ATTENTION MONITORING WITH ELECTROENCEPHALOGRAPHY AND ARTIFICIAL NEURAL NETWORK

Uduwila Arachchi Charitha Senarathne

(139102N)

Degree of Master of Science in Artificial Intelligence

Department of Computational Mathematics

University of Moratuwa

Sri Lanka

December 2015

ATTENTION MONITORING WITH ELECTROENCEPHALOGRAPHY AND ARTIFICIAL NEURAL NETWORK

Uduwila Arachchi Charitha Senarathne

(139102N)

Thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Artificial Intelligence

Department of Computational Mathematics

University of Moratuwa

Sri Lanka

December 2015

Declaration

I declare that this dissertation does not incorporate, without acknowledgment, any material previously submitted for a Degree or a Diploma in any University and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, to be made available for photocopying and for interlibrary loans, and for the title and summary to be made available to outside organization.

Name of Student	Signature of Student
Charitha Senarathne	Date:

Supervised by	Signature of Supervisor
Prof. Asoka S. Karunananda	Date:

Abstract

It’s a well-known fact that people lose attention without notice in many instances. Learning is one of them. If we remain attentive in whole leaning process, it will certainly improve our learning efficacy. If there is any possibility to identify whether we remain attentive during learning process and remind us when we lose the attention, then we can certainly improve our learning effect. In this research, monitoring EEG signals with ANN technology is used to identify whether student remain attentive during learning process.

In normal classroom environment, observation is the main way to identify whether student is attentive to the lecture. However, this needs huge effort from teacher to monitor the students. Distance learning is popular among current society, in that case it is rather difficult to use standard methods like observation to monitor the attention. Neurons in our brain are always active and emit electric pulses all the time, hence we can use those to measure the level of attention in above scenarios.

A research has been conducted to monitor attention in a particular task by a person and to signal the person immediately so that he/she can get the mind back to the task. The solution will collect the EEG data from subjects and transformed them in to frequency domain using Fast Fourier Analysis (FFT). These data are used to train an Artificial Neural Network (ANN) regarding known EEG wave patterns of attention and monitor the current EEG wave forms in a prescribed time interval. Upon receiving the current wave pattern, it will be fed in to the trained neural network and detect whether the person has lost the attention. Then it will generate a vibration alert to the mobile phone if the attention has been lost.

The solution has been tested with in a classroom scenario with 20 students and results shows that 75% of students were able to get back to the class in few seconds.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Prolegomena

With the increasing popularity of in AI, numerous intelligent techniques including Artificial Neural Network [1], Genetic Algorithm [1], Expert Systems [1] are emerging in the software industry. In particular the real world classification problems can be successfully solved by using the AI techniques. This project has been conducted to develop a system to recognize the student attention in classroom environment using the EEG signals and Artificial Neural Network. Among other AI techniques Artificial Neural Network has provided effective solution to problem solving where classification of large volume of data is presented. Brainwave analyzing is an emerging field of computing where exciting applications have been reported such as Mind controlled vehicles [2], Mind controlled games [3] and wheel chairs [4], using brain waves for medical diagnosis [5].

In this connection, this chapter present aim and objectives, background and motivation, problem in brief, novel approach to recognize the degree of attention of student in classroom environment and structure of the overall thesis.

1.2 Aim and Objectives

The aim of this project is to develop a system for recognizing the student attention in classroom environment. In order to reach this aim following objectives are identified.

1. To critically study the domain of brain computer interface field.
2. To critically analyze the current approaches to recognize the attention level in different environments.
3. Recognizing the suitable brain wave monitoring appliances to recognize the attention level in class room environment.

4. Identify processing libraries and environments which can be utilized to analyze the raw EEG signals.
5. Acquire raw brain wave data on attentive sessions from group of students.
6. Develop a mobile application to alert the student when he is not attentive in the lecture.

1.3 Background and Motivation

It’s a well-known fact that people lose attention without notice in many instances. Learning is one of them. If we remain attentive in whole leaning process, it will certainly improve our learning efficacy. If there is any possibility to identify whether we remain attentive during learning process and remind us when we lose the attention, then we can certainly improve our learning effect. In this research, monitoring EEG signals with ANN technology is used to identify whether student remain attentive during learning process.

In normal classroom environment, observation is the main way to identify whether student is attentive to the lecture. However, this needs huge effort from teacher to monitor the students. Distance learning is popular among current society, in that case it is rather difficult to use standard methods like observation to monitor the attention. Neurons in our brain are always active and emit electric pulses all the time, hence we can use those to measure the level of attention in above scenarios.

There are numerous applications to support the meditation by providing the user with the details on meditation session. Most of them are expensive and result of some advanced research such as BrainBot. There are commercial BCI applications available in the market for entertainment, medical diagnosis, meditation and etc.[5] . No one has yet tried to come up with an application which support learning process of a student. Student tend to lose the attention frequently in a lecture which drastically reduce her/his learning outcomes. Hence developing an application to recognize the attention level of a student in classroom environment would be a challenge in research perspective. This kind of solution would also have a greater commercial value as well.

1.4 Problem in Brief

It's a well-known fact that people lose attention without notice in many instances. Learning is one of them. If we remain attentive in whole leaning process, it will certainly improve our learning efficacy. If there is any possibility to identify whether we remain attentive during learning process and remind us when we lose the attention, then we can certainly improve our learning effect. In this research, monitoring EEG signals with ANN technology is used to identify whether student remain attentive during learning process. In normal classroom environment, observation is the main way to identify whether student is attentive to the lecture. However, this needs huge effort from teacher to monitor the students. Distance learning is popular among current society, in that case it is rather difficult to use standard methods like observation to monitor the attention. Neurons in our brain are always active and emit electric pulses all the time, hence we can use those to measure the level of attention in above scenarios. For this purpose we have to use EEG monitoring appliance, most of the EEG monitoring appliances are expensive and not portable as well.

1.5 Novel Approach to monitoring the attention

This research has been conducted to monitor attention in a particular task by a person and to signal the person immediately so that he/she can get the mind back to the task. The solution will collect the EEG data from subjects and transformed them in to frequency domain using Fast Fourier Analysis (FFT). These data are used to train an Artificial Neural Network (ANN) regarding known EEG wave patterns of attention and monitor the current EEG wave forms in a prescribed time interval. Upon receive of current wave pattern, it will be fed in to the trained neural network and detect whether the person has lost the attention. Then it will generate a vibration alert to the mobile phone if the attention has been lost.

1.6 Structure of the thesis

Rest of the thesis is structure as follows. Chapter 2 critically review the domain of Brain wave Interface by highlighting current solution, practices, technologies, limitations defining the research problem. Chapter 3 described essentials of EEG signals and Artificial Neural

Network showing its relevance to implement a solution to recognize the degree of human attention. Chapter 4 present our ANN approach to recognize the level of student attention in classroom environment. Chapter 5 is on the design of Focus Gain application to monitor the student attention. Chapter 6 contains details of implementation of the system which recognize degree of human attention using EEG signals and ANN. Chapter 7 reports on evaluation of the new solution by explaining the evaluation strategy, participants, data collection, data representation and data analysis. Chapter 8 concludes the outcome of the research with the note on further work.

1.7 Summary

This chapter describe the full picture of the whole project showing research problem, objectives, hypothesis and the novel solution. Next chapter will be on literature review of Brain wave Interface domain practices, technologies and issues with a view to define the research problem.

Chapter 2

Emergence of Brain Machine Interfacing

2.1 Introduction

This chapter explains the background for this project, and identifies the main research questions and methods to bring clarity and define the projects focus, based on lessons learned from earlier efforts and new anticipations.

2.2 Related Research

The EEG signal is a voltage signal that arises from synchronized neural activity, that is, the coordinated firing of millions of neurons in the brain. It can be measured by non-invasively placing an electrode on or near the scalp, and for greater accuracy, by implanting an electrode in the skull [6].

The idea of detecting mental states using EEG is not new. In fact, a brain-computer interface (BCI) system is specifically designed to detect the mental state of its user. In other words, a BCI system allows the brain to communicate with the system directly through EEG signals. The study of BCI as a field emerged from the desire for new assistive technology, targeted at handicapped patients, especially those paralyzed [2].

The playstation project builds on work done by Andersen, Juvik, Kjellen, and Storstein, as part of a multidisciplinary course at NTNU in cooperation with Stig Hollup at Department of Psychology, their assignment was to steer a radio controlled helicopter using the principles of neurofeedback. With their solution they were able to adjust the speed of the rotor using the level of concentration in a test person, and lift it off the ground. They developed an application that interfaced EEG software with the radio controller software of the helicopter.

Here are some research done recently and ongoing research on BCI.

As published in Neuroscience Letters, Scherer developed a system [3] for classifying EEG signals using Fisher's linear discriminant analysis (FLDA) and a virtual keyboard for spelling. This identifies left, right and down thoughts.

Yaomanee identified locations on the scalp that are suitable for detecting attention related EEG signals. When subjects are attentive it is noted that β activity was greater.

Li developed an emotional learning system. He uses two learning methods, k-nearest neighbor (kNN) and naive Bayes.

2.2 EEG Appliances

There are numerous EEG appliances available in the current market. Analyze has been conducted to identify the suitable EEG appliance for this project. Two headsets are identified as they fulfill the requirements of this research.

2.2.1 Emotive EPOC

Emotive EPOC can be considered as research grade EEG headset. It provides 14 EEG channels, wireless/Bluetooth connectivity. All the sensors are dry and it's portable device. It is compatible with Windows, Mac, Android and iOS. At the time of writing it cost around $399 [10].

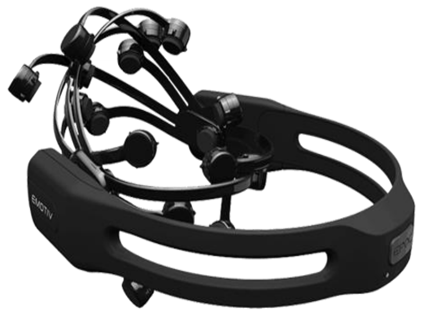


Figure 2.1: Emotive EPOC Headset

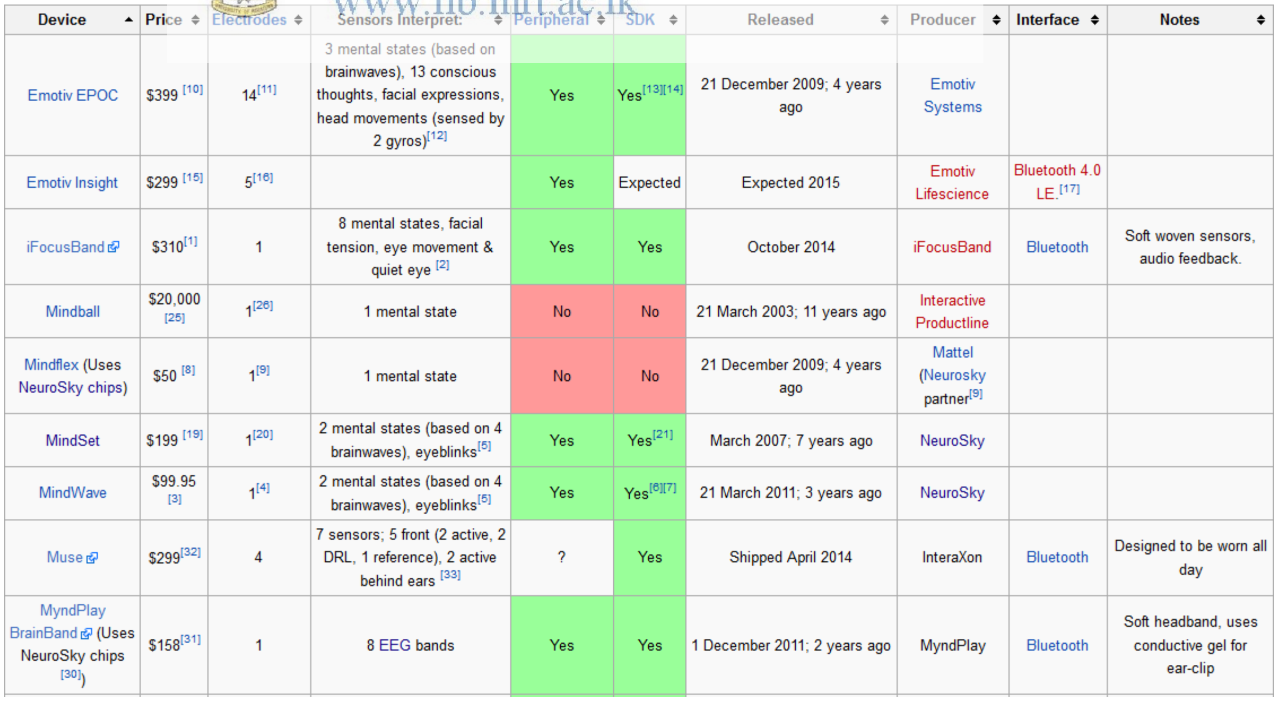
2.2.2 NeuroSky Mindwave Mobile

The NeuroSky Mindwave is a low cost single-electrode EEG headset, and it has been proven effective in detecting user’s mental states []. It has one dry sensor which is fixed to the forehead area. It supports Bluetooth connectivity and able to provide raw EEG data. For the development purpose it provide C#.NET API. At the time of writing it cost around $99. Device is compatible with Windows, Mac, Android and iOS [10]

Figure 2.2: NeuroSky Mindwave Mobile Headset

2.2.3 Comparison of EEG Appliances

Here is a comparison of EEG appliances in the market.



