**Republic of Iraq**
**Ministry of Higher Education**
**and Scientific Research**
**University of Babylon**
**College of Engineering**
**Electrical Engineering Department**

# Prediction of Epileptic Seizures based on Electroencephalography Signals

*A Dissertation*

**Submitted to the Electrical Engineering Department/ College of Engineering / University of Babylon in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Electrical Engineering/ Electronics & Communications**

*by*

***Hussein Mohammed Hussein Aideen***

*Supervised by*

***Prof. Dr. Kasim Karam Abdalla***

***2023 A.D*** ***1444 A.H***

## Supervisor Certification

I certify that this dissertation entitled (**Prediction of Epileptic Seizures based on Electroencephalography Signals**) and submitted by student **(Hussein Mohammed Hussein Aideen**) was prepared under my supervision at the Department of Electrical Engineering, College of Engineering, University of Babylon, as a part of the requirements for the degree of Doctorate of Philosophy in Electronics and Communications Engineering.

Signature:
*Supervisor*: ***Prof. Dr. Kasim Karam Abdalla***
Date: / / 2023

I certify that this dissertation mentioned above has been completed in Electronics and Communications Engineering in the College of Engineering/University of Babylon.

Signature:
*Head of Department:* ***Prof. Dr. Qais Kareem Omran***

Date: / / 2023

## Examining Committee Certificate

We certify that we have read this dissertation entitled "**Prediction of Epileptic Seizures based on Electroencephalography Signals**" and as an examining committee, we examined the student "**Hussein Mohammed Hussein Aideen**", in its contents and that in our opinion it meets the standard of a dissertation for the degree of **Doctorate of Philosophy** in Electrical Engineering/ Electronics and Communications Engineering.

Signature:
Name: ***Prof. Dr. Ehab A. Hussein***
(Chairman)
Date: / / 2023

Signature:
Name: ***Prof. Dr. Hasan Fahad Khazaal***
(Member)
Date: / / 2023

Signature:
Name: ***Prof. Dr. Saad Saffah Hassoon***
(Member)
Date: / / 2023

Signature:
Name: ***Prof. Dr. Qais Kareem Omran***
(Member)
Date: / / 2023

Signature:
Name: ***Prof. Dr. Haider Ismael Shahadi***
(Member)
Date: / / 2023

Approval of Head of Department

Signature:
Name: ***Prof. Dr. Qais Kareem Omran***
Date: / / 2023

Approval of the Dean of Collage

Signature:
Name: ***Prof. Dr. Hatem Hadi Obeid***
Date: / / 2023

# Abstract

Automatic brain disease state recognition and brain state monitoring are becoming important in today's medical equipment and patients' life. These medical tools are based on sensing the brain waves as an electrical signal. Electroencephalography (EEG) represents the electrical activity of brain neurons and it has much information about the brain state conveyed in it, like predicting the incoming seizures in epileptic patients. Due to the nature of the nonlinearity and non-stationarity of EEG signals and their contamination with artefacts, processing and extracting information from these signals are challenging tasks.

In this work, a new method for fine-tuning EEG signals and artefact cancellation based on signal statistics with modification of independent sources extracted by Independent Component Analysis (ICA) of EEG signals is suggested. Visual inspection of the reconstructed signals shows the validity of the proposed method in artefact rejection.

Two deep learning methods are suggested and evaluated; the former model is established on a convolutional neural network (CNN) and the succeeding model is established on Gated Recurrent Unit-Long Short-Term Memory (GRU-LSTM). The main task of the models is distinguishing between the pre-ictal state and inter-ictal state, and then predicting the onset of the seizure. The models were based on patient comfort in using 5 electrodes of Electroencephalography (EEG) signals and reducing the alert period to only 10 minutes before the onset of the seizure. These models are implemented in Python v3 and the results of (Maximum: 90%, Average: 66%) in terms of accuracy and (Maximum: 90%, Average: 62%) in terms of sensitivity were obtained using the CNN model. As well as (Maximum: 71%, Average: 56%) accuracy and (Maximum:73%, Average: 56%) sensitivity were obtained using the GRU-LSTM model.

A seizure prediction algorithm with four rounds based on Machine Learning is proposed and implemented with MATLAB v2021. The algorithm one of the EEG channels as input channel and finds the optimum Pre-ictal period length

(PIL), the optimum length of the sample segment (SEG) and uses simulated annealing method to find a set of extracted features that achieve the best performance in differentiating between the Pre-ictal and Inter-ictal periods. An average sensitivity of 77% and a low false prediction rate of 20% are obtained by testing the algorithm with the Children's Hospital Boston (CHB) and the Massachusetts Institute of Technology (CHB-MIT) scalp dataset.

A real-time seizure prediction algorithm that calculates the Power Spectral Density (PSD) of incoming EEG segments, and uses the rapid frequency changes across bands as signs of incoming seizures is proposed. The results showed the ability of the algorithm to detect 78% seizures in the Siena scalp dataset with an average false positive rate of 0.16.

# Contents

Abstract ..... I
Contents ..... III
List of Figures ..... VII
List of Tables ..... IX
List of Symbols ..... X
List of Abbreviations ..... XI
Introduction ..... 1
1.1 Preface ..... 1
1.2 Literature Review ..... 3
1.2.1 Artefact Removal Research ..... 4
1.2.2 Seizure Prediction with Deep Learning Research ..... 6
1.2.3 Seizure Prediction with Feature Extraction and Machine Learning Research ..... 7
1.3 Problem Statement ..... 10
1.4 Aims of the Work ..... 11
1.5 Contributions ..... 12
1.6 The Proposed System Limitation ..... 12
1.7 Dissertation Organization ..... 12
Medical and Technical Overview ..... 14
2.1 Introduction ..... 14
2.2 Human Brain Anatomy ..... 14
2.3 Human Brains' Neurophysiology ..... 15
2.4 Electroencephalography Signals ..... 16
2.5 Wave Patterns and Brain Rhythms ..... 17
2.6 EEG Recording and Electrodes ..... 18
2.7 EEG Artefacts ..... 20
2.7.1 Ocular Artefact ..... 20
2.7.2 Muscle Artefacts ..... 21

2.7.3 Cardiac Artefact ..... 21
2.7.4 Other Artefacts and Artefact Handling ..... 22
2.7.5 Independent Component Analysis (ICA) ..... 22
2.8 Brain Disease- Epilepsy ..... 24
2.8.1 Types of Epileptic Seizures ..... 25
2.8.2 Predictability of Seizure from the EEG Signals ..... 26
2.9 Feature Extraction Methods ..... 28
٢,٩,١ Empirical Mode Decomposition ..... 28
2.9.2 Short Time Fourier Transform ..... 29
2.9.3 Chaos theory and Dynamical Analysis ..... 29
2.9.4 Simulated Annealing ..... 33
2.10 Machine Learning and Classification Methods ..... 34
2.10.1 Support Vector Machine ..... 34
2.10.2 Two Sample t-Test ..... 39
2.11 Deep Learning Methods ..... 40
2.11.1 Artificial Neural Networks ..... 40
2.11.2 Convolutional Neural Network (CNN) ..... 41
2.11.3 Pooling Layer ..... 42
2.11.4 Error Backpropagation ..... 43
2.11.5 Activation Function ..... 43
2.11.6 Recurrent Neural Networks ..... 43
2.11.7 Long Short-Term Memory ..... 44
2.11.8 Gated Recurrent Unit Layer ..... 44
2.11.9 Fully Connected Network ..... 45
2.11.10 Regularization and Dropout ..... 46
2.11.11 Batch Normalization Layer ..... 46
٢,١٢ Performance Evaluation Metrics for Deep Learning Models ..... 46
The Proposed Models ..... 48
3.1 EEG Signals Artefact Rejection ..... 48
3.1.1 Artefact Rejection Algorithm ..... 48

3.1.2 Using Average Reference Montage ..... 48
3.1.3 EEG Signal Filtering ..... 49
3.1.4 Applying ICA to EEG signals ..... 49
3.1.5 Artefact Identification and Cancelation ..... 50
3.1.6 Restoring EEG Signal from ICA Components ..... 52
3.2 Seizure Prediction Based on Machine Learning ..... 53
3.2.1 Feature Extraction ..... 53
3.2.2 The Proposed Seizure Perdition Algorithm ..... 54
3.3 Seizure Detection and Prediction based on Deep Learning ..... 60
3.3.1 Seizure Detection Model ..... 60
3.3.2 Seizure Prediction Models ..... 62
3.4 The Proposed Real-time Seizure Prediction Algorithm ..... 64
Simulation Results of the Proposed Models ..... 68
4.1 Introduction ..... 68
4.2 EEG Signals Artefact Rejection Results ..... 68
4.2.1 The Utilized Software ..... 70
4.3 Deep Learning Models Results ..... 72
4.3.1 Performance of the Seizure Detection Model ..... 72
4.3.2 Performance of Seizure Prediction Models ..... 74
4.3.3 Discussion And Comparison ..... 79
4.3.4 Utilized Software ..... 80
4.4 Machine Learning Model Results ..... 82
4.4.1 Comparing with other works ..... 86
4.4.2 Results obtained with Siena Scalp Database ..... 86
4.5 Results of Proposed Real-Time Seizure Prediction Algorithm ..... 92
4.6 Chapter Summary ..... 99
Conclusion and Future Works ..... 101
5.1 Conclusions ..... 101
5.2 Future Works ..... 102
References ..... 103

Appendix A ..... A1
Appendix B..... B1
Appendix C..... C1
Seizure Prediction Model 1..... C1
Seizure Prediction Model 2..... C9

# List of Figures

Figure 1.1 (a) ML EEG system, (b) DL EEG system [7]. ..... 3
Figure 2.1 The human brain anatomy [1]. ..... 15
Figure 2.2 Neuron structure [1]. ..... 16
Figure 2.3 Action potential [2]. ..... 16
Figure 2.4 Representation of the normal brain rhythms [1]. ..... 18
Figure 2.5 The scalp electrodes are placed with 10/20 proportion. ..... 19
Figure 2.6 Sample EOG signal [51]. ..... 21
Figure 2.7 Sample EMG signal [51]. ..... 21
Figure 2.8 Sample ECG signal [52]. ..... 22
Figure 2.9 The four states of epileptic brain activity [7]. ..... 25
Figure 2.10 Basic Artificial Neural Network (ANN) node structure [84]. ..... 41
Figure 2.11 Convolutional Neural Network (CNN) architecture [4]. ..... 42
Figure 2.12 Long Short-Term Memory (LSTM) structure [88]. ..... 44
Figure 2.13 A fully connected layer structure [84]. ..... 45
Figure 3.1 The proposed artefact rejection algorithm block diagram ..... 49
Figure 3.2 Examples of noise signals selected by Manual method ..... 50
Figure 3.3 Samples of noise signals detected by Automatic method ..... 51
Figure 3.4 Noise identification and cancelation algorithm block diagram ..... 52
Figure 3.5 The flowchart for the first round ..... 55
Figure 3.6 The flowchart of the second round. ..... 57
Figure 3.7 The flowchart of the third and fourth rounds. ..... 59
Figure 3.8 Time series generation and data preparation for seizure detection model ..... 61
Figure 3.9 The proposed seizure detection model architecture ..... 62
Figure 3.10 Data preparation for seizure perdition models ..... 63
Figure 3.11 The architecture of proposed CNN model ..... 64
Figure 3.12 The architecture of proposed GRU-LSTM model ..... 64
Figure 3.13 Sample EEG signal with its extracted values changes ..... 65
Figure 3.14 The block diagram of the proposed real-time seizure prediction algorithm ..... 67
Figure 4.1 Sample 1 (a) Raw EEG data and (b) Reconstructed EEG data after removing noise sources ..... 68
Figure 4.2 Sample 2 (a) Raw EEG data and (b) Reconstructed EEG data after removing noise sources ..... 69
Figure 4.3 Sample 3 (a) Raw EEG data and (b) Reconstructed EEG data after removing noise sources ..... 69

Figure 4.4 Sample 4 (a) Raw EEG data and (b) Reconstructed EEG data after removing noise sources ..... 70
Figure 4.5 Comparison of variance values between the raw channels (blue line) and the reconstructed channels (red line) ..... 71
Figure 4.6 The implemented graphical user interface for the project ..... 71
Figure 4.7 (a) Accuracy and (b) loss with epochs by feeding all data files ..... 73
Figure 4.8 ROC-AUC curves for Model 1 ..... 75
Figure 4.9 ROC-AUC curves for Model 2 ..... 77
Figure 4.10 ROC-AUC samples for Model 2 with average voting technique ..... 79
Figure 4.11 Samples of confusion matrix of the model. ..... 85
Figure 4.12 Samples of confusion matrix of the model with Siena database. ..... 90
Figure 4.13 Sample 1: The power spectral density of EEG signal bands for patient ID PN00. ..... 93
Figure 4.14 Sample 1: The detected changes in frequency bands and prediction alarm for patient ID PN00. ..... 94
Figure 4.15 Sample 2: The power spectral density of EEG signal bands for patient ID PN06. ..... 95
Figure 4.16 Sample 2: The detected changes in frequency bands and prediction alarm for patient ID PN06. ..... 96
Figure 4.17 Sample 3: The power spectral density of EEG signal bands for patient ID PN11. ..... 97
Figure 4.18 Sample 3: The detected changes in frequency bands and prediction alarm for patient ID PN11. ..... 98

# List of Tables

Table 3.1 The extracted feature set. ..... 56
Table 3.2 Parameters for SA algorithm..... 58
Table 4.1 Results comparison with previous works ..... 73
Table 4.2 Performance metrics for Model 1 ..... 75
Table 4.3 Performance metrics for Model 2 ..... 76
Table 4.4 Comparison of earlier methods with proposed models ..... 81
Table 4.5 The results of channel selection round with seizure type. ..... 83
Table 4.6 PIL, SEG, and optimal features for each patient obtained from the algorithm. ..... 83
Table 4.7 Seizure prediction results achieved with the CHB–MIT scalp EEG database. ..... 84
Table 4.8: Comparing results with recent studies in seizure prediction ..... 87
Table 4.9 The results of channel selection round with seizure type for Siena scalp database. ..... 88
Table 4.10 PIL, SEG, and optimal features for each patient obtained from the algorithm with the Siena scalp database. ..... 88
Table 4.11 Seizure prediction results obtained with the Siena scalp EEG database. ..... 89
Table 4.12 The perdition results of the real-time algorithm for each patient were obtained with the Siena scalp database. ..... 99

# List of Symbols

| | |
|---|---|
| $A$ | Mixing Matrix |
| C | Correlation Sum |
| D | Correlation Dimension |
| ER | Energy Percentage |
| IMFs | Intrinsic Mode Functions |
| J | Negentropy |
| K | Kernel Function |
| Kurt | *Kurtosis* |
| $L_{max}$ | Largest Lyapanove Exponent |
| M | System State Space |
| N | The Number of Inputs to Neural Network |
| $s$ | ICA Components |
| $S_{-}$ | Lower Envelopes |
| $S_{+}$ | Upper Envelopes |
| $s_n$ | System State |
| $S_x$ | Power Spectrum |
| T | Temperature |
| $W$ | De-Mixing Matrix |
| $x$ | EEG signals |
| X | Short Time Fourier Transform |
| α | EEG Alpha Band |
| β | EEG Beta Band |
| γ | EEG Gamma Band |
| δ | EEG Delta Band |
| Θ | The unit step function |
| θ | EEG Theta Band |
| $\sigma$ | Activation Function |
| $\Phi_m$ | Approximate Entropy |

# List of Abbreviations

| | |
|---|---|
| ADC | Analogue to Digital Convertor |
| ASR | Artefact Subspace Reconstruction |
| AUC | Area Under the Curve |
| BNNs | Biological Neural Networks |
| BSS | Blind Source Separation |
| CCR | Common Components Rejection |
| CDF | Cumulative Distribution Function |
| CHB-MIT | Children's Hospital Boston and the Massachusetts Institute of Technology |
| CNN | Convolutional Neural Network |
| CSP | Common Spatial Pattern |
| DL | Deep Learning |
| DWT | Discrete Wavelet Transform |
| ECG | Electrocardiography |
| EEG | Electroencephalography |
| EMD | Empirical Mode Decomposition |
| EMG | Electromyography |
| EOG | Electrooculography |
| ER | Energy Ratio |
| FA | False Alarm |
| fMRI | Functional Magnetic Resonance Imaging |
| FN | False Negative |
| FP | False Positive |
| FPR | False-Positive Rate |
| GRU | Gated Recurrent Unit |
| ICA | Independent Component Analysis |
| KKT | Karush-Kuhn-Tucker Conditions |
| LLE | Largest Lyapunov Exponent |
| LSTM | Long Short-Term Memory |
| LSTM | Long Short-Term Memory |
| MEG | Magneto Encephalography |
| MFCCs | Mel Frequency Cepstral Coefficients |
| ML | Machine Learning |
| NIRS | Near Infra-Red Spectroscopy |

| | |
|---|---|
| NLWPCER | Normalized Logarithmic Wavelet Packet Coefficient Energy Ratios |
| PC | Principal Components |
| PCA | Principal Component Analysis |
| PCD | Principal Components Data |
| PDF | Probability Density Function |
| PIL | Pre-Ictal Interval Length |
| PLI | Phase Lag Index |
| PSD | Power Spectral Density |
| RBF | Radial Base Function |
| ReLU | Rectified Linear Unit |
| RNN | Recurrent Neural Networks |
| ROC | Receiver Operating Characteristics |
| RT | Relative Tolerance |
| SA | Simulated Annealing |
| SEG | EEG Segment Length |
| SOP | Seizure Occurrence Period |
| SPH | Seizure Prediction Horizon |
| STFT | Short Time Fourier Transform |
| SVM | Support Vector Machine |
| t-test | Two sample unequal variance student test |
| TN | True Negative |
| TP | True Positive |
| WOSG | Wavelet Domain Optimized Savitzky–Golay |
| WPLI | Weighted Phase Lag Index |

# Chapter One

# Introduction

## 1.1 Preface

Automatic brain diseases state recognition and brain state monitoring systems becoming important medical assistant methods. Real-time brain state monitoring has a large impact on surgeries and on saving patient life. These developed monitor devices are based on measuring brain signals in a non-invasive way and can recognize different kinds of diseases.

These devices can be divided into four classes based on the signal acquisition method. The four non-invasive methods are Electroencephalography (EEG), Magneto Encephalography (MEG), Near Infra-Red Spectroscopy (NIRS), and functional Magnetic Resonance Imaging (fMRI) [1]. Each one of these methods has its pros and cons in the order of setup complexity, cost, spatial resolution, temporal resolution, and type of data that it produces.

The less costly and easiest setup method above is EEG; which represents the recording of the electrical activity of the human brain's neurons along the scalp. Since EEG signals have a huge impact on human nervous system diseases diagnostic; processing and analysing of EEG signals have been gaining great concern by researchers over the last two decades. EEG signals are very noisy and non-stationary signals, have a high temporal resolution, and suffer from low spatial resolution.

In addition to its non-stationary and non-linear nature, during the recording process of EEG signals, huge amounts of artefacts like power line noise, electromyography (EMG), electrocardiography (ECG) and electrooculography (EOG) may be superimposed with these signals, raising the probability of being unreadable signals or carrying wrong information about the state of nerve cells. In this case, the automatic disease classifier systems fail in the classification process and the diagnosing doctors require more effort to read these signals [2].

During the last four decades, researchers worked to invent algorithms to remove artefacts from EEG signals to speed up and boost the classification process. Different algorithms have been introduced such as Adaptive Filtering, Principal Component Analysis (PCA), Blind Source Separation (BSS), and Independent Component Analysis (ICA) [1].

Some EEG recording systems are equipped with additional electrodes to record probable artefact signals in conjunction with EEG channels so that these artefacts can be easily removed by projecting these additional channels to the original EEG channels [3].

Since most EEG recording devices lack the additional channels and most available EEG datasets for machine learning contain only principal EEG channels, other techniques like averaging the overall available channels or using the independencies property among channels may be used to remove artefacts.

By reviewing research published in the last two decades, it's noted that the best way to deal with and analyse non-stationary signals like EEG is by utilizing Machine Learning (ML). A large number of researchers employed ML for EEG signals processing and recognition. The processing procedure can be summarized as follows: recording EEG signals, filtering, pre-processing, feature extraction, and classification. Nowadays, the motivation of researchers becomes towards Deep Learning (DL). DL has the capability of dealing with noisy raw EEG signals and has inherent feature learning and classification property. Figure 1.1 (a) clarifies the structures of a system based on ML while Figure 1.1 (b) represents a system based on DL [4].

EEG signals have been widely employed in the therapy of one of the most common diseases of the central nervous system, which is epilepsy. Epilepsy is a widespread prolonged brain and nervous disease with nearly 50 million patients all over the world, the premature death rate for people with epilepsy is 2 to 3 times higher than for people without epilepsy, and it presents an intense problem for the patients, their relatives and the society [5, 6].

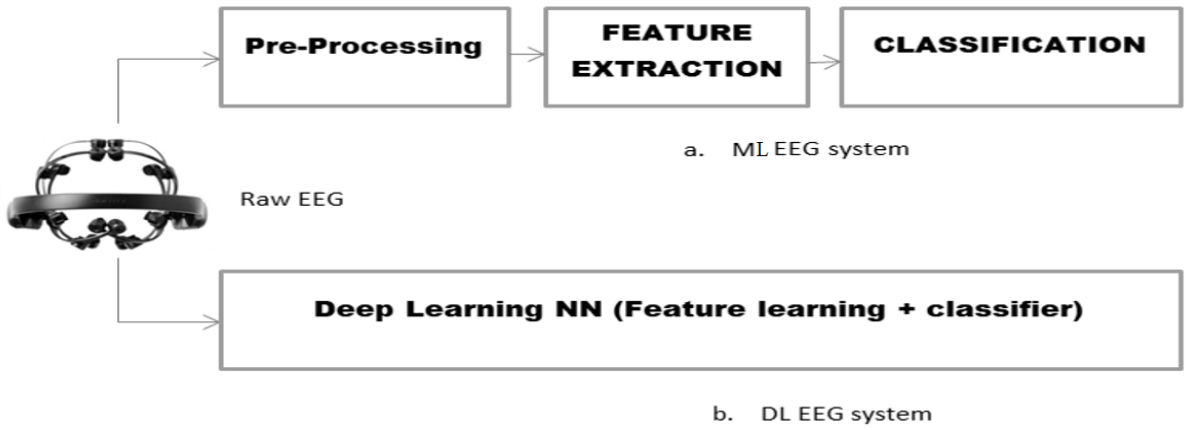



Figure 1.1 (a) ML EEG system, (b) DL EEG system [7].

EEG signal is commonly utilized in epileptic seizure detection and prediction, as epilepsy diagnosis can be performed by identifying the abnormalities of EEG signals [8]. Doctors classify the brain activity of epileptic patients into four states (To be discussed in chapter two), evolutions from the normal state to the abnormal one turns out to gradually change, and this is known as the pre-ictal state [9]. The existence of precautionary symptoms for a large group of epilepsy patients is also indicated in modern studies [10].

Most of the algorithms in researches succeeded in predicting the epileptic seizure in the patients participating in the test, but the issue of seizure prediction is still in the research and development stage to reach an algorithm with high accuracy and a little false prediction percentage and is being examined on a large segment of patients to be approved medically.

## 1.2 Literature Review

This section provides a concise review of the published research in the field of EEG signal processing and seizure prediction through the use of machine/deep learning strategies. It is divided into three parts:

1- Researches published in the field of artefact removal from EEG signals.
2- Researches published in the field of seizure prediction using deep learning.
3- Researches published in the scope of seizure prediction through feature extraction and machine learning.

### 1.2.1 Artefact Removal Research

In this section, the methods in the field of artefact detection and rejection are presented.

J. Dammers et al., **(2008)** [11], decomposed the MEG/EEG signals with ICA and presented a method for artefact suppression based on amplitude and phase statistics of decomposed MEG/EEG signals. They employed a quantitative measurement to control the performance of the artefact rejection. They achieved very high sensitivity in extracting heart beat artefacts.

F. Viola et al., **(2009)** [12], presented a method based on the correlation of ICA inverse weights that finds ICA components which are similar to the template defined by the user. The method selects the components whose similarity with templates passes a certain threshold. Their method provides a threshold automatically or by user input therefore it works in two modes: automatic and manual. The overlap degree between the identified ICs was observed to be > 0.8 for eye-related artefacts, and < 0.65 for heartbeat artefacts.

S. K. Goh et al., **(2017)** [13], employed ICA to decompose EEG signals, based on the assumption that artefacts have a larger amplitude than the signal of interest. They tested their method on EEG recording acquired from 6 subjects and synthesised EEG data, their method identified artefact components automatically and restored clean EEG from useful components.

C. Chang et al., **(2018)** [14], used Artefact Subspace Reconstruction (ASR) algorithm to automatically remove large amplitude artefacts. Their work is summarized as decomposing EEG with ICA, removing large amplitude components with an ASR filter, and reconstructing back EEG signals from cleaned ICA components.

K. Yasoda et al., **(2020)** [15], implemented the removal of artefacts of EEG signal via fuzzy kernel SVM to classify the artefacts in Wavelet-ICA. They used statistical features like mean, variance, standard deviation, and kurtosis to train the SVM classifier.

N. Bajaj et al., **(2020)** [16], suggested an algorithm based on Wavelet Packet Decomposition (WPD) that controls the removal of presumed artefacts, via tuning intuitive parameters. Their algorithm has two regulation parameters and three working modes. They evaluated their algorithm by visual inspection and spectral response and mentioned that their method is better than ICA-based methods.

P. Gajbhiye et al., **(2020)** [17], suggested the Wavelet domain Optimized Savitzky–Golay (WOSG) filtering for the removal of muscle artefacts from EEG signals. The multi-scale analysis using Discrete Wavelet Transform (DWT) of the EEG signals produced sub-bands at different scales. The optimized SG filter was applied to the muscle artefact intermixed approximation sub-band signal, and the artefact free approximation sub-band signal was computed based on the subtraction of the optimized SG filter output from the muscle artefact intermixed sub-band signal.

K. Jindal et al., **(2020)** [18], used a joint application of ICA and General Linear Chirplet Transform for automatic identification and rejection of artefact origins. The Katz-Fractal Sparsity criterion was employed to identify artefact components. The identified components were treated by the General Linear Chirplet Transform-based EEG de-noising method to recover useful cerebral information leaked with artefact origins.

S. Sangmin et al., **(2020)** [19], suggested a method using Bayesian deep learning and attention module to develop the performance of the classifier for ICA sources. The probability value was computed to predict if a component is an artefact and to treat vague inputs.

B. Abdi-Sargezeh et al., **(2021)** [20]**,** introduced two methods for removing EEG artefacts. The first method was performed by extracting the Common Components Rejection (CCR) among EEG channels and eliminating them, and the second method used wavelet decomposition to decompose the EEG signals and then applied the CCR method to remove artefacts in the time-frequency domain.

It is clear from the above that ICA is a very important method in EEG analysis and artefact detection, and although of mentioned research benefits artefact detection but none of them referred to the compensation of the artefact portion besides that many of these methods require an extra information channel to capture artefact reference. However, in this research (presented dissertation), a method for artefact rejection that can compensate for rejected portions is presented, the method has less computation complexity and did not require any extra information channel attached with EEG signals.

### 1.2.2 Seizure Prediction with Deep Learning Research

In this section methods in the field of seizure prediction employing deep learning are presented.

I. Kiral-Kornek et al., **(2018)** [21], trained a deep learning classifier (neural network) to classify pre-ictal and inter-ictal segments, the proposed prediction system attained average sensitivity of 69%, and PIL has been taken to be 15 minutes.

H. Daoud and M. Bayoumi, **(2019)** [22], suggested four deep learning (CNN) models for the seizure prediction, they used Raw EEG signals with no pre-processing as input to models, and they also introduced a channel selection procedure to select the most related EEG channels, they achieved accuracy of 99.6% and minimum false alarm rate of 0.004 with PIL of 1 hour, and multi-channel EEG with a segment size of 5-s.

Y. Xu et al., **(2020)** [23], proposed a deep learning solution using a convolutional neural network (CNN). The CNN model is gauged on CHB–MIT scalp EEG and Kaggle intracranial datasets. Global sensitivity reaches 93.5%, and the false prediction rate reaches 0.063/h in Kaggle intracranial dataset, while in CHB–MIT scalp EEG dataset, overall sensitivity reaches 98.8%, and the false prediction rate reaches 0.074/h. 16 channel EEG with a segment size of 20-s are used with 30min PIL and 5min seizure prediction horizon (SPH) chosen.

R. Jana and I. Mukherjee, **(2021)** [24], used CNN for feature extraction and classification, 6 EEG channels with a segment size of 8-s are used, and PIL has been considered to be only 2 min before seizure event. The average classification accuracy of 99.47 % is achieved.

T. Dissanayake et al., **(2021)** [25], suggested two patient-independent deep learning (CNN) architectures, the models achieve 88.81% and 91.54% accuracy respectively for seizure prediction on the Boston Children Hospital EEG dataset (CHB–MIT), a segment size of 10-s is used then they used the Mel Frequency Cepstral Coefficients (MFCCs) of the segmented signals as training data for the models, and the PIL of 1 hour is taken.

S. Usman et al., **(2021)** [26], solved the class imbalance challenge with generated pre-ictal segments using generative adversarial networks. The features extracted by a convolutional neural network with three layers, combined with handcrafted features from EEG signals are used as a feature array. The feature array is then used to train the classifier. An average specificity of 95.65% and sensitivity of 96.28% with an average seizure occurrence period (SOP) of 33 min on all subjects of the CHB–MIT dataset have been attained by this method.

It is concluded from the pre-mentioned reviews, that the researchers differed on a standard model that is a base for comparison. The researchers used different values for PIL, as well as different methods of training the deep learning models and different methods for evaluating the performance of these models. Therefore, it is difficult to relate the performance of the different models proposed by the researchers. So, in this research (presented dissertation), the focus will be on the comparison with the article [21], which uses values close to the values used in this research, as well as its dependence on the CHB-MIT database.

### 1.2.3 Seizure Prediction with Feature Extraction and Machine Learning Research

In this section, the methods in the field of seizure prediction employing extracting features and classification with machine learning are presented.

D. Snyder et al., **(2008)** [27], used a 5 s sliding window of EEG data displaced in 1 s increments, the beta-band power is used as a feature for k Nearest Neighbours classifier ($k$ = 15 neighbours) between inter-ictal and pre-ictal segments, PIL has been taken to be 1.5 hours, authors presented a statistical method for measure the functioning of EEG-based predictors compared with random (chance) predictors.

N. Wang and M. Lyu, **(2015)** [28], extracted amplitude and frequency features from EEG signals to form an initial feature set, and they used a feature selection process to select the features that increase the discrimination between inter-ictal and pre-ictal states. They used a segment size of 5 sec and achieved 98.8% sensitivity.

S. Elgohary et al., **(2016)** [29], suggested method depends on counting the number of null-crossings in wavelet factors of EEG signals. They used this number as an input feature to a binary classifier that distinguishes pre-ictal and inter-ictal states. They used a flexible procedure for EEG channel choice to find the optimal number of necessary channels. Their method achieved an average accuracy of 94% and sensitivity of 96% by training with only 10 minutes of data.

T. N. Alotaiby et al., **(2017)** [30], used Multichannel EEG signals and extracted Common Spatial Pattern (CSP) based features from EEG signals. They trained a linear discriminant analysis classifier, by leave-one-out cross-validation strategy. Their prediction obtained method achieves a mean sensitivity, and false prediction rate, of 0.89, and 0.39, respectively. They used a 60, 90, and 120-minute prediction horizon.

H. Chu et al., **(2017)** [31], developed a seizure predation method established on attractor state analysis. Their method depends on the power spectral density at low frequencies of the EEG signals which exhibits a comparative increase near a branching point making an abrupt evolution from normal state to seizure state. Their method yielded a sensitivity of (86.67%), a prediction time of 45.3 min on average, and a false prediction rate of 0.367/hour.

S. Usman et al., **(2017)** [32], applied empirical mode decomposition (EMD) to EEG signals prior to extracting time and frequency domain features. They assumed that the pre-ictal period is a few minutes before the seizure onset, and they achieved a true positive rate of 92.23% on the scalp EEG CHB-MIT dataset of 22 patients.

N. D. Truong et al., **(2018)** [33]**,** used Short-time Fourier transform on EEG data with 30 s windows to extract the frequency spectrum and time series information so that the algorithm automatically produces features for each subject. A convolutional neural network is used to classify these features as pre-ictal and inter-ictal states. Their approach achieves a sensitivity of 81.2% and a false prediction rate of 0.16/hour, on the CHB–MIT scalp EEG dataset.

H. Khan et al., **(2018)** [34], used the wavelet transformation of the EEG signal and Convolutional filters to learn assessable signs of seizure. They also calculated the optimum seizure prediction horizon from the data and found it to be 10 minutes. They achieved a mean sensitivity of 87.8% and a false prediction rate of 0.142 FP/hour.

K. Tsiouris et al., **(2018)** [35]**,** recommended Long Short-Term Memory (LSTM) models in epileptic seizure prediction employing EEG data. They utilised a two-layer LSTM network with four dissimilar lengths of the pre-ictal period, varying from 15 min to 2 h. By evaluating the method with the CHB-MIT Scalp EEG database, they achieved false prediction rates of 0.11–0.02 false alarms/hour, subject to the length of the pre-ictal window.

A. Al-Bakri et al., **(2018)** [36], used several electrophysiological features like signal power in the gamma, beta, alpha, theta, and delta bands; and the mean, standard deviation, skewness, kurtosis, etc., in addition to measurements like blood volume pulse, heart rate, and skin temperature; a Naive Bayes classifier were trained with these features. The classifier achieved 64% specificity, 69% sensitivity, and 33% kappa for pre-ictal sleep epochs; and 15%, 93% and 10% respectively for wake epochs.

H. A. Agboola et al., **(2019)** [37], used a method based on time-frequency analysis of EEG and unsupervised feature representation learning. They extracted the feature named Normalized Logarithmic Wavelet Packet Coefficient Energy Ratios (NLWPCER) to be the feature space of the classifier. Their prediction algorithm achieved a sensitivity of 87.26% and a false prediction rate of 0.08/hour with the SVM classifier, an average sensitivity of 75.49% and a false prediction rate of 0.13/hour with the ANN classifier.

P. Detti et al., **(2020)** [38], used two features, Phase Lag Index (PLI) and Weighted Phase Lag Index (WPLI), to discern between inter-ictal and pre-ictal segments, and obtained average specificity of 98.80 using data for eight subjects from Siena scalp database.

S. E. Sanchez-Hernandez et al., **(2022)** [39], investigated the influence of feature selection algorithms and classifier types on the classification of epileptic EEG signals, they found the best F1-score (0.90) with the K-nearest neighbour classifier. They reported that none of the feature selection algorithms is better than the others.

Based on the foregoing, it can be noted that the pre-ictal period PIL used by researchers varies, and the features used also varies. Therefore, we see that it is necessary to calculate the pre-ictal period for each person separately. The distinguishing features set are also different between people, so the distinguishing features should also be found for each person separately. So, we introduce a seizure prediction algorithm that takes these two points into account.

## 1.3 Problem Statement

The following are some problems in the field of EEG signal processing and epileptic seizures prediction:

- The non-linearity of the EEG signal made it extremely complex to analyse and extract information from it. During the observation of the patient to record these signals, it is exposed to a lot of external influences, which must

be disposed of. Therefore, the goal is to adopt a method to cancel these external influences and obtain a pure EEG signal.

- Epilepsy is a difficult challenge for the patient and his family, as it comes with sudden seizures, so expect the arrival of epileptic seizures in an appropriate period of time to take precautions to reduce its effects or to take medications to prevent its occurrence, which is important to organize the patient's life.
- Most of the seizure prediction methods suffer from being not medically approved due to a large number of false warnings or because it was not examined on a wide range of patients. Therefore, the aim is to develop a general algorithm, but their parameters could be modified to suit the patient's EEG records and turn into a patient-specific algorithm and thus could be medically approved.
- Further, most of the present algorithms use multi-channel EEG, in this work the aim is to minimize the number of channels in order to make the algorithm more practical in use.

## 1.4 Aims of the Work

The aim of this work is to design an epileptic seizure prediction system based on EEG signals, and other secondary goals can be summarised as follows:

1- Propose an algorithm for removing artefacts and fine-tuning EEG signals, even if there is no extra artefact channel available for the projection.
2- Suggest and evaluate deep learning models for seizure prediction.
3- Propose a new four-stage seizure prediction algorithm which is based on feature extraction and machine learning.
4- Propose a real-time seizure prediction algorithm which calculates the periodogram of incoming EEG segments, and uses the rapid frequency changes across bands as signs of incoming seizures.

## 1.5 Contributions

The main contributions of this work are:

1- EEG Artefact Rejection Algorithm, the algorithm devotes ICA to suppress multivariate EEG signals into its independent sources and uses a new similarity approach to automatically detect the artefact sources and remove them from the signals. ICA method has been chosen due to its effectiveness in finding the sources of multivariate signals like EEG.

2- Deep Learning Models for Seizure Prediction, five channels from raw EEG data are used as the input signal to deep learning models for essential feature mining from EEG data for the classification of pre-ictal and inter-ictal states. The proposal of a small number of EEG channels which is only five and a practical Pre-ictal Interval Length (*PIL*) of 10 minutes period makes the system more comfortable and less worrying.

3- A Machine Learning Algorithm for Seizure Prediction, the algorithm consists of four stages. The first stage is the EEG channel selection stage, the second stage is the stage of finding the optimum *PIL* and segment length (*SEG*), the third stage is the stage of extracting features from the selected EEG channel and the last stage is the stage of finding the set of features that gives the best prediction accuracy utilizing the Simulated Annealing method.

4- A Real time seizure prediction method based on PSD calculation.

## 1.6 The Proposed System Limitation

The training of the proposed models is patient specific, meaning that for each patient the model must be trained using the EEG recordings for that patient prior to using the model for prediction.

## 1.7 Dissertation Organization

The remainder of this dissertation is organised as follows:

**Chapter Two** reviews some medical and technical topics related to recording and processing EEG signals and brain diseases.

**Chapter Three** introduces the algorithms and models for artefact rejection, and seizure detection and prediction.

**Chapter Four** discusses the results obtained from the methods introduced in chapter three.

**Chapter Five** gives the conclusions gathered from the work and suggests some future research directions.

# Chapter Two
# Medical and Technical Overview

## 2.1 Introduction

Many technologies for processing and analysing EEG signals have been introduced during the late three decades. Depending on the non-linear and non-stationary nature of EEG signals, and can be considered stochastic processes, several methods of analysis gave useful results, such as Joint Time-Frequency representation, Dynamical Analysis, Independent Component Analysis, statistical methods, and Empirical Mode Decomposition.

The field of signal classification and pattern recognition, deep learning using neural networks and other classification schemes such as support vector machines have been widely used.

In this chapter, some medical and technical topics related to recording and processing EEG signals are reviewed.

## 2.2 Human Brain Anatomy

The human brain consists of three major portions: the cerebrum, cerebellum, and brain stem. The cerebrum could be divided into six divisions: frontal lobe, parietal lobe, temporal lobe, occipital lobe, insular lobe, and limbic lobe. The parietal lobe recognizes pain and taste and is concerned with actions associated with solving problems. The temporal lobe is involved in hearing and retention. The occipital lobe is concerned with actions associated with vision. The frontal lobe is generally linked with feelings, speech, puzzle solving, and muscle movements. It includes the primary motor cortex sited forward to the central gyrus as displayed in Figure 2.1 [40, 41].

The cerebellum generally controls complicated body actions, including coordination and muscle tone modulation.

The brainstem is positioned at the bottom of the brain and joins the cerebrum with the spinal cord. It controls important body behaviours, involving

consciousness, respiration, eye movements, chewing, the transmitting of carnal messages, and the heartbeat.

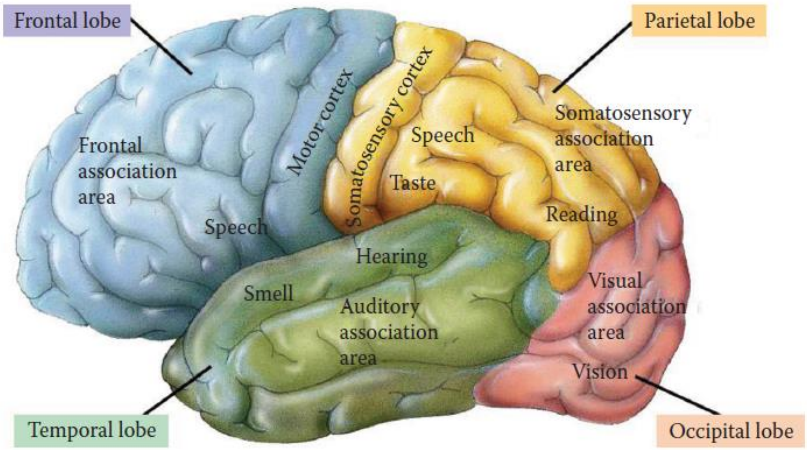



Figure 2.1 The human brain anatomy [1].

## 2.3 Human Brains' Neurophysiology

Neurons (or nerve cells) are operational units of the nervous system of the human body. A mature human brain consists of approximately 100 billion neurons [42]. Information is processed and transmitted by Neurons through chemical and electrical signals. A neuron consists of a soma, dendrites, and an axon (Figure 2.2) [2].

The information from other neurons is received by dendrites, the neuron makes a decision, and the decision send to other neurons across the axon. The neuron is enclosed by the sheath of the cell, which organises the movement of potassium (K+) and sodium (Na+) ions in and out of the cell. The charge outside the soma (cell body) is negative and its resting potential is between −70 mV and -60 mV [40]. The sheath potential turns out to be less negative due to the received electrical current across dendrites (portion A in Figure 2.3) [40] [43]. The cell sheath allows Na+ ions to enter the cell if this depolarisation reaches −55 mV, this results in a temporarily positive action potential (portion B in Figure 2.3) [40].

The cell sheath opens up once newly to allow the K+ ions to leave the cell (portion C in Figure 2.3); this causes the repolarisation of the potential at the sheath [41]. Due to the loss of cell sheath permeability for the K+ ions, the

potential temporarily drops to less than -70 mV causing hyperpolarization. In the end, the action potential steadies at the resting potential which is -70 mV.

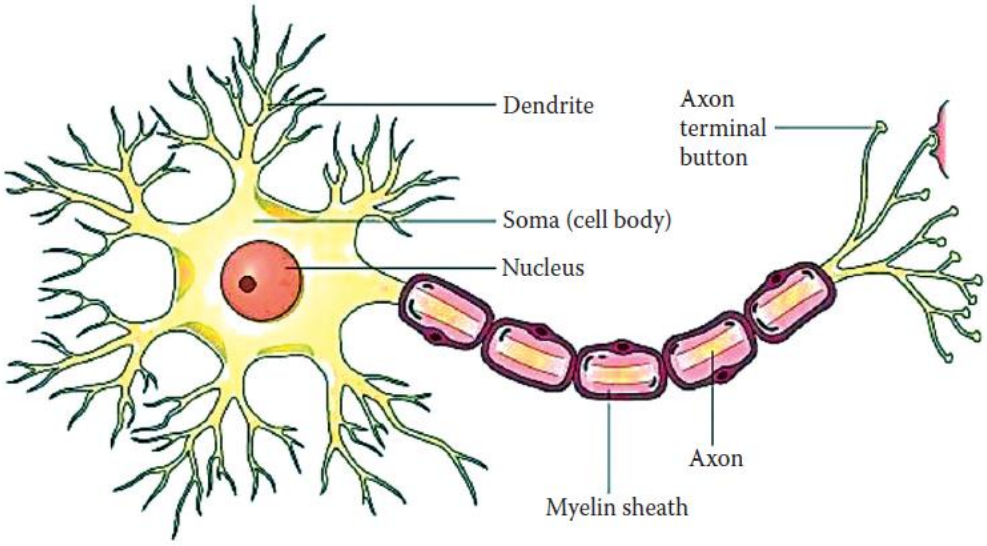



Figure 2.2 Neuron structure [1].

The action potential is transmitted across the axon to other neuron cells. Consequently, the neuron activates if the overall electrical currents from all the entering axons beat a specific level. In this way, the messages (the action potentials) transfer to neighbour neuron cells.

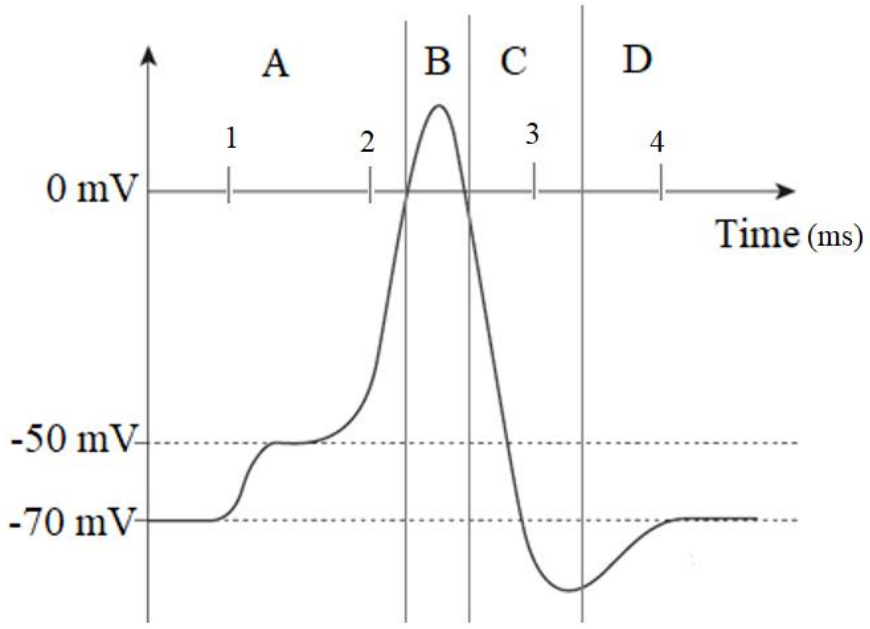



Figure 2.3 Action potential [2].

## 2.4 Electroencephalography Signals

Electroencephalography (EEG) signals are voltage fluctuations that can be sensed and recorded along the entire scalp (non-invasive) or by deeply implanted

electrodes inside the brain (invasive), these fluctuations are due to neuron's electrical activity which is defined as the flow of currents between neurons at firing process (the action potential). Historically EEG is discovered by Richard Caton (1842–1926) but the first one who measured the EEG in the human brain is Hans Berger (1873–1941) in 1929 [1]. Since that time many developments in EEG recording and interpretation methods are achieved, lastly, automatic interpretation is introduced.

The EEG recording cannot track the generated electrical potentials by only a single neuron; the EEG signals are a mirror of the total synchronic activity of a huge group of neurons having a corresponding spatial orientation [1, 44].

