Classification Brainwave Signals based on Deep Learning for Controlling Electronic Devices

A Thesis

Submitted to the Council of

the College of Science at the University of

Sulaimani in Partial Fulfillment of the Requirements

for the Degree of Master of Science

in Computer Science

by

Zhyar Rzgar K. Rostam

B.Sc in Computer Science / 2013

Supervised by

Dr. Sozan Abdullah Mahmood

Assistant Professor

September, 2019 (English) Gelawêj, 2719 (Kurdish)

Supervisor Certification

I certify that the preparation of thesis titled "Classification Brainwave Signals based on Deep Learning for Controlling Electronic Devices" accomplished by Zhyar Rzgar K. Rostam, was prepared under my supervision in the College of Science, at the University of Sulaimani, as partial fulfilment of the requirements for the degree of Master of Science in Computer Science.

Signature:

Name: Dr. Sozan Abdullah Mahmmod

Title: Assistant Professor

Date: 30/July/2019

In view of the available recommendation, I forward this thesis for debate by the examining committee.

Signature:

Name: Mustafa Ibrahim Khaleel

Position: Head of the Department

Title: Lecturer

Date: 30/July/2019

Linguistic Evaluation Certification

I hereby certify that this thesis titled “Classification Brainwave Signals based on Deep Learning for Controlling Electronic Devices” prepared by Zhyar Rzgar K. Rostam, has been read and checked and after indicating all the grammatical and spelling mistakes; the thesis was given again to the candidate to make the adequate corrections. After the second reading, I found that the candidate corrected the indicated mistakes. Therefore, I certify that this thesis is free from mistakes.

Signature:

Name: Kazi Hassan Salih

Position: English Department

College of Language and Education

Charmo University

Date: 25/July/2019

Examining Committee Certification

We certify that we have read this thesis entitled "Classification Brainwave Signals based on Deep Learning for Controlling Electronic Devices" prepared by Zhyar Rzgar K. Rostam, and as Examining Committee, examined the student in its content and in what is connected with it, and in our opinion it meets the basic requirements toward the degree of Masters of Science in Computer Science.

Signature:
Name: Dr.Ahmed M. Khorsheed
Title: Professor
Date: /September/2019
(Chairman)

Signature:
Name: Dr.Mahmud A. Mohammad
Title: Lecturer
Date: /September/2019
(Member)

Signature:
Name: Dr.Aysar A. Abdulrahman
Title: Lecturer
Date: /September/2019
(Member)

Signature:
Name: Dr.Sozan A. Mahmood
Title: Assistant Professor
Date: /September/2019
(Supervisor - Member)

Approved by the Dean of the College of Science.

Signature:

Name: Dr. Soran Mohammed Mamand
Position: Dean of the College
Title: Assistant Professor
Date: /September/2019

Declaration

I declare this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree or professional qualifications.

Signature:

Name: Zhyar Rzgar K. Rostam

Date: 30/July/2019

Dedication

I would like to dedicate this work to

- My lovely **parents**, for always being their for me and having trust in me.
- My beloved wife, **Kaziwa** without her love and support this work would not have been made possible.
- Department of **Computer Scienc**e
- My **family** and **friends**.

and to

Who taught me ”Hello World”

Acknowledgements

First of all, my ultimate gratitude goes to Allah, lord of the whole creations. Who sustained me throughout the duration of this study.

I would like to express my sincere gratitude to my supervisor **Dr. Sozan A. Mahmood** for her inspiration and continuous encouragement throughout this work.

I am very much thankful to the Dean of College **Dr. Soran M. Mamand**, Vice Dean **Dr. Diary A. Mohammed**, Head of the Department **Dr. Mustafa I. Khaleel**, and **Miss. Mihran A. Muhammed** Coordinator of the Department they helped me with all the difficulties I faced.

I would like to extend my utmost gratitude towards my **parents** who supported and constantly encouraged me during my study.

From the deepest of my heart, my appreciation and gratitude goes to my beloved wife - **Kaziwa**, for her advice, patience, and continuous encouragement. She always understood me.

My special thank goes to **Mr. Zhiwar F. Hassan**, **Dr. Shram S. Hassan**, **Mrs. Kaziwa H. Saleh**, **Mr. Delman A. Saleh**, **Mr. Kardo O. Aziz**, and **Mr. Tofiq A. Tofiq** for their participation in creating the dataset. Furthermore, I would like to thank **Assist Prof. Firooz H. Aziz**, **Dr. Kazi H. Salih**, and **Dr. Shanyar Hawrami**.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion this work.

Zhyar Rzgar K. Rostam

October 16, 2019

Abstract

Brainwave signals are read through Electroencephalogram (EEG) devices. These signals are generated from an active brain based on brain activities and thoughts. The classification of brainwave signals is a challenging task due to its non-stationary nature. To address the issue, this work propose two models inspired by biological processes to classify brainwave signals. The models are designed based on a deep learning algorithm and a meta-heuristic algorithm. Convolutional Neural Network (CNN) has been selected as a deep learning algorithm, and Age-Layered Population Structure - Genetic Algorithm(ALPS-GA) as an evolutionary algorithm. Generally, CNN is composed of three main layers: convolution, pooling, and fully connected layer. One of the models was designed by combining CNN with two different Discrete Wavelet Transform (DWT) Coiflet 1, and Symlet 2, individually. Such that, the pooling layer has been replaced by either Coiflet 1 or Symlet 2. Then, the extracted features have been fed to the fully connected layer, for training in this layer Stochastic Gradient Descent (SGD) has been used. The other model, also designed based on CNN, two basic modification has been applied. First, the pooling layer has been replaced with DWT Coiflet 1. Second, the fully connected layer in the third layer has been removed, the ALPS-GA with Symbolic Discriminant Analysis (SDA) have been used instead of it to classify and predict the correct classification.

In order to evaluate the classification performance of the proposed models a dataset is developed by recording brainwave signals for two conditions, which are visible and invisible. In the visible mode, the human subjects focus on the color and shape presented. Meanwhile, in the invisible mode, the subjects think about specific colors or shapes with closed eyes. The dataset has been built from six healthy subjects. Each subject has normal mental

state, normal color vision, and age ranging between 25 to 35 years old. According to the results achieved for all models, the best classification result achieved while applying 'combined CNN with Coiflet 1' and using SGD as a trainer in the fully connected layer. The best accuracy rate is 92% for both visible color, and visible arrow. However, the accuracy rate is 83% for invisible color, and invisible arrow. Eventually, the best model has been selected to classify brainwave signals and interpret the signals to set of commands. Then the commands were used to control a Lego NXT 2.0 robot.

Contents

List of Figures 8

List of Tables

List of Abbreviations

A/D	Analog-To-Digital	**DNI**	Direct Neural Interfaces
ALL	Acute Lymphoblastic Leukemia	**DNN**	Deep Neural Network
ALPS	Age-Layered Population Structure	**DWT**	Discrete Wavelet Transform
AML	Acute Myeloid Leukemia	**EA**	Evolutionary Algorithm
ANN	Artificial Neural Network	**ECoG**	Electrocorticography
AR	Auto Regression	**EEG**	Electroencephalogram
ASIC	Application Specific Integrated Circuit	**FIR**	Finite Impulse Response
BCI	Brain-Computer Interface	**fMRI**	Functional Magnetic Resonance
CAP	Credit Assignment Path	**FT**	Fourier Transform
COM	Communication	**F**	Frontal Lobe
CNN	Convolutional Neural Network	**GA**	Genetic Algorithm
CSP	Direct Neural Interfaces	**GLEM**	Graph Regulated Extreme Learning Machine
CSV	Comma Separated Values	**HCI**	Human-Computer Interaction
DBN	Deep Belief Network	**MRS**	Magnetic Resonance Imaging
DEAP	Database For Emotion Analysis Using Physiological Signals	**NLP**	Natural Language Processing

HMM	Hidden Markov Model	**PCA**	Principle Component Analysis
Hz	Hertz	**PET**	Positron Emission Tomography
ICA	Independent Component Analysis	**O**	Occipital Lobe
k-NN	K-Nearest Neighbor	**ReLU**	Rectified Linear Unit
LCD	Liquid-Crystal Display	**RJ**	Registered Jack
LDA	Linear Discriminant Analysis	**SDA**	Symbolic Discriminant Analysis
LeJOS	Lego Java Operating System	**SGD**	Stochastic Gradient Descent
MEG	Magnetoencephalography	**SVM**	Support Vector Machine
MI	Motor Imagery	**TGC**	Thinkgear Connector
MLP	Multi-Layer Perceptron	**T**	Temporal Lobe
MMI	Mind-Machine Interface	**TGMA**	Thinkgear ASIC Module
MRS	Magnetic Resonance Spectroscopy	**TUH**	Temple University Hospital
NBC	Next Byte Codes	**UCI**	University Of California Irvine

Chapter 1

Introduction

1.1 Introduction

Brain is one of the largest and most complex organs in the human body. It consists of three main parts: cerebrum, cerebellum, and brainstem. The largest part of the human brain is cerebrum accounting for 85% of the organ's weight, and it is composed of four lobes: frontal, parietal, temporal, and occipital. The frontal lobes are responsible for problem solving, judgement, and motor function. Body position, handwriting, and sensation are managed by the parietal lobes. The temporal lobes are involved with memory and hearing. Whereas visual processing, and objects recognition are handled by the occipital lobes [1] [2].

Human's brain continuously produces huge electrical activities, these signals are non-stationary, and they can be measured via different devices, such as: functional magnetic resonance imaging (fMRI), electroencephalogram (EEG), and magnetoencephalography (MEG). The brain signal has five bands alpha, beta, delta, gamma and theta. EEG is an easy and an inexpensive way for recording brainwave signals compared to other methods [3]. Nowadays, EEG is used in many scientific fields such as medical [4] [5], human computer interaction (HCI) [6], and entertainment.

EEG takes a major role in the field of medical applications. Abnormalities in brain activity can be determined by using EEG. It also might be used

for diagnosing brain disorders, especially epilepsy or other seizure disorder, and sleep disorders. Correct classification of seizures as epileptic and non-epileptic helps in the diagnosis of major illnesses such as epilepsy [7]. In many cases, early-stage diagnoses of brain disease is a key to make the treatment process effective. Furthermore, EEG might be used to confirm brain death in someone in a persistent coma [8] and also diagnosis of tumor and stroke.

The communication pathway between devices and human's brain is held through Brain Computer Interface(BCI) system, Mind-Machine Interface (MMI) or sometimes known as a Direct Neural Interface (DNI) systems. BCI system is made up of hardware and software, it provides an interface that enables the human's brain communicate with the outside by interpreting the EEG signals to interact with devices (such as robots and wheelchairs). The most common provided form of BCI system is for assisting paralyzed people or patients suffering from severe motor impairments who cannot express their thoughts as healthy people [6], for basic BCI block diagram see Figure 1.1.

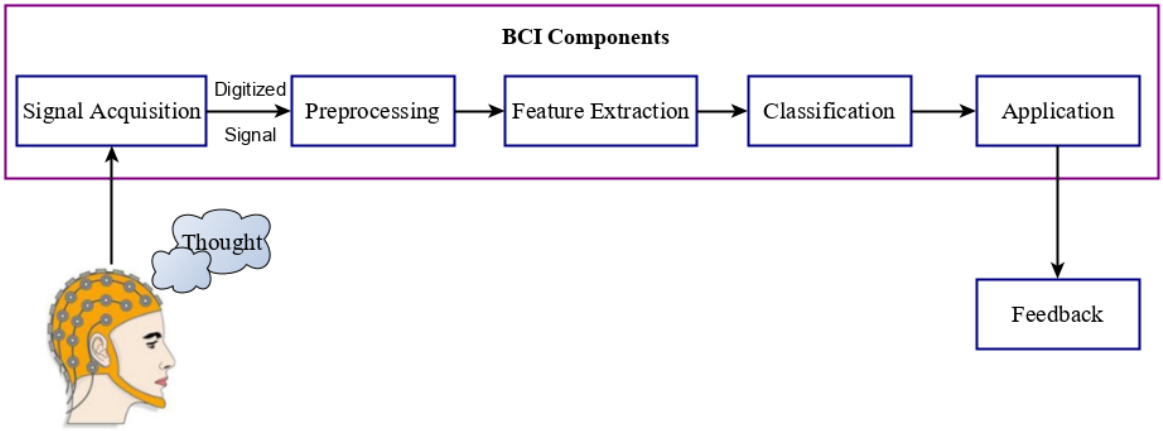


Figure 1.1: BCI block diagram

BCI is classified as invasive, semi-invasive and non-invasive techniques. Neurosurgery is required for implanting the electrode arrays within the brain in both invasive and semi-invasive techniques. Signals in those two tech-

niques mentioned previously have high resolution with a wider frequency range. However, the most common method is non-invasive, because the electrodes need to be placed directly on the scalp without any neurosurgery, whereas the signal quality in non-invasive is weak and vulnerable to noise compared to semi-invasive and invasive techniques [9].

Previously, EEG sensors were expensive, due to this limitation these sensors where only used in special circumstances available in laboratories and hospitals. There are two kinds of EEG electrodes which are: gel-based and dry electrodes [10].

In recent years EEG equipment are more available in the public market. This means that EEG device will not only remain in the medical or BCI fields with specific applications. Nowadays, it is applied for training, education and entertainment purposes.

Most BCI systems consist of sequential steps which are signal acquisition, pre-processing, feature extraction and classification. In the signal acquisition, brainwave signals are recorded via the EEG device. In the pre-processing step, different transformations by using filters and normalization process are applied to the raw EEG signal. In the feature extraction step, the most interesting (point) features are extracted from the input data. Two of the most common method used for extracting features are Fourier Transform (FT), and Wavelet Transform (WT). The classification comes after feature extraction, there are several types of classifiers that can be used. The most common classifier methods with EEG signals are Support Vector Machine (SVM) [11], Linear Discriminant Analysis (LDA) [12], and Neural Network (NN) [3].

Researchers mainly focus on pre-processing and feature extraction techniques for classifying EEG signals to achieve successful classification. Classification brainwave signals is a complex task, due to nonlinear nature. It

requires advanced signal processing methods for analyzing with powerful machine learning algorithms to achieve a certain classification.

Deep learning and evolutionary algorithms (EA) are among successful techniques that are used in image processing and computer games [13], natural language processing (NLP) [14],and mental state detection [15].

Deep learning can be classified as a subfield of machine learning based on structure and function inspired by human brain called artificial neural networks. It consists of multiple layers that progressively extract higher level features from raw of data input. Deep learning can perform classification tasks directly from data, it can achieve the highest accuracy, occasionally exceeding human level-performance. Deep learning models requires large labeled data and neural network architectures that contain many layers [16]. When deep learning comes into sight, researchers use its methods as an alternative of conventional classification methods for classification of EEG signals.

In this study two improved models have been proposed for classifying brainwave signals based on Convolutional Neural Network (CNN). Both models are tested for color identification and shape identification. In order to achieve this, a dataset is developed by recording EEG signals sourced from six participants. In this work the emphasis was on obtaining best classification with higher possible accuracy rate in an efficient way. The best proposed model has been selected to classify signals. Then interpret the classified signals to set of commands for controlling a robot.

1.2 Problem Statement

Using single electrode for performing accurate classification, interpreting, and understanding brainwave signals to computer based data is a challenging

task. The brainwaves change in a timely manner based on our brain activities and thoughts. It is difficult try to control our mind on how we think. Thus, analyzing brainwave signals and interpreting them to a form that is understandable by computer with accurate classification requires advanced techniques and processes. Hence, designing a BCI system for interpreting brainwaves to command, and using these commands for controlling an electronic device require a powerful model to achieve the best classification result.

This thesis focuses on proposing two models for classifying brainwave signals accurately in an efficient way. It then provides a comparison between both proposed models with the inspired (original) models individually. Furthermore, it presents the best achieved results in both models. Finally, it selects the the best classifier model as a solution for the mentioned problems.

1.3 Literature Survey

Yıldırım *et al* [3]. Have proposed a model which is an end-to-end complete structure which doesn't need any preprocessing or feature extraction. The model is used to identify abnormal EEG signals based on deep CNN; the researchers used the TUH EEG abnormal corpus (v2.0.0) dataset which was provided by Temple University Hospital (TUH). The dataset was prepared by collecting EEG signals from temporal occipital (T5-O1) lobes. They have proposed a new one-dimensional convolutional neural (1D-CNN) network model; the model has 23 layers including the input layer. It involved 1D Convolution, MaxPooling, dropout, batch normalization, and fully connected (dense) layers. The convolution process, which comes after input layer, has been carried out between the input data and 8 filters with 3 strides (number of data point window is shifted in single operation), and each has 23 items. The outputs that come after convolution are known as feature maps. The

MaxPooling layer is applied on the features map with size 2 (between each two unit the maximum value is taken), this reduces the size of the data by 2. Dropout technique is used to prevent the model from overfitting. After applying batch normalization, the data is fed to the dense layer. The classification process has been done in the final layer which is softmax layer. Eventually, the model has been able to detect the abnormal EEG signals with the rate of 79.34% and 79.64% accuracy and precision, respectively.

Rajendra *et al* [17]. Have presented a model that uses a deep neural network approach for EEG-based screening of depression. The EEG signals are collected from 15 depressed patients and 15 normal subjects whose age ranging from 20 to 50 years old, the data were recorded from the right and left hemisphere of the brain. The EEG signals were recorded in 5 minutes with eyes open and closed for each participant, the EEG data obtained from the left half (FP1-T3 channel pair) and the right half (FP2-T4 channel pair). Two thousands sampling points are in each record. To overcome the amplitude scaling problem Z-score normalization is used. The model was built using a deep CNN. The proposed CNN architecture consists of 5 convolutional layers, 5 pooling layers, and 3 dense layer. The convolution filters (kernels) and pooling layers are set to 5 and 2, respectively. Stride is fixed to 1 for convolution and 2 for pooling process. The model attained accuracy of 93.5% for right and 96.0% for left hemisphere.

Tang *et al* [11]. Have developed a method that depends on deep CNN for single trail motor imagery EEG. To classify MI tasks (left hand and right hand movements) a model is designed that consists of five-layer CNN, then the experimental dataset is fed to CNN. The dataset was prepared by recording data from 28 active electrodes. The dataset was recorded from two healthy and normal subjects, they were all right handed. Subjects were seated on a

comfortable chair and were faced to a screen. The total trial for each subject was 460. The dataset is involved into random sequence of 230 trails of two class MI tasks. A comparison has been done between the results obtained from this model and other conventional classification methods after tested on the dataset such as power + SVM (Support Vector Machine), CSP (Common Spatial Pattern) + SVM and AR (Auto Regression) + SVM. They conclude that the CNN can improve classification performance, such that the average accuracy reached using CNN 86.41 ± 0.77 is 9.24%, 3.80%, and 5.26% greater than those using power + SVM which is 77.17 ± 7.69%, CSP + SVM which is 82.61 ± 6.15%, and AR+ SVM which is 81.25 ± 3.07%, respectively.

Rajendra *et al* [18]. Have presented a model for detection and diagnosis of seizure using EEG signals based on deep CNN. A dataset used in the study is collected by Andrzejak *et al* [19]. The dataset was recorded from 5 patients and contains three classes of data: normal (Set B), preictal (Set D), and seizure (Set E). Each set contains 100 EEG signals, duration for each record in single channel EEG record is 23.6 seconds. The normal dataset was recorded from 5 healthy subjects, each containing 100 cases. The preictal class also contains 100 data from 5 epileptic patients, during recording the participants did not go through seizure during recording. As for the seizure class, which also contains 100 cases, recording of the data was performed on the same participants when they were having epilepsy. Before the data is fed to the 1D-CNN, the signal is normalized, then normalized data is fed to the 13-layer deep CNN to detect normal, preictal, and seizure classes. The first 10 layers consist of pairs of convolution and Max-pooling layer with different sizes, consecutively. The final 3 layers are dense layers. The accuracy achieved with the proposed model is 88.67%.

Rahmad *et al* [12]. Proposed a model for classification of brainwave signals using Genetic Algorithm (GA) for right and left motion patterns based on Logistic Regression, Linear Discriminant Analysis, k-Neighbors, Decision Tree, Naïve Bayes Gaussian, and SVM which combined five signal bands with GA to get one signal. A dataset was recorded from 10 subjects, where each subject was recorded 10 times, 5 times for each motion (left and right movement). One minute of brainwave was collected for each subject where subject controls right or left of the remote control cars. Only 15 seconds of data in the middle of the recorded data was taken. To adjust the signal magnitude, all signals were normalized and mapped between 0 and 1. The best result achieved for accuracy is 56%, SVM is a method that can achieve the best results compared to other methods.

Shon *et al* [15]. Have presented a model to classify EEG signals to tell whether signals represent a calm state or a stress state. GA and Principle Component Analysis (PCA) based feature selection and K-nearest neighbor (k-NN) classifier have been used to classify stress in human beings; the features that come from GA or (PCA) are used as input to the k-NN to classify the EEG signals. Thirty two channels are used to extract the features which contain: statistical features, Hjorth parameters, band power, and frontal alpha asymmetry. The best accuracy was achieved for GA is 71.76%. However, the accuracy achieved with PCA is 65.3%. The Database for Emotion Analysis using Physiological Signals (DEAP) was used which is a public EEG dataset, the dataset was recorded from 32 healthy participants, when 40 different music videos with 1 minute duration presented for each participant. The researchers used the DEAP dataset for evaluating the performance of the proposed method.

Uyulan *et al* [20]. Have presented a comparison between wavelet families for classification of mental tasks; the wavelet methods are combined with neural network for classification purposes. The aim of the study was to discover a convenient method between the wavelet families such as: haar, coiflets 1, biorthogonal 6.8, daubachies (2-4) and reverse biorthogonal 6.8 for feature extraction, then the extracted feature were fed to ANN for classification. A dataset has been prepared by recording data from a healthy subject who was a 28 year old male. The data were recorded during performing four mental tasks, which were: reciting the alphabet backwards, imagination of rotation of a cube, imaging of the left arm movement, and mentally performing mathematical operation. Recording of EEG signals has been done by using Emotive-Headset device, which has 14 electrodes, the headset sample for each channel is 128 samples/second. To remove artifacts 6th order of Butterworth band pass filter was used between 0.5 and 45 Hz. Depending on accuracy rate of classification result, the best result was achieved from coiflet of order 1.

Siradjuddin *et al* [21]. Have proposed a model based on GA for detecting alcoholic using EEG signals. UCI (University of California Irvine) machine learning repository has been used as a dataset. There are 64 channels in the dataset, because the dataset was recorded from 64 electrodes with sample rate 256 per second. The dataset contains EEG data for 10 alcoholic and 10 normal people. The research involved four steps. First, Independent Component Analysis (ICA) was used for signal separation and noise removal. Second, the Discrete Wavelet Transform (DWT) was used as a feature extractor, which divide signals into low frequency (approximation) and high frequency (detail). Third, the extracted features from DWT were fed to the GA, GA was used as a feature selector, this stage increased the accuracy of detection,

by choosing the best combination of features for the alcoholic detection. Finally, the selected features were fed to the backpropagation neural network. The average accuracy of alcoholic detection achieved was 79.38%.

Zheng *et al* [22]. Show a reliable model for classifying two emotional categories (positive and negative) based on advanced deep learning method. A dataset was created from recording brainwave signals from 6 subjects, two trials for each participant at interval of one week or longer. Sixty-two (62) EEG channels were used for testing algorithm and predicting the emotional state while the subject was emotionally watching movie clips. During the training, a deep belief network (DBN) was used, features were extracted from multichannel EEG. To do classification accurately a hidden markov model (HMM) was integrated with DBN to capture switching emotional stages. The researchers of the study provided a performance comparison between deep models of k-NN, SVM, and Graph regulated Extreme Learning Machine (GLEM). Accuracies were achieved of DBN-HMM was 87.62%, DBN was 86.91%, GELM was 85.67%, SVM was 84.08%, and k-NN was 69.66%. Results showed that the best accuracy was achieved when the dataset was tested with DBN-HMM model.

Table 1.1 presents the datasets details used by pre-mentioned studies.

Table 1.1: datasets details

Author(s)	**Application**	**Source**	**Participants**	**Accuracy**
Yıldırım *et al* [3]	normal and abnormal EEG signals	T5 - O1	2130	79.34%
Rajendra *et al* [17]	depression	Fp1, T3, Fp2, T4	30	93.5% - 96%
Tang *et al [11]*	motor imagery task	28 electrodes active	2	82.61 ± 6.15%
Rajendra *et al* [18]	seizure	Fp1 - T3	15	88.67%
Rahmad *et al* [12]	right and left motion pattern	Fp1	10	56%
Shon *et al* [15]	calm and stress state	48 electrodes	32	71.76%
Uyulan *et al* [20]	mental task	14 electrodes	1	60.3% - 99.6%
Siradjuddin *et al* [21]	detect alcoholic	64 electrodes	20	79.38%
Zheng *et al* [22]	emotional category	62 electrodes	6	87.62%

1.4 The Aims of the Thesis

This study aims to propose two architectures based on CNN for classifying brainwave signals accurately in an efficient way. It also aims to record brainwave signals and build a dataset, from various subjects in different sessions and conditions. The dataset is used to evaluate the accuracy of both proposed models and original CNN, and provide a model for classifying brainwave signals accurately and efficiently. The study eventually, selects the model that achieves the best result for controlling an electronic device based on classified signals.

1.5 Thesis Layout

The rest of thesis is organized as follows:

Chapter Two:

This chapter provides a detailed background the work of EEG device, and describes raw EEG signals behavior with a detailed description of each band. It also presents all theoretical backgrounds such as the concepts of pre-processing techniques, feature extraction, deep learning, and genetic algorithm. In addition it describes the Lego NXT Mindstorms robot with all its hardware details.

Chapter Three:

This chapter covers the overall process of collecting brainwave signals for building a dataset. Then, it explains the detail of CNN architecture, and both proposed models architecture inspired from CNN through the algorithms, flow charts, and block data diagrams.

Chapter Four:

This chapter presents the results and shows the impact of modifications ap-

plied on the CNN for both proposed models, and the parameters that have a direct impact on the accuracy classification. It then provides comparisons between the CNN and both proposed models individually in different circumstances, and between both proposed architectures together.

Chapter Five:

This chapter summarizes and discusses the work of the thesis. It also suggests ideas for future work, and steps to be taken to improve the problems left open by this thesis.

Chapter 2

Background Theory

2.1 Introduction

This chapter covers the aspects of theoretical backgrounds and robot design in terms of hardware specifications used in the context of thesis. First provides an explanation of Brain Computer Interface (BCI) system and its types, and the EEG technology. Then, it describes the different patterns that from the foundation for EEG analysis. After that, the chapter also explains fundamental steps required for classifying brainwave signals. In the fourth part the evolutionary algorithm and deep learning are explained. Finally, a complete description on the Lego NXT robot is provided.

2.2 Brain Computer Interface (BCI) System

BCI is a communication pathway between devices and the human's brain, sometimes known as Mind-Machine Interface (MMI), or Direct Neural Interface (DNI) systems. BCIs are often directed in helping, augmenting or repairing human cognitive or sensory motor functions [23]. Human's brain contains a huge amount of neurons, these neurons are classified into three groups based on their functions which are: sensory neurons, motor neurons, and neural circuit [23] [24]. Each nerve cell is connected to another nerve cell by dendrites and axons. The neurons produce small electrical signals based on every action like: thinking, moving, feeling or remembering something.

The simplest action called electric potential (or action potential), and is a discharge caused by fast opening and closing Na^+ and K^+ ions channels in the neuron membrane. If the membrane depolarize to some threshold, the neuron will *"fire"*. Tracking these discharges over time reveals the brain activity. These signals can be measured and interpreted to what they mean [25]. Broadly, BCI systems are classified into invasive, and non-invasive techniques.

2.2.1 Invasive BCI

Within invasive BCI approach the electrode arrays are buried on the human head through neurosurgery. The invasive technique provides high quality signals of BCI device because they lie in the grey matter (contains most of the brain's neuronal cell bodies) of the brain. However, a permanent hole remains in the skull. Invasive BCI is costly and requires dangerous surgery. Electrocorticography (ECoG) is one of the invasive BCI systems, sometimes known as partially invasive, because when the device is implanted inside the skull the rest remains outside the skull [26] [27].

2.2.2 Non-Invasive BCI

Non-invasive BCI is a popular technique that is commonly used, the electrodes need to be placed outside the skull or on the scalp. Signals received from non-invasive method have low resolutions compared to invasive BCI due to distance of the device from the brain, and the difficulty of recording the inner workings of the brain [26] [27]. The most commonly used non-invasive BCI system is electroencephalogram (EEG).

Beside the electrical activity, neural activity also generates other kinds of signals such as magnetic and metabolic that could be used in a BCI. Magnetoen-

cephalography (MEG) is used for recording brain magnetic fields. Positron Emission Tomography (PET), functional Magnetic Resonance Imaging (fMRI), and optical imaging are used for observing brain metabolic activities which cause changes in the blood flow [27]

2.3 EEG Device

The first human EEG was recorded by a German scientist named Hans Berger about 90 years ago. It is the first non-invasive technique that was discovered for recording the electrical activities of the brain. Signals quality of EEG are often (susceptible) vulnerable to noise such as: electrical interference or movement of electrodes [25] [27] [28]. Using a large number of EEG channels may impact on the signals quality to be noisy or redundant, and also directly affects the convenience in the use of the BCI. Therefore, choosing the minimum number of channels that achieve the best or required accuracy can balance both needs for performance and convenience [27] [29].

There are many advantages of using EEG techniques over some of other techniques including: fMRI, PET, MEG, and ECoG [27], such as:

1. From the hardware perspective, most of other techniques are costly compared to the EEG. The EEG hardware cost is cheap.