Device	Price	Electrodes	Sensors Interpret:	Peripheral	SDK	Released	Producer	Interface	Notes
Emotiv EPOC	$399 [10]	14[11]	3 mental states (based on brainwaves), 13 conscious thoughts, facial expressions, head movements (sensed by 2 gyros)[12]	Yes	Yes[13][14]	21 December 2009; 4 years ago	Emotiv Systems		
Emotiv Insight	$299 [15]	5[16]		Yes	Expected	Expected 2015	Emotiv Lifescience	Bluetooth 4.0 LE.[17]	
iFocusBand	$310[1]	1	8 mental states, facial tension, eye movement & quiet eye [2]	Yes	Yes	October 2014	iFocusBand	Bluetooth	Soft woven sensors, audio feedback.
Mindball	$20,000 [25]	1[26]	1 mental state	No	No	21 March 2003; 11 years ago	Interactive Productline		
Mindflex (Uses NeuroSky chips)	$50 [8]	1[9]	1 mental state	No	No	21 December 2009; 4 years ago	Mattel (Neurosky partner[9]		
MindSet	$199 [19]	1[20]	2 mental states (based on 4 brainwaves), eyeblinks[5]	Yes	Yes[21]	March 2007; 7 years ago	NeuroSky		
MindWave	$99.95 [3]	1[4]	2 mental states (based on 4 brainwaves), eyeblinks[5]	Yes	Yes[6][7]	21 March 2011; 3 years ago	NeuroSky		
Muse	$299[32]	4	7 sensors; 5 front (2 active, 2 DRL, 1 reference), 2 active behind ears [33]	?	Yes	Shipped April 2014	InteraXon	Bluetooth	Designed to be worn all day
MyndPlay BrainBand (Uses NeuroSky chips [30])	$158[31]	1	8 EEG bands	Yes	Yes	1 December 2011; 2 years ago	MyndPlay	Bluetooth	Soft headband, uses conductive gel for ear-clip

Figure 2.3: Comparison of EEG Appliances (Source : Wikipedia)

2.3 Summary

This chapter analyze the current trends in the field of Brain Computer Interface. It discusses past and current researches in the field and approaches which are taken to solve the similar sort of problems which is helpful for this research. It explains the EEG appliances available in the market and their features as well.

Chapter 3

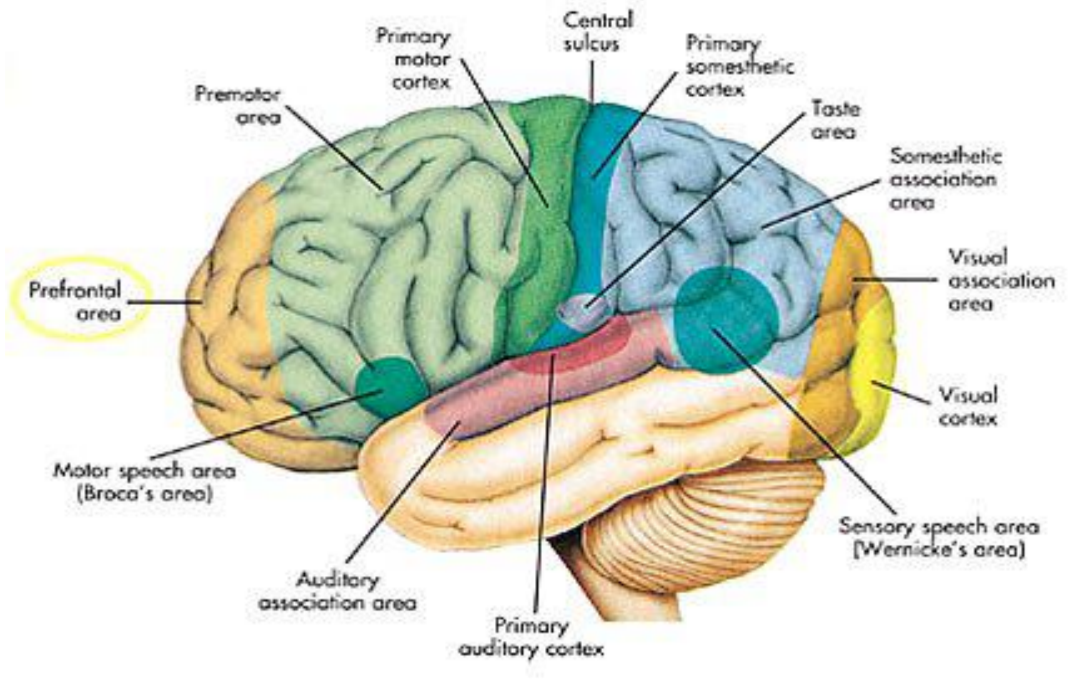
Electroencephalography (EEG) and Artificial Neural Network (ANN)

3.1 Introduction

This chapter presents the major technologies associated with the research. EEG signals are the main input to the system. EEG signals are processed by manipulating Fourier Transformation and Independent Component Analysis (ICA). Artificial Neural Network is used to classify the signals in to attentive or none attentive categories. Mobile Application is used to alert the user. This chapter described those technologies in detail.

3.2 Electroencephalography (EEG)

The brain have always fascinated humans, and particularly a German scientist named Hans Berger, who discover electroencephalography (EEG) about 80 years ago. EEG monitoring can be divided in to two major groups. Invasive and non-invasive. An invasive approach requires physical implants of electrodes in humans or animals, making it possible to measure single neurons or very local field potentials. A non-invasive approach makes use of, for instance, magnetic resonance imaging (MRI) and EEG technology to make measurements [11].



Primary motor cortex
Central sulcus
Primary somesthetic cortex
Premotor area
Taste area
Somesthetic association area
Visual association area
Prefrontal area
Visual cortex
Motor speech area (Broca's area)
Sensory speech area (Wernicke's area)
Auditory association area
Primary auditory cortex

Figure 3.1: Brain functionality Map (Source:Wikipedia)

3.2.1 Placement of Electrodes

The positions for EEG electrodes should be chosen in a way that all cortex regions which might exhibit interesting EEG patterns are covered. For most applications this is usually the whole cortex. An internationally accepted standard for electrode placements is the 10-20 system [12].

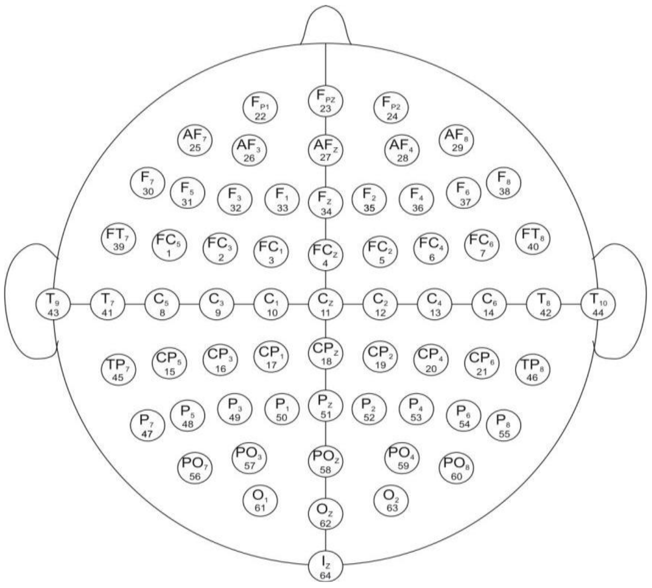


Figure 3.2: Placement for a 64-electrode system using the International 10-20 standard.

The 10–20 system or International 10–20 system is an internationally recognized method to describe and apply the location of scalp electrodes in the context of an EEG test or experiment. This method was developed to ensure standardized reproducibility so that a subject's studies could be compared over time and subjects could be compared to each other. This system is based on the relationship between the location of an electrode and the underlying area of cerebral cortex. The "10" and "20" refer to the fact that the actual distances between adjacent electrodes are either 10% or 20% of the total front–back or right–left distance of the skull.

Each site has a letter to identify the lobe and a number to identify the hemisphere location. The letters F, T, C, P and O stand for frontal, temporal, central, parietal, and occipital lobes, respectively. Note that there exists no central lobe; the "C" letter is used only for identification purposes. A "z" (zero) refers to an electrode placed on the midline. Even numbers (2, 4, 6, 8) refer to electrode positions on the right hemisphere, whereas odd numbers (1, 3, 5, 7) refer to those on the left hemisphere. In addition, the letter codes A, Pg and Fp identify the earlobes, nasopharyngeal and frontal polar sites respectively.

3.2.2 EEG

EEG is none other than voltage value generated by neurons in the brain. EEG can be thought as a wave.

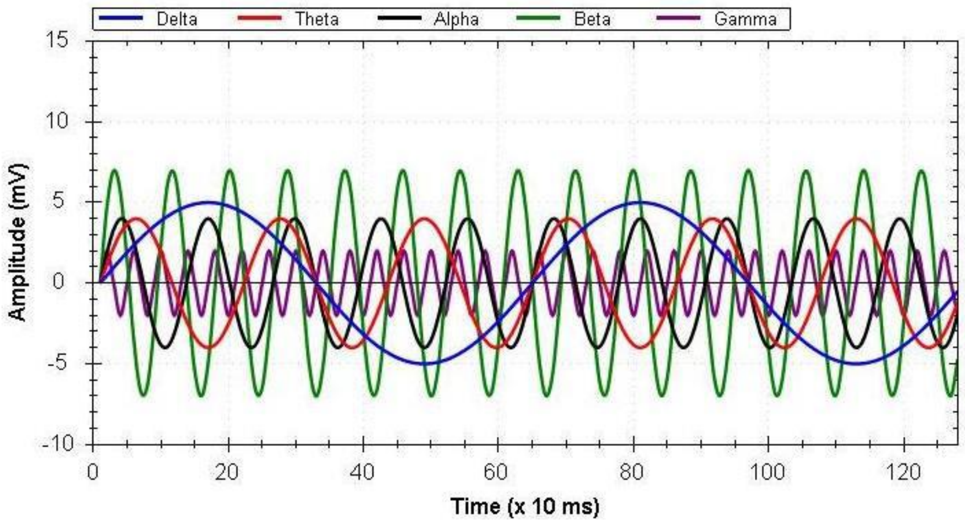


Figure 3.3: EEG wave (Voltage value against time)

Below are the five main EEG frequency bands and their basic information on them.[13]

- α band:

  Frequency Range: 8 - 13 Hz in frequency,

  Amplitude: 30 and 50 μV in amplitude.

  Source: parietal and occipital regions of the brain

  Emotions: consciousness, quiet, or at rest, thinking, blinking.

- β band:

  Frequency Range: 14 - 30 Hz in frequency,

  Amplitude: 5 and 20 μV in amplitude.

  Source: frontal regions of the brain

  Emotions: consciousness, alert, thinking.

- θ band:

  Frequency Range: 4 - 7 Hz in frequency,

  Amplitude: less than 30 μV in amplitude.

  Source: parietal and temporal regions of the brain

  Emotions: emotional pressure, interruptions of consciousness, or deep physical relaxation.

- δ band:

  Frequency Range: 0.5 - 3 Hz in frequency,

  Amplitude: 100 - 200 μV in amplitude.

  Source: parietal and temporal regions of the brain.

  Emotions: deep sleep, unconscious, anesthetized

- γ band:

  Frequency Range: 31 - 50 Hz in frequency,

  Amplitude: 5 - 10 μV in amplitude.

  Emotions: cognition and perceptual activity.

3.3 Artificial Neural Network (ANN)

Artificial Neural Network is a very powerful technique which has information processing paradigm very similar to the human brain. The ability of learning by examples and thereby identifications of hidden and unknown patterns makes the Artificial Neural Network more powerful. Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example.

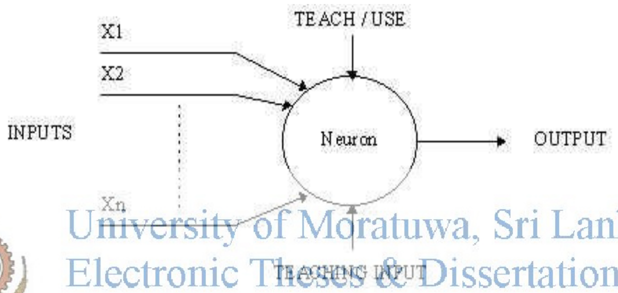


Figure 4.1: Structure of a Simple Neuron

The neuron has two modes of operation. That is the training mode and the using mode. In the training mode, the neuron can be trained to fire (or not), for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not.

In ANN model that the inputs are 'weighted', the effect that each input has at decision making is dependent on the weight of the particular input. Every neural network possesses knowledge which is contained in the values of the connections weights. Modifying the knowledge stored in the network as a function of experience implies a learning rule for changing the values of the weights.

The weight of an input is a number which when multiplied with the input gives the weighted input. These weighted inputs are then added together and if they exceed a pre-set threshold value, the neuron fires. In any other case the neuron does not fire. The function that defining "Thresholding concept" is known as 'Thresholding function' or 'Activation function'.[14]

In mathematical terms, the neuron fires if and only if;

$X1W1 + X2W2 + X3W3 + ... > T$

T = the pre-set threshold value

X = Input

W = Weight

ANN can be trained in two modes.

3.3.1 Supervised training

Supervised training we train the network by giving the decided output. In this we define maximum error, in order to make sure the supervise training is more focus on desired output. If the generated output less than the expected maximum error then we let the neural network to be learn using the input X else we discard the training. Discard the training and initialize new weight to the neural network is one of the major weaknesses of it. [15]

3.3.2 Unsupervised training

In unsupervised training we didn't specify the expected output. These types of ANN mostly use for group the data in to classes. In these types of networks, a neuron in the output layer will generate and output, regarding to the input. The neuron that generates the maximum output indicates the class where the input belongs. A neural network learns on-line if it learns and operates at the same time. Usually, supervised learning is performed off-line, whereas unsupervised learning is performed on-line.

3.3.3 Multilayer Perceptron

Multilayer perceptron means that the neural network consists of an input layer, possibly and minimum one hidden layer, and one output layer, as shown in figure.

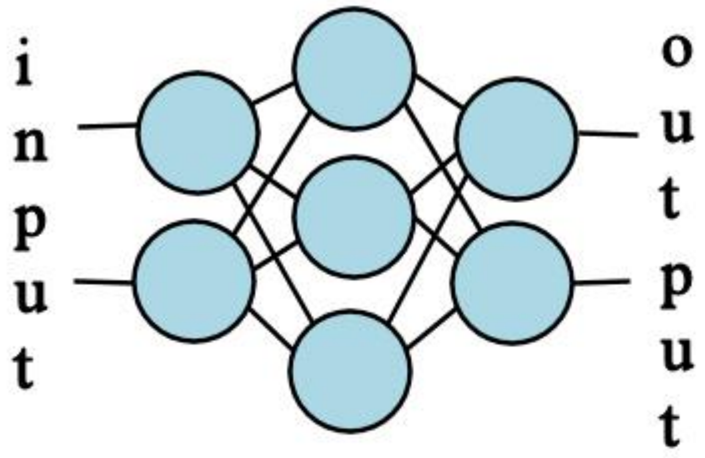


Figure 4.2: Artificial Neural Network with 3 layers

3.3.4 Backpropagation Algorithm

To understand neural networks better, and how the backpropagation works, the algorithm is given in pseudo code below.

```
Initialize the weights in the network
        while stopping criterion has reached do
                for all example e belongs to training set do
                        O = actual, output (network, e); propagate forward
                        T = wanted output for e
                        Calculate error (T - O) at each neuron in the output layer
                        Compute Mean Squared Error value; propagate backward
                        Compute delta weight update for all weights
                Update all the weights in the network such that the sum- squared value
                of error is minimized.
            end for
        end while
```

There are several types of activation function commonly used, these are:

Sigmoid Activation Function: $y = 1/(1 + Exp(-x))$

Linear Activation Function: $y = x$

Logarithmic Activation Function: $y = Log(1 + |x|)$

Sine Activation Function: $y = Sin(x)$

Tanh Activation Function: $y = Tanh(x)$

x = Input

y = Output

3.4 Fourier Transformation (FFT)

The Fourier Transformation and extraction of band powers is by far the most applied method for signal processing and analysis (Lotte et al., 2007) [3]. The algorithm is based on discrete Fourier transform (DFT) equation and by applying that to the EEG signal it makes it possible to separate the EEG rhythms.

