



**TÜRKİYE CUMHURİYETİ
ADANA ALPARSLAN TÜRKES SCIENCE AND TECHNOLOGY
UNIVERSITY**

**INSTITUTE OF GRADUATE SCHOOL
ELECTRICAL AND ELECTRONIC ENGINEERING DEPARTMENT**

**EPILEPTIC SEIZURE PREDICTION USING EEG SIGNALS WITH
DEEP LEARNING**

SAMET ORAN

M.Sc.

ADANA 2023



**TÜRKİYE CUMHURİYETİ
ADANA ALPARSLAN TÜRKES SCIENCE AND TECHNOLOGY
UNIVERSITY**

**INSTITUTE OF GRADUATE SCHOOL
ELECTRICAL AND ELECTRONIC ENGINEERING DEPARTMENT**

**EPILEPTIC SEIZURE PREDICTION USING EEG SIGNALS WITH
DEEP LEARNING**

SAMET ORAN

M.Sc.

**THESIS ADVISOR
Prof. Dr. ESEN YILDIRIM**

ADANA, 2023

DECLARATION OF CONFORMITY

In this thesis study, which was prepared following the thesis writing rules of Adana Alparslan Türkeş Science and Technology University Institute of Graduate School, I declare that I provide all the information, documents, evaluations and results in accordance with scientific ethics and moral codes without resorting to any means or assistance that would be contrary to scientific ethics and traditions. I also declare that I refer to all of the articles I used in this study with appropriate references and accept all moral and legal consequences if a situation is found contrary to my statement regarding my work.

11/08/2023

Samet ORAN

ÖZET

DERİN ÖĞRENME YOLUYLA EEG İŞARETLERİNDEN EPİLEPTİK NÖBET TAHMINİ

Samet ORAN

Yüksek Lisans, Elektrik-Elektronik Anabilim Dalı

Danışman: Prof. Dr. Esen YILDIRIM

Ağustos 2023, 113 sayfa

Epilepsi, antik çağlara kadar uzanan kronik bir hastalıktır ve insanları sadece nöbetler sırasında etkilemektedir. Nöbet başlangıcı ve bitiş zamanı bilinmediği için hastaların yaşam standartlarını ciddi şekilde kötü etkilemektedir. Eğer nöbet başlangıcı yeterli bir süre önceden tahmin edilebilirse, ilaçla tedavi olabilen hastaların nöbetlerini ilaçlarla kriz öncesi önleyebilir veya ilaçlarla nöbeti engellenemeyen hastaların ise nöbeti bitene kadar güvenli bir bölgeye geçme fırsatı sağlanabilir.

Bu tezde, epileptik nöbetleri önceden tahmin etme amacıyla, nöbetin meydana gelmesinden yaklaşık olarak 30-60 dakika önce hastaların gerekli uyarıları alabilmesi için zaman yaratılması hedeflenmiştir. Bu bağlamda, 24 pediatrik hastanın elektroensefalografi (EEG) kayıtlarını içeren bir açık erişim veri seti kullanılmıştır. Kaydedilmiş EEG verileri kullanılarak frekans ve zaman temelli öznitelik çıkarımları uygulanmıştır. Frekans ve zaman temelli yöntemler sırasıyla dalgacık dönüşümü ve doğrusal olmayan enerji metotları kullanılarak gerçekleştirilmiştir.

Bu yöntemler kullanılarak iktal (nöbet), preiktal (nöbet öncesi) ve interiktal (nöbetler arası) olarak adlandırılan 3 sınıflı EEG öznitelikleri elde edilerek derin öğrenme ve makine öğrenmesi sınıflandırıcıları uygulanmıştır. Derin öğrenme algoritmalarından oluşturulan 1-boyutlu evrişimli sinir ağı (1D-CNN) modeli ve makine öğrenimi algoritmalarından yaygın olarak kullanılan k-en yakın komşu (k-NN), rastgele orman algoritması (RF), destek vektör makinesi (SVM) ve C4.5 algoritması (J48), 3 sınıflı çıktıların sınıflandırma performanslarını ölçmek için kullanılmıştır.

Çalışmada farklı özniteliklerin birbiriyle kıyaslanmasının yanı sıra makine öğrenimi ve derin öğrenme sınıflandırıcıları da birbirleriyle karşılaştırmış ve en iyi sonuçlar gözlemlenmiştir. Sınıflandırma sonuçlarına göre, dalgacık özniteliklerine dayalı 1D-CNN sınıflandırıcısıyla %95.30'luk en yüksek ortalama doğruluk elde edilirken, doğrusal olmayan enerji özniteliklerini kullanan SVM sınıflandırıcısıyla %96.60'luk en yüksek ortalama doğruluk sağlanmıştır.

Anahtar Kelimeler: Dalgacık Spektrumu, Derin Öğrenme, Doğrusal Olmayan Enerji, Epileptik Nöbet, Öznitelik Çıkarımı, Sınıflandırma.



ABSTRACT

EPILEPTIC SEIZURE PREDICTION USING EEG SIGNALS WITH DEEP LEARNING

Samet ORAN

M.Sc., Department of Electrical and Electronic Engineering

Supervisor: Prof. Dr. Esen YILDIRIM

August 2023, 113 pages

Epilepsy is a chronic disease that dates back to ancient times and affects individuals only during seizures. Due to the unknown onset and duration of seizures, it significantly impairs the quality of life for patients. If the onset of seizures can be predicted with sufficient advance notice, patients who respond to medication can prevent their seizures using appropriate drugs, while those whose seizures cannot be controlled with medication can be provided an opportunity to move to a safe zone until the seizure subsides.

In this thesis, the main objective is to create a time window of approximately 30-60 minutes before the occurrence of epileptic seizures, allowing patients to receive necessary alerts. In this context, an open-access dataset consisting of electroencephalography (EEG) recordings from 24 pediatric patients was utilized. Frequency-based and time-based feature extractions were performed on EEG data. The frequency-based and time-based methods utilized wavelet transformation and non-linear energy calculations respectively.

These feature extraction methods yield to 3-class EEG features that are: ictal (seizure), preictal (pre-seizure), and interictal (between-seizure). Deep learning and machine learning classifiers were applied on these 3-class outputs. A 1-dimensional Convolutional Neural Network (1D-CNN) model, constructed from deep learning algorithms, and commonly used machine learning algorithms, namely k-Nearest Neighbors (k-NN), Random Forest (RF), Support Vector Machine (SVM), and C4.5 algorithm (J48), were employed to assess the classification performance of 3-class outputs.

In the study, besides comparing different features with each other, machine learning and deep learning classifiers have also been compared, and the best results have been observed. According to the classification results, the highest average accuracy of 95.30% was achieved with a 1D-CNN classifier based on wavelet features, while the highest average accuracy of 96.60% was obtained with an SVM classifier using non-linear energy features.

Keywords: Classification, Deep Learning, Epileptic Seizure, Feature Extraction, Non-Linear Energy, Wavelet Spectrum.



ACKNOWLEDGEMENTS

First and foremost, I would like to express my deep gratitude to my esteemed advisor, Prof. Dr. Esen YILDIRIM, for her close involvement and unwavering support throughout every stage of my thesis. I am grateful for her profound knowledge and expertise, which have continuously enriched my learning experience by imparting new insights and perspectives.

Another thanks to my friends Yağnur TETİKCAN, Ömer AYANA, Özlem BÜYÜKTAŞ for their intellectual and motivational support during the implementation and writing of my thesis.

Lastly, I would like to express my heartfelt gratitude to my father, Dindar ORAN, my mother, Afet ORAN, and my brother, Fırat ORAN, for their unconditional love and unwavering support throughout my journey toward success, regardless of the circumstances. Their constant encouragement has been invaluable to me, and I am thankful for their unwavering belief in me.

This thesis was supported by Adana Alparslan Türkeş Science and Technology University Scientific Research Coordination Unit. with project number 22303005.

TABLE OF CONTENTS

DECLARATION OF CONFORMITY	i
ÖZET	ii
ABSTRACT	iv
ACKNOWLEDGEMENTS	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	ix
LIST OF TABLES	xi
LIST OF ABBREVIATIONS	xiii
LIST OF SYMBOLS	xiv
1. INTRODUCTION	1
1.1. Frequency Based Approach	2
1.2. Time Based Approach	3
2. LITERATURE REVIEW	4
3. MATERIALS AND METHODS	8
3.1. Arrangement of the Dataset	8
3.1.1. EEG recordings	8
3.1.2. Assigning of ictal, preictal, and interictal regions	10
3.1.3. Organizing of EEG Channels	13
3.2. Pre-processing	19
3.2.1. Bandpass filter	20
3.2.2. AC Line noise filter	21
3.2.3. Independent Component Analysis (ICA)	21
3.3. Feature Extraction	24
3.3.1. Wavelet Transform (WT)	24
3.3.2. Non-Linear Energy (NE)	25
3.4. Classification	26
3.4.1. Grid search technique	27
3.4.2. k-Nearest Neighbour (kNN)	28
3.4.3. Support Vector Machine (SVM)	29
3.4.4. Random Forest (RF)	32
3.4.5. J48 Algorithm (C4.5)	32

3.4.6. One Dimensional Convolutional Neural Network (1D-CNN)	34
3.4.6.1. The dimensions of the CNN	35
3.4.6.2. Activation functions	36
3.4.6.2.a. ReLU (Rectified Linear Unit) activation function	37
3.4.6.2.b. Softmax function	38
3.4.6.3. The intuition of the 1D-CNN	38
3.4.6.3.a. Convolutional layer	38
3.4.6.3.b. Pooling Layer	40
3.4.6.3.c. Fully Connected (FC) Layer	40
3.4.6.4. The creation of the model	42
4. RESULTS AND DISCUSSIONS	47
4.1. Grid search Results	48
4.1.1. Optimal kNN parameters for WT and NE features	48
4.1.2. Optimal SVM parameters for WT and NE features	50
4.1.3. Optimal RF parameters for WT and NE features	52
4.1.4. Optimal J48 parameters for WT and NE features	53
4.1.5. Optimal 1D-CNN parameters for WT and NE features	54
4.2. Test Results	59
4.2.1. Machine learning classifiers (kNN, SVM, RF and J48) results for WT and NE	59
4.2.2. Deep learning classifier (1D-CNN) results for WT and NE	69
5. CONCLUSIONS	76
6. RECOMMENDATIONS	77
REFERENCES	78
APPENDIXES	87

LIST OF FIGURES

Figure 3.1. Flowchart of the study	8
Figure 3.2. (a) 4th file of the 1st patient presenting the start and end times of a single ictal moment. (b) 38th file of the 12th patient presenting the start and end times of 5 ictal activities.	11
Figure 3.3. Representation of Ictal, Preictal and Interictal parts on the timeline	11
Figure 3.4. Representation on the timeline that the preictal time is at least 15 minutes	11
Figure 3.5. Combination of ictal cases	12
Figure 3.6. Visualization of the 18-channel EEG signals in EEGLAB	19
Figure 3.7. The changes in 0.1 and 120 Hz parts after bandpass filtering is shown in the power/frequency plot.....	20
Figure 3.8. The filtered AC line noise at 60 Hz is shown in the yellow box	21
Figure 3.9. Transforming mixed signals into independent components using ICA.....	22
Figure 3.10. Applying ICA to EEG recording of an epilepsy patient	23
Figure 3.11. Wavelet hovering over the EEG signal. Variation of (a), (b) and (c) wavelets on the same signal on low, medium and high frequency signals, respectively	24
Figure 3.12. Enclosure of the three closest neighbors around the specified point within a circle	28
Figure 3.13. The representation of the Support Vector Machine (SVM) algorithm on data ..	29
Figure 3.14. Random forest is a classifier that consists of multiple decision trees.....	32
Figure 3.15. The relocation of the tree with excessive information gain.	33
Figure 3.16. General view of CNN algorithm.....	35
Figure 3.17. Representation of the Conv1D, 2D and 3D data with their kernels. (a) In Conv1D, kernel slides along one dimension. (b) While Conv2D kernel slides along two dimensions, (c) Conv3D kernel slides along three dimensions	36
Figure 3.18. Demonstration of an activation function in Neural Networks Architecture	37
Figure 3.19. Visual illustration of the ReLU activation function and its derivative	38
Figure 3.20. Conversion of [142000x90x1] a matrix to [141996x86x90] an activation after convolutional layer.....	39
Figure 3.21. Conversion of [141996x86x90] a matrix to [70998x43x90] an activation after pooling layer.....	40
Figure 3.22. General view of the 1D-CNN with 1 input, 3 hidden and 1 output layers for WT	41
Figure 3.23. Model summary of the WT	42
Figure 3.24. Model summary of the NE.....	42
Figure 3.25. Train-test and train-validation split.....	45
Figure 4.1. The accuracy values with respect to parameter k for different distance metrics for WT	48
Figure 4.2. The accuracy values with respect to parameter k for different distance metrics for NE.....	49

Figure 4.3. The accuracy scores of the model with respect to batch size and epochs for NE.	55
Figure 4.4. The accuracy scores of the model with respect to batch size and epochs for WT	55
Figure 4.5. The model and loss function accuracy scores with best parameters (Batch size = 32 and Epochs = 200) for WT	58
Figure 4.6. The model and loss function accuracy scores with best parameters (Batch size = 256 and Epochs = 200) for NE.....	58



LIST OF TABLES

Table 2.1. Performance comparison of the proposed model with other models given in the recent literature for CHB-MIT dataset	4
Table 3.1. Patient-based information of CHB-MIT EEG records	9
Table 3.2. Age and gender information of patients in the CHB-MIT dataset	10
Table 3.3. Durations of the regions	12
Table 3.4. The number of channels and the changes within the channels vary for each case.	14
Table 3.5. Untitled and deleted non-common channels for all subjects	17
Table 3.6. Durations of ictal, preictal, and interictal instances used for grid search	27
Table 4.1. Top 10 accuracies and corresponding parameters based on the average of 10-fold cross-validation kNN for WT.	49
Table 4.2. Top 10 accuracies and corresponding parameters based on the average of 10-fold cross-validation kNN for NE.	50
Table 4.3. Top 10 accuracies and corresponding parameters based on the average of 10-fold cross-validation SVM for WT.	51
Table 4.4. Top 10 accuracies and corresponding parameters based on the average of 10-fold cross-validation SVM for NE.	51
Table 4.5. Top 10 accuracies and corresponding parameters based on the average of 10-fold cross-validation RF for NE.	52
Table 4.6. Top 10 accuracies and corresponding parameters based on the average of 10-fold cross-validation RF for NE.	52
Table 4.7. Top 10 accuracies and corresponding parameters based on the average of 10-fold cross-validation J48 for WT.	53
Table 4.8. Top 10 accuracies and corresponding parameters based on the average of 10-fold cross-validation J48 for NE.	54
Table 4.9. The top 10 parameters selected results based on 1D-CNN for WT	56
Table 4.10. The top 10 parameters selected results based on 1D-CNN for NE.	56
Table 4.11. Weighted and unweighted accuracy results for WT	59
Table 4.12. The recall results of the machine learning classifiers for WT	61
Table 4.13. The precision results of the machine learning classifiers for WT	62
Table 4.14. The F1 score results of the machine learning classifiers for WT	63
Table 4.15.. Weighted and unweighted accuracy results for NE	65
Table 4.16. The recall results of the machine learning classifiers for NE	66
Table 4.17. The precision results of the machine learning classifiers for NE	67
Table 4.18. The F1 score results of the machine learning classifiers for NE	68
Table 4.19. Weighted and unweighted accuracy, precision, recall and F1score 1D-CNN results for WT	70
Table 4.20.. Precision, recall and F1score results from 3-class classification 1D-CNN (Ictal, preictal and interictal) for WT	71
Table 4.21. Weighted and unweighted accuracy, precision, recall and F1 score 1D-CNN results for NE	72

Table 4.22. Precision, recall and F1score results from 3-class classification 1D-CNN (Ictal, preictal and interictal) for NE	73
Table 4.23. The summary of weighted and unweighted (macro) average accuracy results	74



LIST OF ABBREVIATIONS

kNN	: k-Nearest Neighbor
SVM	: Supported Vector Machine
RF	: Random Forest
WT	: Wavelet Transform
NE	: Non-linear Energy
CNN	: Convolutional Neural Networks
1D-CNN	: 1-Dimensional Convolutional Neural Networks

LIST OF SYMBOLS

p	: Probability function
A	: Mixing matrix
W	: Demixing matrix
F	: Extracted EEG feature
S	: Wavelet spectrum function
NE	: Non-linear energy function
k	: Kernel function

1. INTRODUCTION

Epilepsy is a chronic noncommunicable disease also known as falling sickness. In an epileptic seizure, sudden and uncontrolled electrical discharges occur in neurons and consequently changes in consciousness, involuntary contractions, and emotional changes arise in the patient. Epilepsy is a disease that only affects the patient during the seizures and patients continue to live their lives in a healthy way between those seizures. Approximately 65 million epileptic patients are called in the world and epilepsy is known as the most common neurological disease in childhood and adolescence period, as the second most common neurological disease in adults after cerebrovascular diseases. [1] Lots of different reasons (age, brain infections, childhood seizures, dementia, hereditary family history, vascular disorders, and head traumas) may contribute to the development of seizures. Although there exists no currently available definitive treatment for epilepsy, it is a disorder that can be controlled with anti-seizure strategies and medications.

The first known epileptic observations in history were seen approximately 3000 years ago in the Babylonian period. Babylonians named this disease "Migtu", and it was believed that Migtu was caused by mystical beings attacking humans at that time [2]. For the first time, Hippocrates declared that the center of epilepsy was in the brain, and he named it "Mal Caduque" [3]. In the 1960s, the classification studies of epileptic seizures started, and the Commission on Classification and Terminology was established [4].

Brain-computer interfaces (BCIs) are developed with the purpose of helping people with disabilities due to neuromuscular disorders [5]. The purpose of a BCI system is to analyze the acquired brain signals and send the required command to a device for the desired action. For such systems, brain signals should be captured during mental tasks, which point to the activities of the brain such as imagining counting, raising a hand, or calculating. During the mental effort, physiological recordings are collected for further processing. EEG is one of the most frequently used modalities in cognitive activity detection. EEG is a non-invasive method and economic compared to other neuroimaging modalities, such as functional magnetic resonance imaging (fMRI) and positron emission tomography (PET) and provides higher temporal resolution. At the present time, although the change in EEG signals during epileptic seizures is noticeable

even with the bare eye, computer-assisted expert systems are necessarily needed to be used for detailed analysis and early prediction. Therefore, machine learning and deep learning methods are used extensively in the analysis of epileptic EEG signals [6-10].

The fundamental aim of the study is to predict the seizures 30-60 minutes before the onset with a subject-dependent seizure prediction system. In this way, patients would be able to take their medicine in time which will reduce the overall number of seizures they have and significantly improve their quality of life by preventing accidents and injuries. For this purpose, two different feature extraction methods were used on the “CHB-MIT Scalp EEG Database” [11], which is available as open source in PhysioNet (“<https://physionet.org/content/chbmit/1.0.0/>”), to apply 3-class classification, preictal, ictal, and interictal, using the deep learning (1D-CNN) and machine learning (kNN, RF, SVM, and J48) methods. Therefore, an automated seizure onset prediction system based on EEG measurements was realized.

1.1. Frequency Based Approach

The first of these methods is frequency-based feature extraction carried out by means of wavelet transform. This extraction method is based on frequency analysis of the brain waves. The human brain generates five fundamental frequency-based waves. Delta waves (0.5-4Hz) are the largest and slowest type of wave. They are observed during deep sleep in all age groups [12]. The visualization of delta waves in awake adults is considered abnormal. They appear during the deepest levels of sleep, in a coma state, or when a person is under anesthesia. Delta waves have the lowest frequency among brain waves and can be visualized either concentrated in a specific region or distributed throughout the brain. Theta waves (4-8Hz) typically emerge in adults during emotional stress or disappointment. In children, they appear in the parietal and temporal regions [12]. Alpha waves (8-12Hz) are observed in awake, normal, and calm individuals. They appear predominantly in the occipital region [12] and disappear during sleep. When an awake person directs their attention to something specific, these waves give way to higher-frequency beta waves [13]. Beta waves (12-30Hz) emerge during highly stressful situations when we find it challenging to collect our thoughts and concentrate and they disappear with increased mental activity, similar to alpha waves. Gamma waves are still an area of intensive research [12]. Thus, by examining the frequencies of the brain waves described above, they can be utilized for early seizure prediction in epilepsy patients.

1.2. Time Based Approach

The patient records in the EEG dataset capture the magnitudes obtained from instantaneous measurements over time. Various time-domain feature extraction methods have been used for epilepsy detection. In a review study [14], 20 time-domain feature proposals were considered, out of which 6 (Non-linear energy, Shannon energy, mean, variance, mode, median, skewness, and kurtosis) were implemented. In this thesis, we used non-linear energy metric for capturing signals' time domain characteristics.

This method was introduced in 1990 [15] and demonstrates that the energy of a simple oscillator is directly proportional to the square of the product of its amplitude and frequency. Subsequently, an algorithm is developed by taking three consecutive samples from a signal, making it easy to implement. In this regard, our study aims to demonstrate the successful extraction of epilepsy features using this method (for details of the method see Section 3.3.2).

2. LITERATURE REVIEW

In this section, previous studies that have been documented in the literature using the CHB-MIT dataset, which was utilized in the thesis, have been examined. This section as discusses the methods and classifiers employed, and interpretation of the results obtained from the developed models. Although these studies employ the same dataset, there exist certain differences such as fewer channels, seizure occurrences, and subject numbers within the dataset. Table 2.1 provides a summary of 9 studies focusing on the early detection of epilepsy seizures using the CHB-MIT dataset.

Table 2.1. Performance comparison of the proposed model with other models given in the recent literature for CHB-MIT dataset

References	Features	Classifiers	Number of seizures-subjects-channels	Validation methods	Classifier outputs	Average results Sens. – Acc. (%)
Zhang and Parhi [16] (2016)	Absolute/Relative spectral power	SVM	78 – 17 – single	Leave-one-out cross validation	Interictal / Preictal	98.68 – NM
Ibrahim <i>et al.</i> [17] (2017)	Shannon entropy	kNN	55 – 10 – 23	NM	Normal / Pre-seizure	76.00 – NM
Bhattacharyya <i>et al.</i> [18] (2017)	Empirical wavelet transform	RF	157 – 23 – 5	10-fold cross validation	Seizure / Seizure-free	97.91 – 99.41
Khan <i>et al.</i> [19] (2018)	Wavelet transform	CNN	18 – 15 – NM	10-fold cross validation	Interictal / Preictal / Ictal	87.80 – NM
Truong <i>et al.</i> [20] (2018)	STFT spectral images	CNN	64 – 13 – 22	Leave-one-out cross validation	Interictal / Preictal	81.2 – NM
Ozcan <i>et al.</i> [21] (2019)	Spectral power, Statistical moment, Hjorth parameters	3D-CNN	77 – 16 – 22	Leave-one-out cross validation	Interictal/Preictal /Ictal/Postictal	86.84 – NM 87.01 – NM 85.71 – NM
Yao <i>et al.</i> [22] (2019)	IndRNN	IndRNN	NM – 24 – 17	10-fold cross validation	Seizure / Non-seizure	87.30 – 87.00
Singh <i>et al.</i> [23] (2022)	Spectral power	LSTM	NM – 24 – 23	10-fold cross validation	Interictal / Preictal	98.51– 98.14
Dash <i>et al.</i> [24] (2022)	Wavelet transform	kNN	NM – 23 – 18	Leave-one-out cross validation	Seizure / Non-seizure	99.56 – 98.99
Proposed model	Wavelet transform, Non-linear energy	kNN, RF, SVM, J48, 1D-CNN	185 – 24 – 18	10-fold cross validation, 80% train – 20% test	Interictal / Preictal / Ictal	95.30 – 95.30 96.66 – 96.60

*NM: not mentioned

According to the chronological order of the studies in the Table 2.1, Zhang and Parhi [16] presented a patient-specific algorithm for predicting epileptic seizures. This framework is subject-dependent and aims to reduce the complexity of the recordings by utilizing single or dual-channel information from each patient. Seventeen out of the 24 patients in the dataset were included in this study, and the analysis focused on 78 seizures using single-channel data. Frequency-based methods, namely absolute and relative spectral power, were employed for feature extraction. In the postprocessing stage, a Kalman filter was applied to mitigate irregular effects in the signals. The feature selection process involved a two-layer approach. The Support Vector Machine (SVM) classifier was used for distinguishing between Interictal and Preictal states. The leave-one-out method was employed as the validation technique. The study reported the highest accuracy value, calculated based on recall (sensitivity), which was found to be 98.68%.

Ibrahim et al. [17] selected 10 patients from the dataset and utilized 55 seizure instances obtained from 23 channels for early seizure prediction. The features of the data were computed based on a time-based calculation using Shannon entropy, and the extracted features were categorized as Normal/Pre-seizure by comparing entropy values. The study did not provide information about cross-validation but used the k-Nearest Neighbors (kNN) algorithm for binary classification. The highest sensitivity value obtained was 76.00%.

In the study conducted by Bhattacharyya et al. [18], the sliding window method was used to extract 2-second multivariate EEG signal segments from five automatically selected channels. Three features were extracted from the common values of 1-second segments within each 2-second segment. A total of 157 seizure instances from 23 subjects were utilized in the study. The features were obtained using empirical wavelet transform and classified using the Random Forest (RF) machine learning algorithm. 10-fold cross-validation was chosen as the validation method during classification. The most successful values obtained from the classification were 97.91% for sensitivity and 99.41% for accuracy.

Khan et al. [19] aimed to achieve early seizure prediction by distinguishing Preictal, Ictal, and Interictal states. Fifteen subjects were selected from 24 subjects in the dataset, and it included 18 seizures. Wavelet transform was employed for feature extraction. In this study, which performed three-class classification using a CNN model, the train-test dataset was set to 90%

for training and 10% for testing. 10-fold cross-validation was employed for the validation of the training data. The achieved sensitivity rate in the classification was 87.80%.

Truong et al. [20] proposed a patient-specific approach for seizure prediction. This study included 13 out of 24 subjects and utilized 22 out of 23 channels to predict seizures based on a total of 64 ictal data instances. They employed the Discrete Time Fourier Transform (DTFT) method on 30-second EEG signal windows to extract both frequency and time-based features. The obtained feature magnitudes were visualized to obtain a spectral view, enabling Preictal/Interictal classification using a 6-layer CNN classifier. The train-test ratio of the dataset was not specified. A validation rate of 20% was determined for the training data, and the leave-one-out technique was used for validation. The sensitivity value obtained from the classification was 81.2%.

Ozcan et al. [21], proposed a subject-independent seizure prediction approach. This study utilized data from 16 out of 24 patients with 22 channels, resulting in a total of 77 seizures. They tried three different methods to extract both time and frequency-based features: spectral band power for frequency-based features, statistical moment, and Hjorth parameters for time-based features. These three different features were transformed into a sequence of color images, and a 3D-CNN classifier was employed to obtain outputs categorized as Interictal/Preictal/Ictal/Postictal. The train-test ratio of the utilized data was set to 80% - 20%. Leave-one-out method was used in five parts for validation in the training set. The classification results for spectral band power, statistical moment, and Hjorth parameters were 86.84%, 87.01%, and 85.71% for sensitivity, respectively.

Yao et al. [22] proposed a method that gradually expands time scales for seizure/non-seizure classification, thus extracting both temporal and spatial features. All 24 subjects were included in the study, and 17 of them were used for analysis across 23 channels. They obtained the features and performed classification through the Independently Recurrent Neural Network (IndRNN) method. The train-test-validation ratios in the dataset were set to 70%, 15%, and 15% respectively. For the training set, the 10-fold cross-validation method was employed for validation. The classification results using the IndRNN method were 87.30% for sensitivity and 87.00% for accuracy.

Singh et al. [23] used the spectral power of delta, theta, alpha, beta, and gamma bands as features for early seizure prediction. Using data from 24 subjects and 23 channels, the segmented data of 30-second intervals were classified as interictal/preictal using Long-Short Term Memory (LSTM). The train-test ratio in the study's dataset was 90%-10%. Validation of the train data was performed using the 10-fold cross-validation method. The highest success rate achieved was 98.51% for sensitivity and 98.14% for accuracy.

Dash et al. [24] performed automatic seizure prediction using 18 EEG channels from 23 patients. Frequency-based features were extracted using wavelet transform after applying a Butterworth filter to the data, and Katz fractal dimension was used as a feature selection method after wavelet transform. The k-Nearest Neighbors (kNN) classifier was used for seizure/non-seizure differentiation. Leave-one-out cross validation method was applied in this study. The highest sensitivity value obtained from the classification was 99.56%, and the accuracy value was 98.99%.

This thesis focuses on the early seizure prediction by analyzing a total of 185 seizures using data from all 24 subjects through 18 channels, aiming to establish a subject-dependent framework. To reduce noise originating from patients or external factors, AC noise, bandpass, and Independent Component Analysis (ICA) filters were employed. Time and frequency-based feature extraction was performed using the non-linear energy and wavelet transform methods, respectively. After feature extraction, machine learning and deep learning techniques were employed for classification. The best results were achieved using wavelet transform, with a sensitivity of 95.30% and an accuracy of 95.30%. For the non-linear energy method, the sensitivity was 96.66% and the accuracy was 96.60%.

3. MATERIALS AND METHODS

In this section, we present the available dataset, organization of EEG, pre-processing, and the proposed solution method. Section 3.1 explains the arrangement of the dataset and assigning of regions of the dataset. The pre-processing of the EEG data for feature extraction is described in Section 3.2. Subsequently, in Section 3.3, feature extraction is performed using two different methods. Finally, in Section 3.4, the classification of the extracted features is explained. In Figure 3.1, the flow diagram of the applied steps is illustrated.