2. EEG sensors are available and used in more places than fMRI, PET, or MEG, as these techniques require immobile and special equipment.

3. EEG provides very high temporal resolutions, on the order of milliseconds rather than seconds. In clinical and research settings sampling rates between 250 and 2000 Hz are commonly used for recording. However, modern EEG data collection systems use sampling rates above 20,000 Hz for recording raw EEG.

4. EEG is a silence device, it is useful for better study of the responses to auditory stimuli.

5. EEG sensors are only placed on the scalp, the device does not require any surgery for placing electrodes on the surface of the brain such as in ECoG.

6. EEG does not involve exposure to high-intensity (>1 Tesla) magnetic fields, as in some other techniques, especially MRI and magnetic resonance spectroscopy (MRS). These can cause a variety of undesirable issues with the data, and also prohibit the use of these techniques with participants that have metal implants in their body, such as metal-containing pacemakers.

2.4 10-20 International System

Electrode locations are chosen arbitrarily from the scalp area corresponding to the motor cortical region to record the electrical activity of the brain, based on the BCI study. The 10-20 system or International 10-20 is a system that provides a standardized form for EEG electrode placement. The “10” and “20” refer to the fact that actual distance between two neighbor electrodes are either 10% or 20% of the total front-back or right-left distance of the skull. Figure 2.1 shows the location of electrode according to International 10-20 system. Each electrode placement site has a letter to identify the lobe and a number or another letter to identify the hemisphere location, or the area of the brain it is reading from. Pre-frontal (Fp), Frontal (F), Temporal (T), Parietal (P), Occipital (O), and Central (C) are regions used for electrodes placement. There is also (Z) site that refers to the electrode placed on the midline.
Even numbers 2, 4, 6, and 8 refer to the right hemisphere and odd numbers

1, 3, 5, and 7 refer to the left hemisphere [27].

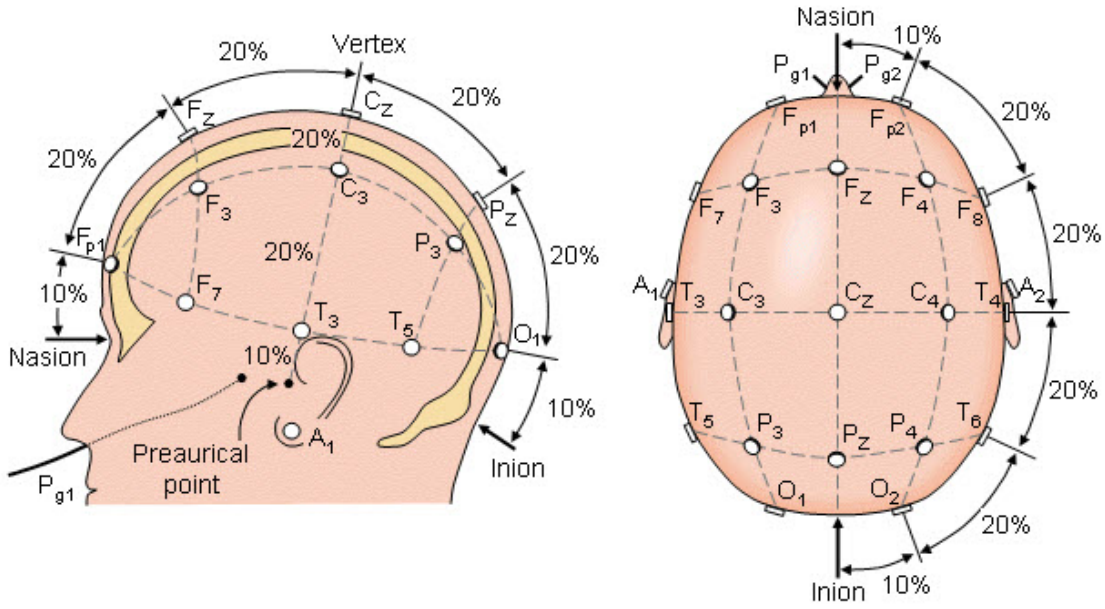


Figure 2.1: the international 10-20 systems of electrode placement

2.5 Data Acquisition

Nowadays, there are many commercial options available for EEG data acquisition in the form of EEG headset and head-cap. NeuroSky is an inexpensive and simple EEG headset, this device can safely measure and read brainwave activities, then transfer these signals via Bluetooth. NeuroSky Mindwave Mobile 2 provides good brainwave signals [30]. It is an upgraded version of mindwave mobile, which is placed on the head and reads the brain activities from the forehead. It is made up of three main parts: a flexible sensor, an ear pad (ear clip and loop), and a headband. Figure 2.2 shows the NeuroSky Mindwave mobile 2.

Every NeuroSky product has ThinkGear technology, through this device the wearer's brainwaves can be monitored and recorded. ThinkGear technology has three main parts: a sensor that touches the forehead, a reference points that is located on the ear pad, and an onboard chip that processes all of the data [31].

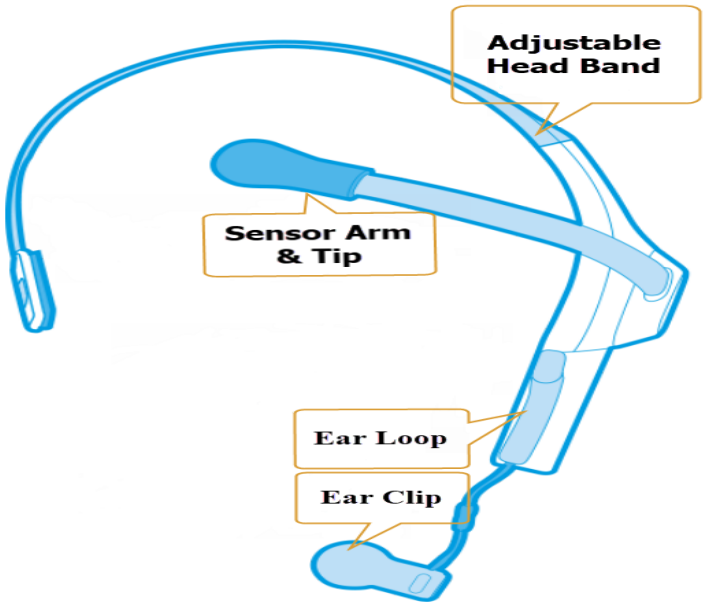


Figure 2.2: NeuroSky MindWave Mobile 2

Both the raw EEG data and the eSense Meters (a way to show how effectively a user is engaged in Attention and Meditation) are calculated on the ThinkGear chip.

NeuroSky technology provides a cost-efficient EEG linked research and products by using inexpensive dry sensors. However, the older EEG devices need to use a conductive gel between the sensors and the head. In addition, for noise reduction NeuroSky technology provides both built-in hardware/software, and embedded (chip level) solutions for signal processing and output [31].

The device includes built-in Bluetooth. Thus, through it, the device can be paired and communicate with other devices such as: computers, smartphones, and tablets. It safely measures and transmits the EEG powers spectrum (alpha, beta, theta, delta, and gamma) waves, attention and meditation meters, and also eye blinks detection. The entire signals can be captured through the ThinkGear Connector (TGC) which is a program that runs as background process on the computer, responsible for directing the mindwave headset data from the serial port to an open network socket [30] [32].

Figure 2.3 and Figure 2.4 show the location to place NeuroSky Mindwave on the head. The device has a single channel dry electrode which is on the flexible arm sensor, it touchs the forehead above the left eyebrow (Fp1 position) while the ear clip is used as a reference ground.

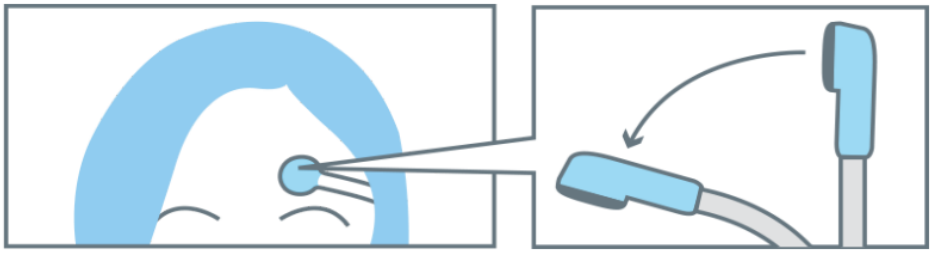


Figure 2.3: Fp1 position

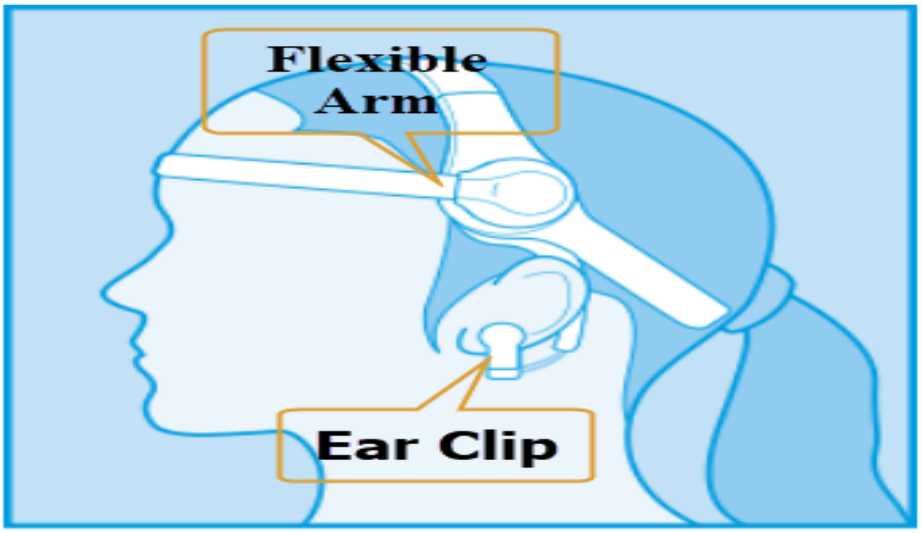


Figure 2.4: install NeuroSky on the head

2.6 Features of EEG Signals

Hans Berger who discovered EEG [33], found that different electrical frequencies could be linked to actions and different stages of conscious. This was done by observing subjects performing different tasks, like solving mathematical problems, while recording their EEG. There are five important types of brainwave bands. [25].

2.6.1 Gamma Wave (Γ)

Gamma brainwaves are in the frequency range of 31 Hz and up. This band reflects the mechanism of consciousness. Gamma waves are the fastest brainwave frequency with the smallest amplitude. Beta and gamma waves together have been associated with attention, perception, and cognition [25] [27] [34] [35]. Figure 2.5 shows the gamma wave pattern.

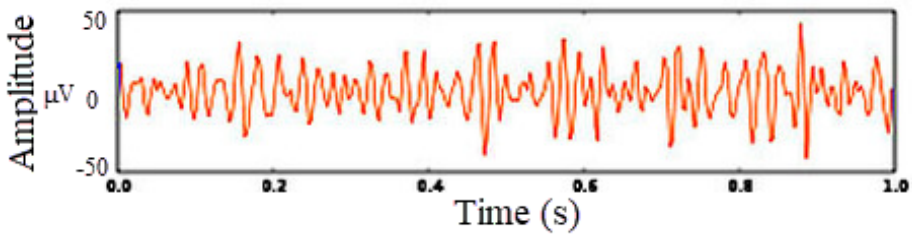


Figure 2.5: Gamma wave pattern

2.6.2 Beta Wave (β)

Beta brainwaves occur between 12 and 30 Hz, but during intense mental activity, they can reach 50 Hz. The wave is often divided into $\beta 1$ and $\beta 2$ to achieve more specific ranges. The waves are small and fast, generally attenuated during active movements like active thinking, active attention, or solving math tasks. The beta wave rhythm with dominant set of frequencies is associated with various pathologies and drug effects, especially benzodiazepines, also it is prominent rhythm in patients who are alert or anxious or who have their eyes open [25] [27] [34] [35]. Figure 2.6 presents the beta wave pattern.

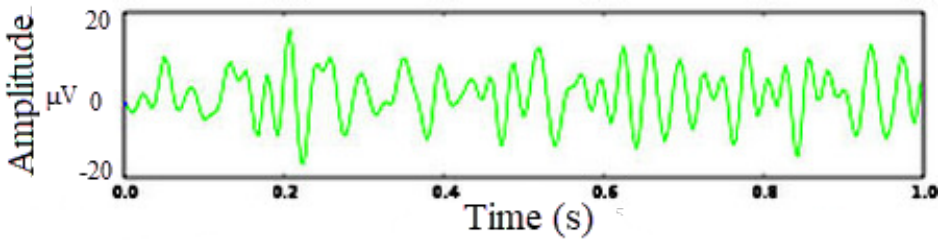


Figure 2.6: Beta wave pattern

2.6.3 Alpha Wave (α)

Alpha brainwaves occur between 7.5 to 12 Hz, they are slower and associated with relaxation and disengagement. In general, increase in the alpha activity occurs when an adult who is awake but relaxed with eyes closed, or thinking of something peaceful with eyes closed. The brain generates alpha waves when trying to relax and clearing minds of wandering thoughts or simply choosing to ignore them. Each region of the brain has a characteristic alpha rhythm but alpha waves with greatest amplitude can be recorded from both occipital and parietal regions in the cerebral cortex [25] [27] [34] [35]. See Figure 2.7 for alpha wave patterns.

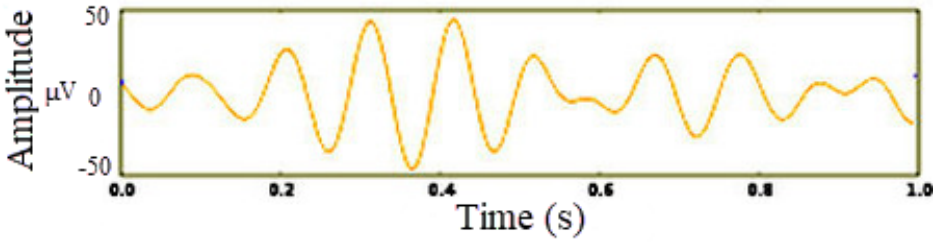


Figure 2.7: Alpha wave pattern

2.6.4 Theta Wave (θ)

Theta wave or theta rhythm is a neural oscillatory pattern that can be seen on an electroencephalogram (EEG). Theta wave ranging from 3.5 to 7.5 Hz, are linked to inefficiency, theta arises from emotional stress, especially frustration or disappointment. Theta wave is generated during daydreaming, fantasy, imagination, ideas, and inspiration thinking. It is also associated with access to unconscious materials such as deep meditation or driving for a long time while on a straight road [25] [27] [34] [35]. See Figure 2.8 for theta wave patterns.

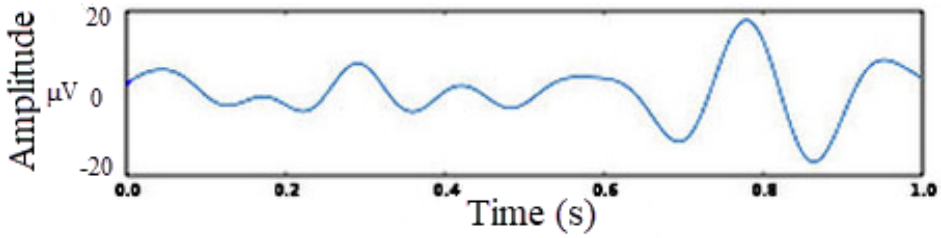


Figure 2.8: Theta wave pattern

2.6.5 Delta Wave (δ)

Delta brainwaves occur between 0.5 to 3.5 Hz. They are the slowest wave, the brain generates delta waves during sleeping when not dreaming. In another word, delta wave is associated with slow wave sleep (during stages 3 and 4 of the sleep stages). In general, these brainwaves are primarily associated with deep sleep, and awake state when thought to indicate physical defects in the brain [25] [27] [34] [35]. Figure 2.9 presents the delta wave patterns.

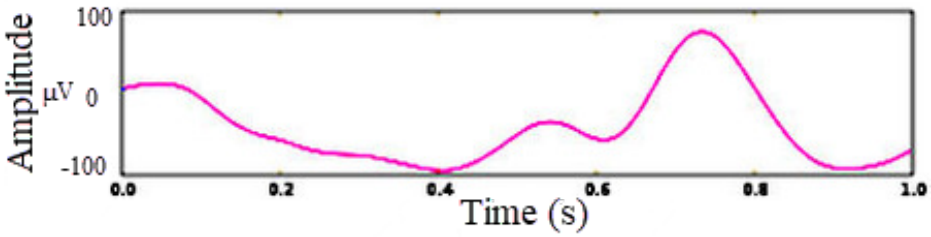


Figure 2.9: Delta wave pattern

2.6.6 Mu Wave (μ)

Mu wave occurs between 8 to 12 Hz. It can be found in the alpha wave frequency range. It is a spontaneous EEG wave that is associated with motor activities and maximum amplitude is recorded over motor cortex. Thus, it basically occurs when there is an actual movement or there is an intent to move the brain [25] [27]. See Figure 2.10 for MU wave patterns.

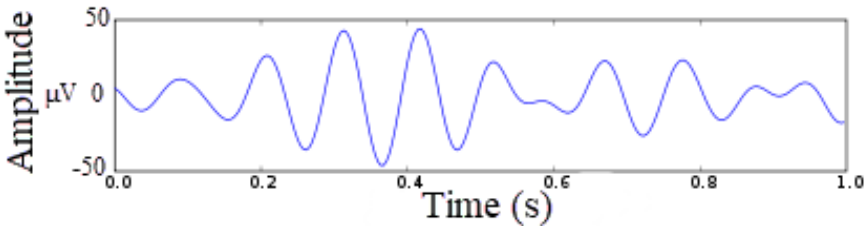


Figure 2.10: MU wave pattern

2.7 Cross-Entropy

Cross-entropy loss, or log loss, measures the performance or measures the error used in machine learning. Both use the same basics from the math perspective. Log loss is usually used when there are just two possible outcomes that can be either 0 or 1. Cross-entropy is usually used when there are three or more possible outcomes. Cross-entropy loss increases as the predicted probability diverges from the actual label [36] [37].

In binary classification, where the number of classes M equals 2, cross-entropy can be calculated according to equation (2.1).

$$L(t|y) = -\frac{1}{N}\sum_{i=1}^{N}[t.\log(y) + (1-t).log(1-y)] \tag{2.1}$$

The equation (2.1) is a cross-entropy for binary classification

where:

y - is the right label,

t - is a predict label,

N = is the number of samples.

if M > 2, calculate individual loss for each class label per observation and sum the result (multiclass classification), see equation (2.2).

$$H(p,q) = -\sum_{x} p(x).\log(q(x)) \tag{2.2}$$

The equation (2.2) is a cross-entropy for multiclass classification.

where:

p(x) - the wanted probability,

q(x) - the actual probability.

2.8 Deep Learning

Deep learning sometimes known as deep structured learning or hierarchal learning, is a shining branch of machine learning based on artificial neural networks. Learning can be supervised, semi-supervised or unsupervised [38]. Structure and function of artificial neural networks is inspired by the human brain. Supervised learning can be classified as a machine learning task of learning a function that maps an input and output based on pairs of input and outputs. The supervised algorithm learns from labeled data. While the learning process is done, the algorithm defines which label should be given to new data based on patterns and associating the patterns to the unlabeled new data [39] [40]. However, unsupervised learning is also classified as a class of machine learning technique refers to learning without a teacher. The given data to unsupervised algorithm are not labeled. Thus, the algorithms are left to themselves to find out the interesting structure in the data [16] [40].

While in semi-supervised learning uses small amount of labeled data with a large amount of unlabeled data. Thus, based on structure and function of semi-supervised learning, it places between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data) [41].

Using deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. Deep learning finds complicated structures in large data

sets by applying backpropagation algorithm to define how machine should change the internal parameters that are used to compute the representation in each layer from representation in the previous layer. It has had a great impact in the field of image processing, video processing, speech and audio processing. However, recurrent neural networks have an impact on sequential data such as text and speech. It is referred to multiple layers that progressively extract higher level features from raw data input [40].

Deep learning can perform classification tasks directly from data, it can achieve the highest accuracy, occasionally exceeding human level-performance. Deep learning models require large labeled data and neural network architectures that contain many layers [16] [38] [40].

Term "deep" in "deep learning" refers to the how many layers the data traveles through. Deep learning systems have a Credit Assignment Path (CAP) depth. CAP is a chain of possibly learn-able from input to outputs. It describes the potential causal connection between input and output. The depth of CAPs for feedforward neural network is the number of hidden layers plus one (output layer counted). However, the CAP depth potential is unlimited in recurrent neural networks, signals may propagate through a layer more than once [38] [40].

2.8.1 Convolutional Neural Network (CNN)

Feedforward neural network or multilayer perceptron with multiple hidden layers in artificial intelligence can be classified as deep neural network (DNN). Convolutional neural network (CNN) or ConvNet is a type of feedforward neural network, inspired by biological processes. It is considered as an efficient recognition algorithm widely used in image processing and pattern recognition. It has a simple structure and less training parameters. It

has a weight shared network structure, this phenomenon make it more similar to biological neural networks. Thus, it overcomes the complexity of the network model and the number of weights [42] [43].

CNN is most commonly applied within images [42], it is a powerful method for classification and identification of objects. The CNN is made up of two layers input and output layer, as well as multiple hidden layers. Except input and output layers, CNN architecture can be divided into three main layers: convolutional layer, pooling, and fully connected layers sometimes known as dense layers. CNN works with tensor (volume) which has width, height and depth.

Convolution and pooling layers work as filtering and feature extraction, and classification is done in the fully connected (dense) layer (see Figure 2.11) [44].

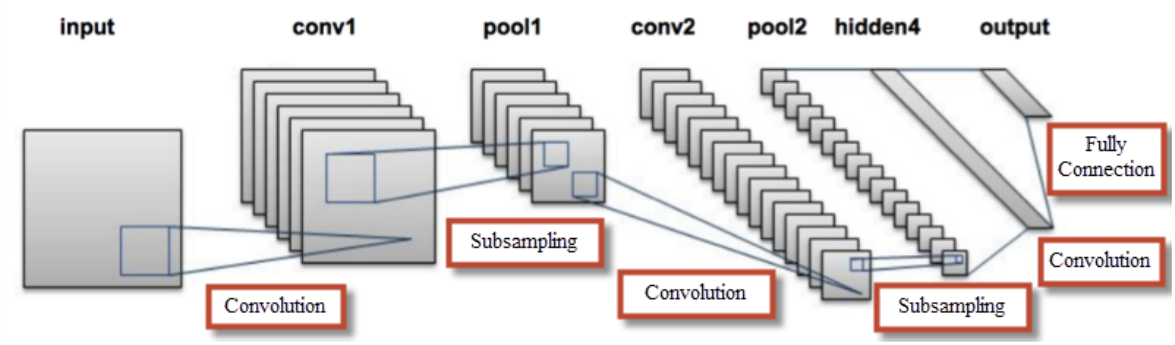


Figure 2.11: an overview of Convolutional Neural Network (CNN) architecture

2.8.2 Convolution

The first layer in CNN is convolution layer. The convolution process is done between the input data and kernels. Kernel, convolution matrix, mask, feature detector, or filter is a small matrix, which has small receptive field, used for blurring, sharpening, embossing, edge detection and more. It is the core of building block of CNN. Each convolutional layer within the CNN should have two attributes: input data which is a tensor (volume) structure,

and number of kernels [3] [42] [43] [44].

Kernel has two main features: width and height. Both features are hyper-parameters, and depth of kernel must be equal to the input data depth. Convolutional layer applies a convolution process in the input data, and defined kernels. The best way to understand how convolution layer works is to imagine that the kernels are flashlights. These flashlights shine over the input data with respect to the size of the kernels (i.e. light of the flashlight shines cover 3 x 3 area), and now the flashlight is sliding over all the areas of the input data. See Figure 2.12, and Figure 2.13 for convolution processes [3] [42] [44].

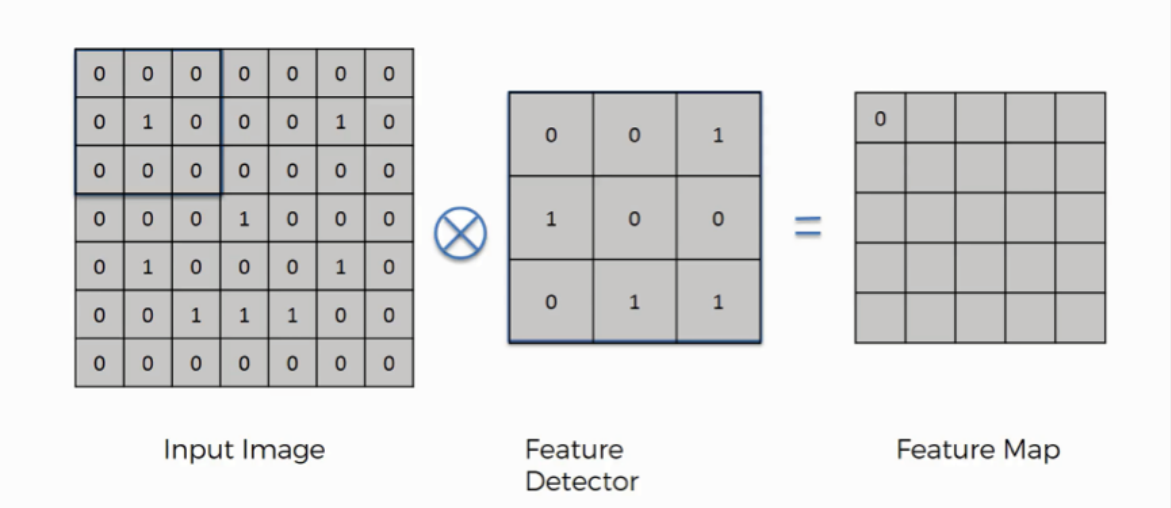


Figure 2.12: convolution process

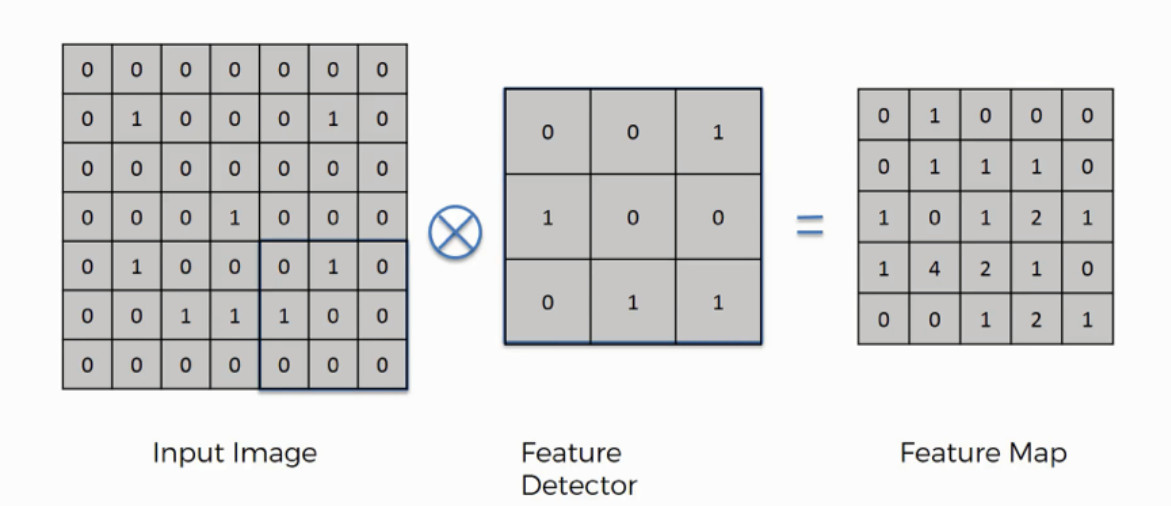


Figure 2.13: convolution process result

The Rectified Linear Unit (RecLU is a positive part of the arguments), shown in Figure 2.14, is an activation function performed inside the convolutional layer after the convolution process. It is applied on each convolved volume (feature map) and passes the results to the next layer. The aim of RecLU is to increase non-linearity, it removes all negative values from feature maps, and sets values to 0. See equation (2.3) for the rectifier function [44].

$$f(x) = x^+ = max(0, x) \tag{2.3}$$

The equation (2.3) is a rectifier linear unit.

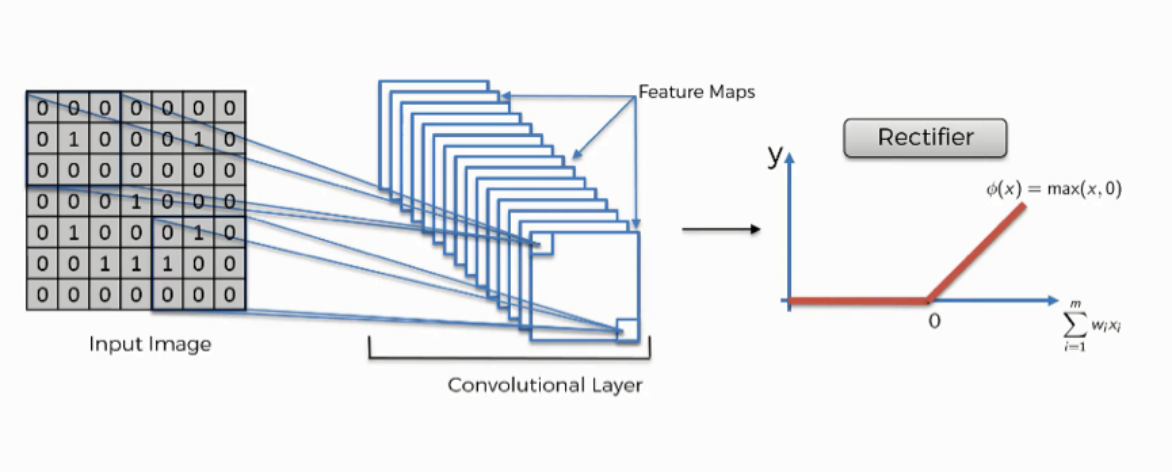


Figure 2.14: Rectified Linear Unit (RecLU)

2.8.3 Pooling

A pooling technique is used for feature extractions. Pooling is applied on the convolved data (feature map). There are different types of pooling. Convolution networks may include local (combines small clusters) or global (acts on all the neurons of the convolution) pooling layers. Max and average pooling are two commonly used techniques in this layer. Max pooling chooses the maximum value from an individual cluster of neurons at the previous layer. However, average pooling uses average value from each cluster of neurons at the previous layer. Hence, pooling layer reduces the dimensions of the data,

by combining the outputs of the neuron clusters at one layer into a single neuron in the next layer. Figure 2.15 presents max pooling [42] [43] [44].