The performance of the DTF is $O(N^2)$, but there is a more efficient algorithm called fast Fourier Transform (FFT), that can compute the same result in only $O(N\log_2 N)$. This is a great improvement and one of the reasons why FFT is the favorable method of analyzing EEG signals, and other waves like sound [3].

3.5 Mobile Applications

Results of the attention monitoring sessions must be convey to the user via some medium. We chose mobile application to convey the results to the student as it is much easier and convenient to be notified via a mobile application. Android operating system powers considerable number of mobile phones and t it provides rich set of API to develop the mobile applications. Hence we will be developing mobile application using Android. We will analyze the EEG in real time and process it and notify the user via mobile application if attention has been lost.

3.6 Summary

Throughout this chapter we discussed technologies which are used to facilitate the measuring of degree of human attention. EEG technology, ANN technology, Fourier Transformation and Mobile Application field is discussed in this chapter.

Chapter 4

ANN Approach to Recognize the Degree of Human Attention

4.1 Introduction

In the previous two chapters we define the research problem as the inefficiency in current approaches to address the brain wave classification for attention. We discuss why ANN technology could be a potential technology to develop novel solution for recognizing the degree of human attention. This chapter presents our approach by describing the hypothesis, input, output, process, features and users for novel solution for recognizing degree of human attention using Artificial Neural Network. The solution has been named as FocusGain.

4.2 Hypothesis

Sustain attention can be monitored by analyzing EEG wave signals and Artificial Neural Network.

4.3 Inputs to System

The system will receive EEG wave signals generated by the area of the brain responsible for maintaining the attention. This is generated by the EEG headset with the signal node.

4.4 Output from the System

Determination of attention disturbance and informing to the user through an alert to the mobile phone.

4.5 Process

System will use Fourier transformation to convert the time domain EEG signals to frequency domain EEG signals and those data will be fed to the ANN to recognize level of attention.

In this process five frequency bands will be used as inputs to the ANN.

These inputs are applied to an already trained neural network regarding maintenance of attention in various sessions.

4.6 Features

The following features are a present in the system.

- Real time EEG wave analyzing and classification.
- Mobile device support to inform the user that he is not attentive
- Cheaper device, affordable.
- Bluetooth connectivity
- Dry sensor, hence portable, easy to use

4.7 Summary

In this chapter we discussed how we approach to solve the problem of measuring the degree of human attention. System inputs, outputs, hypothesis, process and features of the system are explained in here.

Chapter 5

Design of FocusGain Application

5.1 Introduction

In this chapter we are going to discuss how we carried out the designing of this research. This chapter provides a detailed explanation of all methods and classifiers used in the analyses and experiments, as well as the reasons behind using those methods.

We divided our research work into data acquisition and data processing. In data acquisition, we deal with the EEG device and acquiring data through several experiments. After that, we dealt with converting all the acquired data. Artifact removal requires deep attention, since most of the data are noisy. Features need to extracted to enable us eventually classify the brain signals.

We named this application as "Focus Gain". Below is the top level architecture of the system. We will be discussing about each module detailed in this chapter.

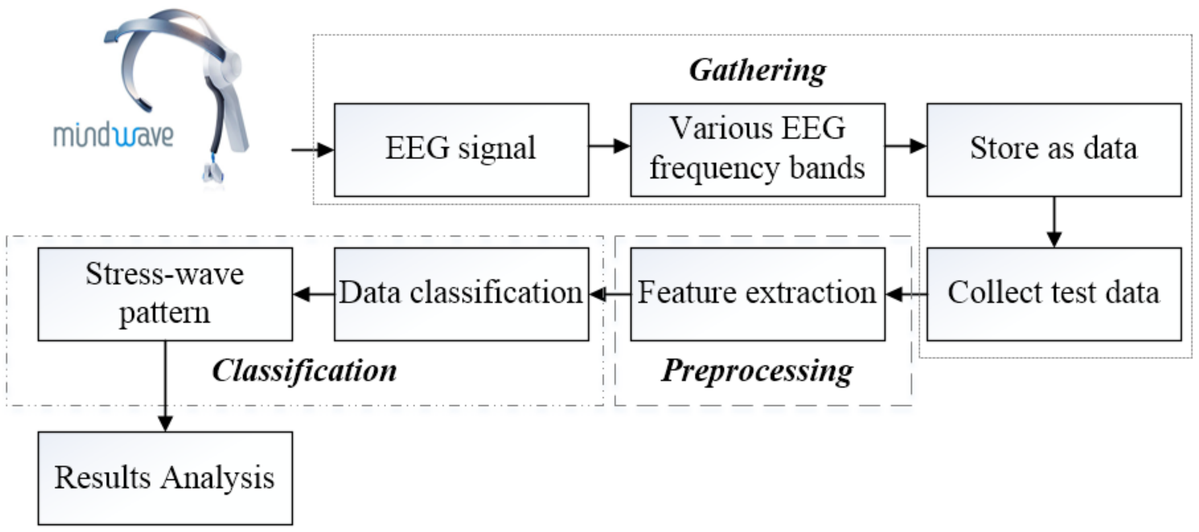


Figure 5.1: Top level architecture of Focus Gain application

Design of the Focus Gain application can be divided in to two sections.

- Data Acquisition
- Data Processing

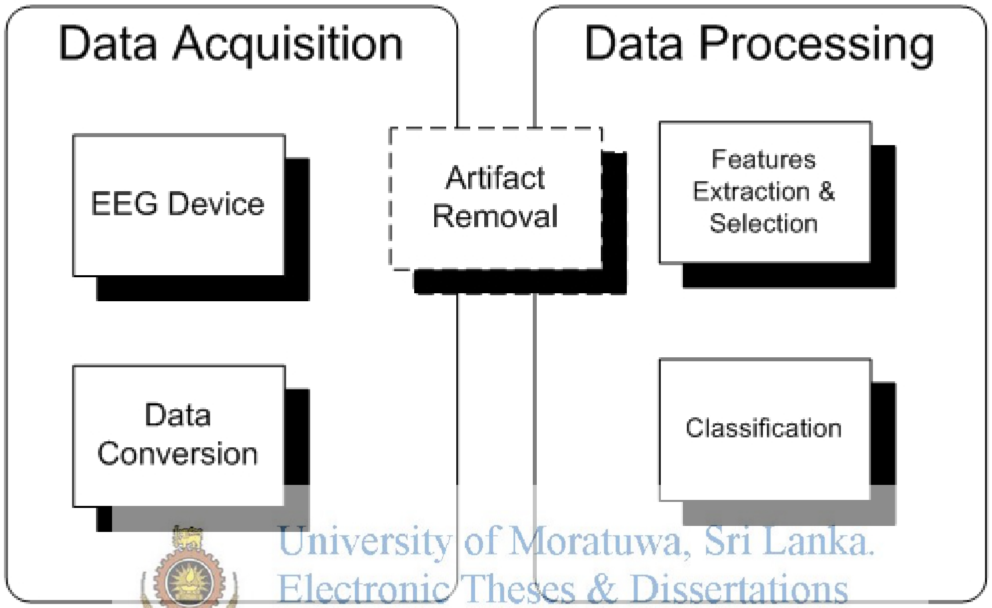


Figure 5.2: Design of the Focus Gain application

We will be looking at each module separately in this section.

5.2 Data Acquisition with NeuroSky Mindwave Headset

To acquire the signals from the brain we are using NeuroSky Mindwave headset. It is cheap and has only one dry sensor. The sensor is fit on to forehead. As the attentional related brain signals are generated from right and left sides of the forehead, sensor location is perfect location to conduct this research.

NeuroSky Mindwave headset output the preprocessed frequency band related data during each second. Raw voltage values are output at frequency of 512. These details are sent via Bluetooth. So we can capture these information using any programming language and make use of them. For this research we are conducting 5 minute sessions where student has to fully concentrate on the lecture. We will be using 20 test subjects and 4 sessions from each. These session details are collected as text files.

5.3 Artifact Removal

In EEG recordings, sensor is used to record the brain activity during attentive sessions. These sensors not only recorded brain activity from neural activities but also other artifacts. These artifacts overlap with the neural recordings. We are concentrating on removing eye-related artifacts. Eye related artifacts have the highest impact on neural recordings.

For this purpose we are first applying a simple filtering based on signal strength. NeuroSky Mindwave headset has signal strength levels defined based on the strength of the signal. The signal strength level is a value between 0-200. Anything higher than 60 would be considered as an acceptable signal. So in the first phase we are removing the unacceptable signals.

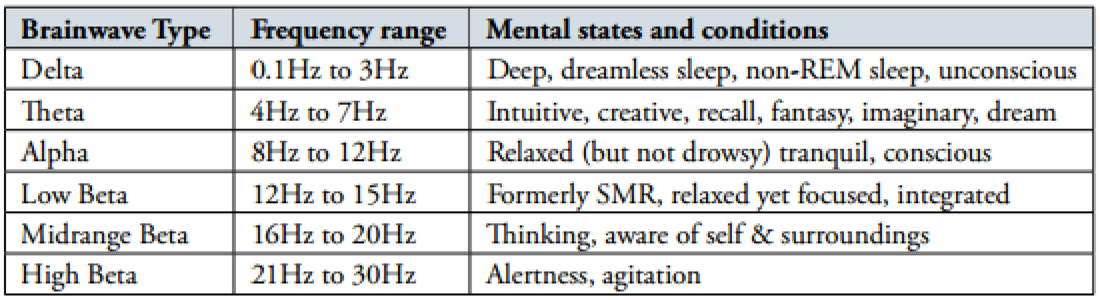
Independent Component Analysis (ICA) is another way to filter the unwanted signals. The purpose of ICA is to decompose these overlapping recorded activities into independent component. The main concern of ICA is to separate between several sources of signals. For this purpose we are going to use Java ICA related libraries.

5.4 Feature Extraction and Selection

Feature extraction is the process of extracting useful information from the signal. Features are characteristics of a signal that are able to distinguish between different emotions. The following features that will be used in the study.

- Delta
- Theta
- LowAlpha
- HighAlpha
- LowBeta
- HighBeta

- LowGamma
- HighGamma



Brainwave Type	Frequency range	Mental states and conditions
Delta	0.1Hz to 3Hz	Deep, dreamless sleep, non-REM sleep, unconscious
Theta	4Hz to 7Hz	Intuitive, creative, recall, fantasy, imaginary, dream
Alpha	8Hz to 12Hz	Relaxed (but not drowsy) tranquil, conscious
Low Beta	12Hz to 15Hz	Formerly SMR, relaxed yet focused, integrated
Midrange Beta	16Hz to 20Hz	Thinking, aware of self & surroundings
High Beta	21Hz to 30Hz	Alertness, agitation

Figure 5.3: EEG Frequency Ranges

EEG data were recorded at a sampling rate of 512. Filtering is applied on EEG data with cut-off frequency of 50 Hz. Then we convert these time domain signals in to frequency domain by using FFT. These five band values and another 3 intermediate band values are later used in ANN as input parameters. There are Java libraries that we can utilize for this purpose.

5.5 Classification with Artificial Neural Network

Figure 5.3 shows the architecture of the neural network. The input string to the network was the five power bands of a sample: delta, theta, alpha, beta and gamma. These were obtained by conducting FFT on the raw EEG data, and then dividing them into buckets according to their frequency range. Each band value is scaled with respect to the highest frequency in each individual sample, ensuring that all values are in the interval between 0 and 1.

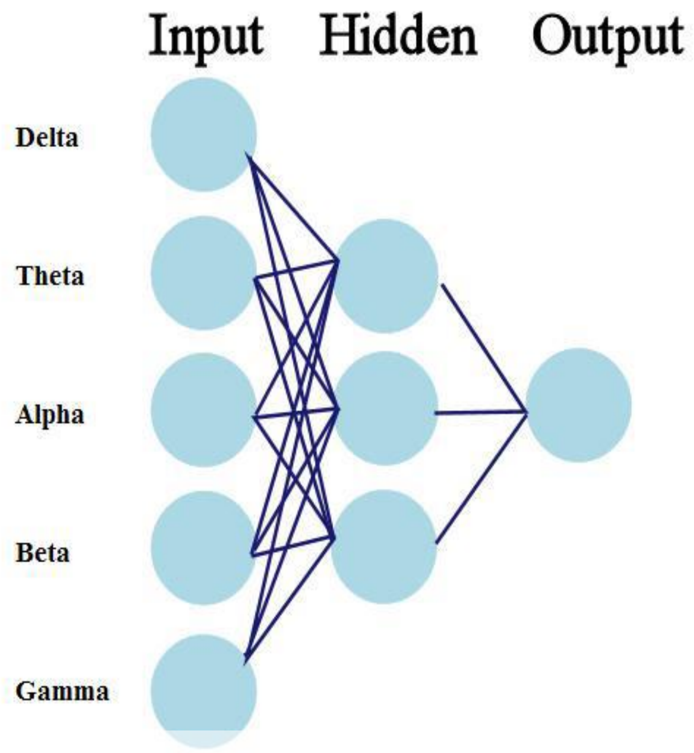


Figure 5.3: Design of the Artificial Neural Network

Training is carried out using Backpropagation algorithm. After training all the samples we are going to serialize the Artificial Neural Network, so it can be easily ported to any other application or programming language.

As per the classification method we use ANN as it can be easily trained if we have adequate sample data. The application needs to be operated in real time. So using an ANN really helps to achieve performance.

