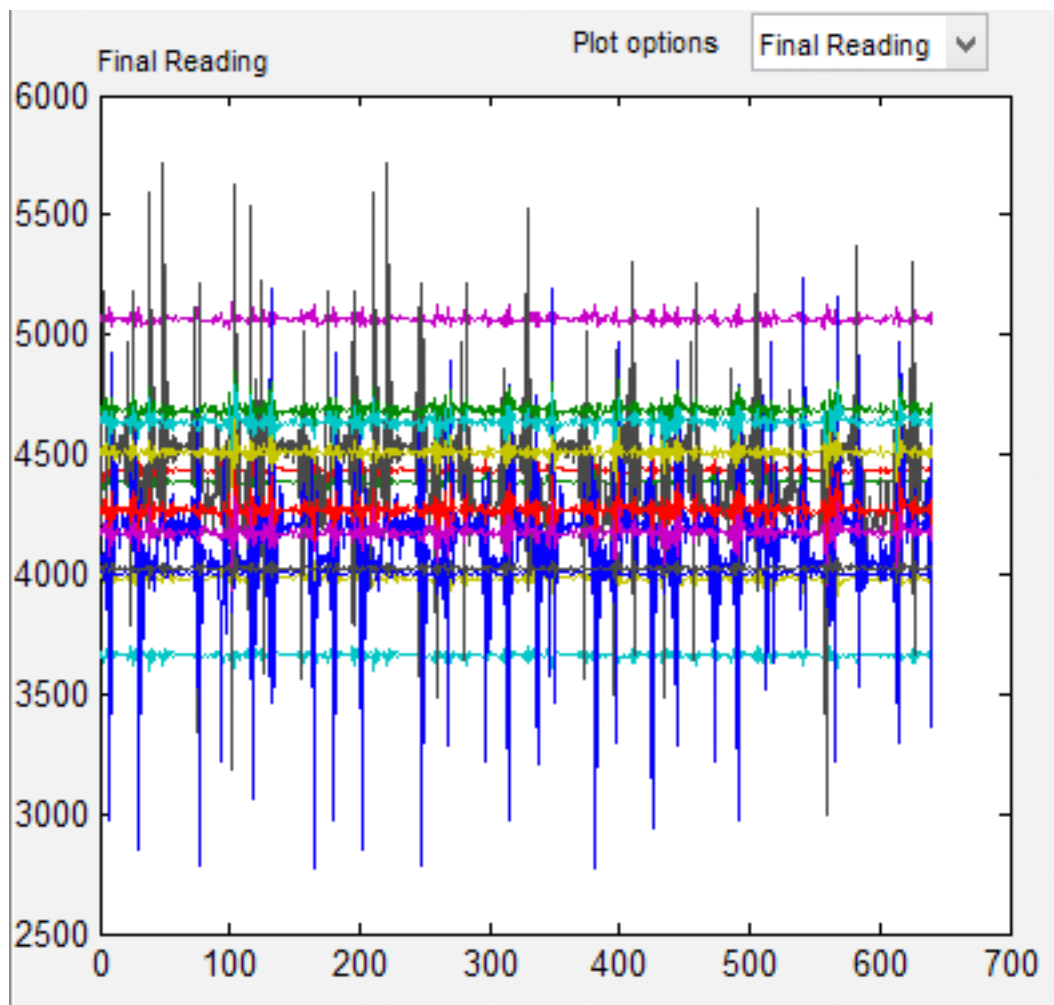


Figure B.10: Total EEG data

Blue Print of EEGRotor Hardware (Side View)

