Thesis No. : CSER-23-09

A Study on Classification of Motor Imagery EEG Signal Based on Deep Learning Approach

By

Ilma Hossain

Roll: 1707105

&

Lamia Hossain

Roll: 1707110

Department of Computer Science and Engineering

Khulna University of Engineering & Technology

Khulna 9203, Bangladesh

June 2025

A Study on Classification of Motor Imagery EEG Signal Based on Deep Learning Approach

By

Ilma Hossain
Roll: 1707105
&
Lamia Hossain
Roll: 1707110

A thesis submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science and Engineering

Supervisor:

S.M. Taslim Uddin Raju
Lecturer
Department of Computer Science and Engineering
Khulna University of Engineering & Technology
Khulna, Bangladesh

Signature

Department of Computer Science and Engineering
Khulna University of Engineering & Technology
Khulna 9203, Bangladesh

June 2025

Acknowledgement

All the praise to the almighty Allah, whose blessing and mercy succeeded us to complete this thesis work fairly. After that, we would like to express our special appreciation and thanks to our supervisor S.M. Taslim Uddin Raju, Lecturer, Department of Computer Science and Engineering, Khulna University of Engineering & Technology, under whose supervision this work was carried out. His intellectual advice, encouragement, and guidance make us feel confident and scientific research needs much effort in learning and applying and need to have a broad view of problems from a different perspective. We would like to convey our heartiest gratitude to all the faculty members, officials, and staff of the Department of Computer Science and Engineering as they have always extended their cooperation to complete this work. Last but not least, we wish to thank our friends and family members for their constant support.

- Authors

Abstract

The classification of motor imagery gives significant scientific insight into the neural processes underlying motor control. Researchers can acquire a greater knowledge of how the brain originates and controls voluntary movements by analyzing the patterns of brain activity related to imagined movement. This understanding has the potential to guide the development of novel therapies for neurological diseases that impact motor performance, such as stroke and Parkinson's disease. Using Artificial Neural Networks (ANNs), Deep Neural Networks (DNNs), Bidirectional Long Short-Term Memory (Bi-LSTM), Random Forest, and K-Nearest Neighbors (KNN) algorithms, we present comparative research of motor imagery classification of left and right-hand movement in this thesis. The objective of this study is to evaluate the performance of these machine learning methods for classifying motor imagery EEG data, which are extensively employed in brain-computer interfaces (BCIs). The dataset has been acquired from e ABSP lab of the Department of Biomedical Engineering at KUET. Artificial Neural Networks (ANNs), Deep Neural Networks (DNNs), Bidirectional Long Short-Term Memory (Bi-LSTM), Random Forest, and K-Nearest Neighbors (KNN) algorithms have been applied to the acquired dataset. DNNs and Bi-LSTM performed better than the other algorithms, with Bi-LSTM obtaining the greatest accuracy. This work concludes with a detailed comparison of several deep learning methods for the classification of motor imagery and highlights the potential of DNNs and Bi-LSTM for EEG-based BCI applications. This study's findings can give useful insights for future research toward the development of efficient and accurate BCIs. This work concludes with a detailed comparison of several deep learning methods for the classification of motor imagery and highlights the potential of DNNs and Bi-LSTM for EEG-based BCI applications. This study's findings can give useful insights for future research toward the development of efficient and accurate BCIs.

Contents

List of Figures

List of Tables

Chapter I

Introduction

1.1 Background

At the start of the twentieth century, EEG (Electroencephalography) was first ever pioneered in humans and since then it has been useful for monitoring cerebral functioning in the human brain. Implementing Brain Computer Interface (BCI) systems using motor imaging brain electrical signals was a mature application of using EEG signals. It was a great help to understand the limb movement of the hand and feet which is very helpful for disabled people who are unable to ensure themselves to perform their daily activities. An unfortunate reality of EEG is that it is less adaptable as different individuals have different physiological functions or even one individual in different mental states. Therefore, understanding EEG signals further contributes to a significance for humankind. Many models have been developed in this scope. However, it is still a challenging task to classify motor imaging EEG signals with good speed [1].

1.2 Motivation

15 million individuals worldwide suffer from stroke each year. Despite extensive rehabilitation programs, five million of these individuals remain chronically incapacitated and are no longer able to care for themselves. Priority is given to lifesaving and thrombolytic treatments in the first few days following an occurrence. However, patients should exercise as soon as possible to accelerate the recovery and neural reorganization process. Various rehabilitation methods are used for post-stroke therapy. One of them is Motor Imagery, a mental process through which a person practices or replicates a certain movement with or without external cues. MI was first intended to improve the performance of athletes and has since been used in stroke rehabilitation programs to aid with motor recovery. It is a process in which a specific action is replicated in working memory without the need of actual movement. Studies demonstrate that some of the same brain regions are active during MI sessions as during

functional tasks [2]. Though there has been vast research to improve this phenomenon, it is still very hard to understand the EEG signal. As feature extraction is one of the most significant steps in EEG signal classification, we have mainly focused on this stage.

1.3 Problem Statement

When the blood circulation to a region of the brain is disrupted, a life-threatening medical condition known as a stroke can occur. Strokes are a medical emergency requiring immediate treatment. The sooner a patient receives treatment for a stroke, the lower the probability of irreversible damage. During motor imagery also known as mental practice, mental imaging, or mental rehearsal, the brain system is triggered when a person imagines doing a task or moving their body without really doing so. Motor imaging has been applied after a stroke to treat the loss of arm, hand, and lower extremity movement, as well as to improve performance in daily living activities, boost mobility, and mitigate the effects of unilateral spatial neglect. Motor imagery can be utilized during the acute, subacute, and chronic stages of recovery. Motor imagery is beneficial on its own, but research indicates that it is most effective when paired with physical activity. In fact, most of the earliest studies on motor imagery focused on whether it improved athletic performance. According to imaging studies of the brain, same brain areas are activated during motor imagery and actual movement. Moreover, according to a study, motor imagery helps the brain reorganize its neural networks, which might benefit in the retraining of motor abilities after a stroke [3].

1.4 Objectives

The fundamental purposes of our study are given below:

- Study the existing systems for motor imagery signal classification.
- Apply suitable filters for best signal pre-processing.
- Apply proper feature extraction technique.
- Apply deep learning algorithms.

1.5 Scope of the Thesis

This system proposes a method to classify two class motor imagery EEG signals. This work includes two-class MI events by imaginary movements of the left hand and right hand. It combines linear and nonlinear features with advanced deep learning models.

Therefore, we develop a system where we can detect two motor imagery tasks: the imagination of movement of the left hand and right hand. For this thesis, we have limited our scope to developing a binary class classification to detect left-hand and right-hand movements. We did not inspect the scope of the system for multiclass classification.

1.6 Methodology

In this system, a suitable Butterworth band-pass filter (0.5–60 Hz) was employed to eliminate out-of-band noise. In addition, a 50 Hz notch filter was utilized to eliminate the remaining powerline noise. To make it easier to track future results, we normalized the entire database. In the step of feature extraction, linear and nonlinear features were extracted from EEG signals. The extracted linear features were Delta Average Band Power, Theta Average Band Power, Alpha Average Band Power, Beta Average Band Power, Gamma Average Band Power, and Theta To Beta Ratio (TBR). And the extracted nonlinear features were Sample Entropy, Dispersion Entropy, and MultiScale Sample Entropy. With the selected features of two-class motor imagery, EEG signals are classified using several machine learning algorithms such as deep neural network (DNN), artificial neural networks (ANN), random forest, and Bidirectional LSTM (BiLSTM).

1.7 Contributions

The main contributions of the thesis are as follows:

- Apply a suitable Butterworth band-pass filter and a 50 Hz notch filter for best signal preprocessing.
- Extract linear and nonlinear features using proper feature extraction technique.

- Apply deep learning algorithms, concerning the extracted features for binary class classification.
- Apply hyperparameter tuning for improving the accuracy

1.8 Thesis Organization

Chapter II: Provides a discussion of the existing works as well as their performance and the background study of this thesis work.

Chapter III: Describes the background and theoretical consideration.

Chapter IV: Provides methodology of our proposed system.

Chapter V: Illustrates results analysis and discussion of our study.

Chapter VI: We summarized the entire thesis, outlined our future plans, and concluded the thesis in this chapter.

Chapter II

Literature Review

In recent decades, the use of brain waves to control objects is a prevalent research direction. Many types of research have been done and have been proposed with different methods and established lots of efficient mechanisms to classify human body parts' movement from EEG signals. In the following sections, some considerable researches related to the task are mentioned and summarized.

2.1 Utilizing Rényi min-entropy-based feature selection from wavelet packet transformation

This paper proposes a novel feature selection method utilizing Rényi min-entropy-based algorithm for achieving a highly efficient brain–computer interface (BCI). This study employs a slightly modified version of the Rényi min-entropy-based technique for selecting features from WPT coefficients. In comparison to the Shannon entropy and mutual information technique, this method selects features from a vast feature set using a unique kind of entropy.

Wavelet packet transformation (WPT) is frequently used to extract features from electroencephalogram (EEG) recordings. In the case of multiple-class problems, classification accuracy is exclusively dependent on the selection of effective WPT features. Shannon entropy and mutual information techniques are frequently used to pick features in traditional procedures. In this study, the author demonstrated that the suggested Rényi min-entropy-based strategy for multiple EEG data categorization outperformed the standard approaches. For this experiment, the BCI competition-IV dataset (including 4-class motor imagery EEG signal) was utilized. Using WPT, the data were preprocessed and split into classes before feature extraction. The Shannon entropy, mutual information, and Rényi min-entropy approaches were then utilized for feature selection. Using the specified features and several machine learning

techniques, four-class motor imagery EEG data were categorized using the selected characteristics. Results indicated that the proposed strategy for multiple-class BCI was superior to standard methods.

As part of the proposal, 4-class motor imagery EEG data from the BCI competitionIV were acquired. Despite the fact that these signals were preprocessed, they were filtered using a 50-Hz notch filter and then filtered again to remove the EOG effect using the EWICA toolbox, as detailed in the preprocessing part of this paper. The EEG data were then evaluated for dual-tree WPT. Since the EEG signals were split according to the schedule of the previously described 4-class MI tasks, they were examined for feature extraction by WPT. In their work, EEG signals were divided into five levels and four distinct characteristics (Energy, Variance, Standard Deviation, and Waveform Length) were extracted using the WPT. According to the claims made in their study, not all of those features were required for the classifier. Therefore, correct features had to be selected from among them. The suggested study uses Rényi's min-entropy-based approach for feature selection to overcome the limitations of the conventional technique.

From the results, it was found that the proposed method outperforms 18% and 6% increment in classification accuracy (on average) than the Shannon entropy and mutual information methods, respectively, in the case of the SVM classifier. On the other hand, applying the random forest and MLP-ANN the classification accuracy could be increased up to 8% with respect to mutual information methods [4].

2.2 Using K-Nearest Neighbor

This paper [5] proposes a classification method for motor imagery tasks-based brain computer interface (BCI). The wavelet coefficients were utilized to extract features from the motor imagery electroencephalographic (EEG) data, and the k-nearest neighbor classifier was employed to categorize the pattern of left or right-hand imagery movement and rest.

At first, they extracted the signals of interest (the signal corresponding to the right/left-hand motor imagery tasks and the signal corresponding to the relaxation) from the acquired signal and then they performed a multiresolution wavelet analysis using those two types of mother wavelet, Coiflet4, and Daubechies2. The components of the features matrix were selected

from the wavelet coefficients of interest. This features matrix was computed for the training set and the test set of the signals.

Classification was conducted between two categories: relaxation and imagined motion. They determined the optimal value of k to optimize classification performance by employing the Euclidean distance measure. The k value was searched in the interval between 1 and 5, using a 1-step size. They classify all of the subjects, and the results are reported as a percentage of precision. BCI2000 has several data conversion utilities, and a script to load BCI2000 data files directly into MATLAB. MATLAB was used to develop a k-NN classification program. The k-NN classifier achieved higher classification accuracy than the LDA classifier of the BCI2000 platform, as displayed at the conclusion of the testing paradigm. C3 and C4 channels exhibited the highest categorization rates.

The results obtained with the KNN classifier were better to those obtained with the BCI2000 software. The approach was applied to 20 participants, and the best classification performance of the suggested method ranged between subjects from 68% to 91%. Significant subject-to-subject variation in classification performance was observed; currently, there is no optimal selection of the mother wavelet and k across all subjects.

The classifier used in BCI2000 can be improved to obtain better accuracy and must be subject oriented [5].

2.3 Using Convolutional Neural Network Framework

In this paper [6], a transfer CNN framework based on VGG-16 is proposed for MI EEG signal classification. It comprises a CNN model that has been pre-trained and a CNN model that is utilized for MI classification. First, STFT was used to transform the raw EEG data from the C3, C4, and Cz electrodes into time-frequency spectrum pictures.

On ImageNet, a VGG-16 CNN model was then pretrained. Finally, the target CNN model for MI classification was fine-tuned on the target dataset. In this study, the target dataset was comprised of converted EEG temporal frequency spectrum pictures. Due to the dissimilarity between the target dataset used for fine-tuning and the ImageNet dataset used for pre-training, only the early layers of the target CNN were frozen and the later layers were fine-tuned. Experimental findings show that the suggested framework enhanced the accuracy

and efficiency of EEG signal categorization in comparison to conventional approaches, such as support vector machine (SVM), artificial neural network (ANN), and standard CNN.