5.6 Mobile Application Development with Android

After successful training of the ANN, we now have trained ANN yet to be tested with real scenario. We ae developing android application to measure the attention of a student. Android application receives brain wave data from NeuroSky Mindwave. Android application uses previously mentioned trained Neural Network. As this is a real time application, data are coming rapidly to the application. So based on this data, Focus Gain measure the attentiveness of the application.

5.7 Summary

In this chapter we have discussed how brain wave data are collected and remove the unnecessary artifacts from them. Then we apply FFT on brain wave data and convert them in to frequency domain. These domain related data are then used to train an ANN. Trained ANN is used within Android application to detect the attentiveness of a person. If person is not attentive then vibration alert is given.

Chapter 6

Implementation of FocusGain Application

6.1 Introduction

In this chapter we are going to discuss how we implement FocusGain application. This chapter discuss each module in detail. Chapter 5 provides design perspective of the application. In this chapter we are going to discuss how each module is developed and technologies, methods, algorithms used to develop the application.

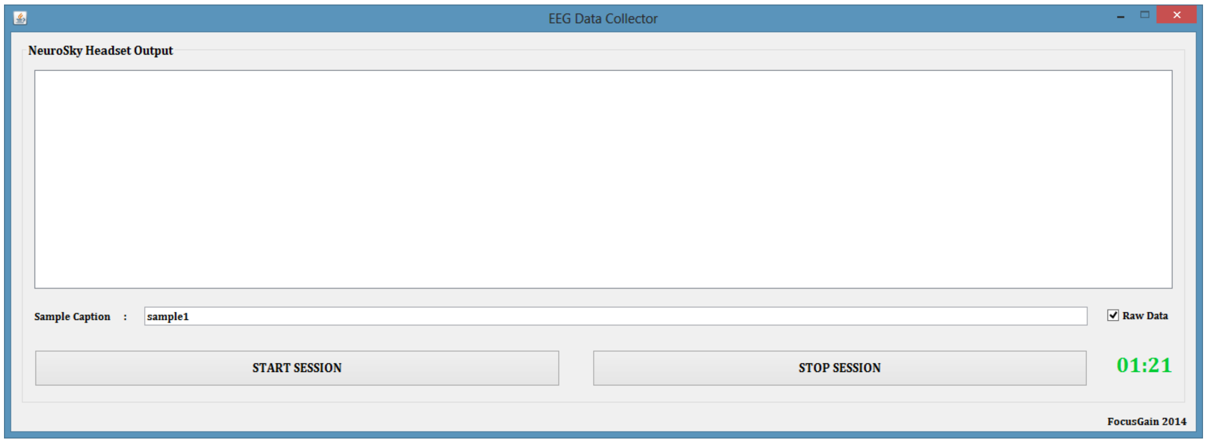
In the design phase we have divided the application in to Data acquisition and Data processing parts. We will be discussing the implementation of each part detailed here.

6.2 Data Acquisition

First phase of the application development is to collect the EEG data from students when they are in attentive to the lecture. For this purpose we use 20 students (10 male and 10 female). Each student is given four 5 minute sessions. Each student is instructed to focus on the lecture and EEG data from NeuroSky headset are stored for each session.

NeuroSky provide rich application development framework to retrieve the data from the headset. They provide .NET API to retrieve the data from headset. But in this research we are developing application using Java. So we have followed alternate method to retrieve the data from NeuroSky headset [Appendix A].

We develop Java application to collect the data from NeuroSky headset. NeuroSky headset provide several ways to retrieve the data. Data Collector application is responsible for acquiring all the data from the headset and storing them as text files.

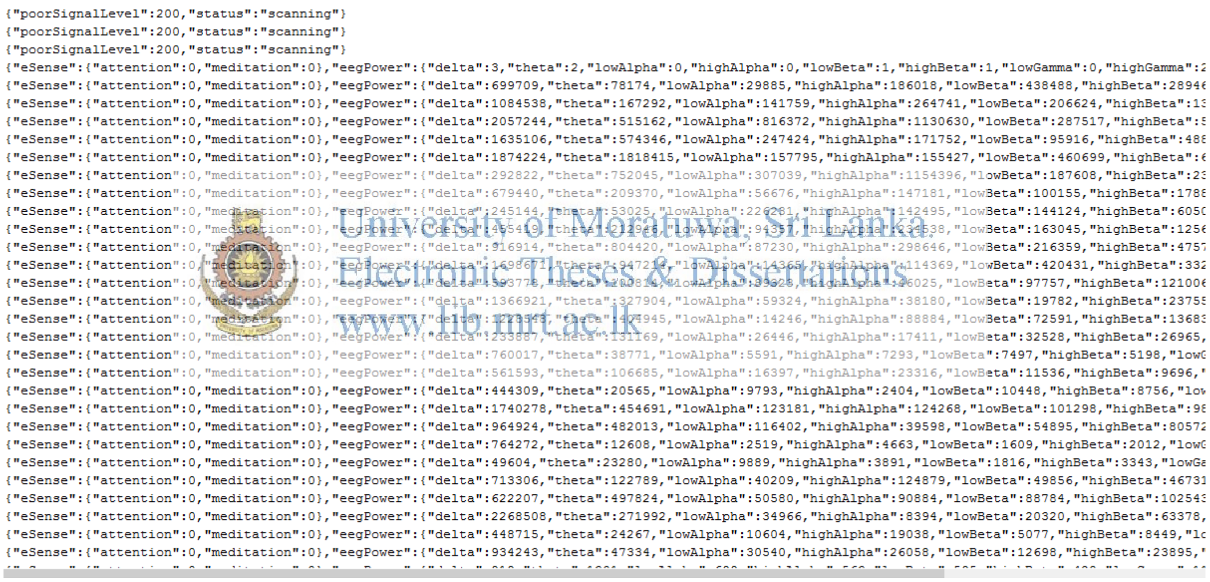


Sample Caption : sample1

[✓ Raw Data]

01:21

Figure 6.1: EEG Data Collector Application



```
{"poorSignalLevel":200,"status":"scanning"}
{"poorSignalLevel":200,"status":"scanning"}
{"poorSignalLevel":200,"status":"scanning"}
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":3,"theta":2,"lowAlpha":0,"highAlpha":0,"lowBeta":1,"highBeta":1,"lowGamma":0,"highGamma":2
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":699709,"theta":78174,"lowAlpha":29885,"highAlpha":186018,"lowBeta":438488,"highBeta":28946
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":1084538,"theta":167292,"lowAlpha":141759,"highAlpha":264741,"lowBeta":206624,"highBeta":13
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":2057244,"theta":515162,"lowAlpha":816372,"highAlpha":1130630,"lowBeta":287517,"highBeta":5
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":1635106,"theta":574346,"lowAlpha":247424,"highAlpha":171752,"lowBeta":95916,"highBeta":488
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":1874224,"theta":1818415,"lowAlpha":157795,"highAlpha":155427,"lowBeta":460699,"highBeta":6
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":292822,"theta":752045,"lowAlpha":307039,"highAlpha":1154396,"lowBeta":187608,"highBeta":23
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":679440,"theta":209370,"lowAlpha":56676,"highAlpha":147181,"lowBeta":100155,"highBeta":1788
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":245144,"theta":53025,"lowAlpha":226281,"highAlpha":142495,"lowBeta":144124,"highBeta":6050
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":455419,"theta":212946,"lowAlpha":94357,"highAlpha":234538,"lowBeta":163045,"highBeta":1256
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":916914,"theta":804420,"lowAlpha":87230,"highAlpha":298646,"lowBeta":216359,"highBeta":4757
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":1698671,"theta":947214,"lowAlpha":14365,"highAlpha":114369,"lowBeta":420431,"highBeta":332
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":393778,"theta":200814,"lowAlpha":39323,"highAlpha":46025,"lowBeta":97757,"highBeta":121006
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":1366921,"theta":327904,"lowAlpha":59324,"highAlpha":38180,"lowBeta":19782,"highBeta":23755
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":1223548,"theta":404945,"lowAlpha":14246,"highAlpha":88884,"lowBeta":72591,"highBeta":13683
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":233887,"theta":131169,"lowAlpha":26446,"highAlpha":17411,"lowBeta":32528,"highBeta":26965,
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":760017,"theta":38771,"lowAlpha":5591,"highAlpha":7293,"lowBeta":7497,"highBeta":5198,"lowG
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":561593,"theta":106685,"lowAlpha":16397,"highAlpha":23316,"lowBeta":11536,"highBeta":9696,"
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":444309,"theta":20565,"lowAlpha":9793,"highAlpha":2404,"lowBeta":10448,"highBeta":8756,"low
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":1740278,"theta":454691,"lowAlpha":123181,"highAlpha":124268,"lowBeta":101298,"highBeta":98
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":964924,"theta":482013,"lowAlpha":116402,"highAlpha":39598,"lowBeta":54895,"highBeta":80572
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":764272,"theta":12608,"lowAlpha":2519,"highAlpha":4663,"lowBeta":1609,"highBeta":2012,"lowG
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":49604,"theta":23280,"lowAlpha":9889,"highAlpha":3891,"lowBeta":1816,"highBeta":3343,"lowGa
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":713306,"theta":122789,"lowAlpha":40209,"highAlpha":124879,"lowBeta":49856,"highBeta":46731
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":622207,"theta":497824,"lowAlpha":50580,"highAlpha":90884,"lowBeta":88784,"highBeta":102543
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":2268508,"theta":271992,"lowAlpha":34966,"highAlpha":8394,"lowBeta":20320,"highBeta":63378,
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":448715,"theta":24267,"lowAlpha":10604,"highAlpha":19038,"lowBeta":5077,"highBeta":8449,"lo
{"eSense":{"attention":0,"meditation":0},"eegPower":{"delta":934243,"theta":47334,"lowAlpha":30540,"highAlpha":26058,"lowBeta":12698,"highBeta":23895,"
```

Figure 6.2: EEG Sample Data file

6.3 Data Pre Processing

After collecting the samples from the students, next step is to process the data. At this point we have 80 samples of data files each containing 5 minutes of data. Before using this data, we need to preprocess them.

6.3.1 Artifact Removal

In EEG recordings, sensor is used to record the brain activity during attentive session. These sensors not only recorded brain activity from neural activities but also artifacts. These artifacts overlap with the neural recordings. We are concentrating on removing eye-related artifacts. First and foremost because it contaminates constantly and has the largest impact on the EEG data.

For this purpose we are first applying a simple filtering based on signal strength. NeuroSky Mindwave headset has signal strength levels defined based on the strength of the signal. The signal strength level is a value between 0-200. Anything higher than 60 would be considered as an acceptable signal. So in the first phase we are removing the unacceptable signals.

Independent Component Analysis (ICA) is another way to filter the unwanted signals. The purpose of ICA is to decompose these overlapping recorded activities into independent component. The main concern of ICA is to separate between several sources of signals. For this purpose we are going to use Java ICA related libraries.

6.4 Feature Extraction and Selection

Feature extraction is the process of extracting useful information from the signal. Features are characteristics of a signal that are able to distinguish between different emotions. We get time domain signal from NeuroSky headset, so we need to convert this in to frequency domain. Fast Fourier Transformation is used for this. Fast Fourier Transformation is much effective in this context than normal Fourier Transformation. After applying FFT we get following bands from the EEG signal. NeuroSky headset internally does this processing and output the band details once per second.

- Delta
- Theta
- LowAlpha
- HighAlpha

- LowBeta
- HighBeta
- LowGamma
- HighGamma

EEG data were recorded at a sampling rate of 512. Filtering is applied on EEG data with cut-off frequency of 50 Hz. Then we convert these time domain signals in to frequency domain by using FFT. These five band values and another 3 intermediate band values are later used in ANN as input parameters.These five band values are later used in ANN as input parameters.

6.5 Designing and Training Artificial Neural Network

Next phase of the development is to design the Artificial Neural Network. This is the most vital part of the project. Careful design of ANN and successful training will eventually lead the project to success. In earlier section, we have identified the features of EEG signal, so now we can use them as inputs to the ANN. Following bands which have integer values (comparison purposes) are used as input parameters to the ANN. Output layer contains only one neuron. After conducting experiments we have identified number of neurons to be used as hidden neuron count.

After doing exhaustive experiments with training sessions, suitable values to be used for momentum and learning rate are identified as follows.

Input Layer - 8 neurons

Hidden Layer – 10 Neurons

Output Layer - 1 neuron

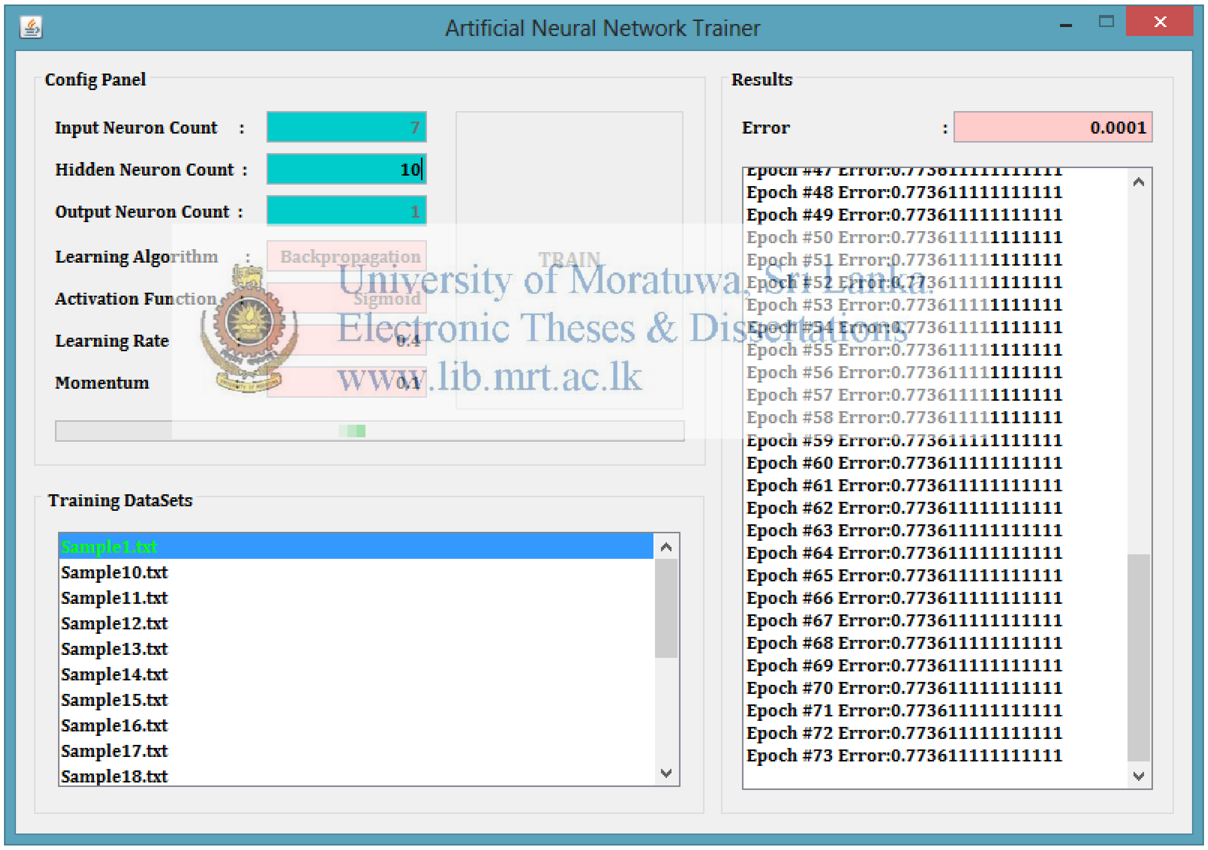
Momentum - 0.1

Learning rate - 0.2

Algorithm - Backpropagation

Each sample data file is fed in to the ANN and supervised training session has been conducted. We have 80 training sessions. It took approximately 1 hour to train the data set in to ANN. After training weight set is written in to separate file which we will be using in our desktop and android application.