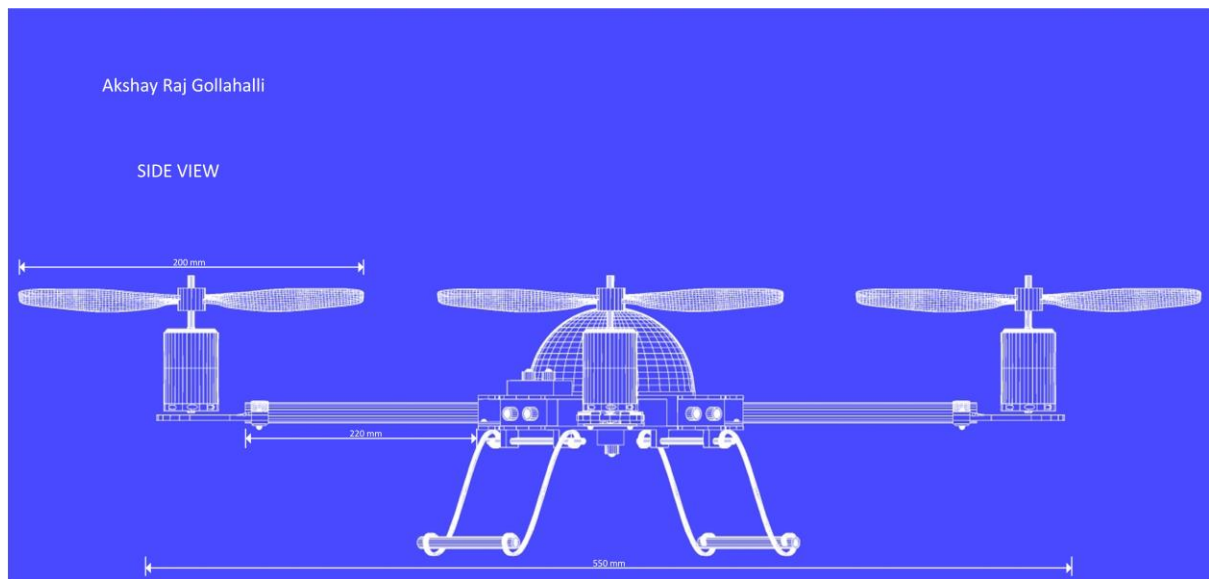


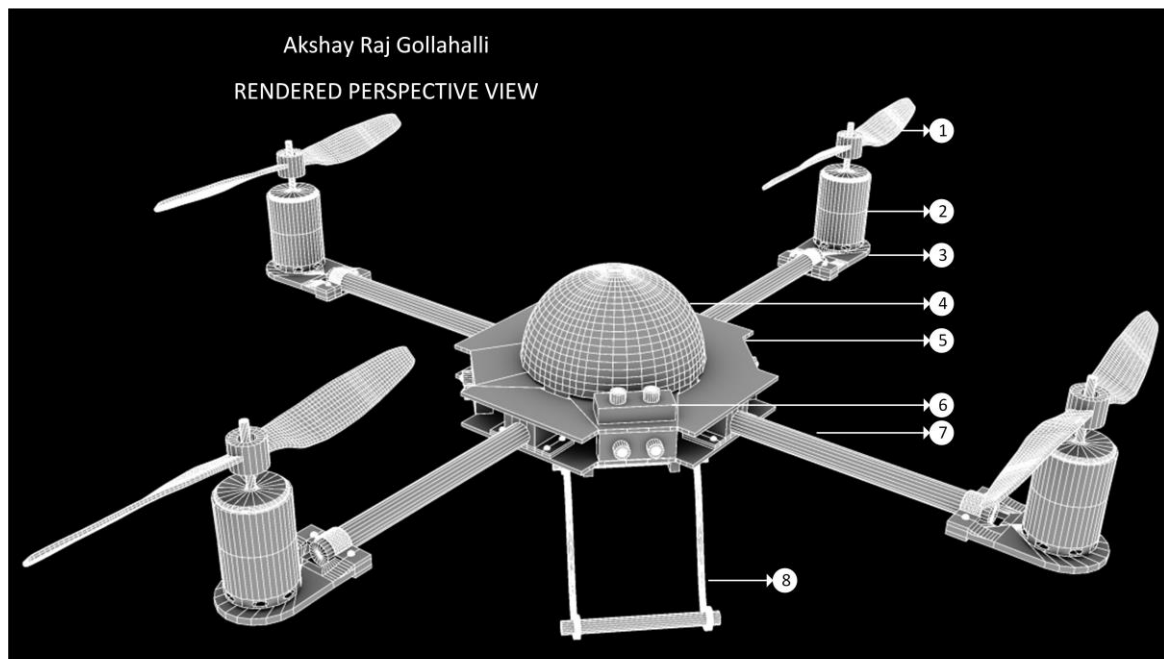
Figure B.11: Blue print of EEGRotor hardware (side view)

Blue Print of EEGRotor Hardware (Top View)



Figure B.12: Blue print of EEGRotor hardware (top view)

Rendered Image of EEGRotor Hardware (Perspective View)



- | | |
|---|-------------------------------------|
| ① Carbon Fiber Fin | ⑥ IR Sensor to measure the distance |
| ② High RPM Motor | ⑦ Carbon Fiber Boom |
| ③ Aluminum Motor Holder | ⑧ Landing gear |
| ④ Dome | |
| ⑤ Carbon Fiber Plates to hold electronics | |

Figure B.13: Rendered image of EEGRotor hardware (perspective view)

Appendix C: Videos and Software Links

NeuCube for BCI running EEGRotor

In this screen recorded video, the author shows a demo of EEGRotor VE using NeuCube for BCI. The video can be views at <https://youtu.be/AvaCuLz9XcQ>

EEGRotor VE Simulation

The author created a simulation to show the working of EEGRotor VE. The video can be viewed at <https://youtu.be/Zne6wMrti1A>

EEGRotor VE Code and Models

Code and assets for the creation of EEGRotor Virtual Environment can be found at <https://github.com/akshaybabloo/EEGRotor-VE>