Since the voltage fields are inversely proportional to the square of the distance according to gauss law as in the Equation 2.1 [45]:

$$E = Q/(4\pi \varepsilon r^2) \tag{2.1}$$

where:

E: is the electric field strength

Q: is the quantity of electric charges

ε: permeability of the material

r: is the distance from the electric field source

and consequently, the well-aligned neurons of the cortex that are triggered simultaneously are believed to produce the majority of EEG activity. It's easy to detect the EEG signals from neurons that lie near the skull, while an invasive method may need to detect EEG signals from deep neurons [46].

## 2.5 Wave Patterns and Brain Rhythms

EEG signals are a reflection of neural activities and can be characterized by the amplitude, frequency, and phase of the fluctuations. Intellectual functions such as the moving of body parts and memory are related to neural activities and synchronization [47, 48]. These signals are frequently used to examine neural activities.

Neurons produce action potentials in a rhythmic pattern. These patterns are considered elementary for information coding. In various neurological syndromes, the root is extreme neural fluctuations. For example, in seizures, extreme synchronization has been noticed. Identical phenomena have been seen in patients having Parkinson's disease.

EEG signals are nonlinear, low amplitude, low frequency, and non-stationary combined with a large number of artefacts like Electromyography (EMG), Electrooculography (EOG), Electrocardiography (ECG), and external noise [49]. However, the human brain can generate five major brain waves or rhythms (see Figure 2.4) which can be categorized into five frequency bands as follows [2]:

1- Delta band (δ): 0.5-4 Hz, 20-100 µV amplitude .
2- Theta band (θ): 4-7.5 Hz, <10 µV amplitude.
3- Alpha (α): 8-13 Hz, with sinusoidal or round shape, 2-100 µV amplitude.
4- Beta (β): 14-26 Hz, <30 µV amplitude.
5- Gamma (γ): >30 Hz, 5-10 µV amplitude.

## 2.6 EEG Recording and Electrodes

EEG signals can be recorded in a non-invasive way using a certain brand of electrodes. These electrodes are made usually with stainless steel, silver, or gold, and placed along the scalp in different locations.

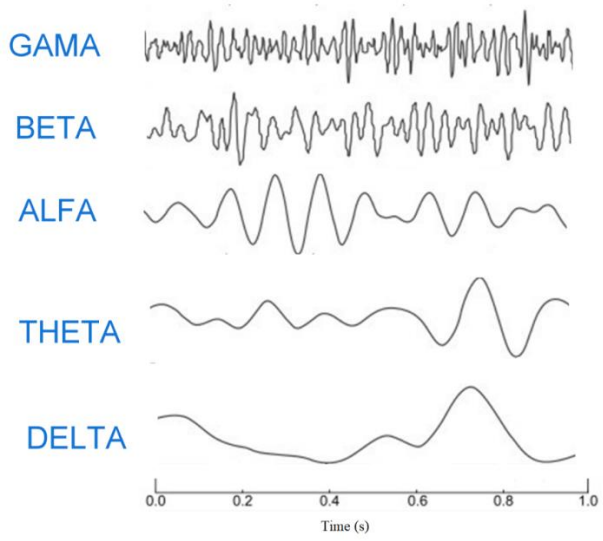



Figure 2.4 Representation of the normal brain rhythms [1].

To standardize the recording state, one should follow the international 10/20 system. Each electrode is placed at a position 10%-20% away from neighbourhood electrodes and they are given names that consist of letters and numbers. Figure 2.5 clarifies the meaning of the 10/20 proportion. The letter represents the lobe of the brain where the electrode is placed, for example, T for the temporal lobe and P for the parietal lobe. The odd numbers are assigned for electrodes in the left hemisphere and the even numbers for electrodes in the right hemisphere.

A high pass filter (corner frequency: 0.5 Hz) should be utilised to remove any low-frequency noise, a notch filter (corner frequency: 50 or 60 Hz) utilised to remove line noise, and a low pass filter (corner frequency: 70 Hz) utilised to remove any high-frequency noise. Analog to Digital Convertor (ADC) with sampling frequencies such as 100, 250, 500, 1000, and 2000 samples/sec should be used to convert EEG into digital form.

A very important concept in EEG signal recording is montage, which is defining the way that the voltage differences between electrodes are recorded. If the voltage between any two electrodes is considered, this is referred to us as bipolar montage. If a single electrode is used as a reference to all other electrodes, this is the referential montage. However, each electrode to its reference can be defined as a channel.

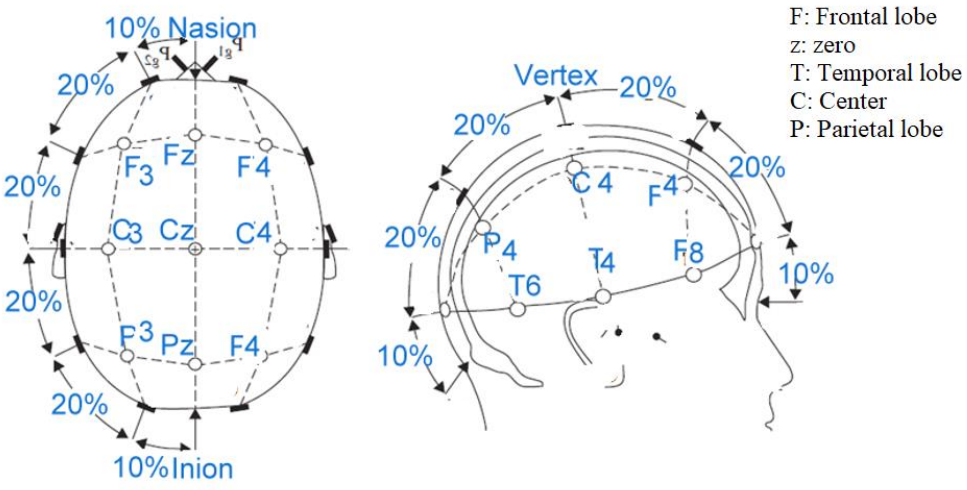



Figure 2.5 The scalp electrodes are placed with 10/20 proportion.

There are other important types of montage namely average montage, where the common reference is the average of all electrode's outputs, and the Laplacian montage, where the reference of an electrode is the weighted average of its surrounding electrodes. Because the recordings are stored in digital format; one can swap between montage types by simply re-referencing [2].

## 2.7 EEG Artefacts

EEG signals are recorded as voltage differences between recording electrodes over the scalp. These signals appear as voltage variations with respect to the recording time. These voltage variations convey more information about the brain state; however, the information is blurred by artefacts and noise. The source of the artefacts may be physiological such as head movements, eye movements, eye blinks, arm and leg movements, and heartbeat or may be external like power line interference, electrode movements, etc.

Some of these mentioned artefacts can be avoided during recording time by using shielding to reduce power line interference or requesting the patient to refrain from movements while recording. However, avoiding the artefacts is difficult or impossible; for example, heartbeat artefact. The most popular artefacts are electromyography (EMG), electrocardiography (ECG) and electrooculography (EOG). The following sections provide some useful notions about these artefacts and their frequency spectrum range.

### 2.7.1 Ocular Artefact

Ocular artefact originates from eye muscle movements which produce a square-like waveform and eye blinks which generate a spike-shape waveform [50] as shown in Figure 2.6. Ocular artefact is considered the most effective artefact to EEG signals due to its location near the brain, it can be recorded by an electrode near the eye, and the recorded signal is referred to as Electrooculography (EOG) signal. The amplitude of the EOG signal is greater than that of the EEG signal (10 uV-5 mV) but the frequencies of both signals are similar (0-100 Hz) [51].

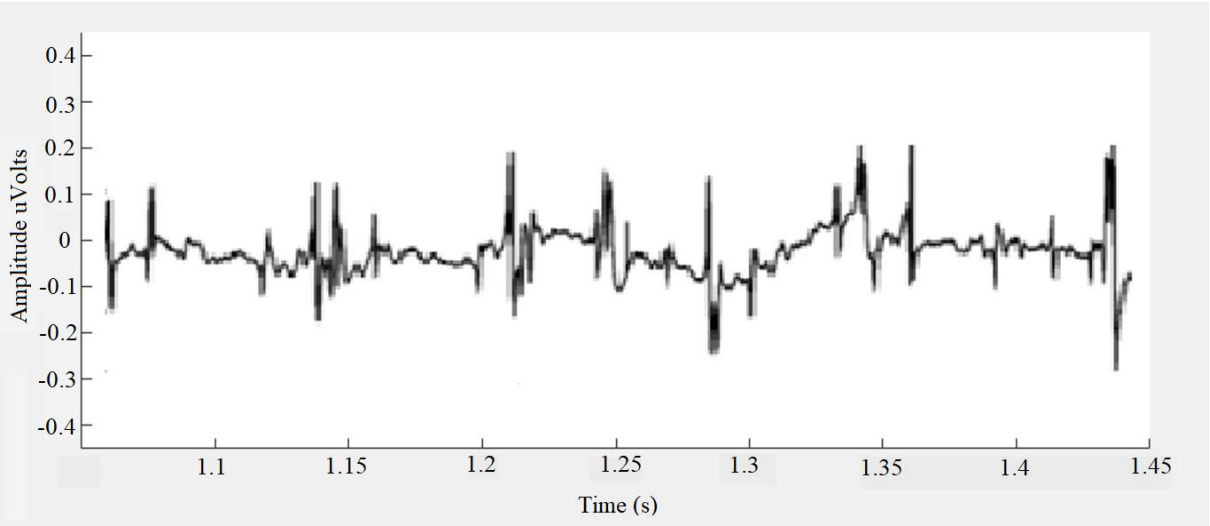



Figure 2.6 Sample EOG signal [51].

### 2.7.2 Muscle Artefacts

Muscle artefacts originated due to the movement of body muscles and are recorded by Electromyography (EMG). EMG signal amplitude ranges between 50 µV and 5 mV and it depends on the type and the location of the muscles being tensioned. EMG signals (Figure 2.7) have a frequency range between 2 Hz and 500 Hz [1].

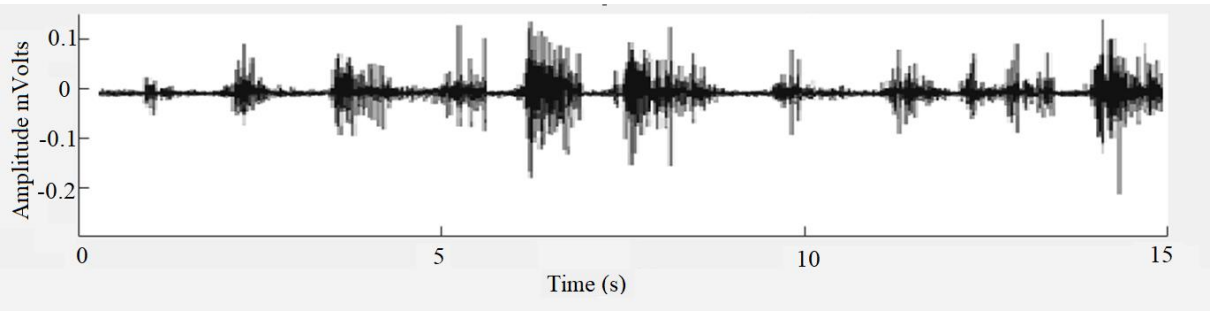



Figure 2.7 Sample EMG signal [51].

### 2.7.3 Cardiac Artefact

When an electrode is placed near a blood vessel, the movement of the vessel is recorded as an artefact to the EEG signal and is referred to as a pulse signal with a frequency of 1.2 Hz. The heart's electrical activity which has a regular pattern, recorded as an ECG signal is considered another artefact. This can be removed easily due to its regular shape for a specific subject. ECG signal

amplitude ranges between 1 mV and 10mV [1]. Figure 2.8 shows an example of an ECG signal.

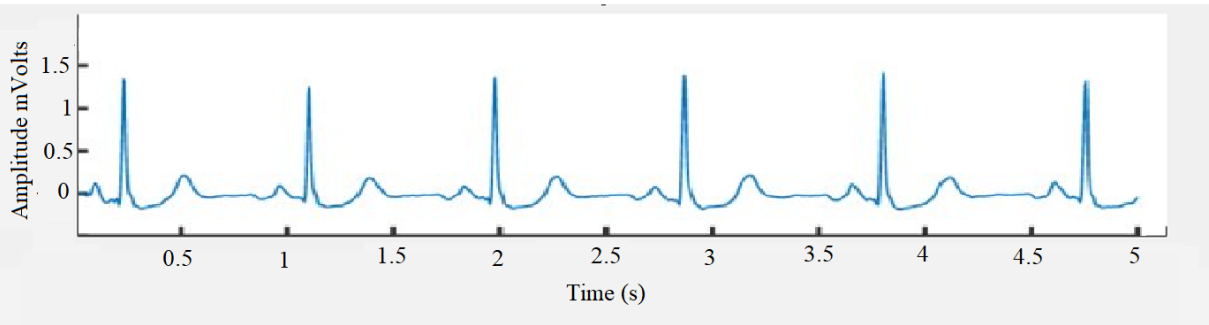



Figure 2.8 Sample ECG signal [52].

### 2.7.4 Other Artefacts and Artefact Handling

Many other types of artefacts are referred to as external or technical artefacts such as AC line, DC noise, improper electrode type and improper placement of electrodes, and power amplifier noise. Some of these artefacts may be avoided during recording.

AC power line artefact can be eliminated simply by using a notch filter at power line frequency (50 Hz or 60 Hz). If extra channels exist to record biological signals like EOG, EMG, and ECG, such artefacts can be easily removed from EEG by using adaptive filters taking the extra channel signal as a reference for the artefact being concerned with. In absence of extra channels, other techniques should be used such as Principal Component Analysis (PCA) and Independent Component Analysis (ICA) which convert multivariate signals to their original independent sources.

### 2.7.5 Independent Component Analysis (ICA)

Independent Component Analysis (ICA) is a technique that finds a linear transformation for a random vector, that minimizes the statistical reliance between its components [53]. Different algorithms had been proposed to compute ICA such as fast ICA [54, 55], Infomax [56], and Picard [57]. However, the goal of ICA is to find mixing matrix *A* that minimizes the dependencies between the

components *s*. If *x* is the random multivariate vector, then the relationship between *x* and *s* is as in Equation (2.2) [58]:

$$x = A\,s \tag{2.2}$$

where:

*x*: EEG signals

*s*: independent sources

*A*: mixing matrix

After estimating the mixing matrix, *A*, the independent components *s* can be found by Equation (2.3) [58]:

$$s = W\,x \tag{2.3}$$

where:

*W:* is the de-mixing matrix and it's equal to the inverse of mixing matrix *A*.

Independent means non-Gaussianity. Therefore, non-Gaussianity measurements are useful for ICA computations like Kurtosis which is a measure of the how often outliers occur and is given as in Equation (2.4) [58]:

$$Kurt(y) = E\{y^4\} - 3(E\{y^2\})^2 \tag{2.4}$$

Where: *E* is the expectation operator or an approximation of negentropy [58] like Equation (2.5) [58]:

$$J(y) \cong [E\{G(y)\} - E\{G(v\}]^2 \tag{2.5}$$

where *v* is the Gaussian variable with unit variance and zero mean and *G* is a non-quadratic function given by Equation (2.6) [58]:

$$G(u) = \frac{1}{a1} log(cosh(a_1\,u)) \tag{2.6}$$

Where:

$a_1$: is some suitable constant $1<a_1<2$.

With the fast ICA algorithm, the non-Gaussianity is measured by Equation (2.4). The computation element usually is a neural network that updates its weights *w* in order to maximize non-Gaussianity of $J(w^Tx)$. The steps of the fast ICA algorithm for single component estimation can be summarized as:

1- Initialize a random weight vector w.

2- Compute $w^{new} = E\{xg(w^T x)\} - E\{g'(w^T x)\}w$ (2.7)

where: g is a non-quadratic function given in Equation 2.6.

$g'$ is the derivative of the function in Equation 2.6.

3- Compute $w = w^{new}/\|w^{new}\|$ (2.8)

4- If w and $w^{new}$ are not converged, or their dot product is not equal to 1, then go back to step 2.

ICA algorithm for multiple components estimation needs multiple computation elements or multiple neurons, i.e., one neuron for each component.

Assuming there are *p* components to estimate, *p* vectors $w_1, \ldots, w_p$ need to be estimated. The vector $w_1$ will be estimated by a single component estimation algorithm and after each iteration the projections for previously estimated vectors i.e. $w_{p+1}^T w_j w_j$ for j= 1, … , *p* are subtracted from $w_{p+1}$. After that, $w_{p+1}$ is normalized by using the following steps:

1- Compute $w_{p+1} = w_{p+1} - \sum_{j=1}^{p} w_{p+1}^T w_j w_j$ (2.9)

2- Compute $w_{p+1} = w_{p+1}/\sqrt{w_{p+1}^T w_{p+1}}$ (2.10)

Using ICA for artefact handling is straightforward; where after applying ICA to EEG signal, the components that represent artefacts are identified and discarded during restoring back the pure EEG from its independent components.

## 2.8 Brain Disease- Epilepsy

Epilepsy is a widespread neurological situation that invokes recurrent and malicious seizures. Epilepsy is a central nervous system disease that triggers abnormal behaviour and even loss of consciousness in a patient. About seventy million people around the globe are smitten by epilepsy. The epileptic seizure is due to some of the brain cell damage or genetics in nature [59].

While most seizures are not life-threatening, the early death rate of epilepsy patients is 2 to 3 times that of normal persons, and it presents a serious problem for the patients and society [5, 6]. EEG signals are widely used in seizure detection and diagnosis.

Doctors classify the brain situations of the epileptic subjects according to the EEG recordings into four states: the inter-ictal state which refers to the period that not belongs to the other three states that are: the pre-ictal state, which is identified by the time duration before the seizure arrival, the ictal state which is identified throughout the seizure occasion, and the post-ictal state that is assigned to the time after the seizure ends, these four states are clarified in Figure 2.9 [60].

Epileptic seizure-related machine learning topics are classified into three common classes. In the epileptic seizure prediction class, the aim is to predict a seizure event by recognizing the pre-ictal state of the brain for a given patient. In the seizure detection class, the aim is to distinguish between seizure (ictal) and non-seizure (inter-ictal) EEG segments. The third class aims to distinguish specific epileptic seizure types (e.g., focal or non-focal seizures) [25]. Here, in this work, epileptic seizure prediction is emphasised because of its significance in improving a patient's daily life.

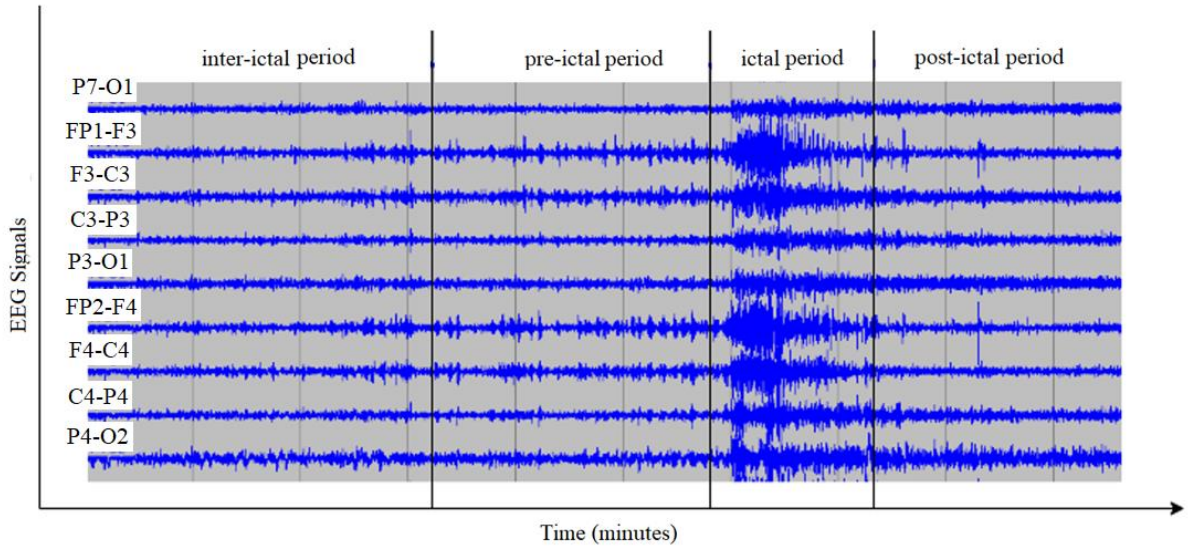



Figure 2.9 The four states of epileptic brain activity [7].

### 2.8.1 Types of Epileptic Seizures

Seizures are grouped into two focal groups based on the origin brain zone of the seizure. Generalized seizures affect nearly the entire brain, partial or focal seizures begin from a certain brain zone and may propagate to other regions of the brain.

#### *2.8.1.1 Generalized Seizures*

Generalized seizures affect nearly the whole brain area and may be classified into:

1- Petit Mal: They act for loss of awareness that usually continues for 20 s. Petit mal seizures are also called absence seizures.
2- Tonic-Clonic: In the tonic period, a patient could lose awareness for a while, and in the clonic period his body starts vibrating and moving in irregular muscular contractions. Tonic-clonic seizures are also called grand mal seizures.
3- Myoclonic: Myoclonic seizures are obvious as slight shakes in the body.
4- Atonic: Atonic seizures are obvious as the absence of arms control and they typically continue for less than 15 s.

#### *2.8.1.2 Partial Seizures*

Partial seizures, or focal seizures, begin from a specific brain region and could propagate to other regions or the whole brain, focal seizures are categorized into:

1- Simple Partial Seizure: This type does not cause loss of awareness, but it may be a symbol of more dangerous upcoming seizures. Simple partial seizures are also named Aura.
2- Complex Partial Seizure: This type causes weakening of awareness or reaction.

### 2.8.2 Predictability of Seizure from the EEG Signals

Seizures occur when a very large cluster of neurons in the cerebral cortex is excited at the same time, giving a synchronized electrical activity that disturbs the natural brain activity. Occasionally, such disruption is manifested by a brief impairment of awareness, but also more complicated series of irregular sensory and motor exhibitions can occur. This causes annoyance and embarrassment to the person with unpredictable time.

The brain is a complex dynamical system, and neuronal networks are basically complex non-linear structures, therefore, their collaborations are also complex and nonlinear in nature. Seizure syndrome recurrence is not absolutely predictive [61], but some patients can self-predict seizures [62] making the possibility of changing EEG signals gradually before seizure onset.

Non-linear dynamics reinforce the theory that seizures can be predicted by analysing the dynamics of EEG signals, while traditional analysis methods fail to detect changes prior to seizure occurrence. The application of nonlinear dynamics to epilepsy seizure prediction is done by Iasemidis et al. [63]. Their concept is that a seizure characterizes an evolution of the epileptic brain from a disorganised state to a more organised state, and consequently, the epileptic brain's dynamical properties are different for different clinical states. Based on the growth of the short-term largest Lyapunov exponent (LLE) with time, they advised that the EEG signals become gradually less chaotic with time as the seizure reaches [64]. Hence, the impression that seizures occur suddenly was replaced with the opinion that the brain follows a dynamic evolution to a seizure state for at worst some forms of epilepsy. Nevertheless, evolutions from the inter-ictal stage to the ictal stage turn out to be gradual changes, and this is known as the pre-ictal state [9].

Researchers suggested many algorithms for seizure prediction, but obtaining a general and accurate algorithm to predict seizures is very difficult due to the following reasons:

1. EEG signals are extremely complex and vary unevenly throughout time [65].
2. The pre-ictal and the inter-ictal states are highly variable among patients.
3. The distinction between pre-ictal and inter-ictal states in an individual is vague and unclear.

There are no fixed rules used in the seizure prediction algorithms, but it is possible to summarize the common points and the sites of difference in the following:

1- The majority of researchers have used EEG signals as a source of information.
2- Different numbers of EEG channels have been employed.
3- Different values of the pre-ictal period have been utilized.
4- Different features are extracted and utilized.
5- Different types of feature learners and feature classifiers have been utilized.

## 2.9 Feature Extraction Methods

The following subsection provides the basics of some theories for feature extraction methods that are used widely in non-linear signal analysis such as EEG signals.

### 2.9.1 Empirical Mode Decomposition

The Empirical Mode Decomposition (EMD) method dissolves an input time sequence *x(t)* into Intrinsic Mode Functions (IMFs) and a residual in a repeated manner. The essential task of the method is basically filtering a sequence *x(t)* to get a new function *Y(t)* as follows: Finding the local maxima and minima of *x(t)*, constructing lower envelope s−(t) and upper envelope s+(t) of x(t) from local extremes, then constructing the residual Y(t) from the mean of the envelopes m(t) as in Equation (2.11) [66, 67, 68], and the details of decomposition process are found in [69].

$$Y(t) = x(t) - m(t) \tag{2.11}$$

The decomposition process is summarized in the following steps [69]:

1- Assume $r_0(t) = x(t)$, and let $i = 0$.
2- Before filtering, check $r_i(t)$:
    a. Find the number of local extreme (TN) of $r_i(t)$.
    b. Find the energy ratio (ER) of $r_i(t)$ using Eq. (2.12):

$$Energy\ Ratio = 10\ log_{10}(\frac{||x(t)||_2}{||r_i(t)||_2}) \tag{2.12}$$

3- If (TN < Maximum Number of Extreme) or (ER > Maximum Energy Ratio) or (number of IMFs > Maximum Number of IMFs) then break the decomposition.
4- Assume $r_{i,Prev}(t) = r_i(t)$.
5- Filter $r_{i,Prev}(t)$ to obtain $r_{i,Current}(t)$.
6- Check $r_{i,Current}(t)$:
    a. Find the relative tolerance (RT) of $r_{i,Cur}(t)$ using Eq. (2.13):

$$Relative\ Tolerance = \frac{||r_{prev}(t) - r_{cur}(t)||_2^2}{||r_{prev}(t)||_2^2} \quad (2.13)$$

    b. Get current filter iteration number (IN).
7- If (IN > Sift Max Iterations) or (RT < Sift Relative Tolerance) then break. An IMF has been found: IMFi(t) = $r_{i,Cur}(t)$. Otherwise, assume $r_{i,Prev}(t) = r_{i,Cur}(t)$ and go to Step 5.
8- Assume $r_{i+1}(t) = r_i(t) - r_{i,Cur}(t)$.
9- Assume i = i + 1. Return to Step 2.

### 2.9.2 Short Time Fourier Transform

Using Short Time Fourier Transform (STFT), the EEG records are partitioned into small portions, and afterwards, a Fourier Transform is devoted to all portions after alternating it by an adequate window function [1]. Hence, the STFT could be described as in Equation (2.14) [1]:

$$X(m, \omega) = \sum_{n=0}^{N-1} x(n) w(n - m) e^{-j\omega n} \quad (2.14)$$

where *m* depicts the location of the window function *w(n)*, and the power spectrum is defined as in Equation. (2.15) [1]:

$$S_x(n, \omega) = |X(n, \omega)|^2 \quad (2.15)$$

### 2.9.3 Chaos theory and Dynamical Analysis

The non-linear nature of EEG signals makes their analysis an extremely hard task. EEG measurements show a non-linear dynamical system that can be understood through deterministic chaos theory [70].

Chaotic signals are random in their appearance but they are controlled by deterministic rules, also they are sensitive to initial conditions. Sensitivity to initial conditions means any little change in initial conditions will lead to large changes in the outcome. This section gives a brief introduction to different methods used in chaotic signal analysis and forecasting like Approximate Entropy, Lyapunov Exponents, and Correlation Dimension.

#### 2.9.3.1 State Space Reconstruction:

The State space of a system is a vector space of that system, each point in this vector space stands for an instantaneous state of the system and is covered by the embedding vectors. The evolution of the embedding vectors with time creates attractors.

An attracter is a curve that represents the possible future states of a system, irrespective of the preliminary circumstances of the system. If the curve has a single point, then it's called a point attracter. If it has a periodic set of points, then it's called a cyclic attracter. If it has aperiodic (chaotic) states, then it's called a strange attracter, this type of attracter is sensitive to initial conditions of the system and it's involved in many physical systems.

The characteristics of the attracter provide more information about the behaviour of the system; therefore, reconstruction of the state space is a very important step in the dynamical analysis. All variables of the system required reconstructing the state space of the system, and if the variables are unknown the reconstruction process fails.

Fortunately, state space may be reconstructed from output time series measurements of the system by embedding methods [71, 72, 73]. Let $M$ be the state space of the system, $s_n \in M$ is the state at time $n$. The system output is a function of state space $y_n = g(s_n)$ and is observed as time series. Since the dimension of $M$ is much larger than the dimension of $y$, it is needed to group several elements of y in order to reconstruct $M$. Take successive $y_n$ values with the method of delays to create a vector as shown in Equation (2.16) [70]:

$$y = [y_n \; y_{n-\tau} \cdots \cdots . y_{n-\tau(m-1)}] \quad (2.16)$$

τ is the lag time and it's chosen to be equal to the time at which the autocorrelation function of $y_n$ reaches null value or the time at which the mutual information function reaches its first minimum, *m* is the embedding dimension of the system and usually is taken to be ≥ d+1, where d is the dimension of the hyper-surface (local dimension) that constructed from system variables.

### 2.9.3.2 Approximate Entropy

Approximate Entropy is a constancy measurement that tells the uncertainty of variations in a time sequence. A comparatively larger rate of Approximate Entropy signifies the possibility that patterns are not pursued by extra identical patterns [74].

Approximate entropy is calculated as follows:

1. First generate a delayed restoration for N instants with lag $\tau$ and embedding dimension *m*.
2. Calculate the number of instants that are near the point *i*, as in Equation (2.17) [74]:

$$N_i = \sum_{i=1, i\neq k}^{N} \gamma(||Y_i - Y_k||_\infty < R) \quad (2.17)$$

where:

$N_i$: number of instants that are near the point *i*

**γ:** is the guide function

**N:** total number of points around point *i*

*R:* is the radius of relationship.

3. The Approximate Entropy is subsequently equal to $\Phi_m - \Phi_{m+1}$, where $\Phi_m$ is given by Equation (2.18) [74]:

$$\Phi_m = \frac{1}{(N-m+1)} \sum_{i=1}^{N-m+1} log\,(N_i) \quad (2.18)$$

### 2.9.3.3 Correlation Dimension

Correlation dimension D is an estimation of local dimension d of the system which is very important in the complexity analysis of the system. The large correlation dimension corresponds to chaotic and random signals, whereas the low correlation dimension corresponds to deterministic signals. D is defined as in Equation (2.19) [75, 76]:

$$D = \lim_{\varepsilon \to 0} \lim_{N \to \infty} \frac{\partial ln\, C(m,\varepsilon)}{\partial ln\, \varepsilon} \tag{2.19}$$

where *C(m,ε)* is a correlation sum of *N* points in state space and is defined as a sum of points that have a distance of $\varepsilon$ between them, and is given by Equation (2.20) [75, 76]:

$$C(m,\varepsilon) = \frac{2}{N(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \Theta(\varepsilon - \|y_i - y_j\|) \tag{2.20}$$

where Θ is the unit step function.

### 2.9.3.4 Maximum Lyapunov Exponent

The maximum Lyapunov exponent indicates the degree of disorder in a dynamic system. It determines with time, the aggressive divergence or convergence of two state space trajectories which are close in their initial condition. If two trajectories converge the negative Lyapunov exponent will produce a deterministic system, while the positive Lyapunov exponent came from chaotic systems. Lyapunov exponent also indicates the epochs of time where predictions are possible.

Wolf's algorithm is a method for estimating the maximum Lyapunov exponent [77]. Wolf's algorithm can be summarized as follows:

1- Construct m dimensional embedding vectors $s(t) = [s(t), s(t - \tau), s(t - (m-1)\tau)]$.
2- Search for the nearest neighbour for one of the vectors and calculate the distance $L$.
3- Evolve the system for some time duration, and calculate the new distance $L'$.

4- Repeat step 3 until $L'$ becomes larger than a specified threshold.

5- Search for a new vector with the same direction as the first neighbour.

6- Estimate the Lyapunov exponent from the Equation (2.21) [77]:

$$L_{max} = \frac{1}{N_a \Delta t} \sum_{i=1}^{N_a} ln \frac{\left|s(t_i+\Delta t)-s(t_j+\Delta t)\right|}{\left|s(t_i)s(t_j)\right|} w \quad (2.21)$$

Where:

$s(t_i)$: is a point at the fiducial trajectory,

$s(t_j)$: is a point at the perturbed trajectory,

$\Delta t$: is the evolution time,

$N_a$: represents the total local $L_{max}$ approximated every $\Delta t$ within the duration T data portion,

and $t_i = t_0 + (i-1)\Delta t$, $t_j = t_0 + (j-1)\Delta t$, where $i \in [1, Na]$ and $j \in [1, Na]$ with j ≠ i.

### 2.9.4 Simulated Annealing

Simulated annealing (SA) is a probabilistic method offered by Kirkpatrick, Vecchi and Gelett (1983), and Cerny (1985) for locating the universal minimum point of a function that has numerous local minima points. It imitates the physical process through which a metal is gradually cooled so that when sooner or later its structure is "frozen," this takes place at a minimum energy arrangement [78]. The basic components of simulated annealing are the following:

1- A finite set *S*; the set of extracted features.

2- A cost function *K* is defined on *S*; the prediction accuracy will be the cost function in the proposed algorithm.

3- For each $i \in S$, a set $S(i) \in S - \{i\}$ is the neighbour set of *i*.

4- A *decreasing* function $T: N \rightarrow (0, \infty)$, called the cooling list. Here *N* is the positive integer set, and *T(t)* is called the temperature at time t.

5- An initial state $x(0) \in S$.

6- If the current state *x(t)* is equal to *i*, choose an adjoin *j* of *i* by random permutation. Once *j* is chosen, the next state *x(t + 1)* is determined by the following procedure:

If $K(j) \leq K(i)$:
$x(t+1) = j$
If $K(j) > K(i)$:
$x(t+1) = j$ with probability $exp[-(K(j) - K(i))/T(t)]$
$x(t+1) = i$ otherwise.

## 2.10 Machine Learning and Classification Methods

Machine Learning is the ability of machines to learn from data [79]. What the machine learns from data depends on the required task. Classification of data is one of the tasks required by machine learning. Where different portions of data are classified into different classes depending on some relevant features.

The first step that enables the machines to learn is the feature extraction step, where relevant features are extracted from given data then a learning algorithm is used to learn which features belong to which class.

Supervised and unsupervised algorithms are the types of learning algorithms in Machin Learning. The data with its associated class names are inserted into learning algorithms in supervised learning algorithms like linear regression, decision trees, and support vector machines. In unsupervised learning algorithms, there are no associated class numbers thus the algorithm tries to find a representation of data and discriminate between classes like k-Means Clustering, and Principal Components Analysis (PCA).

### 2.10.1 Support Vector Machine

Support Vector Machine (SVM) belongs to the supervised learning class of Machine Learning algorithms [80] and is the most popular used for Regression and Classification problems in Machine Learning. SVM algorithm creates the best decision boundary in *n*-dimensional space between two data sets where each set is belonging to a different class. This boundary is called a hyper-plane and it will be in *n-1* dimension. Kernel trick gives SVM the ability of non-linear

classification problems; where there is no linear decision boundary found between two classes.

Given a feature dataset $D = \{(xi, yi)|xi \in R^n, yi \in \{-1,1\}\}_{i=1}^{m}$, and $R^n$ is the vector space with n dimension, SVM defines the equation of decision boundary that separates feature points into two classes. The equation of the decision boundary is called a hyper-plane equation. The hyper-plane equation H that divides the points into two regions, in general, can be written as in Equation (2.22) [81]:

$$H: w.x + b = 0 \tag{2.22}$$

where w is the slope of the hyper-plane and b is a constant.

The hyper-plane should be the best possible plane that makes classification with minimum error, this requires that the distance $d_H$ from the hyper-plane equation and a given point vector $x_0$ should be maximum. The distance $d_H$is given by Equation (2.23) [81]:

$$d_H(x_0) = \frac{|w.x_0+b|}{||w||^2} \tag{2.23}$$

where $||w||^2$ is the Euclidean norm for the length of *w* and is given in Equation (2.24) [81]:

$$||w||^2 = \sqrt{w_1^2 + w_2^2 + w_3^2 + \cdots + w_n^2} \tag{2.24}$$

So, the goal is to maximize minimum distance $d_H$ in order to achieve high accuracy classification (Equation (2.25)) [81]:

$$w^* = arg_w max[min_n d_H(x_n)] \tag{2.25}$$

During training, if the data point is substituted from the positive group in the hyper-plane equation, the value of the prediction will be greater than 0 and it is represented by Equation (2.26) [81]:

$$w.x + b > 0 \tag{2.26}$$

While predictions from negative group points will give values less than 0 and it represented by Equation (2.27) [81]:

$$w.x + b < 0 \tag{2.27}$$

During testing, the prediction of a positive class correctly as positive will make the result greater than zero (two positives make positive). So, the prediction of the negative class correctly will again make the result greater than zero (two negatives make a positive). However, miss classifying will make the result less than zero (one minus, and one plus make a minus). Therefore, the result of multiplying a predicted label with the actual label would be positive on correct predictions, otherwise, it will be negative as shown in Equation (2.28) [81]:

$$y_n[w.x + b] = \begin{cases} \geq 0 \; for \; correct \; predictions \\ < 0 \; for \; incorrect \; predictions \end{cases} \quad (2.28)$$

The optimal hyper-plane categorizes all the points correctly if the dataset is perfectly separable and substituting the optimal values in the weight equation gives Equation (2.29) [81]:

$$w, b^* = arg_{w,b} max \left[ min_n \frac{y_n |w.x_n + b|}{||w||} \right] \quad (2.29)$$

Taking the independent term outside gives (Equation (2.30)) [81]:

$$w, b^* = arg_{w,b} \max \frac{1}{||w||} [min_n \; y_n |w.x_n + \; b|] \quad (2.30)$$

The term $min_n \; y_n |w.x_n + \; b|$ represents the minimum distance of a point to the decision boundary, re-scaling the distance of the closest point to be 1 i.e. $min_n \; y_n |w.x_n + \; b| \; = \; 1$ by substituting $w \rightarrow cw, b \rightarrow cb$ will maintain the vectors in the same direction and the hyper-plane equation will not change (Equation (2.31)) [81]:

$$(cw.x_n) + (cb) = c(w.x_n + b) = 0 \quad (2.31)$$

Now the equation will describe every point to be at least 1/||w|| distance apart from the hyper-plane as in Equation (2.32) [81]:

$$w^* = arg_w \, max \frac{1}{||w||}, \qquad s.t. min_n \; y_n \; [w.x_n + \; b] = 1 \quad (2.32)$$

#### 2.10.1.1 Perfect Separation SVM

For datasets that are perfectly separable, the decision boundary will be perfect for all points and there is no classification error, the minimization problem is a convex quadratic optimization problem as in Equation (2.33) [81]:

$$min_{w,b} \frac{1}{2} ||w||^2; \qquad s.t. \; y_n \; [w.x_n + \; b] \geq 1, \forall n \quad (2.33)$$

The convex quadratic optimization problem can be solved by the Lagrange multiplier method. Lagrange method in finding the minimum value of *f* under the equality constraint *g* is solving the Equation (2.34) [81]:

$$\nabla f(x) - \alpha \nabla g(x) = 0 \tag{2.34}$$

where α is the Lagrange multiplier. Let$f(w) = \frac{1}{2}\|w\|^2$, $g(w,b) = y_n\,[w.x_n + b] \geq 1, \forall n$. The Lagrangian function is then will be as in Equation (2.35) [81]:

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{n=1}^{m} \alpha_n [y_n(w \cdot x + b) - 1] \tag{2.35}$$

To solve $L(w,b,\alpha) = 0$ rewrite it using the duality principle, and the equations for the dual problem are Equation (2.36) and Equation (2.37) [81]:

$$\nabla_w L(w,b,\alpha) = w - \sum_{n=1}^{m} \alpha_n y_n x_n = 0 \tag{2.36}$$

$$\nabla_b L(w,b,\alpha) = -\sum_{n=1}^{m} \alpha_n y_n = 0 \tag{2.37}$$

Equations (2.30) and (2.31) yield $\sum_{n=1}^{m} \alpha_n y_n = 0$ and $w = \sum_{n=1}^{m} \alpha_n y_n x_n$, substituting these two values into the Lagrangian function L yields Equation (2.38) [81]:

$$W(\alpha,b) = \sum_{n=1}^{m} \alpha_n - \frac{1}{2}\sum_{n=1}^{m}\sum_{p=1}^{m} \alpha_n \alpha_p y_n y_p x_n \cdot x_p \tag{2.38}$$

Thus, the problem will be as in Equation (2.39) [81]:

$$max_a \sum_{n=1}^{m} \alpha_n - \frac{1}{2}\sum_{n=1}^{m}\sum_{p=1}^{m} \alpha_n \alpha_p y_n y_p x_n \cdot x_p \,;\; s.t.\; \alpha_n \geq 0,$$
$$\forall n, \sum_{n=1}^{m} a_n y_n = 0 \tag{2.39}$$

Now, the objective function depends only on the Lagrangian multipliers, which is uncomplicated in solving analytically. Actually, the Lagrange multipliers method will be extended to the KKT (Karush-Kuhn-Tucker) conditions, since the inequality constraints. The complementary slackness condition of KKT states that (Equation (2.40)) [81]:

$$\alpha_n [y_n(w \cdot x^* + b) - 1] = 0 \tag{2.40}$$

Where $x^*$ represents the point/points where the optimal is reached. The value of α is positive for these points, so $y_n(w \cdot x^* + b) - 1$ must be zero. These points are termed support vectors, which are the nearest points to the hyper-plane.

After solving the dual problem, w and b, which determine the optimal hyper-plane, can be computed as in Equation (2.41) and Equation (2.42) [81]:

$$w = \sum_{n=1}^{m} \alpha_n y_n x_n \quad (2.41)$$

$$b = y_n - w \cdot x^* = \frac{1}{S} \sum_{n=1}^{S} (y_n - w.x) \quad (2.42)$$

where *S* representing the number of support vectors.

### 2.10.1.2 Non-Perfect Separation SVM

In reality, there are no datasets that are perfect separable, so the model will make a few mistakes while classifying the points and the constraint in equation (3.34) will become as in Equation (2.43) [81]:

$$y_n \, [w.x_n + \, b] \leq 0, \exists n \quad (2.43)$$

Therefore, a slack variable $\beta$ will be added as a penalty for every classification error for each data point. Now, the constraint could be satisfied even if the example does not meet the original constraint. So, if the data point is classified correctly β will be equal to 0, otherwise, β will be greater than 1. So, the optimization problem will be as in Equation (2.44) [81]:

$$min_{w,b,\{\beta\}} \frac{1}{2} \|w\|^2 + C \sum_n \beta_n \; s.t. \; y_n \, [w.x_n + \, b] \geq 1 - \beta_n, \forall n; \; \beta_n \geq 0, \forall n \quad (2.44)$$

where *C* is a regularization hyper-parameter for slack variable β, the value of C regulates how important β should be, which means the amount of tolerance to avoid misclassifying each training example. A smaller C highlights the importance of β and a larger C reduces the importance of $\beta$.

Rewriting the equation (2.38) in the Dual problem by using the Lagrange multipliers method gives Equation (2.45) [81]:

$$max_a \sum_{n=1}^{m} \alpha_n - \frac{1}{2} \sum_{n=1}^{m} \sum_{p=1}^{m} \alpha_n \alpha_p y_n y_p x_n \cdot x_p \; ; \; s.t. \; 0 \leq \alpha_n \leq C, \forall n, \sum_{n=1}^{m} a_n y_n = 0 \quad (2.45)$$

### 2.10.1.3 Kernel Functions

If the dataset is not linearly separable in its own dimension space, it must be transformed to a higher dimension so a hyper-plane that separates the data linearly

could be discovered. But the transformation to higher dimensions is computationally expensive. Fortunately, in optimization problems only the result of the dot product $x_n \cdot x_p$ is important. The kernel function calculates the dot product and gives a result equivalent to the transformation of the data into higher dimensions.

Rewriting the dual problem by defining a kernel function $K(x_n, x_p) = x_n \cdot x_p$, gives Equation (2.46) [81]:

$$max_a \sum_{n=1}^{m} \alpha_n - \frac{1}{2}\sum_{n=1}^{m}\sum_{p=1}^{m} \alpha_n \alpha_p y_n y_p K(x_n \cdot x_p);\ s.t.\ \alpha_n \geq 0, \forall n, \sum_{n=1}^{m} a_n y_n = 0 \quad (2.46)$$

There are several types of kernel functions, for example, linear kernel: $K(x_n, x_p) = x_n \cdot x_p$, polynomial kernel: $K(x_n, x_p) = (x_n \cdot x_p + c)^d$, and Radial Base Function (RBF) kernel: $K(x_n, x_p) = exp(-\gamma \|x_n \cdot x_p\|^2)$.

### 2.10.2 Two Sample t-Test

The two-sample t-test is a statistical test used to compare the means of two different samples to determine if there is a significant difference between them. It is based on the assumption that the samples are drawn from populations with normal distributions. The test statistic is calculated as in Equation (2.47) [82]:

$$t = \frac{\bar{x} - \bar{y}}{\sqrt{\frac{s_x^2}{n_1} + \frac{s_y^2}{n_2}}} \quad (2.47)$$

Where:

$\bar{x}, \bar{y}$: are the sample means,

$s_x$, $s_y$: are the sample standard deviations,

$n_1$, $n_2$: are the sample sizes.

The probability level, also known as the p-value or significance level, is the probability that the test statistic will take a value at least as extreme as the observed value, assuming that the null hypothesis is true. If the p-value is less than the prescribed α, in this case 0.05, the null hypothesis is rejected in favour of

the alternative hypothesis. Otherwise, there is not sufficient evidence to reject the null hypothesis. The p-value is calculated from the t distribution table or by using MATLAB function “ttest2”.

## 2.11 Deep Learning Methods

Deep Learning is part of Machine Learning where the relevant features are learned automatically from data by the learning algorithm. The following subsections introduce the basic elements employed in Deep Learning.

### 2.11.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are very efficient computing elements that have the ability of learning; it has a parallel structure that gives them a robust nature and capability of dealing with very noisy data. Its structure is based on Biological Neural Networks (BNNs) and its operation can be considered as a simulation of BNNs operation. While BNNs are constructed of basic unit cells called Neurons which are cells consist of three parts Cell Body (Soma), Dendrites, and Axon [83]. The information between these Neurons is transferred through axons and dendrites during a chemical process at synaptic gaps. In ANNs the artificial counterpart of soma is Node, dendrites are Inputs, the axon is Output, and synaptic gaps are Weights.

Figure (2.10) shows the structure of a neuron that constructs a single layer and a single neuron ANN. The node function is to sum the values of weighted inputs and apply usually a non-linear activation function to introduce some non-linearity to the output, so its value will be bounded. The output of the node is given by Equation (2.48):

$$Output = \sigma(\sum_{i=1}^{n} I_i W_i) \tag{2.48}$$

where:

$\sigma$: is an activation function.