We use Java to develop a separate application to train the data in to ANN. Although there are Visual tools available for training, they do not provide the functionalities that match our specific requirements.



Input Neuron Count : 7
Hidden Neuron Count : 10
Output Neuron Count : 1
Learning Algorithm : Backpropagation
Activation Function : Sigmoid
Learning Rate 0.4
Momentum 0.1

Error : 0.0001

```
Epoch #47 Error:0.773611111111111
Epoch #48 Error:0.773611111111111
Epoch #49 Error:0.773611111111111
Epoch #50 Error:0.773611111111111
Epoch #51 Error:0.773611111111111
Epoch #52 Error:0.773611111111111
Epoch #53 Error:0.773611111111111
Epoch #54 Error:0.773611111111111
Epoch #55 Error:0.773611111111111
Epoch #56 Error:0.773611111111111
Epoch #57 Error:0.773611111111111
Epoch #58 Error:0.773611111111111
Epoch #59 Error:0.773611111111111
Epoch #60 Error:0.773611111111111
Epoch #61 Error:0.773611111111111
Epoch #62 Error:0.773611111111111
Epoch #63 Error:0.773611111111111
Epoch #64 Error:0.773611111111111
Epoch #65 Error:0.773611111111111
Epoch #66 Error:0.773611111111111
Epoch #67 Error:0.773611111111111
Epoch #68 Error:0.773611111111111
Epoch #69 Error:0.773611111111111
Epoch #70 Error:0.773611111111111
Epoch #71 Error:0.773611111111111
Epoch #72 Error:0.773611111111111
Epoch #73 Error:0.773611111111111
```

Sample1.txt
Sample10.txt
Sample11.txt
Sample12.txt
Sample13.txt
Sample14.txt
Sample15.txt
Sample16.txt
Sample17.txt
Sample18.txt

Figure 6.3: Application to train the ANN

Encog AI framework is used to develop ANN part of the application.

6.6 Android Application Development

After successful training of the ANN, we now have trained ANN yet to be tested with real scenario. We ae developing android application to measure the attention of a student. Android application receives brain wave data from NeuroSky Mindwave. Android application uses previously mentioned trained Neural Network. As this is a real time application, data are coming rapidly to the application. So based on this data, Focus Gain measure the attentiveness of the application. Application provide following facilities.

- View of raw EEG data
- Measure the attention Level
- Provide Visual indictor to show the attention changes
- Vibrate alert to notify when attention is lost

ThinkGear dll and Android framework are used for development of the mobile application [Appendix B].

6.7 Summary

In this chapter we have discussed how EEG data are collected and remove the unnecessary artifacts from them. Then we apply FFT on brain wave data and convert them in to frequency domain. These domain related data are then used to train an ANN. Trained ANN is used within Android application to detect the attentiveness of a person. If person is not attentive then vibration alert is given.

Chapter 7

Evaluation

7.1 Introduction

In this chapter we are going to design an experiment to evaluate the software that we have designed. Here we discussed whether the objectives mentioned in earlier chapters are met and to what extent.

7.2 Experimental Design

This study used 20 test subjects (10 men and 10 women), with an average age of 23 years (the subjects' age ranged between 20 and 25 years).No EEG related training has been provided to the subjects.

To clearly identify the state of attentive EEG signals when students are learning, standard IELTS listening exam lecture material was used as the experiment material for this study. The experiment involved the test subjects listening to a segment of conversations and then answering related questions to ensure that the test subjects could concentrate during the experiment.

There are two scenarios involved in this experiment. Answering the questions with interference and without interference. During the data collection experiment, 40 sessions of unprocessed entries of EEG row data were collected. The length of each session was 5minute s. The test subjects' conditions were manually determined

Now for each person we have calculated mean attention level using the system. For each person we have two sessions which are predetermined as attentive session and non-attentive session. For both sessions we calculate mean attentive level using the system.

7.3 Experimental Results

Following table shows the mean attention level of subjects in attentive and non-attentive sessions.

Subject	Mean Attentive Level (Attentive session)	Mean Attentive Level (Non - Attentive session)
Subject 1	75	20
Subject 2	77	18
Subject 3	88	9
Subject 4	61	5
Subject 5	42	56
Subject 6	22	87
Subject 7	16	33
Subject 8	91	21
Subject 9	60	14
Subject 10	44	17
Subject 11	55	29
Subject 12	78	41
Subject 13	80	34
Subject 14	61	16
Subject 15	42	25

Subject 16	77	21
Subject 17	92	55
Subject 18	85	18
Subject 19	53	24
Subject 20	17	30

Table 7.1: Results from the experiment

7.4 Conclusions from the Experiment

From the above results we can notice that 75% (15/20) success rate in attentive session and regarding the non- attentive session's success rate is 85% (17/20). Focus Gain application is successful in identifying attentive and non-attentive sessions correctly in most cases. These results are gender independent. We have same number of male and female subjects in the test pool. So the experiment results are more generalized. The accuracy can be further improved if we can train the ANN using larger data set.

If you analyze the results closely, you can see some anomalies in the results (Subject 5, 6 and 20). Reason for this may be due to background interference or system is not able to classify the individual's data correctly. So we need to use more sophisticated artifact removal methods and more training to the ANN in order to overcome these kind of anomalies.

7.5 Summary

In this chapter we have evaluated the software solution using proper experiment mechanism. We have designed and conducted experiment to measure the success of the application and presented the results.

Chapter 8

Conclusion

8.1 Introduction

This report presents results that shows that is it possible to build a Brain-Computer Interface system that allow users to measure attention using EEG headset. This have been accomplished by using the NeuroSky mindset EEG equipment featuring only one electrode on the forehead. EEG signals from the user are sent to the android application via bluetooth. The signal is then processed and the wave's band power is calculated. This information is used as input to a neural network that is trained to classify whether student is attentive or not. Then this classification is used to notify the student when the attention has been lost.

8.2 Conclusions

Using ANN to achieve this purpose is hugely successful as after training the ANN, it can process data in real time provide feedback within very short time. Analyzing the experimental results we can notice the success of the project. We could have achieved this using statistical methods, but real time processing is vital to make this project a success. So implementing ANN to monitor the student attention in classroom environment is justified with the test results. Experiment shows 75% of success rate in identifying the attentive sessions and 85% success in identifying non-attentive sessions.

Following objectives mentioned early in the thesis are met successfully during the execution of the research.

The aim of this project is to develop a system for recognizing the student attention in classroom environment. So the aim of the project is successfully achieved.

In order to reach this aim following objectives are identified.

1. To critically study the domain of brain computer interface field.
2. To critically analyze the current approaches to recognize the attention level in different environments.
3. Recognizing the suitable brain wave monitoring appliances to recognize the attention level in class room environment.
4. Identify processing libraries and environments which can be utilized to analyze the raw EEG signals.
5. Acquire raw brain wave data on attentive sessions from group of students.
6. Develop a mobile application to alert the student when he is not attentive in the lecture.

All the objectives are met during the execution of the project.

8.3 Limitations and Further Work

This application is only developed to monitor the attention level in class room environment. As per further work one can extend this research to monitor the overall attention not just in the classroom environment. We are only using one dry sensor for our purpose. With more sensors we can achieve more accuracy in the future. Mobile application can be further enhanced to record the attention details, so later users can analyze the details and made actions to sustain the attention level.

This can also be used as commercial application where parents can use this as a tool to measure the attention of their children. It is very hard to say whether a student is focusing on the lecture or not. But with this kind of application, we can easily get to know the real utilization of a student. Hence we can advise student on how to focus on a lecture if he has constantly losing the attention.

8.4 Summary

In this chapter we have discussed the conclusions that we can finally derived from the research. We consider all the aspects of the project, design, implementation, evaluation. Based on all the facts, we made some final conclusions here. Problems encountered, limitations of the solution and further work are also discussed here.

References

[1] M. H. Alomari, A. AbuBaker, A. Turani, A. M. Baniyounes, and A. Manasreh, "EEG Mouse: A Machine Learning-Based Brain Computer Interface," *Int. J. Adv. Comput. Sci. Appl.*, vol. 5, no. 4, 2014.

[2] T. Publication{"id":3775225, first_name":"TJPRC, last_name":"Publication, page_name":"TJPRCPublication, domain_name":"independent, "display_name":"TJPRC Publication", url":"http://independent.academia.edu/TJPRCPublication, "is_analytics_public":false, photo":"/images/s65_no_pic_borderless.gif, "interests":{"top":[], and "count":0}}, "BRAIN CONTROLLED WHEELCHAIR FOR DISABLED." [Online]. Available: https://www.academia.edu/6986487/BRAIN_CONTROLLED_WHEELCHAIR_FOR_DISABLED. [Accessed: 12-Aug-2014].

[3] R. Bogacz, U. Markowska-Kaczmar, and A. Kozik, "Blinking artefact recognition in EEG signal using artificial neural network," in *Proc. of 4 th Conference on Neural Networks and Their Applications, Zakopane (Poland)*, 1999.

[4] "Community: The Mastermind Project-MIND CONTROLLED ROBOTS USING EEG - National Instruments." [Online]. Available: https://decibel.ni.com/content/docs/DOC-24767. [Accessed: 12-Aug-2014].

[5] F. Ebrahimi, M. Mikaeili, E. Estrada, and H. Nazeran, "Automatic sleep stage classification based on EEG signals by using neural networks and wavelet packet coefficients," in *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, 2008, pp. 1151–1154.

[6] C.-K. A. Lim and W. C. Chia, "Analysis of Single-Electrode EEG Rhythms Using MATLAB to Elicit Correlation with Cognitive Stress."

[7] E. Niedermeyer and F. H. Lopes da Silva, *Electroencephalography: basic principles, clinical applications, and related fields*. Philadelphia: Lippincott Williams & Wilkins, 2005.

[8] G. V. Sridhar and P. M. Rao, "A Neural Network Approach for EEG Classification in BCI."

[9] R. Singla and N. Sharma, "Function Classification of EEG Signals Based on ANN."

[10] W. SA\LABUN, “Processing and spectral analysis of the raw EEG signal from the MindWave,” *Przeglad Elektrotechniczny*, vol. 90, pp. 169–174, 2014.

[11] G. Navalyal and R. D. Gavas, “A Dynamic Approach to Foster Cognitive Computing using the Laws of Thought.”

[12] S. Rodrak and Y. Wongsawat, “Dept. of Biomed. Eng., Mahidol Univ., Bangkok, Thailand,” in *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*, 2012, pp. 1–4.

[13] N.-H. Liu, C.-Y. Chiang, and H.-C. Chu, “Recognizing the Degree of Human Attention Using EEG Signals from Mobile Sensors,” *Sensors*, vol. 13, no. 8, pp. 10273–10286, Aug. 2013.

[14] “855-classification-of-electroencephalogram-using-artificial-neural-networks.pdf.” .

[15] E. A. Larsen, “Classification of EEG Signals in a Brain-Computer Interface System,” 2011.

[16] https://en.wikipedia.org/wiki/10-20_system_(EEG)

Appendix A

Data Acquisition using NeuroSky Headset

A.1 Introduction

It all start with acquiring data from NeuroSky headset. We need to find a way to get the EEG data from NeuroSky headset. We need to understand the formats and conventions used by NeuroSky headset first.

A.2 NeuroSky Mobile Mindwave Headset

ThinkGear is the technology inside every NeuroSky product or partner product that enables the device to interface with the wearers' brainwaves. It includes:

- The sensor that touches the forehead,
- The contact and reference points located on the ear pad, and
- The onboard chip that processes all of the data.
- Both the raw brainwaves and the eSense Meters are calculated on the ThinkGear™ chip. The calculated values are output by the ThinkGear chip, through the headset, to a PC.