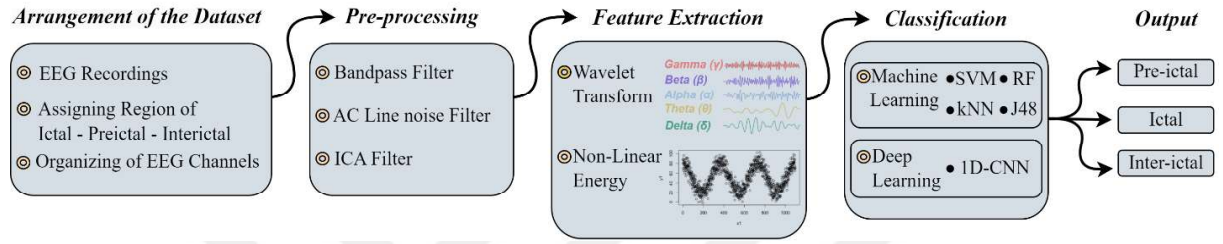


Figure 3.1. Flowchart of the study

3.1. Arrangement of the Dataset

The organization of the pre-processing phase of the dataset in this study was established to ensure the applicability of our proposed solution method and to enable a reliable pre-processing procedure. In order to pre-process the dataset and perform feature extraction, certain metrics need to be examined and adjusted. These include the analysis of EEG recordings, identification of their outputs (Ictal, Preictal, and Interictal), and lastly the adjustment of channel numbers in the recordings to provide the implementation of the proposed method.

3.1.1. EEG recordings

In this study, the publicly available CHB-MIT database [11] was utilized. Collaboratively established by the Massachusetts Institute of Technology (MIT) and Boston Children's Hospital (CHB), the database consisted of 950 hours of raw EEG recordings performed at a sampling frequency of 256Hz, according to the international 10-20 electrode placement system. Within the 950-hour EEG dataset, there are a total of 680 recording files from various patients, encompassing 185 cases of severe epileptic seizures (not controlled with seizure medications) instances.

The cases were divided into consecutive 1-hour recordings, except for case10, which had 2-hour recordings, and case04, case06, case07, case09, and case23, which had 4-hour recordings. The reason for dividing the files into segments is that processing a single file as a whole can be challenging and slow on a computer. Table 3.1 presents the EEG recording durations, the number of seizures, and the number of EEG files for each patient.

Table 3.1. Patient-based information of CHB-MIT EEG records

Case	# of Files	# of Seizures	Duration of Recordings (hh:mm:ss)	Case	# of Files	# of Seizures	Duration of Recordings (hh:mm:ss)
01	42	7	39:28:07	14	26	8	24:12:10
02	36	3	34:46:01	15	40	20	36:20:02
03	38	7	36:55:05	16	19	10	16:01:59
04	42	4	155:28:57	17	21	3	20:00:24
05	39	5	38:10:10	18	36	6	34:42:08
06	18	10	65:10:40	19	30	3	29:31:39
07	19	3	66:33:08	20	29	8	26:13:59
08	20	5	19:15:24	21	33	4	32:10:01
09	19	4	67:12:19	22	31	3	30:29:21
10	25	7	48:51:25	23	9	7	25:21:03
11	35	3	34:17:40	24	22	16	18:52:51
12	18	27	19:38:28	Total	680	185	950:35:09
13	33	12	30:52:08				

The dataset includes scalp EEG signals recorded from 23 pediatric patients 5 males aged 3-22 years, 17 females aged 1.5-19 years, and 1 patient with undisclosed gender and age. Additionally, the case21 was recorded 1.5 years later from the same female subject (case01). The gender and age information for the 24th case (case24), with undisclosed gender and age, was added to this dataset later. The gender and age details of the patients are presented in Table 3.2.

Table 3.2. Age and gender information of patients in the CHB-MIT dataset

Case	Gender	Age	Case	Gender	Age
01	F	11	13	F	3
02	M	11	14	F	9
03	F	14	15	M	16
04	M	22	16	F	7
05	F	7	17	F	12
06	F	1.5	18	F	18
07	F	14.5	19	F	19
08	M	3.5	20	F	6
09	F	10	21	F	13
10	M	3	22	F	9
11	F	12	23	F	6
12	F	2	24	-	-

In order for the dataset to undergo preprocessing smoothly, EEG recordings need to go through two stages. This is because there exists a channel mismatch in the raw EEG recordings, which will be explained in detail later. Additionally, the identification of regions for classification constitutes one of the building blocks of this study. Therefore, the processes to be carried out are the "**Assigning region of ictal, preictal, and interictal regions**" followed by the "**Organizing of EEG channels.**"

3.1.2. Assigning of ictal, preictal, and interictal regions

In this section, in order to extract features from the dataset and enable a 3-class classification, the detection of preictal and interictal regions is necessary, apart from the ictal activities. However, before explaining how they are detected, a brief explanation of the terminology used in these sections should be provided. Preictal refers to the activities that occur before the onset of a seizure. Ictal represents the state during which the seizure occurs, from its beginning to its end, while interictal denotes the normal brain activities that occur outside of these two states. Records containing ictal activity and the start-stop times of the activity are explicitly indicated within the patient recording folders. Figure 3.2 presents the information pertaining to the ictal moment within the text files for subjects 01 and 12.

File Name: chb01_04.edf	File Name: chb12_38.edf
File Start Time: 14:43:12	File Start Time: 03:11:04
File End Time: 15:43:12	File End Time: 4:11:04
Number of Seizures in File: 1	Number of Seizures in File: 5
Seizure Start Time: 1467 seconds	Seizure 1 Start Time: 1548 seconds
Seizure End Time: 1494 seconds	Seizure 1 End Time: 1573 seconds
	Seizure 2 Start Time: 2798 seconds
	Seizure 2 End Time: 2821 seconds
	Seizure 3 Start Time: 2966 seconds
	Seizure 3 End Time: 3009 seconds
	Seizure 4 Start Time: 3146 seconds
	Seizure 4 End Time: 3201 seconds
	Seizure 5 Start Time: 3364 seconds
	Seizure 5 End Time: 3410 seconds

(a)

(b)

Figure 3.2. (a) 4th file of the 1st patient presenting the start and end times of a single ictal moment. (b) 38th file of the 12th patient presenting the start and end times of 5 ictal activities.

As seen in Figure 3.2, each file contains information regarding the number of seizures, the start-stop times of the seizures, and the file duration. The segmentation process is based on setting the length of each segment of EEG recordings to 1 second, considering a sampling frequency of 256 Hz. Based on this information, ictal, preictal, and interictal periods were established using the specified conditions, considering the start and end times of the ictal activities

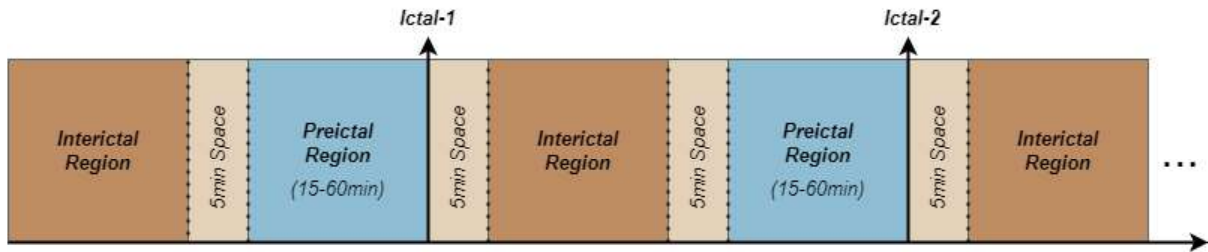


Figure 3.3. Representation of Ictal, Preictal and Interictal parts on the timeline

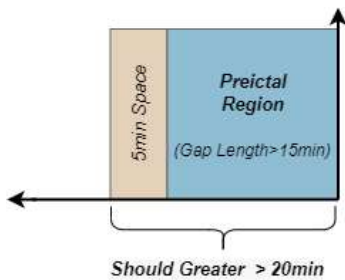


Figure 3.4. Representation on the timeline that the preictal time is at least 15 minutes

The preictal time interval covers a specific period immediately before the ictal activity. This specific time ranges from a minimum of 15 minutes to a maximum of 60 minutes preceding the ictal activity. The interictal interval, on the other hand, encompasses the time interval between two ictal activities, excluding the preictal periods and a 5-minute gap. The 5-minute gap areas in Figure 3.3 are left intentionally to minimize the potential interference between ictal, preictal, and interictal regions.

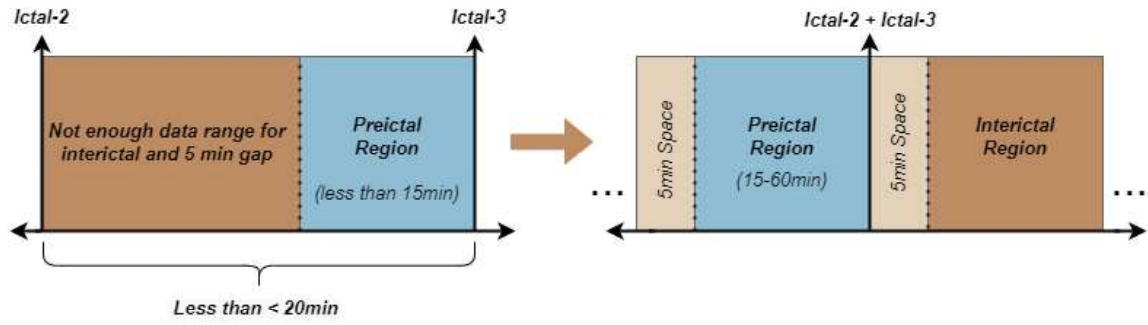


Figure 3.5. Combination of ictal cases

For the preictal region to exist in Figure 3.4, the recording duration should be at least 15 minutes. If there is no sufficient time interval for preictal recording between seizures (less than 15 minutes), those seizures are merged into a single seizure activity, and the intermediate period is removed. This situation is illustrated in Figure 3.5. The duration of ictal, interictal, and preictal periods within the dataset are provided in Table 3.3, as indicated by the configurations mentioned above.

Table 3.3. Durations of the regions

Case	Total duration of Instances (in seconds)			
	Interictal	Preictal	Ictal	Total (hh:mm:ss)
01	120252	21386	449	39:28:07
02	114726	10260	175	34:46:01
03	113475	19021	409	36:55:05
04	546855	12500	382	155:28:57
05	119447	17400	563	38:10:10
06	200264	34123	253	65:10:40
07	228460	10800	328	66:33:08
08	51331	17069	924	19:15:24
09	227259	14400	280	67:12:19
10	150591	24840	454	48:51:25
11	113771	8880	809	34:17:40
12	40072	29620	1016	19:38:28
13	86281	24300	547	30:52:08
14	66011	20942	177	24:12:10
15	83052	45738	2012	36:20:02
16	42128	15497	94	16:01:59
17	62271	9457	296	20:00:24
18	110985	13620	323	34:42:08
19	95800	10260	239	29:31:39
20	75174	18963	302	26:13:59
21	103264	12334	203	32:10:01
22	98754	10800	207	30:29:21
23	71932	18900	431	25:21:03
24	30365	37079	527	18:52:51
Total	820:08:40	127:16:29	03:10:00	950:35:09

When examining the results in Table 3.3, it can be observed that 820 hours, corresponding to 86.27%, are labeled as interictal. This is followed by 127 hours, accounting for 13.38%, labeled as preictal, and 3 hours, corresponding to 0.35%, labeled as ictal. These imbalanced ratios can lead to misleading values when performing 3-class classification. However, we will explain the solution to this issue with the details in Section 4, under the Results section. On the other hand, by examining the durations on a patient basis, Sub04 (case04) has the longest duration, while Sub16 (case16) has the shortest duration.

3.1.3. Organizing of EEG Channels

In order to perform feature extraction from the records of 24 patients in the dataset, it is necessary for the channel names in each case's matrix files to be aligned and have an equal number of channels. Otherwise, comparing different channels with each other would not yield accurate results. Within this study, it has been observed that the records in the dataset are not consistent for each subject. Furthermore, it has been determined that the channel names change over time in the EEG data. Thus, it is possible for the data from a specific channel to continue through another channel after a certain time. To address this issue, our reference point is the recording of when channel name changes occur during data acquisition. Therefore, the identification of common channels and the removal of non-existent channels are necessary. Additionally, if there has been a change in the order of channels, they are rearranged according to the common ranking. Lastly, if there are any mismatched channels based on names, it is necessary to remove the corresponding data from the entire dataset. Based on these records, Table 3.4 clearly displays the number and names of raw dataset channels.

Table 3.4. The number of channels and the changes within the channels vary for each case.

	Case 01	Case 02	Case 03	Case 04	Case 05	Case 06	Case 07	Case 08	Case 09	Case 10	Case 11
Channel 01	FP1-F7	FP1-F7	FP1-F7	FP1-F7	FP1-F7	FP1-F7	FP1-F7	FP1-F7	FP1-F7	FP1-F7	FP1-F7
Channel 02	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7
Channel 03	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7
Channel 04	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1
Channel 05	FP1-F3	FP1-F3	FP1-F3	FP1-F3	FP1-F3	FP1-F3	FP1-F3	FP1-F3	FP1-F3	FP1-F3	-
Channel 06	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	FP1-F3
Channel 07	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	F3-C3
Channel 08	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	C3-P3
Channel 09	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	P3-O1
Channel 10	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	-
Channel 11	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	FZ-CZ
Channel 12	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	CZ-PZ
Channel 13	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	-
Channel 14	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	FP2-F4
Channel 15	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	F4-C4
Channel 16	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	C4-P4
Channel 17	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	P4-O2
Channel 18	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	-
Channel 19	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	FP2-F8
Channel 20	T7-F7	T7-F7	T7-F7	T7-F7	T7-F7	T7-F7	T7-F7	T7-F7	T7-F7	T7-F7	F8-T8
Channel 21	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	T8-P8
Channel 22	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	P8-O2
Channel 23	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	-
Channel 24				ECG					VNS		P7-T7
Channel 25											T7-F7
Channel 26											FT9-FT10
Channel 27											FT10-T8
Channel 28											T8-P8

Table 3.4. The number of channels and the changes within the channels vary for each case (Continued)

	Case 12				Case 13				Case 14	Case 15		Case 16	
	FPI-F7	F7-CS2	FPI-F7	FPI-F7	FPI-F7	FPI-F7	FPI-F7	FPI-F7	FPI-F7	FPI-F7	FPI-F7	FPI-F7	FPI-F7
Channel 01	F7-T7	T7-CS2	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7
Channel 02	T7-P7	P7-CS2	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7
Channel 03	P7-O1	-	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1
Channel 04	-	FPI-CS2	-	-	-	-	-	-	-	-	-	-	-
Channel 05	FPI-F3	F3-CS2	FPI-F3	FPI-F3	FPI-F3	FPI-F3	FPI-F3	FPI-F3	FPI-F3	FPI-F3	FPI-F3	FPI-F3	FPI-F3
Channel 06	F3-C3	C3-CS2	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3
Channel 07	C3-P3	P3-CS2	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3
Channel 08	P3-O1	O1-CS2	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1
Channel 09	-	-	-	-	-	-	-	-	-	-	-	-	-
Channel 10	FZ-CZ	FZ-CS2	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ
Channel 11	CZ-PZ	CZ-CS2	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ
Channel 12	-	PZ-CS2	-	-	-	-	-	-	-	-	-	-	-
Channel 13	FP2-F4	-	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4
Channel 14	F4-C4	FP5-CS2	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4
Channel 15	C4-P4	F4-CS2	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4
Channel 16	P4-O2	C4-CS2	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2
Channel 17	-	P4-CS2	-	-	-	-	-	-	-	-	-	-	-
Channel 18	FP2-F8	O2-CS2	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8
Channel 19	F8-T8	-	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8
Channel 20	T8-P8	F8-CS2	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8
Channel 21	P8-O2	T8-CS2	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2
Channel 22	-	P8-CS2	-	-	-	-	-	-	-	-	-	-	-
Channel 23	P7-T7	-	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7
Channel 24	T7-FT9	CS-CS2	T7-FT9	T7-FT9	T7-FT9	T7-FT9	T7-FT9	T7-FT9	T7-FT9	T7-FT9	T7-FT9	T7-FT9	T7-FT9
Channel 25	FT9-FT10	C6-CS2	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10
Channel 26	FT10-T8	CP2-CS2	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8
Channel 27	T8-P8	CP4-CS2	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8
Channel 28	-	CP6-CS2	-	-	-	-	-	-	-	-	-	-	-
Channel 29	-	-	-	-	-	-	-	-	-	-	-	-	-
Channel 30	-	-	-	-	-	-	-	-	-	-	-	-	-
Channel 31	-	-	-	-	-	-	-	-	-	-	-	-	-
Channel 32	-	-	-	-	-	-	-	-	-	-	-	-	-
Channel 33	-	-	-	-	-	-	-	-	-	-	-	-	-
Channel 34	-	-	-	-	-	-	-	-	-	-	-	-	-
Channel 35	-	-	-	-	-	-	-	-	-	-	-	-	-
Channel 36	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 3.4. The number of channels and the changes within the channels vary for each case (Continued)

	Case 17			Case 18			Case 19			Case 20	Case 21	Case 22	Case 23	Case 24
Channel 01	FP1-F7	FP1-F7	FP1-F7	FP1-F7	FP1-F7	FP1-F7	FP1-F7	FP1-F7	FP1-F7	FP1-F7	FP1-F7	FP1-F7	FP1-F7	FP1-F7
Channel 02	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7	F7-T7
Channel 03	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7	T7-P7
Channel 04	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1	P7-O1
Channel 05	-	-	-	-	-	-	-	-	-	-	-	-	-	FP1-F3
Channel 06	FP1-F3	FP1-F3	FP1-F3	FP1-F3	FP1-F3	FP1-F3	FP1-F3	FP1-F3	FP1-F3	FP1-F3	FP1-F3	FP1-F3	FP1-F3	F3-C3
Channel 07	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	F3-C3	C3-P3
Channel 08	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	C3-P3	P3-O1
Channel 09	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	P3-O1	FP2-F4	FP2-F4
Channel 10	-	-	-	-	-	-	-	-	-	-	-	-	F4-C4	F4-C4
Channel 11	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	FZ-CZ	C4-P4	C4-P4
Channel 12	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	CZ-PZ	P4-O2	P4-O2
Channel 13	-	-	-	-	-	-	-	-	-	-	-	-	FP2-F8	FP2-F8
Channel 14	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	FP2-F4	F8-T8	F8-T8
Channel 15	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	F4-C4	T8-P8	T8-P8
Channel 16	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	C4-P4	P8-O2	P8-O2
Channel 17	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	P4-O2	FZ-CZ	FZ-CZ
Channel 18	-	-	-	-	-	-	-	-	-	-	-	-	CZ-PZ	CZ-PZ
Channel 19	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	FP2-F8	P7-T7	P7-T7
Channel 20	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	F8-T8	T7-F7	T7-F7
Channel 21	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	FT9-FT10	FT9-FT10
Channel 22	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	P8-O2	FT10-T8	FT10-T8
Channel 23	-	-	-	-	-	-	-	-	-	-	-	-	T8-P8	T8-P8
Channel 24	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	P7-T7	-	-
Channel 25	T7-F7	T7-F7	T7-F7	T7-F7	T7-F7	T7-F7	T7-F7	T7-F7	T7-F7	T7-F7	T7-F7	T7-F7	-	-
Channel 26	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	FT9-FT10	-	-
Channel 27	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	FT10-T8	-	-
Channel 28	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	T8-P8	-	-
Channel 29	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Channel 30	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Channel 31	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Channel 32	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Channel 33	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Channel 34	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Channel 35	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Channel 36	-	-	-	-	-	-	-	-	-	-	-	-	-	-

In the Table 3.4, it can be noticed that there are empty channel names within each subject. Additionally, in some cases, channel names have been changed or removed over time. According to the records, a maximum of 6 channel changes have been made (on case-13). To organize the channels systematically, the channel table with the least inclusiveness among the cases has been taken as a reference. As a result, we can easily align the same channel names in rows and conveniently remove non-shared channels. The removal of unnamed and non-shared channels from the respective cases, based on the information from Table 3.4, is illustrated in Table 3.5.

Table 3.5. Untitled and deleted non-common channels for all subjects

Case	# of EEG Channels in Default Case						Untitled Channel/Channels	Deleted Channel/Channels	# of EEG Channels After Regulations
01	23						No Change No Change No Change No Change No Change No Change No Change 5,10,13,18,23 5,10,13,18,23 5,10,13,18,23 5,10,13,18,23 For first order:5,10,13,18,23 For second order:5,10,14,19,24,30 5,10,13,18,23 5,10,13,18,23 5,10,13,18,23 5,10,13,18,23 5,10,13,18,23 5,10,13,18,23 5,10,13,18,23 No Change 5,10,13,18,23 No Change No Change No Change No Change	No Change	23
02	23							No Change	23
03	23							No Change	23
04	23	24						24	23
05	23							No Change	23
06	23							No Change	23
07	23							No Change	23
08	23							No Change	23
09	23	24						24	23
10	23							No Change	23
11	23	28						28	23
12	28	29*		29				29	23
13	28	25	22	28	22	28		25,22,28,22,28	18
14	28							No Change	23
15	31	36						36	18
16	28	22						22	18
17	28	22						22	18
18	22	28						28	18
19	22	28						28	18
20	28							No Change	23
21	28							No Change	23
22	28							No Change	23
23	23							No Change	23
24	23							No Change	23

The "*" sign in the 2nd channel order of Case-12 was deleted.

The "*" sign in the 2nd channel order of Case-12 was deleted.

According to the table above, there are no anonymous channels in Case-01, 02, 03, 05, 06, 07, 08, 10, 23, and 24 among the subjects. As the EEG data for these cases were recorded with regular channel names, no intervention is required. Furthermore, the electrode placements remained unchanged, and no additional channels were added throughout the recording. Thus, the 23-channel EEG data were obtained from these cases.

In Case-14, 20, 21, and 22, the electrode placements remained unchanged throughout the recording, and the data were recorded using regular channel names. However, in these cases, anonymous channels (ch-5, 10, 13, 18, 23) were removed, resulting in obtaining a 23-channel EEG data from these specified patients.

In Case-04, 09, 11, and 15, the electrode arrangement was changed once during the recording. Case-04 and 09 did not have any anonymous channels. However, in these two cases, the additional 24th channels labeled as "ECG" and "VNS" were removed, resulting in the creation of a 23-channel recording data.

In Case-11 and 15, anonymous channels (ch-5, 10, 13, 18, 23) and channels that did not match with additional recordings were removed as indicated in the table. As a result, 23-channel recording data was obtained from Case-11, while 18-channel recording data was obtained from Case-15.

In Case-16, 17, 18, and 19, the electrode arrangement was changed once during the recording. Additionally, these cases have common anonymous channels (ch-5, 10, 13, 18, 23) and mismatched channels. When these two types of channels are removed, 18-channel regular EEG data is obtained from these cases.

In Case-12, the electrode arrangement was changed four times during the recording. However, the EEG channels in column first could not be explicitly matched with any other two channels and were thus removed. The remaining two channels were selected as the channel reference, considering their lower count and excess and anonymous channels were discarded. As a result, 23-channel regular EEG data was obtained.

In Case-13, the electrode arrangement was changed six times, thus this case represents the highest number of channel configuration changes in the dataset. In this dataset, after removing the anonymous channels and selecting one of the six channels with the minimum count as a

reference, 18 channel regular EEG data was obtained. Subsequently, the EEG data to be processed were standardized to have either 18 or 23 channels for each case. The files within the dataset were adjusted to have the same channel names and lengths, making them ready for further processing. Consequently, channel numbers were reduced to 18 to ensure that every subject has the same channel configuration.

3.2. Pre-processing

During the recording of EEG data, both internal and external factors (noises) can affect the quality of the recordings. These noises can originate from the recording device itself or from the individual from whom the EEG data is recorded. As external factors, AC power lines (electrical noise) mainly originating from the EEG device are often encountered. Internal factors may include blinking, eye movements, heartbeat, and motion artifacts exhibited by the patient during the recording [25]. In order to mitigate these noises, bandpass filtering, removal of AC line noise, and finally, Independent Component Analysis (ICA) [26] have been employed to filter out internal factors.

To apply these filters, we import the EEG data into the EEGLAB[27] program. EEGLAB is a MATLAB toolbox distributed for processing data from electroencephalography (EEG), magnetoencephalography, and other electrophysiological signals. Figure 3.6 displays the time-domain plot of the 18 channels within the loaded EEG data. This raw data contains artifacts such as blinking, eye movements, heartbeat, and noise originating from the EEG device.

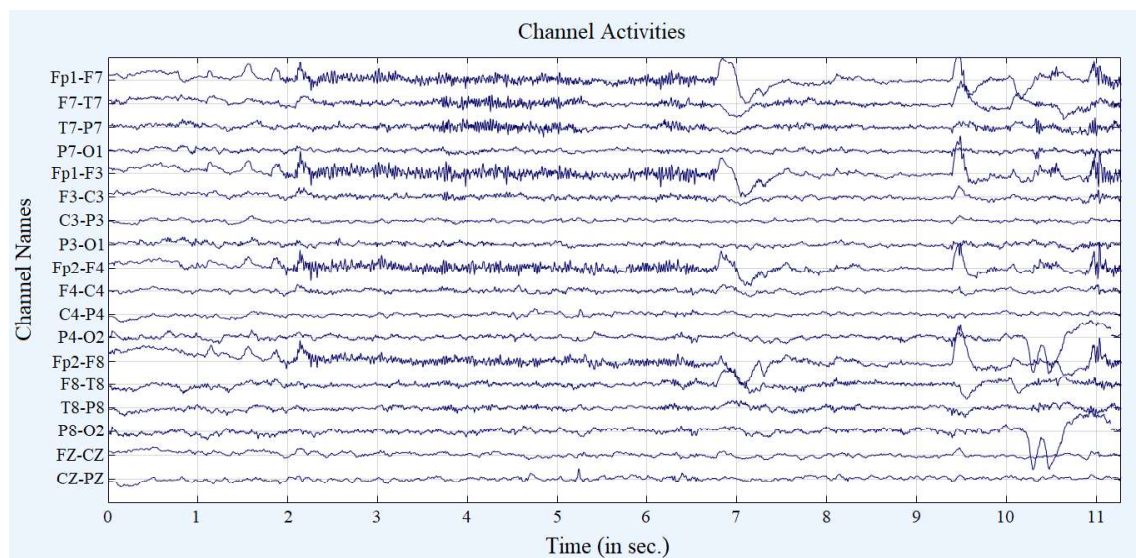


Figure 3.6. Visualization of the 18-channel EEG signals in EEGLAB

3.2.1. Bandpass filter

A bandpass filter is used to isolate or filter specific frequencies within a certain range or band [28]. It can be observed that the slowest oscillation from brain signals, the delta wave, ranges from 0.5 to 4 Hz, while the highest oscillation, the gamma wave, ranges from 30 to 140 Hz. Typically, when analyzing gamma waves, frequencies beyond 80 Hz are not extensively required. However, epileptic seizures are often observed in the gamma band as they exhibit high-frequency activity. Therefore, in order to encompass the entire range of brain signals in this study, data is filtered using a high-pass filter at 0.1 Hz and a low-pass filter at 120 Hz, which is a typical approach. The reason for starting the high-pass frequency at 0.1 Hz can be explained as follows: Event-related potentials (ERPs) are recordings of the brain's generated electrical field, which can involve any brain activity. As the cut-off frequency is increased, it is observed ERP latencies and the components of ERP disappear [29]. However, at the cut-off frequency of 0.1Hz, critical drops are observed in the indicated negative cases. Figure 3.7 illustrates the application of bandpass filtering to an EEG signal in our dataset.

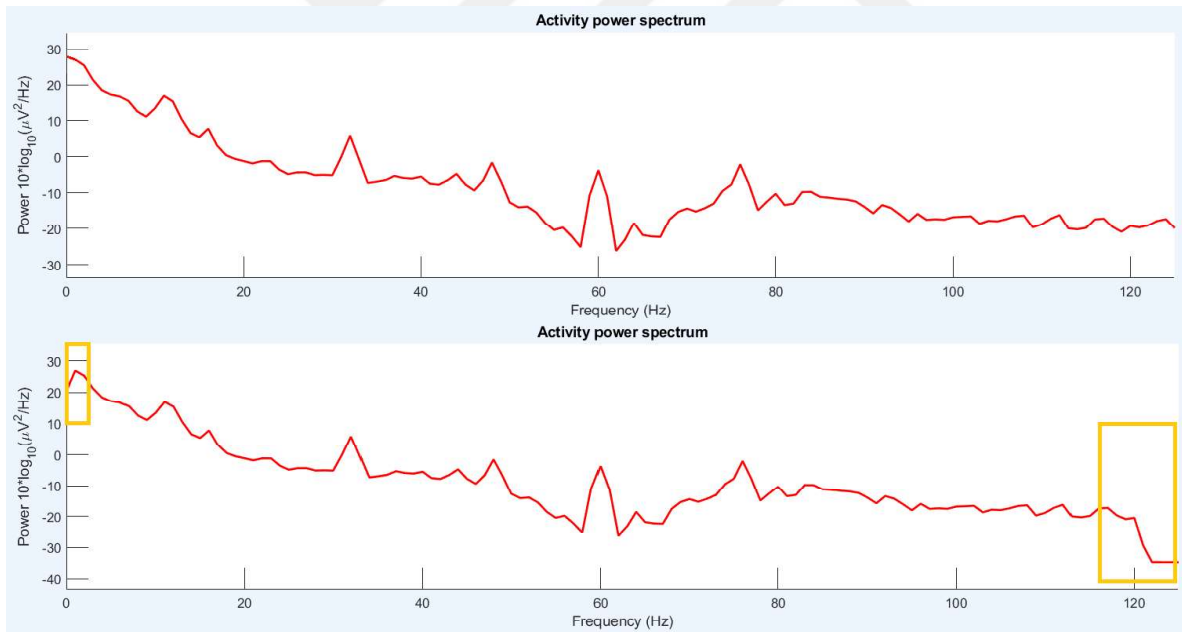


Figure 3.7. The changes in 0.1 and 120 Hz parts after bandpass filtering is shown in the power/frequency plot

3.2.2. AC Line noise filter

AC line noise, also known as electrical line noise, is a sinusoidal oscillation observed in raw EEG data at a frequency of 50/60 Hz. It is typically caused by the use of devices that operate on alternating current as a power source [30]. For instance, in EEG recordings, any sudden and noticeable changes at the 60 Hz point and its harmonics (120, 180, 240 Hz) are often referred to as AC line noise. The standard electricity transmission type in the United States is 120V/60Hz, while in European countries it is 230V/50Hz. Since the EEG data in this study was recorded in the United States, the line noise occurs at 60 Hz and its harmonics. In our dataset with a sampling frequency of 256 Hz, the EEG recordings are shown between 0.1-120 Hz after bandpass filtering. Therefore, the AC line noise, which we can easily observe within the recordings, should be examined at 60 Hz. Figure 3.8 demonstrates a sample for the removal of noise at 60 Hz in an EEG signal after bandpass filtering.