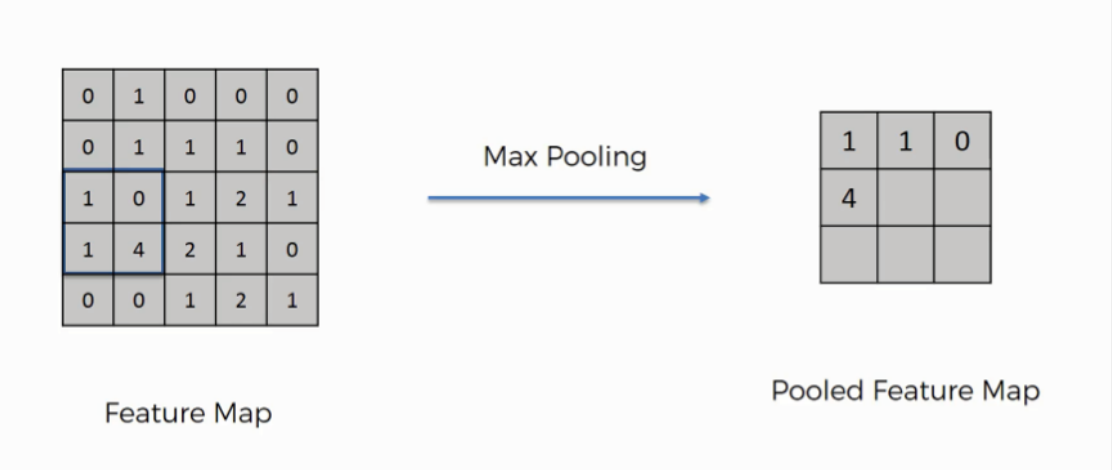


Figure 2.15: max pooling

2.8.4 Fully Connected

The extracted features from pooling layer are flattened, these features are fed to the fully connected layer to classify the features. It is in principle the same as the traditional multi-layer perceptron (MLP) neural network. Each neuron in fully connected layer has connections to all activations in the previous layer (every neuron in one layer has a connection to every neurons in another layer) [44]. Mathematically an artificial neuron can be represented by the sum of product between input values and its respective weight, see equation (2.4).

$$Z = \sum_{i=1}^{n} X_i.W_i + b_i \tag{2.4}$$

Where

X = is input neuron value,

W = is weight value,

b = is bias value.

2.8.5 Optimization

Optimization is a systematically developed process to reach the best solution under defined constraints and assumptions [45]. Also it is the most essential ingredient that helps the neural networks to minimize (or maximize) an error function (objective function).

Error function is a mathematical function based on the model's learnable parameters. These parameters help in computing the target values from a set of predictors used in the model. Within neural network, weights (W) and bias (b) values are considered as internal learnable parameters, which are used for computing the output values, and reaching the optimal solution by learning and updating in direction i.e. minimizing loss by the training process.

There are two kinds of optimization which are: first order and second order optimization [46] [47] [48].

First order optimization algorithm uses its gradient value with respect to parameters to minimize or maximize a loss function. The first order derivative is used to know that a function is increasing or decreasing at a particular point [47]. Gradient descent is the most commonly used first order optimization algorithm.

To properly update weight vector in first order optimization algorithms, the learning algorithm computes a gradient vector, for each weight, defined by what amount the error would be decrease or increase if weight increases by a tiny amount. The weight vector is updated in the opposite direction to the gradient vector [46].

However, second order optimization also known as Hessian (it is a matrix of second order partial derivatives), see equation (2.5) that used second order derivative to minimize or maximize the loss function. The second order derivative defines that whether the first derivative is increasing or decreas-

ing [47].

$$H_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j} \tag{2.5}$$

The equation (2.5) is a second order partial derivative.

Gradient Descent is also the most common optimization algorithm used in optimizing a neural network. It is used to update weights in a neural network model, and minimize the loss function by updating and tuning the model's parameters in a direction.

See equation (2.6) for gradient descent formula [46] [47] .

$$\theta = \theta - \eta.\nabla J(\theta) \tag{2.6}$$

The equation (2.6) is a gradient descent.

Where

η *is the learning rate,*

∇ *J* (θ) *is the gradient of loss function J* (θ).

Furthermore, stochastic gradient descent (SGD) is an optimization method for unlimited optimization problem, it performs a parameter update for each training example. It works by randomly picking out a small number of randomly chosen training inputs. It is used to speedup learning. It computes the average gradient for those examples, and updates the weights accordingly. This process continues for many small sets of training sets until the average of objective function stops decreasing. It is known as stochastic because each set of examples adds a noisy estimate of average gradient over all examples. See equation (2.7) and (2.8) for stochastic gradient descent [47] [49].

$$V_{k+1} = Vk - \eta.\nabla L(W_{ij}) \tag{2.7}$$

$$W = V_{k+1} \tag{2.8}$$

The equations (2.7) and (2.8) are stochastic gradient descent.

Where

W = *weights,*

η = *learning rate,*

∇ *L(W) is the gradient of loss.*

The gradient can be solved using the chain rule of the derive the loss function by weights.

$$\begin{aligned} \nabla L(W_{ij}) &= \frac{dL(W_{ij})}{dWij} \\ &= \frac{dL(t|y)}{dy}.\frac{dy}{dZ}.\frac{dZ}{dW_{ij}} \\ = \frac{1}{N}.\frac{t-y}{y.(1-y)}.y.(1-y).X_{ij} \\ &= \frac{1}{N}.(t-y)X_{ij} \end{aligned}$$

The SGD have a lot of oscillation, to achieve soft oscillation and acceleration the momentum term was invented. The basic idea of momentum is to increase the speed training. See equation (2.9), and (2.10).

$$V_{k+1} = \gamma V_i + \eta.\nabla L(W_{ij}) \tag{2.9}$$

$$W = W - V_{i+1} \tag{2.10}$$

2.9 Genetic Algorithm (GA)

Genetic algorithms are optimization techniques, or can be classified as a search heuristic to find the optimal solution(s), GAs are particular classes belonging to the evolutionary algorithms that use techniques inspired by evolutionary biology such as: inheritance, mutation, selection, and crossover [50]. The genetic algorithm reflects the process of natural selection, which starts

with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics from the parents, and will be added to the next generation. If parents have better fitness, their offsprings will have a better chance than parents at surviving when the parents have better fitness. This process will continue and at the end, achieve a generation with the fittest individuals [12] [50]. The genetic algorithm involves five phases: initial population, fitness function, selection, crossover, and mutation [12] [51].

1. *Initial population:* the initial population begins with randomly generated states (individuals) which is called a population. Each is a solution to the problem that needs to be solved [51].

2. *Fitness function:* the fitness function produces the next generation of states (individuals). It determines how fit an individual (is the ability of individuals to compare with other individuals). A good fitness function returns better states. It gives score to each individual, the probability that an individual of being chosen for reproduction is based on its fitness score [52].

3. *Selection:* based on their fitness scores two pairs of individuals (parents) are selected at random to reproduce. Individuals with high fitness have more chance to be selected for reproduction [52].

4. *Crossover:* is the most significant phase in genetic algorithm. For each pair of parents to be mated, a crossover point is chosen random from within genes. Offspring are created by exchanging the genes of parents among themselves until the crossover point is reached. The new offspring are added to the population [53].

5. *Mutation:* typically mutation occurs with a very low probability, which randomly flips individual bits in the new chromosomes (turning a 0 into a

1 and vice versa). Mutation occurs to maintain diversity within the population and prevents premature convergence. The algorithm terminates if the population has converged (does not produce offspring which are significantly different from the previous generation) [51].

2.9.1 Age-Layered Population Structure (ALPS) Genetic Algorithm (GA)

ALPS is a method that is used to reduce a common problem which occurs during running an EA known as premature convergence. Premature convergence happens when after a number of evolution, the population converges to local optima, that is no improvements occur regardless of how many times the EA is run. The difference between ALPS and any other EA is that it uses multiple layers for each population. ALPS measures how long the genetic materials have been evolving in the population: offspring age starts from 1 plus age of their oldest parent instead of starting from 0 as in the traditional measurement of age. This helps to randomly generate individuals in the youngest layer [54] [55]. The ALPS algorithm starts by configuring the age layers and then creating, and evaluating, an initial random population. After that initial population is created and evaluated, ALPS−Evolutionary Algorithm (EA) starts its main loop which is made up of cycling through the layers, from L_{n-1} to L_0, and then evolves the EA in that layer form one generation.

2.10 Discrete Wavelet Transform (DWT)

Wavelet enables dividing a complicated function into several simpler ones and studies them separately. The discrete wavelet transform (DWT) decomposes the signal into a set of mutually orthogonal wavelet basis functions. Wavelet functions are dilated, translated, and scaled versions of a common

function ψ, known as the mother wavelet. Wavelet theory is commonly used feature extraction methods for signal processing. The wavelet transform is a powerful and efficient time-frequency analysis method for analyzing non-stationary signals [56].

In DWT the original signal passes through two related Finite Impulse Response (FIR) filters, known as low-pass and high-pass filters, and emerges as two signals called approximation coefficients and details coefficients. DWT is more suitable for processing signals such as EEG, because it is powerful in time-frequency localization and multi-scale resolution. The signal x can pass through a series of filters to achieve DWT of the signal. Samples are passed through a low pass filter with impulse response *g* resulting in convolution, and are decomposed simultaneously using a high pass filter *h*. See Figure 2.16 for block diagram of DWT filter analysis [20] [56], and for discrete wavelet transform function see (2.11), and (2.12) [57].

$$W_\theta(j_0, k) = \frac{1}{\sqrt{M}} \sum_x f(x)\theta_{j0}, k(x) \tag{2.11}$$

$$W_\psi(j, k) = \frac{1}{\sqrt{M}} \sum_x f(x)\psi_j, k(x) \tag{2.12}$$

Where:

$\frac{1}{\sqrt{M}}$ is normalizing factor,

f(x), $\theta_{j0,k}(x)$ and $\psi_{j,k}$ are function of discrete variable,

The $\theta_{j0,k}(x)$ is a member of the set of expansion functions derived from a scaling function $\theta(x)$, by translation and scaling using:

$$\theta_{j,k}(x) = 2^{\frac{j}{2}}\theta(2^j x - k) \tag{2.13}$$

The $\theta_{j,k}(x)$ is a member of the set of wavelets derived from wavelet function

$\psi(x)$, by translation and scaling using:

$$\psi_{j,k}(x) = 2^{\frac{j}{2}}\psi(2^j x - k) \tag{2.14}$$

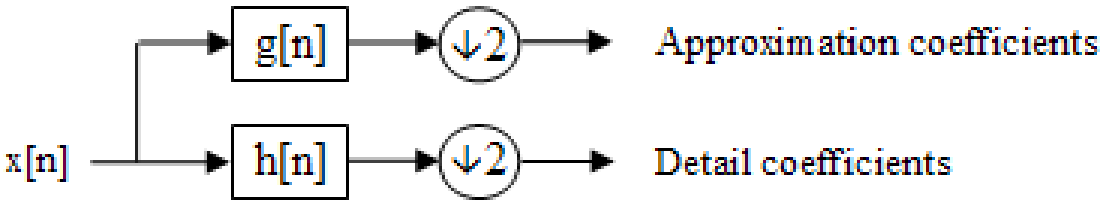


Figure 2.16: Block diagram of DWT filter analysis

In characterizing EEG signals, the low frequency components "approximation" are more important than the high frequency components "details". Equations (2.15), and (2.16) are used for finding high and low pass filtering, respectively.

$$Y_\psi[k] = \sum_n x[n]g[2k - n] \tag{2.15}$$

The equation (2.15) is high filter DWT.

$$Y_\theta[k] = \sum_n x[n]h[2k - n] \tag{2.16}$$

The equation (2.16) is low filter DWT.

The filters have a function of sub-sampling signal by 2. The dilation and shifting procedures are used for extracting the hidden information in the signal. It is necessary to select the correct and efficient wavelet function for specific applications [20].

2.11 Symbolic Discriminant Analysis (SDA)

Linear Discriminant Analysis (LDA) is a simple and effective method for classification. It involves three step:

1. Calculates the distance between the mean of the different classes.
2. Calculates the distance between the mean and the sample of each class.
3. Constructs the lower dimensional space.

   However, it has an obvious limitation and requires pre-specifying the linear discriminant function.

In addition, using groups for optimal classification of observations may not be possible with a linear combination of explanatory variables. To generate linear discriminant score, Sir Ronald Fisher suggests forming linear combination of measurements from multiple variables, see equation (2.17) there are two observations and k variables [58] [59].

$$l_{i,j} = \alpha_1 x_{ij1} + \alpha_2 x_{ij2} + ... + \alpha_k x_{ijk} \tag{2.17}$$

The equation (2.17) is a linear combination of measurements from multiple variables.

For *ith* group and the *jth* observation in the group where each α is a coefficient and each x is an explanatory variable.

SDA is developed to overcome these limitations. Symbolic discriminant analysis (SDA) is a supervised classification method, based on extending symbolic regression and a parallel genetic programming to identify optimal linear or nonlinear discriminant functions and coefficient. The original paper [58] applied this classification method to identify combinations of gene expression variables that differentiated acute myeloid leukemia (AML) from acute lymphoblastic leukemia (ALL).

SDA uses binary expression tree or symbolic equation to generate symbolic discriminant scores for each observation in each group. The function set consists of addition, subtraction, multiplication, and division (in respect to division by zero).

2.12 Interpolation

Interpolation is the process of finding unknown values from known values within the range of discrete set of known data points. There are several types of interpolation algorithm available, such as nearest neighbor, bilinear, bicubic,...etc. The simplest approach of interpolation is nearest neighbor interpolation. In this method each interpolated output data assigned the value of the nearest sample point in the input data point [60] [61].

2.13 Activation Function

Activation functions are used within artificial neural network to learn and make sense of those data that are really complicated and non-linear complex functional mappings between inputs and response variables. The aim of using activation function is to convert an input signal of a node in an artificial neural network to an output signal, sometimes activation function is known as transfer layer [62].

There are several types of activation function commonly used, i.e. sigmoid, RecLU, tanh, linear, and softmax. The sigmoid and softmax equations are displayed below: [25]

Sigmoid Activation Function:

$$y = \frac{1}{1 + e^{(-x)}} \tag{2.18}$$

Softmax Activation Function:

$$y = \frac{e^{x_i}}{\sum_{j=0}^{k} e^{x_j}} \tag{2.19}$$

2.14 Confusion Matrix

Confusion matrix is a table (matrix) used to summarize performance of classifier (classification algorithm) on a set of test data for which the true values are known. The confusion matrix has only two classes which are: positive and negative for binary classification problems [63]. See Table 2.1.

Table 2.1: confusion table

		Predicted	
		Negative	**Positive**
Actual	**Negative**	a	b
	Positive	c	d

Where:

*a is a number of **correct** predictions that an instance is negative,*

*b is a number of **incorrect** predictions that an instance is positive,*

*c is a number of **incorrect** predictions that an instance negative, and*

*d is a number of **correct** predictions that an instance is positive.*

2.15 F-measure

F-measure is the measure of accuracy during testing model. It is the harmonic average of the precision and recall. F-measure reaches the best value at 1 (perfect precision and recall) and worst at 0. Assume that the raw EEGs signal in the test set A. The classifier predicts a class for each raw EEG data, and these expectations will fall into four groups [63] [64].

1. True positive: the number of raw EEGs signal that are in category *A*, which the classifier correctly predicts to be in the category *A* [64].

2. True negative: the number of raw EEGs signal that are not in category *A*, which the classifier correctly predicts not to be in the category *A* [64].

3. False positive: the number of raw EEGs signal that are predicted to be in category *A*, which the classifier falsely predicts to be in the different category [64].

4. False negative: the number of raw EEGs signal that are predicted not to be in category *A*, but are actually in category *A* [64].

The accuracy rate, and error rate are used to evaluate and examine performance of classification algorithm in machine learning and statistics. The percentage of correct predictions determined by accuracy rate and error rate is the percentage of incorrect predictions of the classifier also called misclassification [63] [64]. Accuracy and error rate are computed as the following:

$$Accuracy = \frac{TN + TP}{TP + FP + TN + FN} \tag{2.20}$$

$$Error = \frac{FN + FP}{TP + FP + TN + FN} \tag{2.21}$$

The precision (sometimes known as positive predictive value) value is calculated according to equation (2.22). It answers the question: when it predicts the positive result, how often it is correct? [63] [64].

$$Precision = \frac{TP}{TP + FP} \tag{2.22}$$

Recall sometimes known as sensitivity, is calculated according to equation (2.23). It answers the question: when it is actually the positive result, how often does it predict correctly? [63] [64].

$$Recall = \frac{TP}{TP + FN} \tag{2.23}$$

The F-measure is calculated according to the equation (2.24)

$$F - measure = \frac{2 * recall * precision}{recall + precision} \tag{2.24}$$

2.16 LEGO Mindstorms NXT

LEGO Mindstorms NXT is a type of programmable robotics kit, the first generation released in late July 2006, which replaced Lego Mindstorms kit known as robotic invention system. The base kit has two versions which are retail versions (set 8527) and education base set (9797). It basically uses NXT-G programming software. However, the Lego Mindstorms (see Figure 2.17) can program using LabVIEW or through many other kinds of programming languages such as NBC, NXC, leJOS NXJ, and RoboticC [65] [66].

The second generation of this robot which is known as Lego Mindstorms NXT 2.0, was launched on August 1, 2009, and the third generation was released in September 2013 [67].

LEGO Mindstorms NXT is made up of three main components which are: NXT intelligent brick, sensors, and motors (actuators). The main component in the kit is the brick-shaped computer, it can take input from up to 4 sensors simultaneously, and control up to 3 motors through modified version RJ12 cables. Users can interact with the robot's firmware through a monochrome LCD screen available on the front side of the intelligent brick with four buttons, it has a hierarchal menu. The power is supplied by 6 AA (1.5 V) batteries.

By default the Lego NXT 2.0 has three sensors which are: color sensor, ultrasonic sensor, and touch sensor. Also there are a lot of sensors not included in the Lego Mindstorms NXT which may be bought separately such

Figure 2.17: Chapter 2: Lego Mindstorms NXT 2.0

as: sound sensors, compass, gyroscope...etc. Except the sensors, three actuators (motors) are available with each Lego NXT Mindstorms, the robot performs actions through these actuators [67].

Chapter 3

Design and Implementation

3.1 Introduction

In the previous chapter, the topics related to brainwave signals such as: signal behavior and techniques used for analyzing and classifying these signals have been presented.

In this chapter, the diagram of overall system, procedures of data acquisition, data preparation, data pre-processing, architecture of proposed models for classifying brainwave signals, and the robot design are described in detail.

The proposed models, and the original CNN model have been implemented using C# programming language on Dell XPS 14Z core i7 laptop while the robot has been programmed using LeJOS framework.

3.2 The Proposed Brain Computer Interaction (BCI) System

In this study, two models for classifying brainwave signals have been proposed, and the accuracy of these two models have been investigated by comparing them with the CNN model. These models are trained and tested for color and shape classification.

A complete dataset has been created by recording EEG signals from six participants. Then the CNN with both proposed CNN methods are applied on the experimental dataset within two different states.

3.2.1 First Proposed Model

In the first proposed model, a hybrid model based on deep learning technique and an evolutionary algorithm has been designed. The model was designed by combining CNN as a deep learning technique with ALPS-GA as an evolutionary algorithm. Two basic modifications have been applied in this model. First, replacing pooling (max or avg) layer in the second layer and applying DWT Coiflet 1 instead of it. Second, using ALPS-GA with SDA instead of fully connected layer.

3.2.2 Second Proposed Model

The second proposed model, has been designed based on CNN, the modification in this model composed of removing sub-sampling layer (pooling) in the second layer, and two different DWT methods (Coiflet 1[1], Symlet 2[2]) have been used instead of it, individually. Then, SGD is used as a trainer in the fully connected layer. The aim of this model is to investigate results achieved from using two different DWT (Coiflet 1, Symlet 2) methods individually instead of the pooling layer, with SGD as a trainer in the final (fully connected) layer. Additionally, except both proposed models, the CNN is also has been applied on the experimental dataset. The architecture of this model follows the same stages of CNN steps which are: convolution, pooling (max pooling), and fully connected layers as mentioned in section 2.8.1. In this work the emphasis was on obtaining the best classification with a higher possible accuracy rate. Designing and implementing the whole proposed BCI system model is composed of two stages: the first stage is designing and implementing the classifier model which has the ability to perform classification on EEG signals (stages of BCI system mentioned in section 1.1, see Figure

[1]The coefficients in both low-pass and high-pass filter decomposition are 6 coefficients.
[2]The coefficients in both low-pass and high-pass filter decomposition are 4 coefficients.

1.1). The second stage involves two steps; hardware design of the robot, and developing an application to the robot that is responsible to perform different actions according to the received command. See Figure 3.1 for the proposed model architecture

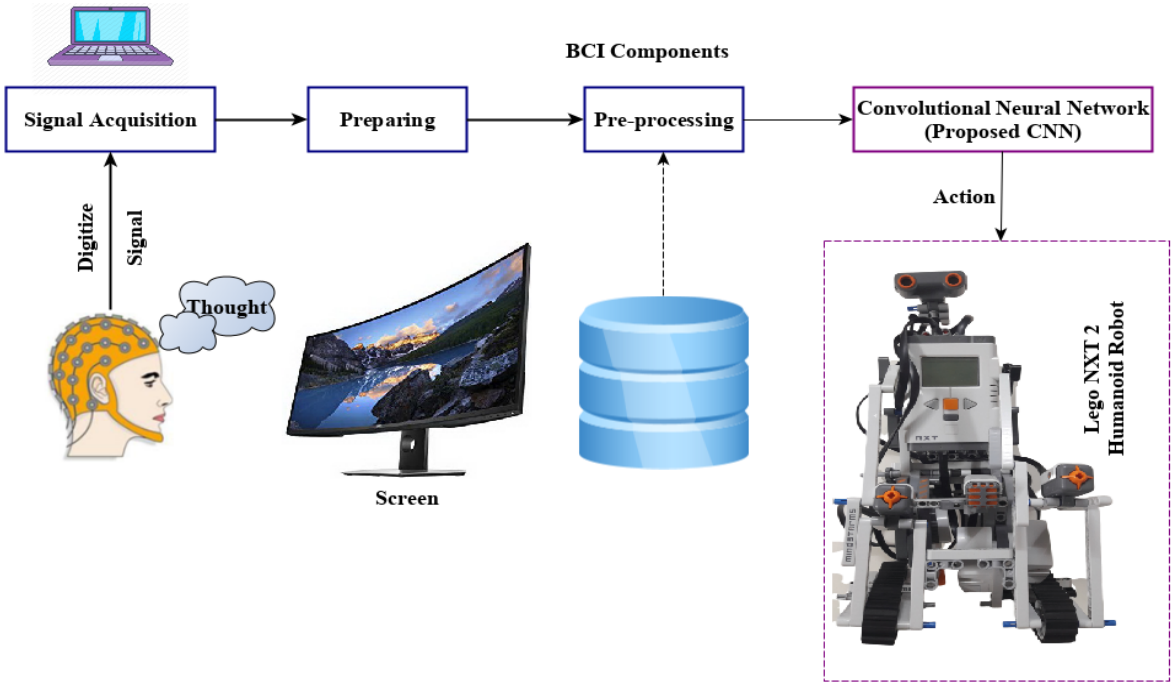


Figure 3.1: proposed model architecture

In this work, to build the classifier model, three steps have been followed. First, recording brainwave signals from subjects of different ages, and storing the signals in separate CSV files. Second, preparing and pre-processing the data. The preparing step refers to fetching the desired EEG signal in each CSV file and merging them in a single file in respect to their mode. Furthermore, the pre-processing has been applied on the prepared data. In rare cases the prepared data had less data points as expected. Thus, interpolation process has been used in this step to overcome this problem. Finally, the third step is the training stage. The results from pre-processing step are fed to the proposed models. The output weight of best results (conv layer = 30 weights) (dense layer = 4994 weights) are stored in the files for further use in the matching (testing) stage. The first model and second model flowcharts

have been displayed in Figure 3.2, and 3.3, respectively. The matching stage implemented in two different ways: first, recording brainwave signals from subjects, then applying preparing and pre-processing steps, and feeding the pre-processed data to the proposed CNN with stored weights, to perform the classification process and predict the output. However, in the second way, the only difference from the first form is using stored EEG signals that were previously recorded instead of directly recording EEG from the subjects, all the other steps remain the same as in the first way mentioned.

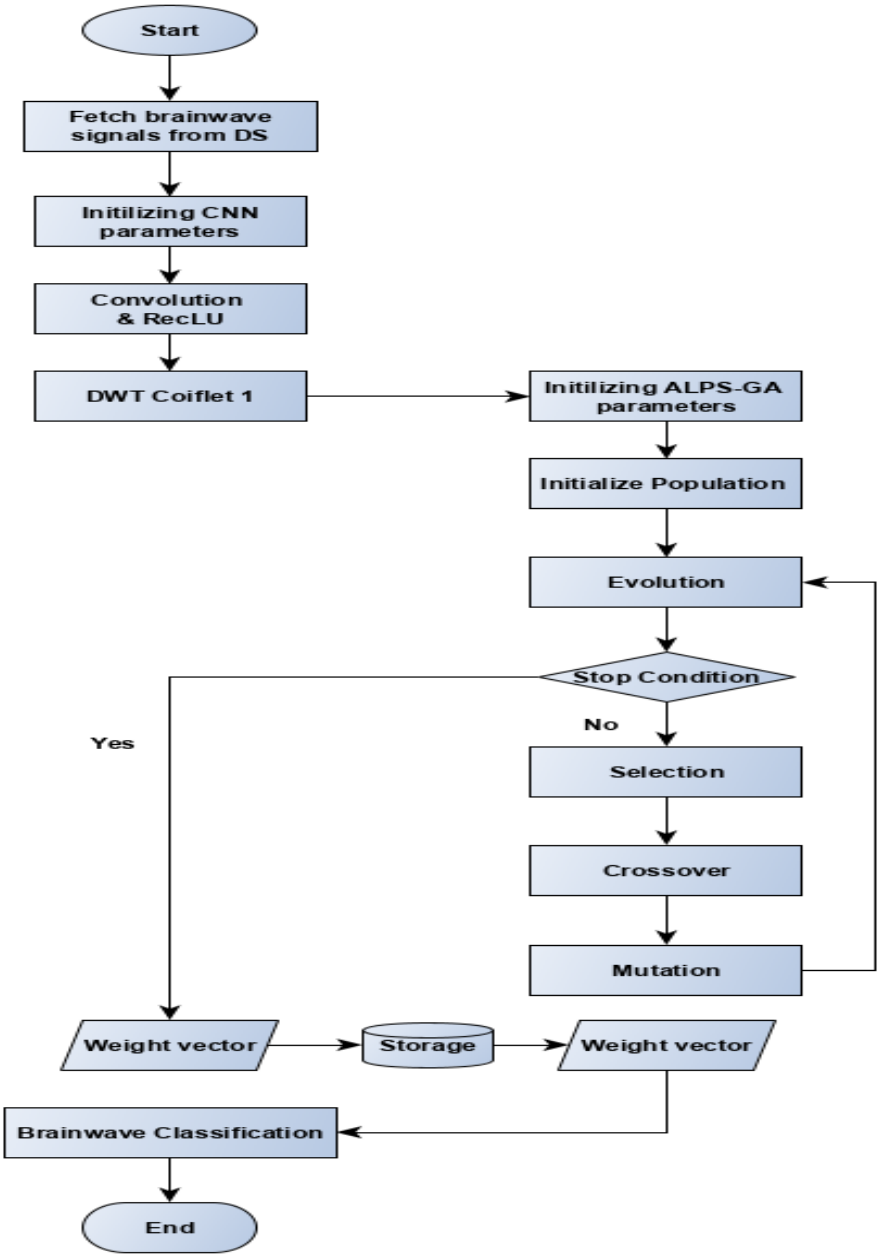


Figure 3.2: first proposed model flowchart

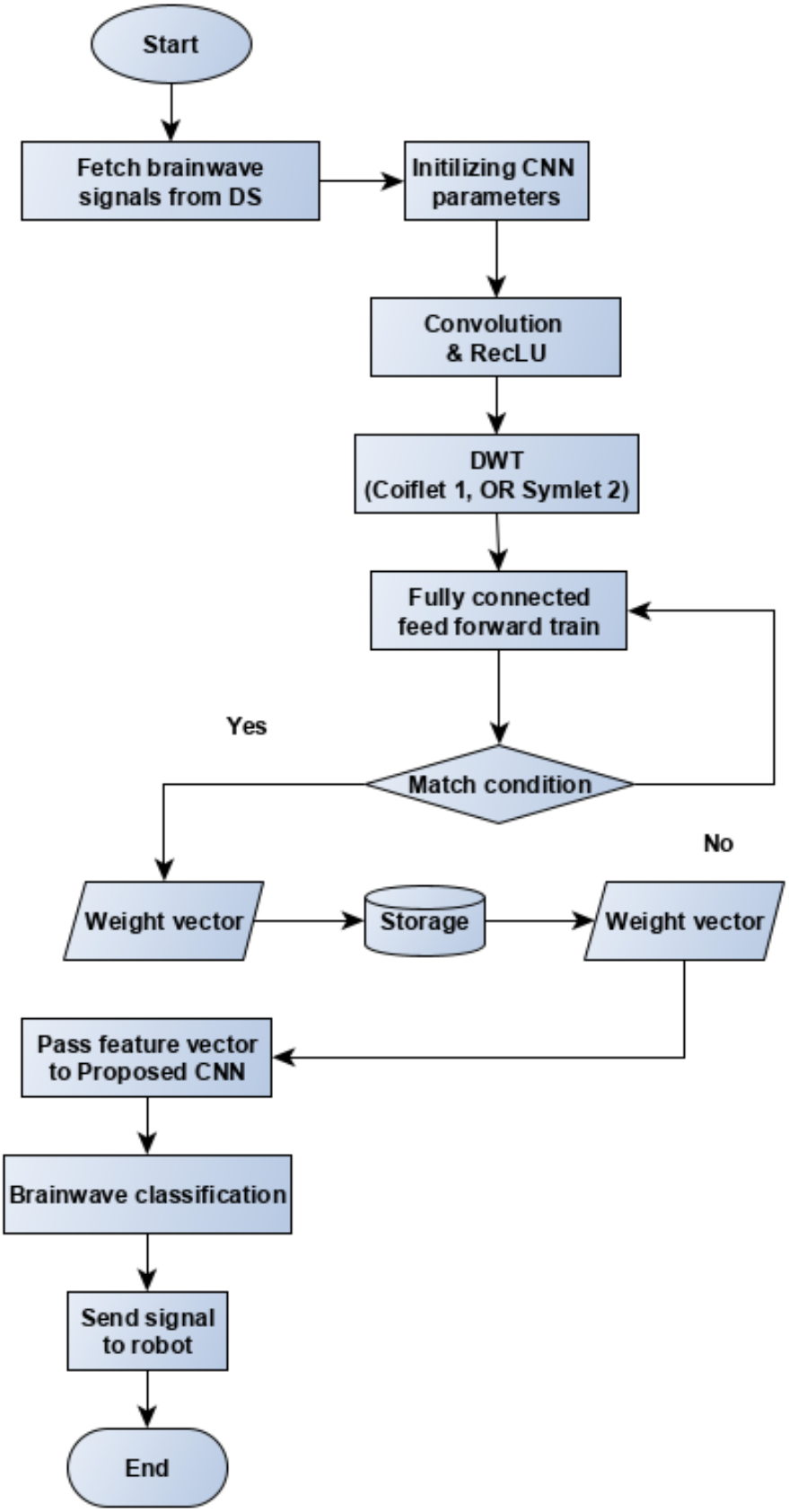


Figure 3.3: second proposed model flowchart

3.3 Brain Monitor Application

In this work a complete dataset has been created from scratch by recording brainwave signals. A comprehensive brain monitor application has been built. This application has been used for establishing the communication with the NeuroSky device, handling and monitoring brain signals.

