FIRAT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
TÜRKİYE

A PERFORMANCE ANALYSIS OF DEEP LEARNING ALGORITHMS FOR EMOTION DETECTION BASED ON EEG

Awf Abdulrahman Ramadhan

Master's Thesis

DEPARTMENT OF SOFTWARE ENGINEERING

AUGUST 2021

FIRAT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
TÜRKİYE

Department of Software Engineering

Master's Thesis

A PERFORMANCE ANALYSIS OF DEEP LEARNING ALGORITHMS FOR EMOTION DETECTION BASED ON EEG

Author
Awf Abdulrahman Ramadhan

Supervisor
Dr. Öğr. Üyesi Muhammet BAYKARA

AUGUST 2021
ELAZIG

FIRAT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

TÜRKİYE

Department of Software Engineering

Master's Thesis

Title:	A Performance Analysis of Deep Learning Algorithms for Emotion Detection Based on EEG
Author:	Awf Abdulrahman Ramadhan
Submission Date:	22 June 2021
Defense Date:	13 August 2021

THESIS APPROVAL

This thesis, which was prepared according to the thesis writing rules of the Graduate School of Natural and Applied Sciences, Fırat University, was evaluated by the committee members who have signed the following signatures and was unanimously approved after the defense exam made open to the academic audience.

		Signature
Supervisor:	Dr. Öğr. Üyesi Muhammet BAYKARA Firat University, Faculty of Technology	Approved
Chair:	Dr. Öğr. Üyesi Ahmet Arif AYDIN İnönü University, Faculty of Engineering	Approved
Member:	Dr. Öğr. Üyesi Yaman AKBULUT Fırat University, Faculty of Technology	Approved

This thesis was approved by the Administrative Board of the Graduate School on

..... / ..... / 20 .....

Signature

Doç. Dr. Kürşat Esat ALYAMAÇ
Director of the Graduate School

DECLARATION

I hereby declare that I wrote this Master's Thesis titled “A Performance Analysis of Deep Learning Algorithms for Emotion Detection Based on EEG” in consistent with the thesis writing guide of the Graduate School of Natural and Applied Sciences, Firat University. I also declare that all information in it is correct, that I acted according to scientific ethics in producing and presenting the findings, cited all the references I used, express all institutions or organizations or persons who supported the thesis financially. I have never used the data and information I provide here in order to get a degree in any way.

13 August 2021

Awf Abdulrahman Ramadhan

PREFACE

This thesis aims to use deep learning techniques to classify human emotions based on physiological brain signals (EEG). The importance of detecting human emotion is to help researchers understand human behavior and the extent to which human emotions affect people's daily lives. The difficulty in recognizing human emotion using an EEG is that the signal is usually unstable. And external noise that affects the signal, such as the hand movement while collecting data from the person. In this thesis, the physiological computer game data set GAMEEMO was used. In this thesis, human emotions are classified using the discrete emotions model (two classes positive and negative model) and the dimensional emotions model (four classes arousal-valence model).

No person, institution, and organization contributed directly or indirectly to the preparation of this thesis.

Awf Abdulrahman Ramadhan
ELAZIG, 2021

TABLE OF CONTENTS

Page

Preface ... iv
Table of Contents ... v
Abstract ... vii
Özet viii
List of Figures ... ix
Lıst of Tables ... xi
List of Appendices ... xii
Symbols and Abbreviations ... xiii

1. INTRODUCTION ... 1
1.1. Overview ... 1
1.2. Problem Statement ... 1
1.3. Aims of the Thesis ... 1
1.4. Literature Review ... 2
1.5. Thesis Layout ... 7

2. BACKGROUND AND GENERAL TECHNIQUES ... 8
2.1. Introduction ... 8
2.2. Emotion Theories In the Literature ... 8
2.3. Electroencephalogram Device ... 9
2.4. The Stimuli ... 10
2.5. Feature Extraction Methods ... 11
2.5.1. Statistical Features ... 11
2.5.2. Welsh Power Spectral Density ... 12
2.5.3. Fast Fourier Transform ... 12
2.5.4. Wavelet Packet Decomposition ... 12
2.6. Artificial Neural Networks ... 13
2.6.1. ANN Architecture ... 13
2.6.2. ANN Components ... 15
2.7. Deep Learning ... 16
2.7.1. Convolutional Neural Networks ... 16
2.7.2. 1D Convolution Neural Network ... 19
2.7.3. Recurrent Neural Networks (RNN) with Long-Short Term Memory (LSTM) ... 19
2.8. Evaluation Metrics of the Semantic Segmentation ... 23
2.8.1. Confusion Matrix ... 24
2.8.2. Global Accuracy, Sensitivity, and Specificity ... 24
2.8.3. Precision, Recall, and F1-Score ... 25
2.9. Python Software Environment ... 25
2.9.1. Deep Learning Toolbox ... 25
2.9.2. Graphic Processing Units (GPUs) ... 26
2.9.3. Collaboratory (Colab): ... 27

3. MATERIAL AND METHODS ... 28
3.1. Introduction ... 28
3.2. GAMEEMO Dataset ... 28
3.3. Feature Extraction Methods ... 30

3.4. Data Preprocessing and Normalization 33
3.5. Data Partitioning 33
.3.6 Building Proposed Models 34
3.6.1. 1D CNN Architecture 34
3.6.2. LSTM Architecture 35
3.6.3. Hybrid Model: 1D CNN + LSTM 36
3.7. Convolutional Layers 38
3.8. ReLU Activation Function Layers 39
3.9. Max-Pooling Layers 39
3.10. Training Our Models 40
4. RESULTS AND DISCUSSION 41
4.1. Introduction 41
4.2. Results 41
4.2.1. Global Accuracy, Sensitivity, and Specificity 41
4.2.2. The Precision, Recall, and F1-Score 43
4.2.3. Confusion Matrix Metric 47
4.2.4. Training Progress Plot 51
4.3. Comparison with Other Related Models 53
5. CONCLUSION AND FUTURE WORKS 54
5.1. Conclusion 54
5.2. Future Works Suggestions 54
References 56
Appendices 61
Curriculum Vitae

ABSTRACT

A Performance Analysis of Deep Learning Algorithms for Emotion Detection Based on EEG

Awf Abdulrahman Ramadhan

Master's Thesis

FIRAT UNIVERSITY
Graduate School of Natural and Applied Sciences

Department of Software Engineering

August 2021, Page: xiii + 65

Human emotions are a person's reaction to events around him. Human emotions play an important role in a person's life and greatly impact his behavior based on his daily mood. The development of computers in the past two decades has led to increased interest in human-machine interaction applications. Among these applications are human emotions applications, which have gained a lot of attention lately by psychologists. The main aim of this thesis is to detect human emotions using deep learning techniques. In this thesis, a physiological data set GAMEEMO was used to classify human emotions based on the discrete emotions model and the dimensional emotion model. The GAMEEMO dataset is an EEG dataset consisting of 128 Hz signals for 28 subjects. There are three main tasks of this thesis. The first task: Four feature extraction methods (statistical features (SF), a combination of SF and Welsh power spectral density (PSD), a combination of SF and Fast Fourier transform (FFT), and the combination of SF and wavelet packet decay (WPD) are applied on the pre-processed EEG signals to extract more features. The second task: Split the data based on two classes and four classes. The third task: Build a convolutional neural network (CNN) using a 1D CNN, a recurrent neural network (RNN) using a long-term memory (LSTM), and a hybrid model 1D CNN + LSTM architectures. This thesis achieved high performance using the WPD+SF feature extraction method, with more than 98% accuracy in both emotions' models.

Keywords: Deep learning, EEG, Emotion recognition, CNN, RNN, LSTM, Wavelet packet decomposition

ÖZET

EEG Sinyallerine Dayalı Duygu Tespiti için Derin Öğrenme Algoritmalarının Performans Analizi

Awf Abdulrahman Ramadhan

Yüksek Lisans Tezi

FIRAT ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü

Yazılım Mühendisliği Anabilim Dalı

Ağustos 2021, Sayfa: xiii + 65

İnsan duyguları, bir kişinin etrafındaki olaylara tepkisidir. İnsan duyguları, bir kişinin hayatında önemli bir rol oynar ve günlük ruh haline göre davranış ve eylemleri üzerinde büyük bir etkiye sahiptir. Son yirmi yılda bilgisayarların gelişimi, insan-makine etkileşimi uygulamalarına olan ilginin artmasına neden olmuştur. Bu uygulamalar arasında son zamanlarda psikologlar tarafından oldukça fazla ilgi gören insan duyguları uygulamaları yer almaktadır. Bu tezin temel amacı, derin öğrenme tekniklerini kullanarak insan duygularını tespit etmektir. Bu tezde, insan duygularını ayrık duygular modeline ve boyutsal duygu modeline göre sınıflandırmak için fizyolojik bir veri seti olan GAMEEMO kullanılmıştır. GAMEMO veri seti, 28 denek için 128 Hz sinyallerden oluşan bir EEG veri setidir. Bu tezin üç ana görevi vardır. İlk görev: Dört özellik çıkarma yöntemi (istatistiksel özellikler (SF), SF ve Welsh güç spektral yoğunluğunun (PSD) bir kombinasyonu, SF ve Fast Fourier dönüşümünün (FFT) bir kombinasyonu ve SF ve dalgacık paket ayrıştırmasının birleşimi (WPD), daha fazla özellik çıkarmak için önceden işlenmiş EEG sinyallerine uygulanır. İkinci görev: Verileri iki sınıfa ve dört sınıfa göre ayırmaktır. Üçüncü görev: 1D CNN kullanarak bir evrişimli sinir ağı (CNN), Uzun süreli bellek (LSTM) kullanan sinir ağı (RNN) ve hibrit bir model 1D CNN + LSTM mimarilerinin oluşturulmasıdır. Bu tezde, WPD+SF öznitelik çıkarma yöntemini kullanarak her iki duygu modelinde de %98'den fazla doğrulukla yüksek performans elde edilmiştir.