EEG signals from the C3, C4, and Cz electrodes can be significantly changed by doing MI activities, according to [7] and [8] . During MI tasks, there is a drop in amplitude in the mu band (8-13 Hz) of these EEG signals, a phenomenon known as event-related desynchronization (ERD). In contrast, there is an increase in amplitude in the beta band (13-30 Hz) due to event-related synchronization (ERS). In consideration of this, STFT was applied to the signal recordings from C3, C4, and Cz electrodes, and only the mu and beta frequency regions of the resulting time-frequency spectra are kept. Furthermore, 4-14 Hz frequency bands were utilized to represent the mu band and 16-32 Hz frequency bands were used to represent the beta band since this yielded superior performance in their verification trials.

The proposed framework consisted of a pre-trained VGG-16 CNN model and a target CNN model, where the pre-trained VGG-16 CNN model was used to extract universal features for common image classification tasks and the target CNN model aimed to classify EEG signals efficiently and accurately using the pre-trained VGG-16 CNN.

Despite the progress made in this article, two constraints must be addressed in the future. The transferability of the different layers of the pre-trained CNN model has not been investigated in this research. Even with GPU-accelerated computation, training the suggested framework is still a time-consuming procedure [6].

2.4 Using PCA, Wavelet and Two-Stage Neural Network

This study [9] proposes a novel technique for classifying four-class MI EEG signals based on the combined use of principal component analysis (PCA), wavelet packet transformation (WPT), and a two-stage machine learning algorithm. From this proposed work, it had been determined that not only is the extraction of novel features necessary, but also the configuration of classifiers with an acceptable strategy was a worry. Although the feature extraction method employing PCA-based wavelet packet transformation was an outstanding technique for determining the features of the EEG signal, it failed to accurately identify the four-class motor imagery signals. In contrast, the suggested two-stage classifier boosted classification accuracy by an astounding 16.34% percent on average. This strategy was applicable to any

classifier with more than two classes. Therefore, their proposal surpassed the usual BCI implementation method.

Popular EEG signal feature extraction techniques include autoregressive (AR) approaches, wavelet transformations (WT) methods, phase-space reconstruction approaches, CSP-based methods, empirical mode decomposition, etc. Although WPT coefficients are commonly utilized for EEG signal classification, WPT-based feature extraction has two significant limitations for multiclass classification: organizing the features and selecting the bases. In this paper [9], they presented a technique that employs a principal component analysis (PCA) to decrease signal dimension, a wavelet packet transform (WPT) to extract its features, and a two-stage artificial neural network (ANN) for identifying the signal's lobe-origin and limb-origin.

But one drawback of the proposed multi-stage training-based classifier is its time requirement for the training stage [9].

2.5 Improved BP Neural Network based recognition and analysis of MI EEG Signal

The main purpose of this research [10] was to convert the individual motor-imagined EEG signals into two-dimensional motion instructions on the computer screen and corresponding mouse cursor movement control instructions.

In this article [10], weight-splitting technology was introduced to the classic BP neural network algorithm based on an upgraded BP neural network algorithm. To overcome the filtering problem, a non-linear mapping function of a conventional BP neural network was employed, and to improve the whole BP algorithm, tiny weight particles were intelligently trained by merging the particle swarm filter method. By combining the upgraded BP neural network algorithm with particle swarm filter algorithm, the problem of poor signal-to-noise ratio (SNR) and unclean filtering in EEG data processing caused by rapid weight degradation in the classic BP method was resolved.

This paper's [10] basic algorithm was the BP algorithm, and its related auxiliary algorithm

was the particle cluster filtering algorithm placed in the hidden layer, which primarily handled the filtering problem during the system identification phase. In terms of recognition and analysis of motor imagery brain wave signals, the suggested algorithm outperformed the conventional BP algorithm.

But in this work, the convergence and accuracy of the recognition and analysis algorithm for motor imagery EEG still have a problem that cannot be acquired at the same time. [10].

2.6 Using of CNN with multilevel weighted feature fusion

In this research [11], the author used EEG motor imaging data to demonstrate the advantages of extracting and merging multilevel convolutional features from distinct CNN layers, which were abstract representations of the input at different levels. The CNN model developed here could learn robust spectral and temporal characteristics from raw EEG data. The multilayer feature fusion outperformed models that utilized only the last layer's characteristics.

The author proposed a methodology for multilayer feature extraction and fusion for EEG MI data using a CNN that had been pretrained as a feature extractor. The construction of a robust feature representation for EEG signals utilizing information hidden in CNN layers. Some blocks of convolutional and max pooling layers were followed by fully connected layers towards the conclusion of the CNN model's layers. The first convolution consists of two layers. They utilized two unique regularization techniques, namely dropout and batch normalization. The optimal size of retrieved features for fusion was determined using weight-based feature fusion. This suggested study achieved 74.5% accuracy, which is more than existing approaches for the BCI dataset. However, the CNN-based multilayer feature fusion algorithms have yet to be evaluated on other EEG datasets. [11].

2.7 Scope of Improvement in Existing Works

All these studies discussed so far have some significant contributions related to the classification of motor imagery EEG signals. But there are still areas of improvement in these work studies. An overview of these researches with corresponding techniques, dataset, and limitations have been depicted in Table. 2.1.

Table 2.1: Brief Summary of the related work.

Authors	Techniques	Dataset	Limitations
Roxana Aldea, Monica Fira, Anca Laz ar [5]	Wavelet Co-efficient, K-Nearest Neighbors	BCI data recordings	can be improved to obtain better accuracy
Zhiwen Zhang, Feng Duan, Jordi Sole-Casals [12]	Convolutional Neural Network, Wavelet Neural Network	Experimental motor imagery EEG data and dataset III from BCI Competition II	Low computational efficiency of WNN (wavelets neural network)
Gaowei XU, Xiaoang Shen, Sirui Chen[13]	Deep Transfer Convolutional Neural Network Framework	2b from the BCI competition and IV	Time-consuming process
Jamal F. Hwaidi, Thomas M. Chen[14]	Deep Autoencoder and Convolutional Neural Network Approach	Physionet dataset	Number of electrodes can be reduced even further
Nuri Korhan, Zumray Dokur, Tamer Olmez[15]	Common Spatial Patterns and Convolutinal Neural Networks	BCI Competition III dataset 3a	Needed to combine CNN with the methods specifically designed for BCI tasks
Pawel Herman, Girijesh Prasad, Thomas Martin McGinnity, Damien Coyle [16]	Spectral approaches, SVM, RFD, LDA	datasets from two different BCI laboratories, the Graz and ISRC	Less practical in real-world BCI application

Chapter III

Theoretical Considerations

In this chapter, we review some of the theoretical concepts which helped us to develop the methodology for our proposed system.

3.1 Electroencephalogram (EEG)

EEG stands for electroencephalography, which is a non-invasive method of measuring brain activity. It works by recording the electrical activity of the brain through electrodes placed on the scalp. These impulses are picked up by electrodes that are placed on the scalp and are then amplified and recorded. The resulting EEG recording is a graph of the electrical activity of the brain over time, which can be analyzed to reveal information about brain function.

EEG is commonly used in the diagnosis of neurological disorders such as epilepsy, as well as for research into brain function and behavior, such as how the brain processes information, responds to different stimuli, and generates consciousness. It can also be used to guide brain-computer interfaces and other brain-computer interactions.

EEG can also be used in the field of brain-computer interfaces (BCI). BCI is a technology that allows people to control devices or communicate with others using only their brain activity. EEG is often used to provide the brain signals that are used to control these devices because it is non-invasive and can be used to measure a wide range of brain activity.

It is important to note that EEG is not a perfect tool and has some limitations like the electrodes placed on the scalp only provide information about the brain activity on the surface of the brain and not in the deeper structures. Also, the signals recorded by EEG are quite weak and are easily influenced by various factors such as muscle activity, eye movements, and other types of noise.

EEG recordings are typically represented as a graph, with the horizontal axis representing time and the vertical axis representing the amplitude of the electrical activity being recorded. The amplitude is usually measured in microvolts (μV) [17].

3.1.1 Types of Signal

The typical EEG recording is composed of several different types of waves, each of which is associated with a different level of brain activity. There are five types of waves which are shown in the below table 3.1:

Table 3.1: Frequency and Mental States of Waves.

Wave	Frequency(Hz)	Mental State
Delta (δ)	0-4	Deep Sleep
Theta (θ)	4-8	Drifting Thoughts, Dreams, Creativity
Alpha (α)	8-13	Calmness, Relaxation, Abstract Training
Beta (β)	13-30	Highly Focused, Highly Alertness
Gamma (γ)	>30	Simultaneous Process, Multi-Tasking

3.1.2 Delta Waves

Delta waves are under the frequency range of 0 – 4 Hz. Mental states associated with these waves are deep sleep, coma or hypnosis, and sometimes awake. In an awake state, it is always considered to be a pathological phenomenon. The higher the amplitude, higher serious the problem is considered. These waves are decreased by age and are normally present in healthy people in their awake state [17].

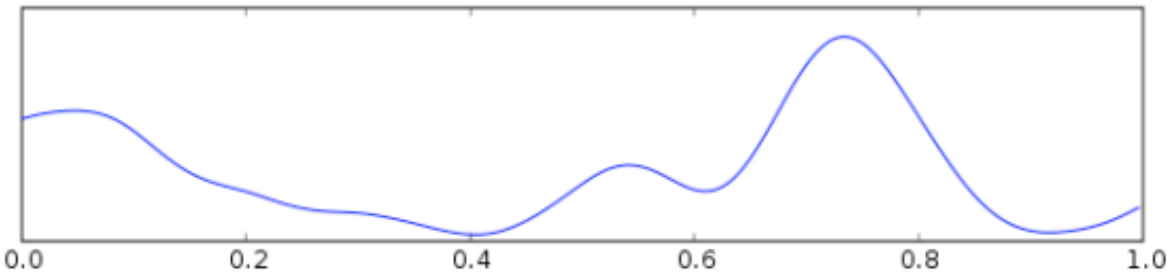


Figure 3.1: Delta Wave.

3.1.3 Theta Waves

Theta waves are under the frequency range of 4 – 8 Hz. Mental states associated with these waves are drifting thoughts, creative thinking, and unconscious materials. These waves appear in the central, temporal, and parietal parts of the head. These waves are normally present in healthy people while they are in deep sleep [17].

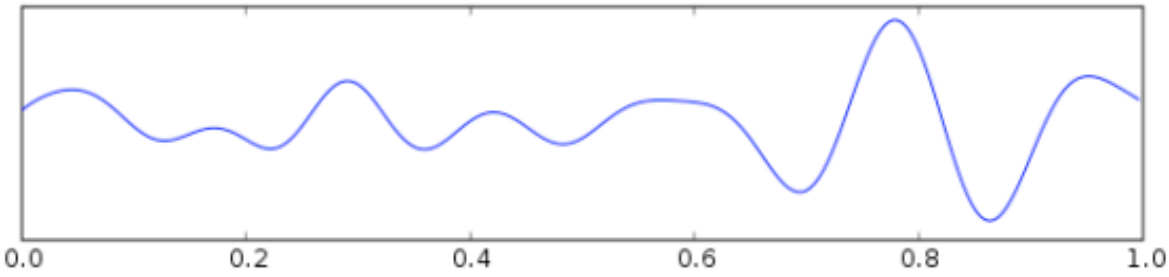


Figure 3.2: Theta Wave.

3.1.4 Alpha Waves

Alpha waves are under the frequency range of 8 – 13 Hz. Mental states associated with these waves are relaxed and calm states. These waves appear on the back side of the head and occipital area of the brain. These waves are of high amplitude as compared to others. This can be observed while the subject is awake and calm. Sometimes, these waves interfere with μ-rhythm. These waves are normally present in people while they are calm and relaxed being in an awake state [17].

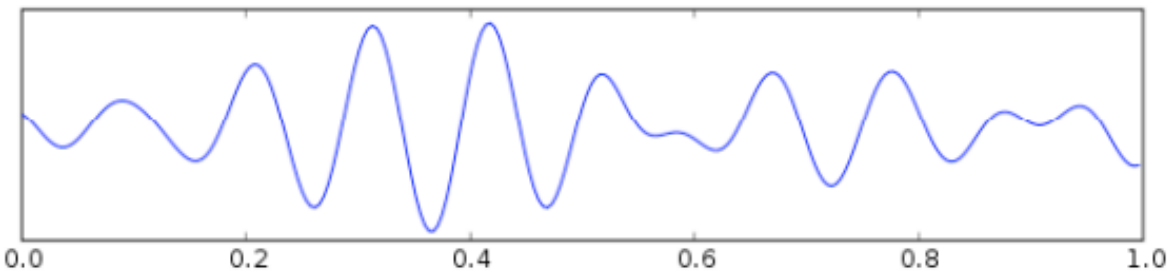


Figure 3.3: Alpha Wave.

3.1.5 Beta Waves

Beta waves are under the frequency range of 13 – 30 Hz. Mental states associated with these waves are highly focused and alert, such as during deep thinking and concentration. Beta

waves are having a large band of frequency as compared to others. These waves appear at the front side of the head and central area of the brain [17].