This application provides an interface to monitor brainwave activities, i.e. plotting the power spectrum for each bands (alpha, beta, theta, delta and gamma), plotting the raw EEG data, presenting meditation, attention, blinking strength, and signal quality level. All mentioned data have been received once in a second, except the raw EEG data have been received at rate 512 Hz. They have been used for creating the dataset according to the requirements in different conditions. The application is made up of two windows. The main window shows the NeuroSky connection manager, session manager, robot operation, EEGs chart options, eSense meter, eSense options, EEG signals chart, and Listening (see Figure 3.4). On the second window, the received raw EEG signals are presented as a chart, see Figure 3.5a. The component of the main window are explained below:

1. NeuroSky connection management: this part is responsible for establishing a connection between the NeuroSky device and brain monitor application through the COM port.

2. Session manager: this part can be used for creating a directory for each subject before recording brainwave signals, then organizing and storing the brainwave signals in the individual CSV files with respect to session number, categories (shape or color), and their mode (visible or invisible)(see Figure 3.5b). It can also be used to display the specified color or shape according to selected options, (i.e. color (red or green) or shape

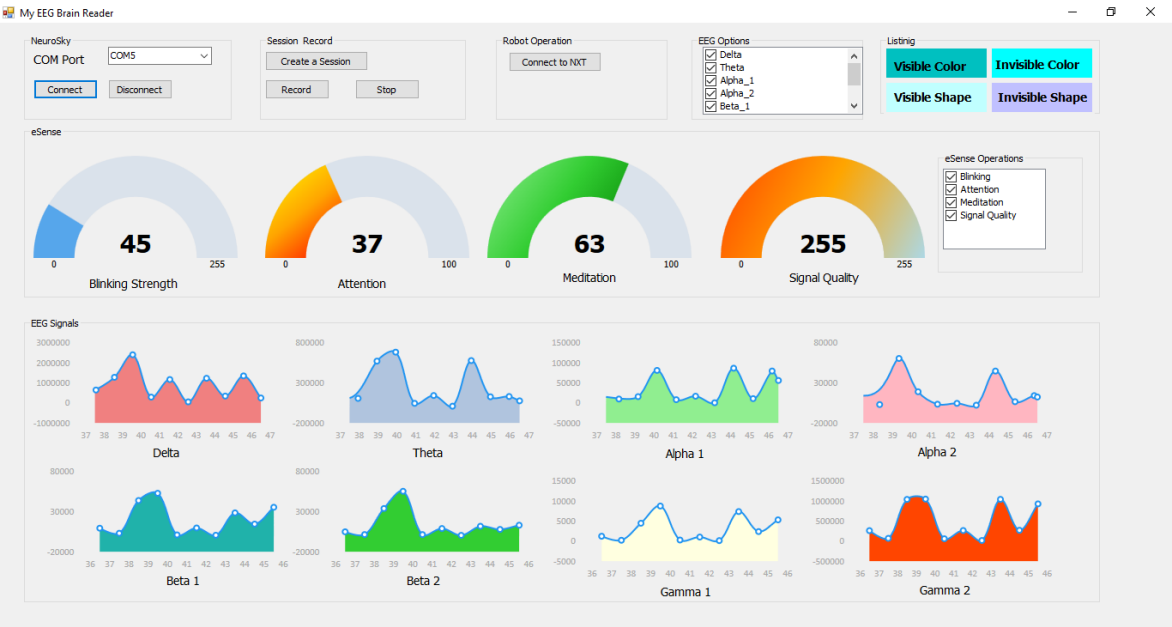
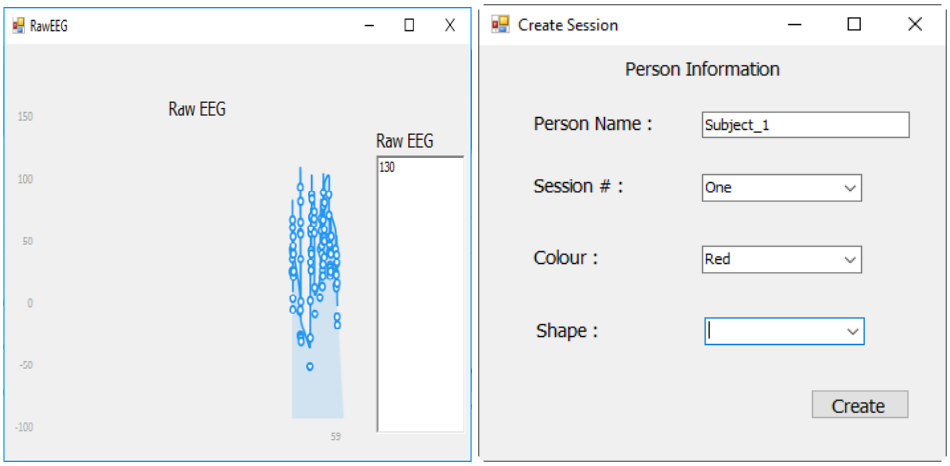


Figure 3.4: brain monitor application.



(a) one second Raw EEG data. **(b)** session manager window.

Figure 3.5: two sub-window

(forward or right)) on the screen (second screen) to the participant.

3. Robot operation: during the testing process, establishing a connection between the application and the robot is handled through this section.

4. EEG options: this region can be used for presenting or hiding EEG signal's band charts. The current EEG signal's band charts can be shown individually or simultaneously in this area.

5. Listening: four labels are shown in this part; visible color, invisible color, visible shape, and invisible shape. They can be used during testing when

the application is trying to classify the received signals. The signals can be interpreted to command then the command will be send to the robot.

6. eSense monitor: This region shows four gauges for monitoring the device signal quality, attention level, meditation level, and blinking strength. Meanwhile, each one of these gauges can be shown or removed individually or simultaneously.

7. EEG bands: All EEG signal's bands are displayed in this section individually.

3.4 Data Acquisition

In this section, the procedure of recording brainwave signals from participants, and circumstances of recording brainwave signals have been presented. The dataset has been built from six healthy subjects. Each subject has normal mental state, normal color vision, and age ranging between 25 to 35 years old.

The dataset has been made for two main categories which are: colors and shapes. Red and green colors were used in the color mode. According to the RGB standard, the red color was constructed by setting the red value to 255, while both green and blue values were set to zero (255, 0, 0). The green color is constructed by setting the green value to 255, with both red and blue value being zero (0, 255, 0). Forward and right arrows have been selected for the shape category. All sessions of brainwave signals have been recorded under two modes, which were: visible and invisible. In the visible mode, subjects focused on the colors and shapes presented to them. However, in the invisible mode, subjects thought about specific colors or the shape with closed eyes (Figure 3.6 presents the architecture of the dataset).

All sessions have been recorded by using the brain monitor application as

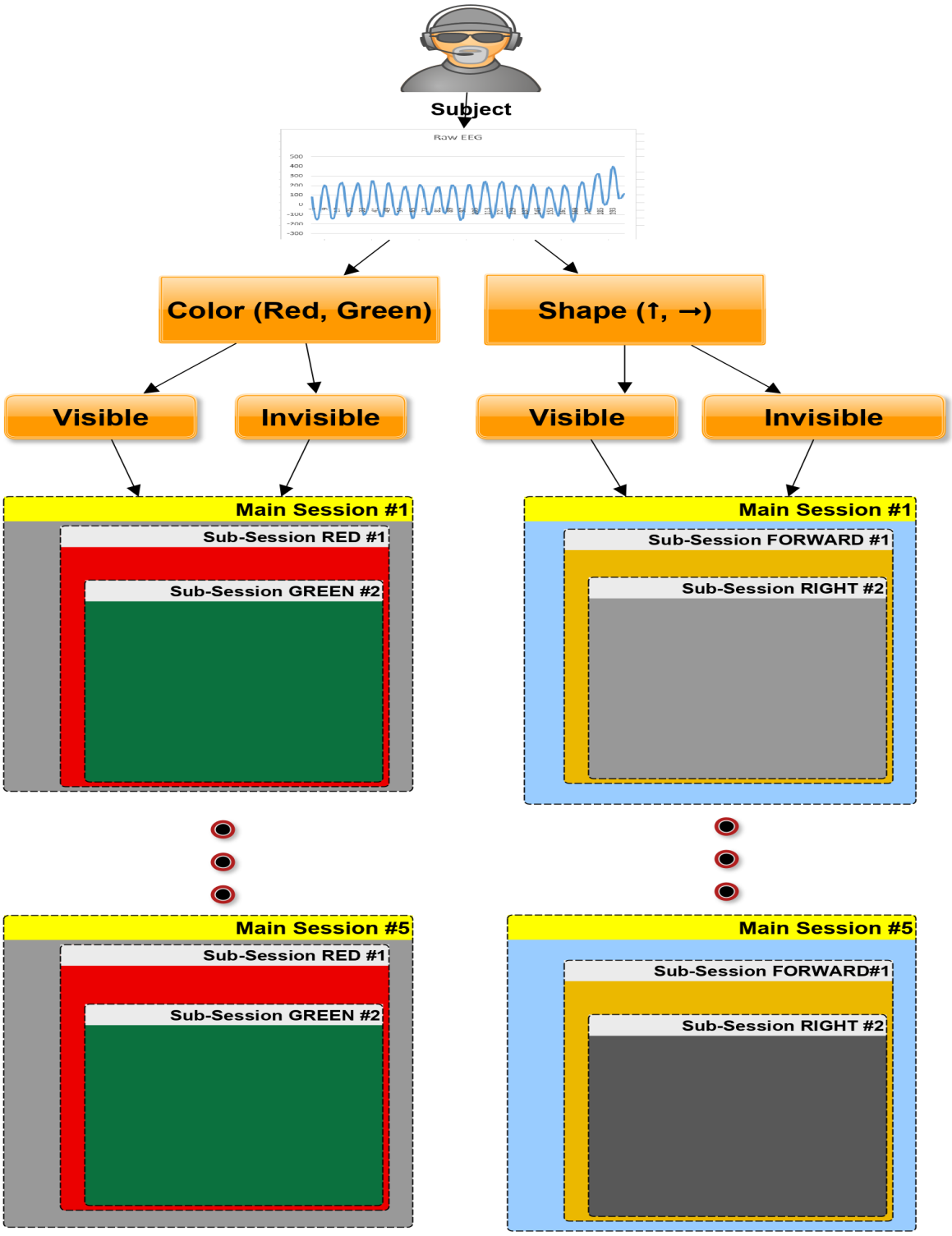


Figure 3.6: dataset architecture.

mentioned in section 3.3, the recording has been administrated for each participant in a dark room. Each subject has been seated on a comfortable chair faced with a 43 inch screen. The screen was set to standard color mode with

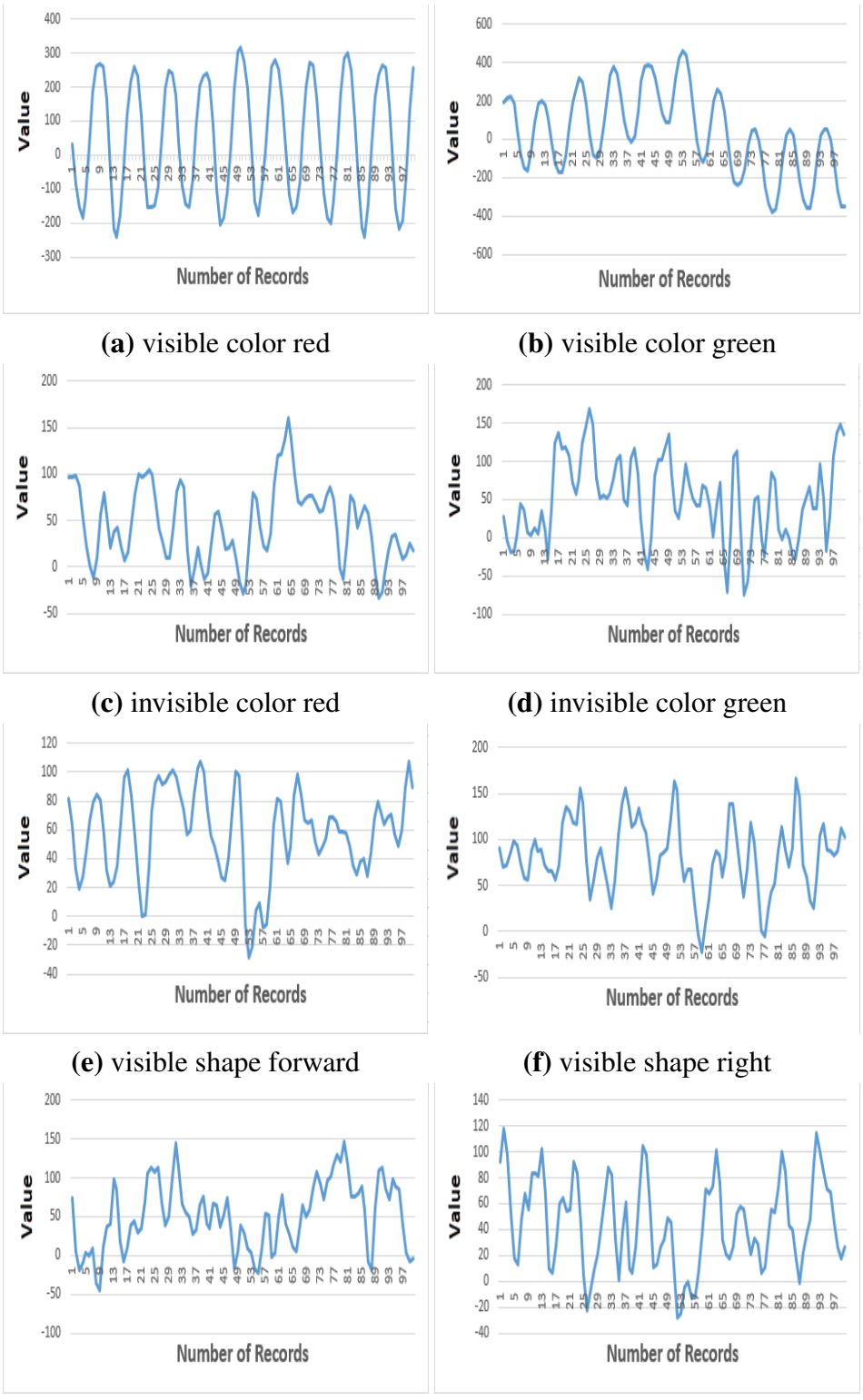
normal brightness and contrast. The distance between the subjects and the screen was 125 centimeters.

Adequate information had been provided for each subject before recording any session, i.e. how many sessions would occur during recording brain signals, and how many times the color and the shapes would be presented. After installing and configuring the EEG device on the subject's head, the signal quality has been checked several times. Then, the participant has asked to look at the screen, do some brain exercises, and try to increase the attention level. In this study, the subject's concentration is compulsory to achieve the quality brainwave signals.

The dataset structure has been organized as follows, the dataset has two main categories which are: colors and shapes. Each category has been recorded under two modes which were: visible and invisible. Each mode have five separate sessions for each individual category. Each session have two subsessions i.e. red, green for colors and forward, right for arrows, each subsession is recorded for 25 seconds.

Two separate CSV files have been created for each sub-sessions and individual subjects. The first file stores the raw EEG signals with its time, while the second file contains recording time, brainwave band (α, β, γ, δ, and θ) values, attention level, meditation level, blinking strength, and signal quality level.

Figures (3.7a, 3.7b, 3.7c, and 3.7d) show the raw EEG of 100 data points, for visible color red, visible color green, invisible color red and invisible color green, respectively. Figure (3.7e, 3.7f, 3.7g, and 3.7h) present the raw EEG signals of visible shape forward, visible shape right, invisible shape forward and invisible shape right, respectively.



(a) visible color red

(b) visible color green

(c) invisible color red

(d) invisible color green

(e) visible shape forward

(f) visible shape right

(g) invisible shape forward

(h) invisible shape right

Figure 3.7: raw EEG record for 100 data points

3.5 Data Preparation

When the process of data collection was completed, it is essential to calculate the total duration of all sub-sessions, then calculate the dataset duration. The duration of all sub-sessions have been calculated by using equation (3.1).

$$s = \sum_{i=1}^{p} n_i.t_i \tag{3.1}$$

where:

n = number of sessions.

t = duration per session.

p = number of subjects.

s = total sub-sessions for each mode.

Also, equation (3.2) has been used for calculating the total time duration for both mode individually with respect to their category (color and shape).

$$ts = \sum_{i=1}^{m} s_i \tag{3.2}$$

Where:

ts = total duration of both (colors or shapes) in each mode.

m = number of colors or shapes.

s = total sub-sessions duration in seconds in each mode.

Table 3.1 contains the total duration recorded in each mode for both colors and shapes individually, and the total duration of the entire dataset.

Therefore, the total duration of the dataset is 6000 seconds, which is equal to 100 minutes of brainwave signals.

As mentioned before, after the brainwave signal was recorded, each sub-session of brainwave signals for each subject was saved in a separate file with CSV format. Each sub-session contains 25 seconds of raw data. During

recording it was noted that file related to the visible mode contained approximately 506 records per second. However, in the invisible mode in rare cases less than 500 records were received per second.

To achieve the accurate brainwave signals relevant to the presented color or shape, the subject's concentration is compulsory. Therefore, in this work, in each sub-session only 10 seconds of 25 seconds of brainwave recording were fetched. These 10 seconds had maximum attention, which were fetched and stored with respect to the order of recording signals.

Table 3.1: dataset duration details

Mode	**Colors**	**Duration/Secs**	**Shapes**	**Duration/Secs**	**Total/Secs**
Visible	Red & Green	1500	Forward & Right	1500	3000
Invisible	Red & Green	1500	Forward & Right	1500	3000
Dataset Total Duration					6000

3.6 NeuroSky Data Pre-processing

ThinkGear ASIC (Application Specific Integrated Circuit) Module (TGAM) is the core of brainwave sensing technology. NeuroSky mindwave mobile 2 uses the TGAM1 module, it has an advanced filter with high noise immunity, which captures the human brain signals, then filters out irrelevant noise and all electrical interference and converts the signals to digital power. However, some pre-processing techniques i.e. A/D, amplifying the signals, and notch filtering frequency are applied on the signal before transmitting the brainwave signals.

3.7 Data Pre-processing

In this study, along with Neurosky pre-processing techniques as mentioned in 3.6, another pre-processing method has been applied on the brain-

wave signals before feeding the signal to the proposed architecture. As mentioned in section 3.5, in rare cases the fetched data had less than 5000 records per 10 seconds. To overcome this problem neighbor interpolation have been applied on the brainwave signals to reach a certain number of brainwave records. Finally, the fetched data were merged into single CSV file according to their categories with respect to their mode. See Figure 3.8 for data preparation flowchart.

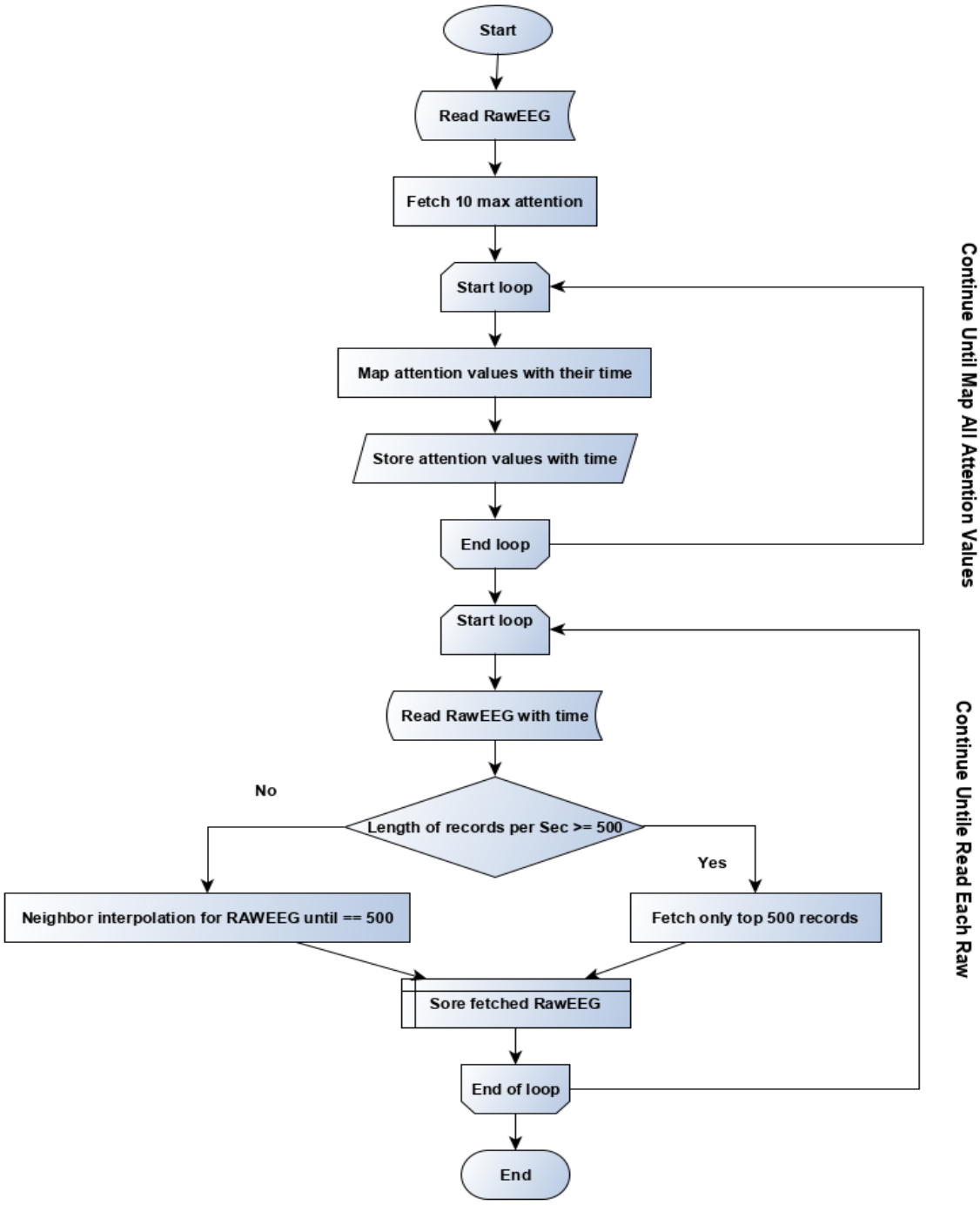


Figure 3.8: data preparation flowchart

The algorithm has been applied on each sub-session file. As the result a CSV file was generated which contains the raw data with respect to their categories (shape and colors) and their mode (visible and invisible). Total records in the CSV files were calculated according to equation (3.3).

$$s = \sum_{i=1}^{p} n_i.r_i \tag{3.3}$$

where:

n = number of sessions.

r = number of records in sub-sessoin per 10 seconds.

p = number of subjects.

s = total sub-session records in each mode.

After data preparation and pre-processing have been done, another dataset was extracted from the dataset. Each 5 seconds of the dataset have been fetched separately to build a new dataset. The aim of extracting a new dataset from the original dataset was to make a comparison when using proposed architectures with different input data.

Table 3.2 provides details on the number of sample, the sample size, number of records per sample, total dataset records, and batch size per category. The dataset has been sampled based on 10 seconds and 5 seconds, and prepared dataset based on maximum attention level for both duration 10, and 5 seconds.

Table 3.2: sample size details

Type	Sample Size	Number of Sample	Number of Records	Batch Size
Dataset (10 Secs)	5̃060 records	600	3̃,036000	-
Dataset (5 Secs)	2̃530 records	1200	3̃036000	-
Extracted #1 (10 Secs. Max Att.)	5000 records	240	1,200000	60 / mode
Extracted #2 (5 Secs. Max Att.)	2500 records	480	1,200000	120 / mode

Ten raw of EEGs data point, have been displayed in Table 3.3, based on different categories with respect to their mode.

Table 3.3: ten raw EEGs data points

Cate.	Type	Ten raw of EEGs Data Point									
Vis. Color	Red	32	-84	-152	-187	-124	19	185	260	268	260
	Green	192	217	219	182	60	-73	-153	-163	-61	90
Inv. Color	Red	96	96	98	87	53	22	1	-12	10	57
	Green	28	-4	-19	-17	3	44	37	7	3	12
Vis. Arrow	Forward	82	64	33	19	27	45	67	80	85	81
	Right	90	70	72	84	98	93	73	56	58	87
Inv. Arrow	Forward	74	6	-19	-10	4	0	9	-35	-45	10
	Right	92	118	98	53	18	13	48	68	56	83

3.8 Defined Kernels

As mentioned in section 2.8.2, the first layer in each CNN architecture is a convolution process. The convolution is an element-wise matrix multiplication between the input data and defined kernels. Kernels have a direct impact on the whole classification process, and to perform convolution process, kernels (filters) are required. There are three fundamental parameters which should be define for making any set of kernels that are: width, height, and number of the kernel (count). The format of a written kernel architecture should be as follows [width - height, count]. In this study three different kernel size, each with three different kernel counts have been investigated. In the beginning kernel values have been generated randomly between 0 and 1.

3.9 Architecture of the CNN

As mentioned in section 2.8.1, the CNN is made up of three main layers which are: convolution, pooling, and fully connected. The CNN accepts input data in the form of volume (tensor). However, in this study, the input data is raw EEG, and the raw EEG consists of a single raw data (one dimensional - 1D). Thus, the input data for the CNN and both proposed models is set to 1, 5000, 1 for width, height, and depth respectively. The model has been experimented many times with different kernel numbers in a different

size (trial and error). Based on the best result achieved the number of kernels used with this model is set to 3 kernels, each kernel size is set to 1, 10 with 1 stride (number of data point shifted in single convolution operation). Kernels are used in the convolution process to filter and extract some features from input data. The convolution process has been done between the raw EEG as input data with kernels (filters), then RecLU activation function has been applied on the each feature map, see equation (2.3). The RecLU is an activation function to set the negative values with 0. The output from convolution layer was fed to pooling layer, in this study the CNN model pooling size is set to 5 (between each five unit the maximum value was taken) and 1 stride. The extracted features from pooling layer was fed to the fully connected layer; and softmax equation has been used in the final layer to predict the output (equation (2.19)), see Figure 3.9.