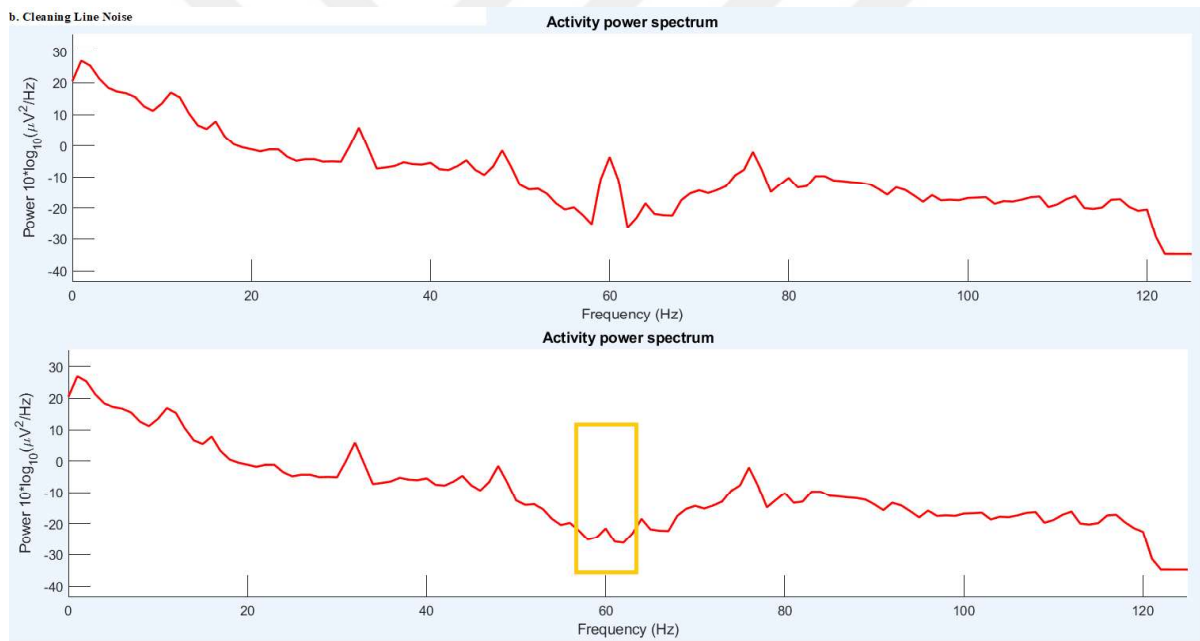


Figure 3.8. The filtered AC line noise at 60 Hz is shown in the yellow box

3.2.3. Independent Component Analysis (ICA)

EEG data is extremely sensitive to artifacts or extra noise from non-cerebral sources. While these artifacts have frequency overlap with the original signal, noise signals cannot be discriminated by filtering. In order to isolate artifacts and independent signals from the EEG signal, a statistical technique known as "Independent Component Analysis (ICA)" is used [26]. There are a couple of key assumptions to ICA.

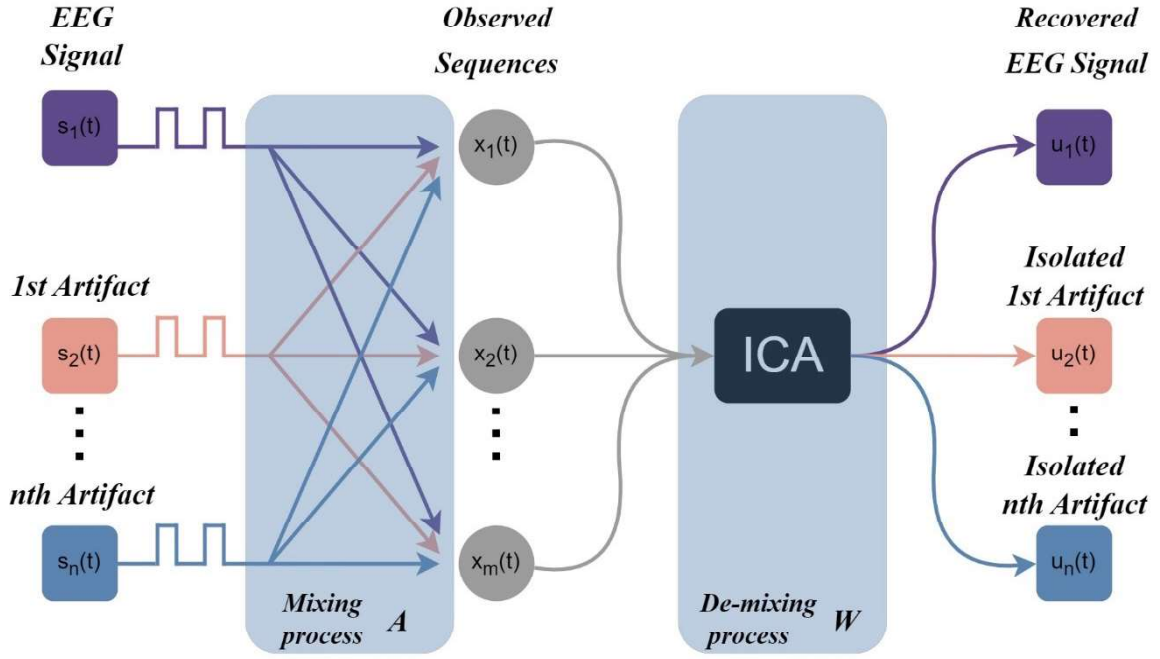


Figure 3.9. Transforming mixed signals into independent components using ICA

The first one is independent components are statistically independent. Statistical independence is typically defined as the joint distribution of two variables x and y being equal to the probability distribution x times and probability of y (Equation 3.1).

$$p(x, y) = p(x)p(y) \quad (3.1)$$

The second key assumption is that independent components are non-gaussian. Normally gaussian method allows us to do a lot of rigorous analysis but this is one of the instances where we need the independent components to be non-gaussian for discrimination of EEG signal to work. The third key assumption is the number of inputs is equal to the number of outputs. Figure 3.9 above shows the separation of a single EEG signal from $n-1$ artifacts. Let $s_1(t)$ be a single EEG source signal

and $s_2(t), \dots, s_n(t)$ be other EEG signal artifacts that are statistically independent, emitted by the n channels. Recorded all signals are denoted by $x_1(t), \dots, x_m(t)$, and can be expressed as a linear equation in Equation 3.2:

$$x_i = \sum_j a_{ij} s_j = a_{i1} s_1 + a_{i2} s_2 + a_{i3} s_3 + \dots + a_{im} s_n \quad (m=n=j) \quad (3.2)$$

Equation 3.2 explains that sample data $x_1, \dots, x_m$ are the linear combination of the n random variables $s_1, \dots, s_n$ mixed where, $i = 1, 2, 3, \dots, m$ is the signal index and $a_{i1}, a_{i2}, a_{i3}, \dots, a_{im}$ are the mixing coefficients. Equation 3.3 and 3.4 show matrix forms,

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}, \mathbf{s} = \begin{bmatrix} s_1 \\ \vdots \\ s_n \end{bmatrix}, \text{ and } \mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix} \quad (3.3)$$

The Equation 3.4 can be defined a vector-matrix notation,

$$\mathbf{x} = \mathbf{A}\mathbf{s} \quad (3.4)$$

where the $\mathbf{S}$ matrix is a non-observable original source variable and is therefore called independent components, $\mathbf{A}$ is a mixing matrix with elements $a_{ij}, i = 1, 2, \dots, n$ and $j = 1, 2, \dots, m$ and $\mathbf{x}$ is a mixture of elements in a random vector. ICA removes the artifacts while the recorded data and EEG data from each channel are preserved. On the other hand, the estimated solution for $s(t)$ is calculated as in Equation 3.5,

$$\mathbf{s} = \mathbf{A}^{-1}\mathbf{x} = \mathbf{W}\mathbf{x} \quad (3.5)$$

where, $\mathbf{W}$ is demixing matrix. By maximizing the statistical independence of the predicted components, ICA determines the independent components. Figure 3.10 shows the EEG signal after ICA is applied to the data recording of an epilepsy patient as a sample.

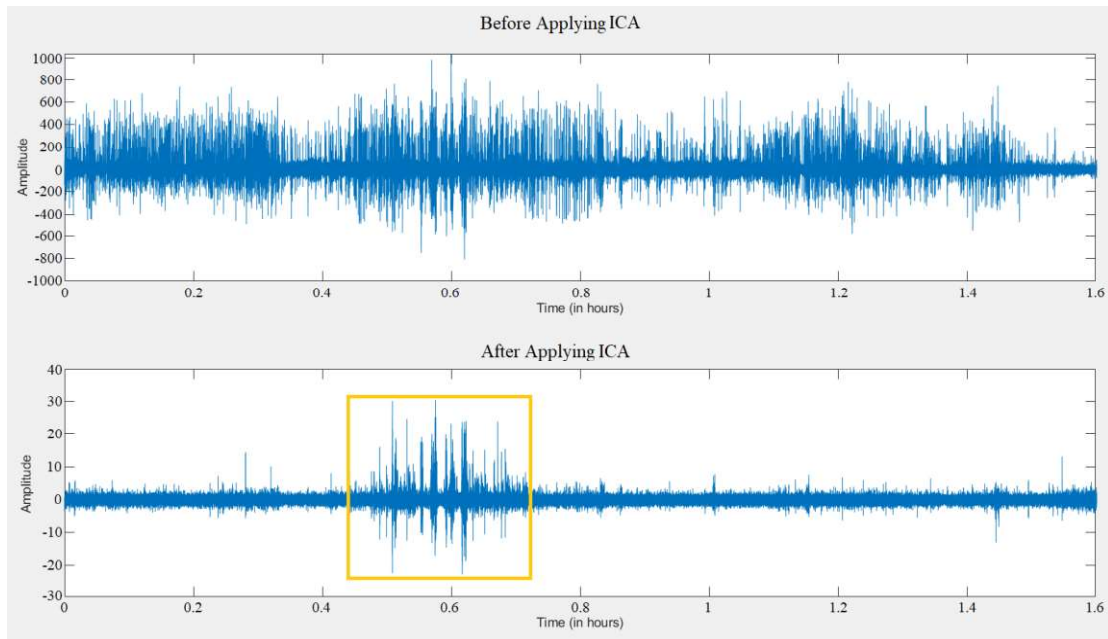


Figure 3.10. Applying ICA to EEG recording of an epilepsy patient

The yellow box in Figure 3.10 represents the region of the seizure and it is clearly seen that higher magnitudes due to the seizure appearance are preserved by ICA while the high-amplitude artifacts in other areas have been removed.

3.3. Feature Extraction

In order to successfully classify EEG signals, it is necessary to obtain signal features. Features are extracted by identifying the characteristic structures of EEG signals to differentiate them from each other. The obtained features should be the ones that best distinguish between different classes. Therefore, feature extraction is an important step in early seizure prediction because the classification success relies on well-selected features [31-33]. In this section, two different methods have been proposed for feature extraction from epileptic EEG signals.

3.3.1. Wavelet Transform (WT)

As a result of the evolution and advancements in signal processing techniques over time, the methods for analyzing EEG signals have also adapted to these changes. EEG signals are not composed of stationary and continuously repeating patterns; in other words, EEG signals are non-stationary, characterized by the unpredictability of their timing, values, and behaviors. Wavelet transform is a method employed for processing non-stationary signals such as EEG [34-36].

In wavelet transform, the window sizes vary, being broad for low frequencies and narrow for high frequencies. As a result, optimal time-frequency resolution can be achieved across all frequency ranges as illustrated in Figure 3.11.

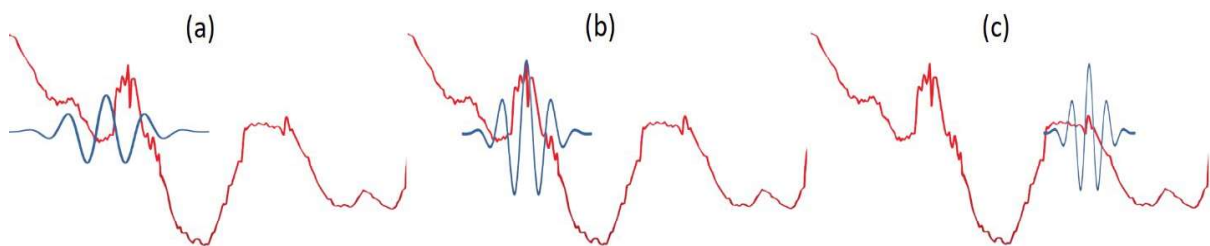


Figure 3.11. Wavelet hovering over the EEG signal. Variation of (a), (b) and (c) wavelets on the same signal on low, medium and high frequency signals, respectively

As can be seen in the figure above, the wavelet behavior is broad on low-frequency EEG signals. As the frequency range of the EEG signal increases, it reaches narrower window sizes (in the case of high frequency). Wavelet coefficients indicate details about the correlation between wavelet and the EEG signal [37]. In this work, the mother wavelet is selected as Morlet to calculate the spectrum for time-frequency analysis. Power spectrum in 5 frequency bands (delta, theta, alpha, beta, and gamma) are achieved for each EEG channel. The Equation 3.6 is utilized to extract the EEG features:

$$F_i^j = \frac{1}{N} \int_{f \in f_i} \int_{\tau} |S_j(f, t)|^2 d\tau df, \quad (3.6)$$

$$i \in (\delta, \theta, \alpha, \beta, \gamma), \text{ and } j = (1, 2, \dots, 18)$$

where, F_i^j is the extracted feature of j^{th} channel and i^{th} sub-band and $S_j(f, t)$ is the wavelet spectrum of the j^{th} channel from the EEG segment. Finally, the number of features for a particular subject obtained was 90 (18 channels x 5 bands). The algorithms used in the application of wavelet transform to the data are presented as Pseudocode-1 and Pseudocode-2 in Appendix-A. The experimental results are presented in Section 4.

3.3.2. Non-Linear Energy (NE)

Nonlinear Energy (NE) is described in 1990 [15], proposing an algorithm for extracting time-based features to calculate the "energy" of a signal and has been previously used as a method for extracting time-based features from EEG signals [38-40]. The Equation 3.7 describes the method:

Let x_n represent the signal samples that indicate the oscillatory body's motion:

$$x_n = A \cos(n\Omega + \phi) \quad (3.7)$$

where Ω is digital frequency in radian and $\Omega = 2\pi f$ where f is sampling frequency, ϕ is the randomly initial phase. Considering that there are two parameters in (3.7), A and ϕ it can obtain the values of these parameters from three samples of the signal x_n . For ease of use, it is decided to assess three samples of x_n placed next to one another and evenly spaced apart. After that, the signal values at these locations are provided by Equation 3.8:

$$\begin{aligned}
x_n &= A \cos(\Omega n + \phi) \\
x_{n-1} &= A \cos((n-1)\Omega + \phi) \\
x_{n+1} &= A \cos((n+1)\Omega + \phi)
\end{aligned} \tag{3.8}$$

Using trigonometric identity as a result it can be obtain in Equation 3.9:

$$A^2 \sin^2(\Omega) \approx A^2 \Omega^2 = x_n^2 - x_{n+1}x_{n-1} \tag{3.9}$$

From the equation above it can be seen that when the sampling frequency is high, then the nonlinear energy is directly proportionate to $A^2 \Omega^2$ as can be shown if a signal moves harmonically with the oscillation frequency Ω and amplitude A . As a result, the higher nonlinear energy values can be expressed as both shifted amplitude values and high frequency oscillations. The final form of the system is given in Equation 3.10:

$$NE(X) = \sum_{j=1}^N \sum_{i=2}^{255} (x^2[(i, j)] - x[(i+1, j)]x[(i-1, j)]) \tag{3.10}$$

where “ i ” is the number of rows in “*segment*” for one channel which is 256 and “ N ” is the total number of EEG channels. In Pseudocode-3, explains a function in the application of the Non-linear Energy method to the data. The expressions used in Pseudocode-3 can be found in Appendix-B.

3.4. Classification

The concept of classification involves the distribution of data into various predefined classes within a dataset. Classification algorithms learn the distribution pattern from a given training set and then attempt to correctly classify test data where the class is unknown [41,42]. The labels, which indicate the assigned classes for the data in the dataset, are used for determining the class of the data during both training and testing. Processing and analyzing large volumes of data are facilitated by machine learning methods, with deep learning being a popular subfield of machine learning [43-46]. Machine learning techniques aim to make predictions by finding the most suitable model for new data based on past data. In this study, four machine learning, namely the k-nearest neighbor (kNN), random forest (RF), support vector machine (SVM), and

J48 algorithm (C4.5), and one deep learning algorithm, 1-Dimensional Convolutional Neural Network (1D-CNN), were used to classify features extracted from EEG data. The classification results are presented in Section 4. The parameters of kNN and SVM classifiers were obtained using the commonly employed “Grid search” approach [47-51]. The detailed explanation of this technique is provided below.

3.4.1. Grid search technique

Grid search is utilized to obtain the optimal hyperparameters (adjustable parameters) of a model to achieve the best possible outcome [52-54]. The process involves creating a grid of hyperparameter values, training the model using each combination of these values, and subsequently evaluating the results obtained. The objective is to identify the specific combination of hyperparameter settings that yield the highest performance of the model on the validation set. Grid search is of great importance for model performance and time savings as it allows the selection of optimal hyperparameters. The objective here is to ensure that the classifier parameters are as common as possible for all cases. Thus, in the ***"Test Results"*** section in Section 4.2, the same parameters can be applied to patient data. For the database utilized in this thesis, includes a significant imbalance in the distribution of ictal, preictal, and interictal data, as observed in Table 3.3.

In order to enable a more effective and efficient performance of parameter grid search across all subjects, a separate dataset was created by maintaining a balanced proportion of the durations of these three classes from all subjects in the dataset. Due to the limited number of available seizure (ictal) data across all subjects, all the seizure data were included. Afterwards, preictal and interictal data were selected randomly for each subject, for a duration equal to the ictal length. Therefore, these sampled data were aggregated from all subjects and merged into a single matrix. The data generated for grid search, obtained randomly and in a balanced manner from each subject, are presented in Table 3.6.

Table 3.6. Durations of ictal, preictal, and interictal instances used for grid search

	Total duration of Instances (in seconds)			
Case	Interictal	Preictal	Ictal	Total (hh:mm:ss)
All cases	11400	11400	11400	09:30:00

A balanced dataset of 9 hours and 30 minutes is created by randomly extracting preictal and interictal segments from all subjects, as there is a 3-hour and 10-minute ictal data available for each subject. Thus, a total of 34,200 seconds of data is obtained, ensuring a balanced distribution across the various segments. When considering the total duration, the 950-hour EEG data from Table 3.3 has been reduced to 9 hours, resulting in a shrinkage of 1/55 ratio. Thus, while finding optimal classifier parameters faster, a stable structure has been established by including data from all subjects. In conclusion, this dataset was utilized to determine the optimal parameters for all classifiers.

3.4.2. k-Nearest Neighbour (kNN)

The kNN algorithm is a machine learning algorithm that classifies a given data point based on its proximity to previously observed data points. The main objective of kNN is to determine the class to which a data point belongs by examining its features [55,56]. To learn the class of a target data point, we look at its k nearest neighbors and assign the label based on the majority class among its neighbors. Figure 3.12 demonstrates the classification process by considering the three closest neighbors ($k = 3$) to the "x" symbol. In this example, for $k = 3$, the classification result yields the purple-colored ball, and the class of the "x" mark is determined to be "purple color". In this study, kNN has two fundamental parameters which are " k " and "*distance metrics*"

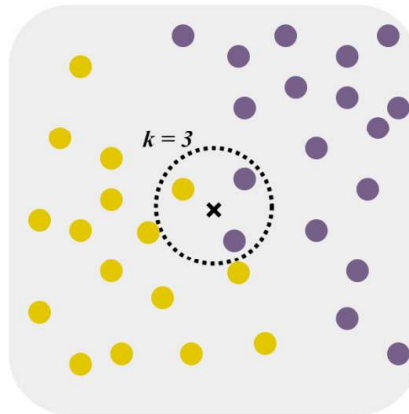


Figure 3.12. Enclosure of the three closest neighbors around the specified point within a circle

The data points to be classified are ranked using the distance between the data which is tested and the training samples. Finally, labels of the majority of the closest k training samples are used for making a decision. In this thesis, odd k values between *1 to 50 (1,3,5,7....49)* and

Minkowski, *Manhattan*, and *Euclidean* distances [57], as values of the distance metric, are utilized for grid search. The best parameters "*k*" and "*distance metric*" within kNN were obtained using Pseudocode-4 in Appendix-C. After applying Pseudocode-4 to the dataset, the most optimal parameter values and their comparisons are explained in Section 4.

3.4.3. Support Vector Machine (SVM)

Support Vector Machines (SVM) are commonly used supervised learning methods in classification problems [58,59]. In order to classify objects, we represent each object as a point in a multi-dimensional space with various features. These features are essentially the coordinates of the point. The SVM (Support Vector Machine) classification technique aims to find a hyperplane (a line in 2D or a plane in 3D) that effectively separates the two categories of objects. The objective is to have all the points of one category on one side of the hyperplane and all points of the other category on the other side. While there might be multiple hyperplanes that can achieve this, SVM seeks to identify the one that maximizes the distance to the nearest points from each category. This distance is known as the margin, and the points that lie exactly on the margin are referred to as the supporting vectors. The SVM illustration is described in Figure 3.13.

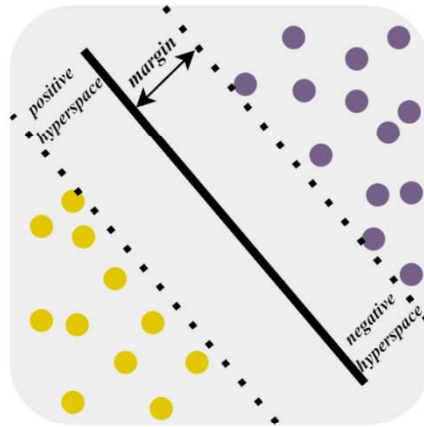


Figure 3.13. The representation of the Support Vector Machine (SVM) algorithm on data

The black line used for classification in Figure 3.13 represents the hyperplane, while the adjacent dashed lines are referred to as the decision boundaries. The distance between the dashed lines provides us with the margin. In order to ensure the continuation of accurate classification when new data is introduced to the dataset, the hyperplane that maximizes the margin is preferred. Multiple hyperplanes can be drawn for a given dataset, but the one that yields the widest margin is selected. In essence, a wide margin implies an increased separation

between classes, thus the best discriminative line is chosen. On the other hand, since this study involves a three-class classification, it is necessary to make modifications based on SVM. We incorporate the One-vs-Rest (or One-vs-All) method as a modification. This method can be used for creating a multiclass SVM classifier. Binary classification is performed by considering one output as positive while treating the other two as negative, and this process continues until it is applied to all outputs. As a result, the three different outputs are compared with each other for classification. The SVM classifier comprises three fundamental parameters: "*gamma*", "*C*" and "*kernels*". In our study, grid search was conducted using the values "*linear*", "*polynomial*" and "*radial basis function (RBF)*" for the kernel, gamma values are "*1, 0.1, 0.01, 0.001 and 0.0001*" and C values are "*0.1, 1, 10, 100 and 1000*". The idea behind the SVM is that it does not actually need to know what each point is mapped to under non-linear transformation. The only crucial thing that it needs to know how each point (x) compares to each other data points (x') after applying nonlinear transformation. Mathematically this corresponds to taking inner product $f(x)$ and $f(x')$. And the quantity is called kernel function.

This formula is shown in Equation 3.11:

$$f(x)^T f(x') := k(x, x') \quad \text{where the "f" is function and "k" is the kernel.} \quad 3.11$$

Kernel parameters:

- **Linear** : Linear kernel gives a flat decision boundary (hyperplane). which is a simple decision separator. This is not good enough to separate the data properly for our case. The linear function and kernel equation are illustrated in Equation 3.12.

$$f(x) = x \quad , \quad k(x, x') = x^T x' \quad 3.12$$

- **Polynomial** : Polynomial kernel takes into account the original features of our dataset just like that linear kernel. On top of that it also considers their interaction. As a result, this kernel gives a curved decision boundary. The function and kernel equation are shown in Equation 3.13.

$$f(x) = (x_1, x_2, x_1 x_2, x_1^2, x_2^2) \quad , \quad k(x, x') = (1 + x^T x')^2 \quad 3.13$$

- **RBF:** RBF kernel is one of the most widely used kernels due to its similarity to the Gaussian distribution. Sometimes it is possible to give a kernel function for which is hard or even impossible to find the corresponding transformation. This kernel has infinite dimensional thus it is impossible to use directly in a computer program. But the kernel expression is incredibly simple and it is shown in Equation 3.14. According to the formula we can change the gamma parameter to make boundary smooth or rough. Low gamma values result in the creation of a smooth boundary, whereas high values lead to the formation of a rough boundary.

$$f(x) = - (None) \quad , \quad k(x, x') = e^{-\gamma \|x - x'\|^2} \quad 3.14$$

Gamma parameter:

As mentioned in RBF, the variation of gamma values in the formula results in changes in the decision boundary. These values should be given between 0 and 1. If the value approaches 0, the decision boundary becomes flatter, while approaching 1 leads to a higher-degree curve. Interpreting these values, a small gamma value implies that distant points are considered similar, resulting in more outputs being grouped together and having fewer accurate decision boundaries. On the other hand, a larger gamma value causes points to be closer to each other. However, excessively large gamma values lead to overfitting.

C parameter:

This parameter manipulates the regularization rate applied to the data. Low C values indicate that the generated model is more prone or tolerant to errors. Conversely, larger values of C ensure a better fit of the model. However, high C values can lead to overfitting.

The grid search performed in Python to determine the optimal selection of the above parameters is illustrated with Pseudocode-5 in Appendix-D. After applying Pseudocode-5, the most optimal parameter values and their comparisons are explained in Section 4.

3.4.4. Random Forest (RF)

The random forest algorithm is one of the supervised classification algorithms that can be used for both regression and classification problems [60]. The algorithm aims to improve the classification performance by generating multiple decision trees. The RF algorithm creates various models by training each decision tree on a different subset of observation examples and combines them to form a classification. The process involves selecting the highest-scoring outcome among the classification decision trees [61]. The model is shown in Figure 3.14. As the number of trees increases, our chances of obtaining a precise outcome also increase. The fundamental difference between the decision tree algorithm and the random forest algorithm lies in the randomness of finding the root node and splitting the nodes in the random forest algorithm.

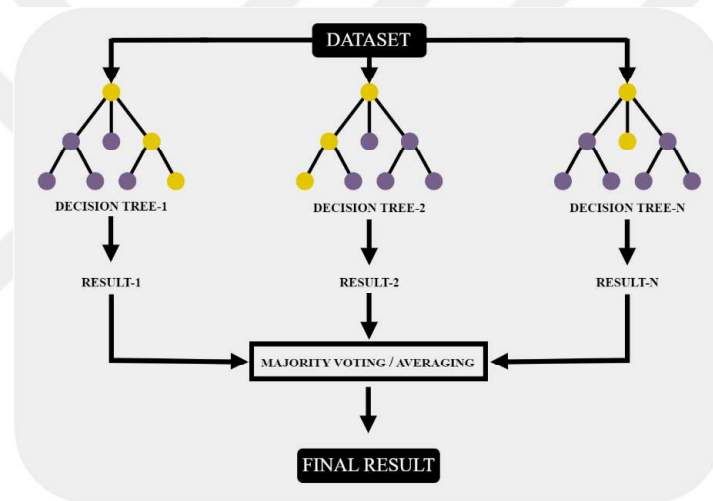


Figure 3.14. Random forest is a classifier that consists of multiple decision trees.

3.4.5. J48 Algorithm (C4.5)

J48, a widely used machine learning algorithm in the WEKA program, is an extension of the ID3 algorithm [62]. It is an algorithm that generates a decision tree created by C4.5 (an extension of ID3). The C4.5 tree is an enhanced version of the ID3 tree and is also known as a statistical classifier. The main difference between the C4.5 tree and the ID3 tree is the use of normalization. In the ID3 tree, entropy (or information gain) is calculated to determine the decision points. In the C4.5 tree, entropy values are maintained as ratios. Additionally, it is possible to move subtrees to different levels based on their access frequencies. Another

difference in the C4.5 tree is the pruning process, which is not present in the ID3 approach. The step-by-step process of these differences is as follows: at each step, all features are examined, and the normalized information gain for each node in the decision tree, and then a sub-list is feature is calculated. The feature that provides the highest information gain is moved to the decision created under this new decision node (a subtree).