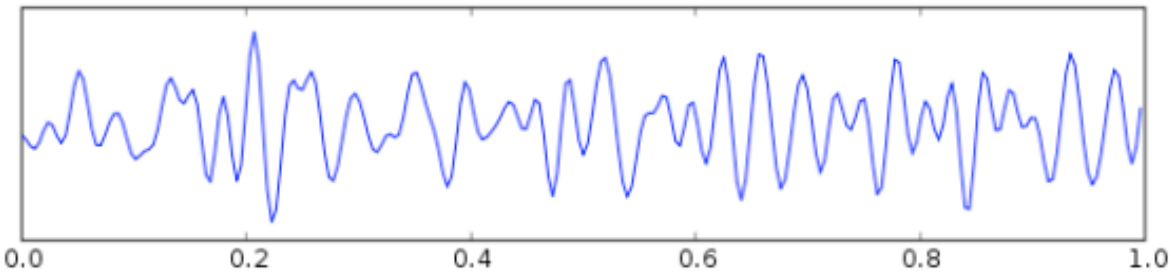


Figure 3.4: Beta Wave.

3.1.6 Gamma Waves

Gamma waves are under the frequency range of 30 Hz. Mental states associated with these are simultaneous work and multitasking. These waves are hard to notice due to their very low amplitude. These waves appear in each part of the brain [17].

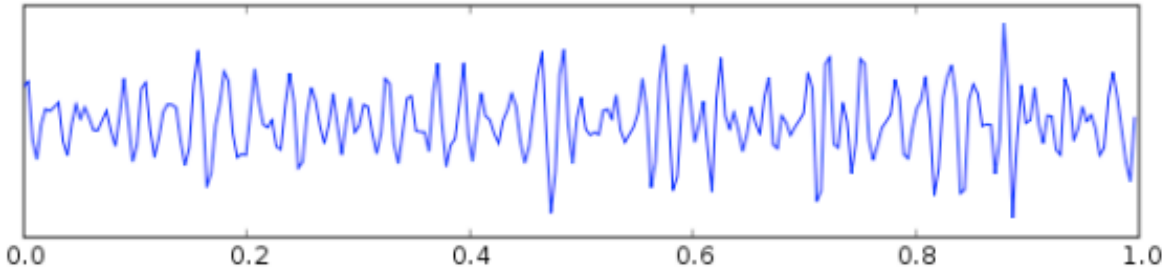


Figure 3.5: Gamma Wave.

3.1.7 Types of Signal based on analysis method

There are several types of EEG figures that can be generated from EEG recordings, depending on the specific analysis method used. Here are a few examples:

- **Time series:** This is a simple graph that shows the electrical activity of the brain over time. The x-axis represents time, and the y-axis represents the amplitude of the electrical activity. This figure can be used to visualize changes in brain activity over time, such as changes in response to a specific stimulus or changes in activity during different stages of sleep.

- **Spectral analysis:** This figure shows the power (or amplitude) of different frequencies of brain activity. The x-axis represents frequency, and the y-axis represents power. This figure can be used to visualize changes in brain activity across different frequencies, such as changes in alpha or beta activity.
- **Topographic map:** This is a two-dimensional map that shows the distribution of brain activity across the scalp. Each point on the map represents an electrode location, and the color or shading at each point indicates the amplitude of the electrical activity at that location. This figure can be used to visualize differences in brain activity across different regions of the scalp, such as changes in activity in the left versus the right hemisphere.

3.2 Motor imagery

Motor imagery is the mental process of rehearsing or simulating a certain movement. It has been used as a study paradigm in cognitive neuroscience and cognitive psychology to explore the content and organization of covert processes (i.e. unconscious) that precede the execution of an action. In some medical, musical, and athletic circumstances, mental rehearsal can be as successful as pure physical rehearsal (practice) of action when combined with physical rehearsal. Motor imaging is a dynamic condition in which an individual replicates a physical action in his or her mind. This sort of phenomenal experience suggests that the individual feels as though they are executing the action. It relates to what sports psychologists refer to as internal imagery (or first-person perspective).

Mental practice refers to the use of visuo-motor imagery to enhance motor performance. Without physical movement, visuo-motor imagery involves the use of one's mind to recreate an activity. It has gained prominence due to the importance of images in improving athletic and surgical performance. Numerous functional neuro-imaging investigations have shown that motor imagery is connected with the particular activation of brain circuits involved in the earliest stage of motor control (i.e., motor programming). This circuit consists of the supplementary motor area, primary motor cortex, inferior parietal cortex, basal ganglia, and cerebellum. Such physiological evidence provides substantial support for the notion that imaging and motor preparation share brain processes. During motor imagining and real

motor performance, measurements of cardiac and respiratory activity demonstrated a correlation between heart rate and lung ventilation, and the level of imagined exertion. Motor imagery engages the motor pathways. During motor imagining, muscular activity generally rises relative to resting levels. When this is the case, EMG activity tends to be proportionate and restricted to the muscles involved in the mimicked motion. [18].

3.2.1 Effects

Motor imaging is currently widely used as a therapy to improve motor learning and stroke patients' neurological rehabilitation. Musicians have shown their usefulness.

- **On motor learning:** Motor imagery is a recognized method of training for athletes. Typical training includes a warm-up, relaxation, and focus, followed by mental modeling of the specific exercise.
- **In neurological rehabilitation:** Motor imagery increases the effects of traditional physiotherapy and occupational treatment. A recent analysis of four randomized controlled studies suggests that there is limited evidence to demonstrate the added advantage of motor imagery above traditional physiotherapy alone for stroke patients. The authors found that motor imagery looks to be an appealing therapy option, as it is simple to learn and use, and the intervention is neither physically taxing nor damaging. Consequently, motor imagery may provide further benefits to patients.
- Motor imagery can serve as a substitute for imagined behavior to produce similar cognitive and behavioral consequences. The frequent simulated consumption of food, for example, can decrease subsequent actual consumption of that item.

3.2.2 Simulation and understanding mental state

Motor imagery is similar to the cognitive and social neuroscience concept of simulation, which is used to explain many processes. A participant in simulation may relive his own prior experience to derive from it enjoyable, motivating, or just informative aspects. The simulation hypothesis asserts that thinking consists of simulated interaction with the environment and is founded on the following three fundamental assumptions:

- **Simulation of actions:** We can activate motor structures of the brain in a way that resembles activity during a normal action but does not cause any overt movement.
- **Simulation of perception:** Imagining perceiving something is essentially the same as actually perceiving it, only the perceptual activity is generated by the brain itself rather than by external stimuli.
- **Anticipation:** There exist associative mechanisms that enable both behavioral and perceptual activity to elicit other perceptual activity in the sensory areas of the brain. Most importantly, a simulated action can elicit perceptual activity that resembles the activity that would have occurred if the action had been performed.

Mental stimulation may also serve as a representational method for comprehending oneself and others. Philosophy of mind and developmental psychology both use simulation to explain our ability to mentalize, i.e., to comprehend the mental states (intentions, wants, emotions, and beliefs) of others (aka theory of mind). In this context, the fundamental concept of simulation is that the attributor employs his psychological resources to imitate the mental activity of the target. The individual imagines herself/himself performing the same action to comprehend the mental state of another when observing the other's behavior; this is a covert simulation that does not result in overt behavior. Critical to the simulation theory of mind is the notion that to attribute mental states to others, an attributor must set aside her mental states and substitute the target.

3.3 Techniques to Classify EEG Data

Various techniques are used to classify EEG data and some of them are successful but lack in accuracy. However, the following techniques we have considered for our work:

- Deep Neural Network (DNN)
- Artificial Neural Network (ANN)
- Random Forest Regressor (RFR)
- K-Nearest Neighbor(KNN)
- Bidirectional Long Short Term Memory (BiLSTM)

3.3.1 Deep Neural Network (DNN)

Deep learning is a machine learning method that enables computers to learn by example in the same way that humans do. Deep learning models can attain state-of-the-art accuracy, even surpassing human performance in some cases. These models are sometimes referred to as deep neural networks because most deep learning approaches use neural network designs. The term "deep" refers to the number of hidden layers in the neural network. Deep neural networks can have up to 150 hidden layers, whereas traditional neural networks only have $2-3$. Large sets of labeled data and neural network architectures that learn features directly from the data without the requirement for manual feature extraction are used to train deep learning models.

Biological Neuron to Artificial Neuron

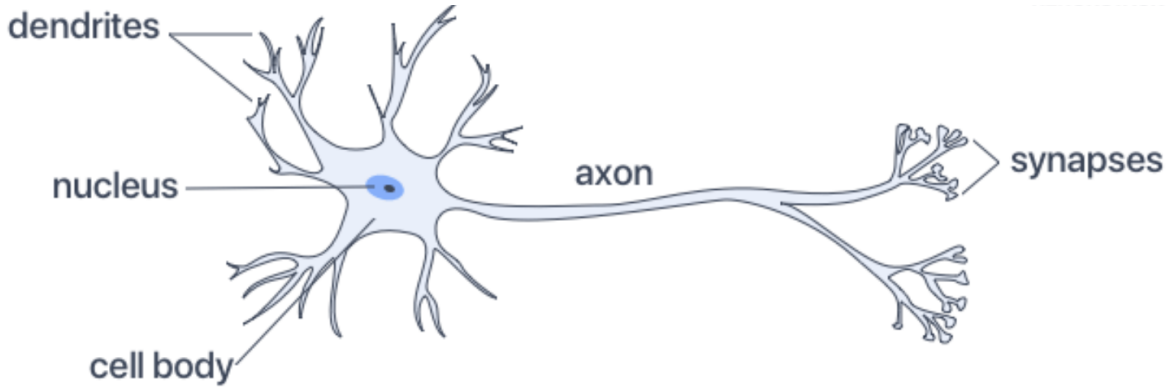


Figure 3.6: Biological neuron.

The relevance of a biological neuron with the concept of the artificial neuron is stated in [19]. A neuron (or nerve cell) is a particular biological cell that has the ability to process information and is the essence of life. A biological neuron in sketch form is depicted in Figure. 3.6. The axon and dendrites are two types of out-reaching tree-like branches that make up the cell body or soma. The cell body has a nucleus that stores information about inherited features and a plasma that houses the molecular machinery that allows the neuron to produce the material it requires. A neuron's dendrites (receivers) receive signals (impulses) from other neurons and transmit signals generated by its cell body via the axon (transmitter), which eventually forks into strands and substrands. Synapses are located at the ends of these strands. A synapse is a basic structure and functional unit that connects two neurons (an

axon strand from one neuron and a dendrite from another). When an impulse reaches the synapse's terminal, neurotransmitters are released. The neurotransmitters diffuse across the synaptic gap, enhancing or inhibiting the receptor neuron's own inclination to emit electrical impulses, depending on the type of synapse. The signals going through the synapse can change its effectiveness, allowing synapses to learn from the activities in which they engage. This reliance on history functions as a memory, and it is thought to be the source of human memory.

Generalized Model of an Artificial Neuron

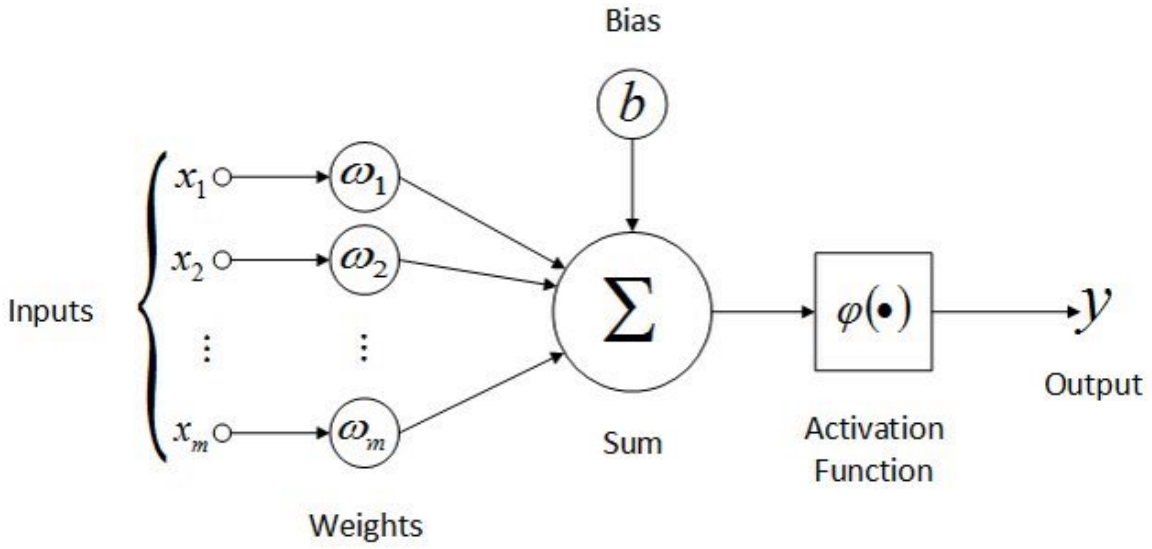


Figure 3.7: Generalized model of an artificial neuron.