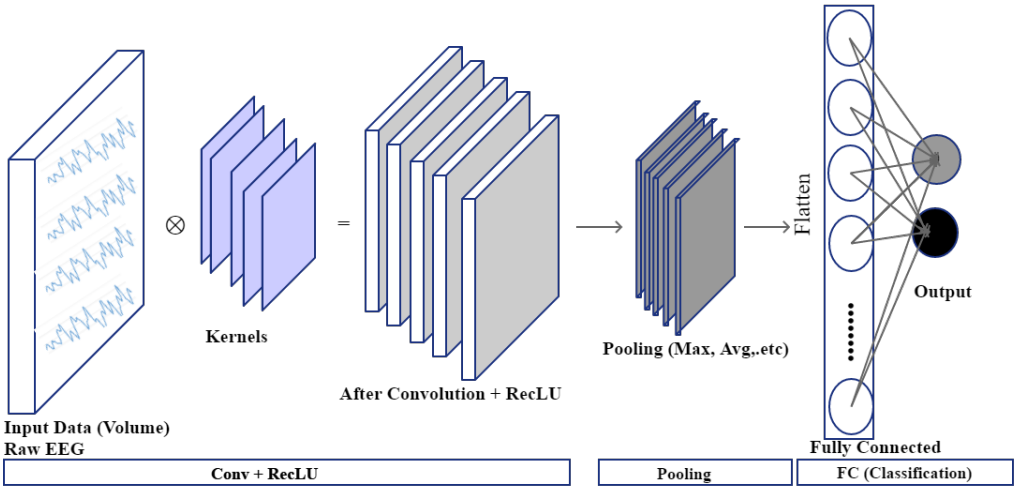


Figure 3.9: CNN architecture.

3.10 Architecture of Proposed Models

In this section, the architecture of the proposed models have been described in detail. In this work two different architectures are proposed that are based on CNN.

3.10.1 First Models

First Model: In this model, two modifications have been applied on the original CNN architecture. First, pooling layer which is placed on the second order layer has been completely removed. Coiflet 1 which is a member of the DWT families and is derived from daubechies wavelet has been placed in the second layer instead of the (max, or average) pooling layer.

The second modification was replaced the fully connected (dense) layer with ALPS-GA and SDA.

The input data flows through these layers: first layer, is convolution layer, The convolution process has been applied between the input data (raw EEG) and the kernels (filters).

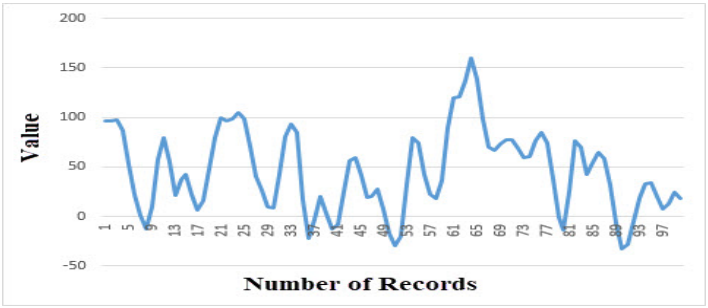
Kernel is a matrix with the dimensions (height, width, and depth), kernels have been taken in small sizes and moved the kernels individually across the whole input data and perform convolution process. Each kernel has a set of values, kernel's values have been used to filter and select features from the input data. Selecting or defining kernels while working with brainwave signals is a complex task compared to selecting filters for image processing, because there are a lot of filters available for image processing such as: sharpening, smoothing, blurring, edge detection, ... etc. Unfortunately, there are no standard kernels available to work with brainwave signals due to non-stationary nature of these signals.

In this work, in the beginning kernel values were defined randomly, the model was experimented many times (trail and error) until the best result has been achieved. Each time the kernel values have been stored, and the best kernel values have been selected as filters to do the convolution process in the first layer of the proposed model.

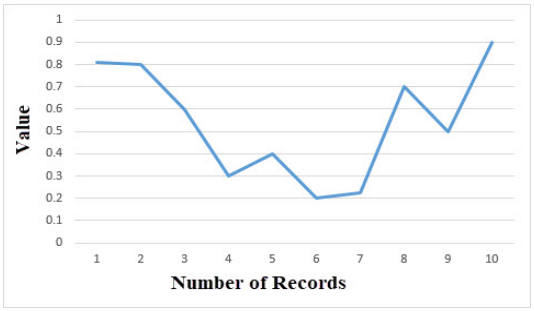
As mentioned in section 3.9, the input volume of this model was set to 1,

5000, 1 for width, height, depth respectively.

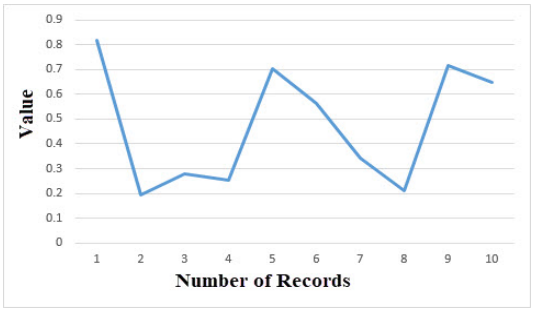
The convolution operation is an element-wise matrix multiplication between the kernels values and the raw EEG data and the resultant values are summed in (see Figure 3.10a, 3.10b, 3.10c, 3.10d, and 3.10e).



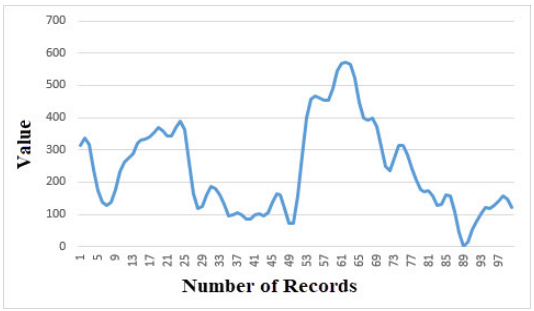
(a) invisible color "red" raw EEG



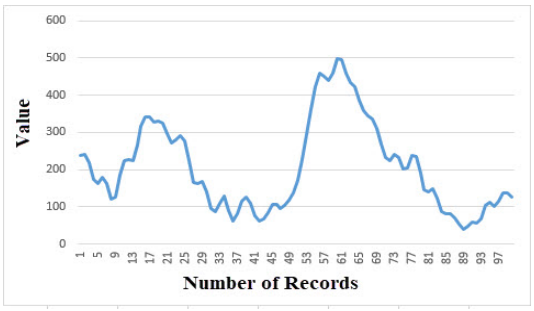
(b) first kernel



(c) second kernel



(d) first feature map

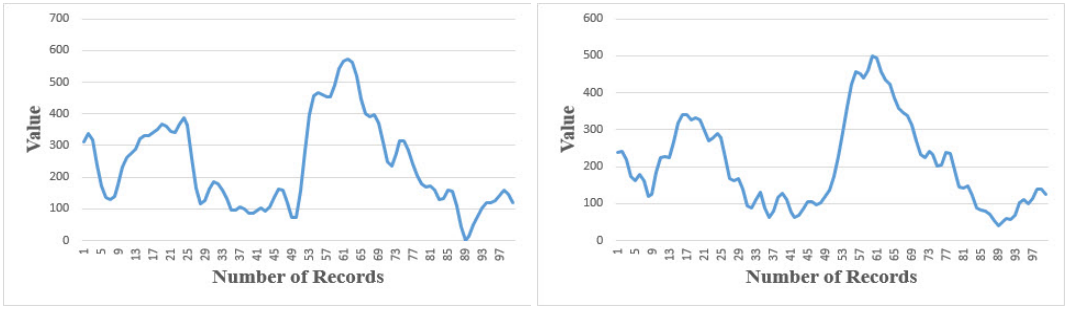


(e) second feature map

Figure 3.10: raw EEG, kernels, and feature maps

The Rectified Linear Unit or RecLU is not a separate process from the convolutional process. It is an activation function has been applied on each feature map (see equation (2.3). The aim of RecLU is to increase non-linearity, it removes all negative values from feature maps, and set values to 0 (see Figure 3.11a, and 3.11b).

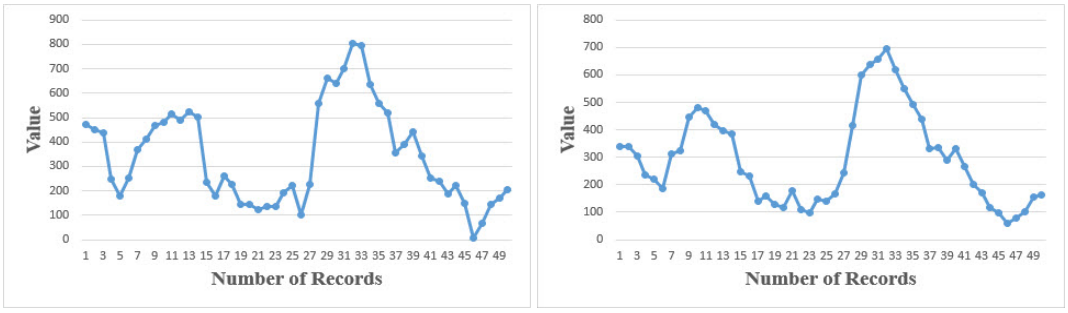
Figure 3.12a, and 3.12b show that the output generated from the first layer (the convolution) layer has been used as an input to the second layer which



(a) RecLU for first raw EEG **(b)** RecLU for second raw EEG

Figure 3.11: RecLU for invisible "red" color feature maps

is defined as DWT Coiflet layer. In the DWT Coiflet 1 layer, the most interesting features have been extracted and the size of the received features has been reduced by 2.



(a) DWT Coiflet 1 for first raw EEG **(b)** DWT Coiflet 1 for second raw EEG

Figure 3.12: DWT Coiflet 1 for invisible color "red"

Finally, extracted features have been fed to the ALPS with the SDA to predict the output.

Figure 3.13 presents the architecture of the proposed model, all modifications in the model are enclosed with blued dashed border.

The following pseudo code explains the algorithm of the proposed model which is applied on the dataset.

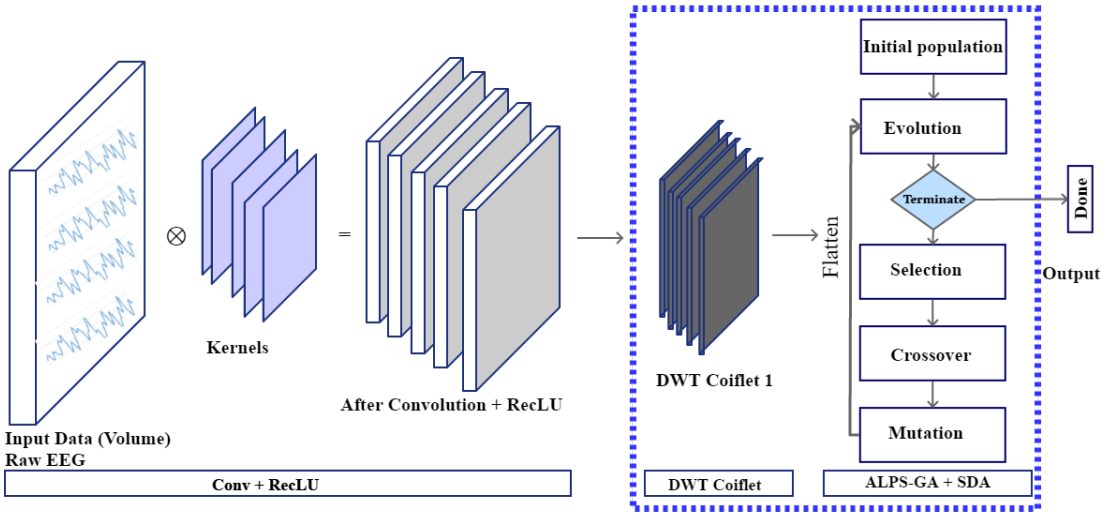


Figure 3.13: first proposed model architecture

Start

Declare variables RawEEGData, Kernels, Convolved, Rectified, Coiflet, MaxGeneration and Counter

Initialize ***Kernels*** *randomly between (0 and 1);*

Counter $\leftarrow$ ***0***

For all *RawEEGData list* ϵ *dataset* ***do***

Convolved $\leftarrow$ *Calculate convolution between (kernels and list)*

End for

For all *Convolved features* ϵ *Convolved* ***do***

Rectified $\leftarrow$ *Math.Max(0, features)*

End for

For all *Rectified row* ϵ *Rectified* ***do***

Coiflet $\leftarrow$ *Apply coiflet of order 1 on each row.*

End for

While *(Counter < MaxGeneration)* ***do***

For all *Coiflet row* ϵ *Coiflet* ***do***

Fed each row of the Coiflet using ALPS algorithm (section 2.9.1), and SDA as classifier

Compute Mean Square Error

$$Error = \frac{1}{N} * \sum_{i=1}^{NumOutput} (T(i) - Output(i))^2$$

End for

Counter $\leftarrow$ *Counter + 1*

End While

End.

3.10.2 Second Model

Second Model: This model is also inspired from CNN the aim of this model is to investigate the effect of applying two different DWT (Coiflet 1, and Symlet 2) methods individually (instead of pooling layer) and combining with SGD as optimizer in the fully connected layer. In this model the first layer convolution remained unchanged, while the second layer (pooling layer) has been removed, instead of it applying DWT (Coiflet 1, or Symlet 2) individually. Then the result received from second layer has been fed to the third layer (fully connected layer), SGD (see section 2.8.5) have been applied in this step. SGD is an optimization method for unlimited optimization problem. A true gradient has been estimated based on a single training example at a time or it performs one update at a time. Finally, the softmax function has been used to predict the output classes (equation (2.19)).

The input data flows through these layers: First, the convolution process has been applied between the input data (raw EEG) and the feature detectors (kernels), the kernels used for filtering and extracting some features from the input data. Then, in order to increase non-linearity, RecLU activation func-

tion (see equation (2.3)) has been used on the (each feature map) convolved features. See Figure 3.14, 3.15a, 3.15b, 3.15c, 3.15d, 3.16a, and 3.16b. Second step, two different of DWT (Coiflet 1, and Symlet 2) methods, individually have applied on the received features to extract the most interesting features. Finally, the features have been flattened and fed to the fully connected layer with respect to SGD in the final layer, and softmax activation function has been used for predicting different classes. In this model, the result after applying DWT Symlet 2 on the signals has been presented in Figure 3.16c, and 3.16d.

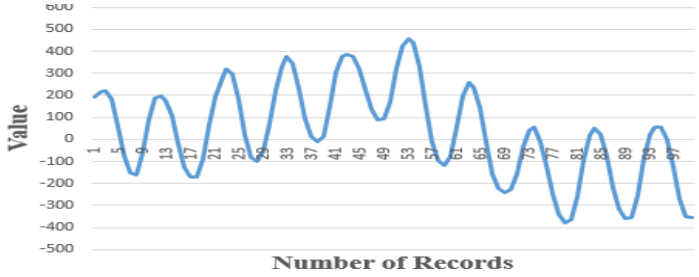
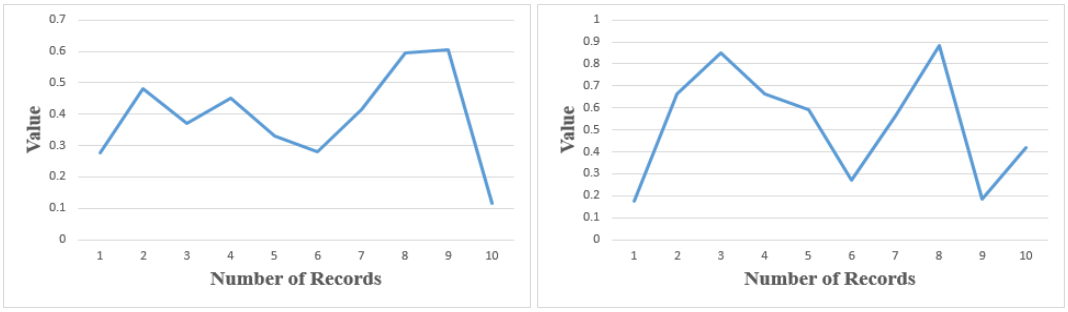
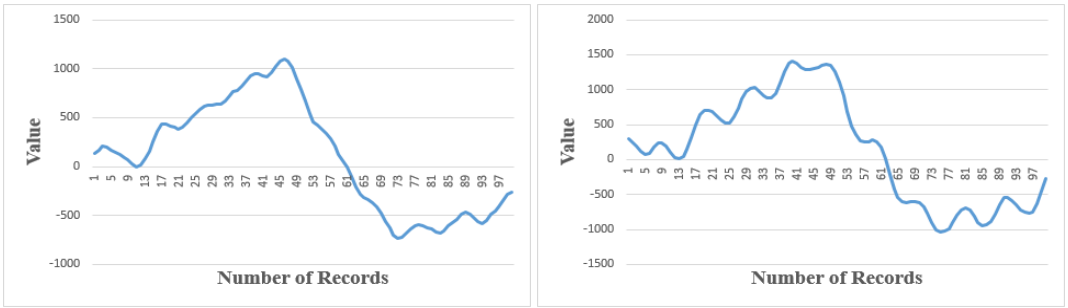


Figure 3.14: visible color "green" raw EEG



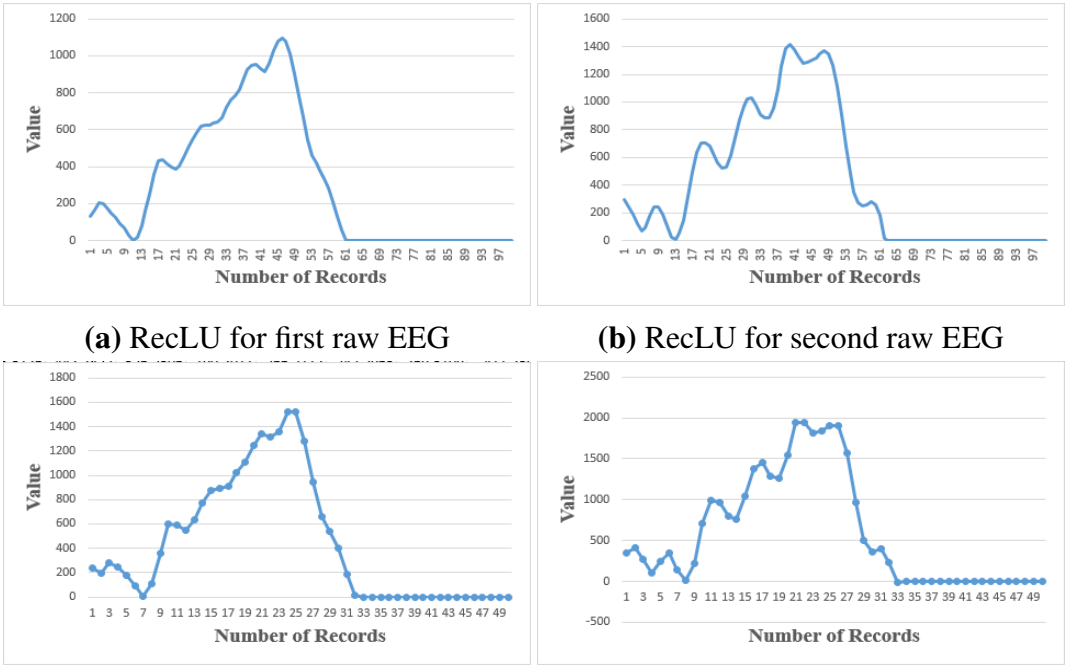
(a) first kernel **(b)** second kernel



(c) first feature map **(d)** second feature map

Figure 3.15: raw EEG, kernels, and feature maps

The input data tensor, size of kernels, number of kernels, and stride number remained as mentioned in the previous model.



(a) RecLU for first raw EEG **(b)** RecLU for second raw EEG

(c) DWT Symlet 2 for first raw EEG **(d)** DWT Symlet 2 for second raw EEG

Figure 3.16: RecLU, and Symlet 2

The architecture of the proposed model has been presented in Figure 3.17, and all modifications in the model are enclosed with blued dashed border.

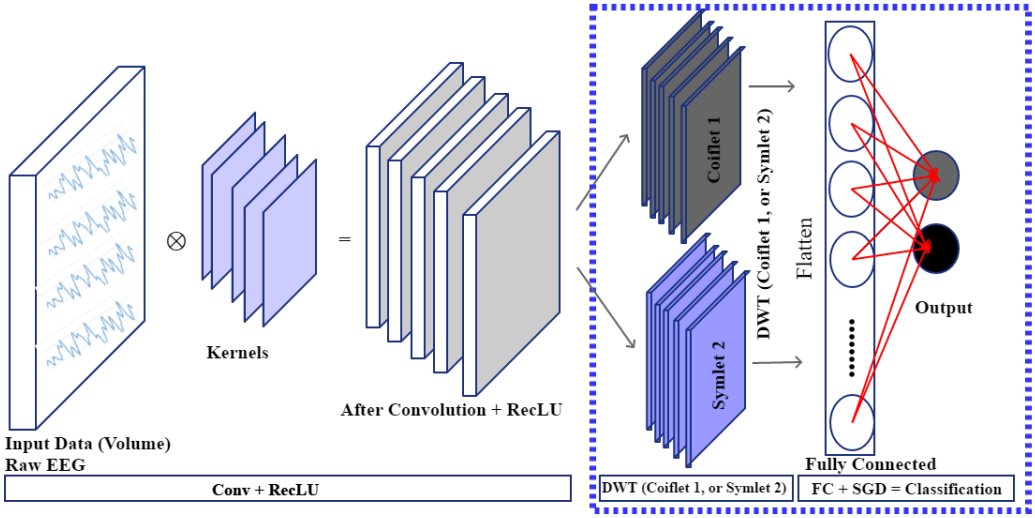


Figure 3.17: second proposed model architecture

The following pseudo code explains the algorithm that is been used in the proposed mode.

Input: *Tensor* ***RawEEGData*** *contains brainwave signals*

Vector ***Output*** *contains desired output*

Number of Kernels ***(NoKernels)***

Size of Kernels ***(SizeKernels)***

Number of input fed to DWT ***(InpDWT)***

Number of input layer's node ***(NoInp)***

Number of output layer's node

Learning rate value ***(*** η ***)***

Momentum value ***(*** γ ***)***

Step 1: Start

Step 2: *Declare variables Convolved, Rectified, MaxEpoch and Counter*

Step 3*: Initialize Kernels randomly between (0 and 1)*

Counter $\leftarrow$ ***0***

Step 4: For all *RawEEGData list* ϵ *dataset* ***do***

Convolved $\leftarrow$ *Calculate convolution between (kernels and list)*

End for Step 5: For all *Convolved features* ϵ *Convolved* ***do***

Rectified $\leftarrow$ *Math.Max(0, features)*

End for

Step 6: For all *Rectified row* ϵ *Rectified* ***do***

DWT $\leftarrow$ *Apply DWT (according to specified method) on each row.*

End for

Step 7: For all *DWT row* ϵ *DWT* ***do***

For j = 1 To NumOut

$$FFresult = \frac{1}{1 + \exp^{-\sum_{i=1}^{NumInput} Inp(i).w(i,j)}}$$

next j

End For

Step 8: *Calculate Log Loss between the obtained output an desired output*

$$L(t|y) = -\frac{1}{N}\sum_{i=1}^{N}[t.\log(y) + (1-t).log(1-y)]$$

Step 9: *Training Process Using SGD*

$$V_{k+1} = \gamma V_i + \eta.\frac{1}{N}.(t-y)X_{ij}$$

$$W = W - V_{k+1}$$

Step 10: *check iteration stopping criteria*

Counter ← Counter + 1

if LogLoss > Threshold or counter < MaxEpoch

increment counter by 1 goto step step 7

End*.*

3.11 Robot Design

The lego NXT has several humanoid designs. These designs depend on the application of the robot. In this work, based on expected tasks to be performed by the robot, a design has been proposed by combining two different lego NXT designs, to achieve the best humanoid robot design.

The proposed design was inspired from combining *M.O.R.P.H* and *Alpha Rex* designs. It is made up of three main parts: one controller box (mind box), three motors (actuators), and three sensors (one ultrasonic, two touch sensors), for the robot design see Figure 3.18.

Ultrasonic sensor is used to measure the distance between the robot and object that is facing, it can detect objects about 230 cm. It also has touch sensors to detect that the robot touched or pressed something.

The robot receives commands via Bluetooth, then according to the received command it takes an action. However, it is programmed like that to have the ability to protect itself from any collision. Before and during performing any action, the robot continuously checks its environment through sensors for any object or obstacle. In any circumstance, if the robot predict that (through sensors) unable to perform tasks, it stops from actions and ignore the received command. Figure 3.19 shows the flowchart of the program that runs on the robot. According to the received command, the robot perform four basic actions (forward, turn right, turn left, and backward). These actions with corresponding classified brainwave signals are listed in Table 3.4.

Table 3.4: actions performed by the Lego NXT robot and their corresponding brainwave signals.

Classified EEG Signal	**Performed Action**
Forward arrow	Go forward ⇑
Red Color	Go reverse ⇓
Right Arrow	Turn right ⇒
Green Color	Turn left ⇐

Figure 3.18: Lego NXT Humanoid Design

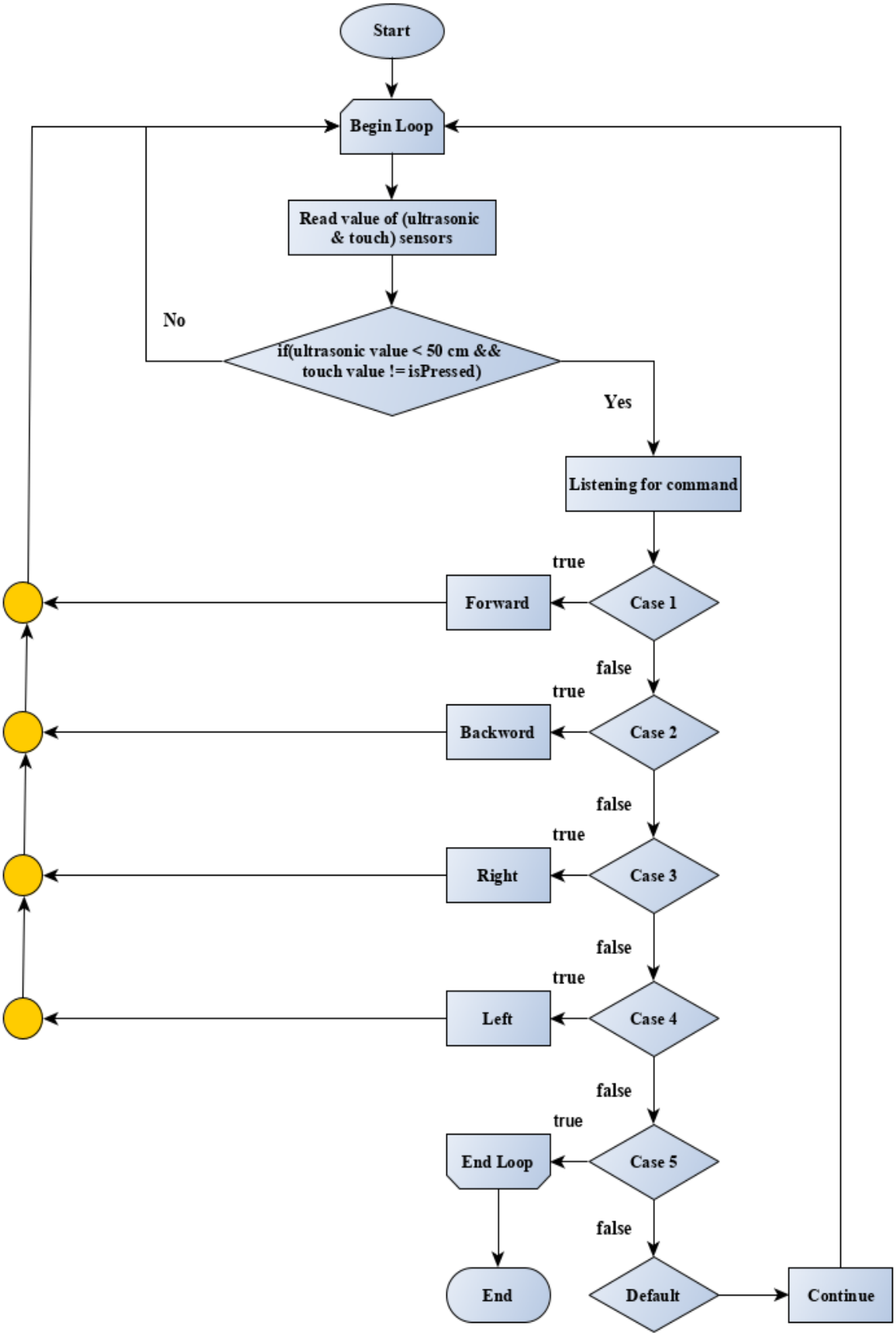


Figure 3.19: flowchart of robotic program

Chapter 4

Experiments and Results

4.1 Introduction

In order to evaluate the accuracy of both proposed models that are designed based on CNN, and the CNN model that have been described in the previous chapter, many experiments and different techniques have been performed. This chapter discusses the most important experimental results achieved, and also explains the impact of different parameters involved in the model on its results.