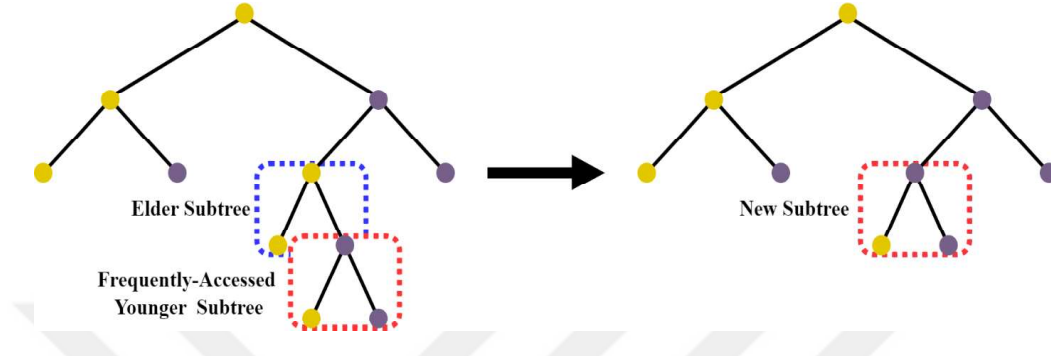


Figure 3.15. The relocation of the tree with excessive information gain.

The two classifiers with the aforementioned decision tree algorithm infrastructure share almost identical optimal parameters. The key distinguishing feature between them lies in the mathematical calculation criterion. While random forest utilizes "**entropy**" as its calculation criterion, J48 employs the "**gini**" criterion. The RF and J48 classifiers comprise three fundamental parameters: "**random states**", "**n estimators**" and "**max depth**". The random states values are "**0, 50, 100**", the n estimators values are "**64, 128, 256, 512**" and the max depth values are "**4, 8, 16, 32**". Additionally, the grid search was conducted using the parameters "**entropy**" and "**gini**" for the criterion. The following paragraph explains a brief explanation of criterion and the roles of the parameters:

- **Random states** : The dataset is randomly split into training and test sets, ensuring different splits each time. If a random state value is specified, it will consistently produce the same split. This value must be a non-negative number and can be thought of as an identifier. Therefore, the specific digits do not hold significant importance.
- **n-estimators** : The dataset is divided into n subsets, and decision trees are constructed from each subset. During the splitting process, a complete separation (without replacement) does not occur. In other words, the data present in subset a may also exist in subset b.
- **max depth**: The maximum depth of the resulting tree is determined, which represents the number of rules to be initially examined in the decision tree. While the number of branches

is directly proportional to complexity, excessive values can lead to overfitting. For example, if a clean (correct) rule is reached from a branch, there is no need to revisit that rule.

- ***criterion-entropy:*** Entropy is a measure of randomness in the information being processed. The higher the entropy, the more difficult it is to derive outcomes from the given variable.
- ***criterion-gini:*** The Gini index can be described as a cost function used to evaluate the splits in a dataset. It is calculated by subtracting the sum of the squares of probabilities for each class from one. It favors larger subsets and is easy to implement, whereas information gain supports smaller subsets with different values. Choosing a feature with a lower Gini index is preferred.

In Pseudocode-6, a grid search is implemented to find the optimal parameter values for the two classifiers mentioned above. The expressions used in Pseudocode-6 can be found in Appendix E.

3.4.6. One Dimensional Convolutional Neural Network (1D-CNN)

Artificial neural networks aim to acquire learning and adaptation capabilities similar to the human brain [63]. Deep Learning can be defined as an architecture that consists of multi-layer artificial neural networks at each layer [64]. What sets Deep Learning apart from other artificial intelligence and machine learning methods in the literature is its ability to handle data and complex problems due to its numerous processing units. While other machine learning methods require pre-given feature information to be provided to the system, deep learning extracts features on its own. Due to its complex architecture and extensive computational capabilities, artificial neural networks require hardware with high computing power. This need is met by utilizing Graphics Processing Units (GPUs) in computers. Despite the long-standing use of artificial intelligence and machine learning methods, the powerful and concurrent computation achieved by GPU processing, coupled with the ability of artificial neural networks to process large volumes of data, has distinguished deep learning from other methods. Numerous studies have been conducted in the literature using deep learning, including image processing, signal processing, natural language processing, healthcare diagnostic systems, pattern recognition, robotics, biomedical applications, finance, advertising, SEO search engine optimizations, autonomous vehicle systems, etc.

In this study, a deep learning algorithm called a Convolutional Neural Network (CNN or ConvNet) was employed [65-68]. If we consider each input in a CNN architecture as a neuron, it can be likened to the frontal lobe of the brain, which is responsible for processing visual stimuli. This analogy treats the visual field as a whole rather than fragmented pieces. Figure 3.16 illustrates the general form of the CNN algorithm.

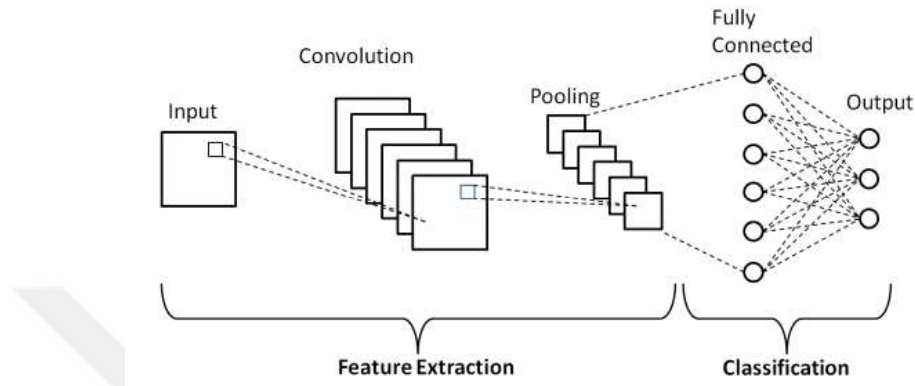


Figure 3.16. General view of CNN algorithm

3.4.6.1. *The dimensions of the CNN*

CNN algorithm is known for producing good performance results in image processing, and it is extensively used in signal processing [69-73]. The CNN architecture is classified dimensionally as 1D-CNN, 2D-CNN, and 3D-CNN. The dimension of the architecture is influenced by the number of directions in which the kernel in the convolutional layer can move. EEG signals, sound, and text data are graphically recorded over time and magnitude. Hence, it is understood that the kernel can only move in the temporal direction (1 dimension). Some studies in the literature that involve EEG data have classified their features using the 1D-CNN algorithm [74-77]. In this study, features extracted from EEG data using wavelet transform and non-linear energy methods were classified using 1D-CNN. Figure 3.17 represents the 1D-CNN, 2D-CNN, and 3D-CNN kernels. The part (a) represents EEG data. The one-dimensional kernel in this case covers all the channels specified in the rows and moves only along the time dimension. The kernel gathers information from the channels at each advancement step. Similarly, part (b) represents image data. Here, there is a two-dimensional kernel movement, and the depth of 3 units represents the RGB colors, which are the fundamental building blocks

of the image, namely "Red," "Green," and "Blue." In part (c), a matrix is depicted where images are stacked together. This can represent a video or an ultrasound matrix.

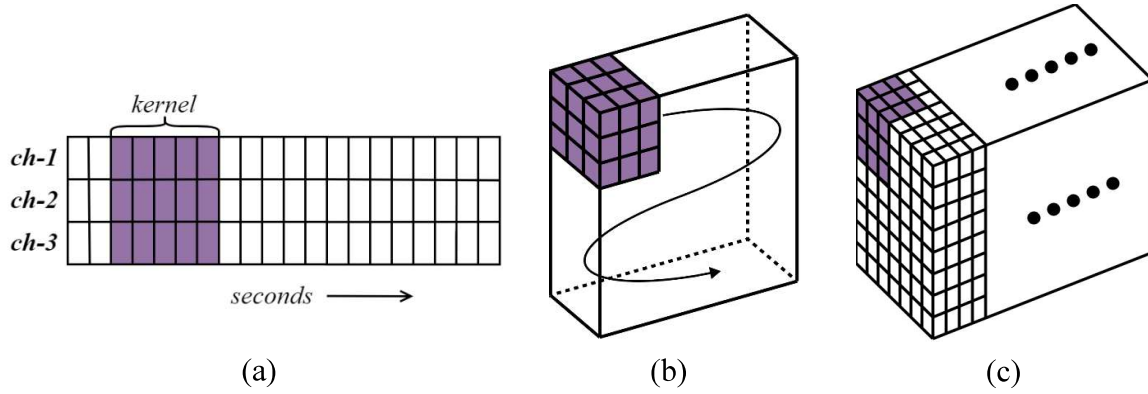


Figure 3.17. Representation of the Conv1D, 2D and 3D data with their kernels. (a) In Conv1D, kernel slides along one dimension. (b) While Conv2D kernel slides along two dimensions, (c) Conv3D kernel slides along three dimensions

3.4.6.2. Activation functions

Right after taking signal data or an image as input, each layer of the neural network is generally passed through an activation function. Activation functions are required to introduce non-linear real-world characteristics to artificial neural networks (Figure 3.18). Without applying an activation function, the output signal would be a simple linear function. Linear functions are only first-degree polynomials. Functions that have multiple degrees are referred to as non-linear functions. Artificial neural networks are designed as universal function approximators and are expected to work towards this goal. This means they should be able to compute and learn any function. Non-linear activation functions enable networks to have stronger learning capabilities. A neural network without an activation function will exhibit limited learning power, behaving more like a linear regression. However, in our study, we aim to teach our neural networks to learn non-linear data. This is because complex real-world information, such as EEG data, will be provided to the constructed neural network for learning. Multi-layer deep neural networks can thus learn meaningful features from the data.

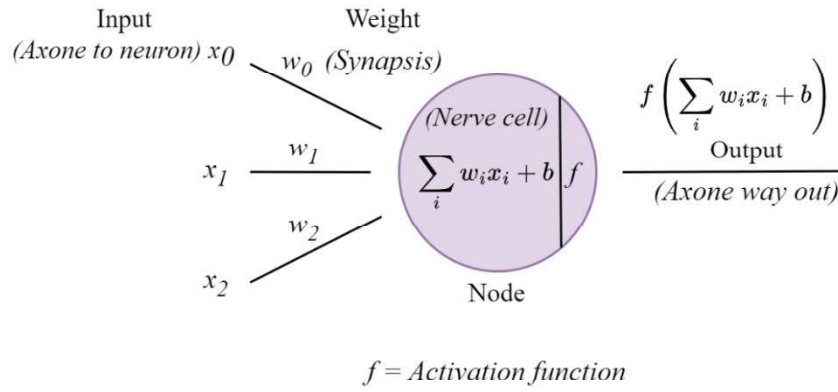


Figure 3.18. Demonstration of an activation function in Neural Networks Architecture

Activation functions directly impact the accuracy of a deep learning network because they provide access to changing the designated coefficients and weights within the network. The fundamental operation is to determine whether a neuron (or information) should be activated or not [78]. During the prediction process, deciding the importance of a neuron's input to the network enhances algorithm performance (generating a feature map that increases network stability) and filters out irrelevant information, preventing data clutter. In this study, the "**ReLU**" and "**Softmax**" activation functions utilized have been briefly described.

3.4.6.2.a. **ReLU (Rectified Linear Unit) activation function**

This function is one of the most commonly used activation function in deep learning by researchers worldwide. It takes values in the range of ReLU $[0, +\infty)$ (Figure 3.19). Let's consider a large neural network with many neurons. Sigmoid and Hyperbolic Tangent were causing almost all neurons to fire/activate in a similar manner. This dense activation requires a significant computational load. Sparse activation, where only some neurons are active, is desired for efficient computation. ReLU helps achieve this [79,80]. The fact that it takes the value of 0 in the negative axis also implies that the network will run faster. The reduced computational load compared to sigmoid and hyperbolic tangent functions has made ReLU preferable in deep networks. However, the problem arises from the fact that the derivative of the zero region, which contributes to the computational speed, is also zero, indicating that learning does not occur in that region.

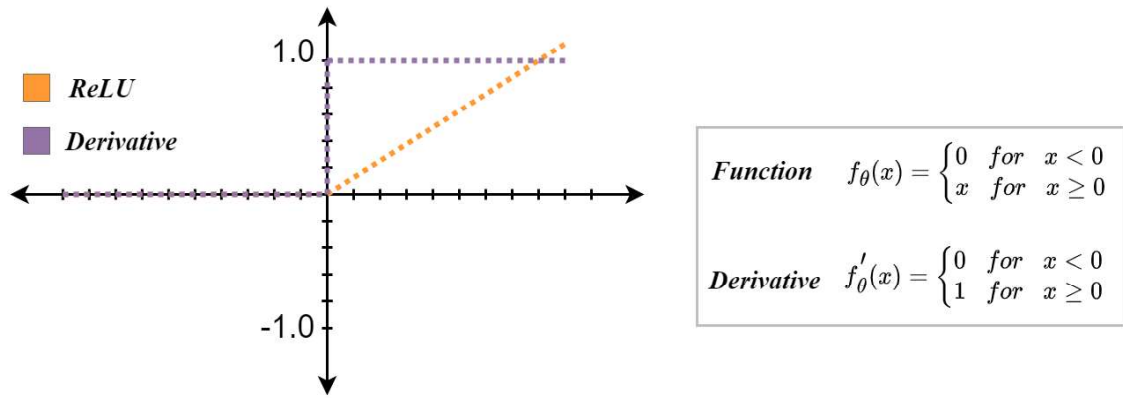


Figure 3.19. Visual illustration of the ReLU activation function and its derivative

3.4.6.2.b. *Softmax function*

It has a structure very similar to the sigmoid function. The most important difference is that it is particularly preferred in the output layer of deep learning models, especially in cases where more than two classes need to be classified [81]. It ensures the determination of the probability of the input belonging to a specific class by generating values in the range of 0-1. In other words, it enables a probabilistic interpretation.

3.4.6.3. *The intuition of the 1D-CNN*

ConvNet is a sequence of layers. Each layer of a ConvNet transformation is the activation of another through a differentiable function. A CNN contains three layers: a “*Convolutional Layer*”, a “*Pooling Layer*” and a “*Fully Connected (FC)*” layer, respectively. The complexity of the CNN increases from the first layer to the last layer. Because of this increased complexity, CNN can analyze layer parts and more complex features of the image until it successfully recognizes the entire object.

3.4.6.3.a. *Convolutional layer*

Most of the computations in CNN, which is known as the foundational layer, take place in this layer. This layer contains a filter or kernel that moves along the signal, checking for the presence of a feature [82,83]. In this way, the kernel scans the entire signal by repetition. After each iteration, a feature map is obtained by taking the dot product of the values in the filter with the corresponding region of the image. Consequently, the CNN interprets the signal based on this

newly generated map (matrix). To gain a better understanding of these layers, an example was presented using the information of subject-01 (case-01) from the dataset employed in the study. As a result of the wavelet transform applied to Case-01, a matrix of dimensions 142000x90x1 was obtained, representing the extracted features.

In this study, the data was passed through “*one input*”, “*three hidden*” and “*one output*” 1D-CNN layers. In the first convolutional layer, a 5x5 kernel was applied 90 times. The stride (each step) of 1 units, while the padding was set to 0. Based on this parameters the formula of volume of size: $F = 5$, $P = 0$ $W_1 = 90$, $H_1 = 142000$ and $S = 1$. Hence, after applying the Equation 3.15-16.

$$\text{Floor}\left(\frac{H_1 - F + 2P}{S} + 1\right) = \text{Floor}(141996) = \mathbf{141996} \quad 3.15$$

$$\text{Floor}\left(\frac{W_1 - F + 2P}{S} + 1\right) = \text{Floor}(86) = \mathbf{86} \quad 3.16$$

The activation size in the subsequent layer of the network will be [141996x86x90]. The value of 90 in the dimension corresponds to the number of filters. This situation is visualized and explained in Figure 3.20.

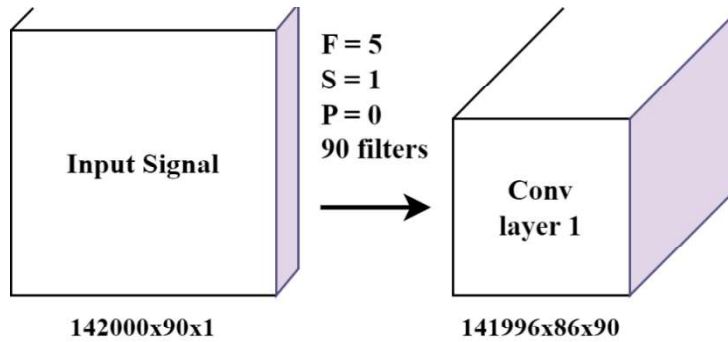


Figure 3.20. Conversion of [142000x90x1] a matrix to [141996x86x90] an activation after convolutional layer

3.4.6.3.b. Pooling Layer

This layer is used between consecutive convolutional layers. It reduces the number of weights and computations, thereby decreasing the computational complexity. Although this reduces the number of parameters and may result in some information loss, it improves the efficiency of the CNN and helps control overfitting. The filters used in this layer operate on average or maximum values [84,85]. The most common form for pooling filter 2x2 with a stride of 2 down-samples is used. Based on these parameters the formula of volume of size: $F = 2$, $W_2 = 86$, $H_2 = 141996$ and $S = 2$. Hence, after applying the Equation 3.17-18.

$$\text{Floor}\left(\frac{H_2 - F}{S} + 1\right) = \text{Floor}(70998) = \mathbf{70998} \quad 3.17$$

$$\text{Floor}\left(\frac{W_2 - F}{S} + 1\right) = \text{Floor}(43) = \mathbf{43} \quad 3.18$$

After the application of the pooling layer, the size of the data has become [70998x43x90]. This situation is illustrated and explained in Figure 3.21.

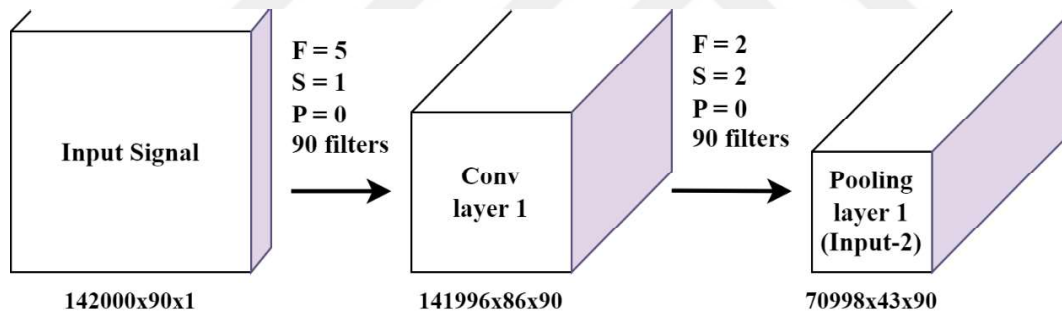


Figure 3.21. Conversion of [141996x86x90] a matrix to [70998x43x90] an activation after pooling layer

3.4.6.3.c. Fully Connected (FC) Layer

This layer, known as the fully connected layer, is the final part in CNN where the classification process takes place based on the extracted features, producing the output [86]. As the name "fully connected" implies, it is a one-dimensional array where all nodes are connected to the activation nodes of the next layer. Figure 3.22 illustrates the stages of all layers with softmax. Finally, after creating a long vector, the softmax function predicted the final output. Thus, accuracy results were obtained by comparing the outputs from a single vector (Section 4).

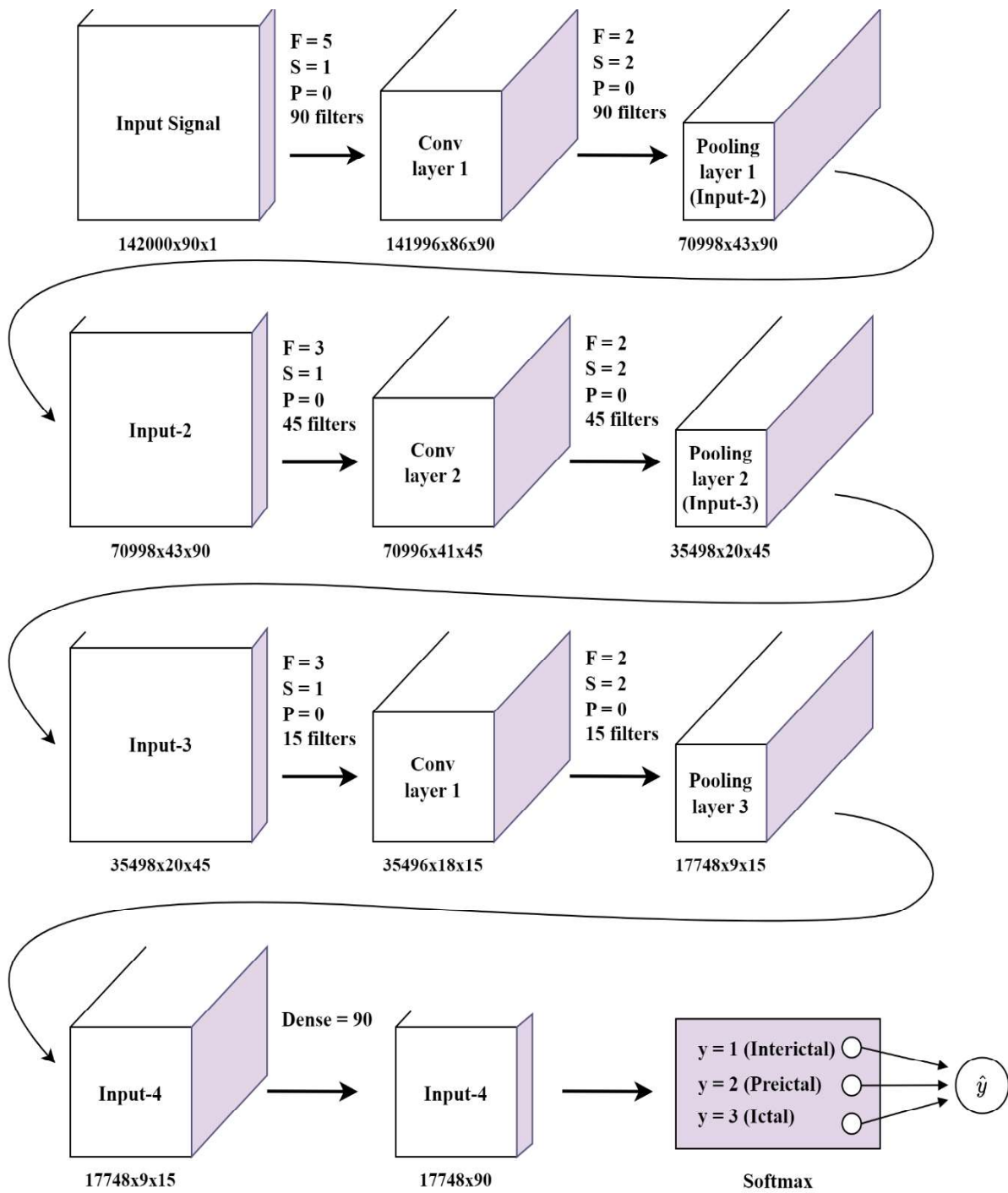


Figure 3.22. General view of the 1D-CNN with 1 input, 3 hidden and 1 output layers for WT

3.4.6.4. The creation of the model

The preparation of the 1D-CNN model in this section can be achieved based on the aforementioned headings. Some of the parameters used in the model construction were obtained through grid search. These parameters represent the best values obtained from the grid search. The grid search results for these parameters are tabulated in Section 4. The WT and NE model summaries are designed with the best parameters is illustrated in Figure 3.23 and Figure 3.24 respectively.

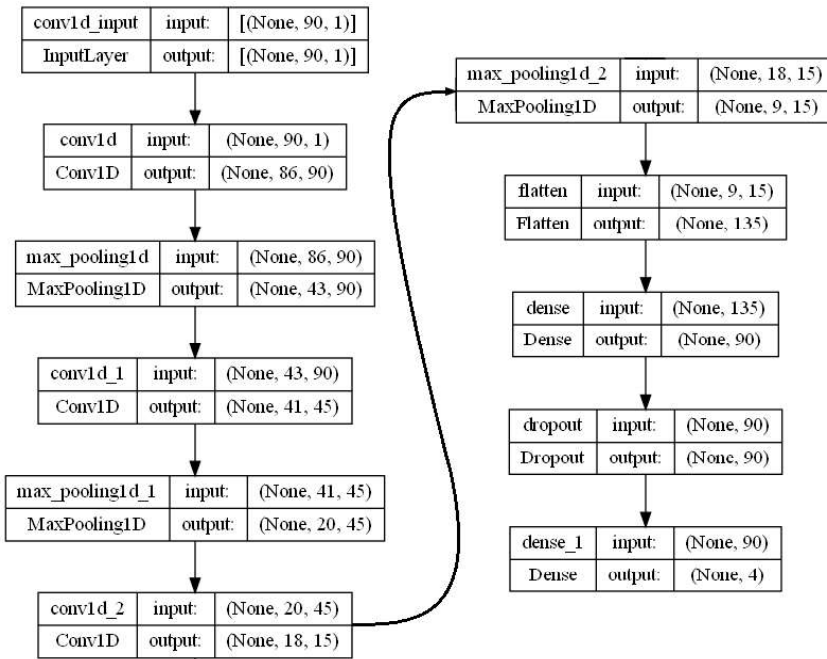


Figure 3.23. Model summary of the WT

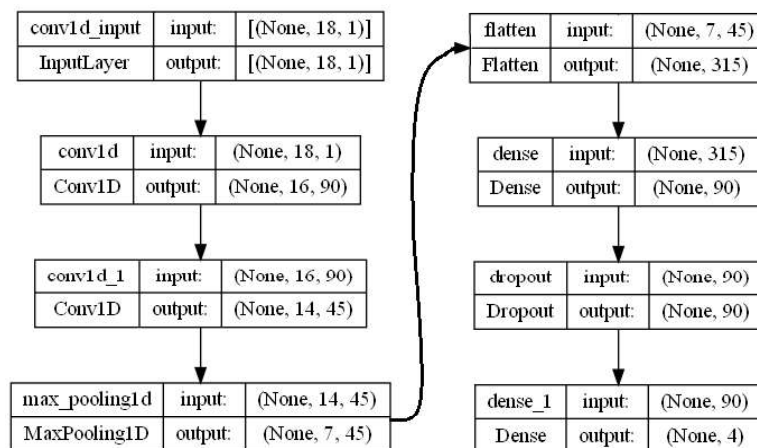


Figure 3.24. Model summary of the NE

For the WT model, after the Input layer, it has been passed through three hidden layers as depicted in Figure 3.22 and 3.23. The Input layer and two of the hidden layers employ 1D-CNN, while the final hidden layer is followed by a classical neural network (ANN) layer with 90 filters. The following parameters have been applied to the hidden layers:

For the input part:

- Convolutional layer: $F = 5$, $S = 1$, $P = 0$ and number of filters is 90.
- Pooling layer: $F = 2$, $S = 2$ and $P = 0$

For the first hidden part:

- Convolutional layer: $F = 3$, $S = 1$, $P = 0$ and number of filters is 45.
- Pooling layer: $F = 2$, $S = 2$ and $P = 0$

For the second hidden part:

- Convolutional layer: $F = 3$, $S = 1$ and $P = 0$. and number of filters is 15
- Pooling layer: $F = 2$, $S = 2$ and $P = 0$

For the third hidden part:

- ANN with 90 filters

For the NE model, after the Input layer, it has been passed through two hidden layers as shown in Figure 3.24. The Input layer and one of the hidden layers utilize 1D-CNN, while the other hidden layer is followed by a classical neural network (ANN) layer with 90 filters. The following parameters have been applied to the hidden layers:

For the input part:

- Convolutional layer: $F = 3$, $S = 1$, $P = 0$ and number of filters is 90.

For the first hidden part:

- Convolutional layer: $F = 3$, $S = 1$, $P = 0$ and number of filters is 45.
- Pooling layer: $F = 2$, $S = 2$ and $P = 0$

For the second hidden part:

- ANN with 90 filters

When examination of the above models, the input shape in the first layer indicates the number of columns and the depth of the data, independent of the number of rows. Since the initial EEG data is initially two-dimensional, but the CNN operates in a three-dimensional manner, the value should be specified as [*rows, columns, 1*]. Each layer consists of parameters such as the number of filters, kernel size, stride, and activation function. While WT has the number of filters is set as 90 in the first layer, 45 in the second layer, and 15 in the final layer, NE has the number of filters is set as 90 and 45 in the final layer. As this value increases, the depth of the data also increases. The choice of 90 filters in the first layer corresponds to the [*channel x brain wave*] count obtained from the wavelet transform. Similar parameters (Figure 3.24) are obtained for non-linear energy features and the model is slightly different, but the filter values remain the same. For all layers, the values are [*15, 45, 90*] for the wavelet transform and [*45, 90*] for the non-linear energy. The number of channels does not necessarily have to be the same as the number of filters. As previously mentioned, the depth of the network is represented by the number of layers. It has been observed that applying 90 filters instead of 18 filters for NE yields greater benefits. Kernel is used as 3x3 and 5x5 in our models. Generally, 1x1 kernel size aims to reduce the number of channels, which is not preferred. On the other hand, kernels such as 2x2 and 4x4 are not suitable for dividing symmetrically the previous layer pixels around the output pixel. For this reason, 3x3 or 5x5 are generally used. It is possible to utilize larger kernel sizes; however, due to the increase in processing time associated with enlarging the kernel size, a *5x5 and 3x3 kernels* have been employed in WT and NE models. The strides, which refer to the kernel steps, should generally be smaller than the kernel size. Therefore, in this model a *stride of 1 unit* has been deemed suitable. The pooling sizes remained as *2x2* for both transformations. The activation function *ReLU* (mentioned in Section 3.4.6.2.a), was applied in each 1D-CNN layers for both models. In the models, *dropout* was applied before softmax activation function. This process involves randomly dropping out (eliminating) a certain percentage of neurons in the neural network, preventing overfitting, and improving performance. The values given to dropout range from 0 to 1. A value of 0 represents the least amount of dropout, while a value of 1 represents the maximum dropout rate. In this study, dropout was set to *0.5*. The activation function of the classifier in the final layer was set to softmax in order to classify 3 class (interictal, preictal and ictal). The *optimizer* parameter enables the use of optimization algorithms to improve the weight values. In this study, the

Adaptive Moment Estimation (*adam*) algorithm was used, which dynamically updates the learning rate for each parameter in real time. The *loss* parameter calculates the difference between the obtained values and the true values after each training iteration. Categorical cross-entropy (*CCE*), based on the concept of entropy, is a measure of the difference between multi probability distributions for a given set of random variables or events. Mean square error (*MSE*) and mean absolute error (*MAE*) were used as the loss function or cost function. Generally, MSE tends to develop a model that reduces the impact of outliers. However, MAE can be used to obtain more interpretable values. In Section 4.1.5, the CCE loss function, which was found to be the most successful through grid search, has been employed for two models. The *metrics* parameter performs an evaluation process after each epoch during the training stage to assess the results. The commonly used metric *accuracy* is applied to examine the performance of the model.