Anahtar Kelimeler: Derin öğrenme, EEG, Duygu tanıma, CNN, RNN, Dalgacık paket ayrıştırması

LIST OF FIGURES

Page

Figure 2.1. Plutchik's theory 9

Figure 2.2. James and Lange's theory, arousal-valence model 2-D space. 9

Figure 2.3. Arousal-valence 3-D model 10

Figure 2.4. The 10-20 system A) front side B) top side 10

Figure 2.5. The WPD tree (A= Approximation coefficient, and D= detail coefficient). 13

Figure 2.6. The Synaptic Connections Between Neurons [53]. 14

Figure 2.7. Feed-Forward MLP Neural Network [55]. 14

Figure 2.8. Feed-Back RNN architecture [56]. 15

Figure 2.9. CNN Layers [63]. 17

Figure 2.10. Converting the input data into a feature Map [65]. 18

Figure 2.11. Max Pooling Operation [64]. 18

Figure 2.12. An example of the architecture of the 1D CNN [70]. 19

Figure 2.13. A simple RNN architecture [77]. 20

Figure 2.14. Single-cell of LSTM network [70]**.** 21

Figure 2.15. The general architecture of the LSTM network [79]. 21

Figure 2.16. LSTM's gates [72]. 22

Figure 2.17. The architecture of the LSTM model for human emotion recognition [70]. 22

Figure 2.18. The sigmoid and tanh function [72]. 23

Figure 3.1. The flowchart of this thesis 28

Figure 3.2. An example of EEG signals using the F3 channel. 29

Figure 3.3. The code of statistical feature method used in this thesis 30

Figure 3.4. The PSD method is used to extract features from the EEG signals. 31

Figure 3.5. The FFT method used to extract features from EEG signals 31

Figure 3.6. The signals were obtained from subject number 20 (according to the G2) using FFT + SF method. 32

Figure 3.7. The WPD method, the first of three levels 32

Figure 3.8. Data normalization method using the Scikit-Learn library 33

Figure 3.9. The code used to make training and testing sets 34

Figure 3.10. The 1D CNN architecture is used in this thesis. 34

Figure 3.11. The LSTM model is used in this thesis. 36

Figure 3.12. The hybrid model (1D CNN + LSTM) architecture [70]. 37

Figure 3.13. Feature Mapping from Input using 3x3 kernels..... 38

Figure 3.14. ReLU activation function 39

Figure 3.15. Max pooling layer 40

Figure 4.1. The performance of models through all methods using binary classification. 43

Figure 4.2. The performance of models through all methods using multi-class classification. 43

Figure 4.3. The precision, recall, and f1-score for binary classification with all methods. 44

Figure 4.4. The performance of the LANV zone among all metrics and all methods. 46

Figure 4.5. The performance of the LAPV zone among all metrics and all methods. 46

Figure 4.6. The performance of the HANV zone among all metrics and all methods. 47

Figure 4.7. The performance of the HAPV zone among all metrics and all methods. 47

Figure 4.8. The confusion matrix metric for the 1D CNN model. (a) represents the SF method with binary classification. (b) describes the SF method with multi-classification. (c) illustrates the PSD method with binary classification. (d) represents the PSD method with multi-classification. (e) describes the FFT method with binary classification. (f) illustrates the FFT method with multi-classification. (g) represents the WPD method with binary classification. (h) describes the WPD method with multi-classification. 48

Figure 4.9. The confusion matrix metric for the LSTM model. (a) represents the SF method with binary classification. (b) describes the SF method with multi-classification. (c) illustrates the PSD method with binary classification. (d) represents the PSD method with multi-classification. (e) describes the FFT method with binary classification. (f) illustrates the FFT method with multi-classification. (g) represents the WPD method with binary classification. (h) describes the WPD method with multi-classification. 49

Figure 4.10. The confusion matrix metric for the hybrid model. (a) represents the SF method with binary classification. (b) illustrates the SF method with multi-classification. (c) describes the PSD method with binary classification. (d) represents the PSD method with multi-classification. (e) describes the FFT method with binary classification. (f) illustrates the FFT method with multi-classification. (g) represents the WPD method with binary classification. (h) describes the WPD method with multi-classification. 50

Figure 4.11. Compiling and fitting 1D CNN model 51

Figure 4.12. Training Progress Plot for 1D CNN model using SF method and multi-classification. 51

Figure 4.13. Training Progress Plot for 1D CNN model using FFT method and binary classification. 51

Figure 4.14. Training Progress Plot for LSTM model using PSD method and multi-classification. 52

Figure 4.15. Training Progress Plot for LSTM model using FFT method and binary classification. 52

Figure 4.16. Training Progress Plot for the hybrid model using the FFT method and binary classification. 52

Figure 4.17. Training Progress Plot for the hybrid model using PSD method and multi-classification. 53

LIST OF TABLES

Page

Table 2.1. The Confusion Matrix for Two Class [81]. 24

Table 3.1. The stimuli used in the GAMEEMO database 29

Table 3.2. The layer parameters of the 1D-CNN model using SF, SF+PSD, and SF + WPD methods, multi-class classification 35

Table 3.3. The layer parameters of the LSTM model using SF method, multi-class classification 36

Table 3.4. The total params for the hybrid model using the SF method. 37

Table 3.5. Training Options of the Modified models. 40

Table 4.1. The accuracy, sensitivity, and specificity parameters for the 1D CNN model. 41

Table 4.2. The accuracy, sensitivity, and specificity parameters for the LSTM model. 42

Table 4.3. The accuracy, sensitivity, and specificity parameters for the hybrid 1D CNN + LSTM model. 42

Table 4.4. The precision, recall, and f1-score parameters for binary classification (discrete model) with all feature extraction methods 44

Table 4.5. The precision, recall, and f1-score for multi-class classification using the Statistical feature method. 45

Table 4.6. The precision, recall, and f1-score for multi-class classification using SF + PSD method. 45

Table 4.7. The precision, recall, and f1-score for multi-class classification using SF + FFT method. 45

Table 4.8. The precision, recall, and f1-score for multi-class classification using SF + WPD method. 45

Table 4.9. The Performance Comparison Between the Proposed Modified models and Some Other Related Work. 53

LIST OF APPENDICES

Page

Appendix- 1: The Python Main Code for the 1D CNN model using the SF method. 61

Appendix- 2: Testing process for the 1D CNN model using SF. 65

SYMBOLS AND ABBREVIATIONS

Abbreviations

EEG : Electroencephalography
ECG : Electrocardiography
PPG : Photoplethysmography
GSR : Galvanic skin response
SKT : Skin temperature
SF : Statistical feature
FFT : Fast fourier transform
WPD : Wavelet packet decomposition
LSTM : Long-short term memory
RNN : Recurrent neural network
CNN : Convolutional neural network
ANN : Artificial neural network
FFN : Feed forward network
1D CNN : One-dimensional convolutional neural network

1. INTRODUCTION

1.1. Overview

Emotion is a complex and complicated state of feelings that causes psychological and physical changes in people, which affects their thinking and behavior based on their mood and personality. Likewise, emotion is an essential element in a person's daily activity because it is considered a person's response to the events around him [1]. For example, a person rejoices when he gets a job and grieves on the contrary [2]. Basic emotions such as joy, sadness, anger, and disgust always affect individuals, whether willingly or unwillingly, and thus are powerfully reflected in society's status. Society usually does not accept people who suffer from negative emotions and are affected by this situation physiologically and psychologically [3]. On the contrary, society accepts people who always have positive emotions, as they live with better living standards and more extended periods [4]. The concept of detecting human emotions has become a hot and interesting topic in the past few decades [5] in many fields such as psychology, neuroscience, computer science, and artificial intelligence [6].