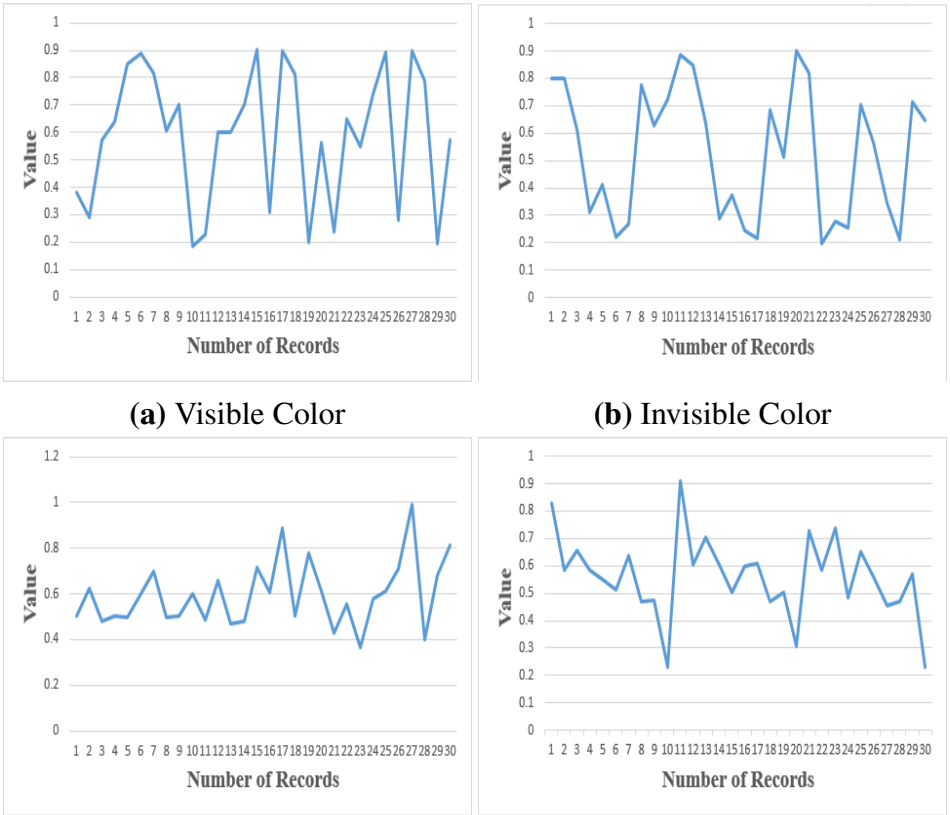
4.2 Selecting Kernels

Kernels (filters) have a direct impact on the classification results. However, unlike image processing there are no kernels (filter coefficients) available while working with EEG signals. Once the data is prepared and processed, convolution process must be performed in the data. For this, kernels (filters) are required. For building a set of kernels, there are three essential parameters that should be defined: width, height, and number of kernels (count). The format to write kernel architectures should be as [width - height, count]. In this work, three different kernels with different sizes have been used, each kernel has been tried with three different counts. In another word, nine different kernels have been experimented on each brainwave mode with respect to their categories, which are [1 - 5, 3], [1 - 5, 5], [1 - 5, 7], [1 - 10, 3], [1 -

10, 5], [1 - 10, 7], [1 - 15, 3], [1 - 15, 5], and [1 - 15, 7]. In the beginning the kernel values were randomly generated between 0 and 1. In order to achieve the best result, each model has been trained several times with different parameters. Each time the kernel values that could obtain a better result have been selected and used to repeat the training of the models using different parameters.

This figure below show the best obtained kernels with respect to their mode, based on the best achieved classification results.

The Figures 4.1a, 4.1b, 4.1c, and 4.1d represent charts of kernels coefficients for visible color, invisible color, visible arrow, and invisible arrow, respectively.



(a) Visible Color **(b)** Invisible Color

(c) Visible Arrow **(d)** Invisible Arrow

Figure 4.1: chart of different kernels coefficients respect to their mode

4.3 Test Material (EEG Signals)

In this research the dataset has been divided into two part:

1. Training samples: The entire prepared dataset was divided into a training set and a test set. The training set includes 80% of the dataset (including brainwave signals for six subjects, with two main categories (color and shape), in two different circumstances (visible and invisible), and five sessions for each). All models (CNN and both proposed models) have been trained on this amount of data from the dataset.

2. Testing samples: The remaining 20% of the dataset is used for testing the models. This amount of the dataset is used for evaluating classification accuracy of different models.

4.4 Results Based on Combined CNN with DWT (Coiflet 1, and Symlet 2) and SGD

As mentioned in section 3.8, the CNN consists of three main layers. The first layer consists of two operations. The first operation applying the convolution process between the kernels and input data (raw EEG signals) see section 2.8.2; the second operation is applying RecLU (see equation (2.3)) on each data point. The result provided by this layer is known as feature maps. In the second layer (see section 3.8) pooling applied on the feature maps. Then, the extracted features from second layer fed to the fully connected layer. Finally, softmax is applied (see equation (2.19)) to perform the classification.

In this proposed model, the pooling layer has been removed. Instead, two different methods of DWT (Coiflet 1, and Symlet 2) has been applied, individually. Then, SGD has been used as a trainer in the third layer (fully

connected layer). The proposed CNN using Coiflet 1, has been applied with two different input volumes (tensor) 10 seconds and 5 seconds of the dataset. The following three subsections present the results obtained from this model.

4.4.1 Combined CNN with DWT Coiflet 1 and SGD with 10 Seconds of Raw EEG

This section investigates the proposed model while the input tensor is set to 1, 5000, 1 (10 seconds). Then, the results are illustrated and the parameters used during the training and testing model are explained.

In this model, the convolution process between defined kernels and input data have been performed. Then, RecLU (see equation (2.3)) has been applied on each data point in the feature map. After that, DWT Coiflet on the generated data from convolution layer has been applied. Finally, the output of the second layer is flattened and it is fed to the fully connected layer. In this layer SGD has been chosen as a trainer and softmax (see equation (2.19)) has been used for predicting the output.

In the training stage, three different sizes of kernels ([1 - 5], [1 - 10], [1 - 15]) with three different filter numbers (3, 5, 7) have been investigated. The model has been trained with five different learning rates for each kernel. Different values of momentum and epoch have been examined. Table 4.1 shows the results achieved while applying the proposed model for visible color. In all of these tests the momentum value is 0.07, and epoch is 2000. Many different learning rates (η) have been examined based on the best accuracy attained, the desired learning rate has been selected for training the model.

Table 4.1: results achieved while the kernel size was [1 - 5] for visible color (combined CNN with DWT Coiflet 1)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 5, 3]	0.1	61%	0.22	1.0	0.88	0.02
	0.01	63%	0.66	0.63	0.64	0.008
	0.001	75%	0.72	0.76	0.74	0.2
	0.05	52%	0.33	0.54	0.41	0.00002
	0.005	69%	0.66	0.70	0.88	0.04
[1 - 5, 5]	0.1	58%	0.16	1.0	0.28	0.00001
	0.01	45%	0.36	0.44	0.4	0.005
	0.001	43%	0.3	0.4	0.34	0.2
	0.05	58%	0.7	0.56	0.62	0.00002
	0.005	46%	0.33	0.45	0.38	0.04
[1 - 5, 7]	0.1	60%	0.95	0.56	0.70	0.00001
	0.01	57%	0.47	0.58	0.4	0.006
	0.001	44%	0.4	0.43	0.34	0.2
	0.05	60%	0.71	0.58	0.62	0.0001
	0.005	54%	0.38	0.57	0.38	0.03

The second kernel size was [1 - 10]. In this case the training process took a longer time, due to the number of operations required to perform classification. All the results and scores attained are shown in Table 4.2.

Table 4.2: results achieved while the kernel size was [1 - 10] for visible color (combined CNN with DWT Coiflet 1)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 10, 3]	0.1	83%	0.83	0.83	0.83	0.00001
	0.01	83%	0.83	0.83	0.83	0.02
	0.001	75%	0.83	0.71	0.76	0.3
	0.05	83%	0.83	0.83	0.83	0.004
	0.005	92%	1.0	0.85	0.92	0.07
[1 - 10, 5]	0.1	82%	0.83	0.8	0.81	0.00001
	0.01	76%	0.76	0.76	0.76	0.004
	0.001	63%	0.5	0.68	0.57	0.2
	0.05	80%	0.76	0.82	0.79	0.001
	0.005	80%	0.76	0.82	0.79	0.01
[1 - 10, 7]	0.1	82%	0.90	0.77	0.83	0.001
	0.01	77%	0.85	0.73	0.79	0.04
	0.001	66%	0.64	0.67	0.65	0.3
	0.05	82%	0.95	0.75	0.84	0.0001
	0.005	64%	0.61	0.65	0.63	0.09

Table 4.2 shows that training (using kernel size [1 - 10]) has the ability to gain better results compared to the maximum result achieved while the kernel

size was [1 - 5]. The best obtained result in this case was 92%.

All results and scores acquired while the kernel size was set to [1 - 15] are presented in Table 4.3.

Table 4.3: results achieved while the kernel size was [1 - 15] for visible color (combined CNN with DWT Coiflet 1)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 15, 3]	0.1	58%	0.16	1.0	0.28	0.7
	0.01	63%	0.77	0.6	0.68	0.1
	0.001	72%	0.72	0.72	0.72	0.3
	0.05	58%	0.16	1.0	0.28	0.3
	0.005	63%	0.5	0.69	0.58	0.1
[1 - 15, 5]	0.1	66%	0.53	0.72	0.61	0.00008
	0.01	60%	0.73	0.57	0.64	0.02
	0.001	63%	0.46	0.7	0.56	0.08
	0.05	58%	0.16	1.0	0.28	0.1
	0.005	70%	0.76	0.67	0.71	0.1
[1 - 15, 7]	0.1	63%	0.78	0.6	0.68	0.001
	0.01	63%	0.76	0.60	0.67	0.06
	0.001	58%	0.42	0.62	0.50	0.4
	0.05	63%	0.78	0.6	0.68	0.005
	0.005	65%	0.76	0.62	0.68	0.1

The best classification result in this model was %92 while the kernel was set to [1 - 10, 3], the confusion matrix has been shown in Table 4.4.

Table 4.4: confusion matrix for visible color (combined CNN with DWT Coiflet 1)

Total Data Point = 60000	**Actual Red**	**Actual Green**
Predict Red	30000 *(TP)*	5000 *(FP)*
Predict Green	0 *(FN)*	25000 *(TN)*

Furthermore, kernels with the same architecture and the same learning rates are used for training purposes for invisible color. The momentum and epoch values are set to 0.005 and 2500, respectively. The results for invisible color in the proposed model (combined CNN with DWT Coiflet 1) have been shown in Table 4.5 while the kernel size was [1 - 5].

Table 4.5: results achieved while the kernel size was [1 - 5] for invisible color (combined CNN with DWT Coiflet 1)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 5, 3]	0.1	52%	0.66	0.52	0.58	0.09
	0.01	52%	0.55	0.52	0.54	0.004
	0.001	47%	0.61	0.47	0.53	0.1
	0.05	55%	0.66	0.54	0.6	0.00002
	0.005	50%	0.61	0.5	0.55	0.02
[1 - 5, 5]	0.1	51%	0.2	0.54	0.29	0.00001
	0.01	60%	0.7	0.58	0.63	0.004
	0.001	56%	0.63	0.55	0.59	0.1
	0.05	51%	0.73	0.51	0.6	0.00002
	0.005	63%	0.73	0.61	0.66	0.02
[1 - 5, 7]	0.1	46%	0.16	0.41	0.23	0.0001
	0.01	50%	0.5	0.5	0.5	0.005
	0.001	46%	0.45	0.46	0.45	0.1
	0.05	51%	0.33	0.51	0.4	0.00001
	0.005	50%	0.52	0.5	0.51	0.03

Based on the results presented in Table 4.5, the best accuracy rate was 63% for kernel size [1 - 5].

Table 4.6 shows the results for invisible color classification, while kernel size is set to [1 - 10].

Table 4.6: results achieved while the kernel size was [1 - 10] for invisible color (combined CNN with DWT Coiflet 1)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 10, 3]	0.1	77%	0.77	0.77	0.77	0.000007
	0.01	83%	0.83	0.83	0.83	0.01
	0.001	69%	0.66	0.7	0.68	0.2
	0.05	75%	0.66	0.8	0.72	0.005
	0.005	83%	0.83	0.83	0.83	0.05
[1 - 10, 5]	0.1	58%	0.53	0.59	0.56	0.00001
	0.01	60%	0.6	0.6	0.6	0.01
	0.001	58%	0.43	0.61	0.5	0.03
	0.05	56%	0.5	0.57	0.53	0.0005
	0.005	65%	0.63	0.65	0.64	0.07
[1 - 10, 7]	0.1	51%	0.26	0.52	0.34	0.00001
	0.01	47%	0.5	0.47	0.48	0.01
	0.001	46%	0.35	0.45	0.4	0.3
	0.05	52%	0.4	0.53	0.45	0.0005
	0.005	50%	0.38	0.5	0.43	0.06

The final filter size applied on the invisible color was [1 - 15], Table 4.7

presents the results for this case.

Table 4.7: results achieved while the kernel size was [1 - 15] for invisible color (combined CNN with DWT Coiflet 1)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 15, 3]	0.1	50%	0.38	0.5	0.43	0.004
	0.01	52%	0.5	0.52	0.51	0.004
	0.001	41%	0.5	0.42	0.46	0.3
	0.05	50%	0.38	0.5	0.43	0.002
	0.005	58%	0.5	0.6	0.54	0.1
[1 - 15, 5]	0.1	51%	0.56	0.51	0.53	0.00006
	0.01	58%	0.6	0.58	0.59	0.04
	0.001	66%	0.53	0.72	0.61	0.3
	0.05	51%	0.56	0.51	0.53	0.004
	0.005	60%	0.56	0.6	0.58	0.1
[1 - 15, 7]	0.1	53%	0.54	0.53	0.54	0.00001
	0.01	55%	0.47	0.57	0.51	0.03
	0.001	54%	0.28	0.6	0.38	0.3
	0.05	53%	0.54	0.53	0.54	0.001
	0.005	61%	0.47	0.66	0.55	0.1

For classifying invisible color using this proposed model, the best accuracy result achieved was 83% when the kernel size was set to [1 - 10, 3].

Table 4.8 shows the confusion matrix for the best classification result of invisible color.

Table 4.8: confusion matrix for invisible color (combined CNN with DWT Coiflet 1)

Total Data Point = 60000	**Actual Red**	**Actual Green**
Predict Red	25000 *(TP)*	5000 *(FP)*
Predict Green	5000 *(FN)*	25000 *(TN)*

The proposed model (combined CNN with Coiflet 1) was also applied on the visible arrow with the same learning rate and the same kernels size. Table 4.9 shows the results of applying this proposed model while the kernel size was [1 - 5]. The momentum and epoch were set to 0.005, and 2000 respectively.

Table 4.9: results achieved while the kernel size is [1 - 5] for visible arrow (combined CNN with DWT Coiflet 1)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 5, 3]	0.1	58%	0.33	0.66	0.44	0.00001
	0.01	69%	0.72	0.68	0.7	0.008
	0.001	66%	0.66	0.66	0.66	0.2
	0.05	63%	0.55	0.66	0.6	0.03
	0.005	63%	0.61	0.64	0.62	0.00005
[1 - 5, 5]	0.1	70%	0.86	0.65	0.74	0.00001
	0.01	70%	0.83	0.65	0.73	0.008
	0.001	75%	0.7	0.77	0.73	0.2
	0.05	68%	0.83	0.64	0.72	0.00002
	0.005	76%	0.8	0.75	0.77	0.03
[1 - 5, 7]	0.1	71%	0.83	0.67	0.74	0.00001
	0.01	73%	0.8	0.7	0.75	0.01
	0.001	77%	0.71	0.81	0.75	0.2
	0.05	65%	0.66	0.65	0.65	0.00001
	0.005	79%	0.83	0.77	0.80	0.2

The best accuracy rate for visible arrow based on Table 4.9 was 79%, while the learning rate was 0.005, and the kernel size was [1 - 5] with seven filters. The second kernel size was [1 - 10]. Table 4.10 shows the results of visible arrow with different kernel counts.

Table 4.10: results achieved while the kernel size was [1 - 10] for visible arrow (combined CNN with DWT Coiflet 1)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 10, 3]	0.1	83%	1.0	0.75	0.85	0.00008
	0.01	83%	1.0	0.75	0.85	0.0004
	0.001	83%	0.83	0.83	0.83	0.02
	0.05	83%	1.0	0.75	0.85	0.3
	0.005	92%	1.0	0.85	0.92	0.05
[1 - 10, 5]	0.1	86%	0.96	0.8	0.87	0.00001
	0.01	86%	0.93	0.82	0.87	0.03
	0.001	81%	0.7	0.91	0.79	0.4
	0.05	86%	0.96	0.80	0.87	0.001
	0.005	83%	0.83	0.83	0.83	0.1
[1 - 10, 7]	0.1	86%	0.97	0.8	0.88	0.00004
	0.01	88%	1.0	0.8	0.89	0.002
	0.001	79%	0.83	0.77	0.8	0.3
	0.05	86%	1.0	0.79	0.88	0.03
	0.005	92%	1.0	0.85	0.92	0.1

As shown in Table 4.10, the best accuracy rate that can be achieved for

visible arrow was 92%, while using the kernel size [1 -10].

Table 4.11 shows the results of visible arrow while the kernel size was set to [1 - 15] with different count filters.

Table 4.11: results achieved while the kernel size was [1 - 15] for visible arrow (combined CNN with DWT Coiflet 1)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 15, 3]	0.1	52%	0.05	1.0	0.28	0.1
	0.01	75%	0.77	0.73	0.75	0.2
	0.001	75%	0.55	0.9	0.68	0.4
	0.05	52%	0.05	1.0	0.1	0.2
	0.005	58%	0.16	1.0	0.28	0.2
[1 - 15, 5]	0.1	81%	0.96	0.74	0.84	0.00003
	0.01	80%	0.9	0.75	0.81	0.04
	0.001	75%	0.8	0.72	0.76	0.3
	0.05	81%	0.96	0.74	0.84	0.003
	0.005	75%	0.8	0.72	0.76	0.1
[1 - 15, 7]	0.1	78%	0.88	0.74	0.8	0.001
	0.01	79%	0.9	0.74	0.81	0.04
	0.001	80%	0.78	0.82	0.8	0.3
	0.05	78%	0.88	0.74	0.8	0.003
	0.005	82%	0.9	0.77	0.83	0.3

Table 4.12 shows the confusion matrix while applying this model on the (arrow) shape category for the visible mode. The best classification result attained was 92%, with the kernel design [1 - 10, 3].

Table 4.12: confusion matrix for visible arrow (combined CNN with DWT Coiflet 1)

Total Data Point = 60000	**Actual Red**	**Actual Green**
Predict Red	30000 *(TP)*	5000 *(FP)*
Predict Green	0 *(FN)*	25000 *(TN)*

Table 4.13 shows the results related to invisible arrow when the proposed model was applied with different kernels sizes and kernel numbers. Many experiments with different parameters have been done to perform the classification process in an efficient and accurate way. The momentum and epoch value were set to 0.05 and 5000, respectively.

Table 4.13: results achieved while the kernel size was [1 - 5] for invisible arrow (combined CNN with DWT Coiflet 1)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 5, 3]	0.1	50%	0.16	0.5	0.25	0.00004
	0.01	58%	0.66	0.57	0.61	0.004
	0.001	47%	0.44	0.47	0.45	0.08
	0.05	63%	0.61	0.64	0.62	0.00002
	0.005	50%	0.5	0.5	0.5	0.01
[1 - 5, 5]	0.1	43%	0.033	0.16	0.05	0.000002
	0.01	71%	0.6	0.78	0.67	0.004
	0.001	55%	0.36	0.57	0.44	0.007
	0.05	55%	0.4	0.57	0.47	0.00001
	0.005	66%	0.5	0.75	0.6	0.01
[1 - 5, 7]	0.1	58%	0.4	0.62	0.49	0.000004
	0.01	71%	0.66	0.73	0.7	0.004
	0.001	69%	0.59	0.73	0.65	0.08
	0.05	63%	0.66	0.62	0.64	0.00003
	0.005	67%	0.59	0.71	0.64	0.01

Based on the results provided in Table 4.13 the maximum accuracy rate gained was 71%.

Table 4.14 shows the results and scores attained for invisible arrow while the kernel size was set to [1, 10] with different kernel counts.

Table 4.14: results achieved while the kernel size was [1 - 10] for invisible arrow (combined CNN with DWT Coiflet 1)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 10, 3]	0.1	83%	0.83	0.83	0.83	0.00005
	0.01	83%	0.83	0.83	0.83	0.01
	0.001	80%	0.78	0.83	0.81	0.1
	0.05	83%	0.83	0.83	0.83	0.00003
	0.005	83%	0.83	0.83	0.83	0.03
[1 - 10, 5]	0.1	71%	0.66	0.74	0.7	0.00005
	0.01	73%	0.7	0.75	0.72	0.01
	0.001	75%	0.7	0.77	0.73	0.1
	0.05	71%	0.66	0.74	0.7	0.0004
	0.005	75%	0.7	0.77	0.73	0.03
[1 - 10, 7]	0.1	72%	0.66	0.75	0.7	0.00003
	0.01	63%	0.66	0.62	0.64	0.01
	0.001	55%	0.69	0.54	0.61	0.1
	0.05	72%	0.66	0.75	0.7	0.0003
	0.005	58%	0.66	0.57	0.61	0.04

As shown in Table 4.14, the best accuracy rate for invisible color when the

kernel size was [1 - 10] was 83%.

The last kernel size was [1 - 15], this kernel size with different numbers have been applied on the invisible arrow. Table 4.15 shows the results attained in this category.

Table 4.15: results achieved while the kernel size was [1 - 15] for invisible arrow (combined CNN with DWT Coiflet 1)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 15, 3]	0.1	63%	0.66	0.63	0.64	0.00003
	0.01	63%	0.66	0.63	0.64	0.02
	0.001	61%	0.77	0.58	0.66	0.2
	0.05	63%	0.66	0.63	0.64	0.003
	0.005	55%	0.66	0.54	0.6	0.05
[1 - 15, 5]	0.1	65%	0.66	0.64	0.65	0.00004
	0.01	56%	0.66	0.55	0.60	0.02
	0.001	51%	0.66	0.51	0.57	0.2
	0.05	66%	0.66	0.66	0.66	0.0006
	0.005	51%	0.66	0.51	0.57	0.06
[1 - 15, 7]	0.1	60%	0.66	0.59	0.62	0.00001
	0.01	60%	0.66	0.59	0.62	0.00
	0.001	57%	0.61	0.56	0.59	0.2
	0.05	61%	0.66	0.60	0.63	0.0004
	0.005	61%	0.66	0.6	0.63	0.05

Table 4.16 presents the confusion matrix for the best achieved results.

Table 4.16: confusion matrix for invisible arrow (combined CNN with DWT Coiflet 1)

Total Data Point = 60000	**Actual Red**	**Actual Green**
Predict Red	25000 *(TP)*	5000 *(FP)*
Predict Green	5000 *(FN)*	25000 *(TN)*

To sum up, according to the attained results, after applying the proposed model (combined CNN with Coiflet 1) on the experimental dataset, while input tensor was 1, 5000, 1 (10 seconds), the best accuracy rate for both visible color and shape was 92%. However, the best accuracy rate for both invisible shapes was 83%. These results have been gained while the kernel size was set to [1 - 10] and kernel count was set to 3.

4.4.2 Combined CNN With DWT Coiflet 1 and SGD with 5 Seconds of Raw EEG

This proposed model is also consist of three main layers: convolution, DWT Coiflet 1, and fully connected layer. All results attained with parameters used for training this proposed model (combined CNN with DWT Coiflet 1 and SGD) have been discussed in this section. According to the results in the previous section, the kernel can obtain best classification result was [1 - 10, 3]. Thus, for training the model the kernel was set [1 - 10, 3]. As shown in Table 4.17, different parameters have been used to train the model, epoch value is 5000, and momentum values are set to 0.01, 0.07 for both visible and invisible colors, respectively. In order to reach the best classification result, the momentum value was set to 0.02 for both visible and invisible arrows.

Table 4.17: results achieved while the tensor size was [1, 2500, 1] for color and arrow categories

Cate.	Mode	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
Color	**Visible**	0.1	87%	0.83	0.9	0.86	0.001
		0.01	76%	0.83	0.73	0.77	0.05
		0.001	70%	0.83	0.66	0.74	0.1
		0.05	83%	0.66	1.0	0.8	0.005
		0.005	70%	0.83	0.66	0.74	0.3
	Invisible	0.1	59%	0.44	0.64	0.52	0.00009
		0.01	68%	0.64	0.7	0.65	0.1
		0.001	68%	0.58	0.72	0.64	0.01
		0.05	59%	0.44	0.64	0.52	0.0005
		0.005	70%	0.66	0.72	0.68	0.02
Arrow	**Visible**	0.1	50%	0.58	0.5	0.53	0.001
		0.01	70%	0.66	0.72	0.69	0.01
		0.001	79%	0.75	0.81	0.78	0.1
		0.05	51%	0.58	0.51	0.54	0.00
		0.005	72%	0.66	0.75	0.7	0.02
	Invisible	0.1	58%	0.41	0.62	0.5	0.003
		0.01	61%	0.63	0.6	0.62	0.06
		0.001	63%	0.75	0.61	0.67	0.3
		0.05	58%	0.41	0.62	0.5	0.01
		0.005	66%	0.75	0.64	0.69	0.1

Confusion matrices for the best accuracy rate achieved in this proposed model while input volume was set to 1, 2500, 1 are presented in Table 4.18 for visible and invisible colors.

Table 4.18: confusion matrices for visible, and invisible colors category (CNN)

Visible Color		
Data Point = 60000	**Actual Red**	**Actual Green**
Predict Red	25000 *(TP)*	2500 *(FP)*
Predict Green	5000 *(FN)*	27500 *(TN)*

Invisible Color		
Data Point = 60000	**Actual Red**	**Actual Green**
Predict Red	20000 *(TP)*	7500 *(FP)*
Predict Green	10000 *(FN)*	22500 *(TN)*

Confusion matrices of the best attained result in this category for both visible and invisible arrows (shapes) have been presented in Table 4.19.

Table 4.19: confusion matrices for visible, and invisible arrow category (CNN)

Visible Arrow		
Data Point = 60000	**Actual For.**	**Actual Right**
Predict For.	22500 *(TP)*	5000 *(FP)*
Predict Right	7500 *(FN)*	25000 *(TN)*

Invisible Arrow		
Data Point = 60000	**Actual For.**	**Actual Right**
Predict For.	22500 *(TP)*	12500 *(FP)*
Predict Right	7500 *(FN)*	17500 *(TN)*

Figure 4.2 provides a comparison between the best accuracy rates attained of the proposed model (combined CNN with Coiflet 1) while the input was 10 secs, and 5 secs.

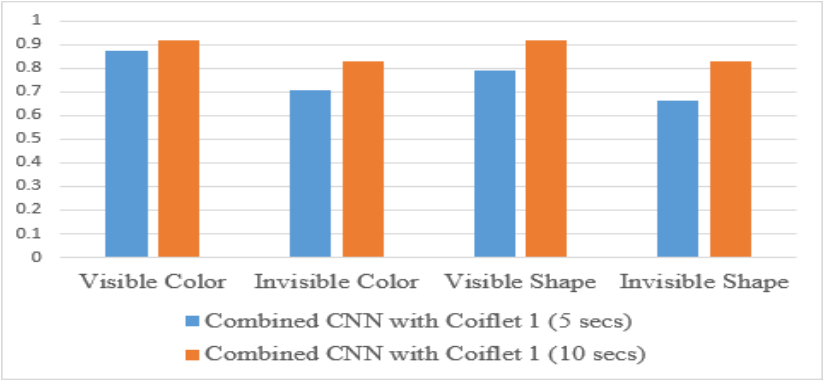


Figure 4.2: accuracy rate comparison between combined CNN with Coiflet 1 (10 secs), and combined CNN with Coiflet 1 (5 secs)

4.4.3 Combined CNN with DWT Symlet 2 and SGD with 10 Seconds of Raw EEG

In this section, the results attained from combined CNN with Symlet 2 have been explained. The input tensor has been set to 1, 5000, 1 based on the best result achieved in the second proposed model while using Coiflet 1. In this model, after applying convolution, and RecLU (see equation (2.3))

on the input data, the generated data has been fed to the DWT Symlet 2 to extract the most interesting features. These features have been fed to the fully connected layer and the SGD has been used as trainer. Finally, the softmax function (see equation (2.19)) have been used to predict the result.

As mentioned in section 4.2, nine different kernels with different sizes have been used for training the model. Each category with respect to their mode has been trained with 5 different learning rates and nine different filters.

Table 4.20 shows the accuracy results attained while kernel size was [1 - 5] with different learning rates. The momentum and epoch value are set to 0.005, and 2000 respectively during the training stage. Moreover, in order to achieve better results, the kernel size of [1 - 15, 5], and [1 - 15, 7] have been used to train model with epoch 2500. The second kernel size used for

Table 4.20: results achieved while the kernel size was [1 - 5] for visible color (combined CNN with DWT Symlet 2)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 5, 3]	0.1	58%	0.16	1.0	0.28	0.00002
	0.01	52%	0.5	0.52	0.51	0.006
	0.001	69%	0.55	0.76	0.64	0.2
	0.05	66%	0.83	0.62	0.71	0.00002
	0.005	58%	0.55	0.58	0.57	0.03
[1 - 5, 5]	0.1	71%	1.0	0.63	0.77	0.00001
	0.01	56%	0.46	0.58	0.51	0.00001
	0.001	55%	0.46	0.56	0.5	0.2
	0.05	56%	0.5	0.57	0.53	0.006
	0.005	55%	0.43	0.56	0.49	0.2
[1 - 5, 7]	0.1	55%	0.5	0.56	0.53	0.000001
	0.01	48%	0.42	0.48	0.45	0.006
	0.001	48%	0.42	0.48	0.45	0.006
	0.05	50%	0.33	0.5	0.4	0.00002
	0.005	47%	0.4	0.47	0.43	0.03

training the model was [1 - 10], in this case, the training process required a longer time compared to the previous case due to the number of operations required to be done. Table 4.21 provides the results achieved for different kernel numbers.