In the next step, the model needs to be fitted in order to train the generated training values. Prior to applying the fit method, the dimensions of the data have been modified to [100 x segment x 90]. This allows for easier separation of the data into train-test and train-validation sets. In this study, the dataset has been set to an 80% train and 20% test ratio. For the 80% of data allocated for training, 10-fold cross-validation has been applied. Figure 3.25 shows a visual representation of the train-test and train-validation splits.

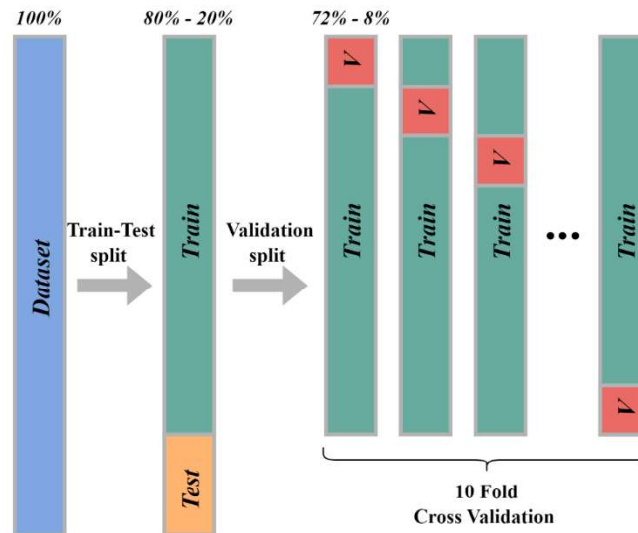


Figure 3.25. Train-test and train-validation split

The fit method enables the model to be trained through validation. Within this method, a grid search has been conducted (Section 4.1.5) on the values of two fundamental parameters that affect the classification outcome: *batch size* and *epochs*. The *batch size* parameter allows input data to be processed in groups rather than individually during training or testing. Batch size values were tested as **32, 64, 128, 256, and 512**. The given values can be adjusted based on the success of the network. The *epochs* parameter is used to specify how many times the dataset will be fed into the model. The epoch number was tested as **5, 10, 20, and 50** for each validation (since 10-fold cross-validation was used during model fitting, these values were represented as **50, 100, 200, and 500** in the tables and figures).

The given values can be adjusted based on the success of the network. Setting a very high value for this parameter can lead to drawbacks in certain cases, such as increasing the processing time or causing the accuracy rates to become unrealistic after a certain point, turning the learning process into memorization.

The **10-fold cross-validation** approach was used for performance evaluation. This approach uses a train set that is split into 10 parts at random. One part of each fold is utilized for validating, while the combination of the remaining parts is used for training. To validate each part, this procedure is performed ten times, and the total accuracy is determined by averaging all of the values. Through validation, the data is trained without overfitting, ensuring that the trained model learns effectively. Consequently, the trained model becomes ready for testing.

In the final stage, after going through the training and validation processes, the data can be predicted using the predict method with the trained model. In this phase, the trained model predicts on a test set that is unseen by the model, revealing the extent to which the data has been successfully trained. Subsequently, various evaluation metrics have been utilized to observe the results. The comparison values are presented in the result tables (Section 4.1.5).

4. RESULTS AND DISCUSSIONS

In this section, the classification results of feature values using machine learning and deep learning methods are presented and analyzed. Each classifier's result is accompanied by two subheadings: "**Grid search results**" and "**Test results**". Grid search results indicate the optimal values of the parameters used in the classifiers. The test results show the final outcomes after applying the optimal parameters to the EEG features in each case.

Results include four fundamental evaluation metrics: "**accuracy**", "**F1 score**", "**precision**" and "**recall**". These metrics are calculated using two average computation parameters: "**weighted**" and "**macro**" (**unweighted**), which are also utilized in the tables. The formulas and parameters of the evaluation metrics are briefly specified below.

Evaluation metrics:

- $Accuracy(acc) = \frac{TP}{k}$ (4.1)

- $Precision = \frac{TP}{TP+FP}$ (4.2)

- $Recall = \frac{TP}{TP+FN}$ (4.3)

- $F1\ score = \frac{2*Precision*Recall}{Precision+Recall}$ (4.4)

Metric parameters:

- **Macro (unweighted) average** = It is one of the most basic averaging methods. It is calculated by taking the arithmetic average of all values per class. In essence, regardless of the data distribution, it treats all classes equally.
- **Weighted average** = The average of all values per class is calculated by evaluating the support of each class. The term "support" refers to the ratio obtained by dividing the number of instances belonging to the specific class in the dataset by the total number of instances.

where true-positive (**TP**) is the number of correctly labeled all states (ictal, preictal, and interictal), false-positive (**FP**) denotes the number of non-ictal (or non-preictal and non-

interictal), states falsely classified as ictal (or preictal and non-interictal), false-negative (FN) is the number of ictal (or preictal and interictal) states which are classified as preictal or interictal (vice versa for preictal and interictal) and k represents the total number of samples on each subject EEG data. In the study, since a 3-class classification was performed, the conditions apply to both preictal and interictal parts.

4.1. Grid search Results

In this section, the best classifier parameters obtained through grid search are selected. Furthermore, the dataset used in this section is Table 3.6, which includes samples from all the data, as specified in Section 3.4.1. Each classifier has obtained separate results based on the WT and NE feature extraction methods.

4.1.1. Optimal kNN parameters for WT and NE features

In order to find the best parameters for kNN classifier, 25 values for k parameter and 3 distance metrics were used in grid search as discussed in Section 3.4.2. Tables 4.1 and 4.2, show the parameter values for top 10 accuracies obtained by grid search for WT and NE features respectively.

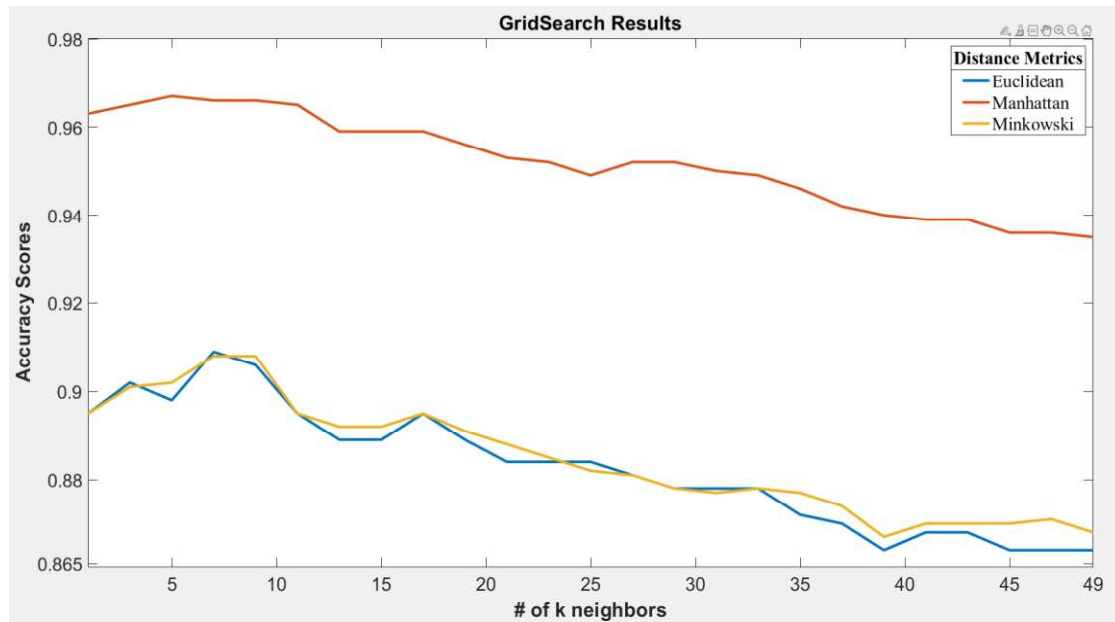


Figure 4.1. The accuracy values with respect to parameter k for different distance metrics for WT

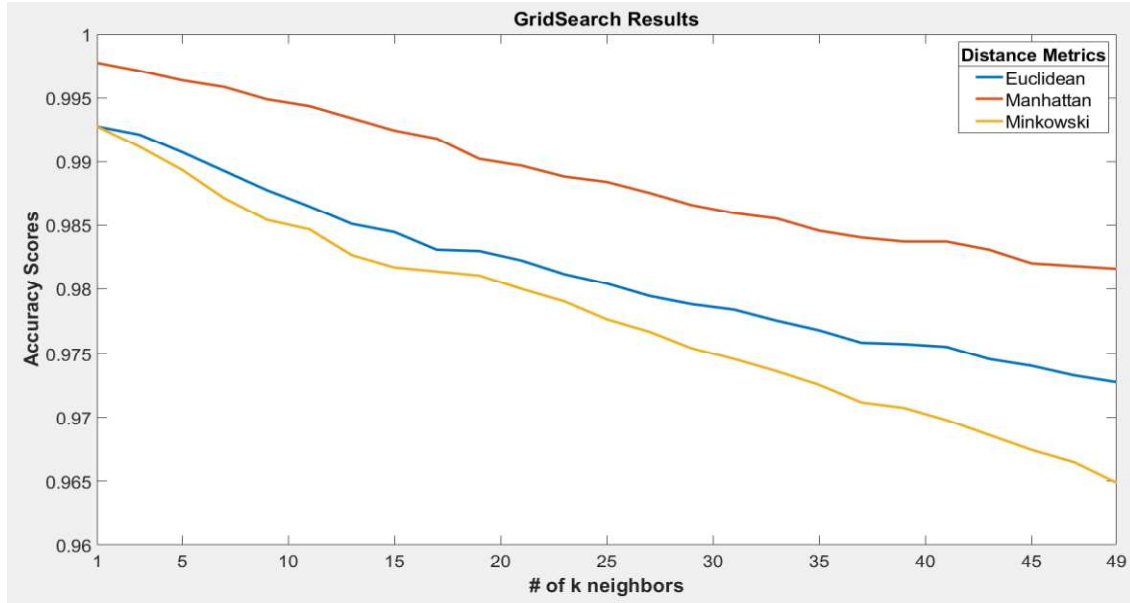


Figure 4.2. The accuracy values with respect to parameter k for different distance metrics for NE

As observed in Figures 4.1 and 4.2, an increase in the value of k resulted in a decrease in the success rate regardless of the distance metrics. The Manhattan distance metric demonstrated noticeably higher accuracy values compared to the others. In Figure 4.1, Minkowski and Euclidean metrics resulted in comparable accuracies, while in Figure 4.2, the Minkowski metric had a lower accuracy value compared to the Euclidean metric.

Table 4.1. Top 10 accuracies and corresponding parameters based on the average of 10-fold cross-validation kNN for WT.

#	Distance metrics	k neighbors	Accuracy	F1 score weighted	F1 score macro	Precision weighted	Precision macro	Recall weighted	Recall macro
1	<i>manhattan</i>	5	<i>0.965</i>	<i>0.963</i>	<i>0.885</i>	<i>0.966</i>	<i>0.963</i>	<i>0.965</i>	<i>0.844</i>
2	manhattan	1	0.964	0.963	0.901	0.964	0.947	0.964	0.870
3	manhattan	3	0.963	0.961	0.891	0.963	0.954	0.963	0.854
4	manhattan	7	0.963	0.960	0.872	0.964	0.969	0.963	0.826
5	manhattan	9	0.963	0.959	0.870	0.964	0.973	0.963	0.823
6	manhattan	11	0.959	0.955	0.850	0.960	0.966	0.959	0.803
7	manhattan	13	0.958	0.954	0.845	0.959	0.968	0.958	0.797
8	manhattan	15	0.956	0.951	0.836	0.957	0.963	0.956	0.789
9	manhattan	17	0.954	0.948	0.826	0.955	0.961	0.954	0.780
10	manhattan	19	0.952	0.946	0.821	0.954	0.963	0.952	0.775

Table 4.2. Top 10 accuracies and corresponding parameters based on the average of 10-fold cross-validation kNN for NE.

#	Distance metrics	k neighbors	Accuracy	F1 score weighted	F1 score macro	Precision weighted	Precision macro	Recall weighted	Recall macro
1	<i>manhattan</i>	1	0.998	0.998	0.996	0.998	0.995	0.998	0.998
2	manhattan	3	0.997	0.997	0.996	0.997	0.995	0.997	0.997
3	manhattan	5	0.996	0.996	0.994	0.996	0.994	0.996	0.994
4	manhattan	7	0.996	0.996	0.993	0.996	0.994	0.996	0.993
5	manhattan	9	0.995	0.995	0.992	0.995	0.993	0.995	0.992
6	manhattan	11	0.994	0.994	0.992	0.994	0.993	0.994	0.990
7	manhattan	13	0.993	0.993	0.990	0.993	0.992	0.993	0.987
8	minkowski	1	0.993	0.993	0.987	0.993	0.985	0.993	0.988
9	euclidean	1	0.993	0.993	0.987	0.993	0.985	0.993	0.988
10	manhattan	15	0.992	0.992	0.986	0.993	0.991	0.992	0.982

Table 4.1 and 4.2 show the top 10 accuracy values and corresponding optimal kNN parameters, which were selected based on the average of 10-fold cross-validation, for WT and NE respectively. Best value of **0.965** accuracy is obtained when "**5 neighbors ($k=5$)**" using the "**manhattan**" distance metric are used for WT features. On the other hand, the best value of **0.998** accuracy is achieved with the parameters "**manhattan**" distance metric and "**1 neighbor ($k=1$)**" for NE features. It is worth to note that, in Table 4.1, the Manhattan distance metric appears as the best for all 10 results. In Table 4.2, "**minkowski**" ranked 8th, while "**euclidean**" ranked 9th. The evaluation metrics in both tables yielded similar and successful results. Consequently, the parameters to be applied for all subjects are as follows:

- For *WT*: Distance metric: **Manhattan**, k neighbors: **5**
- For *NE*: Distance metric: **Manhattan**, k neighbors: **1**

4.1.2. Optimal SVM parameters for WT and NE features

As discussed in Section 3.4.3, we utilized the parameters such as " **C (0.1, 1, 10, 100 and 1000)**", " **γ (1, 0.1, 0.01, 0.001 and 0.0001)**" and kernel types including "**linear**", "**polynomial**" and "**rbf**" the SVM classifier. Therefore, 75 results were obtained. After applying these parameters to the SVM classifier, the top 10 results for WT and NE features were determined and presented in Tables 4.3 and 4.4, respectively.

Table 4.3. Top 10 accuracies and corresponding parameters based on the average of 10-fold cross-validation SVM for WT.

#	C	Gamma	Kernel	Accuracy	F1 score weighted	F1 score macro	Precision weighted	Precision macro	Recall weighted	Recall macro
1	1000	1	<i>rbf</i>	0.936	0.936	0.897	0.937	0.926	0.936	0.876
2	100	1	rbf	0.915	0.914	0.849	0.916	0.909	0.915	0.814
3	1000	0.1	rbf	0.902	0.900	0.830	0.904	0.894	0.902	0.795
4	1000	1	linear	0.891	0.889	0.812	0.890	0.856	0.891	0.785
5	1000	0.0001	linear	0.891	0.889	0.812	0.890	0.856	0.891	0.785
6	1000	0.001	linear	0.891	0.889	0.812	0.890	0.856	0.891	0.785
7	1000	0.01	linear	0.891	0.889	0.812	0.890	0.856	0.891	0.785
8	1000	0.1	linear	0.891	0.889	0.812	0.890	0.856	0.891	0.785
9	100	0.001	linear	0.876	0.870	0.735	0.875	0.827	0.876	0.704
10	100	0.01	linear	0.876	0.870	0.735	0.875	0.827	0.876	0.704

Table 4.4. Top 10 accuracies and corresponding parameters based on the average of 10-fold cross-validation SVM for NE.

#	C	Gamma	Kernel	Accuracy	F1 score weighted	F1 score macro	Precision weighted	Precision macro	Recall weighted	Recall macro
1	1000	1	<i>rbf</i>	0.835	0.826	0.679	0.841	0.834	0.835	0.648
2	1000	1	linear	0.807	0.795	0.611	0.817	0.775	0.807	0.596
3	1000	0.0001	linear	0.807	0.795	0.611	0.817	0.775	0.807	0.596
4	1000	0.001	linear	0.807	0.795	0.611	0.817	0.775	0.807	0.596
5	1000	0.01	linear	0.807	0.795	0.611	0.817	0.775	0.807	0.596
6	1000	0.1	linear	0.807	0.795	0.611	0.817	0.775	0.807	0.596
7	100	1	rbf	0.801	0.788	0.595	0.812	0.771	0.801	0.585
8	1000	0.1	rbf	0.789	0.774	0.572	0.801	0.736	0.789	0.569
9	100	0.0001	linear	0.777	0.760	0.547	0.785	0.653	0.777	0.552
10	100	0.01	linear	0.777	0.760	0.547	0.785	0.653	0.777	0.552

Table 4.3 shows the best parameter values with an accuracy value of **0.936** was obtained with $C = 1000$, $\gamma = 1$, and the *rbf* kernel. Table 4.4 shows the best parameters for NE features, parameters' values resulting with the best accuracy remained the same, but a lower success rate of 0.835 was observed. Additionally, in both tables, the *polynomial* parameter, represented as the kernel type, did not appear among the successful 10 results out of the 75 possible combinations. Therefore, the parameters to be used in all subjects are as follows:

- For both *WT* and *NE*: Kernel type: *rbf*, gamma: 1, C: 1000

4.1.3. Optimal RF parameters for WT and NE features

As discussed in Section 3.4.4, we utilized the parameters such as "*random state* (0, 50 and 100)", "*n estimator* (64, 128, 256 and 512)" and "*max depth* (4, 8, 16 and 32)" the RF classifier. After applying these parameters to the RF classifier, the best 10 results out of the 48 combinations tested for WT and NE features were presented in Tables 4.5 and 4.6, respectively.

Table 4.5. Top 10 accuracies and corresponding parameters based on the average of 10-fold cross-validation RF for WT.

#	Random State	n estimators	Maximum depth	Accuracy	F1 score weighted	F1 score macro	Precision weighted	Precision macro	Recall weighted	Recall macro
1	50	512	32	0.984	0.983	0.953	0.984	0.989	0.984	0.925
2	100	512	32	0.984	0.983	0.952	0.984	0.989	0.984	0.924
3	50	256	16	0.984	0.983	0.952	0.984	0.989	0.984	0.925
4	100	512	16	0.984	0.983	0.952	0.984	0.989	0.984	0.925
5	0	512	32	0.984	0.983	0.952	0.984	0.989	0.984	0.924
6	0	256	16	0.984	0.983	0.950	0.984	0.989	0.984	0.922
7	50	256	32	0.984	0.983	0.951	0.984	0.989	0.984	0.923
8	0	256	32	0.984	0.983	0.950	0.984	0.989	0.984	0.921
9	50	128	16	0.984	0.983	0.949	0.984	0.989	0.984	0.920
10	0	512	16	0.984	0.983	0.950	0.984	0.989	0.984	0.922

Table 4.6. Top 10 accuracies and corresponding parameters based on the average of 10-fold cross-validation RF for NE.

#	Random State	n estimators	Maximum depth	Accuracy	F1 score weighted	F1 score macro	Precision weighted	Precision macro	Recall weighted	Recall macro
1	0	512	32	0.995	0.995	0.989	0.995	0.995	0.995	0.984
2	100	512	16	0.995	0.995	0.990	0.995	0.995	0.995	0.985
3	100	256	16	0.995	0.995	0.989	0.995	0.996	0.995	0.983
4	100	128	16	0.995	0.995	0.990	0.995	0.996	0.995	0.985
5	50	256	32	0.994	0.994	0.988	0.994	0.995	0.994	0.981
6	100	128	32	0.994	0.994	0.989	0.995	0.995	0.994	0.983
7	100	512	32	0.994	0.994	0.988	0.994	0.995	0.994	0.982
8	50	256	16	0.994	0.994	0.987	0.994	0.995	0.994	0.980
9	0	512	16	0.994	0.994	0.988	0.994	0.995	0.994	0.982
10	0	256	32	0.994	0.994	0.988	0.994	0.995	0.994	0.982

Tables 4.5 and 4.6 shows that, it was observed that the top 10 parameters in both tables yielded very close results. Therefore, it can be inferred that parameter changes would not result in significant variations across the entire dataset. When examining the best parameter values in Table 4.5, an accuracy value of **0.984** was obtained with "*random state* = **50**," "*n estimators* = **512**," and "*max depth* = **32**." For Table 4.6, except for the random state, the parameters remained the same, and a higher success rate of **0.995** was observed. The higher values of n estimators in both tables indicate that the tree is more complex and requires more "branches" or "depth" for its computations. Therefore, the parameters that need to be applied to all subjects are as follows:

- For *WT*: random state: **50**, n estimators: **512**, maximum depth: **32**
- For *NE*: random state: **0**, n estimators: **512**, maximum depth: **32**

4.1.4. Optimal J48 parameters for WT and NE features

As discussed in Section 3.4.5, we utilized the J48 classifier employs the same parameters as the RF classifier used in the previous section. The only difference lies in the calculation methods. While RF utilizes *entropy* calculation to generate decision trees, J48 designs decision trees using the *gini* index. After applying these parameters to the J48 classifier, the best 10 results out of the 48 combinations tested for WT and NE features are presented in Tables 4.7 and 4.8, respectively.

Table 4.7. Top 10 accuracies and corresponding parameters based on the average of 10-fold cross-validation J48 for WT.

#	Random State	n estimator	Maximum depth	Accuracy	F1 score weighted	F1 score macro	Precision weighted	Precision macro	Recall weighted	Recall macro
1	100	512	32	0.983	0.982	0.946	0.983	0.988	0.983	0.915
2	100	128	32	0.983	0.982	0.945	0.983	0.988	0.983	0.914
3	100	256	32	0.982	0.981	0.944	0.983	0.988	0.982	0.913
4	0	512	32	0.982	0.981	0.943	0.982	0.988	0.982	0.911
5	50	512	32	0.982	0.981	0.941	0.982	0.987	0.982	0.908
6	100	512	16	0.982	0.981	0.939	0.982	0.987	0.982	0.905
7	100	64	32	0.981	0.980	0.936	0.982	0.987	0.981	0.901
8	0	256	32	0.981	0.980	0.939	0.981	0.987	0.981	0.906
9	50	128	16	0.981	0.980	0.934	0.981	0.987	0.981	0.899
10	100	256	16	0.981	0.980	0.939	0.981	0.987	0.981	0.905

Table 4.8. Top 10 accuracies and corresponding parameters based on the average of 10-fold cross-validation J48 for NE.

#	Random State	n estimator	Maximum depth	Accuracy	F1 score weighted	F1 score macro	Precision weighted	Precision macro	Recall weighted	Recall macro
1	0	256	32	0.995	0.995	0.989	0.995	0.995	0.995	0.984
2	0	128	32	0.994	0.994	0.988	0.994	0.995	0.994	0.982
3	50	512	32	0.994	0.994	0.988	0.994	0.995	0.994	0.980
4	0	512	32	0.994	0.994	0.988	0.994	0.994	0.994	0.981
5	50	256	32	0.994	0.994	0.986	0.994	0.994	0.994	0.979
6	50	128	32	0.994	0.994	0.987	0.994	0.995	0.994	0.979
7	100	512	32	0.994	0.994	0.986	0.994	0.994	0.994	0.979
8	100	128	32	0.994	0.994	0.987	0.994	0.994	0.994	0.980
9	100	64	32	0.994	0.994	0.985	0.994	0.994	0.994	0.977
10	0	64	32	0.994	0.994	0.986	0.994	0.994	0.994	0.978

In Table 4.7, the most successful result obtained from the optimal parameters is an accuracy value of 0.983 for "*random state = 100*", "*n estimators = 512*" and "*max depth = 32*". Additionally, the evaluation metrics closely follow each other in terms of values. The lowest accuracy value in the most successful result is "*recall macro = 0.915*". In Table 4.8, the best result is achieved for "*random state = 0*," "*n estimators = 256*," and "*max depth = 32*," with an accuracy value of **0.995**. This result signifies an increase compared to Table 4.7, albeit slight. The highest results in the NE category in the RF section's table (Table 4.6) align with the most successful result in Table 4.8. However, Table 4.8 achieves the same ratio with less complexity (fewer estimators). As a result, the parameter values to be used in all cases are as follows:

- For *WT*: random state: **100**, n estimators: **512**, maximum depth: **32**
- For *NE*: random state: **0**, n estimators: **256**, maximum depth: **32**

4.1.5. Optimal 1D-CNN parameters for WT and NE features

In Section 3.4.6.4, attempts were made to find optimal values through three different parameters tested within the model. The first parameter is the loss functions used to calculate the error during the model formation (CCE, MSE, and MAE), while the second and third parameters are the batch size and epoch number used for fitting the model. Batch size values were tested as **32, 64, 128, 256, and 512**, while the epoch number was tested as **5, 10, 20, and 50** for each validation (since 10-fold cross-validation was used during model fitting, these values were represented as **50, 100, 200, and 500** in the tables and figures). The grid search method was applied to find the optimal values. Initially, appropriate epoch and batch size values were

determined using the model created with the default CCE loss function. Then, the three loss functions were tested. Figures 4.3 and 4.4 illustrate the combination-based accuracy results obtained for WT and NE, based on the epoch and batch size values derived using the CCE loss function. Tables 4.9 and 4.10 display the top 10 accuracy results for the three 1D-CNN model parameters (batch size, epoch, loss function) for WT and NE, respectively.

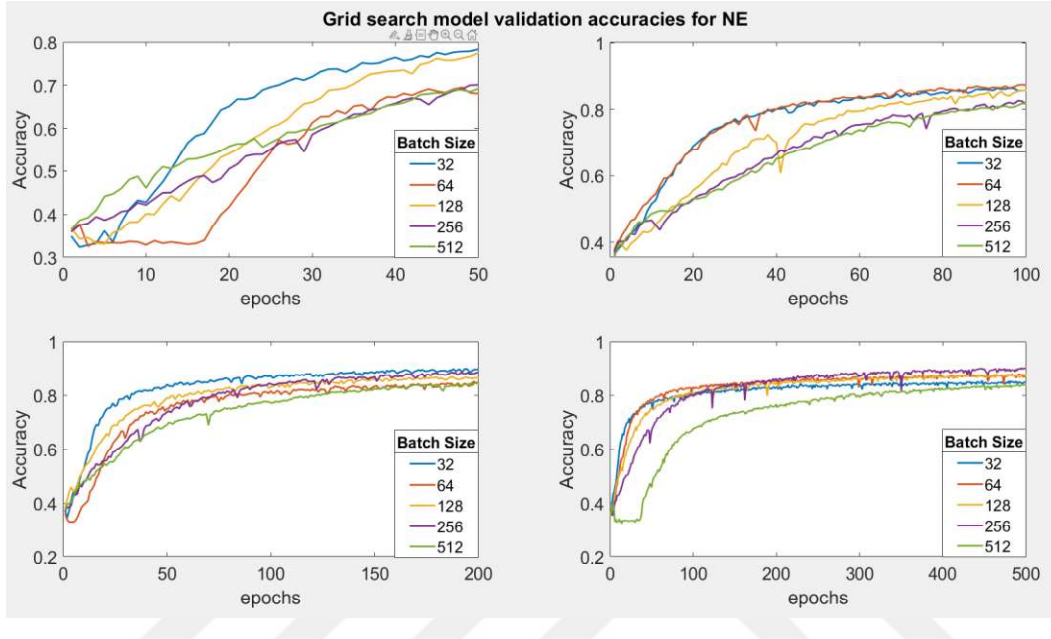


Figure 4.3. The accuracy scores of the model with respect to batch size and epochs for NE

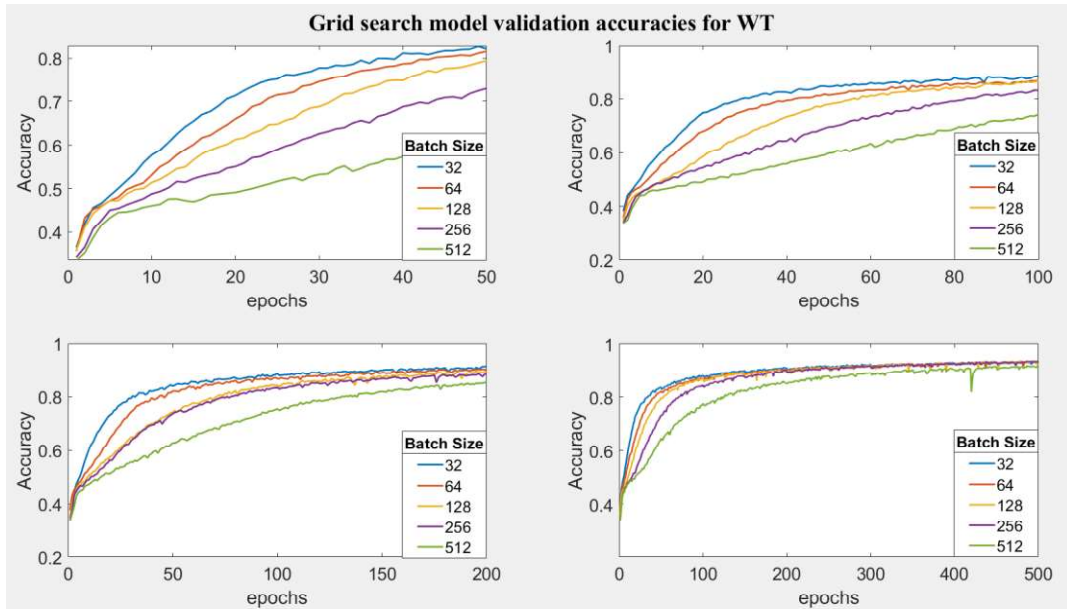


Figure 4.4. The accuracy scores of the model with respect to batch size and epochs for WT

The first notable observation from the above figures is that increasing the epoch value has had a positive impact on the results for WT and NE. As the epoch value increases, the accuracy values reach a saturation point after a certain threshold. Beyond this point, further increasing the epoch value does not significantly affect the results. This phenomenon was observed between epochs **200** and **500**. It indicates that the training data has reached a saturation point and the model starts to memorize the data. When the batch size is varied with a fixed epoch value, it was observed that decreasing the batch size positively contributes to the accuracy values for both figures. However, as the epoch value increases, the batch size values tend to converge toward each other.