$n$: is the number of inputs.

$Ii$: is the i-th input.

$Wi$: is the weight value at i-th input.

An ANN usually consists of a large number of nodes that work together to solve a regression or classification problem. The key function that enables the ANN to learn is adjusting its weights during training. Training is a process of feeding ANN with examples that leads to adjusting weights.

Building an ANN by connecting the output of one neuron to the input of other neurons will produce a multilayer ANN; a Deep Neural Network is an ANN with more than 2 layers. Different models of ANN are developed in the last few decades; the most powerful models will be discussed in the following subsections.

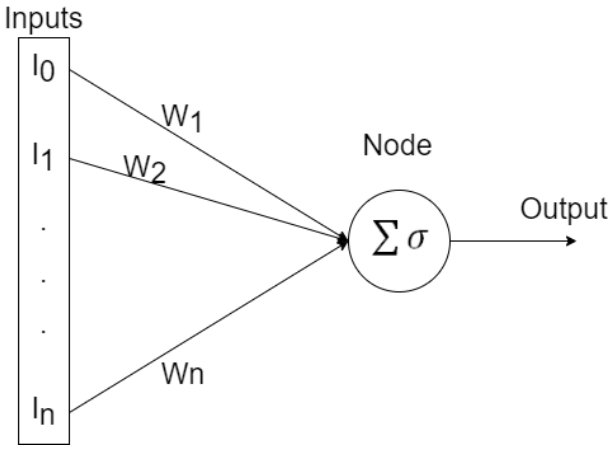



Figure 2.10 Basic Artificial Neural Network (ANN) node structure [84].

### 2.11.2 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) (Figure 2.11) are originally designed to feature extraction from images -since their structure is an interpretation of spatial information between pixels of an image- but they can be used for other types of data having some sort of spatial dimensions [85]. Deep CNN has a multi-layer and each layer usually has two sub-layers namely the Convolution layer, and Pooling layer, and a Rectified Linear Unit (ReLU).

In the convolution layer, a filter is applied to the inputs with convolution operation to extract with the best performance the spatial dependencies between the input elements.

The core task of the convolution layer is feature extraction. *A* and *B* are the number of convolution kernels that are used in each convolution layer to extract different forms of features from the EEG records. A two-dimensional convolution

kernel may be of dimensions (2$n$ +1) (2$m$ +1), where $n$ and $m$ are some integer numbers. The convolution operations are utilised on input data $X$ as in Equation (2.49):

$$Y(t,z) = \sum_{p=-A}^{A} \sum_{q=-B}^{B} W(p,q)X(t-p,z-q) \quad (2.49)$$

where $W(p, q)$ denotes the weight value of the kernel at time p and channel position $q$, $p$ varies from -$A$ to +$A$ and $q$ varies from -$B$ to +$B$, and $Y(t)$ is the data sample at a time ($t$) in the output data [24].

### 2.11.3 Pooling Layer

The pooling layer is located after each convolution layer to down-sample or reduce the size of the feature maps, making the output feature maps of convolution layers less sensitive to the positions of features of the entered data, so the computation time is also reduced. Many non-linear functions like max-pooling, average-pooling, and global-pooling can be used to implement the pooling layer [4].

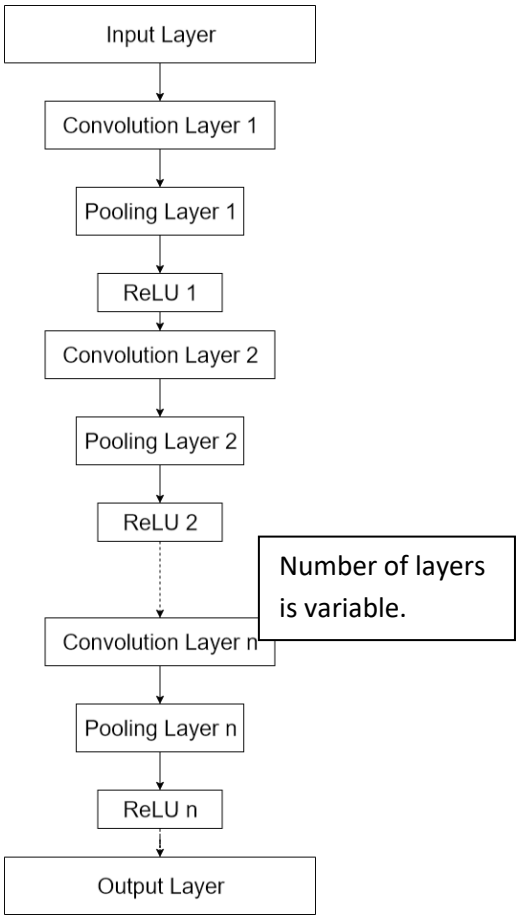



Figure 2.11 Convolutional Neural Network (CNN) architecture [4].

### 2.11.4 Error Backpropagation

In the ANN training process, the examples are fed into ANN with its desired output, the goal of ANN is to minimize the difference between the desired output and the ANN actual output. The value of error is propagated back to the layers of ANN so the weights are adjusted every time to achieve minimum error. Since the desired output is given with each training example during learning; this type of learning is called supervised learning.

Adjusting the weights is performed by multiplying the error by some value called the learning constant. This constant is usually less than, 1 and during backpropagation multiplication of this value by itself causes vanishing Gradient problem or Gradient descent.

### 2.11.5 Activation Function

The commonly used activation function in CNN after each convolution layer is Rectified Linear Unit (ReLU) due to its property of producing large gradients that will be useful at the training time. The function of ReLU is to generate a self-value for all positive inputs, and zero value for all negative inputs as in Equation (2.50), where *x* is the input variable for the function.

$$ReLU(x) = \begin{cases} x, & if\ x \geq 0 \\ 0, & if\ x < 0 \end{cases} \tag{2.50}$$

However, due to the explosion gradient problem; where EEG data causes gradient explosion, Leaky-ReLU is used as an activation function. Leaky-ReLU is similar to ReLU, but it has a small slope for negative values as in Equation (2.51):

$$ReLU(x) = \begin{cases} x, & if\ x \geq 0 \\ ax, & if\ x < 0\ where\ a < 1 \end{cases} \tag{2.51}$$

### 2.11.6 Recurrent Neural Networks

Recurrent neural networks (RNN) are good choices to deal with patterns in sequences of data, such as text, and speech. These algorithms take time and sequence into consideration, so they have a temporal dimension, meaning that

their output is not dependent on just the current example they see, but also on what they have perceived previously. Thus, RNNs have memory [86].

### 2.11.7 Long Short-Term Memory

Long Short-Term Memory (LSTM) (Figure 2.12) is an exceptional kind of RNN. It is capable of managing the vanishing gradient problem faced by the original RNN. It has the ability to learn the short-term and long-term dependencies for information over time. It consists of four gates that coordinate the learning process of the network. The input gate learns which information that should be stored in the memory, the forget gate learns the duration of time for that information, and the output gate is to learn when to read the stored information [87, 88].

### 2.11.8 Gated Recurrent Unit Layer

Gated Recurrent Unit (GRU) is another enhanced form of standard RNN. It uses two gates to solve the vanishing gradient problem; the update gate and the reset gate. The two gates will decide what information will be passed to the output so that they can be trained to retain information from long ago without losing it through time. The features of GRU and LSTM make them good candidates for solving time series problems like seizure prediction in EEG signals.

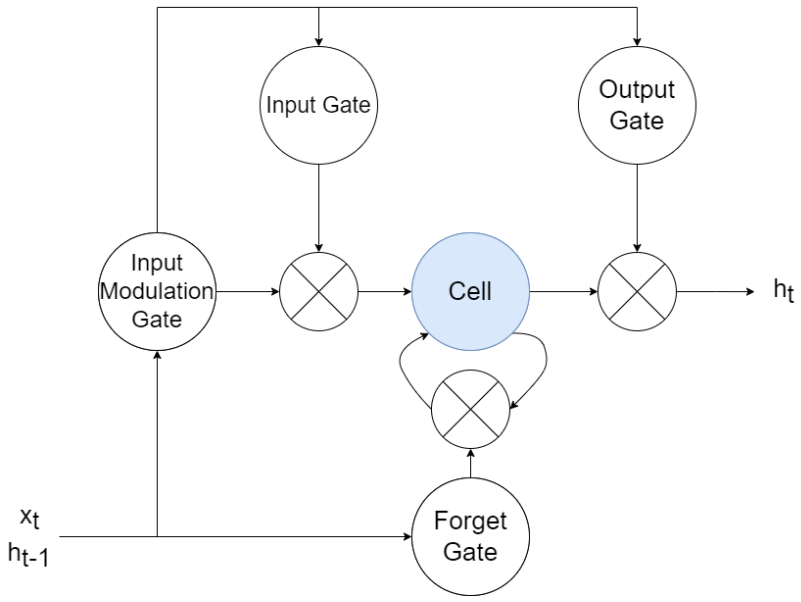



Figure 2.12 Long Short-Term Memory (LSTM) structure [88].

### 2.11.9 Fully Connected Network

In Fully Connected Network or Dense Network, each node in any layer is linked to all nodes in the successive layer (Figure 2.13). This type of network is used for classification and regression problems.

The extracted features must be classified since it's expected to predict the state of the input EEG data points. Thus, single or multiple fully connected layers are required to perform classification tasks [4]. One neuron is required in the last fully connected layer since the EEG classification problem is a binary problem; where output 0 belongs to the inter-ictal class, whereas output 1 belongs to the pre-ictal class. The Sigmoid function is used in the output layer, which works best for binary classification as in Equation (2.52):

$$\boldsymbol{\sigma(x) = \frac{1}{1+e^{-x}}} \tag{2.52}$$

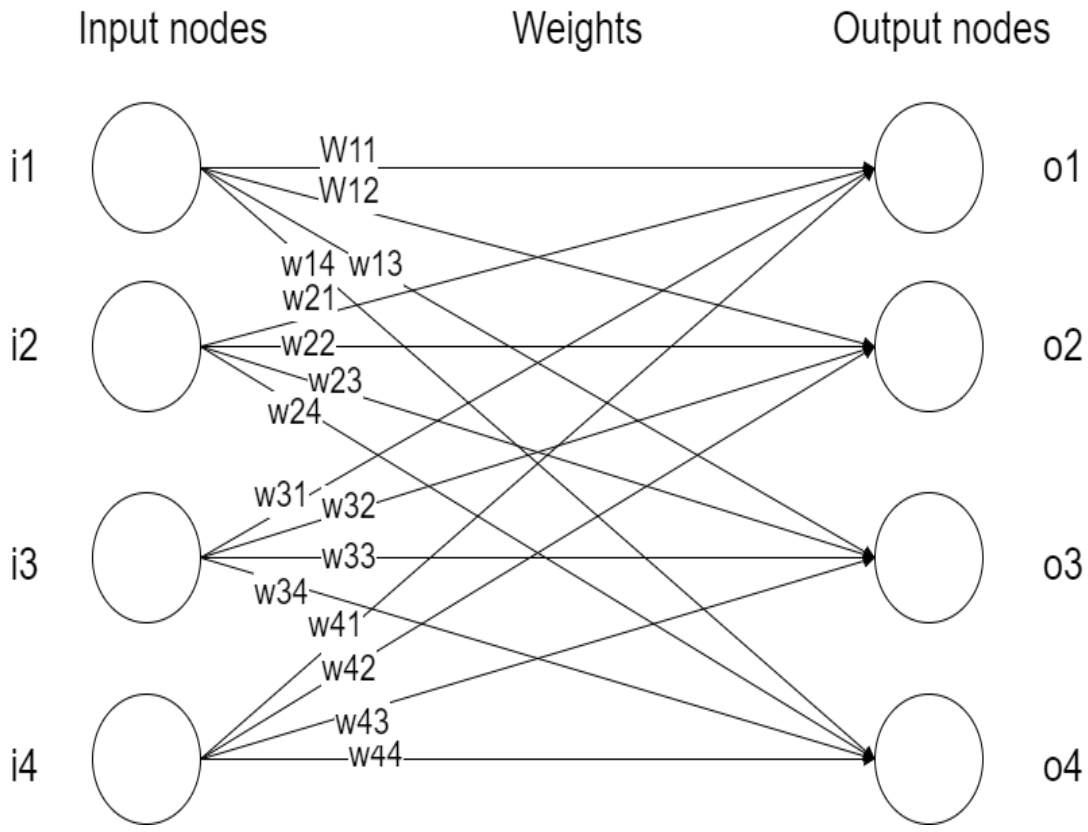



Figure 2.13 A fully connected layer structure [84].

### 2.11.10 Regularization and Dropout

The training data should be regularized to test the accuracy of ANN, many researchers and programmers regularize the training data into 70% training group, 20% validation group and 10% test group. If the accuracy is good for the training group and poor for the test group this means that the ANN is storing the patterns but not learning them; i.e., the generalization of the model is poor and this problem is called overfitting. To mitigate the overfitting problem Dropout method may be used. Dropout is the process of leaving some nodes without adjusting their weights during training.

### 2.11.11 Batch Normalization Layer

Training deep neural networks is a hard task; one reason for this is that the model is updated layer-by-layer backward from the output to the input using an approximation of error that assumes the weights in the former layers are unchanged. Because the weights of all layers are changed throughout the weights updating process; the update process is continually tracking a moving target. The batch normalization layer will coordinate the updating process of multiple layers in the model [89].

Batch Normalization will rescale (standardize) data to have a mean of zero and unit standard deviation per mini-batch which means standardizing the activations of each input variable. Standardizing the output of a layer means that the spread and distribution of inputs to its succeeding layer will not change the weight dramatically during the updating process. This will cause speeding up and stabilise the training phase of deep neural networks [90].

## 2.12 Performance Evaluation Metrics for Deep Learning Models

The effectiveness of the suggested models is evaluated by using classification accuracy, sensitivity, false-positive rate (FPR), and F1 score. The accuracy is the total number of true positive samples over the total number of samples as in Equation (2.53):

$$Accuracy = \frac{TP\ of\ preictal + TP\ of\ interictal}{Total\ number\ of\ preictal\ and\ interictal\ samples} \quad (2.53)$$

The most important metric for the model is sensitivity because the prediction of seizures (True Positive) is the main goal. The sensitivity (Recall) is the ratio between True Positive samples of the pre-ictal state and True Positive of the pre-ictal state plus False Positive of the inter-ictal state as in Equation (2.54):

$$Sensitivity = \frac{TP\ of\ preictal\ state}{TP\ of\ preictal + FP\ of\ interical} \quad (2.54)$$

FPR is defined as the ratio between False Positive of pre-ictal state and True Positive of inter-ictal state with False Positive of the pre-ictal state as in Equation (2.55):

$$FPR = \frac{FP\ of\ preictal\ state}{TP\ of\ interictal\ state + FP\ of\ preictal\ state} \quad (2.55)$$

The F1 score is also used as an evaluation metric which is a weighted harmonic mean of sensitivity and accuracy, such that 1.0 is the top score and 0.0 is the bottom score. Hence, F1 scores are lower than accuracy measures since they use sensitivity and accuracy in the computation. However, the biased average of F1 should be used to compare classifier models, not accuracy. F1 Score is calculated as in Equation (2.56):

$$F1\ Score = \frac{2 \times Sensitivity.Accuracy}{Sensitivity + Accuracy} \quad (2.56)$$

One of the greatest valuable evaluation metrics for examining the classification model's effectiveness is the Area under the curve (AUC) of the Receiver Operating Characteristics (ROC) curve. ROC is a likelihood curve and AUC characterizes the degree or value of separability. It quantifies the ability of the model to distinguish different classes. The higher the AUC means that the model is better in distinguishing pre-ictal and inter-ictal segments of EEG signals.

# Chapter Three

# The Proposed Models

This chapter introduces the proposed models and algorithms for artefact rejection, and seizure detection and prediction.

## 3.1 EEG Signals Artefact Rejection

EEG signals are contaminated with artefacts like eye movement and muscle movement artefacts. Fine-tuning of these signals and automatic rejection of artefacts prior to feature extraction is straightforward. In this section, a novel method for artefact cancellation based on signal statistics with modification of independent sources extracted by Independent Component Analysis (ICA) of EEG signals is suggested. Visual inspection of the reconstructed signals shows the validity of the proposed method in artefact rejection. Moreover, this method did not require any extra information channel attached to EEG signals.

### 3.1.1 Artefact Rejection Algorithm

The proposed algorithm aims to fine-tune EEG signals. The steps of the algorithm can be summarized as:

1- Using average montage for EEG and signal filtering.
2- Applying ICA to extract the independent sources.
3- Modifying independent sources by Artefact identification and cancellation process.
4- Restoring back EEG signals from ICA components.

The block diagram of this tuning process is explained in Figure 3.1.

### 3.1.2 Using Average Reference Montage

For the system to be generally standardized, converting the EEG recording montage to an average reference montage is useful since it rejects any common noise among the channels. Average reference montage is obtained by averaging the sum of all EEG channels. This type of montage will remove the common noise between

all channels. Therefore, it reduces the dimensionality of the signals to some degree of freedom which is the main goal of this paradigm, since the common noise is considered another variable in the system, removing the common noise means removing a variable and thus reducing the dimensionality of the system.

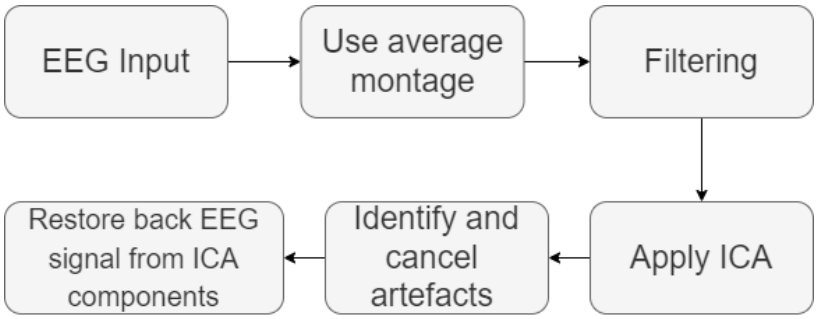



Figure 3.1 The proposed artefact rejection algorithm block diagram

### 3.1.3 EEG Signal Filtering

Filtering is the most famous technique for removing unimportant information and reducing signal dimensionality. The actual EEG signal frequency range is 0.5-100Hz but the important frequency range depends on the required application, for example, the recommended frequency range for seizure detection is 1-70Hz [91]. Therefore, a low pass filter with a corner frequency of 70Hz, and a high pass filter with a corner frequency of 1 Hz are used for filtering EEG signals. Furthermore, a narrow band notch filter with a frequency bandwidth 50 or 60Hz is used to remove the power line artefact.

### 3.1.4 Applying ICA to EEG signals

The next step after EEG signals frequency domain filtering is calculating their independent components. ICA is used to estimate the independent linear sources of EEG signals. Source estimation is done by whitening each channel so that the channels data *x* has zero mean and unit variance, as in Eq. (3.1):

$$\tilde{\boldsymbol{x}} = \frac{\boldsymbol{x}}{\boldsymbol{std}(\boldsymbol{x})} \tag{3.1}$$

The Principal Components (PC) of channels' data are computed by Principal component analysis and are used to estimate de-mixing matrix W. Consequently, the projection of these components with channels' data gives Principal Components Data (PCD) as in Eq. (3.2):

$$PCD = PC.\tilde{x} \tag{3.2}$$

The multiplication of de-mixing matrix W with principal components PCD, gives the linear sources of the channels s as shown in Eq. (3.3):

$$s = W \times PCD \tag{3.3}$$

### 3.1.5 Artefact Identification and Cancelation

The artefact identification is done over ICA sources. The noise part is obvious from the signal of interest. Two methods of artefact identification are introduced; the Manual mode and the Automatic mode. In Manual method, the user is requested to identify which signal part of the sources is exemplified noise. The marked portions of the sources are stored as templates in a file. Figure 3.2 shows examples of selected noise portions from independent sources.

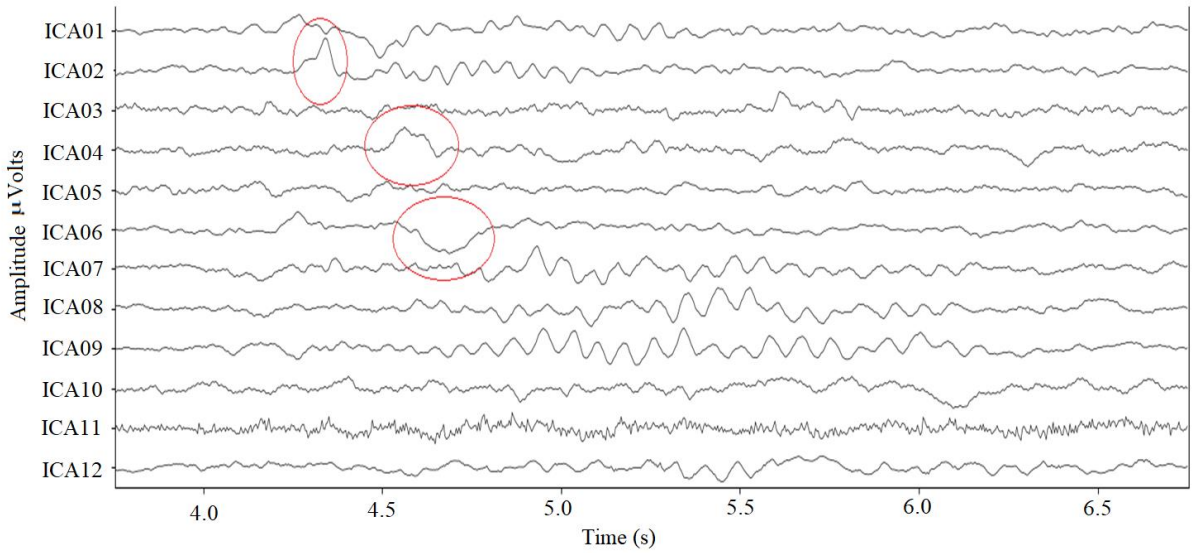



Figure 3.2 Examples of noise signals selected by Manual method

In Automatic method, the identification process of noise is carried out by sliding the noise templates for each source over the source data and taking the Euclidean

distance (which is the length of a line segment between the two points) between them. A list of minimum Euclidean distances is composed. Consequently, a threshold of 0.65 selected by experiments for the distance mean, so any segment over the threshold is identified as a noise signal. Figure 3.3 shows the samples of detected noise signals in Automatic mode.

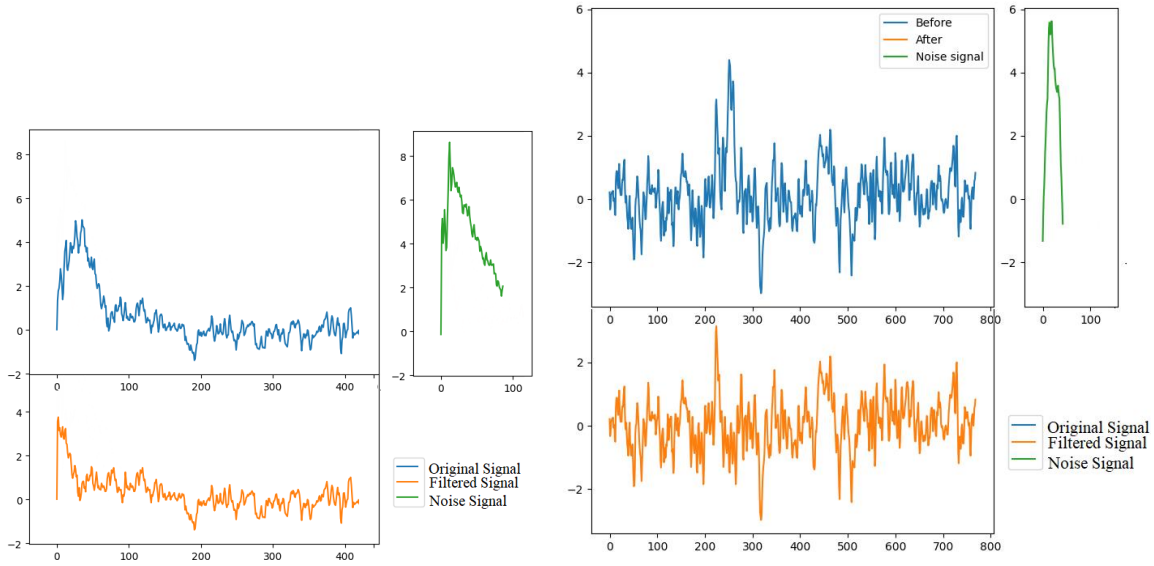



Figure 3.3 Samples of noise signals detected by Automatic method

In both modes, the noise cancellation takes the same procedure. After noise identification, a part of the source signal of noise portion length i.e., the number of data points of the noise portion is replaced by an estimated signal based on the statistical analysis of the source signal. The list of sub-signals of the length noise signal is generated from source data and the Probability Density Function (PDF) of these sub-signals is calculated. The most frequent signal represents an estimation that interchanges the noise signal. Figure 3.4 shows the block diagram of the procedure of the noise identification and cancellation process which can be summarized as:

1- Input *source s*, *noise template t*, and *a threshold value*.
2- Slide *the noise template* over the *source* to make a list of *distances*. $d_i = \{t - si\}$ *for* $i=1,\ldots n$

3- Make a list of *minima* and *minima locations* of *distances*. $minmas = min\{di\}, minma_{loc} = argmin\{di\}$
4- Search for the best minima candidates according to a threshold value that gives *noise locations*. $noise_{loc} = \{minma: minmas < thresold\}$
5- Generate sub-signals with data points in between *noise locations*. $ssi = \{si\}\ for\ noiseloc1 < i < noiseloc2$
6- Find the mode of each sub-signal. $modes = \{mode(ssi)\}$
7- Estimate the substitution of the noise portion by the most frequent sub-signal. $substitute = argmax(modes)$

Since the algorithm depends on statistical features of ICA components, this proposed method did not require any extra information channel attached to EEG signals.

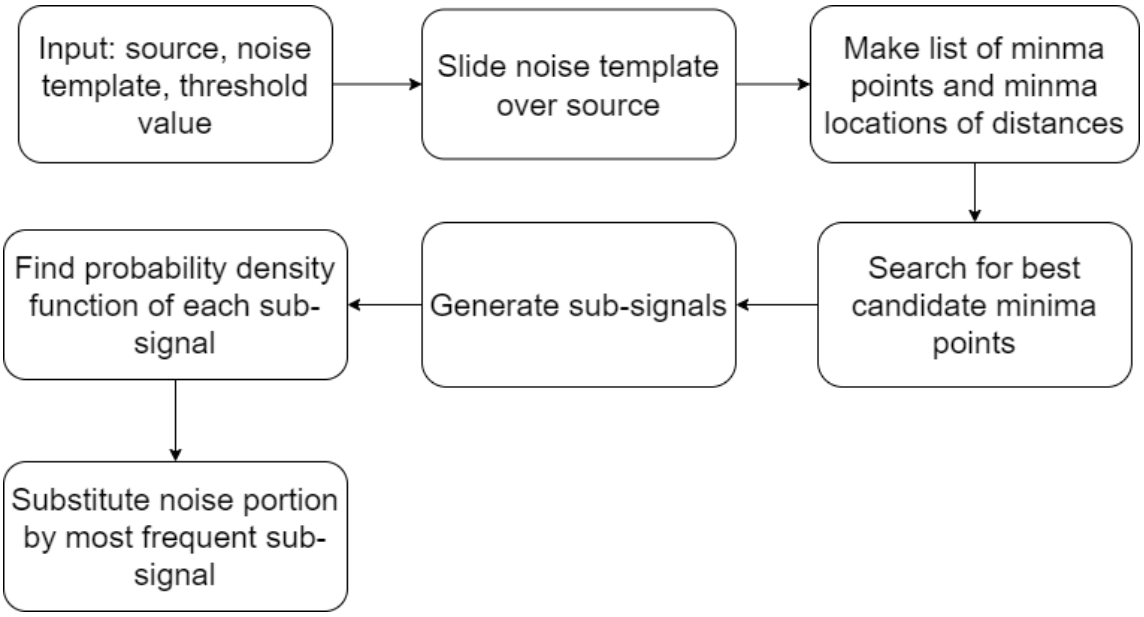



Figure 3.4 Noise identification and cancelation algorithm block diagram

### 3.1.6 Restoring EEG Signal from ICA Components

After identification and cancellation of artefacts in either Manual or Automatic mode, the EEG signal data are restored back from the modified components. The restoration process is done as the inverse of the decomposition process. Hence, the

PCD is calculated as a projection of the inverse de-mixing matrix with modified sources as in Eq. (3.4):

$$PCD = W^{-1} \times s \tag{3.4}$$

Consequently, the channel data are reconstructed with the multiplication of PCD by the inverse of principal components as in Eq. (3.5):

$$\tilde{x} = PC^{-1}.PCD \tag{3.5}$$

This gives the whitened data (data with zero mean and unit variance). So, the next step is to colour $s$ (coloured data have non-unit variance) to produce channels data as in Eq. (3.6):

$$x = \tilde{x}.std(s) \tag{3.6}$$

where std(.) is the standard deviation.

## 3.2 Seizure Prediction Based on Machine Learning

### 3.2.1 Feature Extraction

The extraction of features from EEG signals is done by converting the data of each EEG channel into segments of prescribed length. The MATLAB software and its associated toolboxes are used to extract 63 features from EEG signals, these features are listed in Table 3.1 and each feature given a unique ID so it will be used in optimum features searching process.

The Maximum Lyapunov Exponent calculated for EEG segments using the MATLAB function “lyapunoveExponent” with sampling frequency = 256 Hz, lag time = 10, and embedding dimension =12.

The Approximate Entropy calculated for EEG segments using the MATLAB function “approximateEntropy” with lag time = 10, and embedding dimension =12.

The Short Time Fourier Transform calculated by MATLAB function “stft” with sampling frequency= 256 Hz, overlap length= 10 points.

Finally, the Empirical Mode Decomposition features are extracted by MATLAB function “emd”.

### 3.2.2 The Proposed Seizure Perdition Algorithm

In this section, an algorithm for seizure perdition based on the simulated annealing method and machine learning is proposed. The algorithm works in four rounds to analyse EEG data and train an SVM model to recognize pre-ictal segments from inter-ictal ones.

The proposed algorithm is a multi-round algorithm, at the first round, the channel selection round, the EEG channel that has the most relevant features is selected, so the channel or EEG electrode that is placed near the source area of epilepsy in the brain is selected. Figure 3.5 shows the flowchart of the first round. The round starts by the reading of the recorded EEG signals to convert them into segments of length *SEG=10* seconds, the *PIL* is assumed to be 10 minutes. Then several features (listed in Table 3.1, each feature has a unique ID number) are extracted from pre-ictal and inter-ictal periods. The statistical t-test was applied to features data, the ratio of discriminating features that have a p-value < 0.05 to the total number of features stored for each EEG channel. The channel that has the maximum ratio will be the output of the first round.

The second round is dedicated to optimum *PIL* and *SEG* calculation, based on the fact that the EEG signals are patient specific; the *PIL* also may be patient specific. However, in this round, only EEG data of the selected channel from the first round is used for further computations. Figure 3.6 shows the flowchart of the second round. The round starts to iterate for values of *PIL* (10-30) minutes and *SEG* (10-30) seconds, and for each value, the features were extracted and t-tests were applied. The ratios of discriminating features that have a p-value < 0.05 to the total number of features are stored for each *PIL-SEG* pair. Finally, the *PIL-SEG* pair that gives the maximum ratio is chosen.

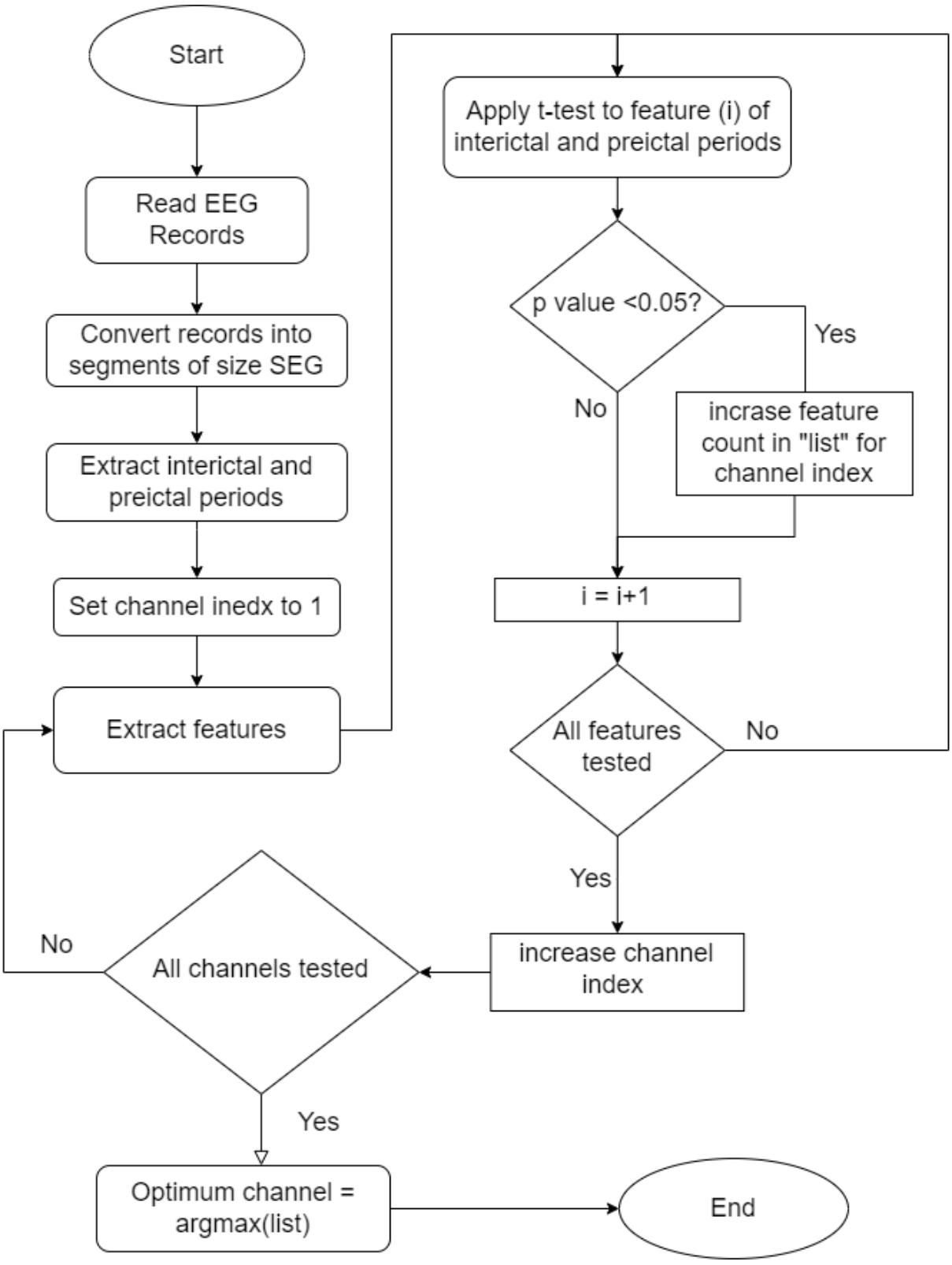



Figure 3.5 The flowchart for the first round

Table 3.1 The extracted feature set.

| Feature ID | Feature Name | Feature ID | Feature Name |
|---|---|---|---|
| 1 | Lyapunov Exponent | 33 | Mean of power spectrum in Alpha band |
| 2 | Approximate Entropy | 34 | Mean Envelope Energy (1) |
| 3 | Skewness | 35 | Mean Envelope Energy (2) |
| 4 | Standard deviation | 36 | Mean Envelope Energy (3) |
| 5 | Kurtosis | 37 | Mean Envelope Energy (4) |
| 6 | Correlation Dimension | 38 | Mean Envelope Energy (5) |
| 7 | Number Extrema (1) | 39 | Mean Envelope Energy (6) |
| 8 | Number Extrema (2) | 40 | - |
| 9 | Number Extrema (3) | 42 | Mean of power spectrum in Beta band |
| 10 | Number Extrema (4) | 43 | Relative Tolerance (1) |
| 11 | Number Extrema (5) | 44 | Relative Tolerance (2) |
| 12 | Number Extrema (6) | 45 | Relative Tolerance (3) |
| 13 | Mean of periodogram | 46 | Relative Tolerance (4) |
| 14 | Mean of Cross Correlation | 47 | Relative Tolerance (5) |
| 15 | Mean of power spectrum in Delta band | 48 | Relative Tolerance (6) |
| 16 | Number Zero crossing (1) | 51 | Mean of power spectrum in Gama band |
| 17 | Number Zero crossing (2) | 52 | Mean of residual signal |
| 18 | Number Zero crossing (3) | 53 | Standard deviation of residual signal |
| 19 | Number Zero crossing (4) | 54 | Mean of IMF1 |
| 20 | Number Zero crossing (5) | 55 | Standard deviation of IMF1 |
| 21 | Number Zero crossing (6) | 56 | Mean of IMF2 |
| 22 | - | 57 | Standard deviation of IMF2 |
| 23 | - | 58 | Mean of IMF3 |
| 24 | Mean of power spectrum in Theta band | 59 | Standard deviation of IMF3 |
| 25 | Number Sifting (1) | 60 | Mean of IMF4 |
| 26 | Number Sifting (2) | 61 | Standard deviation of IMF4 |
| 27 | Number Sifting (3) | 62 | Mean of IMF5 |
| 28 | Number Sifting (4) | 63 | Standard deviation of IMF5 |
| 29 | Number Sifting (5) | 64 | Mean of IMF6 |
| 30 | Number Sifting (6) | 65 | Standard deviation of IMF6 |

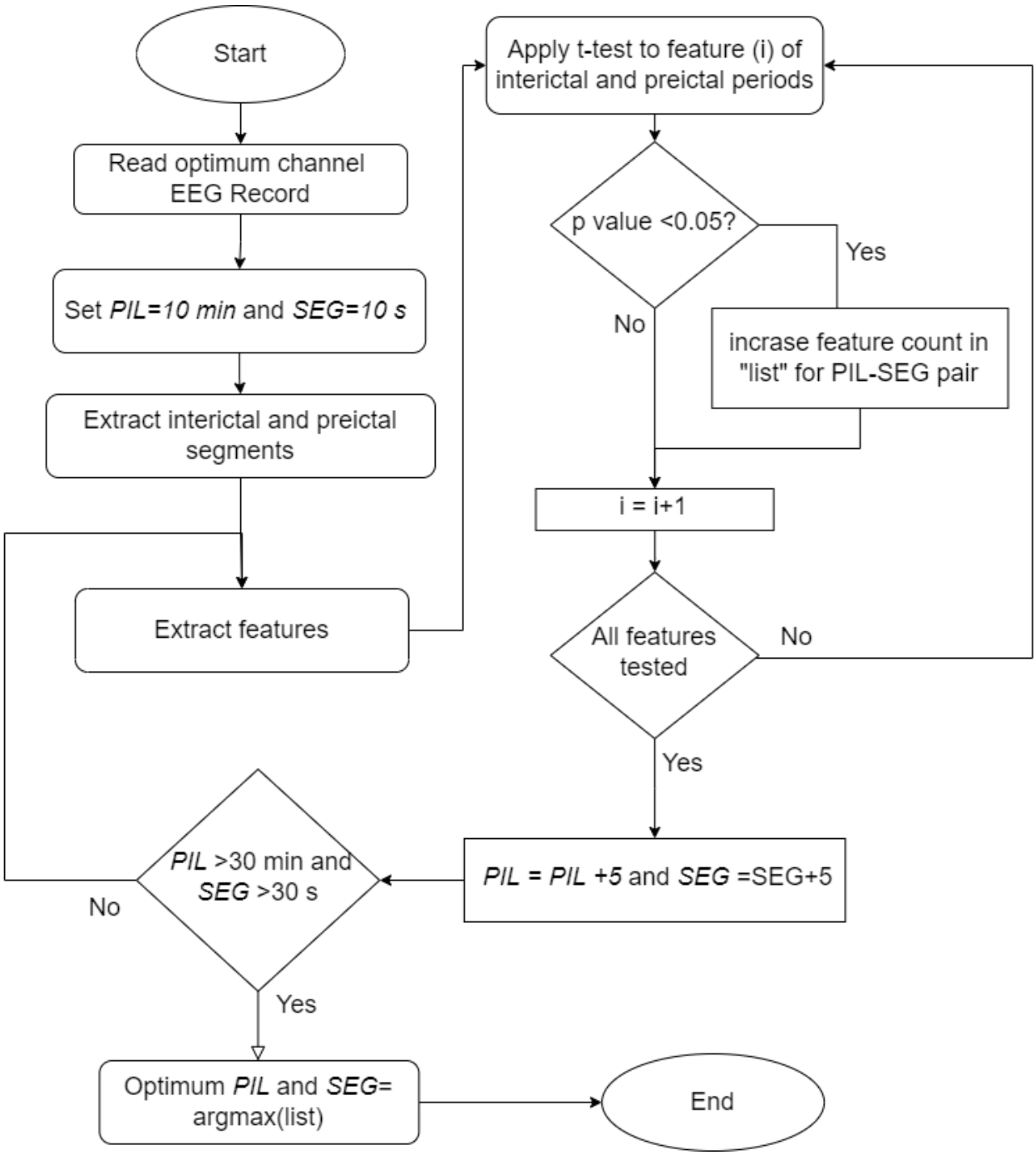



Figure 3.6 The flowchart of the second round.

In the third round (Figure 3.7) the optimum channel and optimum *PIL* and *SEG* were used to form inter-ictal and pre-ictal data sets, and then extract features from EEG data to form the solution space of the simulated annealing (SA) algorithm. In the fourth round (Figure 3.7) an initial subset $x_0$ of 10-features is selected randomly from the solution space. SA algorithm is an optimization

algorithm that finds the optimum solution in an iterative manner. Table 3.2 shows the chosen parameters for the SA algorithm. In each iteration a new solution is chosen randomly from solution space, the available data are split into 70% training set and 30% testing set, and then an SVM model was trained and tested for the given solution. To make sure that the trained model with the current solution is neither over-fitted nor under-fitted both resubstitution loss – which is the difference between predicted labels and actual labels for the train set- and test loss – which is the difference between predicted labels and actual labels for test set- are used in cost function calculation. The cost function will be given as in Equation (3.7):

$$cost\ function = \frac{resubstitution\ loss + test\ loss}{2} \tag{3.7}$$

The cost function of the current solution will be compared with the cost function of its preceding one, if there is any improvement the solution is accepted, else it will be accepted according to the Metropolis rule where the probability of accepting the solution is decreased with temperature decrease. Finally, if the number of iterations completed for the current temperature, the temperature will be decreased according to the cooling schedule and a new solution will be generated. The code for implementing the algorithm is presented in Appendix A.

Table 3.2 Parameters for SA algorithm.

| | |
|---|---|
| Minimum temperature | 1E-8 |
| Annealing schedule or cooling function | 0.95×T |
| Maximum consequence rejections | 1000 |
| Maximum number of successes | 20 |
| Boltzmann constant | 1 |
| Maximum number of iterations | 300 |

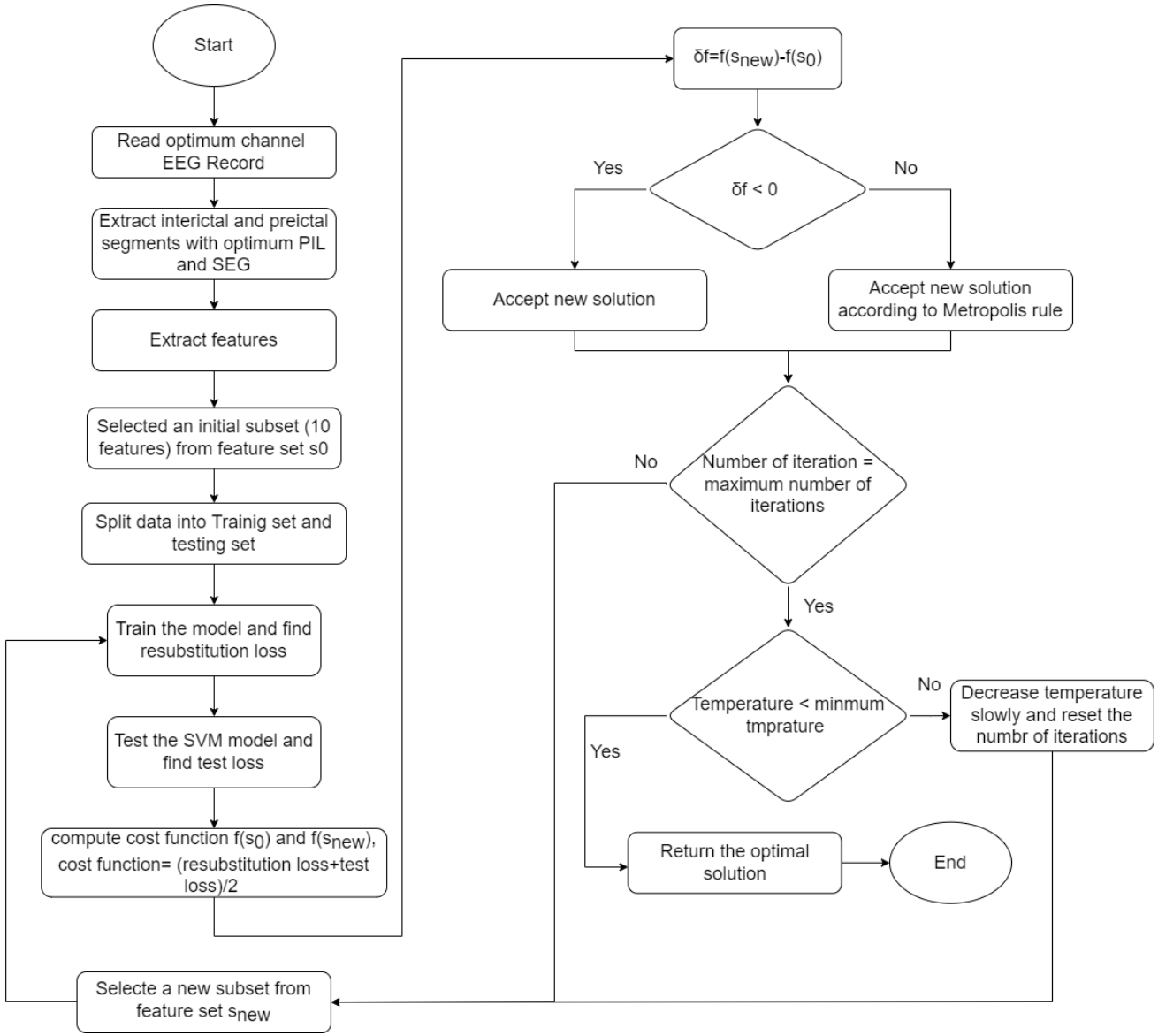



Figure 3.7 The flowchart of the third and fourth rounds.