A generalized type of artificial neuron is depicted in Figure. 3.7, which includes a set of connecting weights, a summing unit, and an activation function. Each input signal is weighted, or multiplied by the weight value of the associated input (similar to the synaptic strength of real-life neuron connections). As a result, the summing unit's output is a mixture of weighted input signals plus an externally provided bias. Depending on whether the bias is positive or negative, it has the effect of boosting or decreasing the net input of the activation function. Finally, the activation function is responsible for a neuron's output. The output of a neuron before passing to an activation function can be expressed mathematically as follows:

$$z = \sum_i w_i x_i + b \tag{3.1}$$

where, w_i, x_i and b denote weights, feature vector and bias to the neuron respectively. If $\phi(\cdot)$ represents the activation function, the final output will be as follows:

$$y = \phi(z) \tag{3.2}$$

Activation Function

An Activation Function determines whether or not a neuron is activated. It uses simpler mathematical operations to determine whether the neuron's input to the network is essential or not throughout the prediction phase. It specifies how a node or nodes in a network layer turn the weighted sum of the input into an output. The activation function chosen has a significant impact on the neural network's capabilities and performance, and different activation functions may be utilized in different portions of the model.

An activation function is used to give a neural network more non-linearity. Without the activation functions, a neural network will merely execute a linear transformation on the inputs using the weights and biases. Because the composite of two linear functions is a linear function, it doesn't matter how many hidden layers we add to the neural network; all layers will act the same. So, learning any complex task would be impossible without an activation function.

i. Rectified Linear Unit (ReLU)

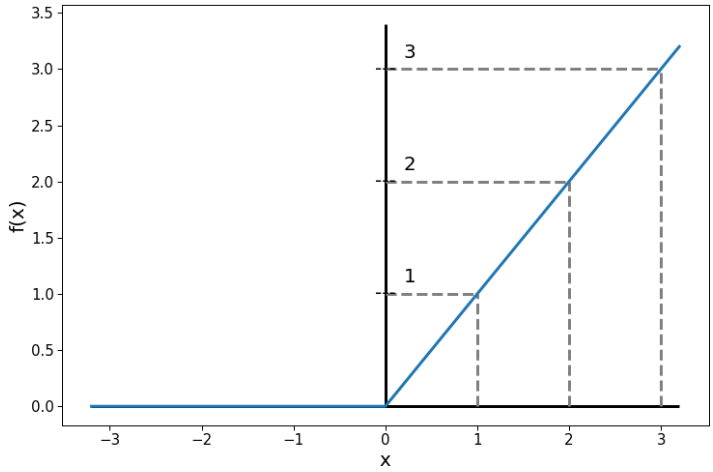


Figure 3.8: Rectified linear unit.

The ReLU (Rectified Linear Unit) activation function is a commonly used activation function in deep neural networks (DNNs) due to its simplicity, speed, and effectiveness.

The function is defined as follows:

$$f(x) = max(0,x), \text{ where x is the input} \tag{3.3}$$

It returns 0 if the input is negative and the input itself if positive. This piecewise linear function is not differentiable at x=0, but that does not impact training in DNNs as the optimization algorithm used does not rely on the derivative of the activation function.

ReLU has been used in successful DNN models, including AlexNet and VGG, and has been found to be effective in tasks such as image classification and speech recognition. However, it has the limitation of the "dying ReLU" problem, where a neuron can become inactive and remain so, leading to reduced performance.

ii. Tanh

The hyperbolic tangent function, or tanh, is a common activation function used in artificial neural networks. It is a smooth, nonlinear function that maps its input to the range of -1 to 1.

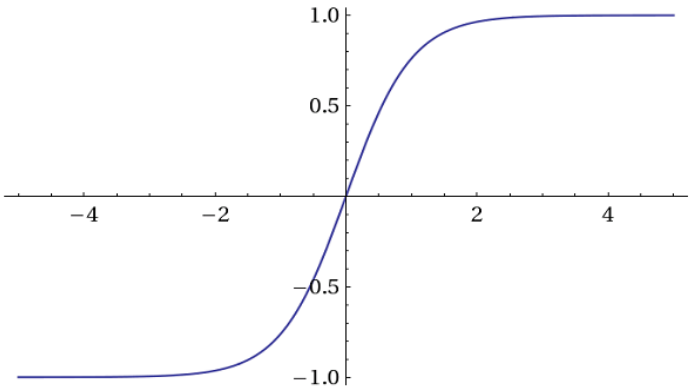


Figure 3.9: Tanh Activation Function.

The formula for the tanh function 3.4 is:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.4}$$

where e is the mathematical constant e (approximately equal to 2.71828) and x is the input to the function.

The tanh function is similar to the sigmoid function, which also maps its input to a range between 0 and 1, but the tanh function is centered at 0, meaning that its outputs are symmetric around 0. This property can be useful in certain types of neural networks.

One advantage of the tanh function over the sigmoid function is that it has a steeper gradient around the origin, which can help speed up learning in neural networks. However, the tanh function can suffer from the same vanishing gradient problem as the sigmoid function when the inputs to the function become very large or very small, which can make it difficult to train deep neural networks using this activation function.

Overall, the tanh function is a commonly used activation function in neural networks, but its use may depend on the specific problem and architecture being used.

iii. Sigmoid

Sigmoid function is used for binary classification. It introduces non-linearity to the output layer.

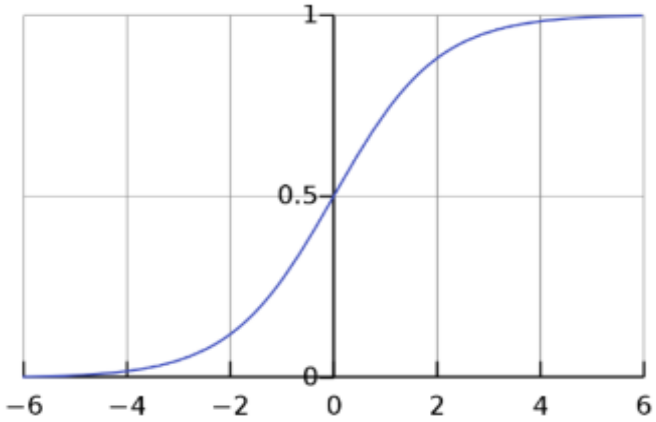


Figure 3.10: Sigmoid Function.

The function is defined as follows:

$$f(z) = \frac{1}{1+e^{-z}} \tag{3.5}$$

Other than these activation functions, there are many other activation functions. Among them, Softmax is also a very popular activation function and is used to get probability distribution over predicted output classes and in multinomial logistic regression. In the proposed CNN model, ReLU activation function is used in the hidden convolutional layer and Sigmoid is used in the dense layer to get the binary classification.

Multilayer Feed-Forward Network

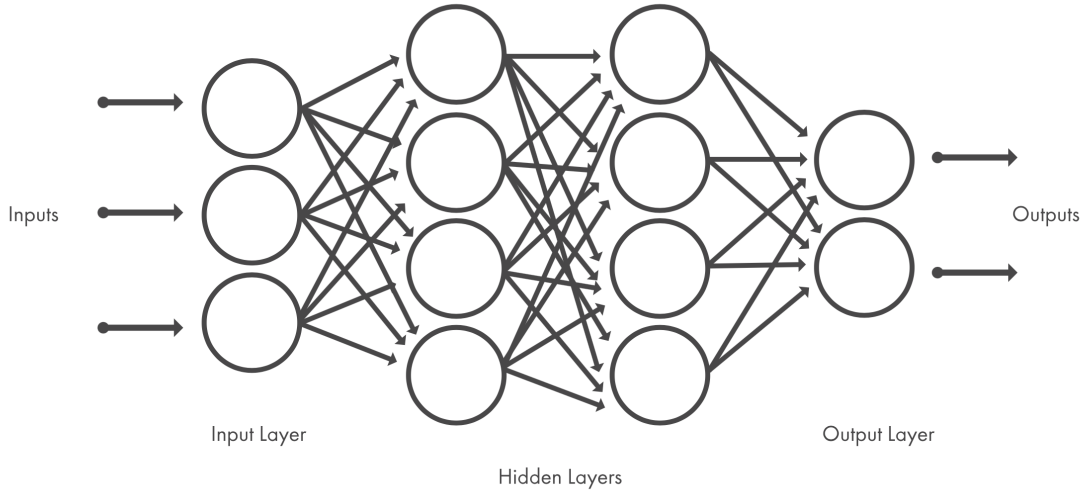


Figure 3.11: Multilayer feed-forward network.

The most prevalent type of artificial neural network (NN) architecture for solving real-world issues is a multilayer feed-forward neural network with several layers of neurons. The typical construction of this network is depicted in Figure. 3.11. The artificial neurons, or units, are arranged in layers, with each unit in a layer having all of its inputs connected to the units of the preceding layer (or to the inputs from the external world in the case of the units in the first layer), but none to the units in the same layer to which it belongs. The layers are stacked one after another, with an input layer, many intermediate layers, and an output layer in between. The intermediate layers, often known as hidden layers, have no input or output to the outside world. In general, the input layer is thought of as a signal distributor from the outside world. There are no restrictions on the number of hidden layers or neurons in the hidden layers. The objective of a hidden layer is to improve a network's functional versatility. The number of neurons in the hidden layer should grow in tandem with the intricacy of the link between the input data and the intended output.

Despite the fact that ANN is capable of simultaneous feature extraction and classification, it is still challenging to directly apply ANN to raw EEG data due to their susceptibility to measurement noise and interferences [20].

Deep Neural Network Training

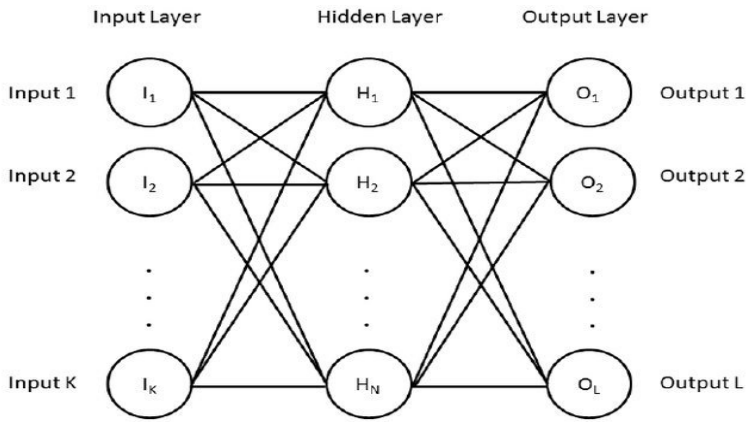


Figure 3.12: A three layer feed-forward network.

A good architecture as well as sufficient weight set values are required to perform a NN for a specific task. As previously said, the architecture of the network is mostly determined by the problem. However, in the field of NN, obtaining a weight set is critical and difficult because the same architecture performs multiple functions for different weight values [19]. The weight values may assign directly for extremely small problems (such as ordinary binary logic), but this is impractical for huge real-world situations. In this aspect, ANN has a natural proclivity for storing experimental knowledge in synaptic weights, which it acquires through a learning process from its surroundings [21].

The goal of NN learning is to change the synaptic weights of the network so that after learning, it correctly predicts a specific pattern from its input set of features [22]. Training is the method utilized to carry out the learning process. The goal of a training algorithm is to adjust the synaptic weights of a NN in an ordered method to achieve a particular target, because synaptic weights hold the information. For multi-layered NNs, back-propagation (BP) [23] is the most prevalent NN training approach. When output errors propagate from the output layer to the input layer in BP, synaptic weights are changed. To use BP, a network's connection weights are set to random values in a small range.

To demonstrate the BP method, we are considering the most popular three-layer NN architecture, as shown in Figure. 3.12. A forward pass and a backward pass are the two basic steps in the BP algorithm. In the forward pass, an example or pattern's input values are presented to the network, actual outputs are measured from the output layer by passing responses from the input layer to the output layer through the hidden layer, and then the pattern's error is calculated based on the pattern's actual output and desired output. The connection weights are modified in the backward pass based on the estimated error. The hidden and output layer weights are modified first, followed by the input and hidden layer weights.

According to BP learning, if a weight w transmits input x to a neuron and f is the neuron's output, the weight correction (w) for that weight is provided by the equation:

$$\Delta w = \eta \ \delta \ x \tag{3.6}$$

Here, η denotes the learning rate and δ refers to the local gradient of the neuron. Learning speed is influenced by the learning rate, which simply displays the proportional size of weight changes. Because of its high value, an update due to one example might significantly modify a weight value's oscillation with regard to others. In typically, the value of η takes into account a modest range of values, such as between 0.1 and 0.3.