Types of data output from ThinkGear chips:

- Raw sampled wave values (128Hz or 512Hz, depending on hardware)
- Signal poor quality metrics
- EEG band power values for delta, theta, alpha, beta, and gamma

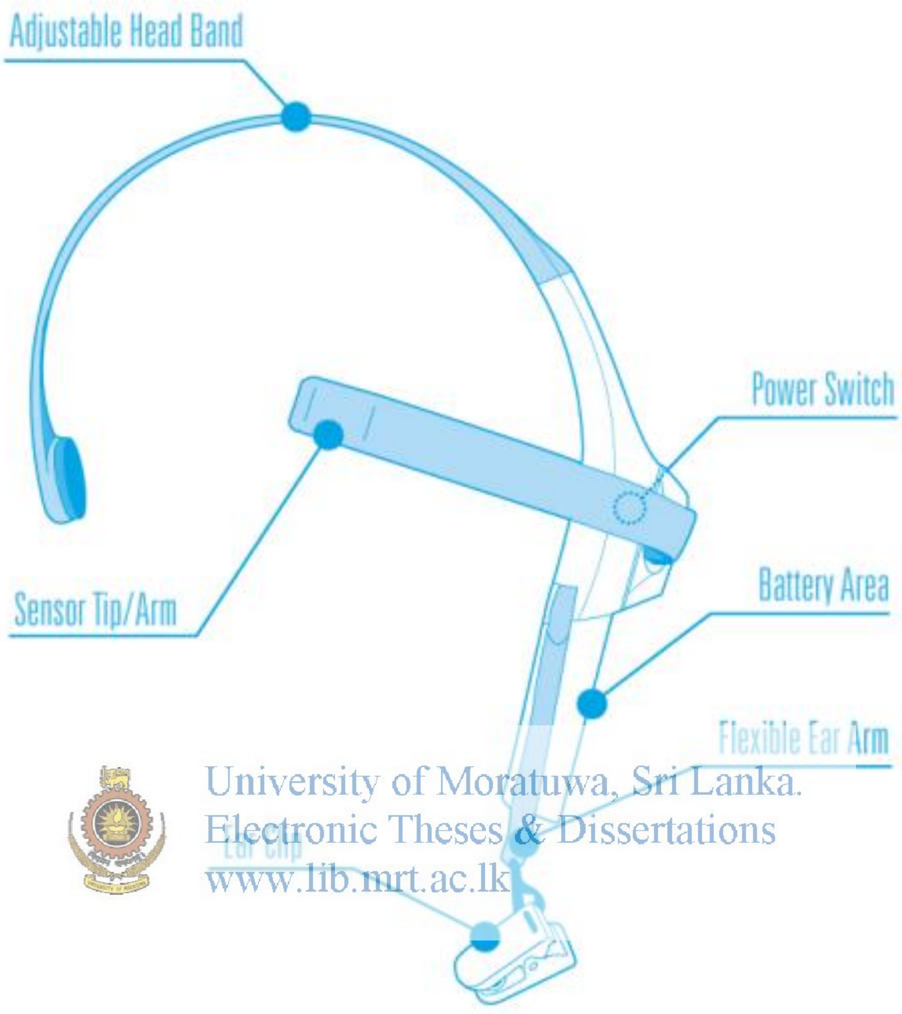
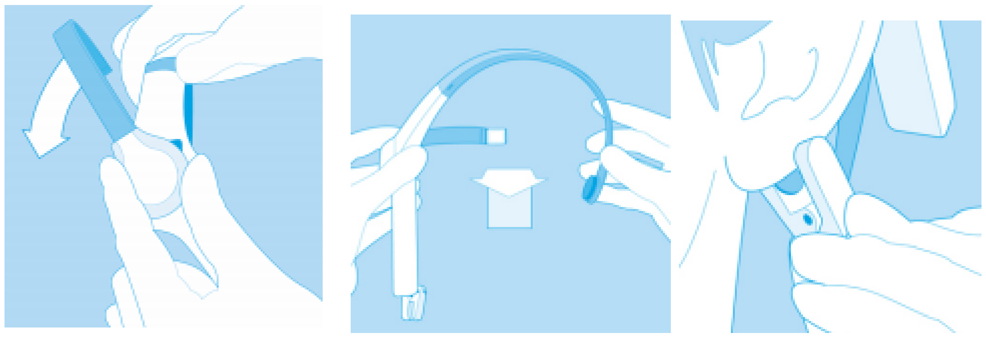


Figure A.1: NeuroSky Headset

To power on the MindWave Mobile headset, slide the switch to the ON (middle) position. When held past the ON position for 3 seconds and then released, the headset will enter Bluetooth pairing mode. If instead the switch is held past the ON position for 6 seconds, the headset's pairing memory will be cleared. While the MindWave Mobile headset is powered on, the LED light on the side of the headset will be turned on. If the MindWave has a low battery, the LED light will flash to indicate low battery status. To turn the MindWave Mobile off, slide the switch back to the OFF position.

A.3 Fitting NeuroSky Mobile Mindwave Headset

- Orient the MindWave with the forehead Sensor Arm on your left hand side. Rotate the Sensor Arm from its base by about 90 degrees. It can be rotated slightly more if necessary to get proper fit and comfort.

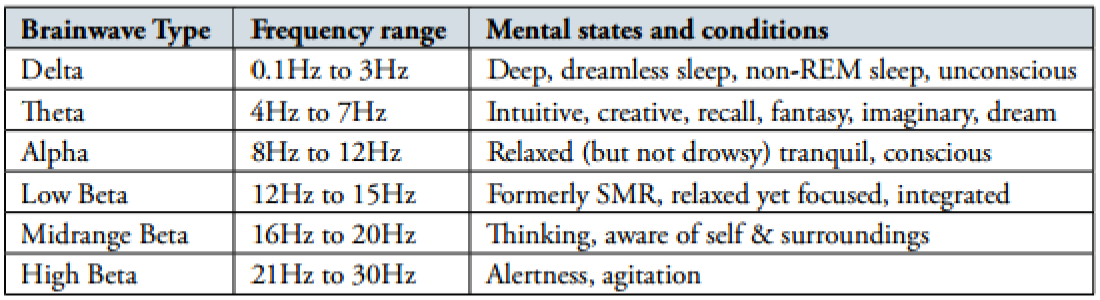


- The overhead band of the MindWave is adjustable and can be extended to fit various sizes. Put on the MindWave. If the sensor does not make contact with the forehead or if the fit is not comfortable, remove the MindWave to readjust the overhead band and the forehead Sensor Arm.

- Make sure the two metal contacts on the inside of both sides of the ear clip make skin contact with your earlobe or ear. Move any hair or obstructions (such as jewelry) out of the way. Readjust the ear clip as necessary to make proper contact with the skin of your ear. You may need to squeeze the ear clip against your ear for a few seconds.

A.4 ThinkGear Technology

The electrical signals emitted by neurons generating in the brain. These patterns and frequencies of these electrical signals can be measured by placing a sensor on the scalp. The Mind Tools line of headset products contain NeuroSky ThinkGear™ technology, which measures the analog electrical signals, commonly referred to as brainwaves, and processes them into digital signals. The ThinkGear technology then makes those measurements and signals available to games and applications. The table below gives a general synopsis of some of the commonly-recognized frequencies that tend to be generated by different types of activity in the brain:



Brainwave Type	Frequency range	Mental states and conditions
Delta	0.1Hz to 3Hz	Deep, dreamless sleep, non-REM sleep, unconscious
Theta	4Hz to 7Hz	Intuitive, creative, recall, fantasy, imaginary, dream
Alpha	8Hz to 12Hz	Relaxed (but not drowsy) tranquil, conscious
Low Beta	12Hz to 15Hz	Formerly SMR, relaxed yet focused, integrated
Midrange Beta	16Hz to 20Hz	Thinking, aware of self & surroundings
High Beta	21Hz to 30Hz	Alertness, agitation

Figure A.2: Brainwave Types, frequencies and mental conditions map

ThinkGear Data Types

POOR_SIGNAL/SENSOR_STATUS

This is integer value provides an indication of how good or how poor the bio-signal is at the sensor. This value is typically output by all ThinkGear hardware devices once per second. This is an extremely important value for any app using ThinkGear sensor hardware to always read, understand, and handle. Depending on the use cases for your app and users, your app may need to alter the way it uses other data values depending on the current value of POOR_SIGNAL/SIGNAL_STATUS.

For example, if this value is indicating that the bio-sensor is not currently contacting the subject, then any received RAW_DATA or EEG_POWER values during that time should be treated as noise not from a human subject, and possibly discarded based on the needs of the app.

Poor signal may be caused by a number of different things. In order of severity, they are:

- Sensor, ground, or reference electrodes not being on a person's head/body
- Poor contact of the sensor, ground, or reference electrodes to a person's skin
- Excessive motion of the wearer

• Excessive environmental electrostatic noise (some environments have strong electric signals or static electricity buildup in the person wearing the sensor).

• Excessive biometric noise (i.e. unwanted EMG, EKG/ECG, EOG, EEG, etc. signals)

RAW_DATA

This data type supplies the raw sample values acquired at the bio-sensor. The sampling rate (and therefore output rate), possible range of values, and interpretations of those values (conversion from raw units to volt) for this data type are dependent on the hardware characteristics of the ThinkGear hardware device performing the sampling. You must refer to the documented development specs of each type of ThinkGear hardware that your app will support for details.

As an example, the majority of inkGear devices sample at 512Hz, with a possible value range of -32768 to 32767.

As another example, to convert TGAT-based EEG sensor values (such as TGAT, TGAM, MindWave Mobile, MindWave, MindSet) to voltage values, use the following conversion:

(rawValue * (1.8/4096)) / 2000

EEG_POWER

This Data Value represents the current magnitude of 8 commonly-recognized types of EEG frequency bands.

The eight EEG powers are: delta (0.5 - 2.75Hz), theta (3.5 - 6.75Hz), low-alpha (7.5 - 9.25Hz), high-alpha (10 - 11.75Hz), low-beta (13 - 16.75Hz), high-beta (18 - 29.75Hz), low-gamma (31 - 39.75Hz), and mid-gamma (41 - 49.75Hz).

These values have no units and are only meaningful for comparison to the values for the other frequency bands within a sample. By default, output of this Data Value is enabled, and it is output approximately once a second.

A.5 ThinkGear Communication Driver

As per this research we are using ThinkGear Communication Driver shipped with NeuroSky Headset. It does not have fancy API like the .NET version. So we have listen to a port where the data are transferred and manipulate the data.

A.6 Android Application Development with NeuroSky MindWave Mobile

- Open the Settings app on the Android device
- Navigate to Wireless and network and enable Bluetooth if not already enabled
- Go to Bluetooth settings
- Power on the MindWave Mobile
- MindWave Mobile will show up in the list of devices
- Touch MindWave Mobile and pairing will complete automatically

Appendix B

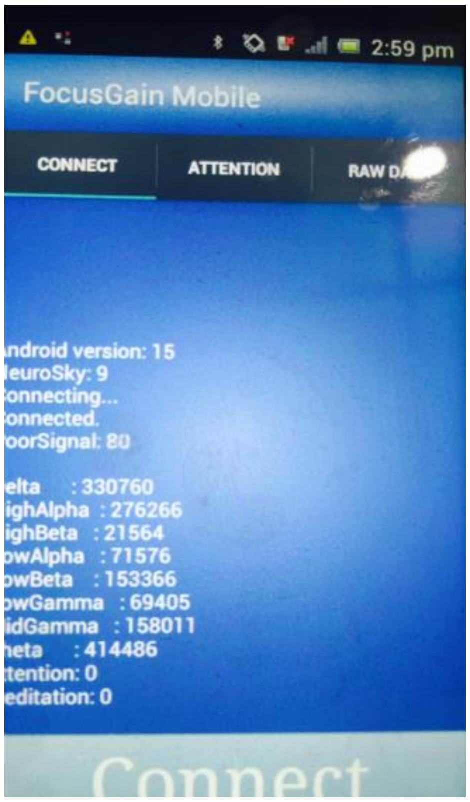
FocusGain Application

B.1 Introduction

This will walk you through the procedure involve with developing Android application for the NeuroSky headset. This is the final output of the project, we named the application as "FocusGain"

B.2 FocusGain Android Application Design

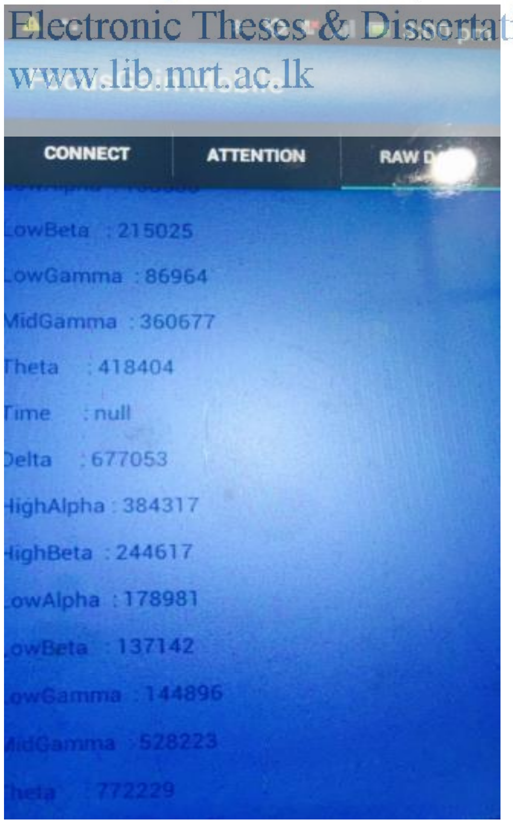
Figure B.1: FocusGain Android Application



```
ndroid version: 15
leuroSky: 9
onnecting...
onnected.
oorSignal: 80

elta      : 330760
ighAlpha  : 276266
ighBeta   : 21564
owAlpha   : 71576
owBeta    : 153366
owGamma   : 69405
idGamma   : 158011
heta      : 414486
ttention: 0
editation: 0
```

Figure B.2: FocusGain Android Application – Connect Pane



```
LowBeta   : 215025
LowGamma  : 86964
MidGamma  : 360677
Theta     : 418404
Time      : null
Delta     : 677053
HighAlpha : 384317
HighBeta  : 244617
LowAlpha  : 178981
LowBeta   : 137142
LowGamma  : 144896
MidGamma  : 528223
Theta     : 772229
```

Figure B.3: FocusGain Android Application – Raw Data Pane

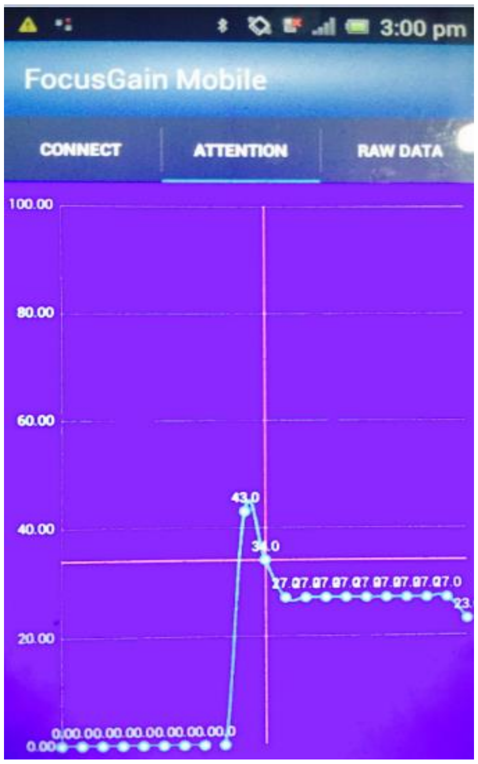


Figure B.4: FocusGain Android Application – Attention Monitoring

Appendix C

Code - FocusGain Application

C.1 Introduction

This section will include code used to develop this application.