1.2. Problem Statement

The tremendous development in computers has increased researchers' interest in using the interaction between humans and computer systems to automatically extract emotions from people [7]. We need access to the sources of emotions in our bodies to understand and recognize human emotions [2] automatically. There are many methods in which the interaction between a computer and persons can be used to access the sources of emotions and extract emotions [8], such as voice signals [9], facial expressions [10], body language, and physiological signals [5]. Physiological signals are among the most reliable methods to extract emotions because a person cannot control his internal body signals. In contrast, he can control and manipulate his body language, voice signals, and even facial expressions. For these reasons, physiological signals have become of paramount importance for emotion. The most common physiological signals in the literature that have been used to detect emotions are EEG (Electroencephalography), ECG (Electrocardiography), PPG (Photoplethysmography), GSR (Galvanic Skin Response), and SKT (Skin Temperature) [11].

1.3. Aims of the Thesis

The thesis aims to analyze the EEG signals to detect human emotions using deep learning algorithms. In this thesis, there are three basic tasks. The first task is to apply four methods of extracting features (statistical features (SF), combination of SF and the welsh power spectral density (PSD) method, the combination of SF and fast Fourier transform (FFT), and the

combination of SF and wavelet packet decomposition (WPD)). The second task is to divide the data obtained after extracting the features in the first task based on the discrete emotions model (two classes negative and positive emotions) and the dimensional emotions model (four classes arousal-valence emotions). The third task is to improve the performance of the standard Convolutional neural network (CNN) using one-dimensional convolutional neural network (1D CNN) architectures, recurrent neural network (RNN) using Long-short term memory (LSTM) architecture, and the combination between 1D CNN + LSTM to discover human emotions using GAMEEMO data set. The main reason to choose this dataset is that it is a new dataset in literature. Also, stimuli were used for the first time in literature (computer games) considered aural-visual stimuli. The process of extracting human emotions is implemented by modifying 1D CNN, LSTM, and hybrid (1D CNN+LSTM) models. This modification involves reducing the number of kernels or neuron numbers at each layer. Changing the kernel's numbers of the 1D CNN and LSTM architecture will significantly reduce the time needed to train the model and increase the performance efficiency in several evaluation metrics.

1.4. Literature Review

Roneel V. Sharan et al, 2020 [12]. A method has been proposed for detecting personality traits by using physiological signals. The research team displayed a set of image and video stimuli on 18 subjects. Data were collected from the participants in the experiment using a portable and wearable EEG device (14 channels). Images and videos were used as stimuli in this study. Then the one-way analysis of variance (ANOVA) method was used to select the lowest set of features. After that, machine learning algorithms were applied (k-Nearest Neighbor (KNN), Logistic Regression (LR), Naïve Bayes (NB), and Support Vector. (SVM) with RBF kernel) to classify the model. The research team obtained the following results (89.24 using KNN, 61,11 using LR, 79,17 using NB, and 95,45 using SVM).

Taruv Harshita Priya et al. 2020 [13] presented a binary classification to detect stress using brain signals (EEG), where they took data from the physical bank and applied Power spectral density (PSD) to extract the features using a fast Fourier transform (FFT). Then, after extracting the features, they used the SVM and KNN machine learning algorithms to create a classification model. This study obtained an accuracy of 99.42% from FP1 using KNN and 98.72% from FP1 using SVM (KS=2.5).

Tomas Matlovic et al. 2016 [14] used two methods to detect emotions: detecting emotions using facial expression recognition and EEG signals. The study proposed an approach to detect six emotions (joy, surprise, sadness, fear, disgust, and anger) and the neutral state for the EEG section. The discrete wavelet transform (DWT) was used to extract features from the EEG signals. Then,

the SVM with different parameters was used to classify the model. This study achieved 53% accuracy in classifying a correct emotion.

Yunyuan Gao et al. 2020 [15] developed a new method for extracting features by combining Oriented Gradient (HOG) with Granger Causal Method (GC) and Transfer Entropy (TE). The SVM classifier was used to classify emotional states of stress and calm. This study achieved an accuracy of 88.93% and 95.21% for both GC and TE with HOG. Respectively. It is also 12% higher than that obtained without using HOG.

Elham S. Salama et al. 2020 [16] proposed a new multimodal framework for extracting human emotions using the Emotions Database (DEAP). Initially, the research team used three approaches to identify human emotions: the first is the emotion recognition approach with EEG signals. The second is the emotion recognition approach via face data. And the last approach is to recognize emotions based on fusion-based. The 3D-CNN deep learning algorithm was used in the first approach to extract features from the EEG signals. Then Two techniques of fusion have been tried: bagging and stacking. The EEG signals divided the data obtained from the EEG channels into small groups (frames) and presented them to the 3D-CNN model by combining five consecutive frames from 32 EEG channels. Finally, the best accuracy achieved from the proposed work is 96.13% and 96.79% for arousal and valence, respectively.

Al-Nafjan et al. 2017 [17] presented a method for detecting human emotions based on an EEG (two classes) using a deep neural network (DNN) and a power spectrum density (PSD) method based on frontal asymmetry features. This paper claimed that frontal activity is related to emotional states. Because of the decrease in power in the alpha band in the frontal lobe. This study was applied to the physiological DEAP data set. This study achieved an accuracy of 82.0%.

Hector A. Gonzalez et al. 2020 [18] aims to design a test for a convolutional neural network for hand-made devices, called BioCNN, dedicated to detecting emotions via EEG signals. In this study, DEAP and DREAMER datasets were used. In addition, the team recorded a new database of 5 subjects using a low-cost device Emotive EpocC, And an IAPS dataset as visual stimuli. The team used CNN to extract features by providing PSD features and using the conventional frequency spectrum, sample entropy, and asymmetrical Indices. The team tested several machine learning algorithms to detect emotions in the same conditions (k-mean clustering, GMM, Support Vector Machine, Decision Trees, and Random Forests). These tests gave them a reason to design BioCNN. This study achieved accuracy using the algorithm designed (83.12% and 76.78%) for valence and arousal, respectively.

Houtan Jebelli et al. 2019 [19] proposed a model for stress recognition based on EEG signals and updating the model continuously whenever new input signals are updated. Also, this study suggests identifying the stress in actual working conditions and not in controlled laboratories. In this study, two sets of data were used (Construction Workers and DEAP). Initially, the noise was

removed from the data by applying a degree filter to filter the noise of the surrounding electrode wires at 60 Hz. Then in the step of extracting the features, the authors varied the window size between 0.5 (64 data points) and 10 seconds (1280 data points) by 0.5-second step (64 data points). Then two techniques were chosen to extract the features: time-domain and frequency-domain features. As for the first technique, the method for extracting features is the average amplitude value of the EEG signal, standard deviation, peak, maximum cumulative maximum, minimum cumulative, average amplitude value, smallest element. The EEG signals are divided into four bands (Delta, Theta, Alpha, and Beta) for the second technique, and then power is made for each band. Likewise, power is made for frequencies equal to 0 and greater than 64 Hz. After extracting the features, they implemented the Online Multitasking Learning (OMTL) algorithm. Finally, the team obtained an accuracy (71.14% in the DEAP dataset and 77.61% in the Construction Workers dataset).

Aasim Raheel et al. 2020 [20] aims to create a dataset for exploring emotions called DEAR-MULSEMEDIA1 based on physiological signals (EEG, GSR, and PPG) to extract emotions, valence, and arousal. Emotions are identified using more than one human sense simultaneously during interaction with the stimuli obtained. Four videos were used from the multimedia dataset and synced to a fan, heater, etc., on 18 subjects (9 males and 9 females). As for EEG signals, data were collected from the participants using five electrodes; one of them is a reference.

In comparison, the GSR data were collected by two electrodes placed on the index and middle fingers, and finally, PPG is a sensor based on a clip placed on the earlobes of the subject. After collecting data from all electrodes, the research team went through a stage of processing this data to remove noise, where they used a Savitzky-Golay Class 3 filter, with frame length 11, to smooth out the data. Then, the research team moved to the feature extraction stage. In the EEG signals, differential asymmetry (DASM) and rational asymmetry (RASM) were used to extract the features. In contrast, the GSR features were extracted by calculating the maximum, minimum, average, and contrast amplitudes. PPG features were extracted by measuring heart rate as the total heartbeats per minute, while HRV was measured as the interval between heartbeats. HR and HRV were calculated by detecting R peaks in PPG data. After the team got the features, they used the KNN machine learning algorithm and obtained an accuracy of 85, 18% for arousal, and 76, 54% for valence.

In Abdul Rehman Aslam et al. 2020 [21], EEG brain signals were introduced to classify four emotions for children with autism spectrum disorder (ASD) by combining a PS (DNN) deep neural network processor with dual-channel AFE sharing. In this study, two features were calculated: Zero-Crossing Detection (ZCD) and Skewness (SK), to classify arousal and valence using DEAP and SEED datasets. The research team got an accuracy of over 85%.