Table 4.21: results achieved while the kernel size was [1 - 10] for visible color (combined CNN with DWT Symlet 2)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 10, 3]	0.1	69%	0.88	0.64	0.74	0.00005
	0.01	83%	1.0	0.75	0.857	0.00
	0.001	77%	0.88	0.72	0.8	0.3
	0.05	66%	0.83	0.62	0.71	0.004
	0.005	77%	0.88	0.72	0.8	0.09
[1 - 10, 5]	0.1	75%	1.0	0.66	0.8	0.00003
	0.01	73%	0.69	0.65	0.78	0.04
	0.001	73%	0.9	0.67	0.77	0.4
	0.05	75%	1.0	0.66	0.8	0.001
	0.005	76%	0.93	0.7	0.8	0.1
[1 - 10, 7]	0.1	75%	0.83	0.71	0.76	0.0001
	0.01	72%	0.8	0.69	0.74	0.04
	0.001	61%	0.57	0.63	0.6	0.3
	0.05	76%	0.88	0.71	0.78	0.001
	0.005	65%	0.64	0.65	0.65	0.1

Table 4.21 shows that the proposed model (combined CNN with Symlet 2) has the ability to achieve better accuracy rate. When applied on the dataset using the kernel size [1 - 10] with rate was 83%.

Table 4.22 provides the accuracy results attained while the kernel size was [1 - 15].

Table 4.22: results achieved while kernel size was [1 - 15] for visible color (combined CNN with DWT Symlet 2)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 15, 3]	0.1	63%	0.77	0.60	0.68	0.00003
	0.01	61%	0.72	0.59	0.65	0.05
	0.001	72%	0.72	0.72	0.72	0.4
	0.05	58%	0.66	0.57	0.61	0.001
	0.005	63%	0.72	0.61	0.66	0.1
[1 - 15, 5]	0.1	56%	0.66	0.57	0.61	0.00004
	0.01	56%	0.63	0.55	0.59	0.04
	0.001	56%	0.46	0.58	0.51	0.3
	0.05	58%	0.66	0.57	0.61	0.001
	0.005	61%	0.7	0.6	0.64	0.1
[1 - 15, 7]	0.1	60%	0.69	0.59	0.63	0.00008
	0.01	61%	0.71	0.6	0.65	0.05
	0.001	69%	0.57	0.75	0.64	0.4
	0.05	60%	0.69	0.59	0.63	0.002
	0.005	61%	0.69	0.6	0.64	0.1

The maximum accuracy result according to Table 4.20, 4.21, and 4.22 for visible color using combined CNN with DWT Symlet 2 was 83%, this result was attained while the kernel was [1 - 10, 3].

Table 4.23 provides a confusion matrix of best attained results for visible color.

Table 4.23: confusion matrix for visible color (combined CNN with DWT Symlet 2)

Total Data Point = 60000	**Actual Red**	**Actual Green**
Predict Red	30000 *(TP)*	10000 *(FP)*
Predict Green	0 *(FN)*	20000 *(TN)*

The same kernel design with the same filter numbers and the same learning rate was used for training the model on the experimental dataset for invisible color.

Table 4.24, presents the results achieved while the kernel size was [1 - 5] with 5 different learning rates.

Table 4.24: results achieved while kernel size was [1 - 5] for invisible color (combined CNN with DWT Symlet 2)

Kernel	**L.R η**	**Accuracy**	**Recall**	**Precision**	**F-Measure**	**Log Loss**
[1 - 5, 3]	0.1	44%	0.16	0.37	0.23	0.00002
	0.01	50%	0.5	0.5	0.5	0.004
	0.001	47%	0.5	0.47	0.48	0.2
	0.05	44%	0.16	0.37	0.23	0.01
	0.005	50%	0.5	0.5	0.5	0.03
[1 - 5, 5]	0.1	55%	0.26	0.61	0.37	0.000001
	0.01	61%	0.73	0.59	0.65	0.005
	0.001	66%	0.83	0.62	0.71	0.2
	0.05	60%	0.8	0.57	0.66	0.00002
	0.005	66%	0.8	0.63	0.7	0.03
[1 - 5, 7]	0.1	50%	0.8	0.5	0.61	0.00001
	0.01	57%	0.52	0.57	0.55	0.00
	0.001	41%	0.3	0.39	0.34	0.2
	0.05	50%	0.8	0.5	0.61	0.00002
	0.005	51%	0.52	0.51	0.51	0.03

Table 4.25 shows the accuracy result for invisible color while the kernel size was set to [1 - 10] for 3 different kernel numbers and 5 learning rates.

Table 4.25: results achieved while the kernel size was [1 - 10] for invisible color (combined CNN with DWT Symlet 2)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 10, 3]	0.1	47%	0.22	0.44	0.29	0.00002
	0.01	44%	0.5	0.45	0.47	0.01
	0.001	52%	0.55	0.52	0.54	0.3
	0.05	41%	0.33	0.4	0.36	0.00007
	0.005	58%	0.61	0.57	0.59	0.07
[1 - 10, 5]	0.1	56%	0.3	0.64	0.4	0.00001
	0.01	51%	0.56	0.51	0.53	0.02
	0.001	48%	0.43	0.48	0.45	0.3
	0.05	53%	0.46	0.53	0.5	0.0001
	0.005	50%	0.53	0.5	0.51	0.0005
[1 - 10, 7]	0.1	47%	0.16	0.43	0.24	0.00003
	0.01	46%	0.5	0.46	0.48	0.01
	0.001	46%	0.47	0.46	0.47	0.3
	0.05	51%	0.35	0.51	0.42	0.00009
	0.005	48%	0.47	0.48	0.48	0.07

Table 4.26 provides the result of the classification of brainwave signals based on combined CNN with Symlet 2 while the kernel size was [1 - 15] with three different kernel numbers and 5 different learning rates.

Table 4.26: results achieved while the kernel size was [1 - 15] for invisible color (combined CNN with DWT Symlet 2)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 15, 3]	0.1	47%	0.5	0.47	0.48	0.0006
	0.01	50%	0.5	0.5	0.5	0.03
	0.001	55%	0.66	0.54	0.6	0.3
	0.05	50%	0.5	0.5	0.5	0.002
	0.005	52%	0.55	0.52	0.54	0.005
[1 - 15, 5]	0.1	50%	0.6	0.5	0.54	0.0001
	0.01	51%	0.5	0.51	0.5	0.04
	0.001	56%	0.43	0.59	0.5	0.3
	0.05	53%	0.6	0.52	0.56	0.001
	0.005	56%	0.5	0.57	0.53	0.1
[1 - 15, 7]	0.1	50%	0.52	0.5	0.51	0.0003
	0.01	44%	0.38	0.43	0.4	0.004
	0.001	47%	0.23	0.45	0.31	0.4
	0.05	50%	0.52	0.5	0.51	0.0008
	0.005	44%	0.26	0.4	0.31	0.1

Table 4.27 shows the confusion matrix for the best accuracy result for invisible color. The best result attained was 66%

Table 4.27: confusion matrix for invisible color (combined CNN with DWT Symlet 2)

Total Data Point = 60000	**Actual Red**	**Actual Green**
Predict Red	25000 *(TP)*	15000 *(FP)*
Predict Green	5000 *(FN)*	15000 *(TN)*

Many experiments have been performed with different parameters to achieve the best classification results. In this model, the same kernel size with the same leaning rate have been applied on both visible and invisible arrows although, momentum and epoch are set to 0.005, and 2000 respectively. However, to attain best result the kernels with size [1 - 15] have also been trained with 2500 epoch.

Table 4.28 illustrates the results attained in visible arrow mode.

Table 4.28: results achieved while the kernel size was [1 - 5] for visible arrow (combined CNN with DWT Symlet 2)

Kernel	**L.R** η	**Accuracy**	**Recall**	**Precision**	**F-Measure**	**Log Loss**
[1 - 5, 3]	0.1	66%	0.83	0.62	0.71	0.04
	0.01	63%	0.77	0.6	0.68	0.008
	0.001	58%	0.61	0.57	0.59	0.2
	0.05	72%	0.83	0.68	0.75	0.00004
	0.005	61%	0.66	0.6	0.63	0.03
[1 - 5, 5]	0.1	58%	0.33	0.66	0.44	0.00001
	0.01	78%	0.9	0.72	0.8	0.008
	0.001	63%	0.66	0.62	0.64	0.2
	0.05	81%	0.83	0.8	0.81	0.00003
	0.005	83%	0.86	0.81	0.83	0.03
[1 - 5, 7]	0.1	55%	0.28	0.63	0.39	0.00001
	0.01	78%	0.38	0.43	0.4	0.007
	0.001	83%	0.95	0.76	0.85	0.2
	0.05	70%	0.57	0.77	0.65	0.00002
	0.005	84%	0.97	0.77	0.86	0.003

Table 4.29, provides the results attained while the kernel size was [1 - 10)] in three different filter numbers and five leaning rates.

Table 4.29: results achieved while the kernel size was [1 - 10] for visible arrow (combined CNN with DWT Symlet 2)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 10, 3]	0.1	92%	1.0	0.85	0.92	0.04
	0.01	86%	1.0	0.78	0.87	0.04
	0.001	83%	0.83	0.83	0.83	0.3
	0.05	92%	1.0	0.85	0.92	0.0007
	0.005	86%	1.0	0.78	0.87	0.1
[1 - 10, 5]	0.1	78%	0.9	0.72	0.8	0.00003
	0.01	80%	0.9	0.75	0.81	0.04
	0.001	76%	0.66	0.83	0.74	0.4
	0.05	78%	0.9	0.72	0.8	0.009
	0.005	83%	0.86	0.81	0.83	0.1
[1 - 10, 7]	0.1	84%	1.0	0.76	0.86	0.0003
	0.01	89%	1.0	0.82	0.9	0.04
	0.001	75%	0.78	0.73	0.75	0.3
	0.05	84%	1.0	0.76	0.86	0.002
	0.005	92%	1.0	0.85	0.92	0.1

The final kernel size was [1 - 15]. This kernel size has been used with different learning rates, and different filter numbers for training the proposed model for classifying visible arrows, see Table 4.30.

Table 4.30: results achieved while the kernel size was [1 - 15] for visible arrow (combined CNN with DWT Symlet 2)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 15, 3]	0.1	80%	0.88	0.76	0.82	0.00002
	0.01	83%	0.88	0.8	0.84	0.04
	0.001	75%	0.61	0.84	0.7	0.4
	0.05	83%	0.88	0.8	0.84	0.0008
	0.005	80%	0.83	0.78	0.81	0.1
[1 - 15, 5]	0.1	80%	0.93	0.73	0.82	0.002
	0.01	76%	0.86	0.72	0.78	0.05
	0.001	66%	0.73	0.64	0.68	0.4
	0.05	80%	0.93	0.73	0.82	0.006
	0.005	75%	0.83	0.71	0.76	0.1
[1 - 15, 7]	0.1	82%	0.95	0.75	0.84	0.006
	0.01	82%	0.92	0.76	0.83	0.05
	0.001	76%	0.64	0.84	0.72	0.4
	0.05	82%	0.95	0.75	0.84	0.007
	0.005	80%	0.88	0.77	0.82	0.1

Confusion matrix for the best accuracy result has been shown in Table 4.31, the best accuracy result attained for the visible arrow based on combined

CNN with DWT Symlet 2 was 92%.

Table 4.31: confusion matrix for visible arrow (combined CNN with DWT Symlet 2)

Total Data Point = 60000	**Actual Forward**	**Actual Right**
Predict Forward	30000 *(TP)*	5000 *(FP)*
Predict Right	0 *(FN)*	25000 *(TN)*

The same kernel design with the same learning rates have been used to train this proposed model for classifying invisible arrows (shapes). Table 4.32 presents results while applying this model with kernel size [1 - 5] in three different sizes with respect to five learning rates.

Table 4.32: results achieved while the kernel size was [1 - 5] for invisible arrow (combined CNN with DWT Symlet 2)

Kernel	**L.R** η	**Accuracy**	**Recall**	**Precision**	**F-Measure**	**Log Loss**
[1 - 5, 3]	0.1	47%	0.27	0.45	0.34	0.00009
	0.01	44%	0.5	0.45	0.47	0.007
	0.001	41%	0.33	0.4	0.36	0.1
	0.05	38%	0.44	0.4	0.42	0.00004
	0.005	47%	0.38	0.46	0.42	0.02
[1 - 5, 5]	0.1	51%	0.33	0.52	0.40	0.000001
	0.01	53%	0.46	0.53	0.5	0.007
	0.001	55%	0.26	0.61	0.37	0.1
	0.05	43%	0.46	0.43	0.45	0.00002
	0.005	60%	0.4	0.66	0.5	0.02
[1 - 5, 7]	0.1	46%	0.11	0.38	0.18	0.000009
	0.01	51%	0.71	0.6	0.65	0.007
	0.001	61%	0.5	0.65	0.56	0.1
	0.05	40%	0.28	0.37	0.32	0.00003
	0.005	60%	0.47	0.64	0.54	0.02

Table 4.33, shows the accuracy results while the kernel size was set to [1 - 10] for 3 different kernel numbers and 5 learning rates.

Table 4.33: results achieved while the kernel size was [1 - 10] for invisible arrow (combined CNN with DWT Symlet 2)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 10, 3]	0.1	66%	0.66	0.66	0.66	0.0002
	0.01	66%	0.66	0.66	0.66	0.02
	0.001	75%	0.66	0.8	0.72	0.1
	0.05	63%	0.66	0.63	0.64	0.001
	0.005	66%	0.66	0.66	0.66	0.06
[1 - 10, 5]	0.1	68%	0.66	0.68	0.67	0.00001
	0.01	65%	0.66	0.64	0.65	0.02
	0.001	58%	0.66	0.57	0.61	0.2
	0.05	68%	0.66	0.68	0.67	0.001
	0.005	65%	0.66	0.64	0.64	0.06
[1 - 10, 7]	0.1	71%	0.66	0.73	0.7	0.0002
	0.01	64%	0.66	0.63	0.65	0.02
	0.001	54%	0.73	0.53	0.62	0.3
	0.05	67%	0.66	0.68	0.67	0.0009
	0.005	57%	0.66	0.56	0.60	0.07

The results achieved while applying the model with kernels size [1 - 15] on invisible shape (arrow), are illustrated in Table 4.34.

Table 4.34: results achieved while kernel size was [1 - 15] for invisible arrow (combined CNN with DWT Symlet 2)

Kernel	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
[1 - 15, 3]	0.1	61%	0.55	0.62	0.58	0.00001
	0.01	58%	0.61	0.57	0.59	0.08
	0.001	52%	0.55	0.52	0.54	0.3
	0.05	61%	0.61	0.61	0.61	0.0002
	0.005	58%	0.61	0.57	0.59	0.08
[1 - 15, 5]	0.1	63%	0.66	0.62	0.64	0.00004
	0.01	58%	0.66	0.57	0.61	0.03
	0.001	41%	0.56	0.43	0.49	0.3
	0.05	63%	0.66	0.62	0.64	0.001
	0.005	55%	0.43	0.56	0.49	0.09
[1 - 15, 7]	0.1	65%	0.66	0.65	0.65	0.00003
	0.01	64%	0.66	0.63	0.65	0.02
	0.001	50%	0.64	0.5	0.56	0.3
	0.05	65%	0.66	0.65	0.65	0.0002
	0.005	63%	0.66	0.62	0.64	0.08

Confusion matrix for the best attained result in invisible arrow has been presented in Table 4.35.

Table 4.35: confusion matrix for invisible arrow (combined CNN with DWT Symlet 2)

Total Data Point = 60000	**Actual Red**	**Actual Green**
Predict Red	20000 *(TP)*	5000 *(FP)*
Predict Green	10000 *(FN)*	25000 *(TN)*

Figure 4.3 shows the accuracy rate comparison between the proposed model based on combined CNN with DWT.

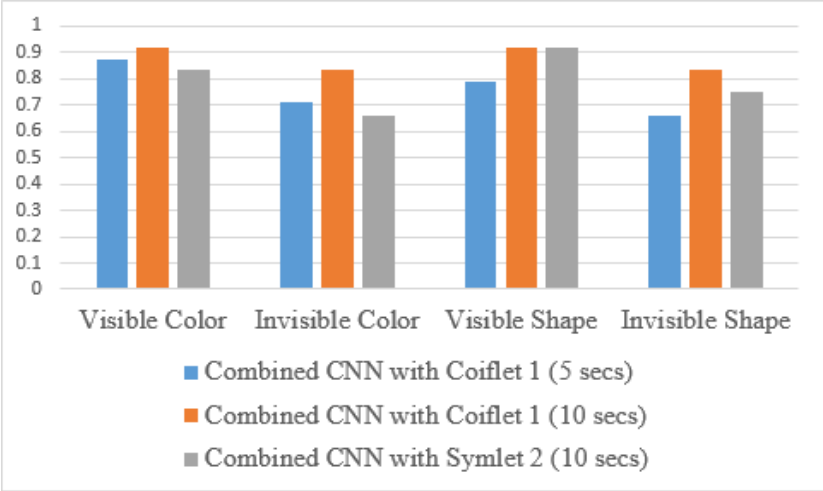


Figure 4.3: accuracy rate comparison between combined CNN with Coiflet 1 (10 secs), combined CNN with Coiflet 1 (5 secs), combined CNN with Symlet 2 (10 secs)

4.5 Results Based on Hybrid CNN and ALPS-GA

The second proposed model is composed of hybrid deep learning and an evolutionary algorithm (see section 3.9). CNN has been selected as a deep learning algorithm, then ALPS-GA has been combined with CNN to build the proposed model. This model is made up of three layers: convolution, DWT Coiflet 1, and ALPS-GA.

The convolution layer follows the same procedure of convolution layer in the CNN which is the convolution process between input and kernels (filters), and applied RecLU activation function on each data point (see equation (2.3)). According to the achieved results in both previous models, in most of

the cases the combined CNN with Coiflet 1 has the ability to provide the best accuracy result while kernel is set to [1 - 10, 3]. In order to achieve a good result with this hybrid proposed model, the DWT Coiflet 1 with kernel [1 - 10, 3] has been used instead of pooling layer. Thus, DWT Coiflet 1 has been applied on the results generated from convolution layer. Finally, the extracted features have been fed to the ALPS-GA and SDA has been used to perform the classification.

This model has been trained on the experimental dataset. All confusion matrices with scores achieved, with parameters used for individual category with respect to their mode are presented in this section. Finally, a comparison between previously mentioned proposed models, and this proposed model has been provided.

Many different parameters have been investigated to attain the best classification results, the mutation probability has been set 18% for visible color, and 15% for all other categories (invisible color, visible arrow, invisible arrow)

Table 4.36 shows the different accuracy rates based on different population sizes used in this model. In this study the best population size (obtain the best accuracy rate) has been selected to train the model.

Table 4.36: accuracy rate attained based on different population size for hybrid CNN and ALPS-GA (10 seconds)

Cate	Mode	Population Size	Accuracy	Precision	Recall	F-Measure
Color	**Visible**	250	77.7%	1.0	0.55	0.66
		300	**92%**	1.0	0.83	0.9
		350	75%	0.8	0.6	0.75
	Invisible	200	77.7%	1.0	0.55	0.66
		250	**83%**	0.83	0.83	0.83
		300	75%	0.71	0.83	0.76
Shape	**Visible**	250	75%	1.0	0.6	0.75
		300	**83%**	0.83	0.83	0.83
		350	83%	0.83	0.83	0.83
	Invisible	150	66%	0.5	0.75	0.6
		200	**75%**	0.8	0.66	0.72
		250	58.3%	0.6	0.5	0.54

Confusion matrices for visible and invisible colors have been shown in the Table 4.37. The best classification result achieved in visible color was 92%.

Table 4.37: confusion matrices for visible, and invisible color (Hybrid CNN with ALPS-GA)

Visible Color		
Data Point = 60000	**Actual Red**	**Actual Green**
Predict Red	25000 *(TP)*	0 *(FP)*
Predict Green	5000 *(FN)*	30000 *(TN)*

Invisible Color		
Data Point = 60000	**Actual Red**	**Actual Green**
Predict Red	25000 *(TP)*	5000 *(FP)*
Predict Green	5000 *(FN)*	25000 *(TN)*

Confusion matrices while performing classification on the shape (arrow) category based on Hybrid CNN and ALPS-GA, have been presented in Table 4.38. The best classification result attained in visible shape was 83%.

Table 4.38: confusion matrices for visible, and invisible arrow (Hybrid CNN with ALPS-GA)

Visible Arrow		
Data Point = 60000	**Actual For.**	**Actual Right**
Predict For.	25000 *(TP)*	5000 *(FP)*
Predict Right	5000 *(FN)*	25000 *(TN)*

Invisible Arrow		
Data Point = 60000	**Actual For.**	**Actual Right**
Predict For.	20000 *(TP)*	5000 *(FP)*
Predict Right	10000 *(FN)*	25000 *(TN)*

Figure 4.4 presents a comparison between accuracy rates achieved in the proposed models (combined CNN with DWT (Coiflet 1, and Symlet 2)) and Hybrid CNN and ALPS-GA.

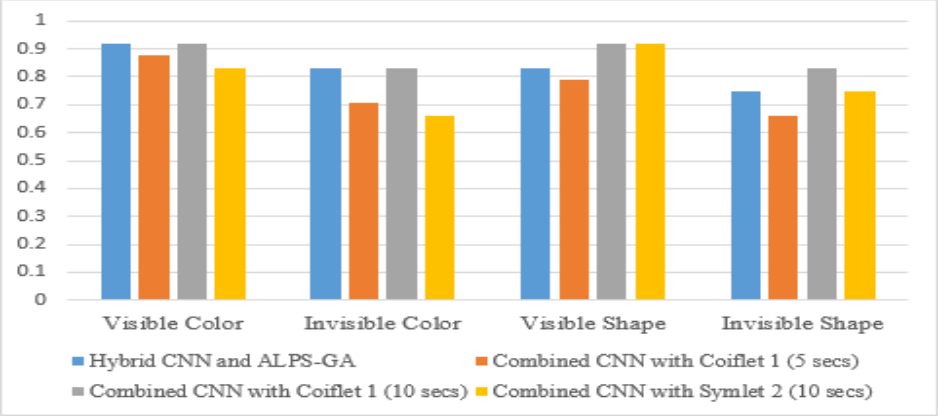


Figure 4.4: accuracy rate comparison between combined CNN with DWT (Coiflet 1, Symlet 2), and hybrid CNN and ALPS-GA

4.6 Results Based on the CNN

This model follows the same architecture of CNN, which involves three main layers: convolution, pooling, and fully connected layer. The convolution layer is made up of two operations, the first operation performs convolution between the input data and filters. Then, the generated data from the convolution process is fed to the rectified linear unit activation function ReLU. The second layer is pooling layer, in this stage the max pooling with stride 5 has been applied on the received data from convolution layer. Finally, the generated data from pooling layer are flattened and fed to the fully connected layer to predict the correct output.

Table 4.39, presents confusion matrices for visible color and invisible color. The best attained accuracy rate was 75%.

Table 4.39: confusion matrices for visible and invisible color category (CNN)

Visible Color		
Data Point = 60000	**Actual Red**	**Actual Green**
Predict Red	20000 *(TP)*	5000 *(FP)*
Predict Green	10000 *(FN)*	25000 *(TN)*

Invisible Color		
Data Point = 60000	**Actual Red**	**Actual Green**
Predict Red	20000 *(TP)*	15000 *(FP)*
Predict Green	10000 *(FN)*	15000 *(TN)*

Table 4.40, present the confusion matrices for arrow category while applying CNN architecture. The best accuracy rate achieved in visible mode which was 83% .

Table 4.40: confusion matrices for visible, and invisible arrow category (CNN)

Visible Arrow		
Data Point = 60000	**Actual For.**	**Actual Right**
Predict For.	30000 *(TP)*	10000 *(FP)*
Predict Right	0 *(FN)*	20000 *(TN)*

Invisible Arrow		
Data Point = 60000	**Actual For.**	**Actual Right**
Predict For.	20000 *(TP)*	10000 *(FP)*
Predict Right	10000 *(FN)*	20000 *(TN)*

In order to design a robust CNN architecture many different parameters (learning rate, momentum, epoch) and different CNN design (multiple convolution layer, multiple pooling layer, and different neural networks) have

been experimented. In this section, the best results and scores achieved in this model with parameters used in individual categories has been presented as follow: visible color, invisible color, visible shape (arrow), and invisible shape (arrow) see Table 4.41.

Table 4.41: results achieved while using CNN

Cate.	Mode	L.R η	Accuracy	Recall	Precision	F-Measure	Log Loss
Color	Visible	0.1	75%	0.83	0.74	0.76	0.000002
		0.01	75%	0.66	0.8	0.72	0.0001
		0.001	66%	0.66	0.66	0.66	0.04
		0.05	75%	1.0	0.66	0.8	0.0002
		0.005	33%	0.5	0.37	0.42	0.0008
	Invisible	0.1	58%	1.0	0.54	0.7	0.000003
		0.01	58%	0.66	0.57	0.611	0.0003
		0.001	50%	0.5	0.5	0.5	0.1
		0.05	50%	0.83	0.5	0.62	0.00008
		0.005	58%	0.5	0.6	0.54	0.1
Shape	Visible	0.1	83%	1.0	0.66	0.85	0.000007
		0.01	58%	0.5	0.6	0.54	0.0003
		0.001	83%	0.83	0.83	0.83	0.1
		0.05	75%	0.83	0.71	0.76	0.00003
		0.005	75%	1.0	0.66	0.8	0.005
	Invisible	0.1	66%	0.66	0.66	0.66	0.000002
		0.01	50%	0.16	0.5	0.25	0.0003
		0.001	41%	0.5	0.42	0.46	0.1
		0.05	66%	0.66	0.66	0.66	0.00002
		0.005	50%	0.66	0.5	0.57	0.0008

4.7 Results from the System (CNN, Hybrid CNN with ALPS-GA, Combined CNN with Coiflet 1, and Combined CNN with Symlet 2)

This section provides a comparison between all the best results achieved in all models for each category with respect to their mode.

As it can be noticed in Table 4.42 the worst accuracy rate attained in the original CNN is when it was applied for classifying color in invisible mode which was 58%. According to the results achieved while applying the proposed models on the dataset, the combined CNN with DWT for Coiflet 1 and Symlet 2 attained the same accuracy rate when they are applied on the visible arrow (shape). However, combined CNN with DWT Coiflet 1, and ALPS-

GA obtained the same result in visible color, and invisible color. Based on the results achieved, the combined CNN with DWT Coiflet 1 has the ability to perform best classification for both categories with respect to their mode. Therefore, the best classifier model is the combined CNN and DWT Coiflet 1 with SGD such that in visible color, and shape, it had the accuracy of 92%, while for invisible color and shape it achieved 83%.

The accuracy rate comparison, and average accuracy rate comparison between the CNN and both proposed models have been presented in Figure 4.5, and 4.6 respectively.

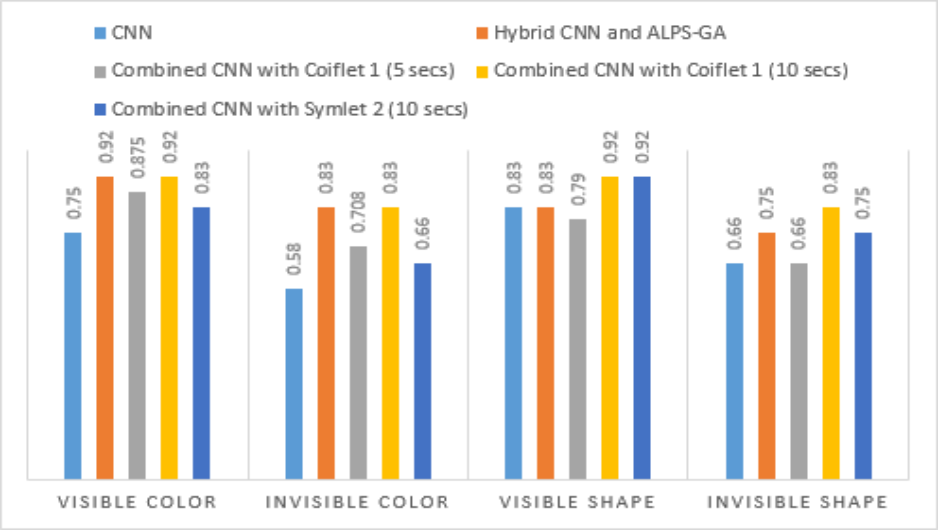


Figure 4.5: accuracy rate comparison between the CNN,Hybrid CNN and ALPS-GA, and combined CNN with Coiflet 1 and SGD

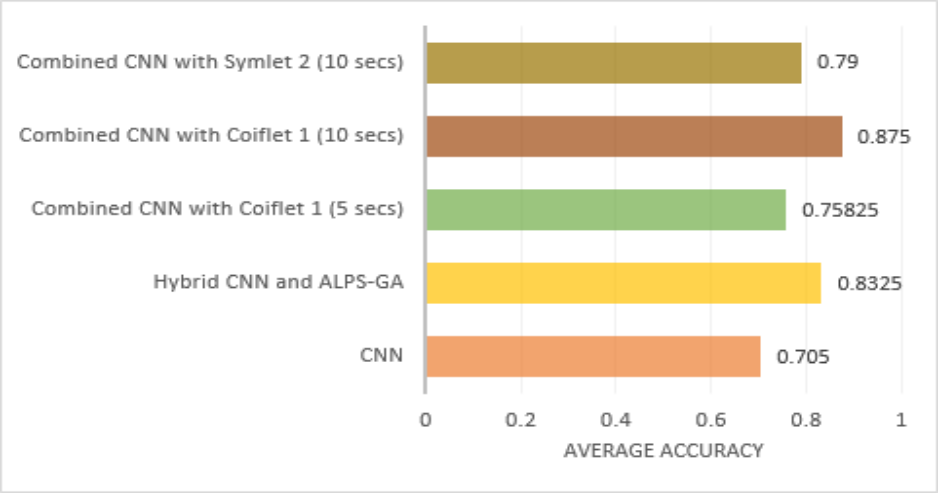


Figure 4.6: average accuracy rate comparison between the CNN,Hybrid CNN and ALPS-GA, and combined CNN with Coiflet 1 and SGD

Table 4.42: comparison table between the CNN, hybrid CNN and ALPS-GA, combined CNN and Coiflet 1 with SGD, and combined CNN and Symlet 2 with SGD

Cate.	Mode	Models	Accuracy	Recall	Precision	F-Measure	MSE / Log Loss
Color	Visible	**CNN**	75%	0.83	0.74	0.76	0.000002
		ALPS-GA	92%	0.83	1.0	0.9	0.07
		Coiflet (5 Secs)	87.5%	0.83	0.9	0.86	0.001
		Coiflet (10 Secs)	92%	1.0	0.85	0.92	0.07
		Symlet (10 Secs)	83%	1.0	0.75	0.857	0.00
	Invisible	**CNN**	58%	1.0	0.54	0.7	0.000003
		ALPS-GA	83%	0.83	0.83	0.83	0.1
		Coiflet(5 Secs)	70%	0.66	0.72	0.68	0.02
		Coiflet (10 Secs)	83%	0.83	0.83	0.83	0.05
		Symlet (10 Secs)	66%	0.83	0.62	0.71	0.2
Arrow	Visible	**CNN**	83%	1.0	0.66	0.85	0.000007
		ALPS-GA	83%	0.83	0.83	0.83	0.1
		Coiflet (5 Secs)	79%	0.75	0.81	0.78	0.1
		Coiflet (10 Secs)	92%	1.0	0.85	0.92	0.05
		Symlet (10 Secs)	92%	1.0	0.85	0.92	0.0007
	Invisible	CNN	66%	0.66	0.66	0.66	0.000002
		ALPS-GA	75%	0.66	0.8	0.72	0.2
		Coiflet (5 Secs)	66%	0.75	0.64	0.69	0.1
		Coiflet(10 Secs)	83%	0.83	0.83	0.83	0.03
		Symlet (10 Secs)	75%	0.66	0.8	0.72	0.1

The training time in all models depends on many parameters. The kernel size and the kernel count have a direct impact on the duration of the training time. The number of epochs also had an effect on the duration of the training process such that the greater the epoch, the more time is required to train the model. During the training the maximum time required for training was in the hybrid CNN with ALPS-GA which required approximately 53 minutes. Table 4.43 presents the best results with the duration of the training process.