C.2 Code for ThinkGear Connection

ThinkGearSocketClient.java

```
package focusgain.eeg.thinkgear;


import java.io.IOException;
import java.net.InetSocketAddress;
import java.nio.CharBuffer;
import java.nio.channels.SocketChannel;
import java.nio.charset.Charset;
import java.nio.charset.CharsetEncoder;
import java.util.Scanner;
import javax.swing.JOptionPane;
import org.apache.log4j.Logger;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author Charitha Senarathne
 */
public class ThinkGearSocketClient
{

   /**
    * Logger for this class
    */
   private static final Logger logger = Logger.getLogger(ThinkGearSocketClient.class);

   public static final String DEFAULT_HOST = "127.0.0.1";
   public static final int DEFAULT_PORT = 13854;
```

```
private static ThinkGearSocketClient INSTANCE = null;

private String host;
private int port;
private boolean connected;
SocketChannel channel;
Scanner in;

/**
 * Default constructor using Thinkgear default host/port
 */
private ThinkGearSocketClient()
{

   this.host = DEFAULT_HOST;
   this.port = DEFAULT_PORT;
   this.connected = false;

}

/**
 * Constructor
 *
 * @param host
 * @param port
 */
public ThinkGearSocketClient(String host, int port)
{

   this.host = host;
   this.port = port;
   this.connected = false;

}

public static ThinkGearSocketClient getInstance()
{
   if(INSTANCE == null)
   {
      INSTANCE = new ThinkGearSocketClient();
   }

   return INSTANCE;
}
```

```
    public String getHost()
    {
        return host;
    }

    public void setHost(String host)
    {
        this.host = host;
    }

    public int getPort()
    {
        return port;
    }

    public void setPort(int port)
    {
        this.port = port;
    }

    public boolean isConnected()
    {
        return this.connected;
    }

    public void connect(boolean enableRawData) throws IOException
    {

        if (!this.connected)
        {
            logger.debug("connect() - Starting new connection...");
            this.channel = SocketChannel.open(new InetSocketAddress(this.host, this.port));

            CharsetEncoder enc = Charset.forName("US-ASCII").newEncoder();
            //String jsonCommand = "{\"timestamp\":true,\"enableRawOutput\":" +
Boolean.toString(enableRawData) + ", \"format\": \"Json\"}\n";
            String jsonCommand = "{\"timestamp\":true,\"enableRawOutput\": false, \"format\":
\"Json\"}\n";
            this.channel.write(enc.encode(CharBuffer.wrap(jsonCommand)));

            this.in = new Scanner(channel);
            this.connected = true;

            System.out.println("Connected");
        } else
        {
            logger.debug("connect() - Already connected...");
            System.out.println("Already connected");
        }

    }

    public void startRecording() throws IOException
    {
        if (this.connected)
        {
            CharsetEncoder enc = Charset.forName("US-ASCII").newEncoder();

            String                                  jsonCommand                                  =
"{\"startRecording\":{\"rawEeg\":true,\"poorSignalLevel\":true,\"eSense\":true,\"eegPower\":
true,\"blinkStrength\":true},\"applicationName\":\"ExampleApp\"}\n";
            this.channel.write(enc.encode(CharBuffer.wrap(jsonCommand)));

            JOptionPane.showMessageDialog(null, "Start Recording");

        } else
        {
            logger.debug("startRecording() - Not connected...");
        }

    }

    public void stopRecording() throws IOException
    {
        if (this.connected)
        {
            CharsetEncoder enc = Charset.forName("US-ASCII").newEncoder();

            String jsonCommand = "{\"stopRecording\":\"ExampleApp\"}\n";
            this.channel.write(enc.encode(CharBuffer.wrap(jsonCommand)));

            JOptionPane.showMessageDialog(null, "Stop Recording");

        } else
        {
            logger.debug("stopRecording() - Not connected...");
        }
    }

    public void cancelRecording() throws IOException
    {
        if (this.connected)
        {
            CharsetEncoder enc = Charset.forName("US-ASCII").newEncoder();

            String jsonCommand = "{\"cancelRecording\":\"ExampleApp\"}\n";
            this.channel.write(enc.encode(CharBuffer.wrap(jsonCommand)));

        } else
        {
            logger.debug("cancelRecording() - Not connected...");
        }
    }

    public boolean isDataAvailable()
    {
        if (this.connected)
        {
            return this.in.hasNextLine();
        } else
        {
            return false;
        }
    }

    public String getData()
    {
        return this.in.nextLine();
    }

    public void close() throws IOException
    {

        if (this.connected)
        {
            logger.debug("close() - Closing connection...");
            this.in.close();
            this.channel.close();
            this.connected = false;
        }
    }

}
```

C.3 Code for EEG Data Collector

EEGDataCollector.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package focusgain.eeg.read;

import focusgain.eeg.thinkgear.ThinkGearSocketClient;
import focusgain.eeg.ui.EEGDataCollectorUI;
import focusgain.eeg.ui.EEGVisualizer;
import static focusgain.eeg.ui.FocusGainUI.eegDataTxtArea;
import focusgain.eeg.utilities.TimerUtil;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

/**
 *
 * @author Chartha Senarathne
 */
public class EEGDataCollector
{

    public static void initialiazeConnection()
    {

        new Thread(new Runnable()
        {
            @Override
            public void run()
            {
                while (!(EEGDataCollector.initialize()))
                {
                    try
                    {
```

```
                    Thread.sleep(10 * 1000);

                } catch (InterruptedException ex)
                {
Logger.getLogger(EEGDataCollectorUI.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
        }
    }).start();
}

private static boolean initialize()
{
    ThinkGearSocketClient client = ThinkGearSocketClient.getInstance();
    try
    {
        client.connect(EEGDataCollectorUI.rawDataCheckBox.isSelected());

    } catch (IOException ex)
    {
        return false;
    }
    return true;
}

public static void startSession()
{
    TimerUtil.startTimer();
    EEGDataCollectorUI.eegDataTxtArea.append("Starting Session\n");
    writeToFile("EEG Sample Data for Attention");

    ThinkGearSocketClient client = ThinkGearSocketClient.getInstance();

    if (client != null)
    {
        while (client.isDataAvailable())
        {
            writeToFile(client.getData());

            String currentEEGReading = client.getData();

            if (currentEEGReading.contains("eSense"))
            {
```

```
EEGDataCollectorUI.eegDataTxtArea.append(currentEEGReading.substring(currentEEGRe
ading.indexOf("eegPower")) + "\n");

EEGDataCollectorUI.eegDataTxtArea.setCaretPosition(EEGDataCollectorUI.eegDataTxtAr
ea.getText().length());
                }
                else
                {
                    EEGDataCollectorUI.eegDataTxtArea.append(client.getData() + "\n");

EEGDataCollectorUI.eegDataTxtArea.setCaretPosition(EEGDataCollectorUI.eegDataTxtAr
ea.getText().length());
                }

            }
        }

    }

    public static void stopSession()
    {
        if (ThinkGearSocketClient.getInstance() != null)
        {
            EEGDataCollectorUI.eegDataTxtArea.append("Stoping Session\n");
            TimerUtil.stopTimer();

            try
            {
                ThinkGearSocketClient.getInstance().close();
            } catch (IOException ex)
            {
                Logger.getLogger(EEGDataCollector.class.getName()).log(Level.SEVERE, null,
ex);
            }
        }

    }

    private static void writeToFile(String data)
    {
        FileWriter fileWriter = null;
        try
        {
            String fileName = EEGDataCollectorUI.sampleCaptionTxt.getText();
            if (fileName.equalsIgnoreCase(""))
            {
               DateFormat dateFormat = new SimpleDateFormat("yyyy_MM_dd_HH_mm_ss");
               Date date = new Date();
               fileName = dateFormat.format(date);

               EEGDataCollectorUI.sampleCaptionTxt.setText(fileName);
            }
            fileName = fileName + ".txt";
            File file = new File( fileName);

            //if file doesnt exists, then create it
            if (!file.exists())
            {
               file.createNewFile();
            }
            //true = append file

            fileWriter = new FileWriter(file.getName(), true);
            BufferedWriter bufferWriter = new BufferedWriter(fileWriter);
            String writeData = data + "\n";
            bufferWriter.write(writeData);
            bufferWriter.close();

         } catch (IOException ex)
         {
            Logger.getLogger(EEGDataCollector.class.getName()).log(Level.SEVERE, null, ex);

         } finally
         {
            try
            {
               fileWriter.close();
            } catch (IOException ex)
            {
               Logger.getLogger(EEGDataCollector.class.getName()).log(Level.SEVERE, null,
ex);
            }
         }
      }
   }
```

C.4 Code for ANN Trainer

<u>AttentionClassificationNeuralNetwork.java</u>

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package focusgain.eeg.ann;

import focusgain.eeg.ui.ANNTrainerUI;
import java.awt.Color;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import javax.swing.JOptionPane;
import org.encog.engine.network.activation.ActivationSigmoid;
import org.encog.neural.data.NeuralData;
import org.encog.neural.data.NeuralDataSet;
import org.encog.neural.data.basic.BasicNeuralData;
import org.encog.neural.data.basic.BasicNeuralDataSet;
import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.layers.BasicLayer;
import org.encog.neural.networks.layers.Layer;
import org.encog.neural.networks.synapse.Synapse;
import org.encog.neural.networks.synapse.WeightedSynapse;
import org.encog.neural.networks.training.Train;
import org.encog.neural.networks.training.propagation.back.Backpropagation;
import org.encog.neural.networks.training.propagation.resilient.ResilientPropagation;

/**
 *
 * @author charitha.s
 */
public class AttentionClassificationNeuralNetwork
{

    private BasicNetwork network;
    private int inputNeuronCount = 8;
    private int hiddenNeuronCount = 10;
    private int outputNeuronCount = 1;
    private double learnRate = 0.7;
    private double momentum = 0.8;
    private double error = 0.2;

    private static final int trainingSetSize = 60 * 60;
    private static final String sampleDataPath = "/Sample_EEG_DATA";
    private static final double attentiveBit = 1;
    private static final double notAttentiveBit = 0;
    private static final double ACCEPTED_SIGNAL_LEVEL = 150;

    public double INPUT[][] = new double[trainingSetSize][inputNeuronCount];
    public double IDEAL[][] = new double[trainingSetSize][outputNeuronCount];

    public void createNetwork()
    {
        setNetwork(new BasicNetwork());

        Layer outputLayer = new BasicLayer(new ActivationSigmoid(), true,
outputNeuronCount);
        Layer hiddenLayer = new BasicLayer(new ActivationSigmoid(), true,
hiddenNeuronCount);
        Layer inputLayer = new BasicLayer(new ActivationSigmoid(), false,
inputNeuronCount);

        Synapse synapseHiddenToOutput = new WeightedSynapse(hiddenLayer, outputLayer);
        Synapse synapseInputToHidden = new WeightedSynapse(inputLayer, hiddenLayer);

        hiddenLayer.addSynapse(synapseHiddenToOutput);
        inputLayer.addSynapse(synapseInputToHidden);

        getNetwork().tagLayer("INPUT", inputLayer);
        getNetwork().tagLayer("OUTPUT", outputLayer);

        getNetwork().getStructure().finalizeStructure();
        getNetwork().reset();
    }

    public boolean train()
    {
        boolean trainingStatus = false;
        final String sampleDataPathDir = "Sample_EEG_DATA";
        File sampleEEGDataPath = new File(sampleDataPathDir);
        if (sampleEEGDataPath.listFiles() != null)
        {
            for (final File fileEntry : sampleEEGDataPath.listFiles())
            {
                ANNTrainerUI.trainingDataSetList.setSelectedValue(fileEntry.getName(), true);
                ANNTrainerUI.trainingDataSetList.setSelectionForeground(Color.GREEN);

                if (fileEntry.isDirectory())
                {
                    continue;

                } else
                {
                    NeuralDataSet trainingSet = getTrainingSet(fileEntry.getAbsolutePath());
                    if (trainingSet != null)
                    {
                        if (!ANNTrainerUI.learningRateText.getText().equals(""))
                        {
                            learnRate =
Double.parseDouble(ANNTrainerUI.learningRateText.getText());
                        }
                        if (!ANNTrainerUI.momentumText.getText().equals(""))
                        {
                            momentum =
Double.parseDouble(ANNTrainerUI.momentumText.getText());
                        }
                        if (!ANNTrainerUI.errorText.getText().equals(""))
                        {
                            error = Double.parseDouble(ANNTrainerUI.errorText.getText());
                        }

                        trainingStatus = trainDataSet(trainingSet);

                    } else
                    {
                        trainingStatus = false;
                    }
                }
            }
        }
        return trainingStatus;
    }

    private boolean trainDataSet(NeuralDataSet trainingSet)
    {
        final Train train = new Backpropagation(getNetwork(), trainingSet, learnRate,
momentum);
```

```
        //final Train train = new ResilientPropagation(getNetwork(), trainingSet);
        int epoch = 0;
        do
        {
            //method is called over and over; each time the network is slightly adjusted for a
better error rate.
            //The following loop will loop and train the neural network until the error rate has
fallen below one percent.
            train.iteration();

            String resultLine = "Epoch #" + epoch + " Error:" + train.getError();
            ANNTrainerUI.resultTextArea.append(resultLine + "\n");

ANNTrainerUI.resultTextArea.setCaretPosition(ANNTrainerUI.resultTextArea.getText().len
gth());
            System.out.println(resultLine);
            epoch++;

            System.out.println("serializing the network");

            try
            {
                FileOutputStream fout = new
FileOutputStream("attention_classification_network.dat");
                ObjectOutputStream oos = new ObjectOutputStream(fout);
                oos.writeObject(getNetwork());
                oos.close();

            } catch (IOException e)
            {
                JOptionPane.showMessageDialog(null, e.getMessage());
            }
            System.gc();

        } while (train.getError() > error);
        System.gc();

        return true;
    }

    private NeuralDataSet getTrainingSet(String sampleDataFilePath)
    {
        NeuralDataSet neuralDataSet = null;
        BufferedReader br = null;
        try
        {
```

```
        br = new BufferedReader(new FileReader(sampleDataFilePath));

        int i = 0;
        String sCurrentLine;
        while ((sCurrentLine = br.readLine()) != null && i <= trainingSetSize)
        {
           if (sCurrentLine.contains("\"eSense\""))
           {
              double attention = 0, meditation = 0, delta = 0, theta = 0, lowAlpha = 0,
highAlpha = 0, lowBeta = 0, highBeta = 0, lowGamma = 0, highGamma = 0,
poorSignalLevel = 0;
              sCurrentLine = sCurrentLine.replace("\"", "");
              sCurrentLine = sCurrentLine.replace("{", "");
              sCurrentLine = sCurrentLine.replace("}", "");
              sCurrentLine = sCurrentLine.replace("eSense:", "");
              sCurrentLine = sCurrentLine.replace("eegPower:", "");

              String[] eegValues = sCurrentLine.split(",");
              for (String eegVal : eegValues)
              {
                 String[] eegReadings = eegVal.split(":");
                 double tempReading = Double.parseDouble(eegReadings[1]);
                 switch (eegReadings[0])
                 {
                    case "attention":
                       attention = tempReading;
                       break;
                    case "meditation":
                       meditation = tempReading;
                       break;
                    case "delta":
                       delta = tempReading;
                       break;
                    case "theta":
                       theta = tempReading;
                       break;
                    case "lowAlpha":
                       lowAlpha = tempReading;
                       break;
                    case "highAlpha":
                       highAlpha = tempReading;
                       break;
                    case "lowBeta":
                       lowBeta = tempReading;
                       break;
                    case "highBeta":
                        highBeta = tempReading;
                        break;
                    case "lowGamma":
                        lowGamma = tempReading;
                        break;
                    case "highGamma":
                        highGamma = tempReading;
                        break;
                    case "poorSignalLevel":
                        poorSignalLevel = tempReading;
                        break;
                    default:
                        break;
                }
            }

            if (poorSignalLevel < ACCEPTED_SIGNAL_LEVEL)
            {
                INPUT[i][0] = delta;
                INPUT[i][1] = theta;
                INPUT[i][2] = lowAlpha;
                INPUT[i][3] = highAlpha;
                INPUT[i][4] = lowBeta;
                INPUT[i][5] = highBeta;
                INPUT[i][6] = lowGamma;
                INPUT[i][7] = highGamma;

                for (int j = 0; j < outputNeuronCount; j++)
                {
                    IDEAL[i][j] = attentiveBit;
                }
                ++i;
            }

        } else
        {
        }

    }

    neuralDataSet = new BasicNeuralDataSet(INPUT, IDEAL);
    br.close();

} catch (IOException ex)
{
    JOptionPane.showMessageDialog(null, ex.getMessage());
```

```
        ANNTrainerUI.progressBar.setIndeterminate(false);
        ANNTrainerUI.trainBtn.setEnabled(true);

    }
     return neuralDataSet;
}

public boolean resumeTraining()
{
    boolean trainingStatus = false;

    setNetwork(new BasicNetwork());
    System.out.println("deserializing the network");
    try
    {
        FileInputStream fin = new FileInputStream("attention_classification_network.dat");

        ObjectInputStream ois = new ObjectInputStream(fin);
        setNetwork((BasicNetwork) ois.readObject());
        ois.close();

    } catch (IOException | ClassNotFoundException e)
    {
        JOptionPane.showMessageDialog(null, e.getMessage());
    }

    System.out.println(getNetwork().toString());

    final String sampleDataPathDir = "Sample_EEG_DATA";
    File sampleEEGDataPath = new File(sampleDataPathDir);
    if (sampleEEGDataPath.listFiles() != null)
    {
        for (final File fileEntry : sampleEEGDataPath.listFiles())
        {
            ANNTrainerUI.trainingDataSetList.setSelectedValue(fileEntry.getName(), true);
            ANNTrainerUI.trainingDataSetList.setSelectionForeground(Color.GREEN);

            if (fileEntry.isDirectory())
            {
                continue;

            } else
            {
                NeuralDataSet trainingSet = getTrainingSet(fileEntry.getAbsolutePath());
                if (trainingSet != null)
                {
                    if (!ANNTrainerUI.learningRateText.getText().equals(""))
                    {
                        learnRate =
Double.parseDouble(ANNTrainerUI.learningRateText.getText());
                    }
                    if (!ANNTrainerUI.momentumText.getText().equals(""))
                    {
                        momentum =
Double.parseDouble(ANNTrainerUI.momentumText.getText());
                    }
                    if (!ANNTrainerUI.errorText.getText().equals(""))
                    {
                        error = Double.parseDouble(ANNTrainerUI.errorText.getText());
                    }

                    trainingStatus = trainDataSet(trainingSet);

                } else
                {
                    trainingStatus = false;
                }
            }
        }
    }
    return trainingStatus;
}


public NeuralData computeOutput(double[] INPUT)
{
    setNetwork(new BasicNetwork());
    System.out.println("deserializing the network");
    try
    {
        FileInputStream fin = new FileInputStream("attention_classification_network.dat");

        ObjectInputStream ois = new ObjectInputStream(fin);
        setNetwork((BasicNetwork) ois.readObject());
        ois.close();

    } catch (IOException | ClassNotFoundException e)
    {
        JOptionPane.showMessageDialog(null, e.getMessage());
    }

    NeuralData output = getNetwork().compute(new BasicNeuralData(INPUT));
```

```
        return output;
    }


    /**
     * @return the network
     */
    public BasicNetwork getNetwork()
    {
        return network;
    }

    /**
     * @param network the network to set
     */
    public void setNetwork(BasicNetwork network)
    {
        this.network = network;
    }
}
```