## 3.3 Seizure Detection and Prediction based on Deep Learning

Epilepsy is one of the most prevalent central nervous system diseases. It is characterized by tantrums between different periods of time and may endanger the patient's life if he is in a state of work or in a critical place or situation. Anticipating the coming of seizures is important in protecting the patient's life, as he will take some measures or take medications to reduce the risks or prevent the seizure from occurring. In this section, one Deep Learning model for seizure detection and two models for seizure prediction are suggested and evaluated as clarified in the incoming sections.

### 3.3.1 Seizure Detection Model

The suggested seizure detection model is based on RNN-CNN utilizing three EEG channels. The accuracy of classification of seizure/non-seizure segments reached 99% with only three epochs of training.

#### 3.3.1.1 Data Preparation

The first process in every deep learning problem is data preparation for the learning process. The learning data are time series data generated from EEG channels of the CHB-MIT Scalp Database. The database is a set of EEG signal recordings available without restrictions for research purposes. The EEG signal data were gathered from subjects and separated into non-seizure and seizure EEG recordings with a sampling frequency of 256 Hz.

The learning data is generated by sliding a window of length 1 sec (number of time steps= sampling frequency= 256) over channels data with step size = 1; in this way, a huge number of batches will be produced with size (Batch size, time steps, number of channels). Figure 3.8 describes an example of time series generation from channel data.

Supervised learning is used in this model, so the samples are labelled. The labelling process is done by means of seizure start time and seizure end time. So,

each sample in the time series that lies between seizure start time and seizure end time belongs to class 1 otherwise it belongs to class 0.

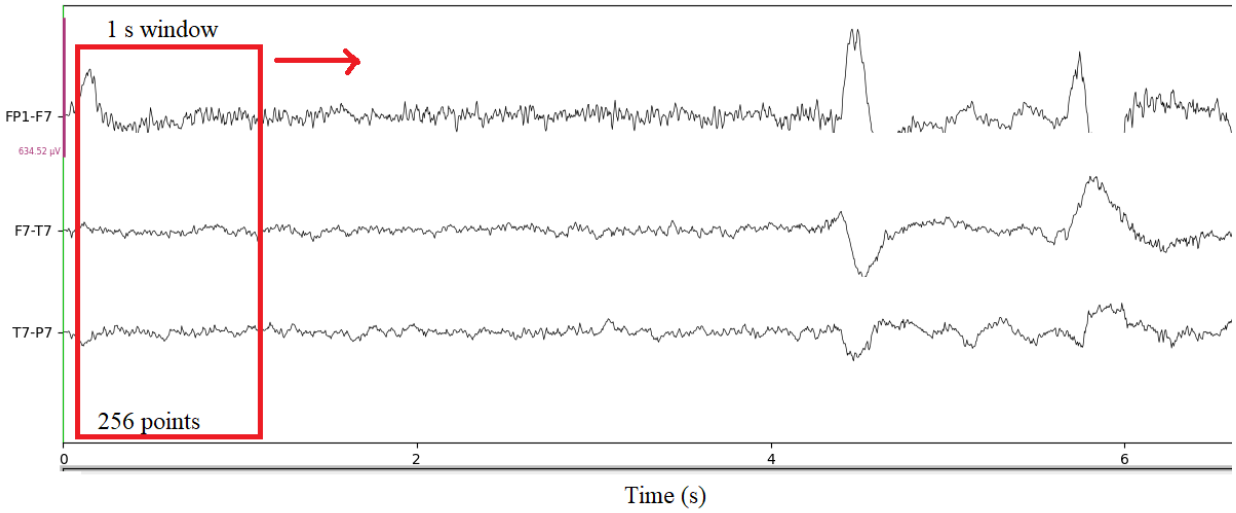



Figure 3.8 Time series generation and data preparation for seizure detection model

### 3.3.1.2 The Proposed Detection Model Architecture

There are no common rules for choosing the appropriate model deep learning model or the model that will work best for any given problem, so one can propose a model and test its performance for given data set and try to improve it. The proposed architecture is shown in Figure 3.9. Three of the 24 channels are chosen as the input signal to the model so the input array shape of the model will be (Batch size, 256, 3).

The first layer of the model is an LSTM layer for each channel that takes the input and produces features of shape (Batch size, 256, 32) that will be an input for the second LSTM layer. The output of these LSTM layers is normalized and stacked to form the input for the CNN layer. The CNN layer will learn the spatial dependencies between channel features and produces an output that will be the input of a dense layer for classification purpose. The combined RNN-CNN architecture of the proposed system gives the ability to the model for learning both time and spatial dependencies feature in an effective fashion.

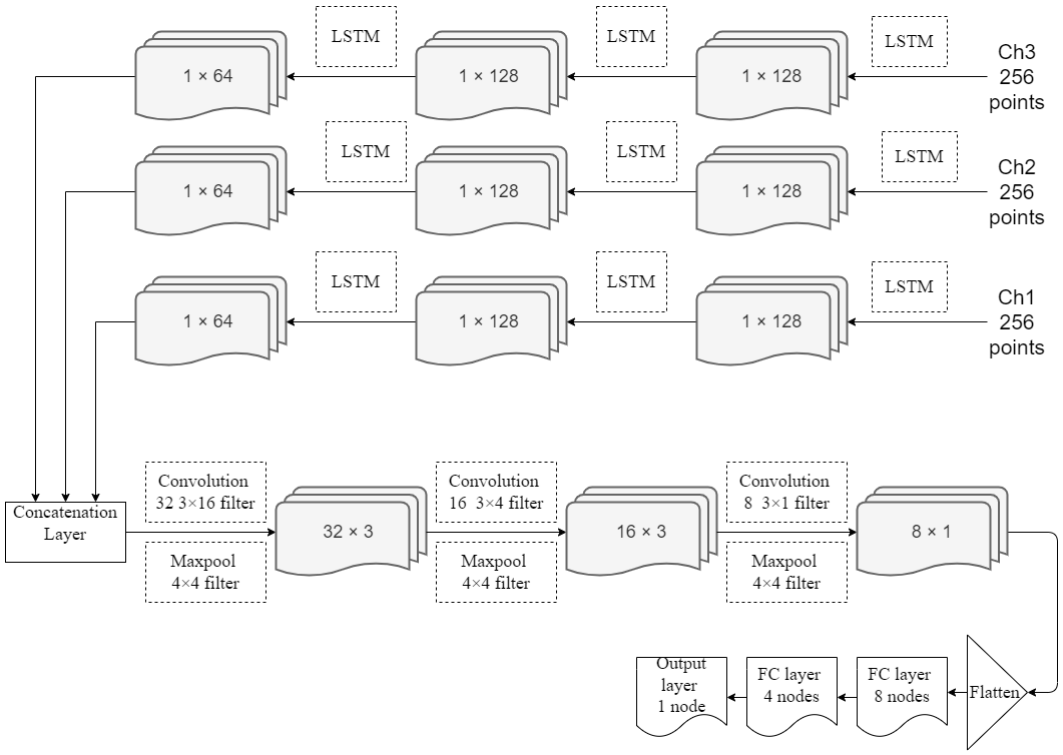



Figure 3.9 The proposed seizure detection model architecture

### 3.3.2 Seizure Prediction Models

In this section, two Deep Learning models for seizure prediction are suggested. The former model is established on CNN and the succeeding model is established on GRU-LSTM. The models were used in order to distinguish between the pre-ictal state and the inter-ictal state and then predict the onset of seizures. The models take into account the patient's comfort in using the device by utilizing 5 electrodes of EEG signals and reducing the alert period to only 10 minutes before the onset of the seizure.

#### 3.3.2.1 Data Preparation

The CHB-MIT dataset has been used as a training and testing dataset. In this dataset, the EEG signal recordings during a natural state of an epilepsy patient are labelled as *class 0*. The label *class 1* has been assigned to a recording that has a seizure event.

PIL has been chosen to be 10 minutes and segment size to be 15 sec, so, any 15-sec segment of data before seizure onset is labelled as pre-ictal. The inter-ictal

data is chosen to be at least 40 minutes before seizure onset as clarified in Figure 3.10. Five channels are chosen to be input for the model, so the input shape of the deep learning model will be (batch size, 5, 3840).

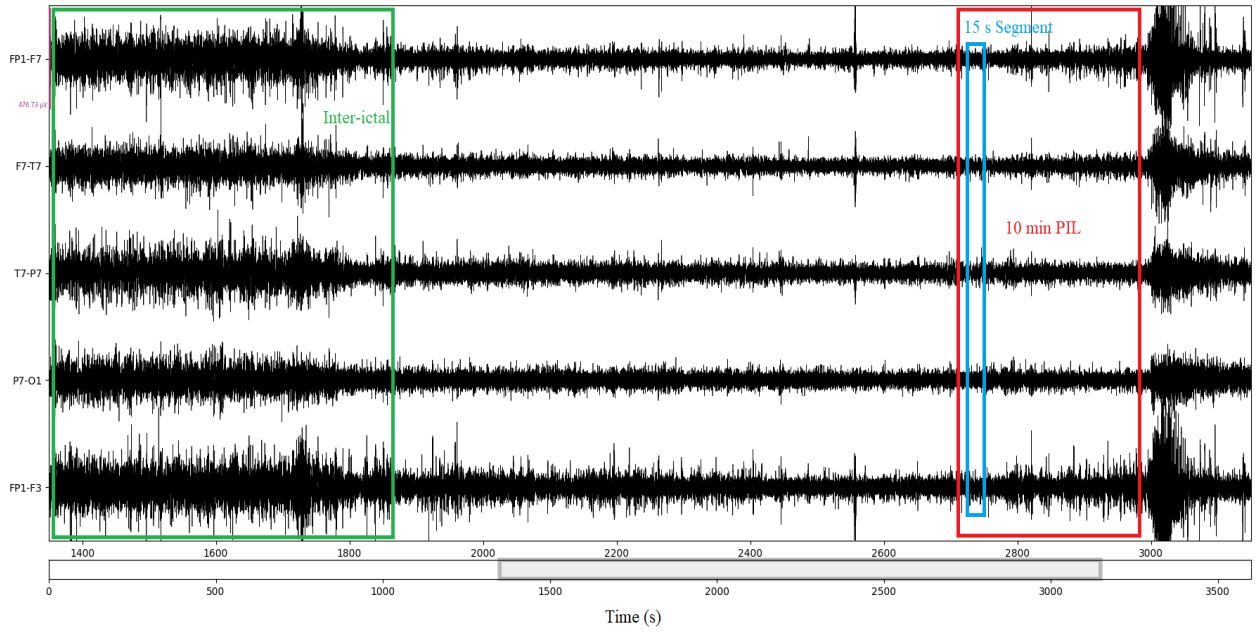



Figure 3.10 Data preparation for seizure perdition models

### 3.3.2.2 The Proposed Prediction Models Architecture

In this section two deep learning models are proposed; the first model is based on CNN and the second model is based on GRU-LSTM. The first proposed prediction model is composed of five convolution layers, each layer followed by an activation function and pooling layer. The classification task is performed at the end of the model by three fully connected layers as is clear from Figure 3.11. The second proposed prediction model is composed of six stacked GRU-LSTM layers, followed by three fully connected layers as clear from Figure 3.12.

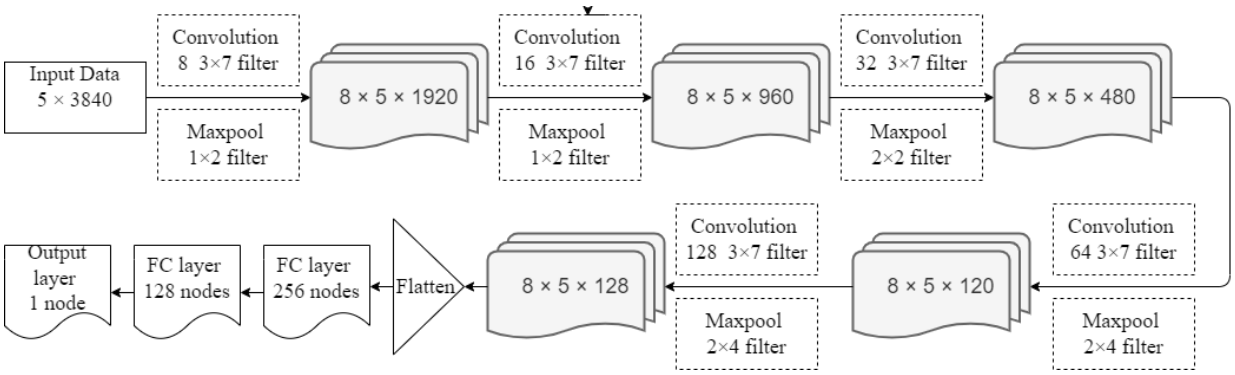



Figure 3.11 The architecture of proposed CNN model

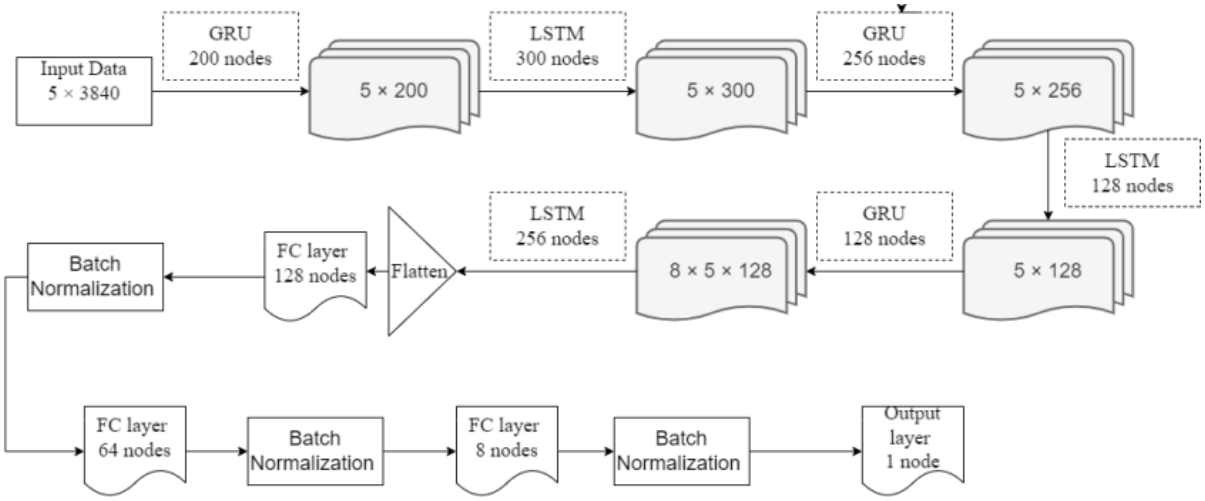



Figure 3.12 The architecture of proposed GRU-LSTM model

## 3.4 The Proposed Real-time Seizure Prediction Algorithm

In this section, a real-time algorithm that analyses EEG signals to predict seizures is proposed. The core of the algorithm detects rapid frequency changes in the EEG channels as a sign of incoming seizures. The relevant EEG channel is selected as in section 3.5 and segmented. The segmentation and processing are carried out at the same time.

The length of the segments is chosen to be 10 sec, processed by a high pass filter with a corner frequency of 0.5 Hz, and a low pass filter with a corner frequency of 120 Hz. Each filter is of type minimum-order having stopband attenuation of 60 dB.

Following the filtering process, the changes in the signal are calculated by taking the difference between two consecutive segments. This will extract any amplitude or frequency changes in the signal, these changes are have belonged to slight seizures where it could be signs of incoming seizures.

In order to maintain values stabilities, further processing will be applied for the segment. The segment will be normalized to have unit variance and zero mean, the values will be shifted up to get rid of negative values and complex numbers in logarithms and the logarithm of each value is taken so the changes will alter the results in a logarithmic way, and the outliers will be compensated with the linear filling method. A sample EEG signal with its extracted changes is shown in Figure 3.13.

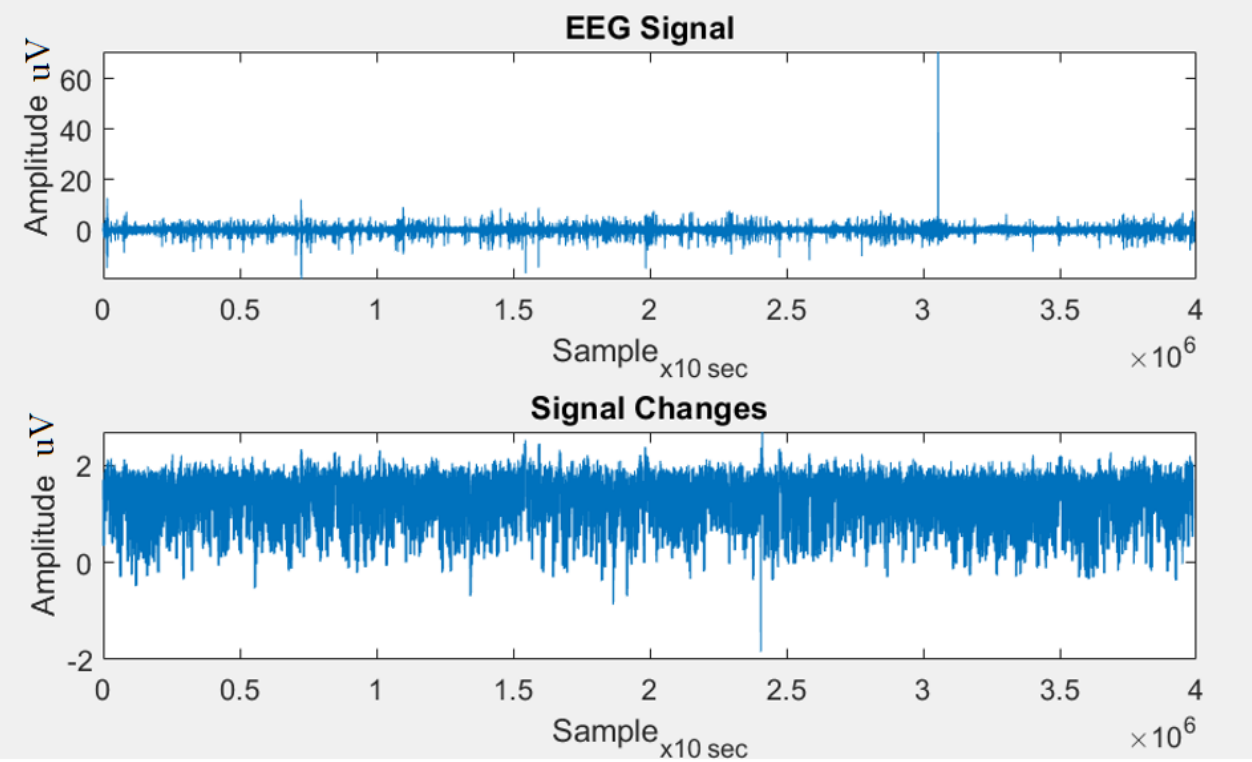



Figure 3.13 Sample EEG signal with its extracted values changes

The segment is transformed into the frequency domain by periodogram power spectral density (PSD) estimate. The periodogram is the Fourier transform of the autocorrelation sequence and it can be calculated as in Equation (3.8):

$$S_p(f) = \frac{1}{N}\left|\sum_{n=0}^{N-1} x(n)e^{-j2\pi fn}\right|^2 \quad (3.8)$$

The PSD was divided into eight frequency bands, and the changes in power values were extracted in each band separately by comparing the current power values with the previous power values. An alarm is raised if sharp changes in the power values are detected in each frequency band, and the alarm values are sent to the decision-making unit.

The decision-making unit receives the values of alarms from each frequency band and learns from the data the values of the parameters, the number of effective bands and the maximum time among band alarms, which are important to correctly predict seizures and shrink the number of false alarms. The values of parameters are specific to each patient.

It is possible to make a neural network to learn the values of the parameters, but in this work, the optimal values were reached by using repeated observations and adjusting the values for each patient in the Siena scalp EEG database, the block diagram of the proposed algorithm is shown in Figure 3.14 and the steps of the algorithm could be summarised as follows:

1- Read 10 sec segment from EEG signal *seg*.
2- Apply low and high pass filters to the segment.
3- Calculate the difference between the current and previous value of the segment.
4- Normalize the segment.
5- Take the logarithm of the segment and fill outliers.
6- Calculate the periodogram of the segment.
7- Split the periodogram into eight bands.
8- Find rapid frequency changes in each band and make pre-alarms.
9- Train a decision-making unit to predict seizures and make an alarm.

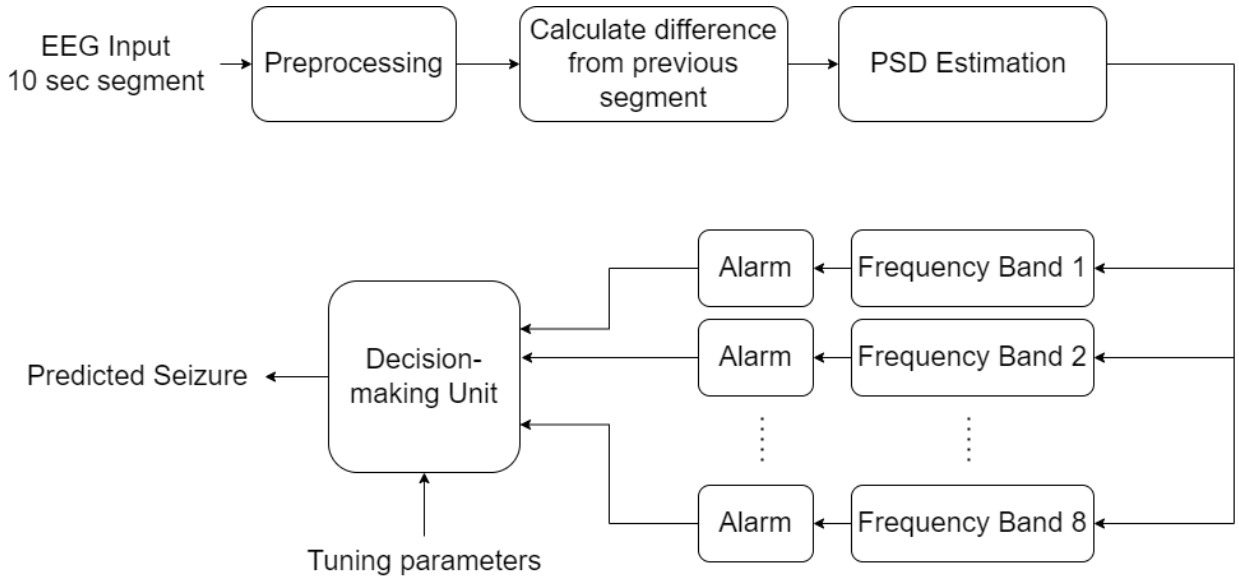



Figure 3.14 The block diagram of the proposed real-time seizure prediction algorithm

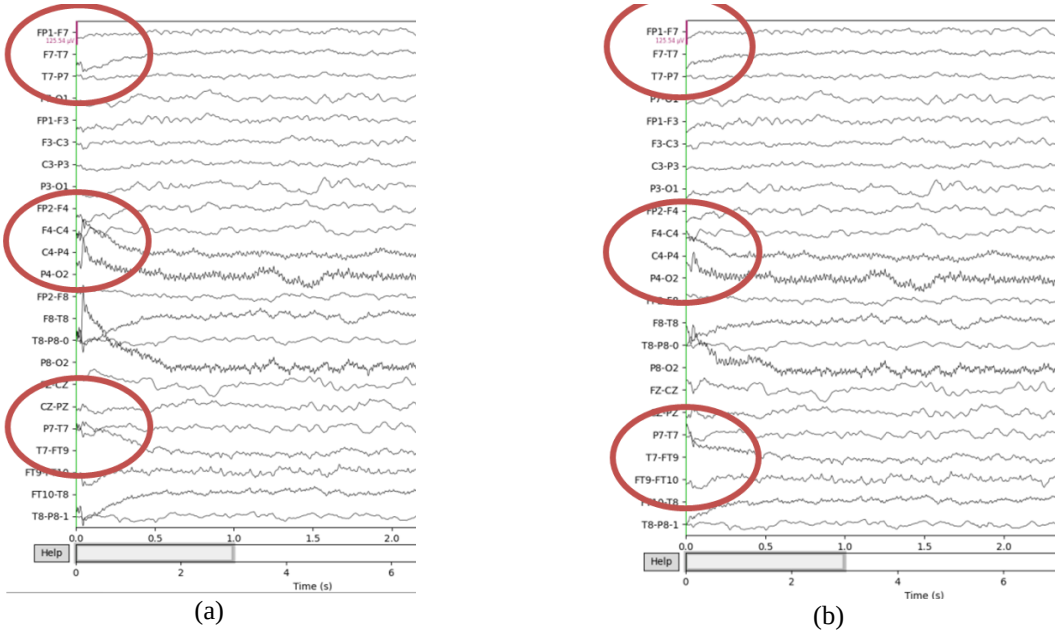
# Chapter Four
# Simulation Results of the Proposed Models

## 4.1 Introduction

In this chapter, the results obtained from the methods and models introduced in chapter three are presented.

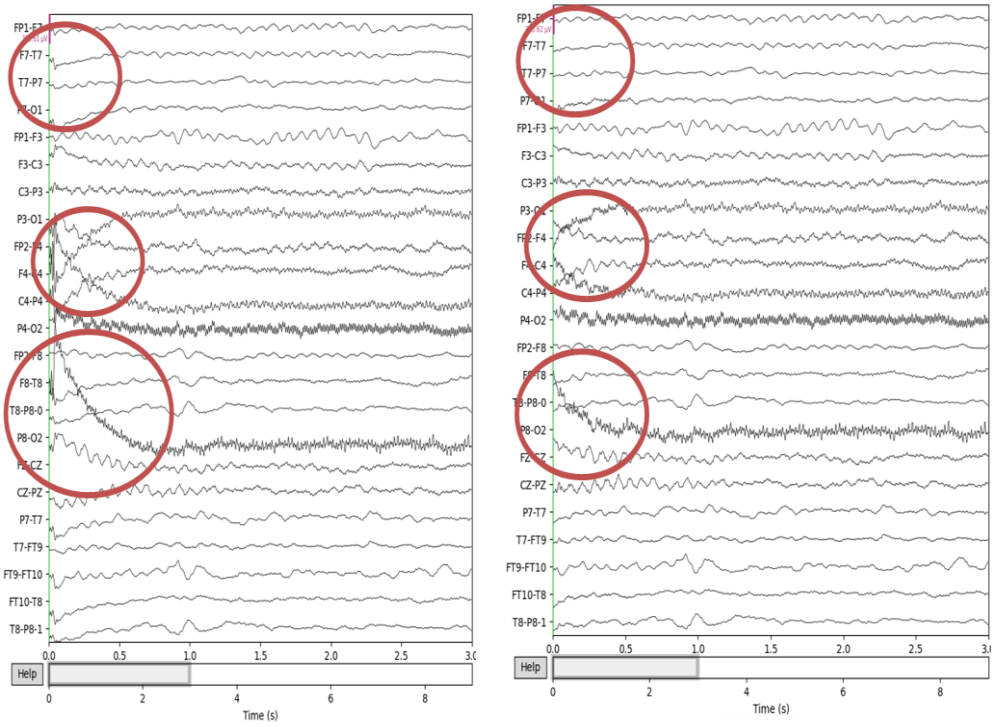
## 4.2 EEG Signals Artefact Rejection Results

CHB-MIT scalp dataset available on PhysioNet.org, and a dataset recorded in Al-Salam Hospital-Mosul have been taken as a database for testing the proposed artefact rejection algorithm. The datasets consist of EEG recordings of different patients of different ages. One of the key measurements of the validation of the algorithm is by visual assessment of EEG recordings before and after applying the algorithm. Figures 4.1, 4.2, 4.3, and 4.4 show samples of raw data compared with its reconstructed version after removing the noise sources. It's clear how the nonlinear signal is fine-tuned by comparing the portion encircled by a red ellipse between the same parts of the a and b figures.



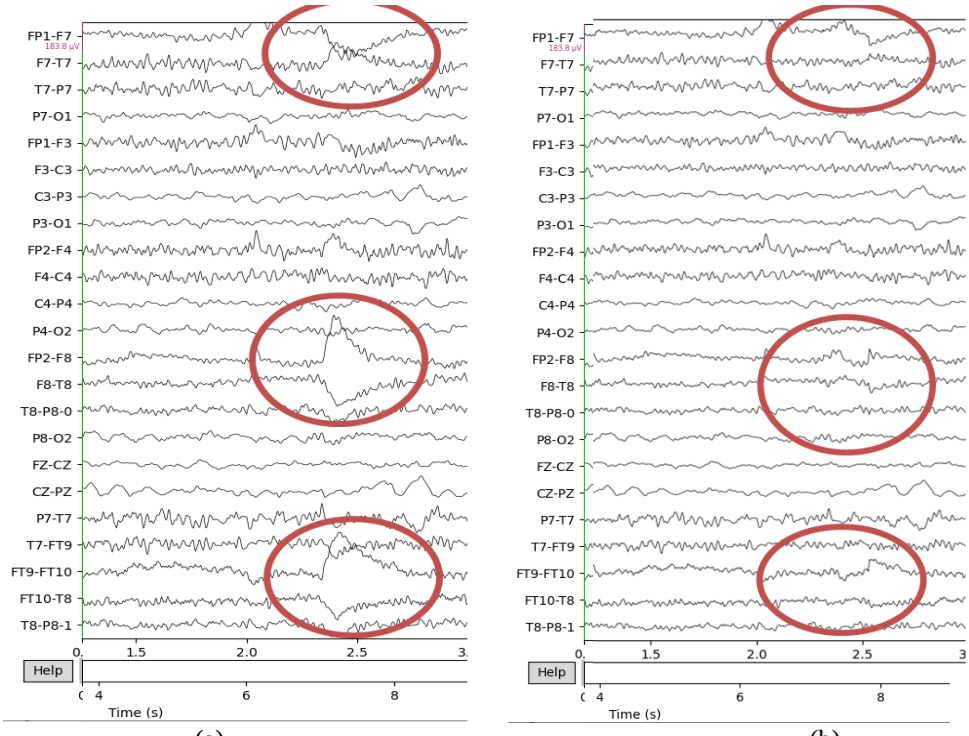

(a) (b)

Figure 4.1 Sample 1 (a) Raw EEG data and (b) Reconstructed EEG data after removing noise sources




(a) (b)

Figure 4.2 Sample 2 (a) Raw EEG data and (b) Reconstructed EEG data after removing noise sources




(a) (b)

Figure 4.3 Sample 3 (a) Raw EEG data and (b) Reconstructed EEG data after removing noise sources

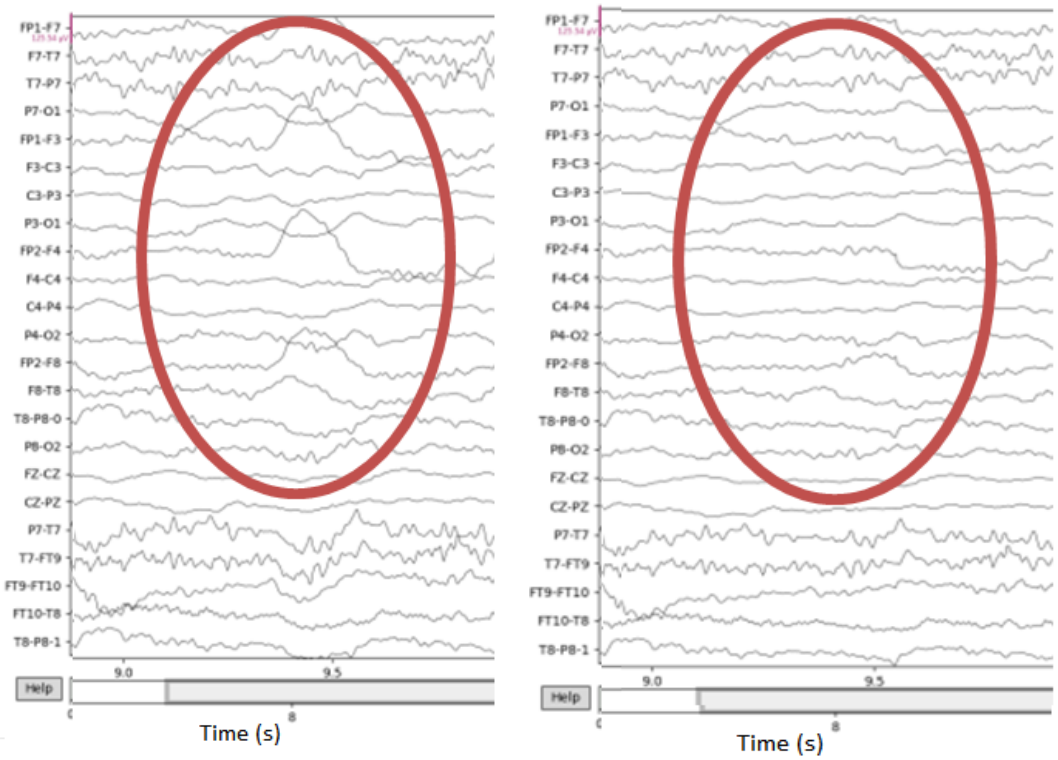



Figure 4.4 Sample 4 (a) Raw EEG data and (b) Reconstructed EEG data after removing noise sources

Another measure used to justify the algorithm readiness is the variance of the channel data. The variances are lower for reconstructed channel data compared with the variances for raw EEG channel data. The trend is to improve the signal-to-noise ratio since lower variances imply lower entropy and higher signal-to-noise ratio. Figure 4.5 shows the comparison of variance values between raw and reconstructed channels over 519 EEG samples.

### 4.2.1 The Utilized Software

The algorithm was implemented using the Python programming language with the Python-MNE library, and a graphical user interface (Figure 4.6) was made for the purpose of allowing the user to choose the length of the EEG segment and view the results. The programming codes are presented in Appendix B.

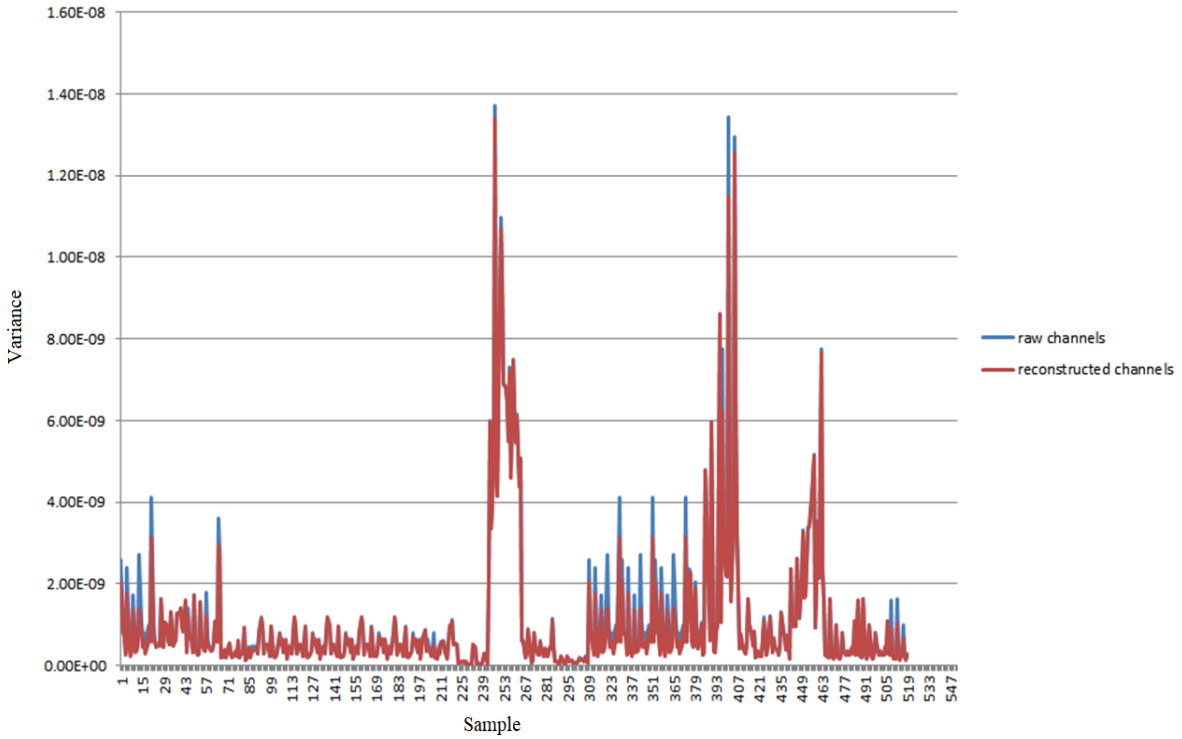



Figure 4.5 Comparison of variance values between the raw channels (blue line) and the reconstructed channels (red line)

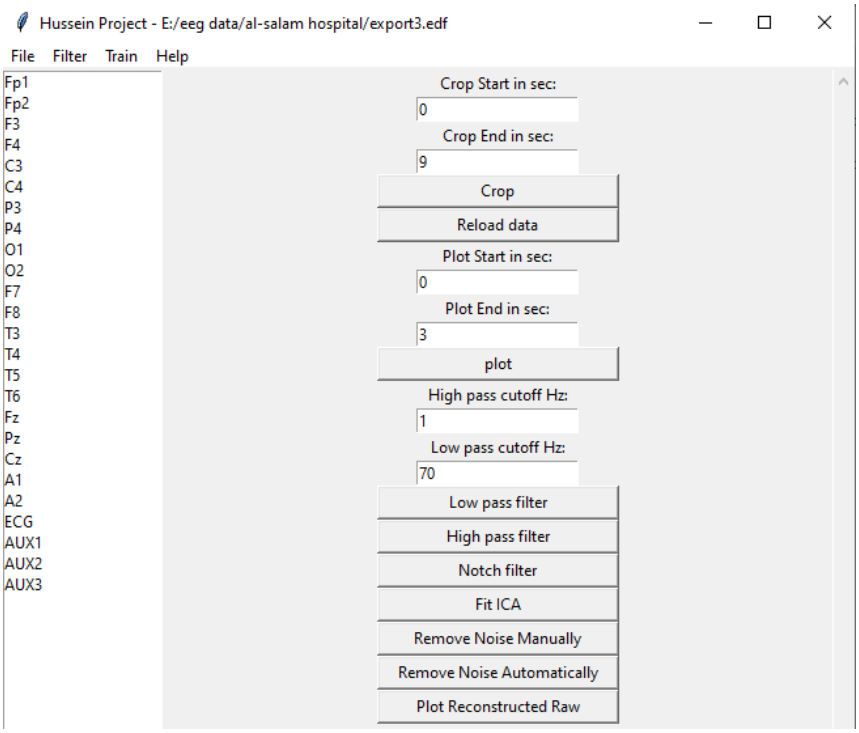



Figure 4.6 The implemented graphical user interface for the project

## 4.3 Deep Learning Models Results

In this unit, the results obtained from deep learning models are tabulated as classification accuracy, sensitivity, FPR, and F1 score. Because of the limited number of recorded seizures and in order to keep balanced training between two classes, for each patient, 10 min (600 s) of inter-ictal data and 10 min (600 s) pre-ictal data have been considered for training, validation, and test checking.

A seizure prediction system will be comfortable for any epilepsy patient if it uses a smaller number of channels; using one channel is the best situation. In the proposed models, five channels are chosen to be the input of the prediction system to balance the patient's comfort with the accuracy of the prediction.

The data consists of 92 seizure recordings from 14 patients. These data were separated into a 70% training set, a 15% validation set, and a 15% test to calculate the performance metrics for the proposed models.

### 4.3.1 Performance of the Seizure Detection Model

The suggested seizure detection model is trained with given data set for 10 epochs with each EDF file. The early stopping method is also included in the training, where the learning stops when there is no improvement in the performance without completing 10 epochs. Two metrics are monitored during the learning process: Loss and Accuracy.

The loss represents the difference between the model’s output and the actual labelled assigned to the class. Figure 4.7 shows the training loss and accuracy by feeding all data files (10 epochs for each file); the accuracy reached 99% for most of the data.

Compared with [92, 93, 94, 95, 96, 97] there is a great improvement in classification accuracy with the proposed model, in [98, 99] the authors not mentioned anything about the accuracy value but in this work, a dataset that have more EEG recordings than that used in [91] [92] is used. There is no significant improvement in accuracy in comparison with [100, 85] as shown in Table 4.1 but the

convergence time is reduced to 3 epochs since the data are fed into the neural network as 1- dimensional time series compared to 2- dimensional image in [100, 85]. However, the spatial features evoked by the model are large enough to be common features to detect any type of disease.

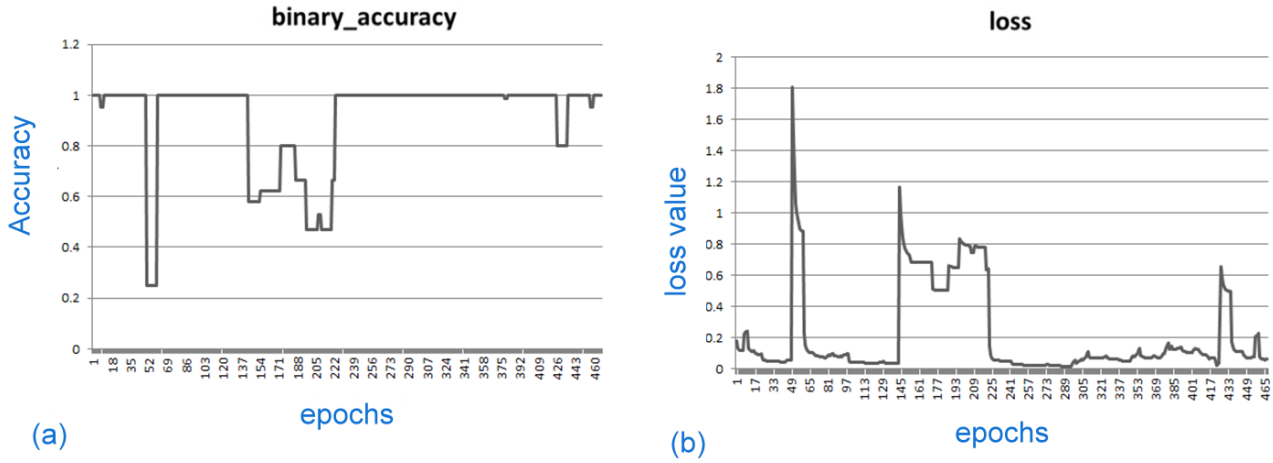



Figure 4.7 (a) Accuracy and (b) loss with epochs by feeding all data files

Another advantage of using time series gives the model property of being used in a real-time application, so it can be used in patient monitoring systems and BCIs applications. Visualization of features can be achieved by monitoring the output of each hidden layer in the model, and the specific features of non-stationary EEG signal can be extracted after the learning process.

Table 4.1 Results comparison with previous works

| Reference | Year | Model used | Accuracy | Database used |
|---|---|---|---|---|
| [92] | 2020 | Multilevel CNN models with auto-encoder. | 74.80% | BCI dataset |
| [100] | 2019 | FFT and WPD, with CNN. | 98.33±0.18% | CHB-MIT dataset |
| [98] | 2017 | Deep CNN. | - | The BCI competition IV dataset 2a |

Table 4.1 (continued)

| | | | | |
|---|---|---|---|---|
| [96] | 2019 | Deep CNN. | 94.67% | Data Collected by Authors |
| [97] | 2019 | Deep CNN. | 87.96% | Temple University Hospital EEG Abnormal Corpus v2.0.0 |
| [99] | 2020 | Wavelet transform, CNN | - | Bern-Barcelona data base |
| [85] | 2020 | Transfer learning, CNN | 98% | CHB-MIT database |
| **Proposed work** | | RNN-CNN | 99% | CHB-MIT database |

### 4.3.2 Performance of Seizure Prediction Models

Results of 90% in terms of accuracy and 90% in terms of sensitivity were obtained using the first model. As well as 71% accuracy and 73% sensitivity were obtained using the second model. Finally, with respect to the second method by using the average voting technique, a portion of the EEG signal with a length of 2 minutes was used the results were obtained: 74% accuracy, 84% sensitivity.

#### *4.3.2.1 Performance of Seizure Prediction Model 1*

The performance is evaluated using only the test set of the data. The achieved mean accuracy, sensitivity, FPR, and F1 score are 66.3%, 62.5 %, 0.234, and 62%, respectively, as presented in Table 4.2 For patient ID-Chb-10 maximum performance metrics were achieved, where the accuracy, sensitivity, FPR, and F1 score are 90%, 90%, 0.05, 90% respectively.

The difference between the measurements' values demonstrates the dependence of the expectation on the nature of the patient's brain signals, as in some patients a seizure can be expected and in another, it is difficult to predict its occurrence. The ROC curve samples are shown in Figure 4.8. The maximum achieved AUC is 0.75 for patient ID-Chb-08 which is acceptable for this type of prediction.