Habib Ullah et al. 2019 [22] proposes a group learning algorithm for the most distinct subgroup of EEG signals (SDEL), as this algorithm reduces the amount of data while improving efficiency. This study efficiently describes individual channels using Radial Basis Functions (RBF) kernel representations on a DEAP dataset. This study also uses kernel representations of training data to formulate a new method for group learning via a moderated diagram that includes an objective function of linear differentiation analysis. The team used SDEL as a pre-processing method to improve the accuracy of various EEG-based emotion recognition algorithms. To extract the features from the EEG signals, the team used several steps to extract the features, including First, extracting the statistical features (mean, median, maximum, minimum, standard deviation, variance, range, skewness, kurtosis, Petrosian Fractal Dimension, Fisher Information Ratio, and entropy values) Second, frequency domain features: where the team converts EEG data into the frequency domain and then extracted the following basic features: STFT based features, discrete cosine transform features, and spectrogram based features. After extracting the features, the team applied several machine learning algorithms and obtained a binary classification accuracy of 70.1% for arousal and 77.4% for valence.

Xin Xu et al. 2019 [23] proposed a method for feature extraction using double tree complex wavelet transform (DTCWT). The team selected 16 people to stimulate them with visual stimuli (video) using Neuroscan and then used the Support Vector Machine (SVM) algorithm to classify three types of emotions: calm, happiness, and sadness. After collecting data from the participants, the data standardization phase began. The signal for people in the calm state was subtracted from the signal in the form induced by the emotional. The standardized assignment of the interval is performed. Then followed the feature extraction phase, in which complex tree wavelet coefficients were used to obtain different frequency bands using. The F3, FP2, C5, C6, P3, and P4 channels were selected to analyze and standardize the data. After this stage, they got four sub-channels for each significant channel. Finally, the team got a rating accuracy of 90.61%.

Sofien Gannouni et al. 2020 [24] presented a new approach to study the brain lobes that show changes during the extraction of emotions from the DEAP dataset. At first, they adjust the electrodes to the brain lobes. Then they have three stages of classification using the clustering technique. They identified nine emotions: happy, pleased, relaxed, excited, neutral, calm, distressed, miserable, and depressed. In the classification stage, the team created three main steps: The first step was to extract the features of the negative emotions, then extract the features of the positive emotions separately, and then applied the classification algorithm to them. The second step is to extract the features of the active emotions, extract the features of the passive emotions separately, and then apply a classification algorithm to them. Finally, they extract the features of the low feelings, extract the high emotions' features separately, and then use a classification

algorithm to them. After the three-stage classification process, the team obtained an accuracy rate (82.35%, 79.95%, and 71.14%) for (Arousal, Valence, and Dominance), respectively.

Talha Burak Alakus et al. 2019 [25] applied the Sequential Forward Selection algorithm (SFS) using Bayesian Classification (BS) on the GAMEEMO dataset (this dataset is based on EEG signals) to extract features that affect the model's accuracy. After applied SFS, they obtained from 50 to 4-9 for each EEG channel. Then, they used the Quadratic Support Vector Machines (QSVM) algorithm to classify the features according to the arousal-valence model. The research team noticed that reducing the features increased the accuracy of the model 10%, from a 55% accuracy rate of 50 features to a 65% accuracy rate of (4- 8 features). Finally, the researchers' team got (HAPV: 85% from the F5 channel, HANV: 96% from the T8 channel, HAPV: 93% from the P7 channel, and HANV: 100% from the F7 channel).

Mehmet Sirac et al. 2017 [26] used a dataset based on audiovisual stimuli to create a classification to extract negative and positive emotions. After collecting the data, the research team used two feature extraction techniques (Discrete Wavelet transform and Statistical Processes). After extracting the features, the research team used two machine-learning algorithms (Multilayer Neural Network (MLPNN), k-NN) in classification. Finally, they obtained an accuracy of 77.14 and 72.92% by using MLPNN and k-NN, respectively.

Akash Gupta et al. 2019 [27] aim to improve texts based on emotions using EEG signals. Twenty-five people participated in the experiment. Their experience went through three stages. The first stage is detecting feelings using video clips as stimuli and collecting data using an EEG device. The second stage is transcribing texts (improving texts) using a correlation finder between words and detected emotion. The last stage is verifying the validity of the converted sentence using LSTM Networks. The overall score they obtained was 74.95% for ranking five emotional states.

Hemanth , 2020 [28] used EEG signals from the DEAP database to create a classification to discover human emotions using Kohonen networks (Kohonen neural network, Modified Kohonen neural network I, and Modified Kohonen neural network II), where the researcher obtained accuracy (86%, 87%, and 86%) for (Kohonen neural network, Modified Kohonen neural network I, and Modified Kohonen neural network II) respectively.

Dongkoo Shon et al. 2018 [29] proposed a new method to extract features based on EEG signal and DEAP database. The techniques are a genetic algorithm (GA)-based feature selection and k-nearest neighbor (k-NN) classifier and a comparison of this technique with principal component analysis (PCA) method. The research team concluded that the proposed method for discovering features (GA) showed an accuracy of 71.76% classification, in contrast to the second method, which showed an accuracy of 65.3% classification.

1.5. Thesis Layout

The remaining parts of this thesis are divided into four chapters. The second chapter focuses on methods, neural network fundamentals, deep learning, CNN architecture, RNN architecture, applications, and the Python framework with its main tools used in this thesis. The third chapter discussed the data set used in this thesis, illustrating the structure of the proposed modification models (1D CNN, LSTM, and hybrid 1D CNN + LSTM) architectures and training both modifiers' models for detecting human emotions. In the fourth chapter, the results of emotion detection after training and testing are discussed. Also, the model performance is evaluated using some evaluation metrics and compared with other related works performances. Finally, chapter five discusses the main conclusions and future works.

2. BACKGROUND AND GENERAL TECHNIQUES

2.1. Introduction

Due to deep learning techniques, convolutional neural networks and recurrent neural networks have been widely used. In the field of computer vision, deep learning is gaining popularity, in part due to massive data sets in various fields. This section discusses theories of human emotions in the literature and the stimuli used in the literature. Also, it discusses the methods of feature extraction that were used in this thesis. Finally, this section discusses the principles of convolutional and recurrent neural networks. We discussed the main evaluation metrics that are used in this thesis to check the proposed model performance.

2.2. Emotion Theories In the Literature

Theories of emotions in literature are three theories, Plutchik's theory [1], Ekman's theory [30], and James Lange's theory [31]. According to Plutchik's theory of emotions, emotions are classified into two main categories. The first category contains eight basic emotions: anticipation, joy, confidence, sadness, fear, surprise, anger, and disgust. The second category contains secondary emotions that are considered a mixture of the primary emotions in the first category, such as aggression, submission, contempt, dread, etc. [32], as can be seen in Figure 2.1[33]. As for the classification of emotions based on Ekman's theory, called the discrete model theory, this theory presents six basic emotions divided into two groups: positive emotions and negative emotions. These basic emotions are happiness, surprise, sadness, anger, disgust, and fear. Then it increased to 15 emotions [32]. James and Lange's theory classifies emotions in a two-dimensional space called the arousal-valence model. This space will generate four regions of emotions: the first region is the high arousal positive valence (HAPV) zone, the second region is the high arousal negative valence (HANV) zone, the third region is the low arousal negative valence (LANV) zone, and the fourth region is the low arousal positive valence (LAPV) zone, As can be seen in Figure 2.2 [32].

According to arousal-valence two-dimensional model, it is easy to distinguish between negative and positive emotions. In contrast, it is impossible to detect the arousal emotions that belong to the same zone of emotions. For example, anger and afraid are both in the same space of negative valence and high arousal. Here you cannot detect the arousal emotions between anger and afraid [34].

For this reason, Mehrabian [35] updated the two-dimensional model of emotion with a three-dimensional model of emotion, As can be seen in Figure 2.3 [33]. The additional axis that he added is called the axis of domination. After adding this axis, it became easy to discover the difference

between anger and afraid, because anger is in the dominant axis. At the same time, afraid is found in the axis of the submissive axis [33].