The output unit's δ_o and hidden unit's δ_h local gradients are specified as follows:

$$\delta_o = -\frac{\partial e}{\partial f_o}\frac{\partial f_o}{\partial x_o} \tag{3.7}$$

$$\delta_h = \sum_o \delta_o w_o \frac{\partial f_h}{\partial x_h} \tag{3.8}$$

Here, the net input (weighted total) and output of an output neuron are represented by x_o and f_o, respectively. The discrepancy between the desired output and the actual response is defined as e. The following equation can be used to define the error function for the n-th training pattern.

$$e(n) = \frac{1}{2}(d(n) - f_o(n))^2 \tag{3.9}$$

Here, $d(n)$ is the desired output, while $f_o(n)$ is the actual output. The BP algorithm requires the partial derivative of (3.9) with regard to the output $f_o(n)$ to update weights, which is

derived as follows.

$$\frac{\partial e_o(n)}{\partial f_o(n)} = -(d(n) - f_o(n)) \tag{3.10}$$

The local gradient of the output unit can be rewritten as follows:

$$\delta_o = \begin{cases} (d(n) - f_o(n)), & \text{if } x_o(n) \geq 0, \\ (d(n) - f_o(n))\dfrac{f_o(n)}{x_o(n)}, & \text{otherwise.} \end{cases} \tag{3.11}$$

and the local gradient for the hidden unit becomes: The local gradient of the output unit can be rewritten as follows:

$$\delta_h = \begin{cases} \sum_o \delta_o w_o, & \text{if } x_h(n) \geq 0, \\ \dfrac{f_h(n)}{x_h(n)} \sum_o \delta_o w_o, & \text{otherwise.} \end{cases} \tag{3.12}$$

The NN's operational flowchart is shown in Figure. 3.13[19]. The forward pass from input to actual output is shown in the upper portion of the figure, followed by error. The local gradient and weight correction generation from the output layer and input layer is shown in the lower section. Every arrow points to the component that must be used to calculate it. According to (3.11), the local gradient of the output layer (i.e., δ_o) necessitates actual output f_o, net input (weighted total) x_o and desired output d_o. Local gradient of hidden layer (i.e., δ_h) requires hidden layer output f_h, net input (weighted total) of hidden neuron x_h, local gradient of output layer (i.e., δ_o), and connecting weights of individual hidden and output nodes w_o, according to (3.12).

The number of neurons in the hidden layer in a NN architecture is determined by the problem difficulty. Additionally, many hidden layers with different numbers of neurons could be used for a better result. In that instance, additional steps will be taken to update the NN by computing local gradients of more hidden layer units, but the calculations will be done in a similar manner. Assuming the architecture of Figure. 3.12 has two hidden layers. In that

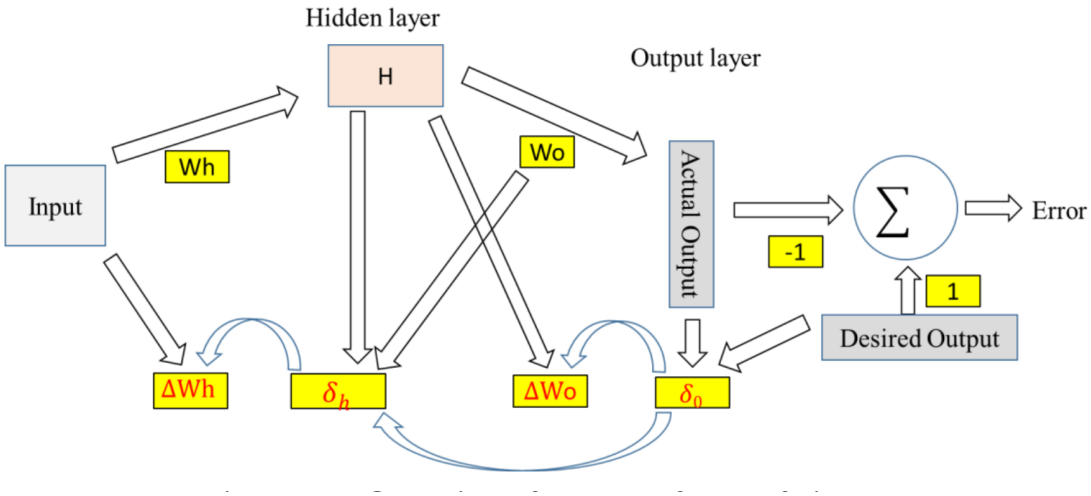


Figure 3.13: Flow chart of operations of a neural network.

case, the local gradient for the second hidden layer would be as follows:

$$\delta_{h2} = \begin{cases} \sum_o \delta_o w_o, & \text{if } x_{h2}(n) \geq 0, \\ \dfrac{f_{h2}(n)}{x_{h2}(n)} \sum_o \delta_o w_o, & \text{otherwise.} \end{cases} \tag{3.13}$$

and the local gradient for the first hidden layer would be as follows

$$\delta_{h1} = \begin{cases} \sum_{h2} \delta_{h2} w_{h2}, & \text{if } x_{h1}(n) \geq 0, \\ \dfrac{f_{h1}(n)}{x_{h1}(n)} \sum_o \delta_o w_o, & \text{otherwise.} \end{cases} \tag{3.14}$$

Loss function

How well an algorithm reflects a dataset is determined by its loss function. The loss function will return a bigger value if the forecasts are entirely inaccurate. If the projections are reasonably correct, the output will be lower. Adjusting the different layers and parameters of the CNN model in an effort to enhance the model can disclose whether or not it is heading in the right direction based on the loss function.

In a deep neural network (DNN) for binary classification, binary cross-entropy loss is the most popular loss function. This loss quantifies the gap between the projected probability distribution and the actual label distribution. Minimizing this loss function using optimization techniques such as stochastic gradient descent (SGD) optimizer trains the DNN to

accurately anticipate the class label.

Optimizer

Optimizers are utilized to modify the neural network's properties, such as weights and learning rate, to minimize losses. Finding the appropriate weights for a deep learning model is a challenging task due to the model's often millions of parameters. The application requires the selection of a suitable optimization algorithm. Optimizers are utilized to resolve optimization issues by reducing function parameters. Therefore, the accuracy is improved and the overall loss is diminished.

Adam

Adaptive moment estimation [24] is the source of the Adam optimizer. This optimization strategy is an extension of stochastic gradient descent for updating network weights throughout training. Adam optimizer, unlike SGD, alters the learning rate at each network weight independently, rather than maintaining a single learning rate throughout training. Adam optimizers inherit both the Adagrad and the RMS prop algorithms. Adam adjusts learning rates by using the second moment of the gradients, as opposed to the first moment (mean) as in RMS Prop.

Early stopping

Early stopping is a type of regularization used in machine learning to reduce overfitting while training a classifier using an iterative method such as gradient descent. These strategies improve the classifier with each iteration so that it better fits the training data.

If there are too many epochs, the model may be overfit to the training data. In contrast, if the number of epochs is insufficient, the resulting model will be underfitted. Early stopping permits the specification of an arbitrary number of training epochs and the termination of training once the model's performance on a hold-out validation dataset ceases to improve.

3.3.2 Random Forest Regressor (RFR)

A Random Forest Regressor [4] is a supervised machine learning algorithm so it requires labeled data to train and make predictions which is used for regression tasks. It is an ensemble

method, which means that it combines the predictions of multiple "base" models (such as decision trees) to create a more accurate and robust final prediction.

In Random Forest Regressor, the base models are decision trees. The algorithm works by training multiple decision trees on random subsets of the data. Each decision tree makes a prediction, and the final prediction is the average (or majority vote) of the predictions made by all of the decision trees. This helps to reduce the overfitting problem that can occur when using a single decision tree.

The following steps explain the working of Random Forest algorithm:

Step 1: Select random samples from a given data or training set.

Step 2: This algorithm will construct a decision tree for every training data.

Step 3: Voting will take place by averaging the decision tree.

Step 4: Finally, select the most voted prediction result as the final prediction result.

The steps of the working of Random Forest algorithm is depicted in the below figure 3.14:

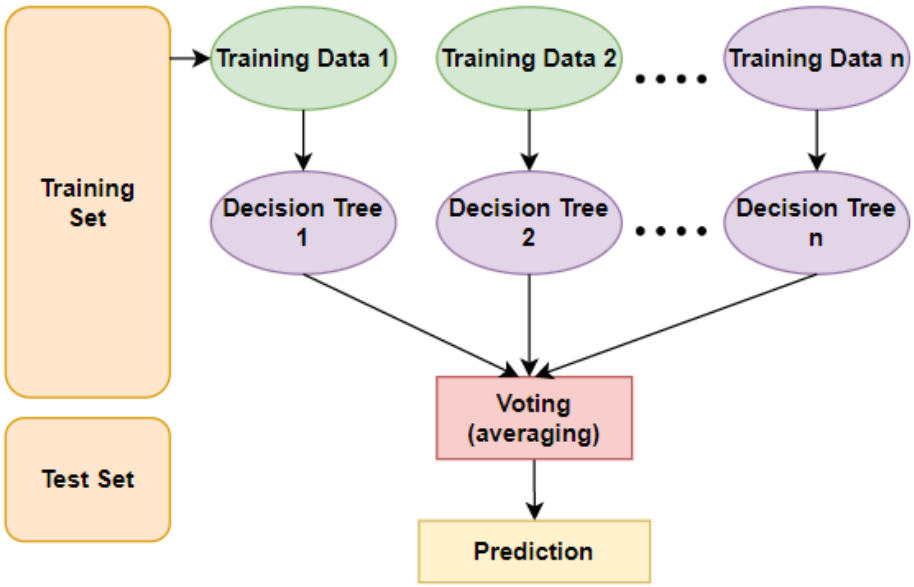


Figure 3.14: Steps of Random Forest.

The random forest regressor has a number of parameters that can be tuned to improve its performance, such as:

- The number of decision trees in the forest

- The maximum depth of each decision tree
- The minimum number of samples required to split an internal node
- The minimum number of samples required to be at a leaf node

Random Forest Regressor is considered to be a powerful and accurate algorithm, and it is widely used in a variety of applications such as stock market prediction, sales forecasting and climate modeling.

3.3.3 K-Nearest Neighbour (KNN)

K-NN is a basic supervised learning method [25] that classifies a sample based on the majority vote of its neighbors, — in other words, the sample is assigned to the class with the highest frequency among its k nearest neighbors. In k-NN, distance metrics are utilized to compute the distance between new and existing samples in a dataset.

In KNN, the value of K is a user-defined parameter and can be chosen based on the specific requirements of the problem. A smaller value of K implies a more complex decision boundary and a higher value of K implies a smoother decision boundary.

KNN is a simple algorithm that is easy to implement and can be useful in many real-world applications such as image classification, speech recognition, and anomaly detection. However, it can be computationally expensive and may not work well when the number of features is large or when the data is highly dimensional.

3.3.4 Long Short Term Memory (LSTM)

Long Short Term Memory Networks is an advanced RNN, a sequential network, that allows for the persistence of information. It is capable of resolving the vanishing gradient problem encountered by RNN. For permanent memory, a recurrent neural network, also known as RNN, is utilized. As a result of the diminishing gradient, RNNs are incapable of remembering long-term dependencies. LSTMs are explicitly intended to prevent difficulties with long-term dependencies [26].

LSTM Architecture

At a high-level LSTM works very much like an RNN cell. Here is the internal functioning of the LSTM network. The LSTM consists of three parts, as shown in the figure 3.15 below and each part performs an individual function.

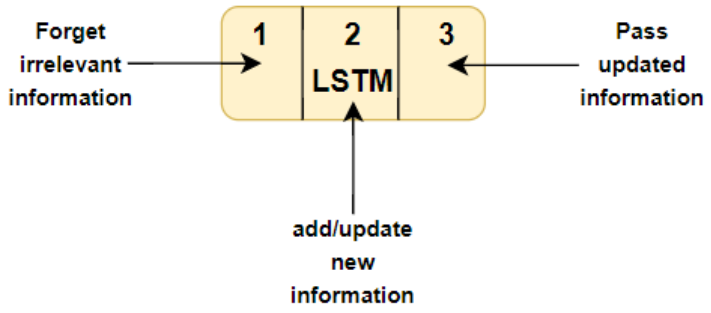


Figure 3.15: Simple LSTM Architecture.

The first part chooses whether the information coming from the previous timestamp is to be remembered or is irrelevant and can be forgotten. In the second part, the cell tries to learn new information from the input to this cell. At last, in the third part, the cell passes the updated information from the current timestamp to the next timestamp. These three parts of an LSTM cell are known as gates. The first part is called Forget gate, the second part is known as the Input gate and the last one is the Output gate which depicted in figure 3.16.

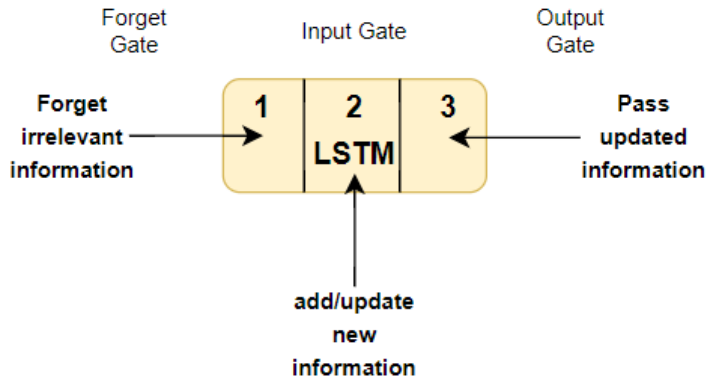


Figure 3.16: Simple LSTM Architecture with Three Gates.

Just like a simple RNN, an LSTM also has a hidden state where H_{t-1} represents the hidden state of the previous timestamp and Ht is the hidden state of the current timestamp. In addition to that LSTM also have a cell state represented by C_{t-1} and C_t for previous and

current timestamp respectively. Here the hidden state is known as Short term memory and the cell state is known as Long term memory. Refer to the following image. The following figure 3.17 refers:

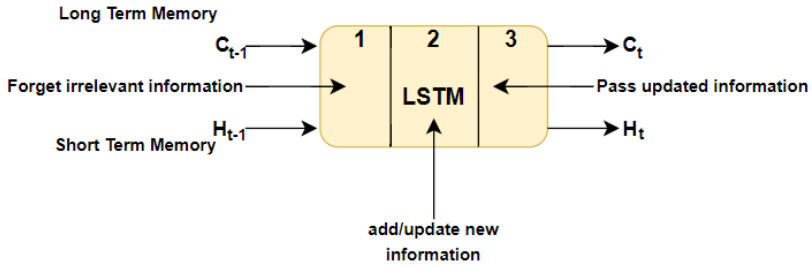


Figure 3.17: Simple LSTM Architecture with Hidden State and Short State.