Table 4.2 Performance metrics for Model 1

| Patient ID | Accuracy | Sensitivity Recall | FPR | F1 Score |
|---|---|---|---|---|
| CHB-P01 | 0.65 | 0.65 | 0.3 | 0.65 |
| CHB -P03 | 0.48 | 0.49 | 0.3 | 0.46 |
| CHB -P04 | 0.57 | 0.56 | 0.3 | 0.55 |
| CHB -P05 | 0.61 | 0.57 | 0.4 | 0.58 |
| CHB -P06 | 0.54 | 0.54 | 0.39 | 0.53 |
| CHB -P08 | 0.74 | 0.61 | 0.2 | 0.63 |
| CHB -P10 | 0.90 | 0.90 | 0.05 | 0.90 |
| CHB -P18 | 0.80 | 0.68 | 0.05 | 0.68 |
| CHB -P20 | 0.58 | 0.51 | 0.22 | 0.49 |
| CHB -P24 | 0.76 | 0.74 | 0.12 | 0.73 |
| Average | 0.663 | 0.625 | 0.23 | 0.62 |

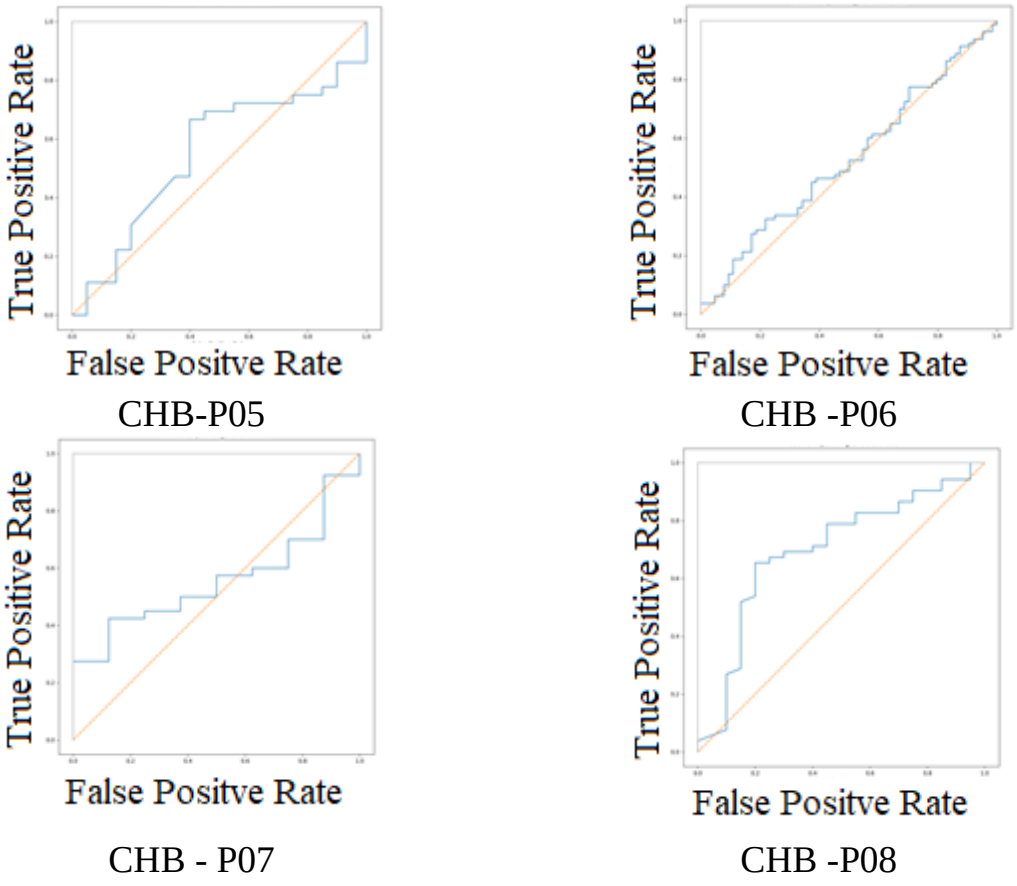
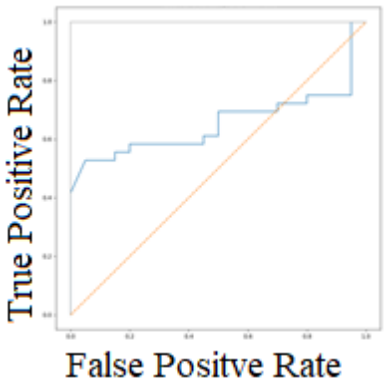
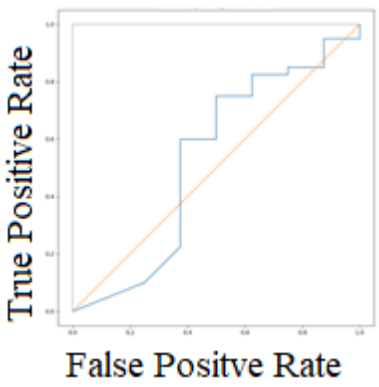



Figure 4.8 ROC-AUC curves for Model 1




CHB – P18




CHB - P22

Figure 4.8 (continued)

#### *4.3.2.2 Performance of Seizure Prediction Model 2*

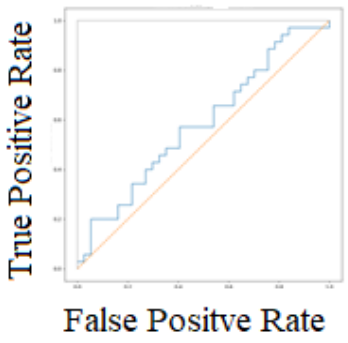
The achieved average accuracy, accuracy, sensitivity, FPR, F1 score, and ROC_AUC are 56.1%, 54.9 %, 56.1%, 0.234, 54.1%, 57.3%, respectively, as presented in Table 4.3. For patient ID CHB-05 maximum performance metrics were achieved, where the accuracy, accuracy, sensitivity, FPR, F1 score and RO_AUC are 71%, 73%, 71%, 0.33, and 72% respectively. The ROC curve samples for Model 2 are shown in Figure 4.9. The maximum achieved AUC is 0.77 for patient ID CHB-05.

Table 4.3 Performance metrics for Model 2

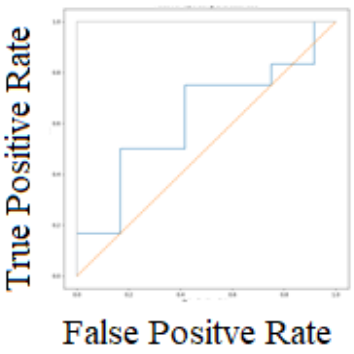
| Patient ID | Accuracy | Accuracy | Sensitivity Recall | FPR | F1 Score | ROC_AUC Score | Train Samples | Validation or Test Samples |
|---|---|---|---|---|---|---|---|---|
| CHB-P01 | 0.58 | 0.58 | 0.58 | 0.4 | 0.58 | 0.583 | 336 | 72 |
| CHB-P02 | 0.50 | 0.25 | 0.5 | 0 | 0.33 | 0.625 | 112 | 24 |
| CHB-P03 | 0.48 | 0.49 | 0.48 | 0.4 | 0.48 | 0.49 | 280 | 60 |
| CHB-P04 | 0.52 | 0.52 | 0.52 | 0.69 | 0.50 | 0.471 | 224 | 48 |
| CHB-P05 | 0.71 | 0.73 | 0.71 | 0.33 | 0.72 | 0.775 | 196 | 42 |
| CHB-P06 | 0.57 | 0.58 | 0.57 | 0.27 | 0.56 | 0.555 | 504 | 108 |
| CHB-P07 | 0.61 | 0.61 | 0.61 | 0.31 | 0.61 | 0.606 | 168 | 36 |
| CHB-P08 | 0.65 | 0.65 | 0.65 | 0.32 | 0.65 | 0.677 | 252 | 54 |
| CHB-P09 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.506 | 224 | 48 |
| CHB-P10 | 0.55 | 0.58 | 0.55 | 0.28 | 0.54 | 0.537 | 364 | 78 |

Table 4.3 (continued)

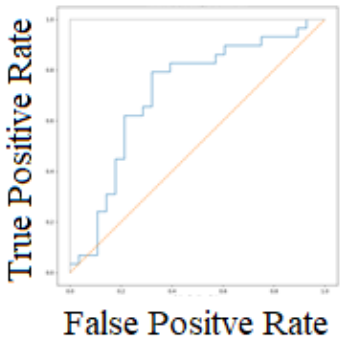
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| CHB-P20 | 0.56 | 0.56 | 0.56 | 0.18 | 0.52 | 0.603 | 336 | 72 |
| CHB-P22 | 0.56 | 0.58 | 0.56 | 0.23 | 0.54 | 0.502 | 168 | 36 |
| CHB-P23 | 0.46 | 0.46 | 0.46 | 0.55 | 0.46 | 0.483 | 364 | 78 |
| CHB-P24 | 0.6 | 0.6 | 0.6 | 0.47 | 0.59 | 0.614 | 728 | 156 |
| Average | 0.561 | 0.549 | 0.561 | 0.35 | 0.541 | 0.573 | | |



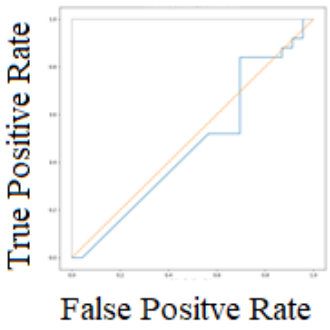

CHB-P01



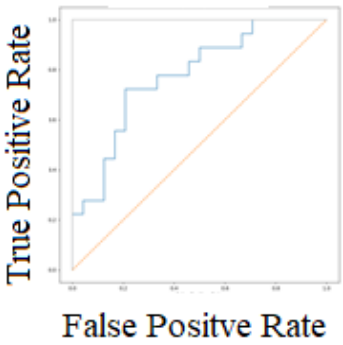

CHB-P02



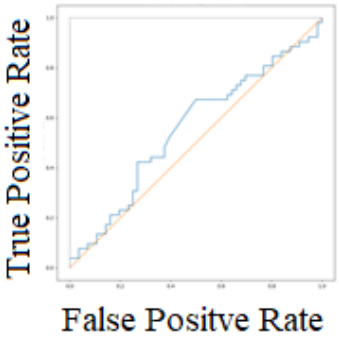

CHB-P03




CHB-P04




CHB-P05




CHB-P06

Figure 4.9 ROC-AUC curves for Model 2

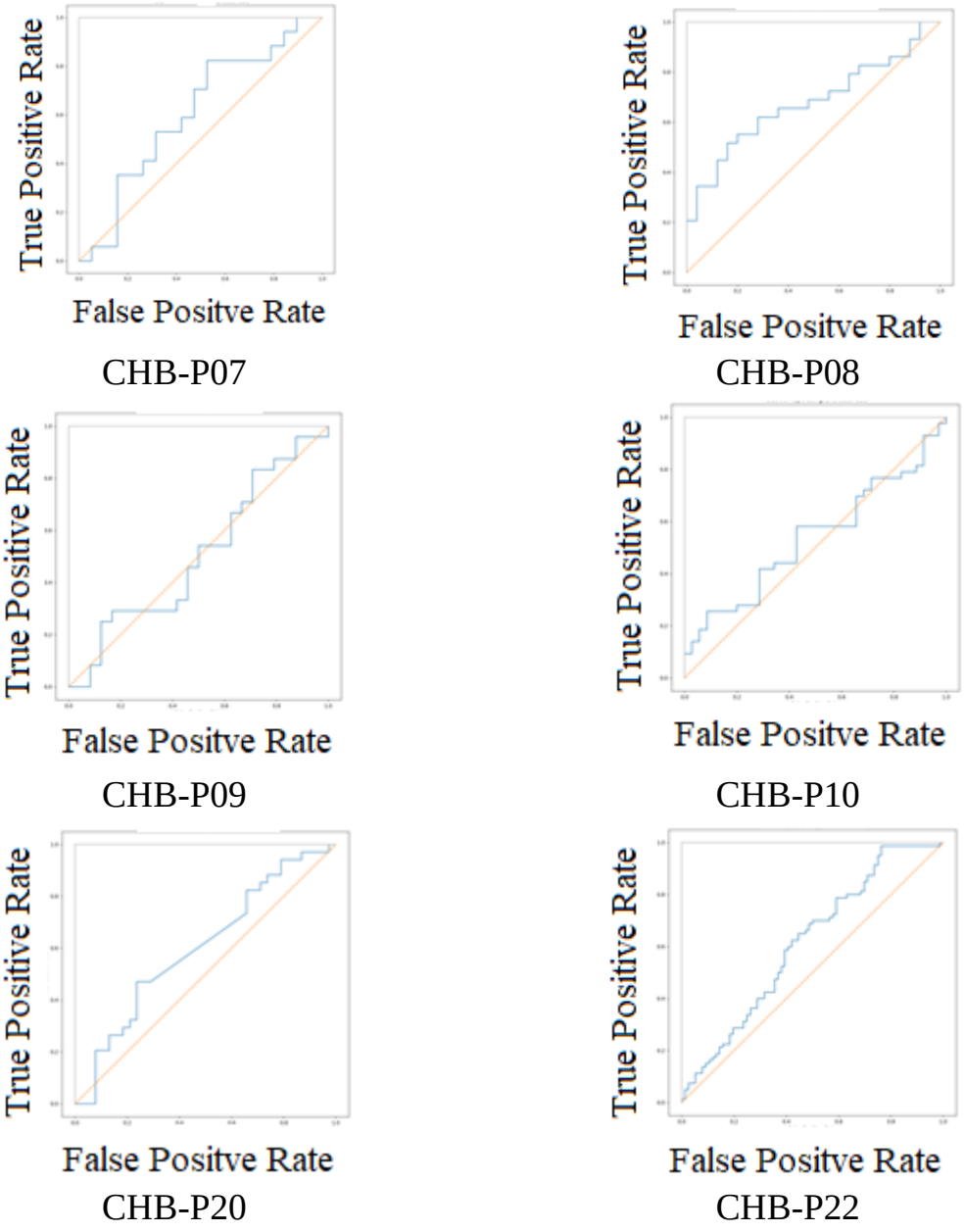


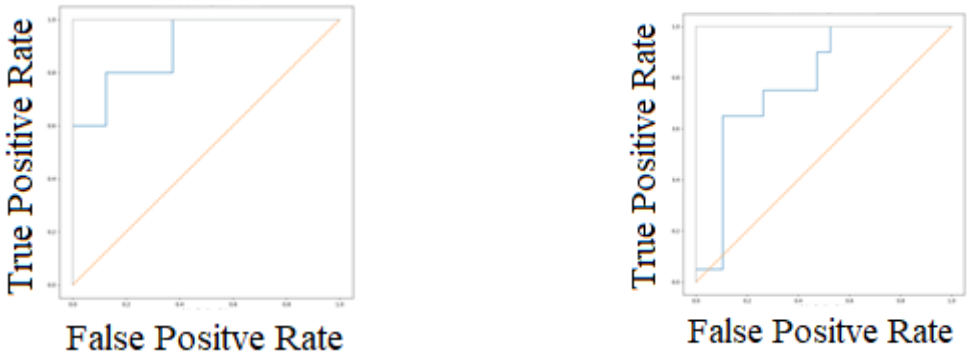

Figure 4.9 (continued)

#### *4.3.2.3 Performance of Seizure Prediction Model 2 using the Average Voting Technique*

A small sample of EEG for a patient is not adequate to predict seizures with high accuracy as advised by health workers. Thus, 1-minute and 2- minutes EEG samples have been respected to predict epilepsy seizures. To do so, a modification of the majority voting technique is used, which is averaging over the multi-classifier decisions. For 1-minute EEG samples, 4 instances of a pre-trained model have been

used to predict seizures, and the output of 4 models was averaged to obtain the final decision. For 2- minute EEG samples, 8 instances were used and the output was also averaged.

The ROC of the two models shown in Figure 4.10 clarifies that the maximum ROC-AUC of the 2-minute sample model is equal to 0.85 beats the 1-minute sample model which has a maximum ROC-AUC equal to 0.79. Correspondingly, the performance metrics for the average voting model, for patient ID-Chb-24, are 74%, 84%, 74%, 0.45, 73%, and 0.85 for accuracy, accuracy, sensitivity, FPR, F1 score, and ROC_AUC respectively, where are better than the 15-second sample model.




Chb-01 Chb-24

Figure 4.10 ROC-AUC samples for Model 2 with average voting technique

### 4.3.3 Discussion And Comparison

A comparative study of earlier works [33, 34, 23, 24] with the proposed models is shown in Table 4.4. By comparing the results with regard to accuracy and sensitivity with most of the research presented in Table 4.4, it appears at first glance that the results are not feasible and the reason depends that these researchers have used different entries for the number of electrodes and *PIL* duration, so the comparison will be unfair.

In addition, these researches did not state the amount of data used in training and whether the data used for evaluation was included the training data or not, as in this work the data are divided into 70% training data, 15% evaluation data and 15% test data.

It can be said that the proposed method is the best practice among the methods presented in Table 4.4 as it uses the least number of electrodes, which has a great benefit for patient comfort. Also, the short warning period of 10 minutes is important to relieve the patient's stress. A short warning period was used in the research [21], and it was 15 minutes long, and it obtained a sensitivity rate of 69% means that the suggested method is better than the method used in the research [21], where a sensitivity of 70-90% is reached.

### 4.3.4 Utilized Software

The models were programmed using the Python programming language with the Keras and Tensor-flow libraries and implemented in a GPU for the purpose of speeding up training and testing. The programming codes are presented in Appendix C.

Table 4.4 Comparison of earlier methods with proposed models

| Research work | Feature extraction method | Machine /Deep learning model type | Number of channels | Number of patients | PIL in minutes | Accuracy | Sensitivity | FPR |
|---|---|---|---|---|---|---|---|---|
| [33] | Short-time Fourier transform | CNN | 23 | 13 | | | 81.2% | 0.1600 |
| [34] | Wavelet transform | CNN | 22 | 15 | | - | 87.8% | 0.1420 |
| [24] | Raw EEG | CNN | 6 | 23 | 2 | 99.47% | 97.83% | 0.0764 |
| [25] | Mel Frequency Cepstral Coefficients (MFCCs) | CNN | 22 | 12 | 60 | 88.81% 90.54% | - | - |
| [22] | Raw EEG | CNN | 22 | 23 | 60 | 99.6% | - | - |
| [21] | Raw EEG | CNN | 22 | 10 | 15 | - | 69% | |
| [27] | Raw EEG | kNN classifier | 22 | 4 | 90 | - | - | - |
| [26] | Raw EEG combined with Handcrafted features | Ensemble of SVM, CNN and LSTM | - | 2 | 33 | | 96.28% | - |
| Proposed Method 1 | Raw EEG | CNN | 5 | 10 | 10 | 90% Average: 66.3% | 90% Average: 62.5 % | 0.05 Average: 0.234 |
| Proposed Method 2 | Raw EEG | GRU-LSTM | 5 | 14 | 10 | 71% Average:56% | 73% Average:55% | 0.33 Average:0.35 |
| Proposed Method 2 utilizing average voting | Raw EEG | GRU-LSTM, Average voting | 5 | 14 | 10 | 74% | 84% | 0.45 |

## 4.4 Machine Learning Model Results

In this section, the performance of the proposed algorithm and obtained results are presented. The performance of the prediction algorithm was measured in terms of specificity and sensitivity. In a seizure prediction test that screens EEG segments for a pre-ictal, each tested segment either is a pre-ictal segment or not. The test result could be positive or negative, which indicates that the segment is pre-ictal or inter-ictal, respectively.

Table 4.5 displays the results of the round with 10 patients, it's clear that the selected channel for each patient is well-matched with the seizure type of that patient. Table 4.6 represents the output of the last three rounds of the algorithm. The results showed a difference in the pre-ictal period between patients (*PIL*), ranging from 20-30 minutes on the CHB-MIT scalp database. Also, the length of the EEG segment (*SEG*) used in the training process differed between patients, ranging from 20 seconds to 30 seconds. These values of *PIL* and *SEG* gave the highest percentage of discrimination between the inter-ictal and pre-ictal traits in patients, which would increase the accuracy and specificity in predicting a seizure.

The in-sample loss and test loss data give evidence of the generality of the algorithm and it's devoid of over-fitting. It's noted that the distinguishing features are different between patients with common features among a number of patients, for example, feature No. 59, which represents the standard deviation of the intrinsic mode function No. 3. Other common features among patients are: Feature No. 1 and No. 33, which represent the largest Lyapunov exponent of EEG segment and the signal power in alpha band respectively. In addition, the important features in distinguishing between the inter-ictal and the pre-ictal periods are the following: standard deviation, skewness and kurtosis of the EEG signal in the time domain, the power of EEG signal in the theta band, alpha band, beta band, and gamma band in the frequency domain, the number of extreme points, sifting, and zero crossing in

empirical mode decomposition, mean and standard deviation of intrinsic mode functions.

Table 4.7 summarizes the overall performance of the algorithm and Figure 4.11 gives samples of the confusion matrix of the model (for only test data which is 30% of overall data). The sensitivity, specificity, accuracy, and false alarm are listed for each patient. High accuracy with a very low false alarm rate was gained for patients 1, 8, and 9. While for other patients, good accuracy with relatively low false alarms is obtained.

Table 4.5 The results of channel selection round with seizure type.

| Patient ID | Gender | Age | Seizure origin | Selected channel |
|---|---|---|---|---|
| CHB-P01 | F | 11 | Frontal | FP2-F8 |
| CHB-P02 | M | 11 | Temporal | T8-P8 |
| CHB-P03 | F | 14 | Frontal | FP1-F7 |
| CHB-P04 | M | 22 | Temporal | T7-FT9 |
| CHB-P05 | F | 7 | Frontal | FP2-F8 |
| CHB-P06 | F | 1.5 | Temporal/Occipital | FP1-F3 |
| CHB-P07 | F | 14.5 | Temporal | FP1-F7 |
| CHB-P08 | M | 3.5 | Frontal | T7-FT9 |
| CHB-P09 | F | 10 | Temporal/Occipital | C4-P4 |
| CHB-P10 | M | 3 | Temporal | T7-P7 |
| CHB-P23 | F | 6 | Temporal | FT9-FT11 |
| CHB-P24 | - | - | - | C3-P3 |

Table 4.6 PIL, SEG, and optimal features for each patient obtained from the algorithm.

| Patient ID | *PIL* min | *SEG* sec | Optimum Channel | In sample loss | Test loss | Optimum feature indexes returned by the algorithm |
|---|---|---|---|---|---|---|
| CHB-P01 | 20 | 25 | 13 | 0 | 0.0347 | 59 57 19 12 16 25 65 3 8 62 |
| CHB-P02 | 20 | 20 | 15 | 0.2024 | 0.1780 | 42 4 59 54 33 55 28 53 34 48 |

Table 4.6 (continued)

| | | | | | | |
|---|---|---|---|---|---|---|
| CHB-P03 | 20 | 25 | 1 | 0.3309 | 0.3221 | 56 4 59 34 62<br>33 29 21 30 9 |
| CHB-P04 | 20 | 25 | 1 | 0.3941 | 0.4839 | 18 12 17 47 21<br>20 5 29 28 60 |
| CHB-P05 | 10 | 30 | 13 | 0.2054 | 0.2194 | 21 11 16 65 61<br>9 59 18 53 39 |
| CHB-P06 | 30 | 30 | 4 | 0.2534 | 0.2572 | 55 43 39 42 28<br>53 51 56 33 27 |
| CHB-P07 | 20 | 30 | 1 | 0.2768 | 0.4444 | 4 52 24 19 11<br>58 8 53 64 3 |
| CHB-P08 | 10 | 20 | 20 | 0.0476 | 0.0610 | 46 19 54 33 36<br>57 18 51 55 5 |
| CHB-P09 | 10 | 30 | 11 | 0 | 0.3281 | 42 57 61 4 51<br>1 63 21 20 59 |
| CHB-P10 | 10 | 20 | 3 | 0.2381 | 0.3100 | 12 21 6 54 11<br>25 59 33 1 16 |
| CHB-P23 | 30 | 15 | 22 | 0 | 0.196 | 52 42 58 34 10<br>51 25 43 3 1 |

Table 4.7 Seizure prediction results achieved with the CHB–MIT scalp EEG database.

| Patient ID | TN samples | FP samples | FN samples | TP samples | sensitivity | specificity | accuracy | FA/h |
|---|---|---|---|---|---|---|---|---|
| CHB-P01 | 239 | 1 | 4 | 236 | 0.983 | 0.996 | 0.999 | 0 |
| CHB-P02 | 105 | 15 | 32 | 88 | 0.733 | 0.875 | 0.804 | 0.125 |
| CHB-P03 | 136 | 56 | 70 | 122 | 0.635 | 0.708 | 0.672 | 0.29 |
| CHB-P04 | 119 | 77 | 87 | 105 | 0.547 | 0.607 | 0.577 | 0.39 |
| CHB-P05 | 72 | 8 | 27 | 53 | 0.663 | 0.9 | 0.781 | 0.1 |
| CHB-P06 | 315 | 78 | 136 | 284 | 0.676 | 0.802 | 0.737 | 0.19 |
| CHB-P07 | 48 | 32 | 20 | 60 | 0.75 | 0.6 | 0.675 | 0.4 |
| CHB-P08 | 133 | 17 | 0 | 150 | 1 | 0.887 | 0.943 | 0.1 |
| CHB-P09 | 75 | 5 | 10 | 70 | 0.875 | 0.938 | 0.906 | 0.06 |
| CHB-P10 | 136 | 74 | 36 | 174 | 0.829 | 0.648 | 0.738 | 0.35 |
| Average | | | | | 0.7691 | 0.7961 | 0.7832 | 0.2 |

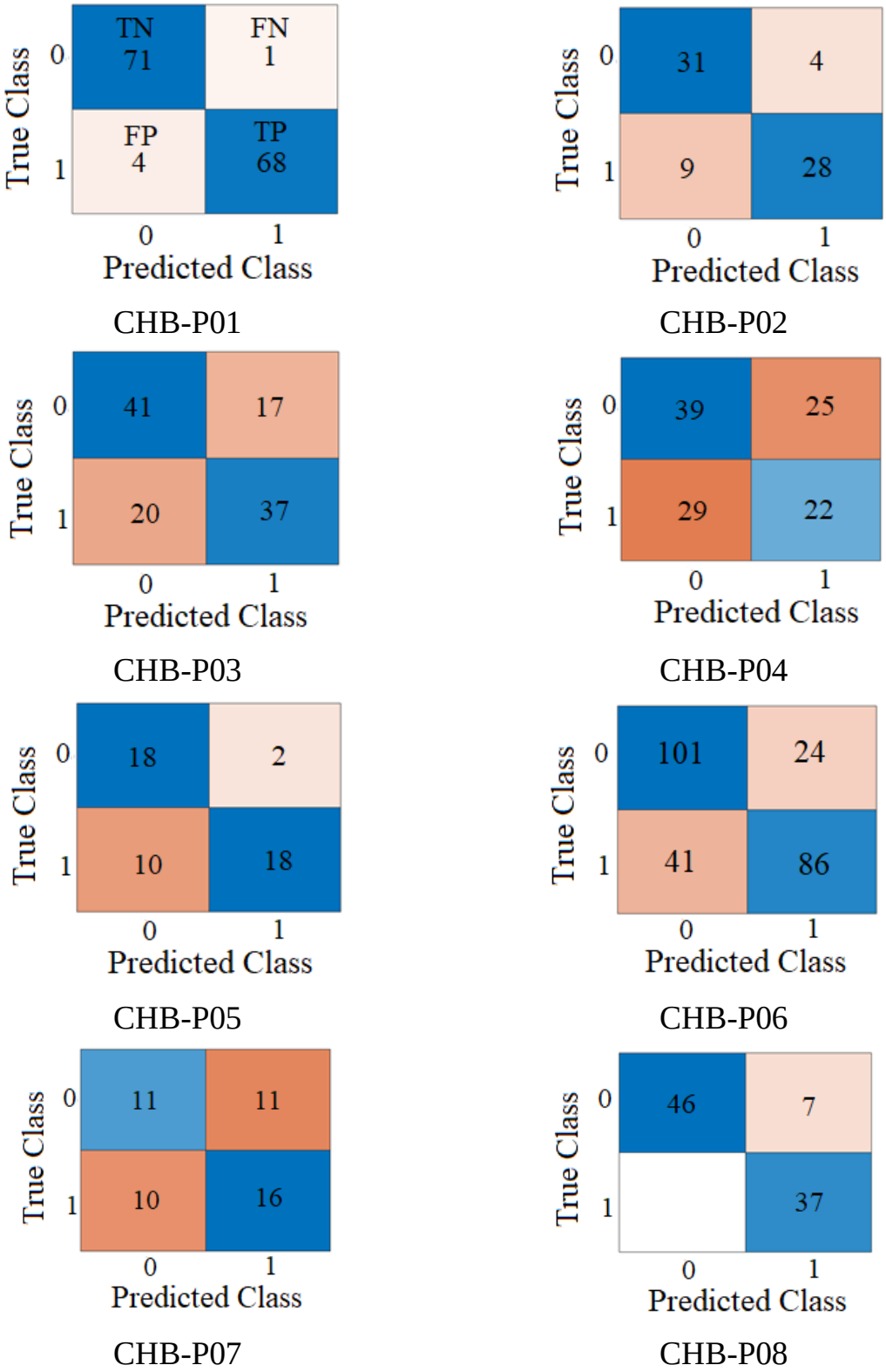



Figure 4.11 Samples of confusion matrix of the model.

### 4.4.1 Comparing with other works

Comparing the proposed algorithm with recent studies showed the superiority of the algorithm over many of them, although in some studies the accuracy and sensitivity are higher than what is found using the proposed algorithm, the algorithm excels in many characteristics, including the number of used channels, as only one EEG channel is used compared to the number of channels in studies [33, 35, 34, 37, 30] which are 23 channels. Also, the relatively short *PIL* period, reduces stress and anxiety in the patient, compared to long periods in other research. As for the extracted features, the algorithm suggests different features for each patient, which will be important in reducing the arithmetic operations required to extract these features. As indicated in Table 4.8 this work succeeded with all patients while other studies fail in prediction with patients 2 and 9 [33, 30]. The average sensitivity is 77% and the average FPR is 0.2.

### 4.4.2 Results obtained with Siena Scalp Database

In this section, results from applying the algorithm with the Siena scalp database are tabulated. Siena scalp database consists of EEG recordings for 14 epileptic patients sampled at 512 Hz. Table 4.9 summarizes the information about 14 patients with an optimum channel for each patient produced by the algorithm. With the Siena scalp database, a small number of discriminating features were observed during experiments; therefore, the number of required features is lowered to only five so the algorithm will work correctly.

The results of the last rounds of the algorithm are tabulated in Table 4.10. Here, the optimum discriminating features also vary among patients. Lyapunov Exponent and Number of Extrema are more frequent features among patients.

Table 4.8: Comparing results with recent studies in seizure prediction

| Patient ID | [33] | | [35] | | [34] | | [37] | | [30] | | [31] | | [25] | | This work. | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Sensitivity | FPR/ hour | Sensitivity | FPR/ hour | Sensitivity | FPR/ hour | Sensitivity | FPR/ hour | Sensitivity | FPR/ hour | Sensitivity | FPR/ hour | Sensitivity | FPR/ hour | Sensitivity | FPR/ hour |
| CHB-P01 | 85.7 | 0.24 | 100 | 0 | - | - | 100 | 0.03 | 100 | 0.33 | 100 | 0.06 | - | - | 98.3 | 0.004 |
| CHB-P02 | 33.3 | 0 | 100 | 0 | - | - | 100 | 0.04 | 67 | 0.42 | - | - | - | - | 73.3 | 0.125 |
| CHB-P03 | 100 | 0.18 | 100 | 0.26 | - | - | 100 | 0.06 | 50 | 0.13 | 66 | 0 | - | - | 63.5 | 0.29 |
| CHB-P04 | - | - | 100 | 0.15 | - | - | 100 | 0.04 | 100 | 0.85 | - | - | - | - | 54.6 | 0.39 |
| CHB-P05 | 80 | 0.19 | 100 | 0.31 | - | - | 66.7 | 0.11 | 80 | 0.56 | 66 | 0.612 | - | - | 66.2 | 0.1 |
| CHB-P06 | - | - | 100 | 0.37 | - | - | 100 | 0.17 | 10 | 0.11 | 66 | 0.52 | - | - | 67.6 | 0.19 |
| CHB-P07 | - | - | 100 | 0 | - | - | 100 | .03 | 100 | 0.45 | - | - | - | - | 75 | 0.4 |
| CHB-P08 | - | - | 100 | 0 | - | - | - | - | 100 | 0.46 | 100 | 0.88 | - | - | 100 | 0.1 |
| CHB-P09 | 50 | 0.12 | 100 | 0.13 | - | - | 50 | 0.06 | 100 | 0.6 | - | - | - | - | 87.5 | 0.06 |
| CHB-P10 | 33.3 | 0 | 100 | 0 | - | - | 100 | 0.18 | 100 | 0.47 | 100 | 1.59 | - | - | 82.5 | 0.35 |
| CHB-P23 | 100 | 0.33 | 100 | 0 | - | - | - | - | 100 | 0.67 | 100 | 0.52 | - | - | | |
| CHB-P24 | - | - | 100 | 0.28 | - | - | 66.7 | 0.08 | 50 | 0.37 | - | - | - | - | | |
| Average | 81.2 | 0.16 | 99.3 | 0.11 | 87.8 | 0.14 | 87.3 | 0.08 | 0.81 | 0.47 | 86.67 | 0.367 | 93.45 | 0.18 | 77 | 0.2 |
| Number of channels | 23 | | 23 | | 23 | | 23 | | 23 | | 1 | | 23 | | 1 | |
| PIL | 5 min | | >15 min | | 10 min | | 60 min | | 60 min | | 86 min | | 1 hour | | >10 min specific for each patient | |
| Features | STFT spectral images | | Statistical moments, zero crossings, Wavelet Transform coefficients, PSD, cross correlation, graph theory | | Wavelet Transform coefficients | | Normalized Logarithmic Wavelet Packet Coefficient Energy Ratios | | Common spatial pattern | | Fourier coefficients of six EEG frequency bands, attractor state analysis | | - | | Patient specific | |

Table 4.9 The results of channel selection round with seizure type for Siena scalp database.

| Patient ID | Gender | Age | Seizure origin | Selected channel |
|---|---|---|---|---|
| PN00 | Male | 55 | Temporal | T6 |
| PN01 | Male | 46 | Temporal | T3 |
| PN03 | Male | 54 | Temporal | Fc1 |
| PN05 | Female | 51 | Temporal | F8 |
| PN06 | Male | 36 | Temporal | F7 |
| PN07 | Female | 20 | Temporal | Cz |
| PN09 | Female | 27 | Temporal | T3 |
| PN10 | Male | 25 | Frontal | P4 |
| PN11 | Female | 58 | Temporal | P4 |
| PN12 | Male | 71 | Temporal | P4 |
| PN13 | Female | 34 | Temporal | Cp5 |
| PN14 | Male | 49 | Temporal | F8 |
| PN16 | Female | 41 | Temporal | C4 |
| PN17 | Male | 42 | Temporal | F4 |

Table 4.10 PIL, SEG, and optimal features for each patient obtained from the algorithm with the Siena scalp database.

| Patient ID | *PIL* min | *SEG* sec | Optimum Channel | In sample loss | Test loss | Optimum feature indexes returned by the algorithm |
|---|---|---|---|---|---|---|
| PN00 | 10 | 10 | T6 | 0.2429 | 0.3191 | 7 63 8 52 1 |
| PN01 | 10 | 10 | T3 | 0.1667 | 0.2181 | 7 16 1 3 6 |
| PN03 | 20 | 10 | Fc1 | 0.238 | 0.213 | 4 20 56 59 54 |
| PN05 | 10 | 10 | F8 | 0.134 | 0.121 | 8 27 10 36 1 |
| PN06 | 20 | 10 | F7 | 0.162 | 0.200 | 10 11 54 55 8 |
| PN07 | 30 | 15 | Cz | 0 | 0.0136 | 25 34 57 63 15 |
| PN09 | 20 | 10 | T3 | 0.1925 | 0.2582 | 19 3 7 42 52 |
| PN10 | 20 | 30 | P4 | 0.173 | 0.275 | 33 10 13 42 34 |
| PN11 | 20 | 20 | P4 | 0.011 | 0 | 34 42 28 7 65 |
| PN12 | 10 | 10 | P4 | 0.0913 | 0.0747 | 26 42 24 57 7 |
| PN13 | 20 | 30 | Cp5 | 0.0952 | 0.1949 | 63 55 59 7 47 |
| PN14 | 20 | 15 | F8 | 0.324 | 0.350 | 24 63 46 59 11 |
| PN16 | 10 | 25 | C4 | 0 | 0.025 | 17 33 24 51 25 |
| PN17 | 30 | 15 | F4 | 0.195 | 0.221 | 33 65 34 55 4 |

Table 4.11 summarizes the overall performance of the algorithm and Figure 4.12 gives samples of the confusion matrix of the model (for only test data which is 30% of overall data). The average sensitivity, specificity, and accuracy are 0.892, 0.826, and 0.859 respectively.

Table 4.11 Seizure prediction results obtained with the Siena scalp EEG database.

| Patient ID | TN samples | FP samples | FN samples | TP samples | Sensitivity | specificity | accuracy |
|---|---|---|---|---|---|---|---|
| PN00 | 184 | 116 | 42 | 258 | 0.86 | 0.613 | 0.737 |
| PN01 | 106 | 14 | 30 | 90 | 0.75 | 0.883 | 0.817 |
| PN03 | 234 | 6 | 5 | 235 | 0.979 | 0.975 | 0.977 |
| PN05 | 138 | 42 | 8 | 172 | 0.955 | 0.766 | 0.861 |
| PN06 | 505 | 95 | 113 | 487 | 0.812 | 0.842 | 0.827 |
| PN07 | 238 | 2 | 0 | 240 | 1 | 0.992 | 0.996 |
| PN09 | 256 | 104 | 49 | 311 | 0.864 | 0.711 | 0.788 |
| PN10 | 207 | 48 | 53 | 187 | 0.779 | 0.812 | 0.796 |
| PN11 | 59 | 1 | 0 | 60 | 1 | 0.983 | 0.992 |
| PN12 | 169 | 11 | 20 | 160 | 0.889 | 0.939 | 0.914 |
| PN13 | 91 | 29 | 2 | 118 | 0.983 | 0.758 | 0.871 |
| PN14 | 160 | 160 | 53 | 267 | 0.834 | 0.5 | 0.667 |
| PN16 | 48 | 0 | 1 | 47 | 0.979 | 1 | 0.99 |
| PN17 | 190 | 50 | 48 | 192 | 0.8 | 0.792 | 0.796 |
| Average | | | | | 0.892 | 0.826 | 0.859 |

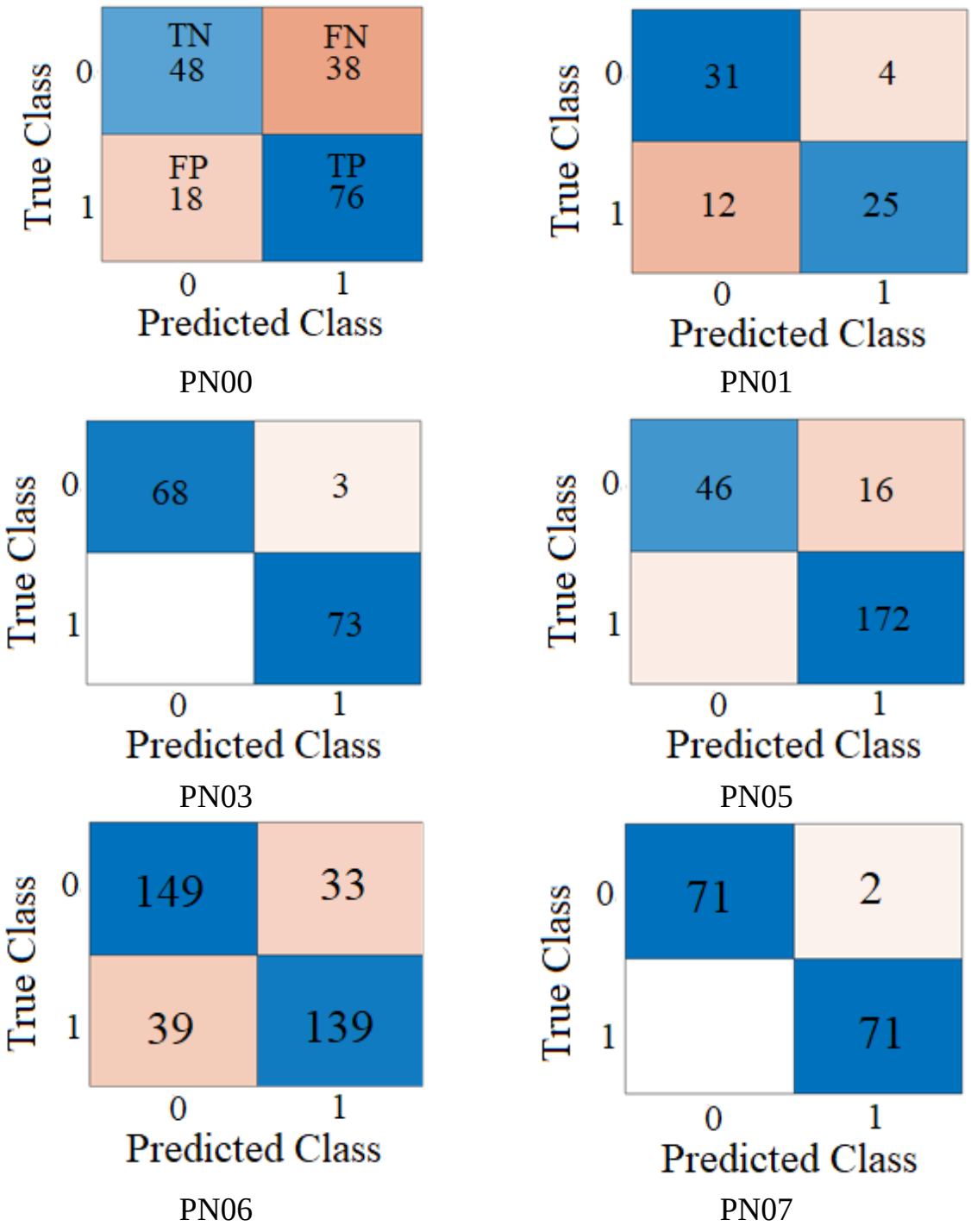
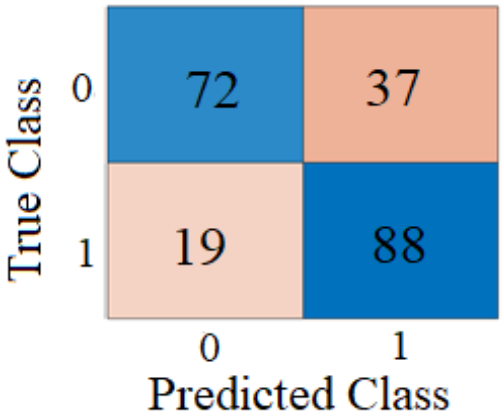
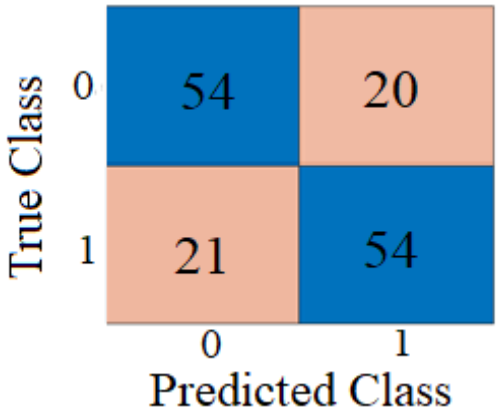


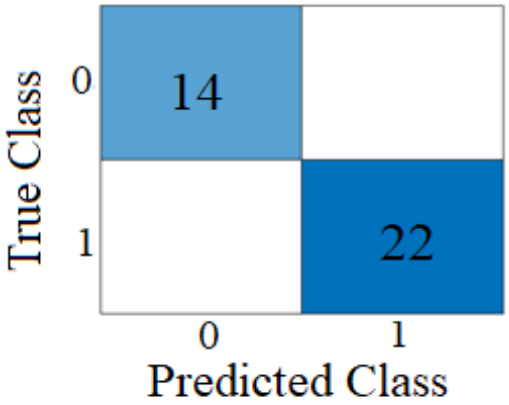

Figure 4.12 Samples of confusion matrix of the model with Siena database.



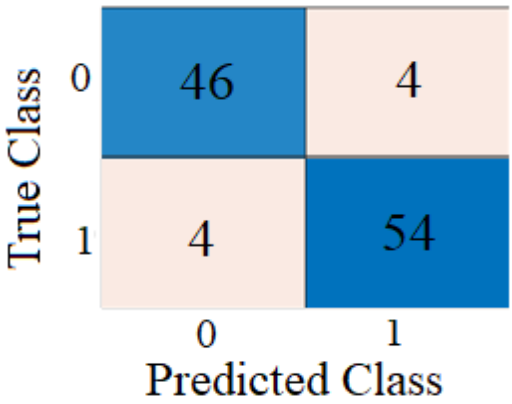

PN09



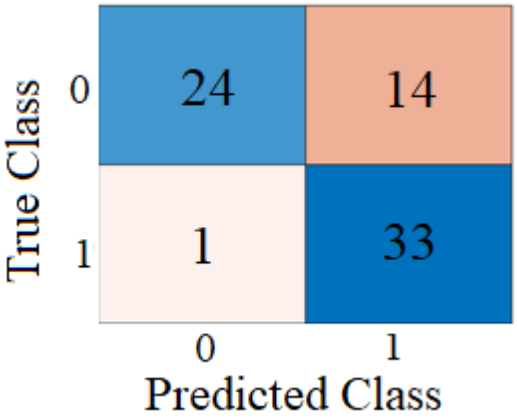

PN10



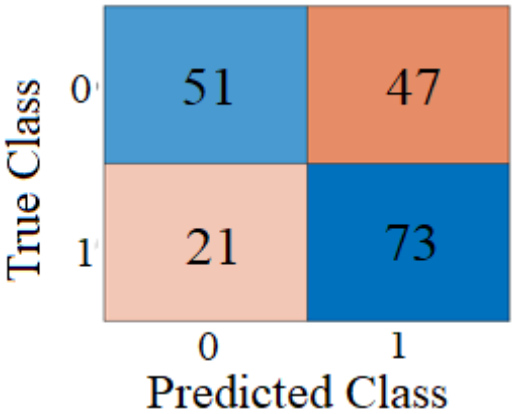

PN11




PN12

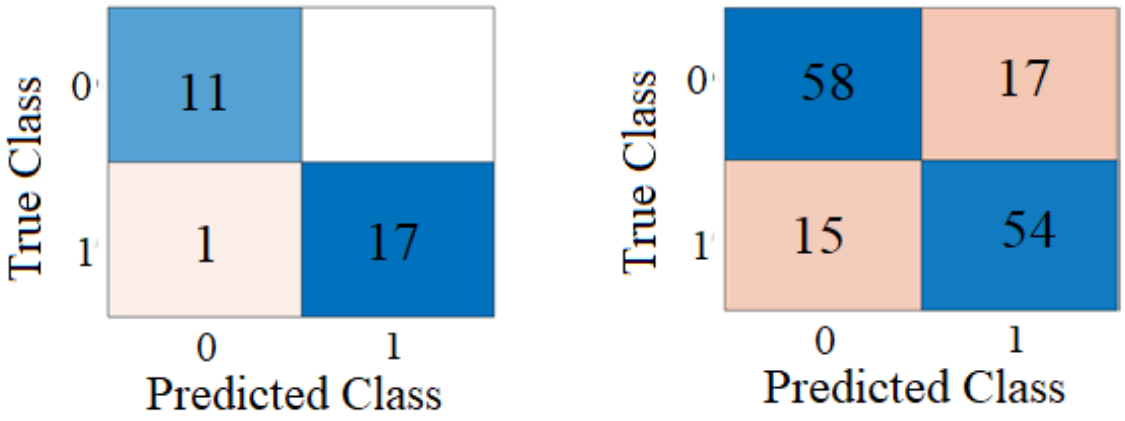



PN13




PN14

Figure 4.12 (continued)




PN16 PN17

Figure 4.12 (continued)

## 4.5 Results of Proposed Real-Time Seizure Prediction Algorithm

In this section, the prediction results of the real-time seizure prediction algorithm are presented. Figure 4.13 shows the power spectral change through time in all eight frequency bands, it's clear that there are rapid changes in power spectral in some of the bands at time stamps between 80-100. In figure 4.14 these changes are detected by raising an alarm for each band. The decision-making unit parameters are chosen as: the number of effective bands = 2 and the maximum time among bands = 60 sec, with this configuration the algorithm predicted 3 of 5 seizures presented in the recorded data of patient ID PN00 with FPR = 0.