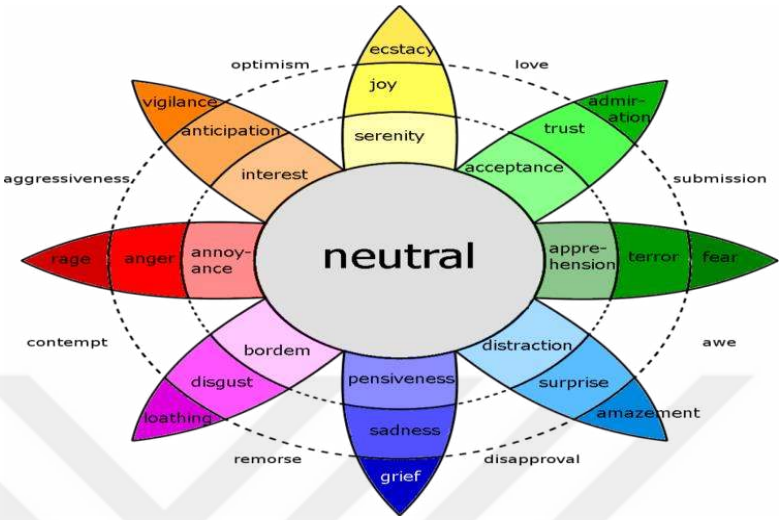

Figure 2.1. Plutchik's theory

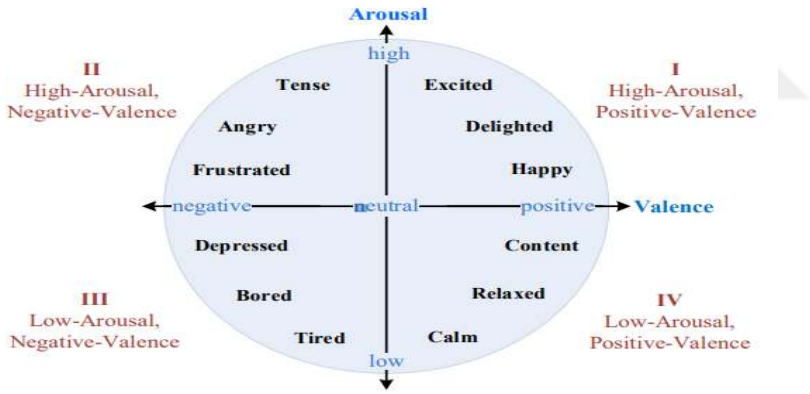

Figure 2.2. James and Lange's theory, arousal-valence model 2-D space.

2.3. Electroencephalogram Device

Electroencephalogram (EEG) technology collects data from people by placing a portable, wearable device on the head in a non-invasive way. This technique is an easy and inexpensive technique for recording brain activity. This device consists of several electrodes placed according to the 10-20 system, as shown in Figure 2.4 [16]. This device records brain signals after the stimuli are applied to the person. Then the data is transferred to a computer for analysis. This technology helped researchers understand people's emotions with disabilities whose emotions are challenging

to discover, suffer from diseases such as depression or epilepsy, etc., through their facial expressions or other methods of detecting emotions [17].

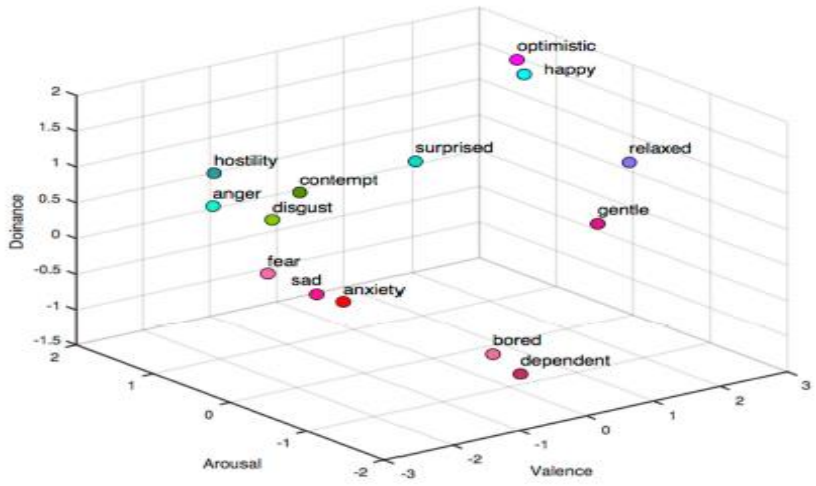

Figure 2.3. Arousal-valence 3-D model

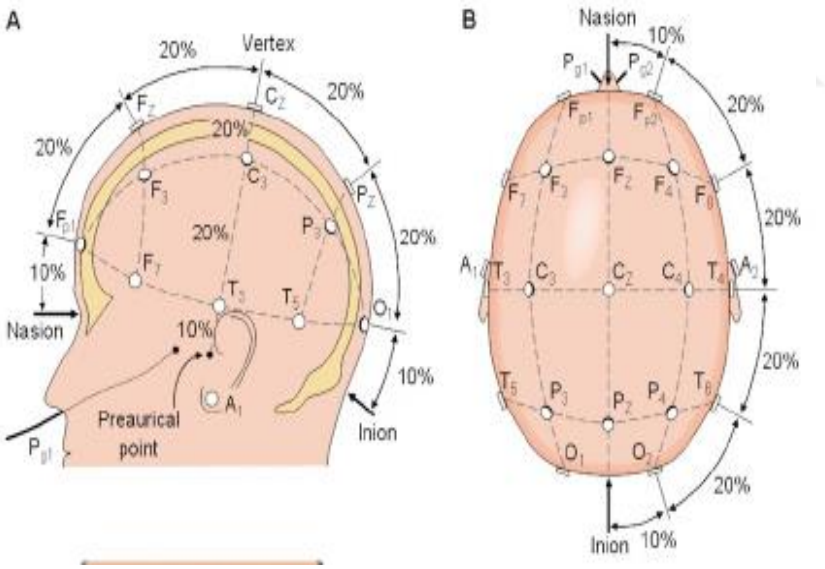

Figure 2.4. The 10-20 system A) front side B) top side

2.4. The Stimuli

Different types of stimuli are used to elicit emotions. There are three common types of stimuli in literature: Aural, visual, and aural-visual stimuli. Aural stimuli such as sad and funny music etc. One of the most important aural stimuli datasets is IADS (International Affective Digitalized Sounds), publicly available to researchers [36]. Visual stimuli are like pictures or

images labeled with different names. One of the most important visual stimuli datasets is IAPS (International Affective Picture System) [37]. As for aural/visual stimuli, which is considered one of the best stimuli for extracting emotions, according to most studies [38], one of the most important aural/visual stimuli datasets is EEGs databases such as DEAP data set.

2.5. Feature Extraction Methods

In this thesis, four methods for classifying emotions based on feature extraction have been proposed. These methods are statistical feature (SF), the combination of welsh power spectral density (PSD) method with a statistical feature, the combination of Fast Fourier transform (FFT) method with a statistical feature, and the combination of wavelet packet decomposition (WPD) method with a statistical feature.

2.5.1. Statistical Features

The mathematical-statistical analysis method is applied to the signal to obtain more information about the signal that cannot be obtained from the mother signal. This method is an easy method to extract features. In this thesis, six statistical features (mean, standard deviation, variance, median, maximum value, and minimum value) were applied to extract six features [39]. Equation 2.1 describes the method for calculating the mean function. Standard deviation calculates the deviation values from the mean value of the EEG samples. Equation 2.2 describes the method for calculating the standard deviation. The middle element of the array refers to the median value, as shown in equation 2.3. Variance is the average of the square deviations. Figure 2.4 shows how to calculate variance values.

$$Avg = \frac{1}{n_e - n_s} \sum_{i=n_s}^{n_e - n_s} X_{iEEG} \tag{2.1}$$

$$Std = \sqrt{\frac{1}{(n_e - n_s) - 1} \sum_{i=n_s}^{n_e - n_s} (X_{iEEG} - \bar{x}_{EEG})} \tag{2.2}$$

$$median(x) = \begin{cases} X\left[\frac{n}{2}\right] & if\ n\ is\ even \\ \frac{(X\left[\frac{n-1}{2}\right] + X\left[\frac{n+1}{2}\right])}{2} & if\ n\ is\ odd \end{cases} \tag{2.3}$$

$$S^2 = \frac{\sum (x_i - \bar{x})^2}{n-1} \tag{2.4}$$

Where n_s: refers to the starting point of the signal while n_e: refers to the end point of the signal, $n_e - n_s$: refers to the total number of samples, the variable i: refers to the values on the horizontal axis, $\bar{x}_{EEG}$: refers to the mean value of the EEG dataset, X: in the median equation refers

to the ordered list of values in data set, n: refers the number of values in the dataset, S^2: refers the samples of variance, x_i: refers to the value of the one observation, $\bar{x}$: refers to the mean value.

2.5.2. Welsh Power Spectral Density

PSD based on the welsh method is a method of splitting the signal into multiple segments to analyze the frequency content of the signals [40]. Spectral analysis is widely used in signal processing to obtain signals of interest [41]. Studies have shown that Welch's method elicits powerful features among the categories of EEG signals [42]. Many applications use spectral analysis, such as radar and sonar signals [43] and spectrum sensing [44]. Further, extract information from relevant data, for example, analyze biomedical signals [45].

The spectral power was calculated in eight EEG bands. The split bands are delta (0.5- 4 Hz), theta (4- 8 Hz), alpha (8- 13 Hz), beta (13- 30 Hz), and four gamma bands (30– 47, 47– 75, 75– 97, and 103– 128 Hz).