It is interesting to note that the cell state carries the information along with all the timestamps.

Forget Gate

In a cell of the LSTM network, the first step is to decide whether we should keep the information from the previous timestamp or forget it. Here is the equation 3.15 for forget gate.

$$f_t = \sigma(x_t * U_t + H_{t-1} * W_f) \tag{3.15}$$

In the equation here,

x_t**:** input to the current timestamp

U_f**:** weight associated with the input

H_{t-1}**:** The hidden state of the previous timestamp

W_f**:** It is the weight matrix associated with hidden state

Later, a sigmoid function is applied over it. That will make f_t a number between 0 and 1. This f_t is later multiplied with the cell state of the previous timestamp as shown in below 3.16 and 3.17.

$$C_{t-1} * f_t = 0 \quad .....if\ f_t = 0\ \ (forget\ everything) \tag{3.16}$$

$$C_{t-1} * f_t = C_{t-1} \quad .....f_t = 1\ \ (forget\ nothing) \tag{3.17}$$

If f_t is 0 then the network will forget everything and if the value of f_t is 1 it will forget nothing.

Input Gate

Input gate is used to quantify the importance of the new information carried by the input. Here is the equation 3.18 of the input gate:

$$i_t = \sigma(x_t * U_i + H_{t-1} * W_i) \tag{3.18}$$

In the equation here,

x_t**:** Input at the current timestamp t

U_i**:** weight matrix of input

H_{t-1}**:** A hidden state at the previous timestamp

W_i**:** Weight matrix of input associated with hidden state

Again we have applied sigmoid function over it. As a result, the value of I at timestamp t will be between 0 and 1.

$$N_t = tanh(x_t * U_t + H_{t-1} * W_c) \ \ (new\ information) \tag{3.19}$$

Now the new information that needed to be passed to the cell state is a function of a hidden state at the previous timestamp t-1 and input x at timestamp t. The activation function here is tanh. Due to the tanh function, the value of new information will between -1 and 1. If the value is of N_t is negative the information is subtracted from the cell state and if the value is positive the information is added to the cell state at the current timestamp. However, the N_t won't be added directly to the cell state. Here is the updated equation 3.20.

$$C_t = (f_t * C_{t-1} + i_t * N_t) \ \ (updating\ cell\ state) \tag{3.20}$$

Here, C_{t-1} is the cell state at the current timestamp and others are the values we have calculated previously.

Output Gate

Here is the equation 3.21 of the Output gate, which is pretty similar to the two previous gates.

$$o_t = \sigma(x_t * U_o + H_{t-1} * W_o) \ \ (updatingcellstate) \tag{3.21}$$

It's value will also lie between 0 and 1 because of this sigmoid function. Now to calculate the current hidden state we will use O_t and tanh of the updated cell state. As shown below in equation 3.22:

$$H_t = o_t * tanh(C_t) \tag{3.22}$$

It turns out that the hidden state is a function of Long term memory (C_t) and the current output. If it is needed to take the output of the current timestamp the SoftMax activation on hidden state H_t is applied.

$$Output = Softmax(H_t) \tag{3.23}$$

Here the token with the maximum score in the output is the prediction. The below figure 3.18 is the more intuitive diagram of the LSTM network.

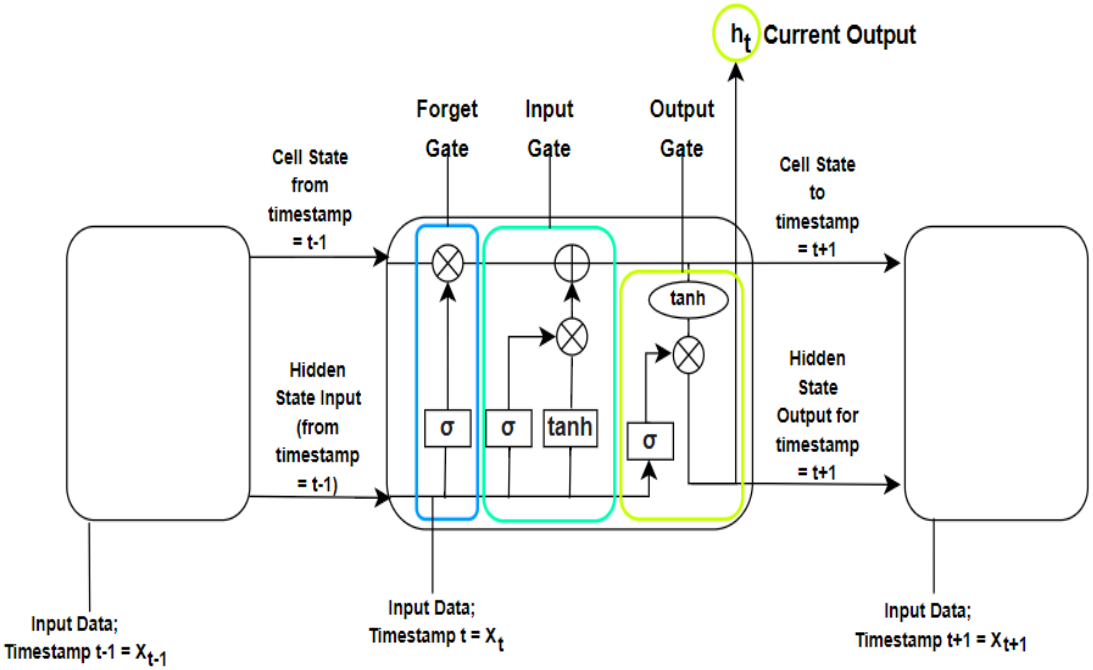


Figure 3.18: Intuitive LSTM Architecture.

3.3.5 Bi-directional long short term memory (Bi-LSTM)

Bidirectional long-short term memory is the process of making any neural network have the sequence information in both directions backwards (future to past) or forward (past to future).

In bidirectional, our input flows in two directions, making a bi-lstm different from the regular LSTM. With the regular LSTM, we can make input flow in one direction, either backwards or forward. However, in bi-directional, we can make the input flow in both directions to preserve the future and the past information.

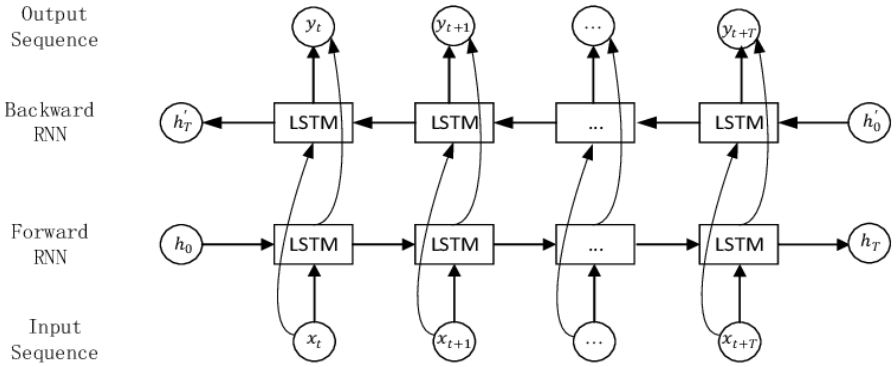


Figure 3.19: BiLSTM Architecture.

In figure 3.19, it can be seen that the flow of information from backward and forward layers. BI-LSTM is usually employed where the sequence to sequence tasks are needed. In a bi-LSTM, there are two separate LSTM layers, one processing the sequence in the forward direction and another processing it in the reverse direction. The hidden states of these two LSTMs are then concatenated and used as the final representation of the sequence. This allows the network to consider both past and future context when making predictions.

Additionally, LSTMs have a memory cell that can preserve information over a long period of time and gates that control the flow of information into and out of the cell, making them well-suited to handle sequences where dependencies between elements span many time steps.

Bi-LSTMs have proven to be effective in many NLP tasks and are widely used in state-of-the-art models. However, they can be computationally expensive and challenging to train, so selecting the appropriate architecture and fine-tuning hyperparameters is important for good performance.

Chapter IV

Methodology

In this chapter, We address a classification method to classify left or right hand imagery movement. After obtaining data, we pre-processed it to remove unnecessary noise. Then we extracted time domain and frequency domain features by using the proper feature extraction method. After extracting the features we classified the data using deep learning models.

The workflow for the task can be separated into 4 major parts:

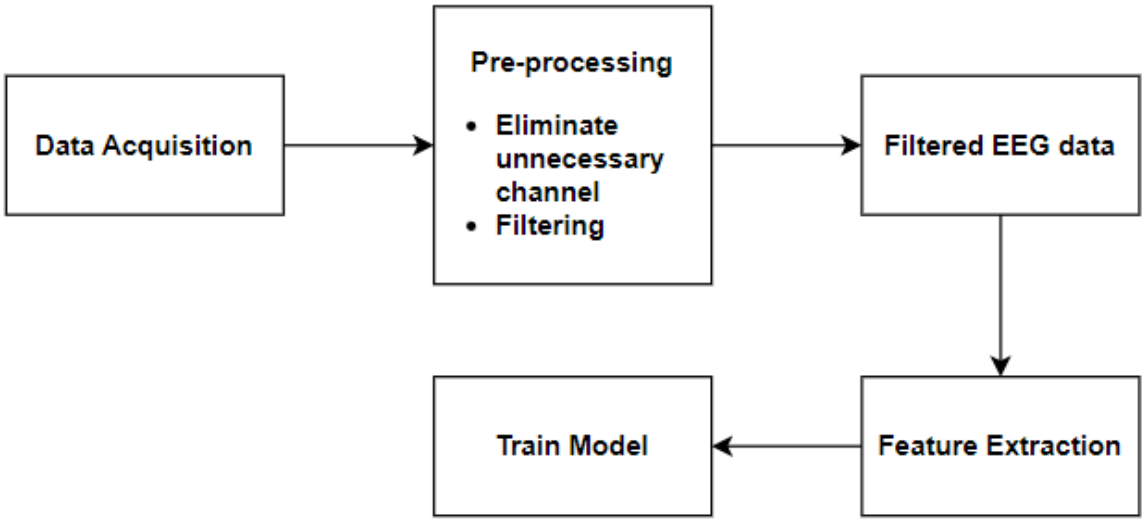
(a) Data Acquisition

(b) Data Pre-processing

(c) Feature Extraction and

(d) Modeling and classification

The overall system for the study is depicted in Figure 4.1:



(a)

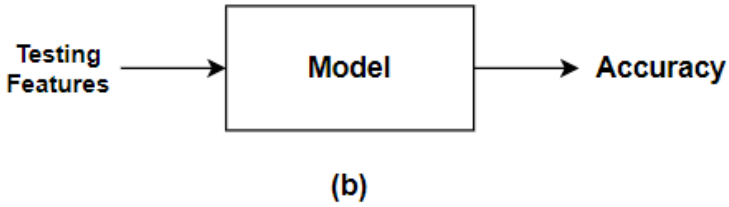


Figure 4.1: Workflow of the thesis (a)The steps of the training the model from the raw EEG signal and (b) the testing phase with the ANN-based predictive model.

4.1 Data Acquisition

According to the guidelines of the Edinburg Handedness Inventory, data was collected from 2 physically fit male volunteers (age = 21$\pm$1.5) who were right-handed. In addition, each subject's physical history is devoid of muscle discomfort, mental disorders, and neurological diseases. Before any data was collected, each participant gave verbal permission, and the whole experiment was explained to them orally. The experiment followed the Helsinki Declaration. The data acquisition techniques were done in the ABSP lab of the Department of Biomedical Engineering at KUET. For EEG data gathering from the scalp, the 9-channel B-AlertX10 system was utilized, which enabled wireless data acquisition without hair removal. Figure Fig. 4.2 shows the procedure for data collection utilized in this study. The EEG data

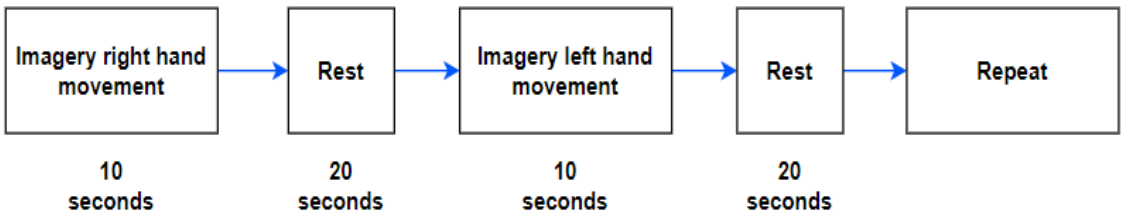


Figure 4.2: Schedule of the data acquisition protocol.

was obtained from all individuals over the course of four sessions, with each subject undergoing five trials per session. A subject participated in 20 trials of a certain class in total. The data set contains two types of motor-imaginary data: right-hand imagery movements and left-hand imagery movements. The data collection sampling rate was 256 Hz [27].