The parameters are changed for each patient in the Siena scalp EEG dataset, for example for patient ID Pn06 the number of effective bands = 4 and the maximum time among bands = 2.5 minutes, for patient ID Pn11 the number of effective bands = 5 and the maximum time among band = 6 minutes. The spectral changes and prediction alarms for these patients are shown in Figures 4.15-18.

Table 4.12 summarises the perdition results of the real-time algorithm. The number of detected seizures, FPR, and SPH for each patient are tabulated. The average prediction over all patents is 78%, and the average FPR is 0.16.

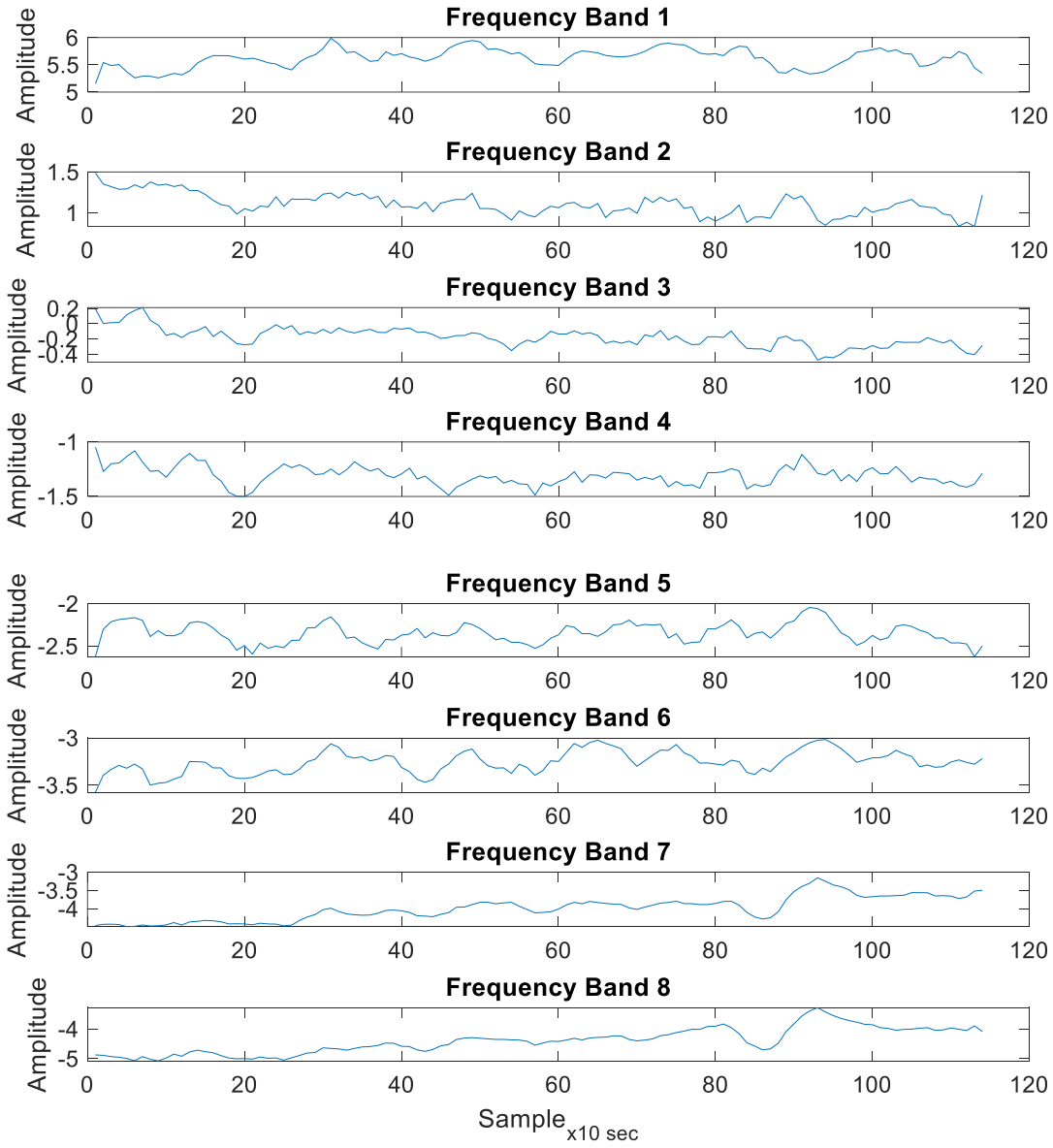



Figure 4.13 Sample 1: The power spectral density of EEG signal bands for patient ID PN00.

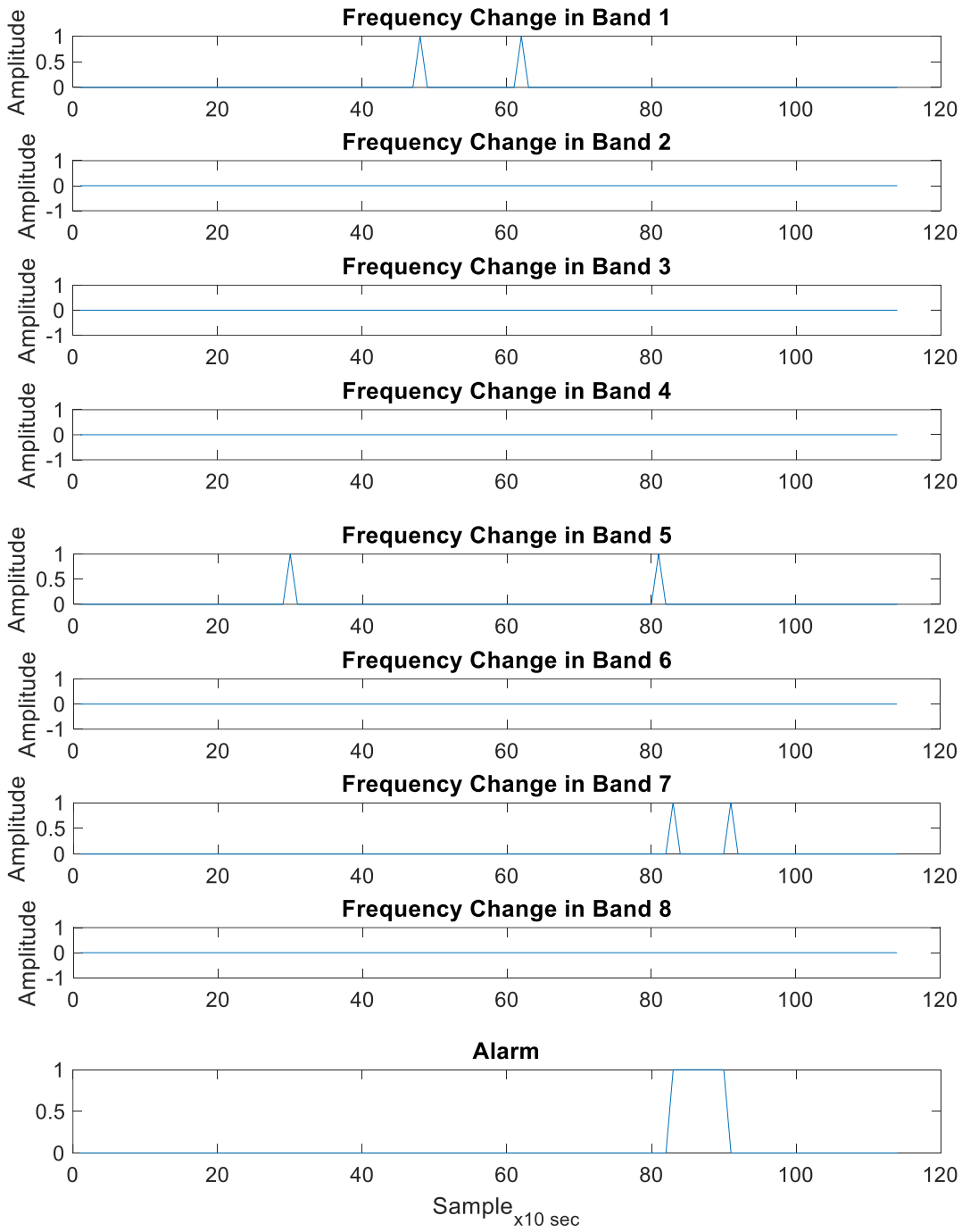



Figure 4.14 Sample 1: The detected changes in frequency bands and prediction alarm for patient ID PN00.

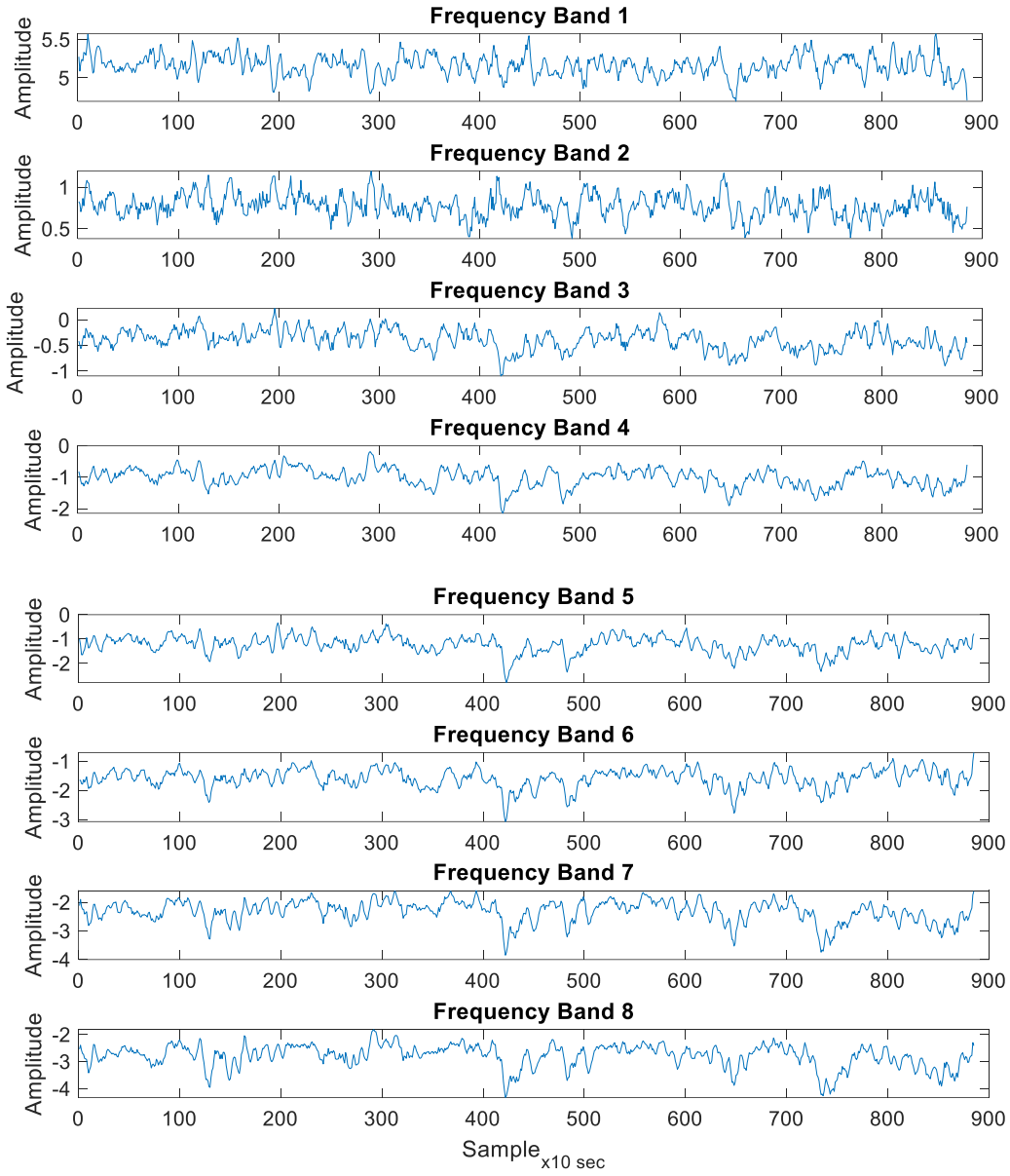



Figure 4.15 Sample 2: The power spectral density of EEG signal bands for patient ID PN06.

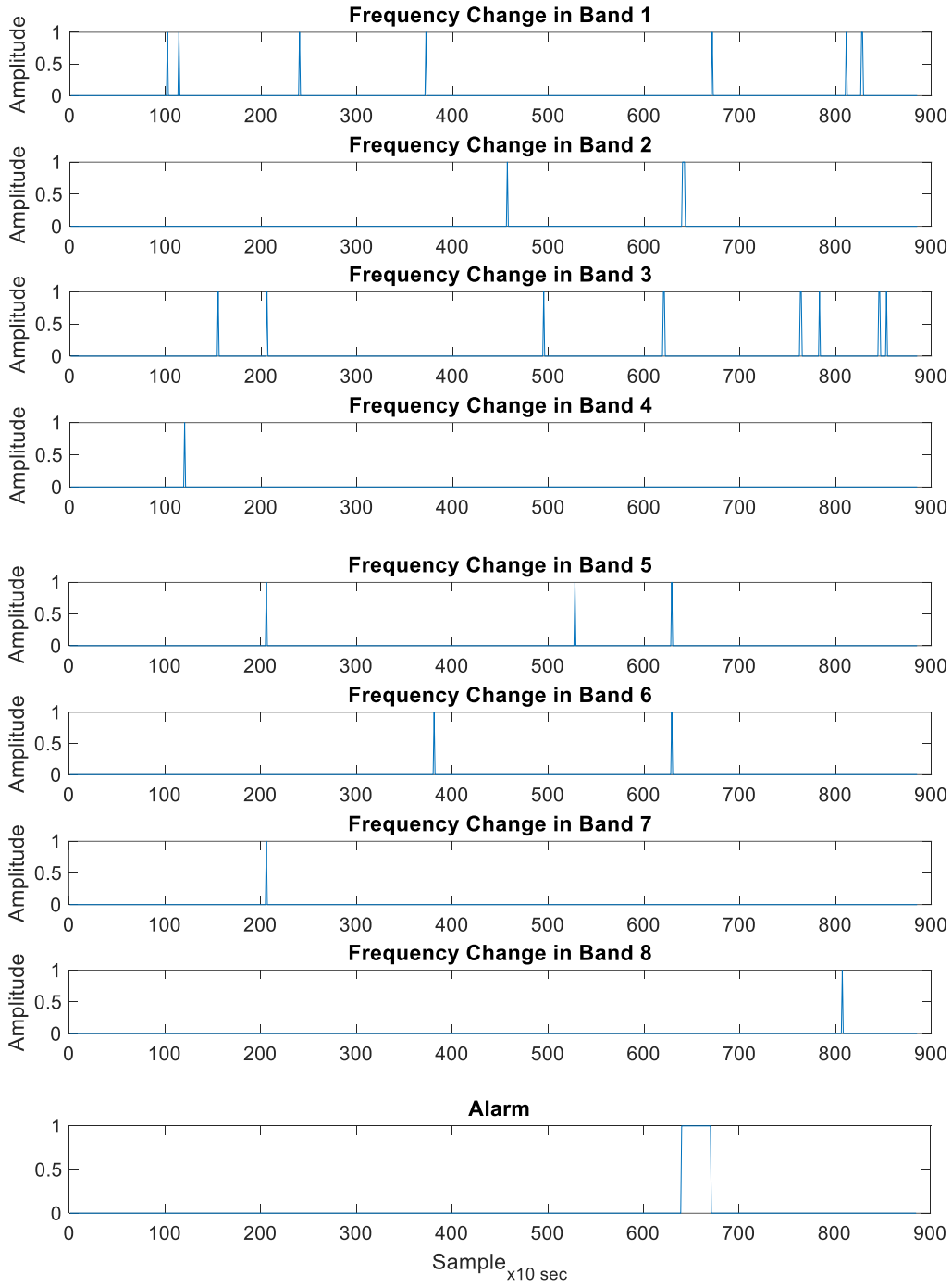



Figure 4.16 Sample 2: The detected changes in frequency bands and prediction alarm for patient ID PN06.

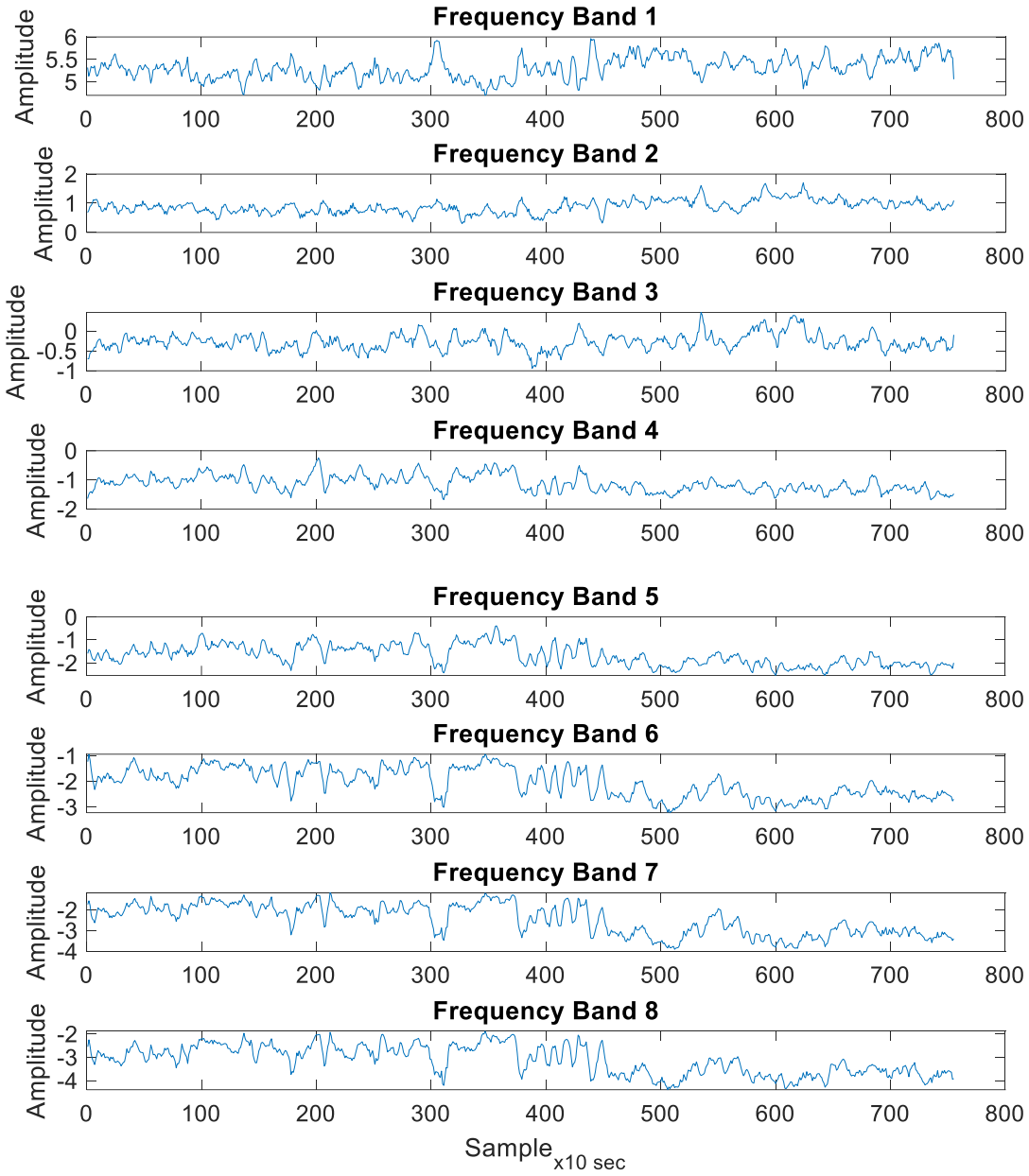



Figure 4.17 Sample 3: The power spectral density of EEG signal bands for patient ID PN11.

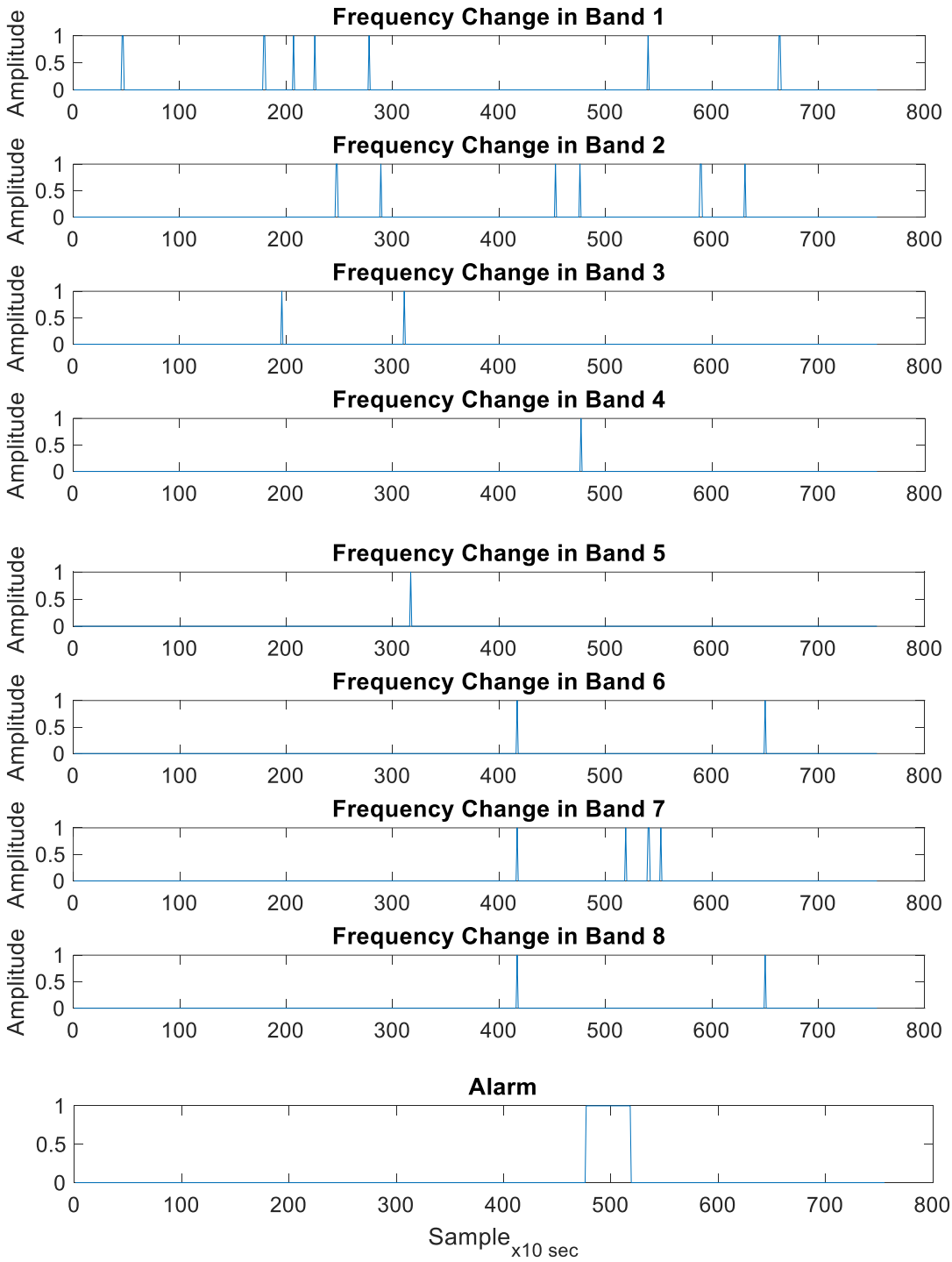



Figure 4.18 Sample 3: The detected changes in frequency bands and prediction alarm for patient ID PN11.

Table 4.12 The perdition results of the real-time algorithm for each patient were obtained with the Siena scalp database.

| Patient ID | Record time minutes | Number of seizures | Number of predicted seizures | FPR/h | SPH minutes |
|---|---|---|---|---|---|
| PN00 | 198 | 5 | 3 | 0 | 5.16 |
| PN01 | 809 | 2 | 1 | 0.15 | 100 |
| PN03 | 752 | 2 | 2 | 0.32 | 157 |
| PN05 | 359 | 3 | 2 | 0.17 | 160.5 |
| PN06 | 722 | 5 | 2 | 0.25 | 28.5 |
| PN07 | 523 | 1 | 1 | 0.23 | 95.3 |
| PN09 | 410 | 3 | 3 | 0.15 | 38.03 |
| PN10 | 1002 | 10 | 9 | 0.48 | 38.3 |
| PN11 | 145 | 1 | 1 | 0 | 46.6 |
| PN12 | 246 | 4 | 3 | 0 | 50.6 |
| PN13 | 519 | 3 | 3 | 0.35 | 40.5 |
| PN14 | 1408 | 4 | 3 | 0 | 64.1 |
| PN16 | 303 | 2 | 2 | 0.2 | 30.3 |
| PN17 | 308 | 2 | 2 | 0 | 40.05 |

## 4.6 Chapter Summary

In section 4.2, an algorithm for EEG signal artefact rejection is presented. One of the most important steps in this algorithm is ICA, which transforms EEG signals into their independent components. A method of independent source modification for noise cancellation is presented based on user identification and signal statistics analysis. The proposed method employs simple calculations of signal statistics besides ICA computation. The performance increases with the user input templates by Manuel mode. Moreover, the more important aspect of the proposed method is that it does not require any extra information channel attached to EEG signals. Visual inspection and reduced variances of reconstructed signals confirm the ability of the algorithm to identify and suppress noise signals effectively.

Section 4.4 presented models for detecting and predicting epileptic seizures using raw EEG data. A model for detecting seizure patterns and classification of raw EEG segments into seizure/non-seizure classes.

Two models for deep learning were used to extract the features of Pre-ictal and Inter-ictal, namely the CNN method and the GRU-LSTM method. The research takes into account the patient's comfort by choosing only five electrodes as an input to the seizure prediction system, as well as reducing the patient's tension and increasing his self-confidence through the use of a short period of alert, which is 10 minutes. By analysing the results, it was found that the accuracy in predicting seizures depends on the nature of the patient's brain signal and on the amount of data recorded for him where the accuracy of the prediction appeared by 50% - 90%.

Section 4.5 presented the results of a seizure prediction algorithm that depends on a short *PIL* period and a single EEG channel only, which gives the patient a smooth workflow and is not restricted to wearing a full EEG helmet, as well as a short *PIL* period that allows the patient to take the necessary precautions without feeling anxious for waiting due to used long *PIL*. Extracting different features for each patient means different required computational capabilities between patients, this means that the required computational capacities for some patients will be few depending on the required features.

Using the proposed method, satisfactory results are obtained with only one EEG channel. A comparison with other research showed that the obtained results are better in terms of the ratio of sensitivity to the number of used EEG channels.

The results of the real-time seizure prediction algorithm in section 4.6 showed the ability of the algorithm to predict 37 of 47 seizures in the Siena scalp EEG dataset with an average FPR of 0.16. Since the algorithm uses a single EEG channel and could be tuned to specific patient's data it makes it a practical prediction algorithm.

# Chapter Five
# Conclusion and Future Works

## 5.1 Conclusions

In this section, the main conclusions from this dissertation are presented. The results obtained from the proposed models gives the following conclusions:

1- The possibility of predicting the seizure using the EEG signals in patients with epilepsy in different proportions depending on the type of EEG signal that the patient's brain emits, as it can be easily predicted in some patients and with difficulty or fundamentally it cannot be expected in others.

2- In deep learning models, it obvious from the results that the CNN model predicts seizures better than GRU-LSTM model.

3- Due to the limitations of disease data, it is preferable to extract features and use machine learning to classify those features to get better results in classification and seizure prediction.

4- Searching for the distinguishing features between Inter-ictal and Pre-ictal periods rises up the prediction rate, since that the distinguishing features differ between the diseased and the healthy condition from one patient to another.

5- The results of the Chaos theory are useful in predicting the seizure in some patients, as the features Lyapanouv Exponent and Approximate Entropy are very distinct between the epileptic and the healthy states in some patients, but it does not give any distinction in others.

6- Comparing between two datasets used in training the models, using Siena scalp dataset which have sampling frequency of 256 Hz gives more robust prediction results than CHB-MIT dataset which have sampling frequency of 128 Hz, this indicates that the transition state into the ictal state becomes more evident with high sampling rate signals.

## 5.2 Future Works

Following are lines of work that could have worthwhile results:

1- The use of cloud computing and the Internet of Things to monitor the patient, as EEG signals are sent to the cloud for analysis, which reduces the size of the device installed on the patient and reduces the amount of energy required to operate it.

2- Designing a special mobile program for addressing EEG signals, where the program is shared with the doctors who label the EEG segments, which enhances the amount of processed data for machine learning.

3- Extracting more features from EEG signals may improve the seizure prediction rate and reach the most influential distinctive features.

4- In the seizure prediction algorithm in section 3.5, the number of features is restricted to 5 and 10, using a variable number of features in features search may produce the best prediction results.

# References


[1] N. Kamel and A. S. Malik, EEG/ERP Analysis Methods and Applications, CRC Press and Taylor & Francis Group, 2015.

[2] S. Sanei and J. A. Chambers, EEG Signal Processing, West Sussex PO19 8SQ, England: John Wiley & Sons Ltd, 2007.

[3] U. Palani, D. Vasanthi and S. R. Begam, "Enhancement of Medical Image Fusion Using Image Processing," *Journal of Innovative Image Processing (JIIP),* vol. 2, no. 4, pp. 165-174, 2020.

[4] L. Deng and D. Yu, "Deep Learning Methods and Applications," *Foundations and Trends in Signal Processing,* vol. 3, no. 4, pp. 197-387, 2014.

[5] L. D. Iasemidis, S. Deng-Shan, C. Wanpracha, S. J. Chris, P. M. Pardalos, J. C. Principe, P. R. Carney, P. Awadhesh, V. Balaji and T. Konstantinos, "Adaptive Epileptic Seizure Prediction System," *IEEE Transactions on Biomedical Engineering,* vol. 50, no. 50, p. 616–627, 2003.

[6] P. Yun, L. Lan, K. K. Parhi and N. Theoden, "Seizure Prediction with Spectral Power of EEG using Cost-Sensitive Support Vector Machines," *Epilepsia,* vol. 52, no. 10, p. 1761–1770, 2011.

[7] T. N. Alotaiby, S. A. Alshebeili, T. Alshawi, I. Ahmad and F. E. A. El-Samie, "EEG seizure detection and prediction algorithms: a survey," *Journal on Advances in Signal Processing,* vol. 1, no. 183, 2014.

[8] A. H. Shoeb, "Application of Machine Learning to Epileptic Seizure Onset Detection and Treatment," 2009. [Online]. Available: http://hdl.handle.net/1721.1/54669. [Accessed 1 10 2021].

[9] H. Lange, J. Lieb, J. Engel and P. Crandall, "Temporo-Spatial Patterns of Pre-ictal Spike Activity in HumanTemporal Lobe Epilepsy," *Electroencephalography and Clinical Neurophysiology,* vol. 56, no. 6, p. 543–555, 1983.

[10] A. Schulze-Bonhage, C. Kurth, A. Carius, B. Steinhoff and T. Mayer, "Seizure anticipation by Patients with Focal and Generalized Epilepsy: A Multicenter Assessment of Premonitory Symptoms," *Epilepsy Research,* vol. 70, no. 1, pp. 83-88, 2006.

[11] J. Dammers, M. Schiek, F. Boers, C. Silex, M. Zvyagintsev, U. Pietrzyk and K. Mathiak, "Integration of Amplitude and Phase Statistics for Complete Artifact Removal in Independent Components of Neuromagnetic Recordings," *IEEE Transactions on Biomedical Engineering,* vol. 55, no. 10, pp. 2353-2362, 2008.

[12] F. Viola, J. Thorne, B. Edmonds, T. Schneider, T. Eichele and S. Debener, "Semi-automatic Identification of Independent Components Representing EEG Artifact," *Clin Neurophysiol,* vol. 120, no. 5, pp. 868-77, 2009.

[13] S. K. Goh, H. A. Abbass, K. C. Tan, A. Al-Mamun, C. Wang and C. Guan, "Automatic EEG Artifact Removal Techniques by Detecting Influential Independent Components," *IEEE Transactions on Emerging Topics in Computational Intelligence,* vol. 1, no. 4, pp. 270-279, 2017.

[14] C.-Y. Chang, S.-H. Hsu, L. Pion-Tonachini and T.-P. Jung, "Evaluation of Artifact Subspace Reconstruction for Automatic EEG Artifact Removal," in *Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC).*, 2018.

[15] K. Yasoda, R. S. Ponmagal2, K. S. Bhuvaneshwari and K. Venkatachalam, "Automatic Detection and Classification of EEG Artifacts using Fuzzy Kernel SVM and Wavelet ICA (WICA)," *Soft Computing,* vol. 24, no. 21, pp. 16011-16019, 2020.

[16] N. Bajaj, J. R. Carrion, F. Bellotti, R. Berta and A. D. Gloria, "Automatic and Tunable Algorithm for EEG ArtifactRemoval using Wavelet Decomposition with Applications in Predictive Modeling during Auditory Tasks," *Biomedical Signal Processing and Control,* vol. 55, 2020.

[17] P. Gajbhiye, N. Mingchinda, W. Chen, S. C. Mukhopadhyay, T. Wilaiprasitporn and R. K. Tripathy, "Wavelet Domain Optimized Savitzky–Golay Filter for the Removal of Motion Artifacts From EEG Recordings," *IEEE Transactions on Instrumentation and Measurement,* vol. 70, pp. 1-11, 2020.

[18] K. Jindal, R. Upadhyay and H. S. Singh, "Application of Hybrid GLCT-PICA De-noising Method in Automated EEG Artifact Removal," *Biomedical Signal Processing and Control,* vol. 60, 2020.

[19] S. Sangmin, K. Lee and G. Kang, "EEG Artifact Removal by Bayesian Deep Learning & ICA," in *2020 42nd Annual International Conference of the IEEE*

*Engineering in Medicine & Biology Society (EMBC)*, Montreal, QC, Canada, 2020.

[20] B. Abdi-Sargezeh, R. Foodeh, V. Shalchyan and M. R. Daliri, "EEG Artifact Rejection by Extracting Spatial and Spatio-spectral Common Components," *Journal of Neuroscience Methods,* vol. 358, 2021.

[21] I. Kiral-Kornek, S. Roy, E. Nurse, B. Mashford, P. Karoly, T. Carroll, D. Payne, S. Saha, S. Baldassano, T. O'Brien, D. Grayden, M. Cook, D. Freestone and S. Harrer, "Epileptic Seizure Prediction Using Big Data and Deep Learning: Toward a Mobile System," *eBioMedicine,* vol. 27, pp. 103-111, 2018.

[22] H. Daoud and M. Bayoumi, "Efficient Epileptic Seizure Prediction based on Deep Learning," *IEEE Transactions on Biomedical Circuits and Systems,* vol. 13, no. 5, pp. 804-813, 2019.

[23] Y. Xu, J. Yang, S. Zhao, H. Wu and M. Sawan, "An End-to-End Deep Learning Approach for Epileptic Seizure Prediction," in *IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, Genova, Italy, 2020.

[24] R. Jana and I. Mukherjee, "Deep Learning based Efficient Epileptic Seizure Prediction with EEG Channel Optimization," *Biomedical Signal Processing and Control,* vol. 68, p. 102767, 2021.

[25] T. Dissanayake, T. Fernando, S. Denman, S. Sridharan and C. Fookes, "Deep Learning for Patient-Independent Epileptic Seizure Prediction Using Scalp EEG Signals," *IEEE SENSORS JOURNAL,* vol. 21, no. 7, 2021.

[26] S. M. Usman, S. Khalid and S. Bashir, "A Deep Learning based Ensemble Learning Method for Epileptic Seizure Prediction," *Computers in Biology and Medicine,* vol. 136, p. 102767, 2021.

[27] D. E. Snyder, J. Echauz, D. B. Grimes and B. Litt, "The Statistics of a Practical Seizure Warning System," *JOURNAL OF NEURAL ENGINEERING,* vol. 5, pp. 392-401, 2008.

[28] W. N and L. MR, "Extracting and Selecting Distinctive EEG Features for Efficient Epileptic Seizure Prediction," *IEEE J Biomed Health Inform,* vol. 19, no. 5, pp. 1648-59, 2015.

[29] S. Elgohary, S. Eldawlatly and M. I. Khalil, "Epileptic Seizure Prediction using Zero-crossings Analysis of EEG Wavelet Detail Coefficients," in *2016*

*IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, Chiang Mai, Thailand, 2016.

[30] T. N. Alotaiby, S. A. Alshebeili, F. M. Alotaibi and S. R. Alrshoud, "Epileptic Seizure Prediction using CSP and LDA for Scalp EEG Signals," *Computational Intelligence and Neuroscience*, vol. 2017, 2017.

[31] H. Chu, C. Chung, W. Jeong and K.-H. Cho, "Predicting Epileptic Seizures from Scalp EEG based on Attractor State Analysis," *Computer Methods and Programs in Biomedicine*, vol. 143, pp. 75-87, 2017.

[32] S. M. Usman, M. Usman and S. Fong, "Epileptic Seizures Prediction Using Machine Learning Methods," *Computational and Mathematical Methods in Medicine*, vol. 2017, 2017.

[33] N. D. Truong, A. D. Nguyen, L. Kuhlmann, M. R. Bonyadi, J. Yang, S. Ippolito and O. Kavehei, "Convolutional Neural Networks for Seizure Prediction using Intracranial and Scalp Electroencephalogram," *Neural Networks*, vol. 105, pp. 104-111, 2018.

[34] H. Khan, L. Marcuse, M. Fields, K. Swann and B. Yener, "Focal Onset Seizure Prediction using Convolutional Networks," *IEEE Transactions on Biomedical Engineering*, vol. 65, no. 9, pp. 2109-2118, 2018.

[35] Tsiouris, V. C. Pezoulas, M. Zervakis, S. Konitsiotis, D. D. Koutsouris and D. I. Fotiadis, "A Long Short-Term Memory Deep Learning Network for The Prediction of Epileptic Seizures using EEG Signals," *Computers in Biology and Medicine*, vol. 2018, pp. 24-37, 2018.

[36] A.-B. AF, V. MF, H. C, B.-O. M and S. S, "Noninvasive Seizure Prediction using Autonomic Measurements in Patients with Refractory Epilepsy," in *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Honolulu, HI, USA, 2018.

[37] H. A. Agboola, C. Solebo, D. S. Aribike, A. E. Lesi and A. A. Susu, "Seizure Prediction with Adaptive Feature Representation Learning," *Journal of Neurology and Neuroscience*, vol. 10, no. 2:294, pp. 1-12, 2019.

[38] P. Detti, G. Vatti and G. Z. M. d. Lara, "EEG Synchronization Analysis for Seizure Prediction: A Study on Data of Noninvasive Recordings," *Processes*, vol. 2020, no. 8, p. 846, 2020.

[39] S. E. Sanchez-Hernandez, R. A. Salido-Ruiz, S. Torres-Ramos and I. Roman-Godinez, "Evaluation of Feature Selection Methods for Classification of Epileptic Seizure EEG Signals," *Sensors,* vol. 2022, no. 22, p. 3066, 2022.

[40] H. Gray, Gray's Anatomy: The Classic Collector's Edition, New York: Random House, 1988.

[41] P. Chen, "Principles of biological science," 2011. [Online]. Available: http://bio1152.nicerweb.com/Locked/src/chap48_g.html. [Accessed 2 2 2022].

[42] N. D. Patel, An EEG-based Dual-channel Imaginary Motion Classification for Brain Computer Interface, Beaumont, TX: Lamar University - Beaumont ProQuest Dissertations Publishing, 2011.

[43] J. Ward, The Student's Guide to Cognitive Neuroscience, London: Taylor and Francis, 2010.

[44] P. L. Nunez and R. Srinivasan, Electric Fields of the Brain: The Neurophysics of EEG, New York: Oxford University Press, 1981.

[45] B. S. Guru and H. R. Hiziroglu, Electromagnetic field theory fundamentals, Cambridge University Press, 2014.

[46] S. Klein and B. M. Thorne, Biological Psychology, New York: Worth, 2006.

[47] J. Fell and N. Axmacher, "The Role of Phase Synchronization in Memory Processes," *Nature Reviews Neuroscience,* vol. 12, p. 105–118, 2011.

[48] A. Schnitzler and J. Gross, "Normal and Pathological Oscillatory Communication in The Brain," *Nature Reviews Neuroscience,* vol. 6, no. 4, p. 285–296, 2005.

[49] P. Laoprasert, Atlas of Pediatric EEG, The McGraw-Hill Companies., 1982.

[50] I. Kaya, "A Brief Summary of EEG Artifact Handling," *Brain-Computer Interface,* 2019.

[51] X. Jiang, G. Bian and Z. Tian, "Removal of Artifacts from EEG Signals: A Review," *Sensors (Basel),* vol. 19, no. 5, p. 987, 2019.

[52] R. He, Y. Liu, K. Wang, N. Zhao, Y. Yuan, Q. Li and H. Zhang, "Automatic Detection of QRS Complexes Using Dual Channels Based on U-Net and Bidirectional Long Short-Term Memory," *IEEE Journal of Biomedical and Health Informatics,* vol. 25, no. 4, pp. 2168-2208, 2021.

[53] P. Comon, "Independent Component Analysis, A New Concept," *Signal Processing*, vol. 36, no. 3, pp. 287-314, 1994.

[54] P. Ablin, J.-F. Cardoso and A. Gramfort, "Faster Independent Component Analysis by Preconditioning with Hessian Approximations," *IEEE Transactions on Signal Processing*, vol. 66, no. 15, p. 4040–4049, 2018.

[55] E. Oja and Z. Yuan, "The FastICA Algorithm Revisited: Convergence Analysis," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, p. 1370 – 138, 2006.

[56] L. Moreno, M. Alem and J. Garc, "Implementation of Infomax ICA Algorithm for Blind Source Separation," in *Electronics, Robotics and Automotive Mechanics Conference*, Cuernavaca, Mexico, 2008.

[57] P. Ablin, J. Cardoso and A. Gramfort, "Faster ICA under Orthogonal Constraint," in *International Conference on Acoustics, Speech, & Signal Processing*, Calgary, AB, Canada, 2018.

[58] A. Hyvarinen and E. Oja, "Independent Component Analysis: Algorithms and Applications," *Neural Networks*, vol. 13, no. 4-5, pp. 411-430, 2000.

[59] N. Ilakiyaselvan, A. N. Khan and A. Shahina, "Deep Learning Approach to Detect Seizure using Reconstructed Phase Space Images," *The Journal of Biomedical Research*, vol. 34, no. 3, p. 240–250, 2020.

[60] C.-Y. Chiang, N.-F. Chang, T.-C. Chen, H.-H. Chen and L.-G. Chen, "Seizure Prediction based on Classification of EEG Synchronization Patterns with On-line Retraining and Post-processing Scheme," in *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Boston, MA, USA, 2011.

[61] M. J. Brodie, S. C. Schachter and P. Kwan, Fast Facts: Epilepsy, Oxford, UK.: Health Press Ltd, 2012.

[62] J. I. S. MD, "Can You Predict Your Own Seizures?," 12 3 2014. [Online]. Available: https://www.epilepsy.com/stories/can-you-predict-your-own-seizures. [Accessed 19 5 2022].

[63] L. D. Iasemidis, D.-S. Shiau, J. C. Sackellares, P. M. Pardalos and A. Prasad, "A Dynamical Resetting of The Human Brain at Epileptic Seizures: Application of Nonlinear Dynamics and Global Optimization Techniques," *IEEE Trans. Biomed. Engng.*, vol. 51, no. 3, p. 493–506, 2004.

[64] L. D. Iasemidis, J. C. Sackellares, H. P. Zaveri and W. J. Willians, "Phase Space Topography and The Lyapunov Exponent of Electrocorticograms in Partial Seizures," *Brain Topography,* vol. 2, p. 187–201, 1990.

[65] Z. Z. Li Hu, EEG Signal Processing and Feature Extraction, Gateway East East, Singapore 189721, Singapore: Springer Nature Singapore Pte Ltd., 2019.

[66] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N.-C. Yen, C. C. Tung and H. H. Liu, "The Empirical Mode Decomposition and the Hilbert Spectrum for Nonlinear and Non-Stationary Time Series Analysis," *Series A: Mathematical, Physical and Engineering Sciences,* vol. 454, no. 1971, pp. 903-95, 1998.

[67] G. Wang, X.-Y. Chen, F.-L. Qiao, Z. Wu and N. E. Huang, "On Intrinsic Mode Function," *Advances in Adaptive Data Analysis,* vol. 2, no. 3, pp. 277-93, 2012.

[68] R. Rato, M. Ortigueira and A. Batista, "On the HHT, Its Problems, and Some Solutions," *Mechanical Systems and Signal Processing,* vol. 22, no. 6, p. 1374–94, 2008.

[69] G. Rilling, P. Flandrin and P. Goncalves, "On Empirical Mode Decomposition and Its Algorithms," in *IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing*, Grado, Italy., 2003.

[70] H. KANTZ and T. SCHREIBER, Nonlinear Time Series Analysis, New York, USA: Cambridge University Press, 2003.

[71] F. Takens, "Detecting Strange Attractors in Turbulence," in *Dynamical Systems and Turbulence*, Warwick, 1980.

[72] X. Jiang and H. Adeli, "Fuzzy Clustering Approach for Accurate Embedding Dimension Identification in Chaotic Time Series," *Integrated Computer-Aided Engineering,* vol. 10, p. 287–302, 2003.

[73] T. Sauer, J. A. Yorke and M. Casdagli, "Embedology," *Journal of Statistical Physics,* vol. 65, no. 3, p. 579–616, 1991.

[74] S. M. PINCUS, "Approximate Entropy as A Measure of System Complexity," *Proceedings of the National Academy of Sciences,* vol. 88, no. 6, pp. 2297-2301, 1991.

[75] P. Grassberger and a. I. Procaccia, "Characterization of Strange Attractors," *Physical Review Letters,* vol. 50, p. 346–349, 1983.

[76] P. Grassberger and I. Procaccia, “Measuring The Strangeness of Strange Attractors,” *Physica D: Nonlinear Phenomena*, vol. 9, p. 189–208, 1983.

[77] A. Wolf, J. B. Swift, H. L. Swinney and J. A. Vastano, “Determining Lyapunov Exponents from A Time Series,” *Physica*, pp. 285-317, 1985.

[78] D. Bertsims and J. Tsitsiklis, “Simulated Annealing,” *Statistical Science,* vol. 8, no. 1, pp. 10-15, 1993.

[79] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, www.deeplearningbook.org, 2016.

[80] N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines and other Kernel-Based Learning Methods, Cambridge University Press, 2000.

[81] S. Abe, Support Vector Machines for Pattern Classification, London: Springer-Verlag London Limited, 2005.