2.5.3. Fast Fourier Transform

FFT is an excellent way to extract features from EEG signals, as it performs large-scale arithmetic operations of real numbers from four different frequency bands (alpha, beta, gamma, and theta). To extract these four bans, Butterworth's fourth Class filter was used. Equation 2.5 is the equation that was used to extract the features [46] [47]:

$$X_f = \sum_{n=0}^{N-1} X_n e^{-i2\pi f \frac{n}{N}}, \quad f = 0,1,2,\ldots\ldots,N-1 \tag{2.5}$$

Where N: refers to the total number of EEG samples, X_f: refers to the FFT coefficients, and n: refers to the total number of points in FFT.

2.5.4. Wavelet Packet Decomposition

WPD is one technique that extracts additional features from wavelet decomposition (WD) by deconstructing the signal into different frequency bands to extract subtle features as it can deconstruct signals with different resolutions. The signal is decomposed into two complementary subspaces: V and W, where V represents low-frequency information and W represents high-frequency information. After the signal has wholly decomposed, this will generate a complete wave tree, as can be seen in Figure 2.5 [48] [49].

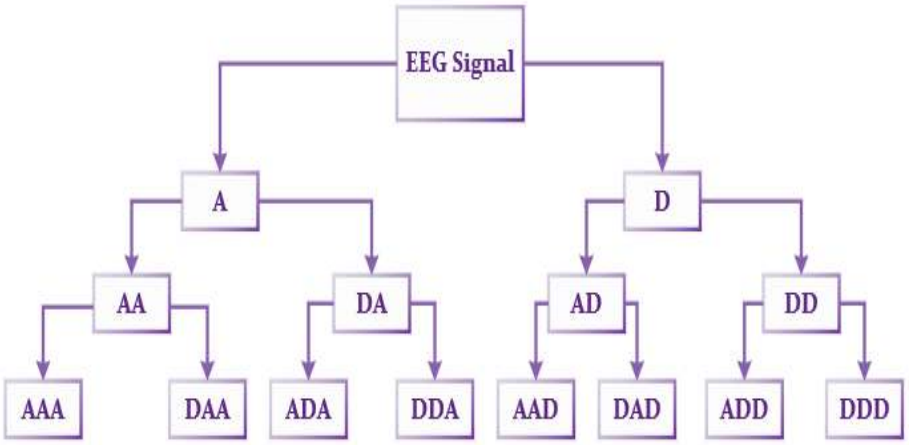

Figure 2.5. The WPD tree (A= Approximation coefficient, and D= detail coefficient).

2.6. Artificial Neural Networks

Artificial neural networks (ANN) are widely used machine learning techniques that simulate the mechanism of learning in biological organisms. The human nervous system consists of many neurons or units connected via axons and dendrites, as shown in Figure 2.6 (a). The connecting regions between axons and dendrites are referred to as synapses. The external motivations determine the strengths of neurons connections which explain how learning happens in living organisms [50][51]. In ANN, the connections between units have weights that work similarly to biological organisms, as shown in Figure 2.6 (b). The input values of a single unit are scaled with their corresponding weight. ANN computes a function of the inputs by propagating the computed values from the input neurons to the output neuron(s) and using the weights as intermediate parameters. The learning process occurs by changing the weights connecting the neurons [52].

2.6.1. ANN Architecture

The basic element in ANN is Perceptron, which is a single neuron that receives weighted connection inputs and produces a single output. ANN consist of a hierarchy of single layer or multilayers architecture. Each layer has a set of neurons or units connected with the units of the other layers. The network is fed throughout its input layer, and the prediction is performed at the output layer. The intermediate layers are called “hidden layers”. Based on the way the information flow throughout the network, the neural networks are classified into two main categories:

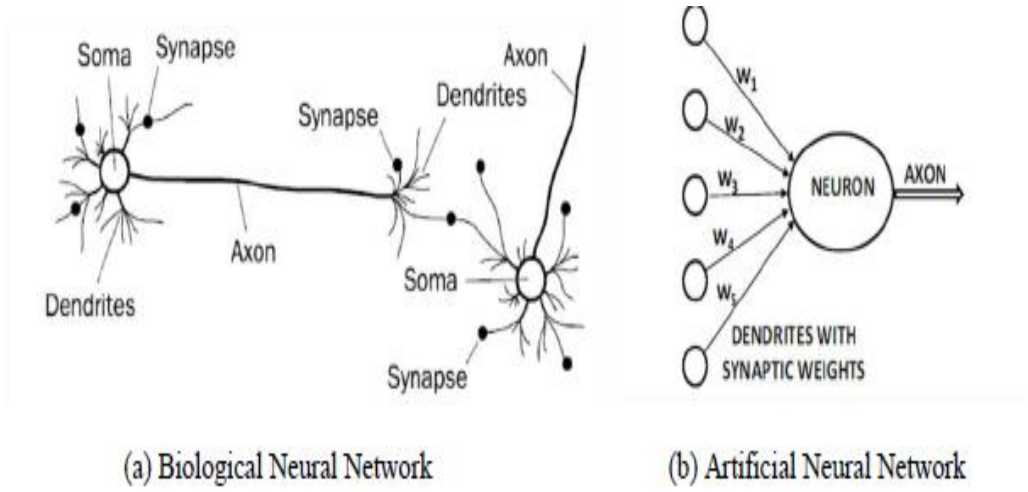

Figure 2.6. The Synaptic Connections Between Neurons [53].

Feed-Forward Networks (FFN) and Feed-Back Networks. In the first category FFN, the information is transferred only in one direction from the input layer toward the output layer passing throughout the hidden layers. Therefore, it is called the Directed Acyclic Graph (DAG). Multi-Layer Perceptron (MLP) is an example for feed-forward networks, as shown in Figure 2.7 [54]. The second category of ANN has connections that form directed cycles (or loops). This architecture allows them to generate sequences of arbitrary sizes. The most important feature in this network is memorization ability and storing information and sequence relationships in their internal memory. Examples of such architectures include Recurrent Neural Network (RNN) and Long-Short Term Memory (LSTM), as shown in Figure 2.8 [54].

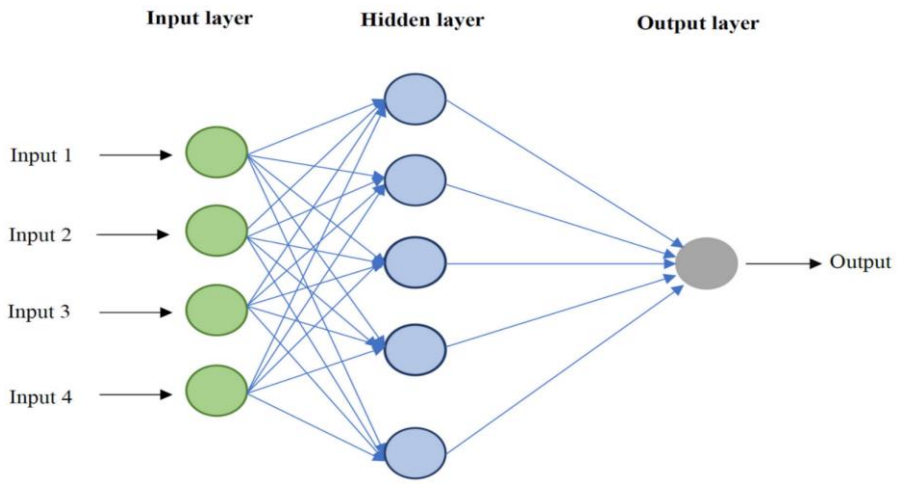

Figure 2.7. Feed-Forward MLP Neural Network [55].

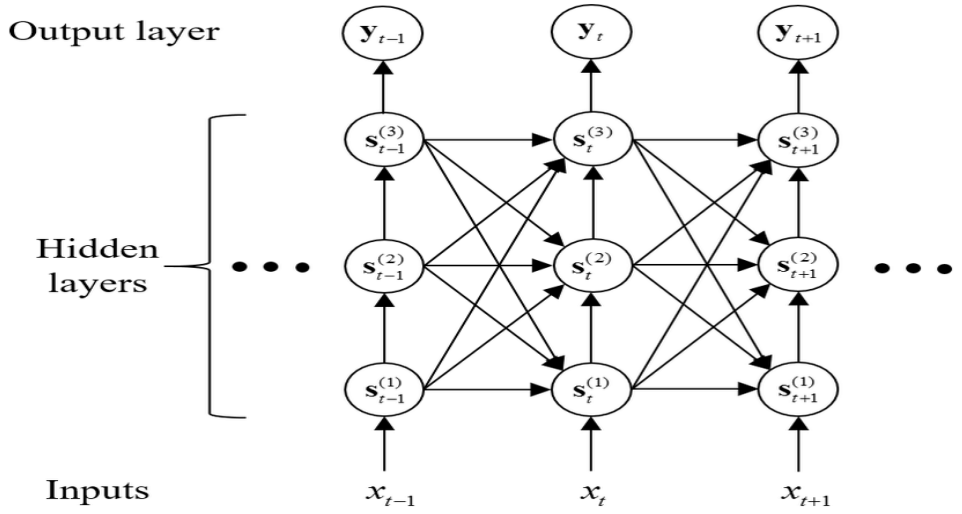

Figure 2.8. Feed-Back RNN architecture [56].