4.2 Data Pre-processing

As our raw acquired signals were the fusion of both signal and noise, it is necessary to pre-process the dataset before feeding the models to get a better outcome. Pre-processing of the dataset for EEG signals includes many steps. The following steps are used to pre-processed the EEG dataset:

Step 1: The data is sampled at the rate of 256 Hz.

Step 2: As left and right hand movement typically include the central electrodes C3 and C4 as well as electrodes over the motor cortex F3 and F4, we remove the data of unnecessary channels from the raw dataset. The detailed positions of the channels are given in figure 4.3.

Step 3: A suitable Butterworth band-pass filter (0.5–60 Hz) is employed to eliminate out-of-band noise from the dataset.

Step 4: The remaining line noise around 50 Hz is eliminated using a notch filter.

Step 5: The high-frequency noise is reduced using a median filter with a window size of 3 samples

Step 6: The EEG signal is filtered using a moving average filter with a window size of 3 samples to further smooth the signal.

Step 7: The entire database has been normalized.

Figure 4.3 depicts the positions of the channels for left hand and right hand movement:

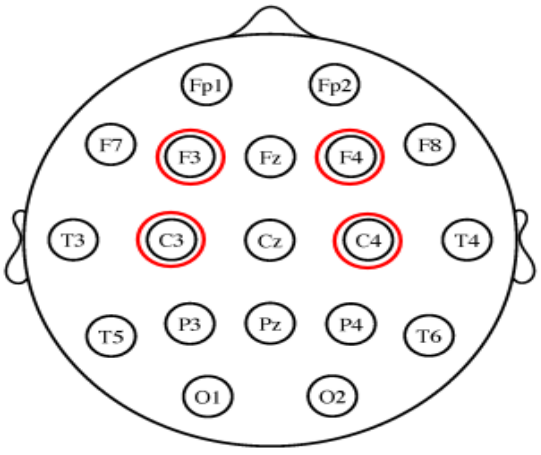


Figure 4.3: Filtered EEG data after step-6.

4.3 Feature Extraction

In the step of feature extraction, linear features (Delta Average Band Power, Theta Average Band Power, Alpha Average Band Power, Beta Average Band Power, Gamma Average Band Power, Theta To Beta Ratio(TBR)) and nonlinear features (Sample Entropy, Dispersion Entropy, MultiScale Sample Entropy) are extracted from EEG signals.

4.3.1 Linear Feature

EEG signals are typically divided into five frequency bands: Delta (0.5-4 Hz), Theta (4-8 Hz), Alpha (8-13 Hz), Beta (13-30 Hz), and Gamma (30-100 Hz). These frequency bands have been associated with specific brain states and functions.

On the basis of the Kaiser window, five Finite Impulse Response (FIR) filters are created to split the preprocessed signals into five frequency subbands. Then we compute the average band power of the mentioned bands using bandpower(). The bandpower function computes the average power of a signal within a specific frequency band. The formula for the bandpower is defined as:

$$\text{bandpower}(x) = \frac{1}{\text{length}(x)} \cdot \sum_{i=1}^{\text{length}(x)} |x_i|^2 \tag{4.1}$$

Then we compute theta to beta ratio(TBR). TBR (Theta to Beta Ratio) is calculated as the ratio of average power in the Theta band to the average power in the Beta band. The calculation is as follows:

$$\text{TBR} = \text{abpTheta.}/\text{abpBeta} \tag{4.2}$$

where abpTheta is the average power in the Theta band and abpBeta is the average power in the Beta band.

All these features are frequency domain features

4.3.2 Non-linear Feature

Here we extract three nonlinear features (Sample Entropy, Dispersion Entropy, and MultiScale Sample Entropy). Sample entropy of the preprocessed signal is calculated using SampEn function which takes three inputs:

- **data :** the input time series data
- **m :** the embedding dimension
- **r :** the tolerance value

Dispersion Entropy is calculated using DispEn function. The DispEn function calculates the Dispersion entropy (RDE) of a time series signal (Sig) using one of several different discretization methods ('linear', 'kmeans', 'ncdf', 'finesort', 'equal'). Additional optional parameters include the embedding dimension (m), delay time (tau), number of discretization bins (c), logarithmic base (Logx), normalization flag (Norm), and discretization type (Typex). The function first discretizes the input signal based on the specified method. Then it forms an embedding matrix with m-dimensional lagged vectors and calculates the RDE. The RDE is a measure of how well the embedded vectors can be distinguished from each other and is given by the sum of the squared probabilities of the unique embedded vectors minus 1 divided by the number of unique embedded vectors.

MultiScale sample entropy(MSE) is calculated using the multiscaleSampleEntropy function. MSE is a measure of the complexity of a time series signal and is an extension of sample entropy (SE).MSE has four parameters - x (the input time series signal), m (the length of the comparison vector), r (the tolerance or neighborhood radius), and tau (the scale factor). The input signal is downsampled by taking the mean of non-overlapping segments of length tau. The resulting signal is referred to as the coarse signal. The coarse signal is divided into (m + 1)-element sequences and stored in a matrix X. The number of sequences that are "matching" (i.e., similar to each other within a certain tolerance) is then calculated using the Chebyshev distance metric and the neighborhood radius (r). The number of matching sequences is stored in variable A. The same process is repeated for m-element sequences and the number of matching sequences is stored in variable B. The final MSE value is calculated as the logarithmic scaling of B by A. If either A or B is zero, NaN is returned as the MSE value.

$$e = \ln\left(\frac{B}{A}\right) \tag{4.3}$$

All these features are time domain features.

4.4 Modeling and classification

Classification is a predictive modeling task in machine learning where a class label is predicted for a given sample of input data. In supervised machine learning, a class label is assigned in a test image based on previous experience from the training set. For classification the following models are adopted:

- Deep Neural Network(DNN)
- Artificial Neural Network(ANN)
- Bidirectional-Long short-term memory (Bi-LSTM)
- Random Forest Regressor(RFR)
- K-nearest neighbors

Stratified K-Fold cross-validation is a technique for evaluating machine learning models by dividing a dataset into K folds (where K is a user-specified number), and then using each fold as a validation set while the remaining K-1 folds are used as the training set. This process is repeated K times, with each fold being used once as the validation set. K-folds cross validation method according to Figure 4.4:

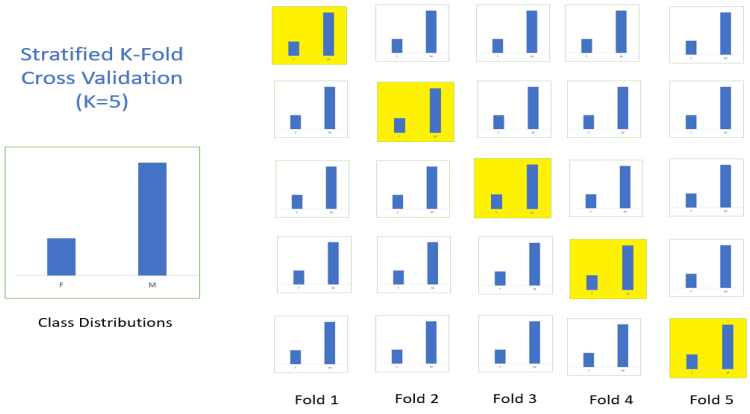


Figure 4.4: Procedure of Stratified K-Fold (K=5) cross-validation.

The main idea behind Stratified K-Fold cross-validation is to ensure that each fold contains roughly the same proportions of each target class as the complete dataset. This is particularly useful when the target classes are imbalanced, as it helps prevent over-representation or under-representation of any class in the validation set.

Stratified K-Fold cross-validation is often used in machine learning to estimate the performance of a model on unseen data, and to tune the hyperparameters of the model for improved performance.

4.4.1 Deep Neural Network(DNN)

Our DNN model has 36 input features, which are fed into the first hidden layer with 78 units and the activation function "relu". The input layer has input dimension parameter set to 36.

The following hidden layer has the same structure with 78 units and "relu" activation function.

Batch normalization and dropout regularization with a rate of 0.5 which randomly drops out 50% of the neurons in each forward pass to prevent overfitting are also used to improve the performance of the DNN. Batch normalization is applied after the second dense layer, while dropout regularization is applied after the third dense layer.

The final layer is a single neuron with a sigmoid activation function that outputs a binary classification. The DNN is compiled with the Adam optimizer and a binary cross-entropy loss function. The accuracy metric is also computed during training. The learning rate of the Adam optimizer is set to 0.01..

We have used cross-validation, specifically StratifiedKFold with 10 folds. The EarlyStopping and ModelCheckpoint classes are used to stop the training early if the validation loss does not improve for a certain number of epochs, and to save the best model, respectively. The training and test sets are created for each fold, and the training set is further split into a training set and a validation set. The model is trained for 200 epochs with a batch size of 64 and early stopping with a patience of 30 epochs if the validation loss does not improve. After training, the model's predictions are rounded to 0 or 1.

4.4.2 Artificial Neural Network(ANN)

Our feedforward artificial neural network (ANN) model has a sequential architecture and contains two dense layers and uses the tanh activation function for the first layer and the sigmoid activation function for the final layer. The first layer has 36 units and an input

dimension of 36. Batch normalization is also applied to the output of the first layer to improve the performance of the ANN.

The ANN is compiled with the Adam optimizer and a binary cross-entropy loss function. The accuracy metric is also computed during training. The learning rate of the Adam optimizer is set to 0.01.

We have used cross-validation, specifically StratifiedKFold with 10 folds. The EarlyStopping and ModelCheckpoint classes are used to stop the training early if the validation loss does not improve for a certain number of epochs, and to save the best model, respectively. The training and test sets are created for each fold, and the training set is further split into a training set and a validation set. The model is trained for 200 epochs with a batch size of 32 and early stopping with a patience of 30 epochs if the validation loss does not improve. After training, the model's predictions are rounded to 0 or 1.

4.4.3 Bidirectional-LSTM

Our bidirectional LSTM model contains has 2 bidirectional LSTM layers, each with 72 and 36 units, respectively.The input shape is 36

After the Bi-LSTM layers, a dense layer with 36 units and the ReLU activation function is added, followed by a dropout layer with a dropout rate of 0.5 to reduce overfitting. Finally, a dense layer with 1 unit and the sigmoid activation function is added to produce the binary classification output.

The Bi-LSTM network is compiled with the Adam optimizer and a binary cross-entropy loss function. The accuracy metric is also computed during training.

We have used cross-validation, specifically StratifiedKFold with 10 folds. The EarlyStopping and ModelCheckpoint classes are used to stop the training early if the validation loss does not improve for a certain number of epochs, and to save the best model, respectively. The training and test sets are created for each fold , and the training set is further split into a training set and a validation set. The model is trained for 200 epochs with a batch size of 64 and early stopping with patience of 30 epochs if the validation loss does not improve. After training, the model's predictions are rounded to 0 or 1.

4.4.4 Random Forest Regressor(RFR)

Our RFR model has been created using the scikit-learn library, with the following hyperparameters specified:

- *n_estimators* : the number of trees in the forest, set to 150.
- *min_samples_split* : the minimum number of samples required to split an internal node, set to 5.
- *max_depth* : the maximum depth of the tree, set to 10.
- *random_state* : the seed for the random number generator, set to 32.

We have used cross-validation, specifically StratifiedKFold with 10 folds. The training data is split into train and test sets for each fold. After training, the model's predictions are rounded to 0 or 1.

4.4.5 K-Nearest Neighbors (KNN)

Our KNN classifier is implemented using the KNeighborsClassifier class from the scikit-learn library. The classifier is created with the following parameters:

- *n_neighbors* : This parameter specifies the number of nearest neighbors to consider for classification. In this case, the value is set to 3.
- *weights* : This parameter determines the weight function used in prediction. It can be set to 'uniform' for uniform weighting, or 'distance' for weighting by the inverse of the distance. In this case, the value is set to 'distance', meaning that closer neighbors will have a higher weight in the prediction.

We have used cross-validation, specifically StratifiedKFold with 10 folds. The training data is split into train and test sets for each fold. After training, the model's predictions are rounded to 0 or 1.

Chapter V

Result Analysis and Discussion

The mentioned methods and terminologies in the previous chapter are applied to the acquired dataset to classify the left-hand movement and right-hand movement.Evaluation metrics for classification are described in section 5.1. Classification results are analyzed in section 5.2. Finally, the chapter is summarized in section 5.3.

5.1 Evaluation Metrics

Evaluation metrics are used to measure the performance of a deep learning model. In the following subsection Evaluation Metrics for classification are described. Using these evaluation metrics, the results of mentioned methods in the previous chapter are analyzed in section 5.2.

5.1.1 Classification Evaluation metric

Classification evaluation metrics are used to evaluate the performance of a binary class classification model. Some common classification evaluation metrics include:

- **Accuracy:** Accuracy is a commonly used evaluation metric for binary classification models. It measures the proportion of correctly predicted labels to the total number of instances. Accuracy is defined as:

  $$Accuracy = \frac{True\ Positive + True\ Negative}{Instances}$$

  Where True Positives (TP) are instances that are correctly predicted as positive and True Negatives (TN) are instances that are correctly predicted as negative. Accuracy is a simple and straightforward metric, but it may not always provide a complete picture of the model's performance. For example, in a situation where the negative class is much more prevalent than the positive class, a model that always predicts the negative class will still have a high accuracy even though it is not detecting the positive class well. In such cases, precision, recall, and F1 score are more appropriate evaluation metrics.