[82] P. S. MANN, Introductory Statistics, JOHN WILEY & SONS, INC, 2012.

[83] J. Ward, The Student's Guide to Cognitive Neuroscience, Oxon: Routledge, 2020.

[84] J. M. Zurada, Introduction to artificial neural systems, New York: WEST PUBLISHING COMPANY, 1992.

[85] B. Zhang, W. Wang, Y. Xiao, S. Xiao, S. Chen, S. Chen, G. Xu and W. Che, “Cross-Subject Seizure Detection in EEGs Using Deep Transfer Learning,” *Computational and Mathematical Methods in Medicine,* vol. 2020, 2020.

[86] Y.-W. Chen and L. C. Jain, Deep Learning in Healthcare Paradigms and Applications, 1 ed., Switzerland AG, Switzerland: Springer Nature, 2020.

[87] F. A. Gers, N. N. Schraudolph and J. Schmidhuber, “Learning Precise Timing with LSTM Recurrent Networks,” *Journal of Machine Learning Research*, vol. 3, pp. 115-143, 2002.

[88] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, p. 1735–1780, 1997.

[89] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” in *International conference on machine learning*, Lille, France, 2015.

[90] J. Brownlee, Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions, Machine Learning Mastery, 2020.

[91] P. Swami, M. Bhatia, M. Tripathi, o. S. Chandra, B. K. Panigrahi and T. K. Gandhi, "Selection of optimum frequency bands for detection of epileptiform patterns," *Healthc Technol Lett.*, vol. 6, no. 5, p. 126–131, 2019.

[92] S. U. Amin, G. Muhammad, W. Abdul, M. Bencherif and M. Alsulaiman, "Multi-CNN Feature Fusion for Efficient EEG Classification," in *IEEE International Conf. On Multimedia & Expo Workshops (ICMEW)*, London, UK, 2020.

[93] K. G. Hartmann, R. T. Schirrmeister and T. Ball, "Hierarchical Internal Representation of Spectral Features in Deep Convolutional Networks Trained for EEG Decoding," in *6th International Conference on Brain-Computer Interface (BCI)*, Gangwon, Korea (South), 2018.

[94] X. Zheng, W. Chen, Y. You, Y. Jiang, M. Li and T. Zhang, "Ensemble Deep learning for Automated Visual Classification using EEG Signals," *Pattern Recognition*, vol. 102, p. 107147, 2020.

[95] E. Q. Wu, P.-Y. Deng, X.-Y. Qiu, Z. Tang, W.-M. Zhang, L.-M. Zhu, H. Ren, G.-R. Zhou and R. S. F. Sheng, "Detecting Fatigue Status of Pilots Based on Deep Learning Network Using EEG Signals," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 13, no. 3, pp. 575 - 585, 2019.

[96] H. Chen, Y. Song and X. Li, "A Deep Learning Framework for Identifying Children with ADHD using an EEG-based Brain Network," *Neurocomputing*, vol. 356, no. 3, pp. 83-96, 2019.

[97] M. Alhussein, G. Muhammad and M. S. Hossain, "EEG Pathology Detection Based on Deep Learning," *IEEE Access*, vol. 7, pp. 27781-27788, 2019.

[98] R. T. Schirrmeister, L. Gemein, K. Eggensperger, F. Hutter and T. Ball, "Deep Learning with Convolutional Neural Networks for Decoding and Visualization of EEG Pathology," in *IEEE International Conf. On Signal Processing in Medicine and Biology Symposium (SPMB)*, Philadelphia, PA, USA, 2017.

[99] M. Shams and A. Sagheer, "A Natural Evolution Optimization based Deep Learning Algorithm for Neurological Disorder Classification," *Bio-Medical Materials and Engineering*, vol. 31, no. 2, pp. 37-94, 2020.

[100] X. Tian, Z. Deng, W. Ying, K.-S. a. W. D. Choi, B. Qin, J. Wang, H. Shen and S. Wang, "Deep Multi-View Feature Learning for EEG-Based Epileptic

Seizure Detection," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27, no. 10, pp. 1962-1972, 2019.

[101] G. P. F. a. P. G. Rilling, "On Empirical Mode Decomposition and Its Algorithms," in *IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing*, Grado, Italy., 2003.

# Appendix A

## MATLAB Codes for Seizure Prediction Algorithm

In this appendix MATLAB codes for implementing the proposed seizure prediction algorithm are presented.

```
clearvars
clc
tr1=72;
tr2=78;
%te=9;

PIL_size = 20; % in minutes SPH
seg_size = 10; % in seconds
opt_ch= 22;

pt_name = "chb23";
paht_name = "E:\eeg data\chb-mit\";

data_files = [["chb01\chb01_03", 2996, 3036];
        ["chb01\chb01_04", 1467, 1494];
        ["chb01\chb01_15", 1732, 1772];
        ["chb01\chb01_16", 1015, 1066];
        ["chb01\chb01_18", 1720, 1810];
        ["chb01\chb01_21", 327, 420];
        ["chb01\chb01_26", 1862, 1963];
        ["chb02\chb02_16", 130, 212];
        ["chb02\chb02_16+", 2972, 3053];
        ["chb02\chb02_19", 3369, 3378];
        ["chb03\chb03_01", 362, 414];
        ["chb03\chb03_02", 731, 796];
        ["chb03\chb03_03", 432, 501];
        ["chb03\chb03_04", 2162, 2214];
        ["chb03\chb03_34", 1982, 2029];
        ["chb03\chb03_35", 2592, 2656];
        ["chb03\chb03_36", 1725, 1778];
        ["chb04\chb04_05", 7804, 7853];
        ["chb04\chb04_08", 6446, 6557];
        ["chb04\chb04_28", 1679, 1781];
```

```
["chb04\chb04_28", 3782, 3898];
["chb05\chb05_06", 417, 532];
["chb05\chb05_13", 1086, 1196];
["chb05\chb05_16", 2317, 2413];
["chb05\chb05_17", 2451, 2571];
["chb05\chb05_22", 2348, 2465];
["chb06\chb06_01", 1724, 1738];
["chb06\chb06_01", 7461, 7476];
["chb06\chb06_01", 13525, 13540];
["chb06\chb06_04", 327, 347];
["chb06\chb06_04", 6211, 6231];
["chb06\chb06_09", 12500, 12516];
["chb06\chb06_10", 10833, 10845];
["chb06\chb06_13", 506, 519];
["chb06\chb06_18", 7799, 7811];
["chb06\chb06_24", 9387, 9403];
["chb07\chb07_12", 4920, 5006];
["chb07\chb07_13", 3285, 3381];
["chb07\chb07_19", 13688, 13831];
["chb08\chb08_02", 2670, 2841];
["chb08\chb08_05", 2856, 3046];
["chb08\chb08_11", 2988, 3122];
["chb08\chb08_13", 2417, 2577];
["chb08\chb08_21", 2083, 2347];
["chb09\chb09_06", 12231, 12295];
["chb09\chb09_08", 2951, 3030];
["chb09\chb09_08", 9196, 9267];
["chb09\chb09_19", 5299, 5361];
["chb10\chb10_12", 6313, 6348];
["chb10\chb10_20", 6888, 6958];
["chb10\chb10_27", 2382, 2447];
["chb10\chb10_30", 3021, 3079];
["chb10\chb10_31", 3801, 3877];
["chb10\chb10_38", 4618, 4707];
["chb10\chb10_89", 1383, 1437];
["chb18\chb18_29", 3477, 3527];
["chb18\chb18_30", 541, 571];
["chb18\chb18_31", 2087, 2155];
["chb18\chb18_32", 1908, 1963];
```

```
["chb18\chb18_35", 2196, 2264];
["chb18\chb18_36", 463, 509];
["chb20\chb20_12", 94, 123];
["chb20\chb20_13", 1440, 1470];
["chb20\chb20_13", 2498, 2537];
["chb20\chb20_14", 1971, 2009];
["chb20\chb20_15", 390, 425];
["chb20\chb20_15", 1689, 1738];
["chb20\chb20_16", 2226, 2261];
["chb20\chb20_68", 1393, 1432];
["chb22\chb22_20", 3367, 3425];
["chb22\chb22_25", 3139, 3213];
["chb22\chb22_38", 1263, 1335];
["chb23\chb23_06", 3962, 4075];
["chb23\chb23_08", 325, 345];
["chb23\chb23_08", 5104, 5151];
["chb23\chb23_09", 2589, 2660];
["chb23\chb23_09", 6885, 6947];
["chb23\chb23_09", 8505, 8532];
["chb23\chb23_09", 9580, 9664];
["chb24\chb24_01", 480, 505];
["chb24\chb24_01", 2451, 2476];
["chb24\chb24_03", 231, 260];
["chb24\chb24_03", 2883, 2908];
["chb24\chb24_04", 1088, 1120];
["chb24\chb24_04", 1411, 1438];
["chb24\chb24_04", 1745, 1764];
["chb24\chb24_06", 1229, 1253];
["chb24\chb24_07", 38, 60];
["chb24\chb24_09", 1745, 1764];
["chb24\chb24_11", 3527, 3597];
["chb24\chb24_13", 3288, 3304];
["chb24\chb24_14", 1939, 1966];
["chb24\chb24_15", 3552, 3569];
["chb24\chb24_17", 3515, 3581];
["chb24\chb24_21", 2804, 2872]
];
```

```
data_files2 = [["chb01\chb01_05", 2996, 3036];
        ["chb01\chb01_06", 1467, 1494];
        ["chb01\chb01_07", 1732, 1772];
        ["chb01\chb01_30", 1015, 1066];
        ["chb01\chb01_31", 1720, 1810];
        %["chb01\chb01_32", 327, 420];
        ["chb01\chb01_33", 1862, 1963];
        ["chb02\chb02_16", 130, 212];
        ["chb02\chb02_16+", 2972, 3053];
        ["chb02\chb02_19", 3369, 3378];
        ["chb03\chb03_01", 362, 414];
        ["chb03\chb03_02", 731, 796];
        ["chb03\chb03_03", 432, 501];
        ["chb03\chb03_04", 2162, 2214];
        ["chb03\chb03_34", 1982, 2029];
        ["chb03\chb03_35", 2592, 2656];
        ["chb03\chb03_36", 1725, 1778];
        ["chb04\chb04_05", 7804, 7853];
        ["chb04\chb04_08", 6446, 6557];
        ["chb04\chb04_28", 1679, 1781];
        ["chb04\chb04_28", 3782, 3898];
        ["chb05\chb05_06", 417, 532];
        ["chb05\chb05_13", 1086, 1196];
        ["chb05\chb05_16", 2317, 2413];
        ["chb05\chb05_17", 2451, 2571];
        ["chb05\chb05_22", 2348, 2465];
        ["chb06\chb06_01", 1724, 1738];
        ["chb06\chb06_01", 7461, 7476];
        ["chb06\chb06_01", 13525, 13540];
        ["chb06\chb06_04", 327, 347];
        ["chb06\chb06_04", 6211, 6231];
        ["chb06\chb06_09", 12500, 12516];
        ["chb06\chb06_10", 10833, 10845];
        ["chb06\chb06_13", 506, 519];
        ["chb06\chb06_18", 7799, 7811];
        ["chb06\chb06_24", 9387, 9403];
        ["chb07\chb07_12", 4920, 5006];
        ["chb07\chb07_13", 3285, 3381];
```

```
["chb07\chb07_19", 13688, 13831];
["chb08\chb08_12", 2670, 2841];
["chb08\chb08_03", 2856, 3046];
["chb08\chb08_15", 2988, 3122];
["chb08\chb08_19", 2417, 2577];
["chb08\chb08_16", 2083, 2347];
["chb09\chb09_06", 12231, 12295];
["chb09\chb09_08", 2951, 3030];
["chb09\chb09_08", 9196, 9267];
["chb09\chb09_19", 5299, 5361];
["chb10\chb10_12", 6313, 6348];
["chb10\chb10_20", 6888, 6958];
["chb10\chb10_27", 2382, 2447];
["chb10\chb10_30", 3021, 3079];
["chb10\chb10_31", 3801, 3877];
["chb10\chb10_38", 4618, 4707];
["chb10\chb10_89", 1383, 1437];
["chb18\chb18_29", 3477, 3527];
["chb18\chb18_30", 541, 571];
["chb18\chb18_31", 2087, 2155];
["chb18\chb18_32", 1908, 1963];
["chb18\chb18_35", 2196, 2264];
["chb18\chb18_36", 463, 509];
["chb20\chb20_12", 94, 123];
["chb20\chb20_13", 1440, 1470];
["chb20\chb20_13", 2498, 2537];
["chb20\chb20_14", 1971, 2009];
["chb20\chb20_15", 390, 425];
["chb20\chb20_15", 1689, 1738];
["chb20\chb20_16", 2226, 2261];
["chb20\chb20_68", 1393, 1432];
["chb22\chb22_20", 3367, 3425];
["chb22\chb22_25", 3139, 3213];
["chb22\chb22_38", 1263, 1335];
["chb23\chb23_06", 3962, 4075];
["chb23\chb23_08", 325, 345];
["chb23\chb23_08", 5104, 5151];
["chb23\chb23_09", 2589, 2660];
["chb23\chb23_09", 6885, 6947];
```

```
            ["chb23\chb23_09", 8505, 8532];
            ["chb23\chb23_09", 9580, 9664];
            ["chb24\chb24_01", 480, 505];
            ["chb24\chb24_01", 2451, 2476];
            ["chb24\chb24_03", 231, 260];
            ["chb24\chb24_03", 2883, 2908];
            ["chb24\chb24_04", 1088, 1120];
            ["chb24\chb24_04", 1411, 1438];
            ["chb24\chb24_04", 1745, 1764];
            ["chb24\chb24_06", 1229, 1253];
            ["chb24\chb24_07", 38, 60];
            ["chb24\chb24_09", 1745, 1764];
            ["chb24\chb24_11", 3527, 3597];
            ["chb24\chb24_13", 3288, 3304];
            ["chb24\chb24_14", 1939, 1966];
            ["chb24\chb24_15", 3552, 3569];
            ["chb24\chb24_17", 3515, 3581];
            ["chb24\chb24_21", 2804, 2872];
            ];
fs = 256;
tau =10;
dim = 12;
ch_index = opt_ch;

PIL_size = PIL_size *60*fs;

seg_size = seg_size * fs;

x_set= [];
y_set = [];
x_label = [];
y_label = [];

for i=tr1:tr2
   file_name2 = data_files2(i,1);
   file_name = data_files(i,1);
   sez_start = str2double(data_files(i,2)) * fs;   %in seconds
   sez_end = str2double(data_files(i,3))  * fs;     %in seconds
   [hdr, record]=edfread(paht_name + file_name+".edf");
```

```
        [hdr2, record2]=edfread(paht_name + file_name2+".edf");
        N2 = hdr2.records * hdr2.samples(1);

    inter_ictal=[];
    seg_shift=0;
    try %inter_ictal = record2(ch_index,1:sez_start  - PIL_size- (10*60*fs));
        %inter_ictal = record2(ch_index,1:PIL_size+1);
        pri_ictal = record(ch_index,sez_start  - PIL_size:sez_start);
        %pri_ictal=rmoutliers(pri_ictal);
        for seg_num=1:(N2/seg_size)-100 %PIL_size/seg_size     %chnge this
            seg_shift = seg_shift + seg_size*1; %*5
            inter_ictal_temp
=record2(ch_index,1+seg_shift:1+seg_shift+(seg_size));
            %inter_ictal_temp =rmoutliers(inter_ictal_temp);
            inter_ictal = [inter_ictal, inter_ictal_temp ];
        end
    catch
        fprintf("no rsults");
        continue;
    end

    for seg=1:round((size(inter_ictal,2)-1)/seg_size)-5  %PIL_size/seg_size %
                x_featuer_set(1) = lyapunovExponent(inter_ictal(seg_size*seg-
seg_size+1:(seg_size*seg-seg_size+1)+seg_size-1),fs,tau,dim);
                x_featuer_set(2) = approximateEntropy(inter_ictal(seg_size*seg-
seg_size+1:(seg_size*seg-seg_size+1)+seg_size-1),tau,dim);
                x_featuer_set(3) = skewness(inter_ictal(seg_size*seg-
seg_size+1:(seg_size*seg-seg_size+1)+seg_size-1));
                x_featuer_set(4) = std(inter_ictal(seg_size*seg-
seg_size+1:(seg_size*seg-seg_size+1)+seg_size-1));
                x_featuer_set(5) = kurtosis(inter_ictal(seg_size*seg-
seg_size+1:(seg_size*seg-seg_size+1)+seg_size-1));
                x_featuer_set(6) = correlationDimension(inter_ictal(seg_size*seg-
seg_size+1:(seg_size*seg-seg_size+1)+seg_size-1));
                x_stft=stft(inter_ictal(seg_size*seg-seg_size+1:(seg_size*seg-
seg_size+1)+seg_size-1),fs,'OverlapLength',10,'FrequencyRange',"onesided");
                x_stft_abs = abs(x_stft);
```

```
[x_imf,x_residual,x_info] = emd(inter_ictal(seg_size*seg-
seg_size+1:(seg_size*seg-seg_size+1)+seg_size-
1),'Interpolation','pchip','MaxNumIMF',8,'Display',0);

x_featuer_set(7) =  x_info.NumExtrema(1);
x_featuer_set(8) =  x_info.NumExtrema(2);
x_featuer_set(9) =  x_info.NumExtrema(3);
x_featuer_set(10) =  x_info.NumExtrema(4);
x_featuer_set(11) =  x_info.NumExtrema(5);
x_featuer_set(12) =  x_info.NumExtrema(6);

x_featuer_set(13) =  mean(periodogram(inter_ictal(seg_size*seg-
seg_size+1:(seg_size*seg-seg_size+1)+seg_size-1)));
x_featuer_set(14) =  mean(xcorr(inter_ictal(seg_size*seg-
seg_size+1:(seg_size*seg-seg_size+1)+seg_size-1)));

x_featuer_set(15) =  mean(mean(x_stft_abs(1:3,:)));

x_featuer_set(16) =  x_info.NumZerocrossing(1);
x_featuer_set(17) =  x_info.NumZerocrossing(2);
x_featuer_set(18) =  x_info.NumZerocrossing(3);
x_featuer_set(19) =  x_info.NumZerocrossing(4);
x_featuer_set(20) =  x_info.NumZerocrossing(5);
x_featuer_set(21) =  x_info.NumZerocrossing(6);


x_featuer_set(24) = mean(mean(x_stft_abs(4:7,:)));

x_featuer_set(25) =  x_info.NumSifting(1);
x_featuer_set(26) =  x_info.NumSifting(2);
x_featuer_set(27) =  x_info.NumSifting(3);
x_featuer_set(28) =  x_info.NumSifting(4);
x_featuer_set(29) =  x_info.NumSifting(5);
x_featuer_set(30) =  x_info.NumSifting(6);

x_featuer_set(33) =  mean(mean(x_stft_abs(8:15,:)));
x_featuer_set(34) =  x_info.MeanEnvelopeEnergy(1);
x_featuer_set(35) =  x_info.MeanEnvelopeEnergy(2);
x_featuer_set(36) =  x_info.MeanEnvelopeEnergy(3);
```

```
            x_featuer_set(37) =  x_info.MeanEnvelopeEnergy(4);
            x_featuer_set(38) =  x_info.MeanEnvelopeEnergy(5);
            x_featuer_set(39) =  x_info.MeanEnvelopeEnergy(6);


           x_featuer_set(42) =  mean(mean(x_stft_abs(16:31,:)));
            x_featuer_set(43) =  x_info.RelativeTolerance(1);
            x_featuer_set(44) =  x_info.RelativeTolerance(2);
            x_featuer_set(45) =  x_info.RelativeTolerance(3);
            x_featuer_set(46) =  x_info.RelativeTolerance(4);
            x_featuer_set(47) =  x_info.RelativeTolerance(5);
            x_featuer_set(48) =  x_info.RelativeTolerance(6);

            x_featuer_set(51) =  mean(mean(x_stft_abs(32:65,:)));
            x_featuer_set(52) = mean(x_residual);
            x_featuer_set(53) = std(x_residual);
            x_featuer_set(54) =  mean(x_imf(:,1));
            x_featuer_set(55) =  std(x_imf(:,1));
            x_featuer_set(56) =  mean(x_imf(:,2));
            x_featuer_set(57) =  std(x_imf(:,2));
            x_featuer_set(58) =  mean(x_imf(:,3));
            x_featuer_set(59) =  std(x_imf(:,3));
            x_featuer_set(60) =  mean(x_imf(:,4));
            x_featuer_set(61) =  std(x_imf(:,4));
            x_featuer_set(62) =  mean(x_imf(:,5));
            x_featuer_set(63) =  std(x_imf(:,5));
            x_featuer_set(64) =  mean(x_imf(:,6));
            x_featuer_set(65) =  std(x_imf(:,6));


            x_set= [x_set; x_featuer_set ];
            x_label = [x_label;0];

end


for seg=1:PIL_size/seg_size
```

```
y_featuer_set(01)= lyapunovExponent(pri_ictal(seg_size*seg-
seg_size+1:(seg_size*seg-seg_size+1)+seg_size-1),fs,tau,dim);
y_featuer_set(02)  = approximateEntropy(pri_ictal(seg_size*seg-
seg_size+1:(seg_size*seg-seg_size+1)+seg_size-1),tau,dim);
y_featuer_set(03) = skewness(pri_ictal(seg_size*seg-
seg_size+1:(seg_size*seg-seg_size+1)+seg_size-1));
y_featuer_set(04) = std(pri_ictal(seg_size*seg-
seg_size+1:(seg_size*seg-seg_size+1)+seg_size-1));
y_featuer_set(05)= kurtosis(pri_ictal(seg_size*seg-
seg_size+1:(seg_size*seg-seg_size+1)+seg_size-1));
y_featuer_set(06)= correlationDimension(pri_ictal(seg_size*seg-
seg_size+1:(seg_size*seg-seg_size+1)+seg_size-1));
y_stft=stft(pri_ictal(seg_size*seg-seg_size+1:(seg_size*seg-
seg_size+1)+seg_size-1),fs,'OverlapLength',10,'FrequencyRange',"onesided");
y_stft_abs = abs(y_stft);
[y_imf,y_residual,y_info] = emd(pri_ictal(seg_size*seg-
seg_size+1:(seg_size*seg-seg_size+1)+seg_size-
1),'Interpolation','pchip','MaxNumIMF',8,'Display',0);

y_featuer_set(07) =  y_info.NumExtrema(1);
y_featuer_set(08) =  y_info.NumExtrema(2);
y_featuer_set(09) =  y_info.NumExtrema(3);
y_featuer_set(10) =  y_info.NumExtrema(4);
y_featuer_set(11) =  y_info.NumExtrema(5);
y_featuer_set(12) =  y_info.NumExtrema(6);


y_featuer_set(13) =  mean(periodogram(pri_ictal(seg_size*seg-
seg_size+1:(seg_size*seg-seg_size+1)+seg_size-1)));

y_featuer_set(14) =  mean(xcorr(pri_ictal(seg_size*seg-
seg_size+1:(seg_size*seg-seg_size+1)+seg_size-1)));

y_featuer_set(15) =  mean(mean(y_stft_abs(1:3,:)));

y_featuer_set(16) =  y_info.NumZerocrossing(1);
y_featuer_set(17) =  y_info.NumZerocrossing(2);
y_featuer_set(18) =  y_info.NumZerocrossing(3);
y_featuer_set(19) =  y_info.NumZerocrossing(4);
```

```
y_featuer_set(20) =  y_info.NumZerocrossing(5);
y_featuer_set(21) =  y_info.NumZerocrossing(6);


y_featuer_set(24) =  mean(mean(y_stft_abs(4:7,:)));

y_featuer_set(25) =  y_info.NumSifting(1);
y_featuer_set(26) =  y_info.NumSifting(2);
y_featuer_set(27) =  y_info.NumSifting(3);
y_featuer_set(28) =  y_info.NumSifting(4);
y_featuer_set(29) =  y_info.NumSifting(5);
y_featuer_set(30) =  y_info.NumSifting(6);
y_featuer_set(33) =  mean(mean(y_stft_abs(8:15,:)));

y_featuer_set(34) =  y_info.MeanEnvelopeEnergy(1);
y_featuer_set(35) =  y_info.MeanEnvelopeEnergy(2);
y_featuer_set(36) =  y_info.MeanEnvelopeEnergy(3);
y_featuer_set(37) =  y_info.MeanEnvelopeEnergy(4);
y_featuer_set(38) =  y_info.MeanEnvelopeEnergy(5);
y_featuer_set(39) =  y_info.MeanEnvelopeEnergy(6);

y_featuer_set(42) =  mean(mean(y_stft_abs(16:31,:)));

y_featuer_set(43) =  y_info.RelativeTolerance(1);
y_featuer_set(44) =  y_info.RelativeTolerance(2);
y_featuer_set(45) =  y_info.RelativeTolerance(3);
y_featuer_set(46) =  y_info.RelativeTolerance(4);
y_featuer_set(47) =  y_info.RelativeTolerance(5);
y_featuer_set(48) =  y_info.RelativeTolerance(6);

y_featuer_set(51) =  mean(mean(y_stft_abs(32:65,:)));


y_featuer_set(52) = mean(y_residual);
y_featuer_set(53) = std(y_residual);

y_featuer_set(54) =  mean(y_imf(:,1));
y_featuer_set(55) =  std(y_imf(:,1));
y_featuer_set(56) =  mean(y_imf(:,2));
```

```
            y_featuer_set(57) =  std(y_imf(:,2));
            y_featuer_set(58) =  mean(y_imf(:,3));
            y_featuer_set(59) =  std(y_imf(:,3));
            y_featuer_set(60) =  mean(y_imf(:,4));
            y_featuer_set(61) =  std(y_imf(:,4));
            y_featuer_set(62) =  mean(y_imf(:,5));
            y_featuer_set(63) =  std(y_imf(:,5));
            y_featuer_set(64) =  mean(y_imf(:,6));
            y_featuer_set(65) =  std(y_imf(:,6));


            y_set = [y_set; y_featuer_set];
            y_label = [y_label;1];
    end

end


    [h,p,ci,stat] = ttest2(y_set,x_set,'Vartype','unequal');


    [~,featureIdxSortbyP] = sort(p,2); % sort the features

    selected_fatures= featureIdxSortbyP .* (p(featureIdxSortbyP)<0.05);

    fatures_idx = selected_fatures(selected_fatures~=0);
    xx=x_set(:,fatures_idx);
    yy=y_set(:,fatures_idx);

    xx( :, all( ~any( xx ), 1 ) ) = [];
    yy( :, all( ~any( yy ), 1 ) ) = [];  %delete all zero cols

    data_X = [xx;yy];

    data_Y = [x_label;y_label];
     [train_idx, ~, test_idx] = dividerand(size(data_X,1), 0.7, 0,0.3);
        % slice training data with train indexes
        %(take training indexes in all 10 features)
     x_train = data_X(train_idx, :);
```

```
 y_train = data_Y(train_idx, :);
    % select test data
 x_test = data_X(test_idx, :);
 y_test = data_Y(test_idx, :);
%

parent = 1:length(xx(1,:));
origin = parent;
%[minimum, fval] = anneal(sa_cost(train_X, train_Y, test_X, test_Y),
parent, ['Generator', sa_new_sol(train_X, test_X)]);



Tinit = 1;       % initial temp
minT = 1e-8;        % stopping temp
cool = @(T) (.95*T);       % annealing schedule
minF = -Inf;
max_consec_rejections = 1000;
max_try = 300;
max_success = 20;
k = 1;                        % boltzmann constant

% counters etc
itry = 0;
success = 0;
finished = 0;
consec = 0;
T = Tinit;
initenergy = sa_cost(x_train, y_train, x_test, y_test);
oldenergy = initenergy;
total = 0;
all_params = [];

fprintf(1,'\n  T = %7.5f, loss = %10.5f\n',T,oldenergy);

while ~finished
   itry = itry+1; % just an iteration counter
   current = parent;

   % % Stop / decrement T criteria
```

```
        if itry >= max_try || success >= max_success
           if T < minT || consec >= max_consec_rejections
              finished = 1;
              total = total + itry;

              break;
           else
              T = cool(T);  % decrease T according to cooling schedule
               %if report==2 % output
               fprintf(1,'  T = %7.5f, loss = %10.5f\n',T,oldenergy);
               %end
              total = total + itry;
              itry = 1;
              success = 1;
           end
        end

        [newparam,new_train_X,new_test_X] =
sa_new_sol(current,x_train,x_test, origin, T);

        newenergy = sa_cost(new_train_X, y_train, new_test_X, y_test);

        B = [newparam, zeros(1, length(origin) -
length(newparam)),newenergy];
        all_params = [all_params; B];
        if (newenergy < minF)
           parent = newparam;
           oldenergy = newenergy;
           break
        end

        if (oldenergy-newenergy > 1e-6)
           parent = newparam;
           oldenergy = newenergy;
           success = success+1;
           consec = 0;
        else
           if (rand < exp( (oldenergy-newenergy)/(k*T) ))
              parent = newparam;
```

```
            oldenergy = newenergy;
            success = success+1;
        else
            consec = consec+1;
            fprintf(1,'Reject\n')
        end
    end
end

minimum = parent;
fval = oldenergy;

bbb= all_params(:,length(origin)+1);
bbb(bbb==0) =1;

%[mmmm,iiiii]= min(all_params(:,length(origin)+1))
[mmmm,iiiii]= min(bbb);

aaaaa = all_params(iiiii,1:length(origin));
bbbbbb = aaaaa(aaaaa~=0);
my_fe= fatures_idx(bbbbbb);
xx=x_set(:,my_fe);
yy=y_set(:,my_fe);

train_X = [xx;yy];

[train_idx, ~, test_idx] = dividerand(size(train_X,1), 0.7, 0,0.3);
    % slice training data with train indexes
    %(take training indexes in all 10 features)
 x_train = train_X(train_idx, :);
 y_train = data_Y(train_idx, :);
    % select test data
 x_test = train_X(test_idx, :);
 y_test = data_Y(test_idx, :);

opts =
struct('Optimizer','bayesopt','ShowPlots',true,'AcquisitionFunctionName','expect
ed-improvement-plus');
```

```
Mdl =
fitcsvm(x_train,y_train,'KernelFunction','rbf','OptimizeHyperparameters','auto','
HyperparameterOptimizationOptions',opts);

isLabels1 = resubPredict(Mdl);
figure
ConfusionMat1 = confusionchart(y_train,isLabels1);
isGenRate = resubLoss(Mdl,'LossFun','ClassifErr');
figure
yyyy = predict(Mdl,x_test);
ConfusionMat2 = confusionchart(y_test,yyyy);
test_cost = loss(Mdl,x_test,y_test);
isGenRate
test_cost
my_fe
function [test_cost] = sa_cost(train_features, train_labels, test_features,
test_labels)


loss1 = resubLoss(Mdl,'LossFun','ClassifErr');

test_cost = (loss1+loss2)/2;

function [new_sol, new_trainX, new_testX] =
sa_new_sol(input_sol,train_X, test_X, origin,T)
num_req_feats = 10;
idxs = randperm(length(origin),num_req_feats);

sol = origin(idxs);

new_trainX=train_X(:,sol);
new_testX = test_X(:,sol);
new_sol =sol;
```

# Appendix B
# Python codes for GUI program and Noise cancelation algorithm

```
# import libraries
import tkinter as tk
import tkinter.filedialog
from edfreader import EDFreader
import numpy as np
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib.pyplot as plt
import pandas as pd
import mne
from scipy.stats import pearsonr
from remove_noise import pdf

file_path = ""

# function for opening edf files
def open_edf(path):
    hdl = EDFreader(path)
    nst = hdl.getTotalSamples(0)
    n = hdl.getNumSignals()
    ibuf = np.empty(nst, dtype=np.int32)
    datarecord = []
    signalnames = []
    for j in range(0, n):
        signalnames.append(str(hdl.getSignalLabel(j)))
        hdl.rewind(j)
        hdl.readSamples(j, ibuf, hdl.getTotalSamples(j))
        datarecord.append(ibuf.copy())
    hdl.close()
    return datarecord, signalnames

# function for finding list of distance between two segments x,y
def diff(x, y, bands):
    #print(distance.euclidean(x, y))
    dists = []
    for i in range(0, len(x), int(len(x) / bands)):
        xx = np.average(x[i:i + int(len(x) / bands)])
        yy = np.average(y[i:i + int(len(x) / bands)])
        dists.append(distance.euclidean(xx, yy))
    print(dists)
    z = 0
```

```
    for each in dists:
        z += np.power(each, 2)
    z = np.sqrt(z)
    # print(z)
    return z

global raw, ica
d = []
n = []
filterd = ""
filter_order = 10

def new_file():
    mylist.delete(0, tk.END)

# filtering functions
def applylowpassfilter():
    global filterd, d, raw
    raw = raw.filter(l_freq=None, h_freq=float(elowpss.get()),
filter_length='auto')
    filterd = filterd + "+lowpass"
    tk.messagebox.showinfo("information",
                           "Low Pass filter with cutoff frquency " + elowpss.get() + "
Hz is applied to all Channels successfully")


def applyhighpassfilter():
    global filterd, d, raw
    raw = raw.filter(l_freq=float(ehighpass.get()), h_freq=None,
filter_length='auto')
    filterd = filterd + "+highpass"
    tk.messagebox.showinfo("information", "High Pass filter with cutoff
frequency "+ehighpass.get()+" Hz is applied to all Channels successfully")


def applynothpassfilter():
    global filterd, d, raw
    freqs = (60, 120)
    raw = raw.notch_filter(freqs=freqs)
    filterd = filterd + "+notch"
    tk.messagebox.showinfo("information","Notch filter with frequencies
60,120 Hz is applied to all Channels successfully")
```

```
# function for plot signals
def plot_figs(picks, plotstart, plotend, picked_ch):
    fig = raw.plot(start=plotstart, duration=plotend - plotstart,
n_channels=len(picks), title=filterd, nn=picked_ch)
    fig.canvas.key_press_event('a')

# function for segment cropping
def crop_load_event(tmin=0, tmax=9):
    global raw
    # raw.crop(tmin=tmin, tmax=tmax).load_data()
    raw.crop(tmin=tmin, tmax=tmax)
    raw.set_eeg_reference('average', projection=False)
    tk.messagebox.showinfo("information", "Segment cropped and average
montage applied successfully")

# function for opening file with GUI
def openfile():
    global file_path, raw
    file_path = tkinter.filedialog.askopenfilename(filetypes=(("EDF Files",
"*.edf"), ("All Files", "*.*")))
    raw = mne.io.read_raw_edf(file_path)
    titel_var.set('Hussein Project - '+file_path)
    m.title(titel_var.get())
    # raw.anonymize()
    print("mes date", raw.info['meas_date'])
    print("sampling fequeny", raw.info['sfreq'])
    # print(raw.info.ch_names)
    for line in range(len(raw.info.ch_names)):
        mylist.insert(tk.END, raw.info.ch_names[line])
    # epochs = mne.make_fixed_length_epochs(raw, duration=3, preload=False)
    # epochs.plot(n_epochs=10)
    print(raw.times[-1])
    my_annot = mne.Annotations(onset=[0],  # in seconds
                               duration=[raw.times[-1]],  # in seconds, too
                               description=['Normal'])
    print(my_annot)
    raw.load_data()


def listclickEvent(event):
    global picks, picked_ch
    seleted = mylist.curselection()
    # print(seleted)
    picked_ch = []
```

```
    for i in seleted:
        picked_ch.append(raw.info.ch_names[i])
    # print(picked_ch)
    picks = mne.pick_channels(raw.ch_names, include=picked_ch)
    # print("piks=", picks)


def plot_event():
    plotstart = int(e1.get())
    plotend = int(e2.get())
    plot_figs(picks, plotstart, plotend, picked_ch)


# function for scroll frame signals
class VerticalScrolledFrame(tk.Frame):
    def __init__(self, parent, *args, **kw):
        tk.Frame.__init__(self, parent, *args, **kw)
        # create a canvas object and a vertical scrollbar for scrolling it
        vscrollbar = tk.Scrollbar(self, orient=tk.VERTICAL)
        vscrollbar.pack(fill=tk.Y, side=tk.RIGHT, expand=tk.FALSE)
        canvas = tk.Canvas(self, bd=0, highlightthickness=0, height=400,
                           yscrollcommand=vscrollbar.set)
        canvas.pack(side=tk.LEFT, fill=tk.BOTH, expand=tk.TRUE)
        vscrollbar.config(command=canvas.yview)
        # reset the view
        canvas.xview_moveto(0)
        canvas.yview_moveto(0)
        # create a frame inside the canvas which will be scrolled with it
        self.interior = interior = tk.Frame(canvas)
        interior_id = canvas.create_window(0, 0, window=interior,
                                           anchor=tk.NW)
        # track changes to the canvas and frame width and sync them,
        # also updating the scrollbar
        def _configure_interior(event):
            # update the scrollbars to match the size of the inner frame
            size = (interior.winfo_reqwidth(), interior.winfo_reqheight())
            canvas.config(scrollregion="0 0 %s %s" % size)
            if interior.winfo_reqwidth() != canvas.winfo_width():
                # update the canvas's width to fit the inner frame
                canvas.config(width=interior.winfo_reqwidth())
        interior.bind('<Configure>', _configure_interior)
        def _configure_canvas(event):
            if interior.winfo_reqwidth() != canvas.winfo_width():
                # update the inner frame's width to fill the canvas
```

```
            canvas.itemconfigure(interior_id, width=canvas.winfo_width())
        canvas.bind('<Configure>', _configure_canvas)


# function down-sampling signals
def down_sample():
    global raw
    current_sfreq = raw.info['sfreq']
    # print(current_sfreq)
    desired_sfreq = 100  # Hz
    decim = np.round(current_sfreq / desired_sfreq).astype(int)
    obtained_sfreq = current_sfreq / decim
    # print(obtained_sfreq)
    raw = raw.resample(sfreq=obtained_sfreq)
    current_sfreq = raw.info['sfreq']
    print("New sampling frequency" + current_sfreq)


def auto_correlation():
    pass


def triangular(x, a, b, c):
    return max(min((x - a) / (b - a), (c - x) / (c - b)), 0)


# U variable represents the Universe of Discourse of numerical variable height

U = np.arange(0, 700, .5)  # generates a sequence of number between 0 and 10, step 0.2

height = {
    'Very Small': [0, 0.1, 100],
    'Small': [.1, 100, 200],
    'very short': [100, 200, 300],
    'Short': [200, 300, 400],
    'Medium': [300, 400, 500],
    'High': [400, 500, 600],
    'Very High': [500, 600, 700],
    'Very Very High': [600, 700, 700.1]
}


def convert_epochs():
```

```
    global raw
    new_events = mne.make_fixed_length_events(raw, id=1, start=0, duration=3)
    events_from_annot, event_dict = mne.events_from_annotations(raw)
    print(event_dict)
    print(events_from_annot[:5])
    epochs = mne.Epochs(raw, events_from_annot, tmin=-0.3, tmax=3,
event_id=event_dict, preload=True)
    print(epochs)
    epochs.plot(events=events_from_annot, event_id=event_dict)

    # epochs = mne.make_fixed_length_epochs(raw, duration=30,
preload=False)
    # epochs.plot()

# GUI initilization
m = tk.Tk()  # where m is the name of the main window object
# add
titel_var = tk.StringVar()
titel_var.set('Hussein Project')
m.title(titel_var.get())
menu = tk.Menu(m)
m.config(menu=menu)
filemenu = tk.Menu(menu)
menu.add_cascade(label='File', menu=filemenu)
filemenu.add_command(label='New', command=new_file)
filemenu.add_command(label='Open...', command=openfile)
filemenu.add_separator()
filemenu.add_command(label='Exit', command=m.destroy)

filtermenu = tk.Menu(menu)
menu.add_cascade(label='Filter', menu=filtermenu)
filtermenu.add_command(label='Low pass filter',
command=applylowpassfilter)
filtermenu.add_command(label='High pass filter',
command=applyhighpassfilter)
filtermenu.add_separator()
filtermenu.add_command(label='Notch filter', command=applynothpassfilter)
filtermenu.add_command(label='Down sample', command=down_sample)
filtermenu.add_command(label='Auto correlation',
command=auto_correlation)

trainrmenu = tk.Menu(menu)
menu.add_cascade(label='Train', menu=trainrmenu)
trainrmenu.add_command(label='Train model with this file', command=None)
```

```
helpmenu = tk.Menu(menu)
menu.add_cascade(label='Help', menu=helpmenu)
helpmenu.add_command(label='About')

scrollbar = tk.Scrollbar(m)
scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
mylist = tk.Listbox(m, yscrollcommand=scrollbar.set,
selectmode=tk.MULTIPLE, width=20, height=10)

mylist.pack(side=tk.LEFT, fill=tk.BOTH)

scrollbar.config(command=mylist.yview)
mylist.bind('<<ListboxSelect>>', listclickEvent)

# to crop signals
label_crop1 = tk.Label(m, text="Crop Start in sec:").pack()
cropstart = tk.StringVar()
e_crop1 = tk.Entry(m)
e_crop1.pack()
e_crop1.insert(10, "0")
label_crop2 = tk.Label(m, text="Crop End in sec:").pack()
cropend = tk.StringVar()
e_crop2 = tk.Entry(m)
e_crop2.pack()
e_crop2.insert(10, "9")

# function for crop and load EEG data
def crop_load_e():
    crop_load_event(int(e_crop1.get()), int(e_crop2.get()))


button = tk.Button(m, text='Crop', width=25, command=crop_load_e)
button.pack()

# function for Re-load all EEG data
def Reload_raw():
    global raw
    raw = mne.io.read_raw_edf(file_path)
    raw.load_data()
    tk.messagebox.showinfo("information", "Channel data are re-loaded
successfully")

button = tk.Button(m, text='Reload data', width=25, command=Reload_raw)
```

```
button.pack()

# to plot signals
label1 = tk.Label(m, text="Plot Start in sec:").pack()
plotstart = tk.StringVar()
e1 = tk.Entry(m)
e1.pack()
e1.insert(10, "0")
label2 = tk.Label(m, text="Plot End in sec:").pack()
plotend = tk.StringVar()
e2 = tk.Entry(m)
e2.pack()
e2.insert(10, "3")
button = tk.Button(m, text='plot', width=25, command=plot_event)
button.pack()

# function for saving processed signals in a text file plot signals
def save_for_deep_learning():
    global raw,  file_path, raw_dat_file, reonst_dat_file

    for i in mylist.curselection():
        # raw_dat_file = "e:\\results2\\"+ file_path[-10:-5] +
raw.info.ch_names[i]+ str(time.time()) + "_raw.dat"
        deep_dat_file = "e:\\Deep EEG\\" + targets[int(v.get())] + "\\" +
file_path[-10:-4] + "-CH" + \
                        raw.info.ch_names[i] + "-S" + str(e_crop1.get()) + "-E" +
str(e_crop2.get()) + "-" + targets[
                            int(v.get())] + ".dat"
        np.savetxt(deep_dat_file, raw._data[i][:], delimiter=',')

    tk.messagebox.showinfo("information", "Channel data and targets are
saved in directory successfully")


labelhighpass = tk.Label(m, text="High pass cutoff Hz:").pack()
highpasscutoff = tk.StringVar()
ehighpass = tk.Entry(m)
ehighpass.pack()
ehighpass.insert(10, "1")
labellowpass = tk.Label(m, text="Low pass cutoff Hz:").pack()
lowpsscutoff = tk.StringVar()
elowpss = tk.Entry(m)
elowpss.pack()
elowpss.insert(10, "70")
```

```
button = tk.Button(m, text='Low pass filter', width=25,
command=applylowpassfilter)
button.pack()

button = tk.Button(m, text='High pass filter', width=25,
command=applyhighpassfilter)
button.pack()

button = tk.Button(m, text='Notch filter', width=25,
command=applynothpassfilter)
button.pack()

from remove_noise import remove_noise, _invrse_transform, _transform,
whitening

# function for fit ICA
def fit_ica(plotstart=plotstart, plotend=plotend):
    global raw, ica, ica_sources, ica_ch_names, ica_sources_EDF, aaaaaaaa,
bbbbbbbb, ica2, data, aa, aa_, ica_sources_, raw_csv_file_name, snrs1, var1,
reconstructed_raw
    # set up and fit the ICA
    ica = mne.preprocessing.ICA(n_components=23, random_state=97,
max_iter=2000,
                                method='fastica')
    ica.fit(raw)
    plotstart = int(e1.get())
    plotend = int(e2.get())
    ica.plot_sources(raw, show_scrollbars=True, show=True, title="Select Noise
signals", start=plotstart, stop=plotend)
    ica_sources_ = ica.get_sources(raw)
    data = raw._data.copy()
    raw_csv_file_name = "e:\\results2\\" + file_path[-10:-5] + str(time.time()) +
"_raw.csv"
    #f = open(raw_csv_file_name, 'w', newline='')
    snrs1 = []
    var1 = []
    for i in range(0, 22):
        #print("snr values for  raw data", snr_cal(np.asarray(data[i][:])))
        snrs1.append(str(scipy.stats.entropy(pdf(np.asarray(data[i][:])))))
        var1.append(np.var(np.asarray(data[i][:])))

    ica_sources = ica_sources_._data
    pca_data = np.dot(np.linalg.inv(ica.unmixing_matrix_), ica_sources)
```

```
    data = np.dot(np.linalg.inv(ica.pca_components_), pca_data)
    data *= ica.pre_whitener_

    reconstructed_raw = data
    aa_ = mne.io.RawArray(reconstructed_raw, raw.info, first_samp=0,
copy='auto', verbose=None)
    # ica_sources_EDF.info['ch_names'] = ['ICA%03d' % ii for ii in range(0,
22)]
    aa_.plot(start=0, duration=3, n_channels=23, title="Test for reconstructed
signals",
             nn=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22])
    raw.plot(start=0, duration=3, n_channels=23, title="raw",
             nn=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22])


global z
import time, os, scipy, csv
from scipy.spatial import distance
from scipy.fft import rfft, irfft, rfftfreq

# function for applying ICA and de-noising signals
def apply_ica(mode='Manual'):
    global ica_ch_names, reconstructed_raw, reconstructed_raw_EDF,
noise_data, reconst_csv_file_name, snrs2, var2
    if mode == 'Manual':
        nWindow.focus_force()
    elif mode == 'Auto':
        # ch_indexes = [int(index[3:]) for index in ica_ch_names]
        for ch in range(23):  # ch_indexes:
            for templ_filename in os.listdir("E:\\ICA templet\\" + str(ch) + "\\"):
                noise_data = np.loadtxt("E:\\ICA templet\\" + str(ch) + "\\" +
templ_filename, delimiter=',')

                pure_signal = remove_noise(np.asarray(ica_sources[int(ch)][:]),
noise_data)

                ica_sources[int(ch)][:] = pure_signal

        pca_data = np.dot(np.linalg.inv(ica.unmixing_matrix_), ica_sources)
        data = np.dot(np.linalg.inv(ica.pca_components_), pca_data)
        data *= ica.pre_whitener_
```

```
        reconstructed_raw = data

        reconstructed_raw_EDF = mne.io.RawArray(reconstructed_raw, raw.info,
first_samp=0, copy='auto', verbose=None)

        # reconstructed_raw= mne.io.RawArray(reconstructed_raw_, raw.info,
first_samp=0, copy='auto', verbose=None)

        # print(reconstructed_raw)
        reconst_csv_file_name = "e:\\results2\\" + file_path[-10:-5] +
str(time.time()) + "_reconst.csv"
        #f = open(reconst_csv_file_name, 'w', newline='')
        snrs2 = []
        var2 =[]
        print("Time to gain results:----------------------------------------------------
------")
        print("raw results")
        for ri in range(0, 22):
            print(scipy.stats.describe(raw._data[ri]))
        print("reconstructed results:")
        for ri in range(0, 22):
            print(scipy.stats.describe(reconstructed_raw_EDF._data[ri]))

        for i in range(0, 22):
            #print("snr values for reconstructed raw data",
snr_cal(np.asarray(reconstructed_raw[i][:])))

snrs2.append(str(scipy.stats.entropy(pdf(np.asarray(reconstructed_raw[i][:])))))
            var2.append(np.var(np.asarray(reconstructed_raw[i][:])))
            #axbox.boxplot(reconstructed_raw[i][:])
            #axbox.boxplot(raw._data[i][:])

        snrs = np.asarray([snrs1, snrs2])
        snrs = np.transpose(snrs)
        df = pd.DataFrame(snrs)

        f= "e:\\results2\\snrs.csv"
        df.to_csv(f, index=False,mode='a',header=False)

        vars = np.asarray([var1, var2])
        vars = np.transpose(vars)
        df2 = pd.DataFrame(vars)

        f = "e:\\results2\\vars.csv"
```

```
        df2.to_csv(f, index=False, mode='a',header=False)
        #np.savetxt(f, np.ndarray([snrs1, snrs2]), delimiter='1')


global ica_1_out

# function for plotting reconstructed signals
def plot_reonst():
    global raw, ica_1_out, reconstructed_raw, picks, pickd_ch,
reconstructed_raw_EDF
    # .plot(start=0, duration=3, n_channels=23, title="Reconstruvted RAW",
    #                   nn=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19, 20, 21, 22])
    fig2 = reconstructed_raw_EDF.plot(start=0, duration=3,
n_channels=len(picks), title="Reconstructd", nn=picked_ch)
    fig2.canvas.key_press_event('a')

button = tk.Button(m, text='Fit ICA', width=25, command=fit_ica)
button.pack()

button = tk.Button(m, text='Remove Noise Manually', width=25,
command=apply_ica)
button.pack()


def apply_ica_auto():
    apply_ica(mode='Auto')

def save_features():
    global reconstructed_raw
    figbox, axbox = plt.subplots()
    # ax.plot(dists, label="dists")
    axbox.boxplot(reconstructed_raw[0][:])#, label="reconst")
    axbox.boxplot(raw._data[0][:])#, label="raw")

    axbox.legend()
    figbox.show()

def save_long_ch_data():
    global raw
    for i in mylist.curselection():
        f = "e:\\nf data\\long ch\\" + file_path[-10:-4] + "-CH-" +
raw.info.ch_names[i] + ".dat"
        np.savetxt(f, raw._data[i][:], delimiter=',')
```

```
    tk.messagebox.showinfo("information", "Channel data is saved in
directory successfully")

button = tk.Button(m, text='Remove Noise Automatically', width=25,
command=apply_ica_auto)
button.pack()

button = tk.Button(m, text='Plot Reconstructed Raw', width=25,
command=plot_reonst)
button.pack()

def save_reconstructed_channals_data():
    global raw, reconstructed_raw, file_path,raw_dat_file,reonst_dat_file

    for i in mylist.curselection():
        #raw_dat_file = "e:\\results2\\"+ file_path[-10:-5] +
raw.info.ch_names[i]+ str(time.time()) + "_raw.dat"
        reonst_dat_file = "e:\\nf data\\"+ targets[int(v.get())] + "\\" + file_path[-
10:-4] + "-CH" + raw.info.ch_names[i] + "-S" + str(e_crop1.get())+ "-E" +
str(e_crop2.get()) + "-" + targets[int(v.get())]+".dat"
        #np.savetxt(raw_dat_file, raw._data[i][:], delimiter=',')

        np.savetxt(reonst_dat_file, reconstructed_raw[i][:], delimiter=',')

    tk.messagebox.showinfo("information", "Channel data and targets are
saved in directory successfully")

v = tk.IntVar()
v.set(0)

targets = {
          0: 'Non_Patient',
          1: '15m_Seizure',
          2: '10m_Seizure',
          3: '5m_Seizure',
          4: 'Seizure'
         }

label_tagts = tk.Label(m, text="Select label for current
segment:").pack(anchor=tk.W)
t1 = tk.Radiobutton(m, text='Non-Patient', variable=v, value=0)
t1.pack(anchor=tk.W)
```

```
t2 = tk.Radiobutton(m, text='15m-Seizure', variable=v, value=1)
t2.pack(anchor=tk.W)

t3 = tk.Radiobutton(m, text='10m-Seizure', variable=v, value=2)
t3.pack(anchor=tk.W)

t4 = tk.Radiobutton(m, text='5m-Seizure', variable=v, value=3)
t4.pack(anchor=tk.W)

t5 = tk.Radiobutton(m, text='Seizure', variable=v, value=4)
t5.pack(anchor=tk.W)


button = tk.Button(m, text='Save Reconstructed channals data with Labels',
width=50, command=save_reconstructed_channals_data)
button.pack()

button = tk.Button(m, text='Save long channel data', width=25,
command=save_long_ch_data)
button.pack()

button = tk.Button(m, text='Save Filtered data and Features for Machine
Learning', width=50, command=save_features)
button.pack()

def select_all():
    mylist.select_set(0, tk.END)
    global picks, picked_ch

    seleted = mylist.curselection()
    # print(seleted)
    picked_ch = []
    for i in seleted:
        picked_ch.append(raw.info.ch_names[i])
    # print(picked_ch)
    picks = mne.pick_channels(raw.ch_names, include=picked_ch)



tk.Button(m, text='select all channels', command=select_all).pack()

button = tk.Button(m, text='Save Filtered data and Labels for Deep
Learning', width=50, command=save_for_deep_learning)
```

```
button.pack()

# desktop = VerticalScrolledFrame(m)
# desktop.pack()

def exit_pro():
    m.destroy()
    nWindow.destroy()
    fWindow.destroy()


button = tk.Button(m, text='Exit Program', width=25, command=exit_pro)
button.pack()

# noise remove window

nWindow = tk.Tk()
nWindow.title('Select Noise Signals')
nWindow.geometry("1600x1500")
ch_label = tk.Label(nWindow, text="IC component:")
ch_label.grid(column=0, row=0, sticky=tk.E, padx=5, pady=5)

ic_com_labels = [str(i) for i in range(0, 23)]
ic_com_variable = tk.StringVar(nWindow)
ic_com_variable.set('0')

combo_txt_var = tk.StringVar(nWindow)

combo_ch = tk.OptionMenu(nWindow, ic_com_variable, *ic_com_labels)
combo_ch.grid(column=1, row=0, sticky=tk.W, padx=5, pady=5)

n_start_lb = tk.Label(nWindow, text="Start of Noise signal:")
n_start_lb.grid(column=2, row=1, sticky=tk.E, padx=5, pady=5)

e_s = tk.Entry(nWindow)
e_s.grid(column=3, row=1, sticky=tk.W, padx=5, pady=5)
e_s.insert(10, "0")

n_end_lb = tk.Label(nWindow, text="End of Noise signal:")
n_end_lb.grid(column=2, row=2, sticky=tk.E, padx=5, pady=5)

e_end = tk.Entry(nWindow)
e_end.grid(column=3, row=2, sticky=tk.W, padx=5, pady=5)
e_end.insert(10, "10")
```

```
# function for plotting noise signal
def plot_noise():
    global ica_sources, noise_signal
    time1 = float(e_s.get())
    time2 = float(e_end.get())
    ch = int(ic_com_variable.get())
    print(ch)
    print(time1, time2)

    noise_signal = np.asarray(ica_sources[ch][int(time1 * 256):int(time2 *
256)]).copy()
    print(noise_signal.shape)
    # times = ica_sources[int(time1):int(time2)][ch][1]
    figure = plt.Figure(figsize=(10, 3), dpi=100)
    ax = figure.add_subplot(111)

    ax.plot(noise_signal)

    chart_type = FigureCanvasTkAgg(figure, canvas)
    chart_type.get_tk_widget().grid(column=0, row=3, sticky=None, padx=5,
pady=5)


button_plot_noise = tk.Button(nWindow, text='Plot Noise Signal', width=25,
command=plot_noise)
button_plot_noise.grid(column=4, row=2, sticky=None, padx=5, pady=5)

canvas = tk.Canvas(nWindow)

canvas.grid(column=0, row=3, sticky=tk.E, padx=5, pady=5, columnspan=4)

import scipy.stats

# function for removing noise
def Remove_noise_signal():
    global noise_signal, reconstructed_raw, aaaaaaaa, ica_sources,
reconstructed_raw_EDF, snrs2
    ch = int(ic_com_variable.get())
    aa = np.transpose(ica_sources[ch][:])
    pure_signal = remove_noise(aa, noise_signal)
    ica_sources[ch][:] = pure_signal

    pca_data = np.dot(np.linalg.inv(ica.unmixing_matrix_), ica_sources)
```

```
    data = np.dot(np.linalg.inv(ica.pca_components_), pca_data)
    data *= ica.pre_whitener_
    reconstructed_raw = data
    reconstructed_raw_EDF = mne.io.RawArray(reconstructed_raw, raw.info,
first_samp=0, copy='auto', verbose=None)
    # reconstructed_raw_EDF.plot(start=0, duration=3, n_channels=23,
title="Reconstruvted RAW",
    #   nn=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
22])
    reconst_csv_file_name = "e:\\results2\\" + file_path[-10:-5] +
str(time.time()) + "_reconst.csv"
    #f = open(reconst_csv_file_name, 'w', newline='')
    snrs2 = []
    var2 = []
    for i in range(0, 22):
        #print("snr values for reconstructed raw data",
snr_cal(np.asarray(reconstructed_raw[i][:])))

snrs2.append(str(scipy.stats.entropy(pdf(np.asarray(reconstructed_raw[i][:])))))
        var2.append(np.var(np.asarray(reconstructed_raw[i][:])))

    snrs = np.asarray([snrs1, snrs2])
    snrs = np.transpose(snrs)
    df = pd.DataFrame(snrs)
    f = "e:\\results2\\snrs.csv"
    df.to_csv(f, index=False, mode='a', header=False)

    vars = np.asarray([var1, var2])
    vars = np.transpose(vars)
    df2 = pd.DataFrame(vars)
    f = "e:\\results2\\vars.csv"
    df2.to_csv(f, index=False, mode='a', header=False)
    print("Time to gain results:----------------------------------------------------------
---")
    print("raw results")
    for ri in range(0, 22):
        print(scipy.stats.describe(raw._data[ri]))
    print("reconstructed results:")
    for ri in range(0, 22):
        print(scipy.stats.describe(reconstructed_raw_EDF._data[ri]))


# function for saving noise signal to template
def save_noise_signal():
```

```
    global noise_signal
    ch = int(ic_com_variable.get())
    np.savetxt('E:\\ICA templet\\' + str(ch) + "\\Noise" + str(time.time()) +
'.csv', noise_signal, delimiter=',')


button_save_noise = tk.Button(nWindow, text='Save Noise Signal', width=25,
command=save_noise_signal)
button_save_noise.grid(column=0, row=4, sticky=None, padx=5, pady=5)

button_remove_noise = tk.Button(nWindow, text='Remove Noise Signal',
width=25, command=Remove_noise_signal)
button_remove_noise.grid(column=1, row=4, sticky=None, padx=5, pady=5)

# featurs window
fWindow = tk.Tk()
fWindow.title('Chaos analysis - features extration')
fWindow.geometry("1600x1500")
f_ch_label = tk.Label(fWindow, text="Channel Name:")
f_ch_label.grid(column=0, row=0, sticky=tk.E, padx=5, pady=5)

f_ic_com_labels = [str(i) for i in range(0, 23)]
f_ic_com_variable = tk.StringVar(fWindow)
f_ic_com_variable.set('0')

f_combo_txt_var = tk.StringVar(fWindow)

f_combo_ch = tk.OptionMenu(fWindow, f_ic_com_variable, *f_ic_com_labels)
f_combo_ch.grid(column=1, row=0, sticky=tk.W, padx=5, pady=5)

f_start_lb = tk.Label(fWindow, text="Start segment:")
f_start_lb.grid(column=2, row=1, sticky=tk.E, padx=5, pady=5)

f_e_s = tk.Entry(fWindow)
f_e_s.grid(column=3, row=1, sticky=tk.W, padx=5, pady=5)
f_e_s.insert(10, "0")

f_end_lb = tk.Label(fWindow, text="End segment:")
f_end_lb.grid(column=2, row=2, sticky=tk.E, padx=5, pady=5)

f_e_end = tk.Entry(fWindow)
f_e_end.grid(column=3, row=2, sticky=tk.W, padx=5, pady=5)
f_e_end.insert(10, "10")
```

```
def f_plot_signal():
    global raw, phase_space_input
    time1 = float(f_e_s.get())
    time2 = float(f_e_end.get())
    ch = int(f_ic_com_variable.get())
    #print(ch)
    #print(time1, time2)

    phase_space_input = np.asarray(raw._data[ch][int(time1 * 256):int(time2 *
256)]).copy()

    figure = plt.Figure(figsize=(10, 3), dpi=100)
    ax = figure.add_subplot(111)

    ax.plot(phase_space_input)

    chart_type = FigureCanvasTkAgg(figure, f_canvas)
    chart_type.get_tk_widget().grid(column=0, row=3, sticky=None, padx=5,
pady=5)


f_button_plot = tk.Button(fWindow, text='Plot Signal', width=25,
command=f_plot_signal)
f_button_plot.grid(column=4, row=2, sticky=None, padx=5, pady=5)

f_canvas = tk.Canvas(fWindow)

f_canvas.grid(column=0, row=3, sticky=tk.E, padx=5, pady=5, columnspan=4)

import hu_phasespace
# function for phase space reconstruction
def phase_space():
    global phase_space_input
    LLE = 20
    f_lle2_lb.config(text=str(LLE))

    phs = hu_phasespace.phase_space(phase_space_input, 4,8)

    q1 = phs[0] * np.cos(phs[1])
    q2 = phs[2] * np.cos(phs[3])

    phase_space_output = [q1,q2]
```

```
    figure = plt.Figure(figsize=(10, 3), dpi=100)
    ax = figure.add_subplot(111)
    ax.set_ylim(auto=True)
    ax.set_xlim(auto=True)
    ax.scatter(phase_space_output[0],phase_space_output[1])

    chart_type = FigureCanvasTkAgg(figure, f_canvas2)
    chart_type.get_tk_widget().grid(column=0, row=3, sticky=None, padx=5,
pady=5)

f_button_ly= tk.Button(fWindow, text='Phase Space', width=25,
command=phase_space)
f_button_ly.grid(column=4, row=4, sticky=None, padx=5, pady=5)


f_canvas2 = tk.Canvas(fWindow)

f_canvas2.grid(column=0, row=5, sticky=tk.E, padx=5, pady=5, columnspan=4)

f_lle_lb = tk.Label(fWindow, text="Largest LE:")
f_lle_lb.grid(column=0, row=6, sticky=tk.E, padx=5, pady=5)

f_lle2_lb = tk.Label(fWindow, text="0000")
f_lle2_lb.grid(column=1, row=6, sticky=tk.E, padx=5, pady=5)

nWindow.mainloop()
fWindow.mainloop()

m.mainloop()
```