Table 4.9. The top 10 parameters selected results based on 1D-CNN for WT

#	Loss function	Batch size	Epochs	Accuracy	F1 score weighted	F1 score macro	Precision weighted	Precision macro	Recall weighted	Recall macro
1	CCE	32	200	0.878	0.878	0.878	0.878	0.878	0.878	0.878
2	CCE	32	500	0.876	0.876	0.876	0.876	0.876	0.876	0.876
3	CCE	64	500	0.869	0.869	0.869	0.869	0.869	0.869	0.869
4	CCE	64	200	0.867	0.867	0.867	0.867	0.867	0.867	0.867
5	CCE	64	100	0.866	0.866	0.866	0.866	0.866	0.866	0.866
6	CCE	32	100	0.864	0.865	0.865	0.865	0.866	0.864	0.865
7	MSE	128	500	0.863	0.863	0.863	0.864	0.864	0.863	0.863
8	MSE	256	500	0.862	0.862	0.862	0.862	0.862	0.862	0.862
9	MSE	128	100	0.852	0.853	0.853	0.853	0.854	0.852	0.852
10	MSE	32	50	0.847	0.847	0.847	0.848	0.848	0.847	0.847
CCE: “categorical cross entropy”, MSE: “mean square error”										

Table 4.10. The top 10 parameters selected results based on 1D-CNN for NE.

#	Loss function	Batch size	Epochs	Accuracy	F1 score weighted	F1 score macro	Precision weighted	Precision macro	Recall weighted	Recall macro
1	CCE	256	200	0.891	0.890	0.891	0.891	0.891	0.891	0.891
2	CCE	32	200	0.887	0.887	0.887	0.887	0.887	0.887	0.887
3	CCE	256	500	0.886	0.886	0.886	0.887	0.887	0.886	0.887
4	CCE	128	100	0.883	0.882	0.882	0.883	0.883	0.883	0.883
5	MSE	64	500	0.879	0.879	0.879	0.880	0.880	0.879	0.880
6	CCE	64	100	0.877	0.877	0.877	0.878	0.878	0.877	0.878
7	MSE	32	100	0.877	0.877	0.877	0.877	0.877	0.877	0.877
8	MSE	128	200	0.872	0.872	0.872	0.873	0.873	0.872	0.873
9	MSE	128	500	0.871	0.871	0.871	0.871	0.871	0.871	0.871
10	MSE	64	200	0.861	0.861	0.862	0.864	0.865	0.861	0.861
CCE: “categorical cross entropy”, MSE: “mean square error”										

The results presented in Tables 4.9 and 4.10 show the top 10 values obtained through the applied parameters. When examining the loss function parameter, the *MAE* function could not be included in the top 10 for both tables. On the other hand, although *CCE* shows similar values to the MSE loss function, it has the highest values. When examining the batch size, the value of 32 stands out for WT. Moreover, as shown in Figures 4.3 and 4.4, there is almost no significant difference between the epochs of **200** and **500** with the same batch size (32). When the batch size parameter was examined in the NE table, the value of **256** emerged as the prominent choice, followed by **32** as the second most successful value. For the WT table, the best values were obtained when the CCE loss function was used, with a batch size of **32** and an epoch of **200**, resulting in a value of **0.878**. On the other hand, for the NE table, a value of **0.891** was achieved when the CCE loss function was used with a batch size of **256** and an epoch of **200**. When comparing the two tables, NE had a slight advantage with the best value. However, it should be noted that these values are limited to the data obtained from all subjects, and they may vary when applied to the entire subject data. As a result, the parameter values to be used in all cases are as follows:

- For *WT*: loss function: *CCE*, batch size: **32**, # of epochs: **200**
- For *NE*: loss function: *CCE*, batch size: **256**, # of epochs: **200**

Below, the output of accuracy and error values obtained from the best parameters for WT and NE in Figures 4.5 and 4.6, respectively, is provided. Additionally, the behavior of the loss functions in terms of accuracy is demonstrated for the best batch size and epoch values. Upon analyzing the below figures, it can be observed that the optimal batch size and epoch values were applied. Examining the model accuracy results, it can be seen that the data, which was sufficiently trained, reached saturation at the **200th. epoch**. The WT model showed a rapid recovery after the **20th epoch**, as observed from the results of the Categorical Cross-Entropy (CCE) loss function. On the other hand, the NE model reached stability after the **40th epoch**. Regarding the loss function, *CCE* had the highest value for both figures. Although *MSE* performed well in both cases, it did not surpass the CCE values. Finally, the *MAE* yielded inadequate results for WT, while it achieved a significantly higher accuracy rate for NE. The results obtained from the grid search for 1D-CNN appear promising. However, the final decision can be made after applying these parameters to the subject data.

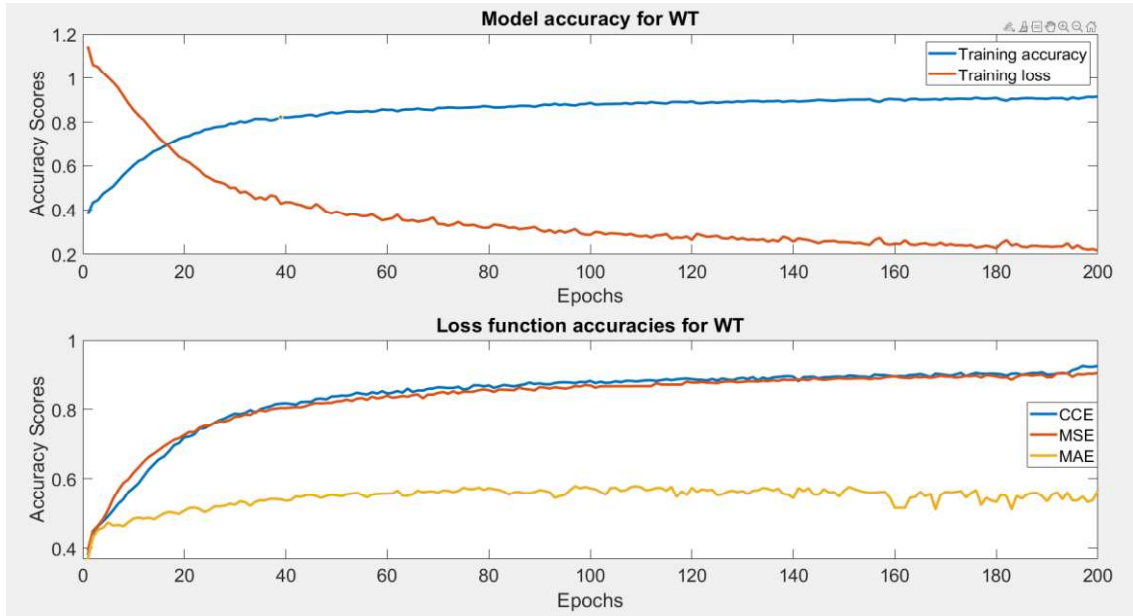


Figure 4.5. The model and loss function accuracy scores with best parameters (Batch size = 32 and Epochs = 200) for WT

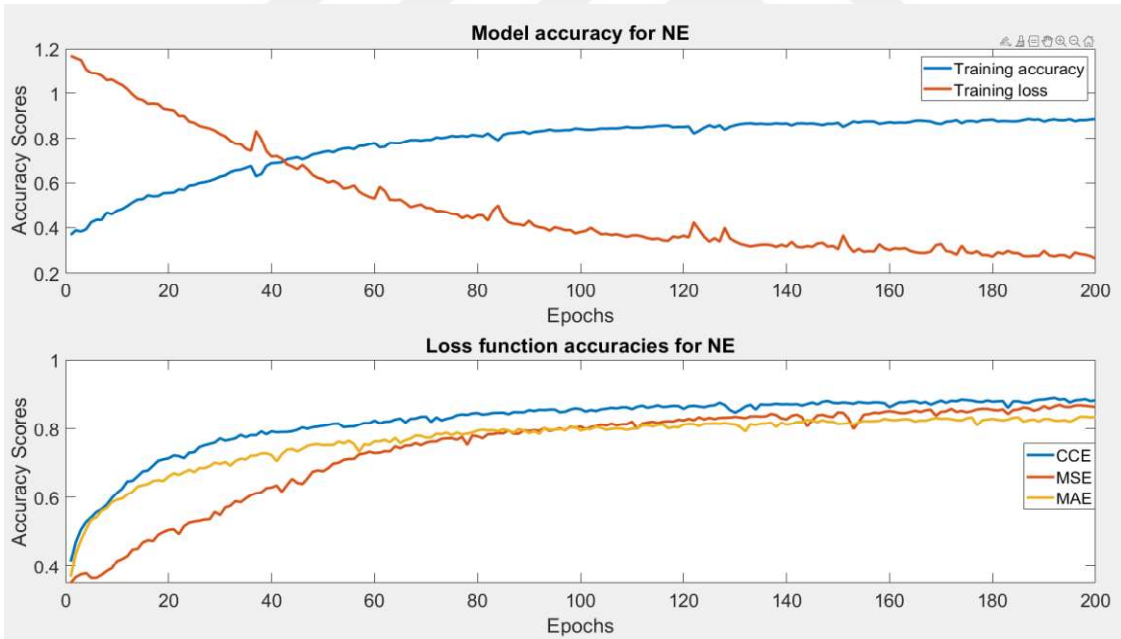


Figure 4.6. The model and loss function accuracy scores with best parameters (Batch size = 256 and Epochs = 200) for NE

4.2. Test Results

The best parameters obtained through the grid search technique on the balanced dataset were applied to the subject data from the imbalanced dataset.

4.2.1. Machine learning classifiers (kNN, SVM, RF and J48) results for WT and NE

The optimal machine learning classifier parameters obtained in the grid search phase was applied to both the WT and NE features.

Table 4.11. Weighted and unweighted accuracy results for WT

Unweighted (Macro) Accuracy					Weighted Accuracy				
Case	kNN	Random Forest	J48	SVM	Case	kNN	Random Forest	J48	SVM
01	91.35%	75.11%	86.33%	97.50%	01	99.78%	99.56%	99.23%	99.95%
02	86.94%	75.71%	83.44%	96.88%	02	99.74%	99.74%	99.28%	99.97%
03	85.94%	75.31%	84.02%	97.35%	03	99.49%	99.60%	98.99%	99.95%
04	90.41%	70.86%	75.62%	93.85%	04	99.77%	99.69%	99.54%	99.95%
05	80.61%	90.48%	89.96%	99.43%	05	96.54%	99.71%	99.33%	99.98%
06	92.08%	68.40%	75.47%	79.30%	06	99.89%	98.49%	98.68%	99.86%
07	93.02%	76.84%	84.34%	98.31%	07	99.74%	99.81%	99.53%	99.98%
08	91.51%	95.76%	94.72%	99.65%	08	99.03%	99.74%	98.75%	99.96%
09	92.36%	72.19%	79.18%	91.17%	09	99.97%	99.83%	99.63%	99.97%
10	94.33%	82.30%	84.12%	95.40%	10	99.46%	99.54%	98.46%	99.86%
11	93.31%	92.97%	92.75%	99.51%	11	99.50%	99.64%	99.05%	99.97%
12	94.57%	89.04%	90.21%	96.59%	12	99.37%	99.19%	97.69%	99.66%
13	90.35%	84.69%	88.70%	93.60%	13	97.78%	97.84%	98.04%	99.74%
14	89.43%	80.13%	83.49%	90.18%	14	99.91%	99.79%	99.35%	99.90%
15	94.85%	94.15%	90.82%	97.51%	15	98.38%	99.50%	97.88%	99.71%
16	89.59%	69.15%	75.44%	82.87%	16	99.82%	97.97%	98.16%	99.79%
17	99.73%	84.34%	86.98%	98.78%	17	99.91%	99.63%	99.16%	99.94%
18	89.29%	80.46%	83.85%	96.89%	18	99.60%	99.64%	99.22%	99.93%
19	95.76%	77.43%	81.89%	93.79%	19	99.90%	99.61%	99.33%	99.89%
20	86.69%	77.11%	89.10%	97.62%	20	99.69%	99.67%	99.51%	99.93%
21	89.50%	72.92%	80.07%	94.23%	21	99.76%	99.60%	99.12%	99.96%
22	94.58%	84.60%	89.37%	99.67%	22	99.77%	99.84%	99.35%	100.00%
23	89.29%	76.77%	81.44%	98.11%	23	99.32%	98.79%	97.62%	99.93%
24	95.00%	75.54%	79.16%	93.11%	24	99.79%	99.04%	96.71%	99.76%
Average	91.27%	80.09%	84.60%	95.05%	Average	99.41%	99.39%	98.82%	99.87%

In this part, frequency-based features were extracted using Wavelet Transform. *kNN*, *RF*, *SVM*, and *J48* classifiers were used to calculate the 3-class classification accuracy. The results obtained are shown in Table 4.11, which shows weighted and unweighted average accuracies. The unweighted average accuracy is known as the arithmetic mean, which treats all recall values equally without considering the sample numbers. The weighted average accuracy is produced by multiplying each component by a coefficient favoring the values for classes with a higher number of samples.

Table 4.11 shows that the weighted average results are close to each other for all patients. The highest average accuracy value belongs to *SVM* with **99.87%**. This accuracy value is followed by *kNN*, *RF*, and *J48* with values of **99.41%**, **99.39%**, and **98.82%** respectively. It is worth noting that individual accuracies are above **96%** for all results. Examining the unweighted average accuracies, the results have quite a wide range compared to weighted average recall. The highest average accuracy is obtained by *SVM* as **95.05%**. The average accuracy results for *kNN*, *J48*, and *RF* are **91.26%**, **84.61%**, and **80.09%** respectively.

When the table is analyzed on a per-subject basis, the lowest accuracy values are as follows: *RF* (**68.43%**), *J48* (**75.47%**), *SVM* (**79.30%**), and *kNN* (**80.06%**). The highest accuracy values, in decreasing order, are: *kNN* (**99.77%**), *SVM* (**99.67%**), *RF* (**95.77%**), and *J48* (**94.70%**). The remarkable difference in values and ranges for weighted and unweighted accuracies is due to the uneven distribution of samples in classes. Since the area occupied by the ictal data, as shown in Table 3.3, on the whole dataset is very small (**0.35%**), the margin of error factor of ictals in the weighted average recall table is ineffective compared to preictal and interictal. For this reason, weighted average accuracy values are much higher than the unweighted accuracy values. The unweighted average accuracy is used to justify this problem for unevenly distributed classes. In this way, the ictal classification accuracies are reflected in average results more accurately. Another point to be noted is that, although the age range is very high (from an 18-month-old baby to a young person at 22 years old) the accuracy results are very promising.

The recall, precision, and F1 scores of WT in the classification results are presented in Tables 4.12, 4.13, and 4.14, respectively.

Table 4.12. The recall results of the machine learning classifiers for WT

Case	Recall results											
	SVM			J48			Random Forest			kNN		
	Ictal	Preictal	Interictal	Ictal	Preictal	Interictal	Ictal	Preictal	Interictal	Ictal	Preictal	Interictal
01	0.927	0.999	1.000	0.617	0.976	0.997	0.267	0.986	1.000	0.748	0.992	1.000
02	0.909	0.998	1.000	0.549	0.958	0.997	0.291	0.980	1.000	0.623	0.986	0.999
03	0.922	0.999	1.000	0.557	0.968	0.995	0.271	0.988	1.000	0.599	0.980	0.999
04	0.832	0.983	1.000	0.372	0.899	0.998	0.241	0.885	1.000	0.788	0.925	0.999
05	0.984	0.999	1.000	0.725	0.977	0.997	0.728	0.986	1.000	0.599	0.833	0.986
06	0.383	0.996	1.000	0.316	0.955	0.993	0.123	0.935	0.995	0.767	0.996	1.000
07	0.951	0.998	1.000	0.582	0.950	0.998	0.326	0.979	1.000	0.817	0.975	0.999
08	0.990	0.999	1.000	0.872	0.976	0.993	0.876	0.998	0.999	0.764	0.985	0.996
09	0.736	0.999	1.000	0.404	0.973	0.998	0.179	0.987	1.000	0.771	1.000	1.000
10	0.868	0.995	1.000	0.584	0.948	0.992	0.491	0.978	1.000	0.848	0.985	0.997
11	0.988	0.998	1.000	0.853	0.934	0.996	0.822	0.967	1.000	0.845	0.954	0.999
12	0.902	0.998	0.998	0.747	0.976	0.984	0.679	0.994	0.998	0.845	0.996	0.996
13	0.814	0.995	0.999	0.717	0.955	0.989	0.605	0.946	0.990	0.771	0.953	0.986
14	0.706	1.000	1.000	0.520	0.989	0.996	0.407	0.998	1.000	0.684	0.999	1.000
15	0.929	0.997	0.999	0.764	0.974	0.987	0.831	0.995	0.999	0.876	0.982	0.987
16	0.489	0.997	0.999	0.309	0.966	0.989	0.128	0.957	0.990	0.691	0.997	0.999
17	0.966	0.997	1.000	0.639	0.975	0.996	0.544	0.986	1.000	0.997	0.996	1.000
18	0.910	0.997	1.000	0.551	0.968	0.996	0.433	0.980	1.000	0.693	0.987	0.998
19	0.820	0.994	1.000	0.490	0.970	0.997	0.347	0.976	1.000	0.879	0.994	1.000
20	0.930	0.998	1.000	0.685	0.990	0.998	0.318	0.996	1.000	0.606	0.996	0.999
21	0.828	0.999	1.000	0.443	0.963	0.996	0.207	0.981	0.999	0.690	0.997	0.998
22	0.990	1.000	1.000	0.715	0.969	0.997	0.546	0.992	1.000	0.845	0.993	0.999
23	0.944	0.999	1.000	0.508	0.949	0.986	0.327	0.983	0.993	0.689	0.995	0.994
24	0.795	1.000	0.999	0.433	0.974	0.968	0.275	0.999	0.992	0.852	0.999	0.999
Average	0.855	0.997	1.000	0.581	0.964	0.993	0.428	0.977	0.998	0.762	0.979	0.997

Table 4.13. The precision results of the machine learning classifiers for WT

Case	Precision results											
	SVM			J48			Random Forest			kNN		
	Ictal	Pretictal	Interictal	Ictal	Pretictal	Interictal	Ictal	Pretictal	Interictal	Ictal	Pretictal	Interictal
01	0.998	1.000	1.000	0.710	0.982	0.995	1.000	0.998	0.995	0.963	0.998	0.998
02	1.000	1.000	1.000	0.722	0.963	0.996	1.000	1.000	0.997	0.940	0.988	0.998
03	0.997	0.999	1.000	0.637	0.972	0.994	1.000	0.998	0.996	0.928	0.991	0.996
04	1.000	1.000	0.999	0.495	0.923	0.997	1.000	0.999	0.997	0.971	0.977	0.998
05	1.000	1.000	1.000	0.788	0.983	0.996	1.000	1.000	0.997	0.911	0.899	0.974
06	1.000	0.998	0.999	0.476	0.960	0.992	1.000	0.966	0.988	1.000	0.998	0.999
07	0.997	1.000	1.000	0.721	0.965	0.997	1.000	1.000	0.998	0.893	0.975	0.999
08	1.000	0.999	1.000	0.891	0.979	0.992	0.999	0.996	0.998	0.938	0.988	0.992
09	1.000	1.000	1.000	0.538	0.982	0.998	1.000	1.000	0.998	0.991	0.998	1.000
10	0.992	0.997	0.999	0.673	0.950	0.991	1.000	0.998	0.995	0.939	0.979	0.997
11	1.000	0.999	1.000	0.890	0.953	0.994	1.000	0.999	0.996	0.959	0.988	0.996
12	0.996	0.995	0.998	0.799	0.977	0.981	0.999	0.991	0.992	0.986	0.992	0.996
13	0.985	0.995	0.998	0.775	0.961	0.987	0.940	0.959	0.984	0.832	0.949	0.987
14	1.000	0.998	0.999	0.643	0.988	0.996	1.000	0.997	0.998	1.000	0.999	0.999
15	0.983	0.997	0.998	0.791	0.974	0.986	0.997	0.994	0.995	0.872	0.978	0.990
16	1.000	0.996	0.999	0.426	0.969	0.987	0.800	0.971	0.983	0.985	0.997	0.998
17	0.997	0.999	0.999	0.668	0.975	0.995	1.000	0.998	0.996	0.974	0.999	0.999
18	1.000	0.998	0.999	0.664	0.971	0.996	1.000	0.998	0.996	0.961	0.982	0.998
19	1.000	0.999	0.999	0.565	0.977	0.996	0.988	0.999	0.996	0.946	0.998	0.999
20	1.000	0.999	0.999	0.818	0.990	0.997	1.000	0.997	0.997	0.989	0.994	0.998
21	0.994	0.999	1.000	0.481	0.968	0.995	1.000	0.991	0.997	0.927	0.984	0.999
22	1.000	1.000	1.000	0.831	0.973	0.996	1.000	1.000	0.998	0.972	0.986	0.999
23	1.000	0.998	1.000	0.587	0.947	0.986	1.000	0.968	0.993	0.983	0.974	0.998
24	0.984	0.997	0.998	0.542	0.971	0.968	1.000	0.986	0.996	0.987	0.998	0.998
<i>Average</i>	<i>0.997</i>	<i>0.998</i>	<i>0.999</i>	<i>0.672</i>	<i>0.969</i>	<i>0.992</i>	<i>0.988</i>	<i>0.992</i>	<i>0.995</i>	<i>0.952</i>	<i>0.984</i>	<i>0.996</i>

Table 4.14. The F1 score results of the machine learning classifiers for WT

Case	F1 Score											
	SVM			J48			Random Forest			kNN		
	Ictal	Pretictal	Interictal	Ictal	Pretictal	Interictal	Ictal	Pretictal	Interictal	Ictal	Pretictal	Interictal
01	0.961	0.999	1.000	0.660	0.979	0.996	0.422	0.992	0.998	0.842	0.995	0.999
02	0.952	0.999	1.000	0.623	0.961	0.996	0.451	0.990	0.999	0.749	0.987	0.999
03	0.958	0.999	1.000	0.595	0.970	0.995	0.427	0.993	0.998	0.728	0.986	0.997
04	0.909	0.991	1.000	0.425	0.911	0.998	0.388	0.938	0.998	0.870	0.950	0.999
05	0.992	0.999	1.000	0.755	0.980	0.996	0.843	0.993	0.998	0.722	0.865	0.980
06	0.554	0.997	0.999	0.380	0.957	0.992	0.218	0.950	0.991	0.868	0.997	0.999
07	0.973	0.999	1.000	0.644	0.957	0.998	0.492	0.989	0.999	0.854	0.975	0.999
08	0.995	0.999	1.000	0.881	0.977	0.993	0.933	0.997	0.999	0.842	0.986	0.994
09	0.848	1.000	1.000	0.461	0.978	0.998	0.303	0.994	0.999	0.867	0.999	1.000
10	0.926	0.996	0.999	0.625	0.949	0.992	0.659	0.988	0.997	0.891	0.982	0.997
11	0.994	0.998	1.000	0.871	0.943	0.995	0.902	0.983	0.998	0.899	0.971	0.998
12	0.946	0.997	0.998	0.772	0.976	0.983	0.808	0.993	0.995	0.910	0.994	0.996
13	0.891	0.995	0.999	0.745	0.958	0.988	0.736	0.952	0.987	0.801	0.951	0.986
14	0.828	0.999	0.999	0.575	0.988	0.996	0.578	0.997	0.999	0.812	0.999	1.000
15	0.955	0.997	0.998	0.777	0.974	0.986	0.906	0.995	0.997	0.874	0.980	0.989
16	0.657	0.997	0.999	0.358	0.967	0.988	0.220	0.964	0.986	0.813	0.997	0.999
17	0.981	0.998	1.000	0.653	0.975	0.996	0.705	0.992	0.998	0.985	0.997	1.000
18	0.953	0.997	1.000	0.602	0.970	0.996	0.605	0.989	0.998	0.806	0.984	0.998
19	0.901	0.996	0.999	0.525	0.973	0.997	0.514	0.987	0.998	0.911	0.996	0.999
20	0.964	0.999	1.000	0.746	0.990	0.997	0.482	0.996	0.998	0.752	0.995	0.998
21	0.903	0.999	1.000	0.462	0.965	0.995	0.343	0.986	0.998	0.791	0.990	0.999
22	0.995	1.000	1.000	0.769	0.971	0.996	0.706	0.996	0.999	0.904	0.990	0.999
23	0.971	0.999	1.000	0.545	0.948	0.986	0.493	0.975	0.993	0.810	0.985	0.996
24	0.879	0.998	0.998	0.481	0.972	0.968	0.432	0.993	0.994	0.914	0.999	0.998
Average	0.912	0.998	0.999	0.622	0.966	0.992	0.565	0.984	0.996	0.842	0.981	0.997

When examining the recall results in Table 4.12, the classifier that best distinguishes ictal true positive (TP) data from false positive (FP) data is SVM with a value of 0.885. It is followed by 0.762 (kNN), 0.581 (J48), and 0.428 (RF). In this case, RF and J48 algorithms perform poorly for the ictal class. On the other hand, all classifiers achieve successful results with an average of at least 0.964 for preictal and 0.993 for interictal.

Examining the precision results in Table 4.13, the classifier that best distinguishes ictal TP data from false negative (FN) data is SVM with a value of 0.997. It is followed by 0.988 (kNN), 0.952 (RF) and 0.672 (J48). The J48 algorithm falls behind the other three classifiers for the ictal class.

When examining the preictal and interictal results, the best precision value is achieved by SVM with 0.998 for preictal and 0.999 for interictal. The precision values for interictal and preictal are also close to the SVM results for the other classifiers (for preictal: kNN: 0.984, RF: 0.992, and J48: 0.969 / for interictal: kNN: 0.996, RF: 0.995, and J48: 0.992).

Analyzing the F1 score results in Table 4.14, balancing precision and recall are controlled. The classifier that best separates the ictal data is SVM with a value of 0.912. It is followed by 0.842 (kNN), 0.622 (J48) and 0.565 (RF). J48 and RF could assume fail to perform well for the ictal class. On the other hand, all classifiers achieve successful results with at least 0.962 (J48) for interictal and 0.981 (kNN) for preictal on average.

When examining the three tables, SVM achieves the best average results for the ictal class. Similarly, when calculating preictal and interictal outputs, SVM outperforms the other three classifiers. This indicates that the accuracy of the SVM classifier is not random.

After the WT classification results, the features obtained from NE, which were determined to be successful in Section 4.1, are tabulated by applying the best machine learning parameters. Table 4.15 provides the weighted and unweighted (macro) results for all subjects.

Table 4.15. Weighted and unweighted accuracy results for NE

Unweighted (Macro) Accuracy					Weighted Accuracy				
Case	kNN	Random Forest	J48	SVM	Case	kNN	Random Forest	J48	SVM
01	98.85%	94.73%	88.50%	96.16%	01	99.93%	99.88%	99.49%	99.90%
02	98.37%	90.48%	88.49%	96.09%	02	99.88%	99.91%	99.61%	99.96%
03	95.94%	88.77%	85.71%	99.40%	03	99.88%	99.83%	99.41%	99.95%
04	96.04%	80.44%	80.35%	96.23%	04	99.92%	99.85%	99.73%	99.90%
05	98.55%	95.32%	91.46%	99.17%	05	99.91%	99.90%	99.58%	99.94%
06	94.27%	80.29%	79.96%	95.45%	06	99.67%	99.65%	98.74%	99.83%
07	99.36%	91.57%	90.34%	99.80%	07	99.95%	99.94%	99.74%	100.00%
08	99.28%	98.71%	96.21%	99.49%	08	99.77%	99.89%	99.26%	99.94%
09	97.61%	77.89%	79.44%	98.74%	09	99.95%	99.89%	99.74%	99.93%
10	97.56%	91.87%	87.90%	98.54%	10	99.87%	99.84%	99.26%	99.96%
11	98.25%	96.22%	95.54%	99.24%	11	99.87%	99.80%	99.42%	99.98%
12	98.22%	95.50%	92.12%	98.77%	12	99.73%	99.63%	98.55%	99.88%
13	98.39%	94.54%	92.09%	99.16%	13	99.70%	99.73%	98.69%	99.83%
14	94.88%	88.30%	88.09%	99.05%	14	99.93%	99.89%	99.66%	99.98%
15	97.26%	95.72%	90.06%	98.11%	15	99.42%	99.61%	98.10%	99.77%
16	90.35%	72.48%	78.85%	93.14%	16	99.47%	99.67%	98.83%	99.86%
17	97.31%	96.26%	92.97%	98.21%	17	99.77%	99.89%	99.56%	99.93%
18	95.80%	93.99%	89.18%	97.55%	18	99.84%	99.87%	99.52%	99.79%
19	95.19%	85.17%	85.59%	98.69%	19	99.87%	99.80%	99.60%	99.93%
20	98.71%	93.17%	91.41%	99.19%	20	99.93%	99.87%	99.64%	99.96%
21	96.49%	84.80%	82.37%	97.99%	21	99.94%	99.89%	99.42%	99.98%
22	98.04%	97.48%	95.36%	98.81%	22	99.89%	99.95%	99.68%	99.97%
23	94.27%	91.56%	87.96%	63.44%	23	99.67%	99.79%	98.98%	99.66%
24	96.19%	85.31%	84.35%	97.92%	24	99.61%	99.46%	98.12%	99.80%
Average	96.48%	90.02%	88.10%	96.60%	Average	99.81%	99.81%	99.26%	99.90%

In this part, time-based features were extracted using Non-linear energy. kNN, RF, SVM, and J48 classifiers were used to calculate the 3-class classification accuracy. The highest average accuracy value belongs to *SVM* with **99.90%**. This accuracy value is followed by *kNN*, *RF*, and *J48* with values of **99.81%**, **99.81%**, and **98.26%** respectively. Examining the unweighted average accuracies, the results have small range compared to weighted average recall. The highest average accuracy is obtained by *kNN* as **96.48%**. The average accuracy results for *SVM*, *RF*, and *J48* are **96.60%**, **90.02%**, and **88.10%** respectively. When the table is analyzed on a

per-subject basis, the lowest accuracy values are as follows: *SVM* (63.44%), *J48* (78.85%), *RF* (72.48%), and *kNN* (94.27%). The highest accuracy values, in decreasing order, are: *kNN* (99.80%), *SVM* (99.80%), *RF* (98.71%), and *J48* (96.21%). The recall, precision, and F1 scores of NE in the classification results are presented in Tables 4.16, 4.17, and 4.18, respectively.