C.5 Code for FocusGain Mobile Application

ConnectFragment.java

```
package mobile.focusgain.focusgainmobile.fragments;

import android.bluetooth.BluetoothAdapter;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.os.Vibrator;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.text.method.ScrollingMovementMethod;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.neurosky.thinkgear.TGDevice;
import com.neurosky.thinkgear.TGEegPower;

import java.text.SimpleDateFormat;
import java.util.Date;

import mobile.focusgain.focusgainmobile.R;
import mobile.focusgain.focusgainmobile.thinkgear.ThinkGearConnector;

/**
 * Created by Charitha Senarathne on 5/4/2015.
 */
public class ConnectFragment extends Fragment implements View.OnClickListener {

    BluetoothAdapter bluetoothAdapter;
    TextView tv;
    Button b;
    TGDevice tgDevice;


    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
@Nullable Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.connect, container, false);

        tv = (TextView) view.findViewById(R.id.textView);
        tv.setMovementMethod(new ScrollingMovementMethod());
        tv.setText("");
        tv.append("Android version: " + Integer.valueOf(android.os.Build.VERSION.SDK) +
"\n");

        b = (Button) view.findViewById(R.id.button);
        b.setOnClickListener(this);

        // Check if Bluetooth is available on the Android device
        bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        if (bluetoothAdapter == null) {

            // Alert user that Bluetooth is not available
            Toast.makeText(getActivity().getBaseContext(), "Bluetooth not available",
Toast.LENGTH_LONG).show();
            //finish();
        } else {
            // create the TGDevice
            tgDevice = new TGDevice(bluetoothAdapter, handler);
            ThinkGearConnector.setTgDevice(tgDevice);

            tv.append("NeuroSky: " + tgDevice.getVersion());
            tv.append("\n");
        }

        return view;
    }

    public void doStuff(View view) {

        if (ThinkGearConnector.getTGDevice() != null) {

            if (ThinkGearConnector.getTGDevice().getState() !=
TGDevice.STATE_CONNECTING && ThinkGearConnector.getTGDevice().getState() !=
TGDevice.STATE_CONNECTED) {

                ThinkGearConnector.getTGDevice().connect(rawEnabled);

            }
        } else {

            Toast.makeText(getActivity().getBaseContext(), "Bluetooth not available",
Toast.LENGTH_LONG).show();
        }
    }


    final boolean rawEnabled = false;
    int focusLostCount = 0;
    final Handler handler = new Handler() {
        @Override
        public void handleMessage(Message msg) {

//          SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MMM-yy hh.mm.ss
aa");
//          Date date = new Date();
//          String formattedDate = dateFormat.format(date.toString());

            switch (msg.what) {
                case TGDevice.MSG_STATE_CHANGE:

                    switch (msg.arg1) {
                        case TGDevice.STATE_IDLE:
```

```
            break;
          case TGDevice.STATE_CONNECTING:
            tv.append("Connecting...\n");
            break;
          case TGDevice.STATE_CONNECTED:
            tv.append("Connected.\n");
            ThinkGearConnector.getTGDevice().start();
            break;
          case TGDevice.STATE_NOT_FOUND:
            tv.append("Could not connect any of the paired BT devices.\n");
            break;
          case TGDevice.STATE_NOT_PAIRED:
            tv.append("No Bluetooth devices paired.\n");
            break;
          case TGDevice.STATE_DISCONNECTED:
            tv.append("Disconnected.\n");
        } /* end switch on msg.arg1 */

        break;

    case TGDevice.MSG_POOR_SIGNAL:
        tv.append("PoorSignal: " + msg.arg1 + "\n");
        break;

    case TGDevice.MSG_HEART_RATE:
        tv.append("Heart rate: " + msg.arg1 + "\n");
        break;

    case TGDevice.MSG_RAW_DATA:
        int rawValue = msg.arg1;
        tv.append("Raw Data: " + rawValue + "\n");
        break;

    case TGDevice.MSG_MEDITATION:
        tv.append("Meditation: " + msg.arg1 + "\n");
        break;

    case TGDevice.MSG_BLINK:
        tv.append("Blink: " + msg.arg1 + "\n");
        break;
    case TGDevice.MSG_RAW_COUNT:
        //tv.append("Raw Count: " + msg.arg1 + "\n");
        break;

    case TGDevice.MSG_LOW_BATTERY:
```

```
                  Toast.makeText(getActivity().getApplicationContext(), "Low battery!",
Toast.LENGTH_SHORT).show();
                  break;

              case TGDevice.MSG_EEG_POWER:
                  TGEegPower ep = (TGEegPower) msg.obj;
                  tv.append("\n"+
                          "Delta      : " + ep.delta + "\n" +
                          "HighAlpha  : " + ep.highAlpha + "\n" +
                          "HighBeta   : " + ep.highBeta +   "\n" +
                          "LowAlpha   : " + ep.lowAlpha +   "\n" +
                          "LowBeta    : " + ep.lowBeta +     "\n" +
                          "LowGamma   : " + ep.lowGamma +   "\n" +
                          "MidGamma   : " + ep.midGamma +   "\n" +
                          "Theta      : " + ep.theta + "\n");

                  Intent dataIntent = new Intent();

dataIntent.setAction("mobie.focusgain.focusgainmobile.EEG_POWER_BANDS");

                 dataIntent.putExtra("formattedDate", formattedDate);
                 dataIntent.putExtra("hours", date.getHours());
                 dataIntent.putExtra("minutes", date.getMinutes());
                 dataIntent.putExtra("seconds", date.getSeconds());

                  dataIntent.putExtra("Delta", Integer.toString(ep.delta));
                  dataIntent.putExtra("HighAlpha", Integer.toString(ep.highAlpha));
                  dataIntent.putExtra("HighBeta", Integer.toString(ep.highBeta));
                  dataIntent.putExtra("LowAlpha", Integer.toString(ep.lowAlpha));
                  dataIntent.putExtra("LowBeta", Integer.toString(ep.lowBeta));
                  dataIntent.putExtra("LowGamma", Integer.toString(ep.lowGamma));
                  dataIntent.putExtra("MidGamma", Integer.toString(ep.midGamma));
                  dataIntent.putExtra("Theta", Integer.toString(ep.theta));
                  getActivity().sendBroadcast(dataIntent);

                  boolean attentive = new
ANNClassifier().recognizeAttention(getApplicationContext(),ep);
                 Vibrator vibrator = (Vibrator)
getSystemService(Context.VIBRATOR_SERVICE);

                  if(!attentive )
                  {
                     ++focusLostCount;
                  }
                  if( focusLostCount == 20)
                  {
                    tv.setBackgroundColor(Color.RED);

                    // Vibrate for 500 milliseconds
                    vibrator.vibrate(2000);
                }
                if( focusLostCount == 22)
                {
                    tv.setBackgroundColor(Color.WHITE);
                    vibrator.cancel();
                    focusLostCount = 0;
                }

            default:
                break;

        } /* end switch on msg.what */

    } /* end handleMessage() */

  }; /* end Handler */

  @Override
  public void onClick(View v) {

      doStuff(v);
  }
}
```

ANNClassifier.java

```
package mobile.focusgain.focusgainmobile.ann;

import android.app.Activity;
import android.content.Context;

import com.neurosky.thinkgear.TGEegPower;

import org.encog.neural.data.NeuralData;
import org.encog.neural.data.basic.BasicNeuralData;
import org.encog.neural.networks.BasicNetwork;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
```

```
/**
 * Created by Charitha Senarathne on 4/28/2015.
 */
public class ANNClassifier {

    private BasicNetwork trainedANN;

    public boolean recognizeAttention( Context context, TGEegPower powerBands) {

        boolean attentiveStatus = false;
        if (getTrainedNetwork() != null) {
            loadTrianedANN(context);
        }

        double[] INPUT = new double[8];
        INPUT[0] = powerBands.delta;
        INPUT[1] = powerBands.theta;
        INPUT[2] = powerBands.lowAlpha;
        INPUT[3] = powerBands.highAlpha;
        INPUT[4] = powerBands.lowBeta;
        INPUT[5] = powerBands.highBeta;
        INPUT[6] = powerBands.lowGamma;
        INPUT[7] = powerBands.midGamma;


        NeuralData output = computeOutput(INPUT);

        double[] allData = output.getData();
        double outputReading = (double) Math.round(allData[0]);

        if (outputReading > 0.5) {
            attentiveStatus = true;
        } else {
            attentiveStatus = false;
        }

        return attentiveStatus;
    }

    public void loadTrianedANN(Context context) {

        setTrainedNetwork(new BasicNetwork());

        try {
            FileInputStream fin = (FileInputStream)
context.getAssets().open("attention_classification_network.dat");

            ObjectInputStream ois = new ObjectInputStream(fin);
            setTrainedNetwork((BasicNetwork) ois.readObject());
            ois.close();

        } catch (IOException | ClassNotFoundException e) {

        }
    }

    public NeuralData computeOutput(double[] INPUT) {
        NeuralData output = getTrainedNetwork().compute(new BasicNeuralData(INPUT));

        return output;
    }


    public BasicNetwork getTrainedNetwork() {
        return trainedANN;
    }


    public void setTrainedNetwork(BasicNetwork trainedANN) {
        this.trainedANN = trainedANN;
    }
}
```