# Appendix C

# Python codes for Deep Neural Network Models

## Seizure Prediction Model 1

```
import numpy as np
from edfreader import EDFreader
import tensorflow as tf
import matplotlib.pyplot as plt
import keras
import csv
import os
from keras.utils.vis_utils import plot_model
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
from keras.layers.advanced_activations import LeakyReLU
from sklearn.model_selection import train_test_split

physical_devices = tf.config.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(physical_devices[0], enable=True)

seg_size = 15
n_chs = 5
drop_rate = 0.3
fs = 256

input_shape = [n_chs, int(seg_size * 256 / 10), 1]


def init_w(seed):
    return tf.keras.initializers.RandomNormal(mean=0.0, stddev=0.05,
seed=seed)


# prepare model

def rnd_num():
    return np.random.randint(1, 50)


def get_compiled_model(input_shape):
    global model
    # Make net
```

```
    inputs = keras.layers.Input(shape=input_shape)
    x = inputs  # [:, :, :, tf.newaxis]

    x = keras.layers.Conv2D(1024, kernel_size=[3, 7],
kernel_regularizer=tf.keras.regularizers.l1_l2(l1=1e-5, l2=1e-4),
                            padding='same',
                            kernel_initializer=init_w(rnd_num()))(x)
    x = LeakyReLU()(x)

    x = keras.layers.MaxPooling2D([1, 2], [1, 2], padding='same')(x)

    x = keras.layers.Conv2D(712, kernel_size=[3, 7],
kernel_regularizer=tf.keras.regularizers.l1_l2(l1=1e-5, l2=1e-4),
                            padding='same',
                            kernel_initializer=init_w(rnd_num()))(x)
    x = LeakyReLU()(x)
    x = keras.layers.MaxPooling2D([1, 2], [1, 2], padding='same')(x)

    x = keras.layers.Conv2D(512, kernel_size=[3, 7],
kernel_regularizer=tf.keras.regularizers.l1_l2(l1=1e-5, l2=1e-4),
                            padding='same',
                            kernel_initializer=init_w(rnd_num()))(x)
    x = LeakyReLU()(x)

    x = keras.layers.MaxPooling2D([1, 2], [1, 2], padding='same')(x)

    x = keras.layers.Conv2D(315, kernel_size=[3, 7],
kernel_regularizer=tf.keras.regularizers.l1_l2(l1=1e-5, l2=1e-4),
                            padding='same',
                            kernel_initializer=init_w(rnd_num()))(x)
    x = LeakyReLU()(x)
    x = keras.layers.MaxPooling2D([2, 2], [2, 2], padding='same')(x)
    # x = keras.layers.BatchNormalization()(x)
    x = keras.layers.Conv2D(128, kernel_size=[3, 7],
kernel_regularizer=tf.keras.regularizers.l1_l2(l1=1e-5, l2=1e-4),
                            padding='same',
                            kernel_initializer=init_w(rnd_num()))(x)
    x = LeakyReLU()(x)
    x = keras.layers.MaxPooling2D([2, 4], [2, 4], padding='same')(x)
    # x = keras.layers.BatchNormalization()(x)
    x = keras.layers.Conv2D(64, kernel_size=[3, 7],
kernel_regularizer=tf.keras.regularizers.l1_l2(l1=1e-5, l2=1e-4),
                            padding='same',
                            kernel_initializer=init_w(rnd_num()))(x)
```

```
    x = LeakyReLU()(x)
    x = keras.layers.MaxPooling2D([2, 4], [2, 4], padding='same')(x)
    # x = keras.layers.BatchNormalization()(x)

    # x = keras.layers.concatenate([ch1_layer7,ch2_layer7])
    x = keras.layers.Flatten()(x)

    # x = keras.layers.TimeDistributed()(x)
    # x = keras.layers.LSTM(32, activation='tanh', return_sequences=True)(x)

    x = keras.layers.Dense(32, activation='tanh',
kernel_initializer=init_w(rnd_num()))(x)
    # x = keras.layers.BatchNormalization()(x)
    x = keras.layers.Dense(16, activation='tanh',
kernel_initializer=init_w(rnd_num()))(x)
    outputs = keras.layers.Dense(1, activation='sigmoid')(x)
    model = keras.Model(inputs=inputs, outputs=outputs)
    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate=0.0001, decay=1e-5,
epsilon=0.1, clipnorm=1, clipvalue=1),
        # loss=keras.losses.CategoricalCrossentropy(from_logits=True),
        loss=keras.losses.BinaryCrossentropy(from_logits=True),
        # metrics=[keras.metrics.SparseCategoricalAccuracy(),
        metrics=[keras.metrics.BinaryAccuracy()],
    )
    # keras.utils.plot_model(model, "multi_input_and_output_model.png",
show_shapes=True)
    print(model.summary())

    return model


# Prepare a directory to store all the checkpoints.
checkpoint_dir = "./new_age_ckpt"
if not os.path.exists(checkpoint_dir):
    os.makedirs(checkpoint_dir)


def make_or_restore_model(input_shape):
    # Either restore the latest model, or create a fresh one
    # if there is no checkpoint available.
    checkpoints = [checkpoint_dir + "/" + name for name in
os.listdir(checkpoint_dir)]
    if checkpoints:
```

```
        latest_checkpoint = max(checkpoints, key=os.path.getctime)
        print("Restoring from", latest_checkpoint)
        return keras.models.load_model(latest_checkpoint)
    print("Creating a new model")
    model = get_compiled_model(input_shape)
    plot_model(model, to_file='new_age.png', show_shapes=True,
show_layer_names=True)
    return model


# make model
with tf.device("GPU:0"):
    model = make_or_restore_model(input_shape)

# set callbackes for fit
callbacks = [
    # This callback saves a SavedModel every epoch
    # We include the current epoch in the folder name.
    tf.keras.callbacks.CSVLogger("new_age_log.csv", separator=",",
append=True),

    #                       write_images=True),
    keras.callbacks.ModelCheckpoint(
        filepath=checkpoint_dir + "/new_age_ckpt-file.h5",
save_best_only=True,
        monitor="val_loss")
]

# create csv for test results
if not os.path.exists("./new_age_test_results.csv"):
    print("create new file for test results")
    with open('./new_age_test_results.csv', mode='w') as file:
        writer = csv.writer(file, delimiter=',', quotechar='"',
quoting=csv.QUOTE_MINIMAL)
        writer.writerow(['loss', 'acc'])

data_files = [["chb01/chb01_03", 2996, 3036],
              ["chb24/chb24_21", 2804, 2872],
              ]
data_files = np.array(data_files)

print("starting to get data from files")
```

```
def gen():
    for i in range(79, 94):  # data_files.shape[0]-split_data): 54
        sez_start = data_files[i, 1]
        # print(sez_start)
        sez_end = data_files[i, 2]
        # print(sez_end)
        file_name = data_files[i, 0]
        # read file by EDFreader
        path = "E:/eeg data/chb-mit/" + str(file_name) + ".edf"
        print("open file:", path)
        hdl = EDFreader(path)
        nst = hdl.getTotalSamples(0)
        ibuf = np.empty(nst, dtype=np.int32)
        dbuf = np.empty(nst, dtype=np.float_)
        datarecord = []

        for j in range(0, n_chs):

            hdl.readSamples(j, ibuf, hdl.getTotalSamples(j))
            # normalize data between 0 and 255

            ibuf[:] = (255 * (ibuf[:] - np.min(ibuf[:])) / np.ptp(ibuf[:])).astype(int)

            # hdl.readSamples(i, dbuf, hdl.getTotalSamples(i))
            datarecord.append(ibuf.copy())
        hdl.close()

        datarecord = np.array(datarecord)

        data1 = datarecord[:, (int(sez_start) - 40 * 60) * int(fs):(int(sez_start) - 30 *
60) * int(fs)]
        # data1 = data1 / np.linalg.norm(data1)

        data2 = datarecord[:, (int(sez_start) - 10 * 60) * int(fs):(int(sez_start) - 0 *
60) * int(fs)]
        # data2 = data2 / np.linalg.norm(data2)

        data3 = datarecord[:, (int(sez_start) - 0 * 60) * int(fs):(int(sez_end) - 0 *
60) * int(fs)]
        # data3 = data3 / np.linalg.norm(data3)

        data = np.concatenate((data1, data2), axis=1)

        # plt.plot(data[3, :])
```

```
        targets = np.zeros(int(data.shape[1] / 256 / seg_size))
        targets[int(data.shape[1] / 256 / seg_size / 2):] = 1

        data_set_ = []
        for kk in range(0, data.shape[1], seg_size * int(fs)):
            data_set_.append(data[:, kk:kk + seg_size * int(fs)])

        data_set_ = np.array(data_set_)

        data_set = np.zeros([data_set_.shape[0], data_set_.shape[1],
int(data_set_.shape[2] / 10)])
        for i1 in range(0, data_set_.shape[0]):
            for i2 in range(0, data_set_.shape[1]):
                data_set[i1, i2] = data_set_[i1][i2].reshape(-1, 10).mean(axis=1)

        X_train, X_test, y_train, y_test = train_test_split(data_set, targets,
test_size=0.1, shuffle=True)
        X_train, X_val, y_train, y_val = train_test_split(X_train, y_train,
test_size=0.1, shuffle=True)

        BATCH_SIZE = 32
        SHUFFLE_BUFFER_SIZE = 5

        train_dataset = tf.data.Dataset.from_tensor_slices((X_train, y_train))
        val_dataset = tf.data.Dataset.from_tensor_slices((X_val, y_val))
        test_dataset = tf.data.Dataset.from_tensor_slices((X_test, y_test))

        return
train_dataset.shuffle(SHUFFLE_BUFFER_SIZE).batch(BATCH_SIZE), \
               val_dataset.shuffle(SHUFFLE_BUFFER_SIZE).batch(BATCH_SIZE),
\
               test_dataset.shuffle(SHUFFLE_BUFFER_SIZE).batch(BATCH_SIZE)
\

BATCH_SIZE = 32
SHUFFLE_BUFFER_SIZE = 5
print("Training start")
model.fit_generator(gen, epochs=130, verbose=1, callbacks=callbacks,
shuffle=True)

# mask = np.isnan(data)
# data[mask] = np.interp(np.flatnonzero(mask), np.flatnonzero(~mask),
data[~mask])
```

```
y = model.predict(x=train_dataset, batch_size=32)

print(classification_report(y_true=targets, y_pred=y.round()))

cm = confusion_matrix(y_true=targets, y_pred=y.round())

sns.heatmap(cm, annot=True, fmt='g', xticklabels=['inter-ictal', 'pre-ictal'],
yticklabels=['inter-ictal', 'pre-ictal'])

print("starting to get data from files")

for i in range(94, 95):  # data_files.shape[0]-split_data): 54
    sez_start = data_files[i, 1]
    # print(sez_start)
    sez_end = data_files[i, 2]
    # print(sez_end)
    file_name = data_files[i, 0]
    # read file by EDFreader
    path = "E:/eeg data/chb-mit/" + str(file_name) + ".edf"
    print("open file:", path)
    hdl = EDFreader(path)
    nst = hdl.getTotalSamples(0)
    ibuf = np.empty(nst, dtype=np.int32)
    dbuf = np.empty(nst, dtype=np.float_)
    datarecord = []

    for j in range(0, n_chs):

        hdl.readSamples(j, ibuf, hdl.getTotalSamples(j))
        # normalize data between 0 and 255

        ibuf[:] = (255 * (ibuf[:] - np.min(ibuf[:])) / np.ptp(ibuf[:])).astype(int)

        # hdl.readSamples(i, dbuf, hdl.getTotalSamples(i))
        datarecord.append(ibuf.copy())
    hdl.close()

    datarecord = np.array(datarecord)

    data1 = datarecord[:, (int(sez_start) - 40 * 60) * int(fs):(int(sez_start) - 30 *
60) * int(fs)]

    data2 = datarecord[:, (int(sez_start) - 10 * 60) * int(fs):(int(sez_start) - 0 * 60)
```

```
* int(fs)]


    data3 = datarecord[:, (int(sez_start) - 0 * 60) * int(fs):(int(sez_end) - 0 * 60) *
int(fs)]


    data = np.concatenate((data1, data2), axis=1)

    # plt.plot(data[3, :])

    targets = np.zeros(int(data.shape[1] / 256 / seg_size))
    targets[int(data.shape[1] / 256 / seg_size / 2):] = 1

    data_set_ = []
    for kk in range(0, data.shape[1], seg_size * int(fs)):
        data_set_.append(data[:, kk:kk + seg_size * int(fs)])

    data_set_ = np.array(data_set_)

    data_set = np.zeros([data_set_.shape[0], data_set_.shape[1],
int(data_set_.shape[2] / 10)])
    for i1 in range(0, data_set_.shape[0]):
        for i2 in range(0, data_set_.shape[1]):
            data_set[i1, i2] = data_set_[i1][i2].reshape(-1, 10).mean(axis=1)

    BATCH_SIZE = 32
    SHUFFLE_BUFFER_SIZE = 5

    train_dataset = tf.data.Dataset.from_tensor_slices((data_set, targets))

    train_dataset =
train_dataset.shuffle(SHUFFLE_BUFFER_SIZE).batch(BATCH_SIZE)

y = model.predict(x=train_dataset, batch_size=32)

print(classification_report(y_true=targets, y_pred=y.round()))

cm = confusion_matrix(y_true=targets, y_pred=y.round())

sns.heatmap(cm, annot=True, fmt='g', xticklabels=['inter-ictal', 'pre-ictal'],
yticklabels=['inter-ictal', 'pre-ictal'])
```

**Seizure Prediction Model 2**

```
import numpy as np
from edfreader import EDFreader
import tensorflow as tf
import matplotlib.pyplot as plt
import keras
import csv
import os
from keras.utils.vis_utils import plot_model
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
from keras.layers.advanced_activations import LeakyReLU
from sklearn.model_selection import train_test_split

physical_devices = tf.config.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(physical_devices[0], enable=True)

seg_size = 15
n_chs = 5
drop_rate = 0.3
fs = 256
f1=76
f2=91
f_n="chb24"
divide_by = 1
input_shape = [n_chs, int(seg_size * fs / divide_by)]



def init_w(seed):
    return tf.keras.initializers.RandomNormal(mean=0.0, stddev=0.05,
seed=seed)

# prepare model

def rnd_num():
    return np.random.randint(1, 50)


#model 4 with drop out 0.4

def get_compiled_model(input_shape):
    global model
    # Make net
```

```
    drop_rate = 0.4
    inputs = keras.layers.Input(shape=input_shape, name="Input1")
    x = keras.layers.GRU(32, activation='tanh',
kernel_initializer=init_w(rnd_num()), return_sequences=True)(inputs)
    x = keras.layers.LSTM(52, activation='sigmoid',
kernel_initializer=init_w(rnd_num()), return_sequences=True)(x)
    x = keras.layers.GRU(64, activation='tanh',
kernel_initializer=init_w(rnd_num()), return_sequences=True,
                         dropout=drop_rate)(x)
    x = keras.layers.LSTM(100, activation='tanh',
kernel_initializer=init_w(rnd_num()), return_sequences=True,
                          dropout=drop_rate)(x)
    x = keras.layers.LSTM(128, activation='sigmoid',
kernel_initializer=init_w(rnd_num()), return_sequences=True)(x)
    x = keras.layers.GRU(200, activation='tanh',
kernel_initializer=init_w(rnd_num()), return_sequences=True)(inputs)
    x = keras.layers.LSTM(300, activation='tanh',
kernel_initializer=init_w(rnd_num()), return_sequences=True)(x)
    x = keras.layers.GRU(256, activation='tanh',
kernel_initializer=init_w(rnd_num()),return_sequences=True,dropout=drop_rate
)(x)
    x = keras.layers.LSTM(128,
activation='tanh',kernel_initializer=init_w(rnd_num()),
return_sequences=True,dropout=drop_rate)(x)
    x = keras.layers.GRU(64,
activation='sigmoid',kernel_initializer=init_w(rnd_num()),
return_sequences=True)(x)
    x = keras.layers.LSTM(32,
activation='tanh',kernel_initializer=init_w(rnd_num()),
return_sequences=False,dropout=drop_rate)(x)
    x = keras.layers.Flatten()(x)
    x = keras.layers.Dense(m, activation='tanh')(x)
    x = keras.layers.BatchNormalization()(x)
    x = keras.layers.Dense(64, activation='sigmoid')(x)
    x = keras.layers.BatchNormalization()(x)
    x = keras.layers.Dense(8, activation='tanh')(x)
    x = keras.layers.BatchNormalization()(x)
    outputs = keras.layers.Dense(1, activation='sigmoid')(x)
    model = keras.Model(inputs=inputs, outputs=outputs)
    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate=0.0001, decay=1e-5,
epsilon=0.1, clipnorm=1, clipvalue=1),
        # loss=keras.losses.CategoricalCrossentropy(from_logits=True),
        loss=keras.losses.BinaryCrossentropy(from_logits=True),
```

```
        # metrics=[keras.metrics.SparseCategoricalAccuracy(),
        metrics=[keras.metrics.BinaryAccuracy()],
    )
    keras.utils.plot_model(model, "model2.png", show_shapes=True)
    print(model.summary())

    return model


# Prepare a directory to store all the checkpoints.
checkpoint_dir = "./new_age_ckpt"
if not os.path.exists(checkpoint_dir):
    os.makedirs(checkpoint_dir)


def make_or_restore_model(input_shape):
    # Either restore the latest model, or create a fresh one
    # if there is no checkpoint available.
    checkpoints = [checkpoint_dir + "/" + name for name in
os.listdir(checkpoint_dir)]
    if checkpoints:
        latest_checkpoint = max(checkpoints, key=os.path.getctime)
        print("Restoring from", latest_checkpoint)
        return keras.models.load_model(latest_checkpoint)
    print("Creating a new model")
    model = get_compiled_model(input_shape)
    plot_model(model, to_file='new_age.png', show_shapes=True,
show_layer_names=True)
    return model


# make model
with tf.device("GPU:0"):
    model = make_or_restore_model(input_shape)

# set callbackes for fit
callbacks = [
    # This callback saves a SavedModel every epoch
    # We include the current epoch in the folder name.
    tf.keras.callbacks.CSVLogger("new_age_log"+f_n+".csv", separator=",",
append=True),
    # keras.callbacks.EarlyStopping(patience=2, monitor="val_loss",
restore_best_weights=True),
     keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.2,
```

```
        patience=5, min_lr=0.001),
    # keras.callbacks.TensorBoard(log_dir='./march5new_logs',
histogram_freq=1,
    #                             write_graph=True, update_freq="epoch",
    #                             write_images=True),
    keras.callbacks.ModelCheckpoint(
        filepath=checkpoint_dir + "/new_age_ckpt-file.h5",
save_best_only=True,
        monitor="loss")
]

# create csv for test results
if not os.path.exists("./"+f_n+"test_results.csv"):
    print("create new file for test results")
    with open("./"+f_n+"test_results.csv", mode='w') as file:
        writer = csv.writer(file, delimiter=',', quotechar='"',
quoting=csv.QUOTE_MINIMAL)
        writer.writerow(['loss', 'acc',"val_loss","val_acc"])

data_files = [["chb01/chb01_03", 2996, 3036],
            ["chb01/chb01_04", 1467, 1494],
                    ["chb24/chb24_21", 2804, 2872],
            ]
data_files = np.array(data_files)

print("starting to get data from files")

X= np.empty((0,n_chs,int(seg_size * fs/divide_by)))
Y= np.empty((0))

for i in range(f1, f2):  # data_files.shape[0]-split_data): 54
        sez_start = data_files[i, 1]
        # print(sez_start)
        sez_end = data_files[i, 2]
        # print(sez_end)
        file_name = data_files[i, 0]
        # read file by EDFreader
        path = "E:/eeg data/chb-mit/" + str(file_name) + ".edf"
        print("open file:", path)
        hdl = EDFreader(path)
        nst = hdl.getTotalSamples(0)
        ibuf = np.empty(nst, dtype=np.int32)
        dbuf = np.empty(nst, dtype=np.float_)
        datarecord = []
```

```
    for j in [0,3,5,8,12]:#,, n_chs):

        hdl.readSamples(j, ibuf, hdl.getTotalSamples(j))
        # normalize data between 0 and 255

        ibuf[:] = (255 * (ibuf[:] - np.min(ibuf[:])) / np.ptp(ibuf[:])).astype(int)

        # hdl.readSamples(i, dbuf, hdl.getTotalSamples(i))
        datarecord.append(ibuf.copy())
    hdl.close()

    datarecord = np.array(datarecord)

    data1 = datarecord[:, (int(sez_start) - 40 * 60) * int(fs):(int(sez_start) - 30 *
60) * int(fs)]
    # data1 = data1 / np.linalg.norm(data1)

    data2 = datarecord[:, (int(sez_start) - 10 * 60) * int(fs):(int(sez_start) - 1 *
60) * int(fs)]
    # data2 = data2 / np.linalg.norm(data2)

    data3 = datarecord[:, (int(sez_start) - 0 * 60) * int(fs):(int(sez_end) - 0 *
60) * int(fs)]
    # data3 = data3 / np.linalg.norm(data3)

    data = np.concatenate((data1, data2), axis=1)

    # plt.plot(data[3, :])

    targets = np.zeros(int(data.shape[1] / 256 / seg_size))
    targets[int(data.shape[1] / 256 / seg_size / 2):] = 1

    data_set_ = []
    for kk in range(0, data.shape[1], seg_size * int(fs)):
        data_set_.append(data[:, kk:kk + seg_size * int(fs)])

    data_set_ = np.array(data_set_)

    data_set = np.zeros([data_set_.shape[0], data_set_.shape[1],
int(data_set_.shape[2] / divide_by)])
    for i1 in range(0, data_set_.shape[0]):
        for i2 in range(0, data_set_.shape[1]):
            data_set[i1, i2] = data_set_[i1][i2].reshape(-1,
```

```
divide_by).mean(axis=1)


    X = np.concatenate((X,data_set), axis=0)
    Y = np.concatenate((Y,targets), axis=0)



BATCH_SIZE = 32
SHUFFLE_BUFFER_SIZE = 5

X_train, X_val, y_train, y_val = train_test_split(X, Y, test_size=0.3,
shuffle=True)
X_val, X_test, y_val, y_test = train_test_split(X_val, y_val, test_size=0.5,
shuffle=True)

train_dataset =
tf.data.Dataset.from_tensor_slices((X_train,y_train)).batch(BATCH_SIZE)
val_dataset = tf.data.Dataset.from_tensor_slices((X_val,
y_val)).batch(BATCH_SIZE)
test_dataset = tf.data.Dataset.from_tensor_slices((X_test,
y_test)).batch(BATCH_SIZE)


print("Training start")
model.fit(train_dataset, validation_data=val_dataset, batch_size=128,
epochs=1000, verbose=1, callbacks=callbacks, shuffle=True)

y = model.predict(x=X_test, batch_size=32)
print(classification_report(y_true=y_test, y_pred=y.round()))

cm = confusion_matrix(y_true=y_test, y_pred=y.round())
sns.heatmap(cm, annot=True, fmt='g', xticklabels=['inter-ictal', 'pre-ictal'],
yticklabels=['inter-ictal', 'pre-ictal'])

from sklearn.metrics import roc_curve, roc_auc_score

false_positive_rate, true_positive_rate, threshold = roc_curve(y_test, y)
print('roc_auc_score: ', roc_auc_score(y_test, y))


plt.subplots(1, figsize=(10,10))
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate)
```

```
plt.plot([0, 1], ls="--")
plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

تعطي أفضل أداء في التمييز بين فترة قبيل النوبة Pre-ictal والفترة بين النوبات Inter-ictal. تم الحصول على متوسط حساسية ٧٧٪ ومتوسط تنبؤ خاطئ منخفض بمقدار ٠.٢ عند اختبار الخوارزمية باستخدام بيانات اشارات مخطط كهربائية الدماغ CHB-MIT.

تم اقتراح خوارزمية توقع النوبات في الزمن الحقيقي التي تحسب كثافة طيف القدرة Power Spectral Density (PSD) لقطاعات تخطيط كهربائية الدماغ الواردة ، وتستخدم التغيرات السريعة في التردد عبر النطاقات كعلامات على النوبات الواردة. أظهرت النتائج قدرة الخوارزمية المقترحة على اكتشاف ٧٨٪ من النوبات في مجموعة بيانات Siena وبمتوسط تنبؤ خاطئ مقداره ٠،١٦.

## الخلاصة

تعد معدات مراقبة حالة الدماغ وتمييز حالتها بشكل آلي من المعدات الطبية المهمة في حياة المرضى في الوقت الراهن. هذه المعدات قائم على أساس تحسس موجات الدماغ على شكل إشارات كهربائية Electroencephalography (EEG) تمثل نشاط الخلايا العصبونية وتحتوي في طياتها الكثير من المعلومات لحالة الدماغ، مثال ذلك تنبؤ حدوث نوبات الصرع لدى مرضى الصرع العصبي. وبسبب طبيعة إشارات الدماغ غير الخطية وغير المستقرة واحتوائها على الضوضاء الناجمة من المؤثرات الخارجية فإن معالجة واستخلاص المعلومات من هذه الإشارت تعد مهمة صعبة.

في هذه الأطروحة، تم تقديم طريقة جديدة لتصفية إشارات الدماغ والغاء تأثير المؤثرات الخارجية قائم على أساس التحليل الاحصائي وتعديل المصادر المستقلة لتلك الإشارات والناتجة عن تحليل المركبات المستقلة Independent Component Analysis (ICA). يظهر الفحص البصري للإشارات المسترجعة صلاحية الطريقة المقترحة في الغاء تأثير المؤثرات الخارجية. يضاف الى ذلك بان الطريقة المقترحة لا تحتاج الى قناة معلومات إضافية ملحقة بقنوات مخطط كهربائية الدماغ.

تم اقتراح وتقييم نموذجين للتعلم العميق، النموذج الأول مبني على الشبكات العصبية التلافيفية Convolutional Neural Network (CNN) والنموذج الثاني مبني على الوحدة المتكررة ذات البوابة مع الذاكرة طويلة– قصيرة المدى Gated Recurrent Unit-Long Short-Term Memory. المهمة الأساسية للنموذجين هي التمييز بين فترة قبيل النوبة Pre-ictal والفترة بين النوبات Inter-ictal وتوقع بداية حصول نوبة الصرع. يراعي النموذجين راحة المريض من حيث استخدام ٥ اقطاب من اشارات الدماغ وتقليل زمن التنبيه الى ١٠ دقائق فقط قبل حدوث النوبة.

تم تنفيذ النموذجين في Python v3 وتم الحصول على نتائج (الحد الأقصى: ٩٠٪ ، المتوسط: ٦٦٪) من حيث الدقة و (الحد الأقصى: ٩٠٪ ، المتوسط: ٦٢٪) من حيث الحساسية باستخدام نموذج الشبكات العصبية التلافيفية. وكذلك (الحد الأقصى: ٧١٪ ، المتوسط: ٥٦٪) الدقة و (الحد الأقصى: ٧٣٪ ، المتوسط: ٥٦٪) تم الحصول عليها باستخدام نموذج الوحدة المتكررة ذات البوابة مع الذاكرة طويلة– قصيرة المدى.

تم اقتراح خوارزمية تعلم آلة مكونة من أربعة مراحل وتنفيذها باستخدام MATLAB 2021. تقوم الخوارزمية باختيار قناة واحدة فقط من قنوات مخطط كهربائية الدماغ كإدخال ثم تقوم بحساب الطول الأمثل لفترة قبل ما قبل التنبؤ Preictal Period Length (PIL) والطول الأمثل لقطعة العينة (SEG)، وكذلك باستخدام طريقة محاكاة التلدين Simulated Annealing للبحث عن مجموعة الميزات المستخلصة التي

## شكر وتقدير

أتقدم بالشكر والعرفان إلى الأستاذ الدكتور قاسم كرم عبد الله لإشرافه على الأطروحة وتقديمه يد العون والمساعدة في كل المجالات. كما يسرني ان اتقدم بالشكر والأمتنان لعمادة وكادر كلية الهندسة/ جامعة بابل وعمادة وكادر كلية هندسة الإلكترونيات/ جامعة نينوى وإدارة وكادر مستشفى السلام التعليمي/ دائرة صحة نينوى وإلى كل من قدم يد العون والمساعدة في انجاز هذا العمل.

الباحث

بسم الله الرحمن الرحيم


الإسراء ٨٥

## اقرار لجنة المناقشة

نحن أعضاء لجنة المناقشة، نشهد بأننا أطلعنا على أطروحة الدكتوراه فلسفة الموسومة

**التنبؤ بنوبات الصرع إعتماداً على إشارات التخطيط الكهربائي للدماغ**

وقد ناقشنا الطالب ( حسين محمد حسين ايدين) في محتوياتها وفيما له علاقة بها، نؤيد أنها جديرة بالقبول لنيل درجة الدكتوراه فلسفة في الهندسة الكهربائية/ هندسة الألكترونيك والإتصالات.

رئيس اللجنة
التوقيع:
الأسم: أ.د. إيهاب عبد الرّزاق حسين
التاريخ:

عضو اللجنة
التوقيع:
الأسم: أ.د. حسن فهد خزعل
التاريخ:

عضو اللجنة
التوقيع:
الأسم: أ.د. سعد سفاح حسون
التاريخ:

عضو اللجنة
التوقيع:
الأسم: أ.د. قيس كريم عمران
التاريخ:

عضو اللجنة
التوقيع:
الأسم: أ.د. حيدر إسماعيل شهادي
التاريخ:

عضو اللجنة (مشرفاً)
التوقيع:
الأسم: أ.د. قاسم كرم عبد الله
التاريخ:

مصادقة رئيس القسم
التوقيع:
الأسم: أ.د. قيس كريم عمران
التاريخ:

مصادقة عميد الكلية
التوقيع:
الأسم: أ. د. حاتم هادي عبيد
التاريخ:

اشهد ان اعداد هذه الأطروحة والموسومة ب ( **التنبؤ بنوبات الصرع إعتماداً على إشارات التخطيط الكهربائي للدماغ**) والمقدمة من قبل الطالب ( **حسين محمد حسين ايدين**) جرت تحت اشرافي في قسم الهندسة الكهربائية/ كلية الهندسة/ جامعة بابل وهي جزء من متطلبات نيل درجة الدكتوراه في هندسة الألكترونيك والاتصالات.

التوقيع:
المشرف: أ. د. قاسم كرم عبد الله
التاريخ: / / ٢٠٢٣

اشهد ان هذه الأطروحة المذكورة اعلاه قد استكملت في الهندسة الكهربائية في كلية الهندسة / جامعة بابل

التوقيع:
رئيس القسم: أ. د. قيس كريم عمران
التاريخ: / / ٢٠٢٣

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية الهندسة
قسم الهندسة الكهربائية

# التنبؤ بنوبات الصرع إعتماداً على إشارات التخطيط الكهربائي للدماغ

الاطروحة

مقدمة إلى كلية الهندسة / جامعة بابل

كجزء من متطلبات الحصول على درجة الدكتوراه

في الهندسة الكهربائية/ الالكترونيك والاتصالات

من قبل

**حسين محمد حسين ايدين**

بأشراف

**الاستاذ الدكتور قاسم كرم عبدالله**

٢٠٢٣ م ١٤٤٤ هـ