Table 4.16. The recall results of the machine learning classifiers for NE

Case	Recall results											
	SVM			J48			Random Forest			kNN		
	Ictal	Precision	Interictal	Ictal	Precision	Interictal	Ictal	Precision	Interictal	Ictal	Precision	Interictal
01	0.889	0.996	1.000	0.673	0.984	0.998	0.846	0.995	1.000	0.969	0.997	1.000
02	0.886	0.997	1.000	0.680	0.977	0.998	0.720	0.994	1.000	0.960	0.992	1.000
03	0.985	0.997	1.000	0.592	0.982	0.998	0.667	0.996	1.000	0.883	0.996	1.000
04	0.927	0.960	1.000	0.474	0.938	0.999	0.463	0.950	1.000	0.898	0.984	1.000
05	0.979	0.997	1.000	0.758	0.987	0.998	0.863	0.996	1.000	0.961	0.996	1.000
06	0.874	0.990	1.000	0.451	0.955	0.994	0.427	0.982	1.000	0.842	0.988	0.998
07	0.994	1.000	1.000	0.741	0.970	0.999	0.753	0.994	1.000	0.988	0.993	1.000
08	0.986	0.999	1.000	0.903	0.988	0.996	0.963	0.998	1.000	0.984	0.996	0.998
09	0.968	0.995	1.000	0.400	0.984	0.999	0.343	0.994	1.000	0.932	0.997	1.000
10	0.958	0.998	1.000	0.665	0.976	0.996	0.762	0.994	1.000	0.932	0.996	0.999
11	0.978	0.999	1.000	0.909	0.960	0.997	0.905	0.982	1.000	0.956	0.993	0.999
12	0.965	0.999	1.000	0.787	0.986	0.990	0.869	0.997	0.999	0.951	0.997	0.999
13	0.980	0.996	0.999	0.799	0.971	0.992	0.843	0.995	0.999	0.960	0.994	0.998
14	0.972	1.000	1.000	0.650	0.995	0.998	0.650	1.000	1.000	0.847	0.999	1.000
15	0.947	0.997	0.999	0.735	0.979	0.988	0.876	0.997	0.999	0.928	0.994	0.996
16	0.798	0.997	1.000	0.394	0.979	0.993	0.181	0.994	1.000	0.723	0.990	0.997
17	0.949	0.997	1.000	0.804	0.987	0.998	0.892	0.996	1.000	0.926	0.995	0.998
18	0.932	0.996	0.998	0.697	0.981	0.998	0.827	0.993	1.000	0.882	0.992	0.999
19	0.967	0.994	1.000	0.586	0.984	0.998	0.565	0.990	1.000	0.862	0.994	0.999
20	0.977	0.999	1.000	0.752	0.992	0.998	0.798	0.997	1.000	0.964	0.998	1.000
21	0.941	0.999	1.000	0.498	0.976	0.997	0.547	0.997	1.000	0.897	0.998	1.000
22	0.966	0.998	1.000	0.879	0.983	0.999	0.928	0.997	1.000	0.947	0.995	0.999
23	0.891	0.013	1.000	0.666	0.978	0.995	0.749	0.998	0.999	0.835	0.995	0.998
24	0.941	1.000	0.997	0.562	0.985	0.984	0.564	0.999	0.996	0.892	0.997	0.997
Average	0.944	0.955	1.000	0.669	0.978	0.996	0.708	0.993	1.000	0.913	0.994	0.999

Table 4.17. The precision results of the machine learning classifiers for NE

Case	Precision results											
	SVM			J48			Random Forest			KNN		
	Ictal	Precision	Interictal	Ictal	Precision	Interictal	Ictal	Precision	Interictal	Ictal	Precision	Interictal
01	0.998	1.000	0.999	0.780	0.988	0.997	1.000	0.999	0.999	0.984	0.999	0.999
02	1.000	0.999	1.000	0.799	0.982	0.998	1.000	1.000	0.999	0.971	0.995	0.999
03	1.000	1.000	0.999	0.752	0.985	0.996	1.000	0.998	0.998	0.963	0.998	0.999
04	0.861	0.999	0.999	0.683	0.962	0.998	1.000	1.000	0.998	0.917	0.987	1.000
05	0.998	0.999	0.999	0.821	0.990	0.997	0.998	1.000	0.999	0.956	0.999	0.999
06	0.991	0.998	0.998	0.717	0.963	0.992	1.000	0.998	0.996	0.955	0.990	0.998
07	1.000	0.999	1.000	0.844	0.982	0.998	1.000	1.000	0.999	0.985	0.995	1.000
08	0.997	0.999	1.000	0.921	0.986	0.996	1.000	0.998	0.999	0.982	0.995	0.999
09	0.996	0.994	1.000	0.583	0.989	0.998	1.000	1.000	0.999	0.978	0.995	1.000
10	0.998	1.000	1.000	0.740	0.978	0.996	1.000	0.998	0.998	0.968	0.996	0.999
11	0.995	0.999	1.000	0.935	0.970	0.996	1.000	0.999	0.998	0.991	0.993	0.999
12	0.999	0.999	0.999	0.842	0.987	0.988	0.999	0.997	0.995	0.978	0.998	0.997
13	1.000	0.997	0.999	0.817	0.972	0.992	1.000	0.994	0.998	0.965	0.993	0.998
14	0.972	1.000	1.000	0.723	0.994	0.998	1.000	0.998	0.999	0.955	0.999	0.999
15	0.992	0.998	0.998	0.771	0.979	0.987	0.990	0.996	0.996	0.946	0.993	0.996
16	0.987	0.999	0.998	0.487	0.982	0.992	1.000	0.998	0.996	0.872	0.992	0.996
17	1.000	0.999	0.999	0.826	0.988	0.997	1.000	0.999	0.999	0.929	0.990	0.999
18	0.990	0.986	0.999	0.798	0.983	0.997	1.000	0.999	0.999	0.969	0.995	0.999
19	0.991	1.000	0.999	0.693	0.987	0.998	1.000	0.998	0.998	0.904	0.996	0.999
20	1.000	1.000	1.000	0.880	0.994	0.998	1.000	0.999	0.999	0.973	0.999	0.999
21	1.000	1.000	1.000	0.673	0.979	0.996	1.000	1.000	0.999	0.963	0.997	1.000
22	0.995	0.999	1.000	0.933	0.988	0.998	1.000	0.999	1.000	0.990	0.995	0.999
23	0.995	0.998	0.996	0.776	0.979	0.994	1.000	0.997	0.998	0.925	0.992	0.998
24	0.998	0.997	0.999	0.610	0.985	0.983	1.000	0.993	0.997	0.963	0.997	0.996
<i>Average</i>	<i>0.990</i>	<i>0.998</i>	<i>0.999</i>	<i>0.767</i>	<i>0.982</i>	<i>0.995</i>	<i>0.999</i>	<i>0.998</i>	<i>0.998</i>	<i>0.958</i>	<i>0.995</i>	<i>0.999</i>

Table 4.18. The F1 score results of the machine learning classifiers for NE

F1 score results															
Case	SVM			J48			Random Forest			KNN					
	Ictal	Preictal	Interictal	Ictal	Preictal	Interictal	Ictal	Preictal	Interictal	Ictal	Preictal	Interictal			
01	0.940	0.998	0.999	0.722	0.986	0.997	0.917	0.997	0.999	0.976	0.998	1.000			
02	0.939	0.998	1.000	0.735	0.979	0.998	0.837	0.997	1.000	0.966	0.993	0.999			
03	0.993	0.998	1.000	0.662	0.984	0.997	0.801	0.997	0.999	0.921	0.997	0.999			
04	0.893	0.979	0.999	0.560	0.949	0.999	0.633	0.974	0.999	0.907	0.985	1.000			
05	0.988	0.998	1.000	0.789	0.989	0.998	0.926	0.998	0.999	0.958	0.997	1.000			
06	0.929	0.994	0.999	0.553	0.959	0.993	0.598	0.990	0.998	0.895	0.989	0.998			
07	0.997	1.000	1.000	0.789	0.976	0.999	0.859	0.997	1.000	0.986	0.994	1.000			
08	0.991	0.999	1.000	0.911	0.987	0.996	0.981	0.998	0.999	0.983	0.996	0.999			
09	0.982	0.994	1.000	0.475	0.986	0.999	0.511	0.997	0.999	0.954	0.996	1.000			
10	0.978	0.999	1.000	0.701	0.977	0.996	0.865	0.996	0.999	0.949	0.996	0.999			
11	0.986	0.999	1.000	0.922	0.965	0.997	0.950	0.991	0.999	0.973	0.993	0.999			
12	0.981	0.999	0.999	0.814	0.987	0.989	0.929	0.997	0.997	0.964	0.998	0.998			
13	0.990	0.996	0.999	0.808	0.972	0.992	0.915	0.994	0.999	0.962	0.994	0.998			
14	0.972	1.000	1.000	0.685	0.994	0.998	0.788	0.999	0.999	0.898	0.999	1.000			
15	0.969	0.998	0.998	0.752	0.979	0.988	0.930	0.996	0.998	0.937	0.993	0.996			
16	0.882	0.998	0.999	0.435	0.980	0.992	0.306	0.996	0.998	0.791	0.991	0.997			
17	0.974	0.998	1.000	0.815	0.988	0.998	0.943	0.997	0.999	0.927	0.993	0.999			
18	0.960	0.991	0.999	0.744	0.982	0.998	0.905	0.996	0.999	0.924	0.994	0.999			
19	0.979	0.997	1.000	0.635	0.985	0.998	0.722	0.994	0.999	0.882	0.995	0.999			
20	0.988	0.999	1.000	0.811	0.993	0.998	0.888	0.998	0.999	0.968	0.999	1.000			
21	0.970	0.999	1.000	0.572	0.978	0.997	0.707	0.998	0.999	0.929	0.998	1.000			
22	0.980	0.999	1.000	0.905	0.985	0.998	0.962	0.998	1.000	0.968	0.995	0.999			
23	0.940	0.025	0.998	0.717	0.979	0.994	0.857	0.997	0.999	0.878	0.994	0.998			
24	0.969	0.999	0.998	0.585	0.985	0.983	0.721	0.996	0.996	0.926	0.997	0.996			
Average	0.965	0.956	0.999	0.712	0.980	0.995	0.810	0.995	0.999	0.934	0.995	0.999			

When examining the recall results in Table 4.16, the best classifier for identifying the ictal class is *SVM* with a value of **0.944**. It is followed by **0.913** (*kNN*), **0.708** (*RF*), and **0.669** (*J48*). Comparing the recall values of WT and NE, the most successful classifiers are the same, but NE achieves a higher result (*SVM* had a recall of 0.855 in WT). When looking at the preictal values, the highest recall is **0.994**, achieved by *kNN*. Similarly, for interictal, a value of **0.999** is obtained using *kNN*.

In Table 4.17, examining the precision results reveals that *RF* achieves the highest precision value of **0.999** for the ictal class, which is different from the previous results. The WT precision value for the ictal class is **0.997** with *SVM*. For preictal, both *SVM* and *RF* have a precision value of **0.998**. In interictal, this value is **0.999** for both *kNN* and *SVM*.

When analyzing the F1 score results in Table 4.18, *SVM* has the highest score of **0.965** for the ictal class. In the WT table, *SVM* also has the highest score of **0.912**. Overall, although there is a small difference, NE achieves a more successful result for the ictal class. Comparing preictal values, *kNN* and *RF* yield a score of **0.995**, which is lower than the WT result of 0.998. In the last part, when considering interictal, *SVM* achieves a value of **0.999**, which matches the WT result. When examining the three tables, *SVM* has found the best average result for the ictal class.

Comparing the WT and NE results in Tables 4.11 and 4.15, it can be observed that the classifiers generally yield similar values, but *SVM* stands out with a noticeable difference among machine learning algorithms. Additionally, *SVM* demonstrates a dominant success in detecting the ictal class.

4.2.2. Deep learning classifier (1D-CNN) results for WT and NE

In this section, the best 1D-CNN parameters found in Section 4.1.5 were applied to all subjects with WT and NE features, and the results were analyzed. The WT results are shown in Tables 4.19 and 4.20, while the NE results are presented in Tables 4.21 and 4.22.

Table 4.19. Weighted and unweighted accuracy, precision, recall and F1score 1D-CNN results for WT

Cases	Accuracy	Balanced accuracy	Precision weighted	Precision macro	Recall weighted	Recall macro	F1 score weighted	F1 score macro
01	0.993	0.985	0.993	0.981	0.993	0.986	0.992	0.986
02	0.995	0.993	0.993	0.989	0.994	0.946	0.993	0.968
03	0.992	0.995	1.000	0.995	0.990	0.930	0.989	0.960
04	0.995	0.997	0.989	0.987	0.990	0.907	0.989	0.948
05	0.997	0.995	0.997	0.995	0.997	0.982	0.997	0.989
06	0.978	0.991	0.998	0.992	0.978	0.928	0.977	0.957
07	0.997	0.994	0.997	0.994	0.997	0.931	0.997	0.960
08	0.996	0.988	0.996	0.988	0.996	0.979	0.996	0.983
09	0.998	0.984	0.998	0.965	0.991	0.964	0.998	0.974
10	0.995	0.987	0.995	0.983	0.995	0.983	0.995	0.985
11	0.996	0.997	0.996	0.997	0.996	0.943	0.996	0.969
12	0.993	0.986	0.993	0.986	0.993	0.948	0.993	0.966
13	0.992	0.983	0.992	0.983	0.992	0.923	0.992	0.950
14	0.998	0.998	0.998	0.998	0.998	0.969	0.998	0.983
15	0.991	0.979	0.991	0.979	0.991	0.943	0.991	0.960
16	0.995	0.976	0.994	0.912	0.995	0.941	0.994	0.957
17	0.998	0.992	0.998	0.992	0.998	0.951	0.998	0.970
18	0.997	0.989	0.997	0.989	0.997	0.923	0.997	0.953
19	0.997	0.998	0.997	0.998	0.997	0.961	0.997	0.979
20	0.997	0.987	0.997	0.987	0.997	0.929	0.997	0.955
21	0.997	0.992	0.997	0.989	0.997	0.968	0.997	0.980
22	0.998	0.998	0.998	0.998	0.998	0.958	0.998	0.977
23	0.994	0.985	0.994	0.982	0.994	0.985	0.994	0.985
24	0.991	0.988	0.991	0.982	0.991	0.988	0.991	0.988
<i>Average</i>	<i>0.995</i>	<i>0.953</i>	<i>0.995</i>	<i>0.985</i>	<i>0.994</i>	<i>0.953</i>	<i>0.994</i>	<i>0.970</i>

This section presents the results obtained from the 1D-CNN classification of WT features. When applied to all subjects, the average accuracy value is **0.995**. The average balanced (unweighted) accuracy value is **0.953**. The convergence of weighted and unweighted accuracy values indicates the successful classification of the ictal outputs. Table 4.20 provides a more detailed explanation of the three-class classification. Particularly noteworthy is that the F1 score, precision, and recall results are all above **0.970**.

Table 4.20. Precision, recall and F1 score results from 3-class classification 1D-CNN (Ictal, preictal and interictal) for WT

Cases	Recall			Precision			F1 score		
	Interictal	Preictal	Ictal	Interictal	Preictal	Ictal	Interictal	Preictal	Ictal
01	0.997	0.994	0.967	0.992	0.997	0.967	0.994	0.996	0.967
02	0.999	0.960	0.879	0.996	0.982	1.000	0.997	0.971	0.935
03	0.999	0.955	0.835	0.992	0.992	1.000	0.996	0.973	0.910
04	1.000	0.802	0.918	0.995	0.997	1.000	0.998	0.889	0.957
05	1.000	0.982	0.966	0.997	0.998	0.991	0.998	0.990	0.978
06	1.000	0.858	0.927	0.976	0.997	1.000	0.988	0.922	0.962
07	1.000	0.964	0.829	1.000	0.982	1.000	1.000	0.973	0.906
08	0.999	0.992	0.945	0.998	0.994	0.972	0.998	0.993	0.958
09	1.000	0.982	0.911	0.999	0.998	0.953	0.999	0.990	0.932
10	0.999	0.979	0.970	0.996	0.994	0.970	0.998	0.986	0.970
11	1.000	0.960	0.871	0.996	0.994	1.000	0.998	0.977	0.931
12	0.995	0.994	0.854	0.995	0.990	0.973	0.995	0.992	0.910
13	0.994	0.988	0.787	0.997	0.976	0.977	0.995	0.982	0.872
14	0.999	0.997	0.911	0.999	0.995	1.000	0.999	0.996	0.953
15	0.996	0.987	0.847	0.991	0.991	0.955	0.994	0.989	0.898
16	0.997	0.992	0.833	0.997	0.992	0.938	0.997	0.992	0.882
17	0.999	0.993	0.860	0.998	0.996	0.980	0.999	0.995	0.916
18	0.999	0.982	0.789	0.997	0.993	0.978	0.998	0.987	0.874
19	1.000	0.981	0.903	0.998	0.997	1.000	0.999	0.989	0.949
20	1.000	0.988	0.800	0.996	0.999	0.966	0.998	0.993	0.875
21	1.000	0.983	0.922	0.998	0.999	0.979	0.999	0.991	0.949
22	1.000	0.990	0.886	0.999	0.996	1.000	0.999	0.993	0.939
23	0.995	0.994	0.964	0.998	0.981	0.976	0.997	0.988	0.970
24	0.993	0.996	0.975	0.995	0.994	0.975	0.994	0.995	0.975
<i>Average</i>	<i>0.998</i>	<i>0.971</i>	<i>0.889</i>	<i>0.996</i>	<i>0.993</i>	<i>0.981</i>	<i>0.997</i>	<i>0.981</i>	<i>0.932</i>

The above table provides recall, precision, and F1 score results that allow us to easily observe the balanced distribution of data obtained from the confusion matrix. According to these results, all three classifications have achieved success starting from **0.971** for *interictal* and *preictal* values. When examining the *ictal* class, the *recall*, *precision*, and *F1 score* are **0.889**, **0.981**, and **0.932**, respectively. The lowest ictal result based on recall is observed as **0.787** for *Subject-13*, while the highest ictal result is **0.975** for *Subject-24*.

Table 4.21. Weighted and unweighted accuracy, precision, recall and F1 score 1D-CNN results for NE

Cases	Weighted accuracy	Balanced accuracy	Precision weighted	Precision macro	Recall weighted	Recall macro	F1 score weighted	F1 score macro
01	0.994	0.778	0.994	0.963	0.994	0.778	0.993	0.831
02	0.995	0.726	0.995	0.992	0.995	0.726	0.995	0.774
03	0.993	0.732	0.992	0.822	0.993	0.732	0.992	0.760
04	0.986	0.490	0.986	0.991	0.986	0.490	0.983	0.562
05	0.993	0.810	0.993	0.973	0.993	0.810	0.993	0.865
06	0.986	0.695	0.985	0.987	0.986	0.695	0.985	0.739
07	0.994	0.705	0.994	0.993	0.994	0.705	0.994	0.768
08	0.992	0.967	0.992	0.984	0.992	0.967	0.992	0.975
09	0.995	0.650	0.994	0.658	0.995	0.650	0.995	0.654
10	0.992	0.759	0.992	0.985	0.992	0.759	0.992	0.816
11	0.990	0.883	0.990	0.991	0.990	0.883	0.990	0.931
12	0.983	0.856	0.983	0.989	0.983	0.856	0.982	0.905
13	0.986	0.871	0.986	0.979	0.986	0.871	0.986	0.915
14	0.996	0.783	0.996	0.997	0.996	0.783	0.996	0.839
15	0.968	0.647	0.958	0.647	0.968	0.647	0.962	0.647
16	0.985	0.690	0.984	0.878	0.985	0.690	0.984	0.718
17	0.994	0.830	0.994	0.963	0.994	0.830	0.994	0.878
18	0.994	0.830	0.993	0.962	0.994	0.830	0.993	0.879
19	0.995	0.688	0.995	0.997	0.995	0.688	0.995	0.718
20	0.996	0.868	0.996	0.991	0.996	0.868	0.996	0.916
21	0.996	0.735	0.996	0.996	0.996	0.735	0.996	0.781
22	0.996	0.923	0.996	0.965	0.996	0.923	0.996	0.943
23	0.981	0.736	0.981	0.968	0.981	0.736	0.980	0.786
24	0.979	0.739	0.979	0.987	0.979	0.739	0.977	0.787
<i>Average</i>	<i>0.990</i>	<i>0.766</i>	<i>0.989</i>	<i>0.944</i>	<i>0.990</i>	<i>0.766</i>	<i>0.989</i>	<i>0.808</i>

When examining the 1D-CNN results obtained from NE in the table, a difference of up to **22.4%** was observed between weighted and unweighted accuracy values. A similar difference was also observed in the F1 score (**18.1%**). The main reason for the higher unweighted values, as mentioned in previous sections, is the imbalanced nature of the dataset. The high number of interictal data creates a "*balloon effect*" in the accuracy values, which is detrimental to the classification of the ictal class. This outcome is unfavorable for the ictal classification.

Table 4.22. Precision, recall and F1score results from 3-class classification 1D-CNN (Ictal, preictal and interictal) for NE

Cases	Recall			Precision			F1 score		
	Interictal	Preictal	Ictal	Interictal	Preictal	Ictal	Interictal	Preictal	Ictal
01	0.999	0.981	0.355	0.995	0.991	0.905	0.997	0.986	0.510
02	0.998	0.968	0.212	0.996	0.981	1.000	0.997	0.974	0.350
03	0.998	0.977	0.220	0.995	0.984	0.488	0.997	0.980	0.303
04	1.000	0.588	0.875	0.991	0.991	1.000	0.995	0.738	0.933
05	0.999	0.968	0.462	0.993	0.994	0.931	0.996	0.981	0.617
06	0.996	0.932	0.158	0.988	0.972	1.000	0.992	0.951	0.273
07	1.000	0.888	0.229	0.994	0.985	1.000	0.997	0.934	0.372
08	0.996	0.981	0.923	0.993	0.989	0.971	0.995	0.985	0.946
09	0.999	0.951	0.022	0.996	0.978	1.000	0.997	0.964	0.043
10	0.999	0.965	0.313	0.992	0.994	0.969	0.996	0.979	0.473
11	1.000	0.893	0.755	0.990	0.992	0.991	0.995	0.940	0.857
12	0.995	0.981	0.592	0.980	0.987	1.000	0.987	0.984	0.743
13	0.994	0.963	0.657	0.989	0.976	0.973	0.992	0.969	0.785
14	0.999	0.993	0.356	0.996	0.995	1.000	0.998	0.994	0.525
15	0.992	0.948	0.005	0.964	0.975	1.000	0.978	0.962	0.011
16	0.994	0.964	0.111	0.986	0.983	0.667	0.990	0.973	0.190
17	0.998	0.984	0.509	0.996	0.986	0.906	0.997	0.985	0.652
18	0.998	0.967	0.526	0.995	0.981	0.909	0.997	0.974	0.667
19	1.000	0.967	0.097	0.995	0.996	1.000	0.997	0.981	0.176
20	1.000	0.990	0.614	0.996	0.999	0.977	0.998	0.995	0.754
21	0.999	0.990	0.216	0.997	0.991	1.000	0.998	0.990	0.355
22	0.999	0.970	0.800	0.996	0.996	0.903	0.998	0.983	0.848
23	0.992	0.953	0.262	0.985	0.963	0.957	0.989	0.958	0.411
24	0.975	0.995	0.248	0.986	0.974	1.000	0.980	0.984	0.397
<i>Average</i>	<i>0.997</i>	<i>0.948</i>	<i>0.396</i>	<i>0.991</i>	<i>0.986</i>	<i>0.939</i>	<i>0.994</i>	<i>0.964</i>	<i>0.508</i>

According to the evaluation metrics shown in Table 4.22, the recall, precision, and F1 score values were consistently above **0.99** on average for the *interictal* class and above **0.964** for the *preictal* class. However, the most significant observation is that the *ictal* class obtained recall and F1 score values of **0.396** and **0.508**, respectively. This indicates that the optimized 1D-CNN classifier with NE features failed to perform successfully. Furthermore, another notable observation is that although NE yielded successful results in a grid search, it did not achieve the same level of success when applied to all subjects.

All the results presented in this section are summarized in Table 4.23. Subsequently, comparisons were made based on this table.

Table 4.23. The summary of weighted and unweighted (macro) average accuracy results

Classification Algorithms Feature Extraction Methods	Machine Learning Algorithms								Deep Learning Algorithm	
	kNN		RF		J48		SVM		1D-CNN	
	W	U	W	U	W	U	W	U	W	U
Wavelet Transform (WT)	99.41%	91.27%	99.39%	80.09%	98.82%	84.60%	99.87%	95.05%	99.50%	95.30%
Non-linear Energy (NE)	99.81%	96.48%	99.81%	90.02%	99.26%	88.10%	99.90%	96.60%	99.00%	76.60%
*W = weighted and U = unweighted										

According to Table 4.23, the accuracy results obtained using WT features are as follows:

- Machine learning algorithms achieved the highest weighted accuracy result of **99.87%** with *SVM*, while the lowest result of **98.82%** was obtained with *J48*. Deep learning algorithm *1D-CNN* achieved a result of **99.50%**. Based on these results, *SVM* ranked slightly ahead of other classification methods.
- Machine learning algorithms achieved the highest unweighted value of **95.05%** with *SVM*, while the lowest result of **80.09%** was obtained with *RF*. In the case of *1D-CNN*, the accuracy was **95.30%**. According to these results, *1D-CNN* had the highest accuracy value.

According to Table 4.23, the accuracy results obtained using NE features are as follows:

- Machine learning algorithms achieved the highest weighted accuracy result of **99.90%** with *SVM*, while the lowest result of **99.26%** was obtained with *J48*. The same applies to the deep learning algorithm *1D-CNN*, which achieved a result of **99.00%**.
- Machine learning algorithms achieved the highest unweighted value of **96.60%** with *SVM*, while the lowest result of **88.10%** was obtained with *RF*. In the case of *1D-CNN*, the accuracy was **76.60%**.

When comparing NE and WT results, regardless of the weighted/unweighted condition, NE values consistently outperformed WT values in terms of accuracy in the machine learning domain. In 1D-CNN, the weighted *WT* (**99.50%**) achieved a more successful result compared to *NE* (**99.00%**). However, when comparing the unweighted accuracies between WT and NE,

the highest accuracy value for *WT* is obtained with *ID-CNN* at **95.40%**, while for *NE*, it is achieved with *SVM* at **96.60%**.



5. CONCLUSIONS

Epilepsy is a prevalent neurological disorder that affects more than 40 million individuals globally. Although large numbers of antiepileptic drugs have been produced to treat millions of patients, most patients do not respond to these interventions. Epileptic seizures happen at random to impede normal brain function; most patients also incur from a slew of other unanticipated side effects, such as depression, memory loss, and other psychiatric issues. It is critical to predict an upcoming seizure at an early stage to assist the patient in taking necessary actions/healthcare in advance to avoid unintended effects and safeguard the patient's health. Consequently, besides seizure identification and epilepsy diagnosis, predicting an epileptic seizure in a period prior to the seizure onset, in order to have time to take the necessary precautions, has been a significant topic in recent decades.

In this thesis, the aim is to create a time window for patients to receive necessary alerts approximately 30-60 minutes before the occurrence of epileptic seizures, with the purpose of predicting seizures in advance. Frequency and time-based feature extractions have been applied using recorded EEG (Electroencephalogram) data. These frequency and time-based methods were respectively implemented through wavelet transform and nonlinear energy approaches. Subsequently, machine learning and deep learning algorithms were employed for classification purposes.

As a result, in the early seizure prediction, WT features achieved a high average accuracy of **95.30%** using the 1D-CNN classifier with unweighted results. On the other hand, when WT features were classified using machine learning algorithms, SVM achieved the highest average accuracy of **95.05%** with unweighted results. As for NE features, the 1D-CNN yielded an average accuracy of **76.60%** with unweighted results. This indicates that the deep learning technique is not suitable for NE features. Conversely, for NE, machine learning techniques like kNN and SVM produced very close and successful results with unweighted accuracies of **96.48%** and **96.60%**, respectively, with SVM having the highest success rate. Based on the above findings, it can be concluded that even though the dataset is imbalanced, a high level of success is achieved in the 3-class classification from the imbalanced dataset in the early seizure prediction task. Therefore, this method can be successfully applied in early seizure prediction.