2.6.2. ANN Components

The main three components of the ANN are weight, bias, and activation function. A weight component is a number that refers to the strength of the connection between two units in the network. Hence, increasing the weight value leads to an increase in the connection strength and vice versa. The adjustment of weight values can be utilized to perform a particular task. For example, suppose a unit has n inputs; we have n weight because each input has its connection strength with the unit. The weighted summation of input signals will give the total amount of signal received by the unit. Depending on the weight w_n, the signal will be either amplified or attenuated compared to the other inputs. If the weight of the particular input is larger than others, it will have more impact on the unit's output. If we denote the input as x_n And the output as y, then the mathematical representation of what we have described becomes, as shown in equation 2.6 [57]. The bias parameter is responsible for setting the threshold value of the unit. The threshold is important for regulating the firing conditions of the unit and can be expressed as a number added to the weighted summation of unit inputs. Equation 2.7 explains how to calculate the bias parameter. Biases enable shifting the unit output, thus controlling the amount of input to activate the neuron [58]. The main rule for the activation function component is to enable the unit to decide whether it should fire after processing the given parameters. Depending on the strength of electrochemical reactions in the biological brain, neurons transmit signals with different strengths. Equation 2.8 explains how to activate the function parameter.

$$y = \sum_n (w_n . x_n) \tag{2.6}$$

$$y = \sum_n (w_n . x_n) + b \tag{2.7}$$

$$y = \begin{cases} 1, if \ \sum_n (w_n . x_n) + b > 0 \\ 0, if \ \sum_n (w_n . x_n) + b < 0 \end{cases} \quad (2.8)$$

In equation 2.8, if y = 1, the unit will pass the signal further, otherwise not. The rule of bias is evident in shifting the activation level: if we increase the bias, it takes less strength of the input signals to fire the neuron. Despite having simplicity as its advantage, this activation function is quite unpopular due to its inability to give continuous outputs. Sometimes, it is necessary to test the reliability of the activation function and the activation state [58].

2.7. Deep Learning

Deep learning is a neural network with multiple layers of nonlinear processing units. Each successive layer takes the output from the previous layer as input. The network can extract the complex hierarchy features from a large amount of data by using these layers. Deep learning can represent functions with higher complexity if the number of layers and units in a single layer increases. Given enough labeled training datasets and suitable models, deep learning approaches can help humans establish mapping functions for operation convenience [59]. Also, Deep learning is a field of machine learning that teaches computers to do what comes naturally to humans: learn from experience. Machine learning algorithms use computational methods to "learn" information directly from data without relying on a predetermined equation model [57]. Deep learning is especially suited for image recognition, which is important for solving problems such as facial recognition, motion detection, and many advanced driver assistance technologies such as autonomous driving, lane detection, pedestrian detection, and autonomous parking [60]. Deep learning includes four main architectures. They are Restricted Boltzmann Machines (RBMs), Deep Belief Networks (DBNs), Autoencoder (AE), and Convolutional. This thesis focused on using the convolutional and Recurrent neural networks to develop a detection model for extracting human emotion using EEG signals.

2.7.1. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are one of the most popular categories of deep learning. First designed by Yann LeCun, this network was first implemented to recognize handwritten numbers. Convolutional Neural Networks (CNNs) can extract features from data and classify them into a single structural model, unlike traditional Artificial Neural Networks (ANNs) [61]. In recent years, in computer vision, CNN has also played a good performance, mainly due to its application to the image data of the particular network structure [62]. It is usually used with high-dimensionality data such as images and videos. The mechanism of CNN working is similar to that in ANN. The main difference is that CNN uses two-dimensional filters or weights that are

convolved over the image data. CNN can be applied for both supervised and unsupervised model learning. Figure 2.9 explains the general structure of CNN. Although CNN's was initially designed to recognize objects in two-dimensional signals such as images and video frames, they have also been widely used in 1-D signal processing. The use of CNNs for 1-D signal processing requires 1D to 2D conversion. [61].

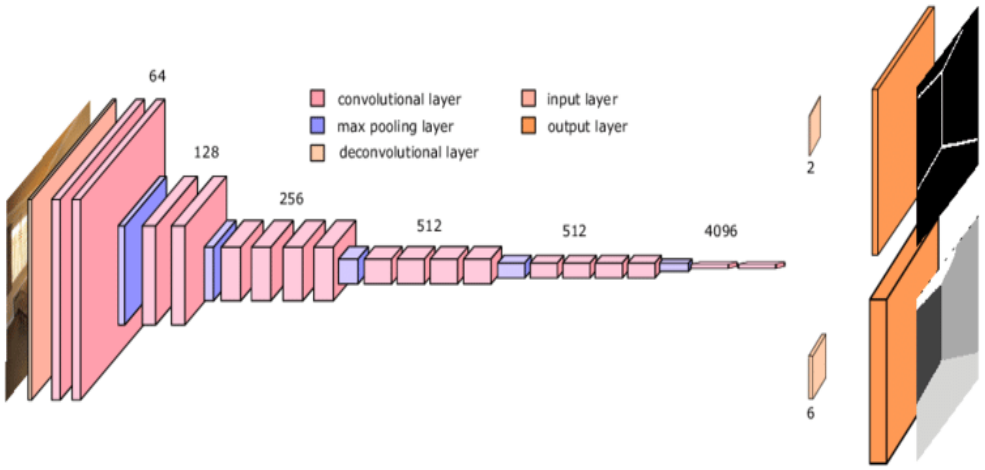

Figure 2.9. CNN Layers [63].

The most important components in the CNNs are the convolutional layers, Pooling Layers, Dropout layers, and Fully Connected Layers. The convolutional layers contain a group of filters or kernels that are convolved with the input data to extract the feature map. These filters are a group of numbers (weights) that are organized in two-dimensional (in case of single-channel) or three-dimensional (in case of three channels) matrix. The learning operation includes many iterations. The weights are initialized randomly and updated continuously according to the expected output. The filters convolve with the convolution layer input, as shown in Figures 2.10 (a) and 2.10 (b). These figures illustrate the convolution processes between a 5x5 input feature map and a 3x3 filter to generate a 3x3 output feature map. The filter is multiplied with a highlighted patch (also 3x3) of the input feature map and sums up all values to generate one value in the output feature map. Note that the filter slides along the input feature map's width and height, and this process continues until the filter can no longer slide further. The pooling operation is the process of sliding a two-dimensional filter over each channel of the feature map and summarizing the features lying within the region covered by the filter. Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network. The pooling layer summarizes the features present in a region of the feature map generated by a convolution layer. So, further operations are performed on summarized features

instead of precisely positioned features generated by the convolution layer. This makes the model more robust to variations in the position of the features in the input image. Many pooling functions reduce the input feature maps such as max, min, and average. Max-pooling is the widely used function, as shown in Figure 2.11. Max pooling is a pooling operation that selects the maximum element or the average element from the region of the feature map covered by the filter. Thus, after the max-pooling layer, the output would be a feature map containing the most prominent features of the previous feature map [64], [54].

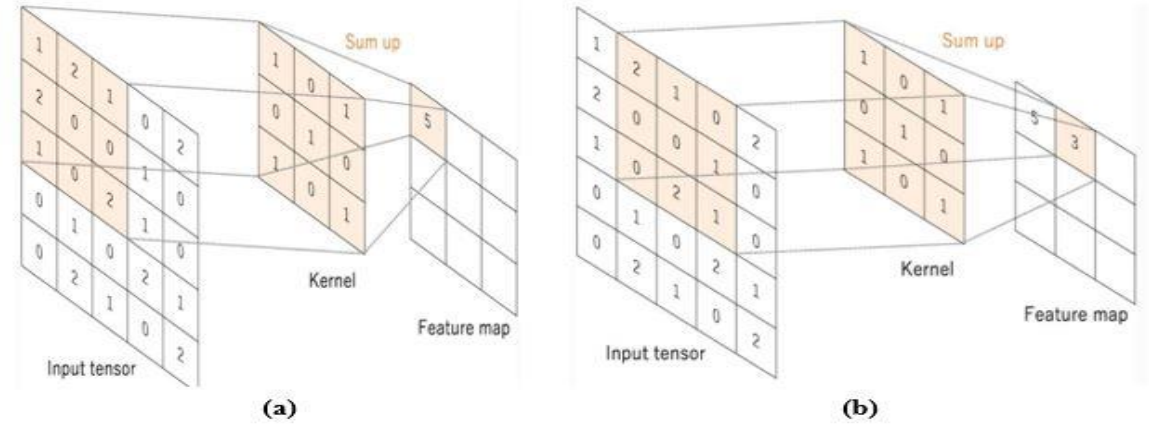

Figure 2.10. Converting the input data into a feature Map [65].