- **Precision:** Precision is a commonly used evaluation metric for binary models. It measures the proportion of true positive predictions to the total number of positive predictions made by the model. It is defined as:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Where True Positives (TP) are instances that are correctly predicted as positive and False Positives (FP) are instances that are incorrectly predicted as positive. Precision is a measure of the model's ability to correctly identify positive instances and is particularly important when the cost of a false positive is high, such as in medical diagnosis or fraud detection.

It's worth noting that precision should be combined with recall to get a complete picture of the model's performance. A high precision model is only desirable if it can detect the majority of positive instances (high recall).

- **Sensitivity:** Recall, also known as sensitivity or true positive rate, is an evaluation metric used in binary and multi-class classification models. It measures the proportion of positive instances that are correctly predicted as positive by the model. Recall is defined as:

$$Sensitivity\ or\ Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Where True Positives (TP) are instances that are correctly predicted as positive and False Negatives (FN) are instances that are incorrectly predicted as negative. Recall is a measure of the model's ability to correctly identify positive instances, especially in cases where the positive class is rare. For example, in a fraud detection setting, recall can be used to measure the model's ability to identify all fraudulent transactions.

It's worth noting that recall should be combined with precision to get a complete picture of the model's performance. A high recall model is only desirable if it can correctly identify the majority of positive instances (high precision).

- **Specificity:** Specificity is an evaluation metric used in binary classification models that measures the proportion of negative instances that are correctly predicted as negative by the model. Specificity is defined as:

$$Specificity = \frac{True\ Negative}{True\ Negative + False\ Positive}$$

Where True Negatives (TN) are instances that are correctly predicted as negative and False Positives (FP) are instances that are incorrectly predicted as positive. Specificity is a measure of the model's ability to correctly identify negative instances, especially in cases where the negative class is prevalent. For example, in a medical diagnosis setting, specificity can be used to measure the model's ability to correctly identify patients who do not have a particular disease.

It's worth noting that specificity should be combined with sensitivity (also known as recall or true positive rate) to get a complete picture of the model's performance. A high specificity model is only desirable if it can detect the majority of positive instances (high sensitivity).

- **F1 Score:** The F1 score is a commonly used evaluation metric for binary and multi-class classification models. It is the harmonic mean of precision and recall and provides a single score that balances both measures. The F1 score is defined as:

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Where *Precision = True Positives / (True Positives + False Positives)* and *Recall (Sensitivity) = True Positives / (True Positives + False Negatives)*. A high F1 score indicates that the model has a good balance between precision and recall. A score of 1.0 indicates perfect precision and recall, while a score of 0.0 indicates that the model is not making any positive predictions.

It's worth noting that the F1 score is only a single number and doesn't provide a comprehensive view of the model's performance. Other evaluation metrics, such as precision, recall, accuracy, and confusion matrix, should also be considered.

The mentioned statistical measures are calculated using the four terms from the confusion matrix: TP, FP, TN, and FN. A confusion matrix is a table used to evaluate the performance of a classification algorithm. It compares the actual class labels of instances to the class labels predicted by the model. The confusion matrix displays the number of correct and incorrect predictions made by the model, allowing us to calculate several important evaluation metrics. The structure of confusion matrix is shown in Table 5.1

Table 5.1: Confusion Matrix.

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive	False Negative
Actual Negative	False Positive	True Negative

A true positive is a correct positive prediction made by the model and a true negative is a correct negative prediction made by the model. False negative is a mistake made by the model where it fails to detect a positive case and false positive is a mistake made by the model where it falsely identifies a negative case as positive.

5.2 Classification Results

For subject 1 the bidirectional LSTM model has achieved 85.31% accuracy which is the highest among all the 5 models that are trained on the acquired dataset. For subject 2 the bidirectional LSTM model has achieved 87.58% accuracy which is the highest among all the 5 models that are trained on the acquired dataset. A comparison chart of accuracy among the models are shown in table 5.2.

Table 5.2: Average Classification Performances For Model Bi-LSTM, DNN, ANN, Random Forest, and KNN.

SL No	**Model**	**Accuracy of subject 1**	**Accuracy of subject 2**
1	Bidirectional LSTM	85.31%	87.58%
2	DNN	85.06%	83.92%
3	ANN	84.03%	81.89%
4	Random Forest	68.76%	67.13%
5	KNN	67.13%	66.01%

5.2.1 Window Size 3 for Median Filter and Moving Average Filter

During signal processing operations on the raw signal, we have used 4 filters (Butterworth band-stop filter, Notch filter, Median filter, and Moving average filter). In this case window size 3 is used for Median filter and Moving average filter. Evaluation metrics of 5 models based on window size are documented in table 5.3 and 5.4 for subjects 1 and 2 respectively.

Table 5.3: Accuracy, Sensitivity, Specificity, Precision and f1 score of 5 models of subject 1 for window size 3.

SL No	Model	Accuracy	Sensitivity	Specificity	Precision	F1 Score
1	Bidirectional LSTM	94.94%	92.50%	97.44%	97%	94.87%
2	DNN	94.94%	94.87%	90%	90%	92.50%
3	ANN	89.87%	87.50%	92.30%	92%	89.74%
4	Random Forest	67.38%	72.15%	74.36%	74%	71.79%
5	KNN	71.25%	65%	77.50%	74%	69.33%

Table 5.4: Accuracy, Sensitivity, Specificity, Precision and f1 score of 5 models of subject 2 for window size 3.

SL No	Model	Accuracy	Sensitivity	Specificity	Precision	F1 Score
1	Bidirectional LSTM	98.73%	97.50%	100%	100%	98.73%
2	DNN	94.93%	95%	94.87%	95%	95%
3	ANN	91.14%	92.30%	90%	90%	91.14%
4	Random Forest	75.95%	71.75%	80%	78%	74.67%
5	KNN	75.95%	82.05%	70%	73%	77.11%

5.3 Discussion

In this chapter, the results of left-hand movement vs right-hand movement using different models are described briefly. Maximum accuracy for motor imagery classification was found by using the Bidirectional LSTM and DNN model with a 94.94% accuracy score for subject 1 and for subject 2, Bidirectional LSTM model has achieved the highest accuracy with 98.73%.

Chapter VI

Conclusions and Future Works

6.1 Conclusion

Left-handed vs right-handed movement classification of motor imagery is required for several applications, including brain-computer interfaces, rehabilitation, and motor skill training. By classifying the kind of motor imagery, researchers can acquire insights into how the brain regulates and coordinates movement, so enhancing our understanding of motor control and movement disorders including Parkinson's disease and stroke. Additionally, it can also be used for brain-computer interface applications, where the intention of the user to move a specific limb is decoded from their brain activity. In our work, Five models have been trained on two different versions of the preprocessed dataset. The Bidirectional LSTM model has achieved the highest accuracy among the five models for both subject 1 and subject 2. The DNN model for subject 1 has the same highest accuracy as Bidirectional LSTM. For subject 1 the highest accuracy is 94.94% and for subject 2 it is 98.73%.

6.2 Limitations

Some aspects of this thesis work are yet to be considered. There are some limitations of the thesis work which are listed below:

- More participants should be included.
- Random forest regressor and KNN did not work well for our dataset.
- Though BiLSTM achieved the second highest average accuracy, it is a time consuming process.

6.3 Recommendations For Future Research

This thesis work can be extended in several ways. The following includes some possible areas that are recommended to extend the present work.

1. The dataset can be increased and diversified so that the classifier becomes more robust and exceeds the highest accuracy.
2. A more robust model for classification of motor imagery data can be built and trained with more data.

References

[1] Huang, Liu, J. S., Yao, W. S., Wang, B., Chen, Z. X., Sun, S. F., and Fang, W., "Electroencephalogram-based motor imagery classification using deep residual convolutional networks," *Frontiers in Neuroscience*, vol. 15, 2021.

[2] Zimmermann-Schlatter, Schuster, A., Puhan, C., Siekierka, M. A., Steurer, E., and Johann, "Efficacy of motor imagery in post-stroke rehabilitation: a systematic review," *Journal of NeuroEngineering and Rehabilitation*, vol. 15, 2008.

[3] Ogourtsova, T., McDermott, A., Kim, A., Kagan, A., Belley, E., PT, M. P.-V. B., Filion, J.-A., Nutter, A., Saulnier, M., Shedleur, S., Wan, T. T., Sitcoff, E., and Korner-Bitensky, N., "Motor imagery / mental practice," Ph.D. dissertation, 2017.

[4] Rahman, M. A., Khanam, F., Ahmad, M., and Uddin, M. S., "Multiclass eeg signal classification utilizing rényi min-entropy-based feature selection from wavelet packet transformation," *Brain Informatics*, vol. 7, 2020.

[5] Aldea, R., Fira, M., and Lazăr, A., "Classifications of motor imagery tasks using k-nearest neighbors," pp. 0–4, 2014.

[6] XU, G., SHEN, X., CHEN, S., ZONG, Y., ZHANG, C., YUE, H., MIN LIU, F. C., and CHE, W., "A deep transfer convolutional neural network framework for eeg signal classification," *IEEE Access*, vol. 7, pp. 112 767–112 776, 2019.

[7] Tabar, Y. R. and Halici, U., "A novel deep learning approach for classification of eeg motor imagery signals," *Journal of Neural Engineering*, vol. 14, no. 1, 2017.

[8] Poortinga and Eke, "Event-related eeg/meg synchronization and desynchronization: basic principles," *Clinical Neurophysiology*, vol. 110, pp. 1842–1857, 1999.

[9] Rahman, M. A., Khanam, F., Hossain, M. K., Alam, M. K., and Ahmad, M., "Four-class motor imagery eeg signal classification using pca, wavelet and two-stage neural network," *International Journal of Advanced Computer Science and Applications*, vol. 10, pp. 481–490, 2019.

[10] LIU, L., "Recognition and analysis of motor imagery eeg signal based on improved bp neural network," *IEEE Access*, vol. 7, pp. 47 794–47 803, 2019.

[11] Amin, Alsulaiman, S. U., Muhammad, M., Bencherif, G., Hossain, M. A., and Shamim, M., "Multilevel weighted feature fusion using convolutional neural networks for eeg motor imagery classification," *IEEE Access*, vol. 7, pp. 18 940–18 950, 2019.

[12] Zhang, Duan, Z., Sole-Casals, F., Dinares-Ferran, J., Cichocki, J., Yang, A., Sun, Z., and Zhe, "A novel deep learning approach with data augmentation to classify motor imagery signals," *IEEE Access*, vol. 7, pp. 15 945–15 954, 2019.

[13] Xu, Shen, G., Chen, X., Zong, S., Zhang, Y., Yue, C., Liu, H., Chen, M. I. N., Che, F. E. I., and Wenliang, "A deep transfer convolutional neural network framework for eeg signal classification," *IEEE Access*, vol. 7, pp. 112 767–112 776, 2019.

[14] Chen and M, T., "Classification of motor imagery eeg signals based on deep autoencoder and convolutional neural network approach," *IEEE Access*, vol. 10, pp. 48 071–48 081, 2022.

[15] Korhan and Nuri, "Motor imagery based eeg classification by using common spatial patterns and convolutional neural networks," *2019 Scientific Meeting on Electrical-Electronics Biomedical Engineering and Computer Science (EBBT)*, pp. 1–4, 2019.

[16] Herman, Prasad, P., Member, G., Mcginnity, S., Coyle, T. M., and Damien, "Comparative analysis of spectral approaches to feature extraction for eeg-based motor imagery classification," vol. 16, pp. 317–326, 2008.

[17] Sohaib, T., Qureshi, A., and Shahnawaz, "An empirical study of machine learning techniques for classifying emotional states from eeg data," 2012.

[18] "Motor imagery," https://en.wikipedia.org/wiki/Motor_imagery, wikipedia.

[19] Akhand, D. M. A. H., *Deep Learning Fundamentals- A Practical Approach to Understanding Deep Learning Methods,.* ISBN:978-984-35-0812-6, UGC, Bangladesh, 2021.

[20] Wang, Jiang, P., Liu, A., Shang, X., Zhang, J., and Li, "Lstm-based eeg classification

in motor imagery tasks," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 26, pp. 2086–2095, 2018.

[21] Haykin, S., *Neural networks and learning machines, 3/E.* Pearson Education India, 2010.

[22] Siddique, N. and Adeli, H., *Computational intelligence: synergies of fuzzy logic, neural networks and evolutionary computing*. John Wiley & Sons, 2013.

[23] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[24] Bock, Goppold, S., Weiß, J., and Martin, "An improvement of the convergence proof of the adam-optimizer," pp. 1–5, 2018.

[25] Isa, M., Amir, N. E., Ilyas, A., Razalli, M. Z., and Shahrazel, M., "The performance analysis of k-nearest neighbors (k-nn) algorithm for motor imagery classification based on eeg signal," *MATEC Web of Conferences*, vol. 140, pp. 0–5, 2017.

[26] Saxena, S., "Introduction to long short term memory (lstm)," https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/, accessed: January 31, 2023.

[27] Saha, P. K., Rahman, M. A., and Mollah, M. N., "Frequency domain approach in csp based feature extraction for eeg signal classification," *2nd International Conference on Electrical, Computer and Communication Engineering, ECCE 2019*, 2019.