Table 4.43: time duration comparison between the CNN, hybrid CNN and ALPS-GA, combined CNN and Coiflet 1 with SGD, and combined CNN and Symlet 2 with SGD

Cate.	Mode	Models	Accuracy	Time Duration
Color	**Visible**	**CNN**	75%	15 min
		ALPS-GA	92%	49 min
		Coiflet (5 Secs)	87.5%	14 min
		Coiflet (10 Secs)	92%	11 min
		Symlet (10 Secs)	83%	10 min
	Invisible	**CNN**	58%	15 min
		ALPS-GA	83%	51 min
		Coiflet(5 Secs)	70%	14 min
		Coiflet (10 Secs)	83%	12 min
		Symlet (10 Secs)	66%	10 min
Arrow	**Visible**	**CNN**	83%	15 min
		ALPS-GA	83%	49 min
		Coiflet (5 Secs)	79%	13 min
		Coiflet (10 Secs)	92%	11 min
		Symlet (10 Secs)	92%	10 min
	Invisible	CNN	66%	18 min
		ALPS-GA	75%	53 min
		Coiflet (5 Secs)	66%	14 min
		Coiflet(10 Secs)	83%	12 min
		Symlet (10 Secs)	75%	11 min

Chapter 5

Conclusions and Future Directions

5.1 Conclusions

In this study, the classification of brainwave signals based on algorithms inspired by biological process has been investigated. Two different methods have been selected for designing the models which are: convolutional neural network as a deep learning algorithm, and an age-layered population structure as meta-heuristic algorithm. These models are designed to classify colors (red, green), and shapes (forward, right) based on brainwave signals. In order to train and test the models, a dataset has been recorded. This study proposed two different models based on CNN, conclusions for these models are summarized below:

1. Preparing and pre-processing: Preparing and pre-processing of the dataset based on maximum attention to train the model could have a direct impact to perform good learning and achieve good classification results.

2. Kernels: Three different kernel sizes [1 - 5], [1 - 10], and [1 - 15], each with three different kernel counts [3, 5, 7] have been experimented. In another word, nine different kernels have been tested, each has been trained with five different learning rates. The best results were obtained while using kernel [1 - 10, 3].

3. Combined CNN with DWT (Coiflet 1, and Symlet 2): This model is trained in three circumstances. First, the proposed CNN combined with

DWT Coiflet 1 has been applied while input data is set to 5 seconds of brainwave signals, the results attained were: 87.5%, 70.5%, 79%, and 66%. Second, the same model has been applied while the input data was set to 10 seconds of brainwave signals the results achieved were: 92%, 83%, 92%, and 83%. Finally, the proposed CNN combined with DWT Symlet 2 has been applied while the input data is set to 10 seconds of brainwave signals, the results reached were: 83%, 66%, 92%, and 75% for visible color, invisible color, visible shape and invisible shape respectively.

4. Hybrid CNN with ALPS-GA: This model had the ability to classify brainwave signals with high accuracy rate compared to CNN. The proposed model has been experimented with different population sizes, and different mutation probabilities. The accuracy rates achieved in this model for visible color, invisible color, visible shape and invisible shape were 92%, 83%, 83%, and 75%, respectively.

5. Average accuracy: for each category with respect to their mode the average accuracy rate achieved were 79%, 87%, 75%, 83%, and 70% for combined CNN with Symlet 2, combined CNN with Coiflet 1 (10 secs), combined CNN with Coiflet 1 (5 secs), Hybrid CNN and ALPS-GA, and CNN respectively.

6. Results: According to the results, it was observed that in most of the cases the classification of brainwave signals in the visible mode can achieve better accuracy compared to results acquired in the invisible mode for both categories (color and shape (arrow)).

7. Data points: During the recording number of data points per a second received in the visible modes are more than data points received in the

invisible mode per a second.

8. Brainwave behavior: Brainwave signals are noisy signals. It also has non-stationary, sensitive, and similar structure. Thus, pre-processing is a required step to obtain better signal quality. However, it is compulsory to know how to pre-process it. Complexity of pre-processing technique had counter effect on the classification result.

9. Architecture: The models have been experimented with more than one design, such as adding another convolution layer, adding hidden layer, different order of DWT methods, and using back-propagation. However, it was noted that increasing the complexity of the model's architecture (increasing processing) had a reverse impact on the classification results, and sometime it was unable to achieve correct classification.

   To conclude, when CNN was combined with Coiflet 1, and 10 seconds of brainwave signals were used as the input data, the accuracy was higher or equal (in rare cases) compared to all other implementations. Furthermore, the CNN without any modification was also applied on the experimental dataset, it achieved lower classification results compared to other models.

5.2 Limitations of the Study

Every study has limitations, here are limitations of this study:

1. EEG device: NeuroSky mindwave has a single electrode. Thus, it is able to record brainwave signals only from Fp1 electrode.

2. Dataset: The recorded dataset has been divided into two sets which are train set (80%) and test set (20%). Both training and testing sets have been recorded from the same participants. However, to be more precise

it is better to record the test set from different participants.

3. Applications: in this study, to achieve accurate brainwave signals the emphasis was on recording brainwave signals with highest participant's concentration. However, to be more applicable it is better to achieve good classification results with low concentration.

5.3 Future Work

The work performed in this study is limited in scope and as such it can be improved in the future as follows:

1. EEG device: Another EEG device can be used, which can collect brainwave signals from different brain lobs, and has the ability to record signals from multiple electrodes.

2. Kernel weights: The kernels weights can be initialized randomly using Xavier initialization algorithm, or using random Gaussian distribution.

3. Feature extraction: The extracted features from both DWT methods Coiflet 1 and Symlet 2 can be combined, and then the common features can be used to train the models.

4. Fully connected layer: Different trainer algorithms can be tried while training the network in the fully connected layer such as Newton, or Quasi-Newton method.

5. Training the model with functions for controlling the prosthetic hand.

Bibilography

[1] Tim Wheelock. An introduction to human neuroanatomy. Technical report, Harvard Brain Tissue Resource Center, McLean Hospital, Belmont, MA 02478, 1-800-BRAIN BANK, 2014.

[2] Tonya Hines. Anatomy of the brain. https://mayfieldclinic.com/pe-anatbrain.htm, Apr 2018. [Online; accessed 25-May-2019].

[3] Özal Yıldırım, Ulas Baran Baloglu, and U Rajendra Acharya. A deep convolutional neural network model for automated identification of abnormal eeg signals. *Neural Computing and Applications*, pages 1–12, 2018.

[4] Elena Ratti, Shani Waninger, Chris Berka, Giulio Ruffini, and Ajay Verma. Comparison of medical and consumer wireless eeg systems for use in clinical trials. *Frontiers in Human Neuroscience*, 11:398, 2017.

[5] M. Z. Parvez and M. Paul. Seizure prediction using undulated global and local features. *IEEE Transactions on Biomedical Engineering*, 64(1):208–217, Jan 2017.

[6] Razali Tomari, Rozi Roslinda Abu Hassan, Wan Nurshazwani Wan Zakaria, and Rafidah Ngadengon. Analysis of optimal brainwave concentration model for wheelchair input interface. *Procedia Computer Science*, 76:336–341, 2015.

[7] Carl E Stafstrom and Lionel Carmant. Seizures and epilepsy: an overview for neuroscientists. *Cold Spring Harbor perspectives in medicine*, 5(6):a022426, 2015.

[8] CUI Gaochao, ZHU Li, WANG Dongsheng, and CAO Jianting. Eeg analysis for differentiating between brain death and coma in humans. *International Journal of Computers & Technology*, 15(11):7189–7201, 2016.

[9] Stephan Waldert. Invasive vs. non-invasive neuronal signals for brain-machine interfaces: will one prevail? *Frontiers in neuroscience*, 10:295, 2016.

[10] Kyle E Mathewson, Tyler JL Harrison, and Sayeed AD Kizuk. High and dry? comparing active dry eeg electrodes to active and passive wet electrodes. *Psychophysiology*, 54(1):74–82, 2017.

[11] Zhichuan Tang, Chao Li, and Shouqian Sun. Single-trial eeg classification of motor imagery using deep convolutional neural networks. *Optik-International Journal for Light and Electron Optics*, 130:11–18, 2017.

[12] Cahya Rahmad, Rudy Ariyanto, and Dika Rizky Yunianto. Brain signal classification using genetic algorithm for right-left motion pattern. *International Journal of Advanced Computer Science and Applications*, 9(11):247–251, 2018.

[13] Hayit Greenspan, Bram Van Ginneken, and Ronald M Summers. Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Transactions on Medical Imaging*, 35(5):1153–1159, 2016.

[14] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[15] Dongkoo Shon, Kichang Im, Jeong-Ho Park, Dong-Sun Lim, Byungtae Jang, and Jong-Myon Kim. Emotional stress state detection using genetic algorithm-based feature selection on eeg signals. *International journal of environmental research and public health*, 15(11):2461, 2018.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[17] U Rajendra Acharya, Shu Lih Oh, Yuki Hagiwara, Jen Hong Tan, Hojjat Adeli, and D Puthankattil Subha. Automated eeg-based screening of depression using deep convolutional neural network. *Computer methods and programs in biomedicine*, 161:103–113, 2018.

[18] U Rajendra Acharya, Shu Lih Oh, Yuki Hagiwara, Jen Hong Tan, and Hojjat Adeli. Deep convolutional neural network for the automated detection and diagnosis of seizure using eeg signals. *Computers in biology and medicine*, 100:270–278, 2018.

[19] Ralph G Andrzejak, Klaus Lehnertz, Florian Mormann, Christoph Rieke, Peter David, and Christian E Elger. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review E*, 64(6):061907, 2001.

[20] Caglar Uyulan and Turker Tekin ERguzel. Comparison of wavelet families for mental task classification. 2016.

[21] Indah Agustien Siradjuddin, Rivaldi Septasurya, M Kautsar Sophan, Noor Ifada, and Arif Muntasa. Feature selection with genetic algorithm for alcoholic detection using electroencephalogram. In *2017 International Conference on Sustainable Information Engineering and Technology (SIET)*, pages 230–234. IEEE, 2017.

[22] Wei-Long Zheng, Jia-Yi Zhu, Yong Peng, and Bao-Liang Lu. Eeg-based emotion classification using deep belief networks. In *2014 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2014.

[23] Gerwin Schalk, Dennis J McFarland, Thilo Hinterberger, Niels Birbaumer, and Jonathan R Wolpaw. Bci2000: a general-purpose brain-computer interface (bci) system. *IEEE Transactions on biomedical engineering*, 51(6):1034–1043, 2004.

[24] Yu Zhang, Guoxu Zhou, Jing Jin, Qibin Zhao, Xingyu Wang, and Andrzej Cichocki. Sparse bayesian classification of eeg for brain–computer interface. *IEEE transactions on neural networks and learning systems*, 27(11):2256–2267, 2015.

[25] Erik Andreas Larsen. Classification of eeg signals in a brain-computer interface system. Master's thesis, Institutt for datateknikk og informasjonsvitenskap, 2011.

[26] JJ Szafir. Non-invasive bci through eeg: An exploration of the utilization of electroencephalography to create thought-based brain-computer interfaces. *Boston College*, 2010.

[27] Rozi Roslinda Abu Hassan. Eeg signal classification for wheelchair control application. Master's thesis, Universiti Tun Hussein Onn Malaysia, 2015.

[28] Ozan Gunaydin and Mehmed Ozkan. Design of a brain computer interface system based on electroencephalogram (eeg). In *4th European Education and Research Conference (EDERC 2010)*, pages 150–154. IEEE, 2010.

[29] Mahnaz Arvaneh, Cuntai Guan, Kai Keng Ang, and Chai Quek. Optimizing the channel selection and classification accuracy in eeg-based bci. *IEEE Transactions on Biomedical Engineering*, 58(6):1865–1873, 2011.

[30] MindWave Mobile Incorporation. Mindwave mobile:user guid. Technical report, MindWave Mobile Incorporation, Aug 2015.

[31] MindWave Mobile Incorporation. Tgam1 spec sheet. Technical report, NeuroSky Brain-Computer Interface Technologies, Mar 2010.

[32] MindWave Mobile Incorporation. Thinkgear connector user guide. Technical report, NeuroSky Brain-Computer Interface Technologies, Mar 2015.

[33] Anton Coenen and Oksana Zayachkivska. Adolf beck: A pioneer in electroencephalography in between richard caton and hans berger. *Advances in cognitive psychology*, 9(4):216, 2013.

[34] Jim. Brain waves. http://www.crystalblueent.com/brain-waves.html, 2015. [Online; accessed 25-May-2019].

[35] Alamgir Hossan and AM Mahmud Chowdhury. Real time eeg based automatic brainwave regulation by music. In *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, pages 780–784. IEEE, 2016.

[36] Brendan Fortuner. Loss functions. https://ml-cheatsheet.readthedocs.io/en/latest/loss-functions.html, 2017. [Online; accessed 25-May-2019].

[37] James D. McCaffrey. Log loss and cross entropy are almost the same. https://jamesmccaffrey.wordpress.com/2016/09/25/log-loss-and-

cross-entropy-are-almost-the-same/, Sep 2016. [Online; accessed 25-May-2019].

[38] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[39] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[40] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[41] Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

[42] Jianxin Wu. Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, pages 5–23, 2017.

[43] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[44] Sumit Saha. A comprehensive guide to convolutional neural networks —the eli5 way. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, Dec 2018. [Online; accessed 20-May-2019].

[45] John Wang. *Encyclopedia of business analytics and optimization*. IGI Global, 2014.

[46] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[47] Anish Singh Walia. Types of optimization algorithms used in neural networks and ways to optimize gradient descent. https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f, Jun 2017. [Online; accessed 01-June-2019].

[48] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[49] Vitaly Bushaev. How do we 'train' neural networks ? https://towardsdatascience.com/how-do-we-train-neural-networks-edd985562b73, Nov 2017. [Online; accessed 02-June-2019].

[50] Vijini Mallawaarachchi. Introduction to genetic algorithms. https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3, Jul 2017. [Online; accessed 02-June-2019].

[51] RA Welikala, Muhammad Moazam Fraz, Jamshid Dehmeshki, Andreas Hoppe, V Tah, S Mann, Thomas H Williamson, and Sarah A Barman. Genetic algorithm based feature selection combined with dual classification for the automated detection of proliferative diabetic retinopathy. *Computerized Medical Imaging and Graphics*, 43:64–77, 2015.

[52] Pedram Ghamisi and Jon Atli Benediktsson. Feature selection based on hybridization of genetic algorithm and particle swarm optimization. *IEEE Geoscience and remote sensing letters*, 12(2):309–313, 2014.

[53] Vipul Jain, Amit Kumar, Sameer Kumar, and Charu Chandra. Weight restrictions in data envelopment analysis: a comprehensive genetic al-

gorithm based approach for incorporating value judgments. *Expert Systems with Applications*, 42(3):1503–1512, 2015.

[54] Audrey Opoku-Amankwaah and Beatrice Ombuki-Berman. An age layered population structure genetic algorithm for the multi-depot vehicle problem. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2017.

[55] Gregory S Hornby. The age-layered population structure(alps) evolutionary algorithm. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. Citeseer, 2009.

[56] K. Poulose Jacob, Sunny Sonia, and P Serrano David. A comparative study of wavelet based feature extraction techniques in recognizing isolated spoken words. In *SiPS 2013*, 2013.

[57] Mohammed Gulam Ahamad D.Ravichandran, Ramesh Nimmatoori. Mathematical representations of 1d, 2d and 3d wavelet transform for image coding. *International Journal on Advanced Computer Theory and Engineering (IJACTE)*, 5(3):1–27, 2016.

[58] Jason H Moore, Joel S Parker, and Lance W Hahn. Symbolic discriminant analysis for mining gene expression patterns. In *European Conference on Machine Learning*, pages 372–381. Springer, 2001.

[59] Marco Peixeiro. Classification—linear discriminant analysis. https://towardsdatascience.com/classification-part-2-linear-discriminant-analysis-ea60c45b9ee5, Dec 2018. [Online; accessed 25-May-2019].

[60] Gabriele Garnero and D Godone. Comparisons between different interpolation techniques. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 5:W3, 2013.

[61] Sameer Deshmukh. One dimensional interpolation: Introduction and implementation in ruby. v0dro.in/blog/2014/11/29/one-dimensional-interpolation-introduction-and-implementation-in-ruby/, 2017. [Online; accessed 28-May-2019].

[62] Ayyüce Kızrak. Comparison of activation functions for deep neural networks. https://towardsdatascience.com/comparison-of-activation-functions-for-deep-neural-networks-706ac4284c8a, May 2019. [Online; accessed 12-June-2019].

[63] *Mach. Learn.*, 30(2-3), 1998.

[64] Kevin Markham. Simple guide to confusion matrix terminology. https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/, Mar 2014.

[65] LEGO Group. Lego mindstorms nxt educationkit. Technical report, Mindstorms Education, 2009.

[66] Wikipedia contributors. Lego mindstorms — Wikipedia, the free encyclopedia, 2019. [Online; accessed 25-May-2019].

[67] Wikipedia contributors. Lego mindstorms nxt — Wikipedia, the free encyclopedia, 2019. [Online; accessed 25-May-2019].

پۆلێنکردنی شەپۆلەکانی مێشک لە سەر بنەمای فێربونی قوڵ بۆ کارپێکردنی ئامێرە ئەلیکترۆنیەکان

نامەیەکە پێشکەش کراوە بە
ئەنجومەنی کۆلێجی زانست / زانکۆی سلێمانی
وەک بەشێک لە پێداویستیەکانی بەدەست هێنانی
بروانامەی ماستەر لە زانستی کۆمپیوتەر

ئامادە کردن و نوسینەوەی لەلایەن

ژیار ڕزگار کوێخا ڕۆستەم

بەسەرپەرشتی

د.سۆزان عبدالله محمود

پڕۆفیسۆری یاریدەدەر

ئەیلول ٢٠١٩ زایینی گەلاوێژ ٢٧١٩ کوردی

پوختەی تێز

هەموو مێشکێکی زیندو ژمارەیەکی زۆر شەپۆل بەرهەم دەهێنێت، ئەم شەپۆلانە گۆڕانکاریان بەسەردادێت لەسەر بنەمای ئەو کارانەی یان ئەو بیرکردنەوانەی ئەنجام دەدرێت. ئەم شەپۆلانە دەتوانریت لە ڕێگەی ئامێری ئیلیکترۆئینسیفلۆگرامەوە (Electroencephalogram) بخوێنرێنەوە.
پۆلێنکردنی ئەم شەپۆلانە و وەرگێڕانیان کارێکی ئاسان نیە، بە هۆی ئەو گۆڕانکاریە بەردەوامانەی دروست دەبێت لە شەپۆلە بەدەست هاتوەکاندا. وەک چارەسەرێک بۆ پۆلێنکردنی ئەم شەپۆلانە، ئەم توێژینەوە پێشنیاری دوو شیوازی کردوە کە هەردوو شیوازەکە بنەماکەیان لە پڕۆسەی بایۆلۆژیەوە وەرگیراوە.
شیوازەکان بنیادنراون لەسەر بنەمای Deep learning وە خواریزمی Meta-Heuristic .
Convolutional Neural Network (CNN) بەکارهێنراوە وەک یەکێک لە خواریزمەکانی Deep learning، وە هەروەها Age-Layered Population Structure (ALPS) بەکارهاتووە وەک یەکێک لە خواریزمە جینیەکان. بەشێوازێکی گشتی (CNN) پێک هاتوە لە سێ چینی جیاوازبریتین لە (convolution, pooling, and fully connected). یەکێک لە شێوازە پێشنیار کراوەکان بنادنراوە بە کۆکردنەوە و تێکەڵکردنی CNN بەجیا لەگەڵ دوو ڕێگەی جیاواز لە Discrete Wavelet Transform (DWT)، ڕێگەکان بریتین لە (Coiflet 1, Symlet 2). لەم شیوازەدا چینی دوەەم کە pooling لادراوە و Coiflet 1, Symlet 2 لە جێگەی دانراوە. وە هەروەها لەم شێوازدا لە کۆتا چین fully connected خواریزمی Stochastic Gradient Descent (SGD) وەک فێرکەر بەکار هاتووە.
شێوازی دووەەم، بەهەمان شێوازی پێشوو لەسەر بنەمای (CNN) بنیادنراوە، بە زیادەی چەند گۆڕانکاریەک. چینی دوەەم pooling لادراوە و خواریزمی DWT Coiflet 1 لە جێگەی دانراوە، وە هەروەها کۆتا چین fully connected لادراوە خواریزمی ALPS لەگەڵ Symbolic Discriminant Analysis لە بری بەکارهاتووە.
بۆ تاقیکردنەوەی توانا و لێهاتوی ئەم دوو مۆدێلە پێشنیار کراوە، سێتێک لە داتای مێشک کۆکراوەتەوە لە شەش بەشدار بوو، کە هەریەکێکیان توانای بینینی ئاسایان هەبووە، لەگەڵ ڕەوشێکی جێگیری تەندروستی و دەروونی، وە تەمەنیان لە نێوان ٢٥ – ٣٥ ساڵیدا بوو. داتاکان کۆکراونەتەوە لە کاتی جیاوازدا، و لە ٢ باری جیاواز کە بریتیە لە بینراو (بەشداربوو چاو دەکاتەوە و هۆش و هەستی دەخاتە سەر ئەو ڕەنگە یان ئەو ئاڕاستەی پیشانی دراوە)، نەبینراو (بەشداربوو چاو دادەخات و هۆش و هەستی دەخاتە سەر ئەو ڕەنگە یان ئەو ئاڕاستەی پیشانی دراوە).

جگە لە دوو مۆدێلە پێشنیار کراوەکە، CNN ئاسایی بنیادنراوەتەوە و تاقیکراوەتەوە. لە سەر بنەمای ئەنجامە بەدەست هاتوەکان، باشترین ئەنجام بۆ پۆلێنکردنی شەپۆلەکان لە کاتی بەکارهێنانی شێوازی combined CNN with Coiflet 1 وە بەکارهێنانی SGD وەک فێرکەر بەدەست هاتووە. باشترین ئەنجامی وردی پۆلێنکردن بەدەست هاتووە لە گروپی بینراو بۆ شێوازی ڕەنگ و ئاڕستەکان بریتیە لە ٩٢٪، وە باشترین ئەنجامی وردی پۆلێنکردن بۆ گروپی نەبینراو بۆ شێوازی ڕەنگ و ئاڕاستەکان بریتیە لە ٨٣٪.
لە ئەنجامدا باشترین شێوازی پێشنیار کراو بەکار هات بۆ کۆنترۆڵکردن و ئاڕاستەکردنی ئەو ڕۆبۆتەی کە لەم توێژینەوەدا دیزاینکراوە.

تصنيف إشارات موجات الدماغ على أساس التعلم العميق للتحكم على الأجهزة الإلكترونية

رسالة مقدمة الى

مجلس كلية العلوم / جامعة السليمانية

كجزء من متطلبات نيل شهادة ماجستر في علوم الحاسبات

من قبل

ژیار ڕزگار کوێخا ڕۆستەم

باشراف

د.سۆزان عبدالله محمود

الاستاذ المساعد

السنة الميلادية

ايلول ٢٠١٩

السنة الهجرية

محرم 1441

المستخلص

ان الأدمغة الحية تنتج عادة عدداً كبيراً من الموجات، وبناءاً على الأعمال التي تتم انجازها او نوعية التفكير المطلوب فان تملك الموجات تتعرض للتغيير و التحول، ومن الممكن قراءة تلك الأمواج من خلال استعمال الجهاز المعروف باسم Electroencephalogram.

يعد تصنيف هذه إشارات مهمة صعبة نظرًا لطبيعتها غير الثابتة، وكحلٍ لمسألة تصنيف تلك الأمواج، فإن هذا البحث يقترح استخدام نمطين، كلاهما مأخوذان من اصول خاصة بالعمليات البايولوجية.

وأُسس هذان النمطان على مبدأ التعلم العميق و كذالك مبدأ Meta-heuristic الخوارزمي.

وقد أستعمل طرقة الشبكة العصبية التلافيفية Convolutional Neural Network المعروفة اختصاراً باسم CNN كإحدى مكونات المبدأ المعرف باسم Deep Learning وكذالك إستخدام إحدى خوارزميات المورثة و المعرفة باسم Age-Layered Population Structure او ALPS.

بصورة عامة CNN يتكون من ثلاث طبقات متابانية وهي convolution, pooling, fully connected. والواقع فان إحدى الأنماط المقترحة مبنية على أساس الجمع و الخط بين نمط CNN مع طريقتين متباينتين من Discrete Wavelet Transform (DWT) و هاتين الطريقتين هما (Coiflet 1, Symlet 2) و بإستخدام هذا النمط فان الطبقة الثانية المعرفة ب pooling قد أستبعدت في العملية و استخدام بدلاً عنها Coiflet 1, Symlet 2 وفي نفس الوقت فان الطبقة الثالثة و الاخيرة المعرفة باسم fully connected الخوارزمي Stochastic Gradient Descent (SGD) قد استعمل كعامل للتعليم.

اما النمط الثاني فانه قد بنى على نفس الأسس التي بني عليها CNN وذالك عن طريق إجراء بعض التعديلات عليها، وفقد تم إستبعاد الطبقة الثانية pooling، و استبدال بالمكون الخوارزمي DWT Coiflet 1 الذي وضع مكانه. وكما ان الطبقة الاخيرة المعروفة باسم fully connected قد أستبعد، و استبدل بالمكون الخوارزمي ALPS وكذالك Symbolic Discriminant Analysis الذي وضع مكانه.

ولتجربة وتقييم فعالية هذين النمطين المقترحين آنفاً، فقد تم جمع مجموعة من معطيات (data) الدماغ من ستة أشخاص من المتطوعين الذين كانوا يملكون قدرة عادية على الرؤية، مع تمتعهم بشروط نفسية سوية وكانت اعمارهم تتراوح بين 25 – 35 وقد جمعت المعطيات في اوقات متبانية، في حالتين مختليفتين تتعلقان بالرؤية او المشاهدة، يطلب من المتطوع ان يفتح عينيه و يركز بكل جوارحه على ذالك اللون او ذالك الإتجاه الذي عرض عليه. اما في حالة كون المتطوع مغمض العيون، فان المتطوع المشترك التجربة يقوم بحجب عينيه و يطلب منه ان يستذكر اللون او الأتجاه الذي سبق و ان شاهده.

إضافة الى النمطين المقتريحين، فقدتم إعادة بناء النمط المعروف ب CNN كما تم تجربة. و بناءاً على النتائج التي تم الحصول عليها، فانه قد تم الحصول على الأفضل النتائج لتصنيف الأمواج وذالك في حالة

إستخدام SGD كعامل للتعليم، كما تم الحصول على أفضل النتائج الدقيقة للتصنيف و للتمييز بقدر 92% وذالك ضمن المجموعة الخاصة بالرؤية من نمط اللون والأتجاه، اما افضل النتائج الدقيقة بخصوص المجموعة المغضمة العيون (المجموعة التي لم تعتمد على المشاهدة بالعين المجردة) فان دقة الرؤية تبلغ 83%.