6. RECOMMENDATIONS

This thesis presents the creation of a 3-class classification model using two different feature extraction methods to facilitate early diagnosis of epilepsy. Suggestions are provided both to improve the model's performance and for future studies:

1. In addition to the data utilized in deep learning classification, different EEG recordings can be incorporated into the same model for joint utilization.
2. Apart from the 1D-CNN deep learning model, the LSTM model can be employed by identifying suitable parameters.
3. The dataset can be run directly by adding it as input to the 1D-CNN model without applying feature extraction.

REFERENCES

- [1] J. Sirven, I. Epilepsy: a spectrum disorder. Cold Spring Harbor perspectives in medicine, 5(9). 2015.
- [2] C. Günbey, and M. Topçu, Epilepsinin Tarihçesi. Edit: S. Kumandaş and M. Canpolat, Çocukluk Çağı Epilepsileri. 1. Publication. Ankara: Türkiye Klinikleri. p.1-3. 2015.
- [3] Y. Lebrun, and F. Fabbro, Language and Epilepsy, John Wiley & Sons, 2008.
- [4] A. T. Berg, S. F. Berkovic, M. J. Brodie, J. Buchhalter, H. J.. Cross. E. V. W. Boas, J. Engel...I. E. Scheffer, Revised Terminology and Concepts For Organization Of Seizures And Epilepsies: Report Of The ILAE Commission On Classification And Terminology, 2005-2009. Epilepsia, 51(4), 676–685. 2010.
- [5] C. J. Lin, and M. H. Hsieh, Classification of Mental Task from EEG Data Using Neural Networks Based on Particle Swarm Optimization. Neurocomputing, 72(4-6), 1121-1130, 2009.
- [6] H. U. Amin, A. S. Malik, R. F. Ahmad, N. Badruddin, N. Kamel, M. Hussain, and T. W. Chooi, Feature Extraction and Classification for EEG Signals Using Wavelet Transform And Machine Learning Techniques. Australasian Physical & Engineering Sciences in Medicine, 38, 139-149, 2015.
- [7] K. Rasheed, A. Qayyum, J. Qadir, S. Sivathamboo, P. Kwan, L. Kuhlmann, ... & A. Razi, Machine Learning For Predicting Epileptic Seizures Using EEG Signals: A Review, IEEE Reviews in Biomedical Engineering, 14, 139-155, 2020.
- [8] I. Ullah, M. Hussain, and H. Aboalsamh, An Automated System For Epilepsy Detection Using EEG Brain Signals Based On Deep Learning Approach, Expert Systems with Applications, 107, 61-71, 2018.
- [9] K. M. 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, 99, 24-37, 2018.

- [10] A. Craik, Y. He, and J. L. Contreras-Vidal, Deep Learning for Electroencephalogram (EEG) Classification Tasks: A Review. *Journal Of Neural Engineering*, 16(3), 031001, 2019.
- [11] H. A. Shoeb, Application Of Machine Learning To Epileptic Seizure Onset Detection And Treatment (Doctoral dissertation, Massachusetts Institute of Technology), 2019, Accessed From: <https://dspace.mit.edu/handle/1721.1/54669>.
- [12] H. Batar, EEG İşaretlerinin Dalgacık Analiz Yöntemleri Kullanılarak Yapay Sinir Ağları Ile Sınıflandırılması, (Master's Thesis), Department of Electrical and Electronics Engineering, Institute of Sciences, Kahramanmaraş Sutcu Imam University, Kahramanmaraş. Turkey. 2005.
- [13] E. Yazgan, and M. Korürek, Tıp Elektronigi. ITU of Publication Publisher, İstanbul, 1996.
- [14] Boonyakitantont, P., Lek-Uthai, A., Chomtho, K., & Songsiri, J. A review of feature extraction and performance evaluation in epileptic seizure detection using EEG. *Biomedical Signal Processing and Control*, 57, 101702, 2020.
- [15] J. F. Kaiser, On A Simple Algorithm to Calculate The 'energy' of A Signal. In *International Conference on Acoustics, Speech, And Signal Processing*, pp. 381-384, IEEE. 1990
- [16] Z. Zhang, and K. K. Parhi, Low-Complexity Seizure Prediction From Ieeg/Seeg Using Spectral Power And Ratios Of Spectral Power. *IEEE Transactions on Biomedical Circuits and Systems*, 10(3), 693-706, 2015.
- [17] S. W. Ibrahim, R. Djemal, A. Alsuwailem, and S. Gannouni, Electroencephalography (EEG)-Based Epileptic Seizure Prediction Using Entropy And K-Nearest Neighbor (KNN). *Communications in Science and Technology*, 2(1). 2017.
- [18] A. Bhattacharyya, and R. B. Pachori, A Multivariate Approach For Patient-Specific EEG Seizure Detection Using Empirical Wavelet Transform. *IEEE Transactions on Biomedical Engineering*, 64(9), 2003-2015. 2017.
- [19] H. Khan, L. Marcuse, M. Fields, K. Swann, and B. Yener, Focal onset seizure prediction using convolutional networks. *IEEE Transactions on Biomedical Engineering*, 65(9), 2109-2118. 2017.

- [20] 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*, 105, 104-111. 2018.
- [21] A. R. Ozcan, and S. Erturk, Seizure Prediction in Scalp EEG Using 3D Convolutional Neural Networks with An Image-Based Approach. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(11), 2284-2293. 2019.
- [22] X. Yao, Q. Cheng, and G. Q. Zhang, A Novel Independent Rnn Approach to Classification Of Seizures Against Non-Seizures. *arXiv preprint arXiv:1903.09326*. 2019.
- [23] K. Singh, and J. Malhotra, Two-Layer LSTM Network-Based Prediction Of Epileptic Seizures Using EEG Spectral Features. *Complex & Intelligent Systems*, 8(3), 2405-2418, 2022.
- [24] D. P. Dash, M. H. Kolekar, and K. Jha, Surface EEG based epileptic seizure detection using wavelet-based features and dynamic mode decomposition power along with KNN classifier. *Multimedia Tools and Applications*, 1-21. 2021.
- [25] G. Repovs, Dealing with Noise In EEG Recording And Data Analysis. In *Informatica Medica Slovenica*, 15(1), 18-25, 2010.
- [26] J. V. Stone, Independent Component Analysis: An Introduction. *Trends In Cognitive Sciences*, 6(2), 59-64. 2002.
- [27] Delorme, A., & Makeig, S. EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. *Journal of neuroscience methods*, 134(1), 9-21, 2004.
- [28] L. J. Christiano, and T. J. Fitzgerald, The Band Pass Filter. *International Economic Review*, 44(2), 435-465. 2003.
- [29] D. J. Acunzo, G. MacKenzie, G., and M. C. Van Rossum, Systematic Biases in Early ERP And ERF Components as A Result Of High-Pass Filtering. *Journal Of Neuroscience Methods*, 209(1), 212-218. 2012.

- [30] C. D. McManus, K. D. Neubert, and E. Cramer, Characterization and elimination of AC noise in electrocardiograms: a comparison of digital filtering methods. *Computers and Biomedical research*, 26(1), 48-67. 1993.
- [31] L. Guo, D. Rivero, J. Dorado, J. R. Rabunal, and A. Pazos, Automatic Epileptic Seizure Detection In Eegs Based On Line Length Feature And Artificial Neural Networks. *Journal Of Neuroscience Methods*, 191(1), 101-109. 2010.
- [32] R. Jenke, A. Peer, and M. Buss, Feature extraction and selection for emotion recognition from EEG. *IEEE Transactions on Affective computing*, 5(3), 327-339. 2014.
- [33] I. Guler, E. D. Ubeyli, Multiclass Support Vector Machines For EEG-Signals Classification. *IEEE Transactions On Information Technology In Biomedicine*, 11(2), 117-126. 2007.
- [34] H. Adeli, Z. Zhou, and N. Dadmehr, Analysis of EEG records in an epileptic patient using wavelet transform. *Journal of neuroscience methods*, 123(1), 69-87. 2003.
- [35] N. Hazarika, J. Z. Chen, A. C. Tsoi, and A. Sergejew, Classification of EEG signals using the wavelet transform. *Signal processing*, 59(1), 61-72. 1997.
- [36] M. M Shaker, EEG Waves Classifier Using Wavelet Transform and Fourier Transform. *Brain*, 2(3), 169-174, 2006.
- [37] R. S. Pathak, *The Wavelet Transform*. Springer Science & Business Media. 2009.
- [38] K. M. Puk, S. Wang, J. Rosenberger, K. C. Gandy, H. N. Harris, Y. B. Peng, ... and J.C. Chiao, Emotion recognition and EEG analysis using ADMM-based sparse group lasso. *IEEE Transactions on Affective Computing*, 13(1), 199-210. 2019.
- [39] Z. Xu, T. Wang, J. Cao, Z. Bao, T. Jiang, T., and F. Gao, BECT Spike Detection Based On Novel EEG Sequence Features And LSTM Algorithms. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 29, 1734-1743. 2021.
- [40] K. S. Biju, S. Indu, "Energy Modeling of EEG Signal Using the Autoregressive–Empirical Mode Decomposition Method", *Energy and Exergy for Sustainable and Clean Environment*, Vol: 1, pp.373, 2022.

- [41] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, Supervised Machine Learning: A Review Of Classification Techniques. *Emerging Artificial Intelligence Applications In Computer Engineering*, 160(1), 3-24. 2007.
- [42] B. Pang, L. Lee, and S. Vaithyanathan, Thumbs Up? Sentiment classification using machine learning techniques. *arXiv preprint cs/0205070*. 2002.
- [43] B. Mahesh, Machine learning algorithms-a review. *International Journal of Science and Research (IJSR)*. 9, 381-386. 2020.
- [44] G. Bonaccorso, *Machine Learning Algorithms*. Packt Publishing Ltd. 2017.
- [45] A. Shrestha, and A. Mahmood, Review Of Deep Learning Algorithms And Architectures. *IEEE Access*, 7, 53040-53065. 2019.
- [46] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes,... & S. S. Iyengar, A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5), 1-36. 2018.
- [47] I. Syarif, A. Prugel-Bennett, A., and G. Wills, SVM Parameter Optimization Using Grid Search And Genetic Algorithm To Improve Classification Performance. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 14(4), 1502-1509. 2016.
- [48] X. Wang, G. Gong, N. Li, and S. Qiu, Detection analysis of epileptic EEG using a novel random forest model combined with grid search optimization. *Frontiers in human neuroscience*, 13, 52. 2019.
- [49] B. C. Munsell, C.Y. Wee, S. S. Keller, B. Weber, C. Elger, L. A. T. da Silva, ... & L. Bonilha, Evaluation of machine learning algorithms for treatment outcome prediction in patients with epilepsy based on structural connectome data. *Neuroimage*, 118, 219-230. 2015.

- [50] K. Fukumori, H. T. T. Nguyen, N. Yoshida, N., and T. Tanaka, Fully Data-Driven Convolutional Filters With Deep Learning Models For Epileptic Spike Detection. In ICASSP 2019-2019 IEEE International Conference On Acoustics, Speech And Signal Processing (ICASSP). pp. 2772-2776. IEEE. 2019.
- [51] E. Gleichgerrcht, B. Munsell, S. Bhatia, W. A. Vandergrift III, C. Rorden, C. McDonald, ... and L. Bonilha, Deep Learning Applied To Whole-Brain Connectome To Determine Seizure Control After Epilepsy Surgery. *Epilepsia*, 59(9), 1643-1654. 2018.
- [52] Anonymous. h2o.ai. Website: <http://h2o.ai/wiki/grid-search/> 2023.
- [53] P. M. Lerman, Fitting Segmented Regression Models By Grid Search. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 29(1), 77-84. 1980.
- [54] P. Liashchynskyi, P. Liashchynskyi, Grid Search, Random Search, Genetic Algorithm: A Big Comparison For NAS. *arXiv preprint arXiv:1912.06059*. 2019.
- [55] S. Zhang, D. Cheng, Z. Deng, M. Zong, and X. Deng, "A Novel Knn Algorithm With Data-Driven K Parameter Computation," *Pattern Recognition Letters*, vol. 109, pp. 44–54, 2018.
- [56] Z. Deng, X. Zhu, D. Cheng, M. Zong, and S. Zhang, "Efficient knn classification algorithm for big data," *Neurocomputing*, vol. 195, pp. 143–148, 2016.
- [57] V. B. Prasath, H. A. A. Alfeilat, A. Hassanat, O. Lasassmeh, A. S. Tarawneh, M. B. Alhasanat, and H. S. E. Salman, Distance and Similarity Measures Effect on the Performance of K-Nearest Neighbor Classifier--A Review. *arXiv preprint arXiv:1708.04321*. 2017.
- [58] K.-L. Du, M. Swamy, "Support vector machines," in *Neural Networks and Statistical Learning*. Springer, pp. 469–524, 2014.
- [59] M. Li, W. Chen, and T. Zhang, "Automatic epilepsy detection using wavelet-based nonlinear analysis and optimized svm," *Biocybernetics and biomedical engineering*, vol. 36, no. 4, pp. 708–718, 2016.

- [60] T. K. Ho, "Random decision forests," in Proceedings of 3rd international conference on document analysis and recognition, vol. 1. IEEE, pp. 278–282, 1995.
- [61] L. Breiman, Random Forest, vol. 45, Mach Learn, vol. 1, 2001
- [62] N. Bhargava, G. Sharma, R. Bhargava, and M. Mathuria, Decision Tree Analysis On J48 Algorithm For Data Mining. Proceedings Of International Journal Of Advanced Research In Computer Science And Software Engineering, 3(6). 2013.
- [63] Winston, P. H. Artificial intelligence. Addison-Wesley Longman Publishing Co., Inc. 1984.
- [64] Y. LeCun, Y. Bengio, and G. Hinton, Deep Learning. Nature, 521(7553), 436-444. 2015.
- [65] J. Wu, Introduction To Convolutional Neural Networks. National Key Lab for Novel Software Technology. Nanjing University. China, 5(23), 495. 2017.
- [66] S. Albawi, T. A. Mohammed, and S. Al-Zawi, Understanding Of A Convolutional Neural Network. In 2017 International Conference On Engineering And Technology (ICET) (pp. 1-6). IEEE. 2017.
- [67] P. Kim, P. Kim, Convolutional Neural Network. Matlab Deep Learning: With Machine Learning, Neural Networks And Artificial Intelligence, 121-147. 2017.
- [68] G. W. Lindsay, Convolutional Neural Networks As A Model Of The Visual System: Past, Present, And Future. Journal Of Cognitive Neuroscience, 33(10), 2017-2031. 2021.
- [69] F. Mattioli, C. Porcaro, G. Baldassarre, A 1D CNN for high accuracy classification and transfer learning in motor imagery EEG-based brain-computer interface. Journal of Neural Engineering, 18(6), 066053. 2022.
- [70] S. Zhao, J. Yang, Y. Xu, and M. Sawan, Binary single-dimensional convolutional neural network for seizure prediction. In 2020 IEEE International Symposium on Circuits and Systems (ISCAS) (pp. 1-5). IEEE. 2020.

- [71] M. Saini, U. Satija, M. D. Upadhyay, Light-Weight 1-D Convolutional Neural Network Architecture for Mental Task Identification and Classification Based on Single-Channel EEG. arXiv preprint arXiv:2012.06782. 2020.
- [72] B. J. Singstad, and C. Tronstad, Convolutional Neural Network and Rule-Based Algorithms for Classifying 12-lead ECGs. In 2020 Computing in Cardiology, IEEE, 1-4, 2020.
- [73] D. Shah, K. G. Gopan, and N. Sinha, N. An investigation of the multi-dimensional (1D vs. 2D vs. 3D) analyses of EEG signals using traditional methods and deep learning-based methods. *Frontiers in Signal Processing*, 52. 2022.
- [74] G. C. Jana, R. Sharma, and A. Agrawal, A 1D-CNN-spectrogram based approach for seizure detection from EEG signal. *Procedia Computer Science*, 167, 403-412. 2020.
- [75] A. Shoeibi, D. Sadeghi, P. Moridian, N. Ghassemi, J. Heras, R. Alizadehsani, ... & J. M. Gorriz, Automatic diagnosis of schizophrenia in EEG signals using CNN-LSTM models. *Frontiers in neuroinformatics*, 58. 2021.
- [76] G. Xu, T. Ren, Y. Chen, and W. Che, A One-Dimensional Cnn-Lstm Model For Epileptic Seizure Recognition Using Eeg Signal Analysis. *Frontiers In Neuroscience*, 14, 578126. 2020.
- [77] Ö. Yıldırım, U. B. Baloglu, and U. R. Acharya, A deep convolutional neural network model for automated identification of abnormal EEG signals. *Neural Computing and Applications*, 32, 15857-15868. 2020.
- [78] S. Sharma, S. Sharma, and A. Athaiya, Activation functions in neural networks. *Towards Data Sci*, 6(12), 310-316. 2017.
- [79] A. F. Agarap, Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375.2018.
- [80] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, Understanding deep neural networks with rectified linear units. arXiv preprint arXiv:1611.01491. 2016.
- [81] W. Liu, Y. Wen, Z. Yu, and M. Yang, Large-Margin Softmax Loss For Convolutional Neural Networks. arXiv preprint arXiv:1612.02295. 2016.

- [82] S. Albawi, T. A. Mohammed, S. Al-Zawi, Understanding of A Convolutional Neural Network. In 2017 International Conference on Engineering And Technology (ICET), pp. 1-6, IEEE. 2017.
- [83] K. O'Shea, R. Nash, An Introduction to Convolutional Neural Networks. Arxiv Preprint Arxiv:1511.08458. 2015.
- [84] M. Sun, Z. Song, X. Jiang, J. Pan, Y. Pang, Learning Pooling For Convolutional Neural Network. Neurocomputing, 224, 96-104. 2017.
- [85] H. Gholamalinezhad, and H. Khosravi, Pooling Methods in Deep Neural Networks, A Review. Arxiv Preprint Arxiv:2009.07485. 2020.
- [86] A. G. Schwing, and R. Urtasun, Fully Connected Deep Structured Networks. Arxiv Preprint Arxiv:1503.02351. 2015.

APPENDIXES

APPENDIX A: Wavelet Transform pseudocode and its explanation.

- ***“mainFiles” (string array)***, is the directory where the folder names of the subjects are located.
- ***“featMat” (numeric array)***, is the location where the energy band values are saved.
- ***“labelMat” (numeric array)***, are values assigned as 1, 2, and 3 respectively to indicate the data labels (interictal, preictal, and ictal) within the array.
- ***“SubjectMat” (numeric array)***, is where all the energy bands and data outputs obtained from a subject are concatenated side by side.
- ***“files” (string array)***, is the location where the data filenames within each subject folder are found.
- ***“data” (numeric array)***, is where the EEG data inside the folder is imported into the program.
- ***“segment” (numeric array)***, represents one array based on the sampling frequency (256Hz).
- ***“feat” (numeric array)***, after each segment is sent to wavelet transform and the energy bands are obtained, it is the location where the array is saved.
- ***“p” (integer variable)***, is a number of the segment. It also called counter that can be updated until data/256.

Pseudocode-1:

```
1: mainFiles  $\leftarrow$  list of all subjects' folder names.
2: FOR (i from 1 to length of mainFiles) do:
3:   featMat = []
4:   labelMat = []
5:   subjectMat = []
6:   p = 0
7:   files  $\leftarrow$  list of EEG data names in a subject
8:   FOR (j from 1 to length of files) do:
9:     data  $\leftarrow$  import each EEG data from files
10:    FOR (k from 1 to row size of data, step 256) do:
11:      p = p+1
12:      segment  $\leftarrow$  each part of the data divided by 256
13:      feat  $\leftarrow$  getFeature(segment)
14:      featMat  $\leftarrow$  insert all feats line by line
15:      IF files(j) = “Interictal”:
16:        labelMat  $\leftarrow$  1*p
17:      ELSE IF files(j) = “Prerictal”:
18:        labelMat  $\leftarrow$  2*p
19:      ELSE:
20:        labelMat  $\leftarrow$  3*p
21:   subjectMat  $\leftarrow$  merge featMat and labelMat
```

The "*mainFiles*" holds the files for 24 patients and allows accessing the data of each patient by iterating over the length of this array. The "*files*" array lists the names of the data within a subject. Then, this list is taken into a loop, and the relevant EEG data of the recorded names are imported into the "*data*" array. The "*data*" array is then "*segmented*" in a new loop according to the sampling frequency (256Hz). This allows extracting the energy in brain waves for each sampling moment in an easy and orderly manner. "*getFeature*" is a function within Pseudocode-1. It takes the "*segment*" as input and uses wavelet transform to obtain the energy bands. Pseudocode-2 provides a detailed explanation of this function. Each energy band obtained as the output of the function is saved in the "*feat*" array. Since the recorded "*feat*" arrays of a single patient follow each other in time, they are appended row by row into the "*featMat*" array. By leveraging the file names, the EEG feature outputs are determined (labeled) as interictal, preictal, and ictal, respectively, and numbered 1, 2, and 3. The numbered labels are replicated for the number of segments generated ("*p*") and saved in the "*labelMat*" array. The row lengths of "*labelMat*" and "*featMat*" are the same. Based on this, all energy bands and data outputs obtained from a subject are combined and saved in the "*SubjectMat*". In its final form, the EEG features of a subject are ready for classification.

- "**waves**" (**numeric array**), is the location where the power values obtained from all brain waves are saved.
- "**ch**" (**numeric variable**), represents the parameter that the for loop that iterates over the number of channels within a segment (18).
- "**info**" (**numeric array**), is the section where all the data within the selected channel is stored.
- "**getWaveletSpectrum**", is the function where the average spectrum values in the frequency range of 0.1-120Hz are obtained for the selected channel.
- "**spect and freq**" (**numeric array**), are output arrays returned from "**getWaveletSpectrum**" and finds the spectrum and frequency values respectively.
- "**delta, theta, alpha, beta, gamma**" (**numeric array**), holds the spectrum values for the frequency ranges (0.1-3Hz), (3-7Hz), (7-12Hz), (12-30Hz), and (30-120Hz) respectively.
- "**Pdelta, Ptheta, Palpha, Pbeta and Pgamma**" (**float variable**), stores power value of the spectrums that we found previously.

Pseudocode 2:

```
1: FUNCTION getFeature(segment)
2:   waves = []
3:   FOR (ch from 1 to length of segment column) do:
4:     info  $\leftarrow$  EEG data inside the current channel in a segment
5:     [spect , freq] = getWaveletSpectrum(info , 256 , 0.1 , 120)
6:     delta  $\leftarrow$  spectrum value between 0.1-3Hz
7:     theta  $\leftarrow$  spectrum value between 3-7Hz
8:     alpha  $\leftarrow$  spectrum value between 7-12Hz
9:     beta  $\leftarrow$  spectrum value between 12-30Hz
10:    gamma  $\leftarrow$  spectrum value between 30-120Hz
11:    Pdelta, Ptheta, Palpha, Pbeta, Pgamma  $\leftarrow$  power values of the brain waves
12:    waves  $\leftarrow$  merge all power values of the brain waves
```

Pseudocode-2 is the "**getFeature**" function within Pseudocode-1, where the energy bands of brain waves obtained through wavelet transform from the "*segment*" are calculated. In Pseudocode-2, "*segment*" is taken as input. The main objective here is to determine the spectrum values of brain waves (delta, alpha, beta, theta, gamma) between 0.1 and 120Hz and calculate their energies. In the first for loop, there is a variable called "*ch*" representing each channel among the 18 channels in the "*segment*" for calculating the energies sequentially. The data taken from a channel in the "*segment*" is stored in the "*info*" array and passed to the "*getWaveletSpectrum*" function to obtain the spectrum values. Thus, the spectrum values of brain waves obtained from one channel are taken from the intervals (0.1-3Hz), (3-7Hz), (7-12Hz), (12-30Hz), and (30-120Hz) for delta, theta, alpha, beta, and gamma, respectively. Subsequently, using the spectrum values, the power values (*Pdelta*, *Ptheta*, *Palpha*, *Pbeta*, and *Pgamma*) are calculated. Finally, the "*waves*" array is the output of the function, containing the concatenated energy bands.

APPENDIX B: Non-Linear Energy pseudocode and its explanation.

- “*energy*” (*numeric array*), is the location where the time-based non-linear energies obtained from all brain waves are recorded.
- “*allEnergy*” (*numeric variable*), all energy data within a subject is collected in a single array

Pseudocode 3:

```
1: FUNCTION nonLinearEnergy(segment)
2:   energy = []
3:   allEnergy = []
4:   FOR (i from 1 to length of segment column) do:
5:     FOR (j from 2 to 255) do:
6:       energy ← apply formula 3.7
7:   allEnergy ← summation of the time-based energies
8: END FUNCTION
```

The time-based "*nonLinearEnergy*" function involves calculating the energy obtained from EEG data. The "*energy*" within this function is the array that is recorded after calculating the energies from each channel within a segment using Formula 3.10. Finally, "*allEnergy*" is the array where the energies from all channels within a segment are recorded, and it serves as the output of the function.

APPENDIX C: Pseudocode of the kNN's grid search

- **"kNN" (function)**, is a function that is called to run the kNN classifier.
- **"k" (numeric array)**, is the assigned values to enclose the number of nearest neighbors within a circle.
- **"metric" (string array)**, is array that holds the distance calculation methods.
- **"kNNParams" (numeric array)**, is an array that combines the values of *k* and *metric* parameters.
- **"gridSearchCV" (function)**, is the estimator function where the grid search is performed.
- **"gridSearch" (function)**, is the function where the gridSearchCV function is executed with its parameters.
- **"gridResults" (numeric array)**, is the best parameters obtained as a result of adding the dataset as an input.
- **"trainFeatures, trainLabels" (numeric arrays)**, are the features and outputs of the dataset, respectively,

Pseudocode 4:

```
1: kNN = KNeighborsClassifier () ← call the kNN function.
2: gridParams, gridSearch, gridResults = []
3: k = range (1,50,2) ← import k parameter
4: metric = ["euclidean", "manhattan","minkowski"] ← import distance parameter
5: kNNParams = [k, metric] ← combine 2 parameters
6: gridSearch = gridSearchCV (kNN, kNNParams) ← add kNN function and its parameters
```

In Pseudocode-4, first, the *KNeighborsClassifier* function, which is the function of the "kNN" classifier, is introduced to the system. Then, a variable called "kNNParams" is created, which contains the classifier parameters ("k" and "metrics"). Subsequently, "kNNParams" and "kNN" estimator are passed to enable grid search. This way, a grid search framework is established using the specified parameters for the kNN model. Finally, the data is added to the "gridSearch" part to find the most optimal values.

APPENDIX D: Pseudocode of the SVM's grid search

- **"svm" (function)**, is a function that is called to run the SVM classifier.
- **"C" (numeric array)**, regularization parameter for tolerance of errors.
- **"gamma" (numeric array)**, is the parameters given for the hyperspace curve lie within the range of 0 to 1.
- **"kernels" (string array)**, non-linear transformation methods
- **"SVMParams" (numeric array)**, is an array that combines the values of *C*, *gamma* and *kernel* parameters.
- **"gridSearchCV" (function)**, is the estimator function where the grid search is performed.
- **"gridSearch" (function)**, is the function where the gridSearchCV function is executed with its parameters.
- **"gridResults" (numeric array)**, is the best parameters obtained as a result of adding the dataset as an input.
- **"trainFeatures, trainLabels" (numeric arrays)**, are the features and outputs of the dataset.

Pseudocode 5:

```
1: svm = SupportVectorClassification() ← call the SVM function.  
2: gridParams, gridSearch, gridResults = []  
3: C = [0.1, 1, 10, 100, 1000] ← import C parameter  
4: gamma = [1, 0.1, 0.01, 0.001, 0.0001] ← import gamma parameter  
5: kernels = ['linear', 'RBF', 'polynomial'] ← import kernel parameter  
5: SVMParams = [C, gamma, kernel] ← combine 3 parameters  
6: gridSearch = gridSearchCV (svm, SVMParams) ← add SVM function and its parameters
```

In Pseudocode-5, first, the *SupportVectorClassification* function, which is the function of the "svm" classifier, is introduced to the system. Then, a variable called "SVMParams" is created, which contains the classifier parameters ("C", "gamma" and "kernels"). Subsequently, "SVMParams" and "svm" estimator are passed to enable grid search. This way, a grid search framework is established using the specified parameters for the SVM model. Finally, the data is added to the "gridSearch" part to find the most optimal values.

APPENDIX E: Pseudocode of the RF and J48's grid search

- **“rf” (function)**, is a function that is called to run the RF classifier.
- **“randomState” (numeric array)**, regularization parameter for tolerance of errors.
- **“n_estimators” (numeric array)**, is the parameters given for number of subsets.
- **“max_depth” (string array)**, stores number of branches.
- **“criterion” (string array)**, includes two types of calculation methods.
- **“RFParams” (list)**, is an array that combines the values of randomState, n_estimators, max_depth and criterion parameters.
- **“gridSearchCV” (function)**, is the estimator function where the grid search is performed.
- **“gridSearch” (function)**, is the function where the gridSearchCV executed.
- **“gridResults” (numeric array)**, is the best parameters obtained as a result of adding the dataset as an input.

Pseudocode 6:

```
1: rf = RandomForestClassifier() ← call the RF function.  
2: gridParams, gridSearch, gridResults = []  
3: randomState = [0 50 100] ← import random parameters  
4: n_estimators = [64 128 256 512] ← import number of subsets  
5: max_depth = [4 8 16 32] ← import maximum branches of the tree  
5: criterion = [entropy, gini] ← import calculation methods  
6: RFParams = [random state, n_estimators, criterion] ← combine 4 parameters  
7: gridSearch = gridSearchCV (rf, RFParams) ← add RF function and its parameters
```