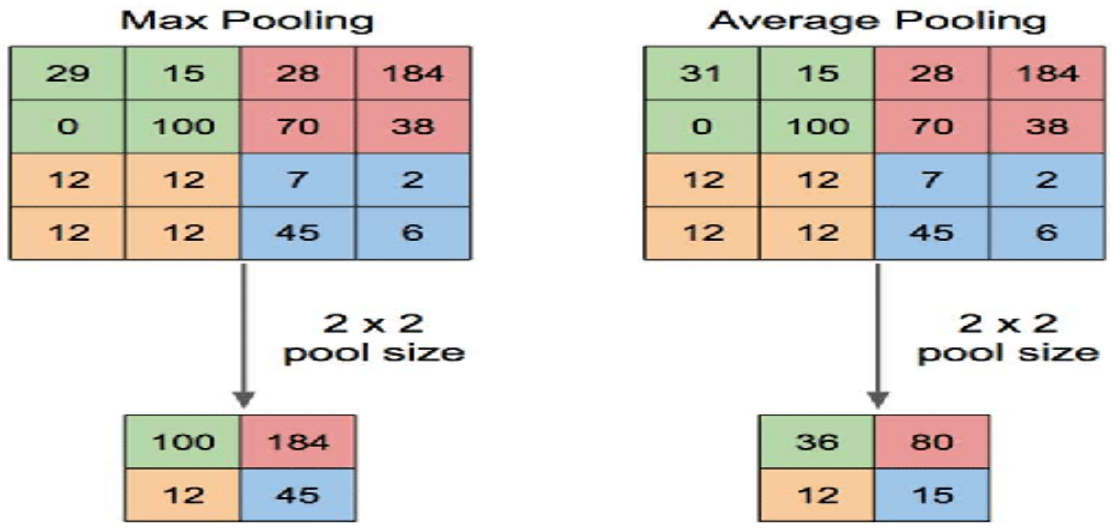

Figure 2.11. Max Pooling Operation [64].

Fully connected layers can be considered convolution layers that have weights with 1x1dimensions. The units of these layers are connected with all the units of the previous layer. The

operation of these layers is similar to that in ANN that includes multiplication of the feature maps inputs with the corresponding weights and adding the bias vector, then applying the nonlinear activation function. Fully connected layers are widely used in image classification and pattern recognition [66].

2.7.2. 1D Convolution Neural Network

1D CNNs are used in a lot of signal processing applications, especially EEG signals. Where the signal is converted to spectrograms and analysis [67]. 1D CNNs have become a popular signal processing method because these networks can take advantage of the signal's fine time structure. For example, Kim et al. [68] proposed a 1D CNN architecture to tag musical signals automatically. The basic architecture of 1D CNNs is somewhat similar to a regular neural network. While it differs from a neural network in that it receives raw data as input rather than handcrafted features. Then it analyzes the data through several trainable convolutional layers to extract features. Finally, these networks classify the features in the same architecture [69]. Figure 2.12 shows the general form of the 1D CNN architecture.

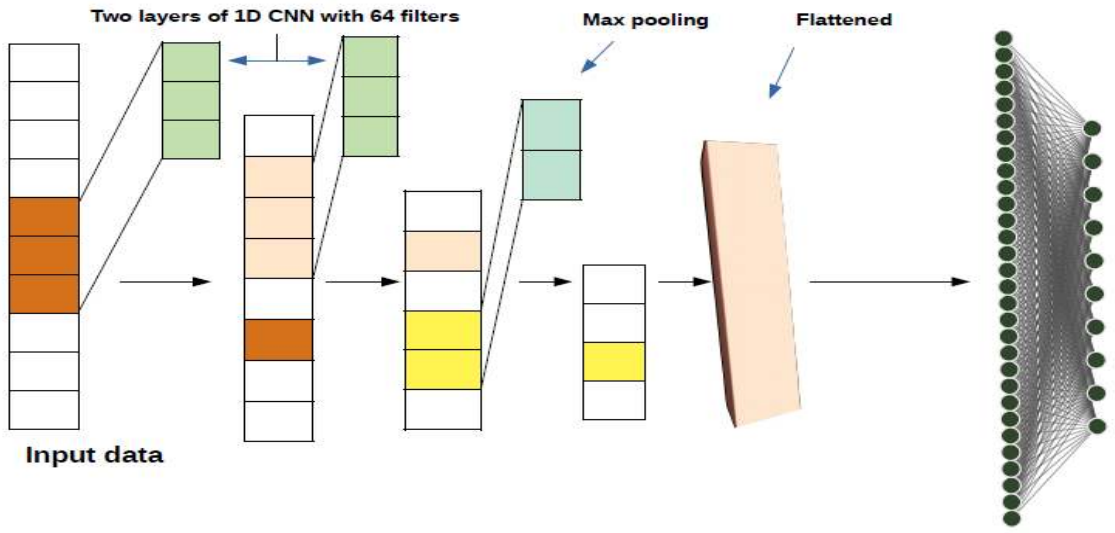

Figure 2.12. An example of the architecture of the 1D CNN [70].

2.7.3. Recurrent Neural Networks (RNN) with Long-Short Term Memory (LSTM)

A recurrent neural network (RNN) is a neural loop network that has internal memory. The RNN performs the same function repeatedly for all input data, and the output of the current input depends on the previous operation [71]. This means this network deals with sequence data such as text and climate data. In the texts, the sentence consists of sequential words linked together, and in

the climate, there is a relationship between today's temperature and yesterday's temperature [72]. An RNN mainly consists of multiple neurons (nodes). Each node has a time-varying value. Also, each connection has an actual weight that can be modifiable in any situation. The neuron output at time t-1 will be passed to the input at time t and add the data of itself at time t to generate the output at time t. The formulas of RNN are shown in equations 2.9 and 2.10 [72].

$$h_t = \tanh\left(w_h h_{t-1} + w_x x_t\right) \tag{2.9}$$

$$y_t = w_y h_t \tag{2.10}$$

Where h_t: refers to the hidden layer vector, x_t: refers to the input vector, y_t: refers to the output vector, and w_h: refers to the weighting matrix.

After the output is obtained, it is sent back to the recurrent network to make the appropriate decision. An RNN takes into account the current inputs, and the outputs learned from the previous inputs. This redundancy can make the network somewhat opaque because RNN tends to be over-processed and requires good organization. But it is no different from a standard neural network. Each copy transmitting a message to the next copy can be considered multiple copies of the same network. Figure 2.13 represents the loop process in the RNNs. In this figure, Part Net looks at the input (x_i), and then gives the output (o_i). Some of the applications of RNN are: language modeling [73], speech recognition [74], machine translation [75], etc. [76].

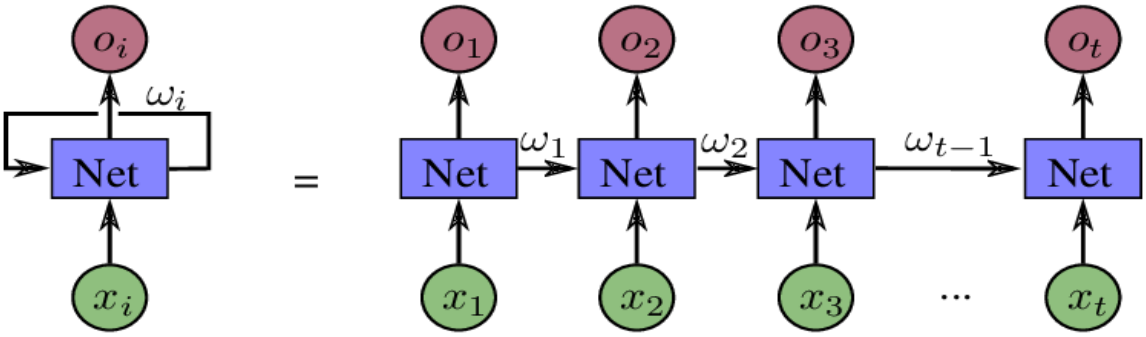

Figure 2.13. A simple RNN architecture [77].

Long-Short term memory (LSTM) was first proposed to improve RNNs by Hochreiter and Schmidhuber (1997) [77]. An LSTM network is based on control gates that control the flowing

data and prevent problems that unnecessary data may cause. Later, the LSTM network was updated by Gers et al. (2000) [78]. Forgetting gates have been added to forget useless memories from the memory cell. The number of gates in the LSTM network becomes three: input gate, forget gate, and output gate. Figure 2.14 shows the three gates of a single cell structure of the LSTM network. Also, Figure 2.15 shows the recurrent process in the LSTM network. The LSTM network differs from the RNN in learning skills by using a complex gate approach. The input gates learn the importance of the input information. The network gate learns the importance of the volume of relevant information, in addition to the forgotten gate. A study was presented in 2012 [78], in which LSTM was used for a modeling task using English and French. This study concluded that LSTM outperformed RNN by 8%. Figure 2.16 shows the architecture of each gate. Also, Figure 2.17 shows the architecture of the LSTM model for human emotion recognition.

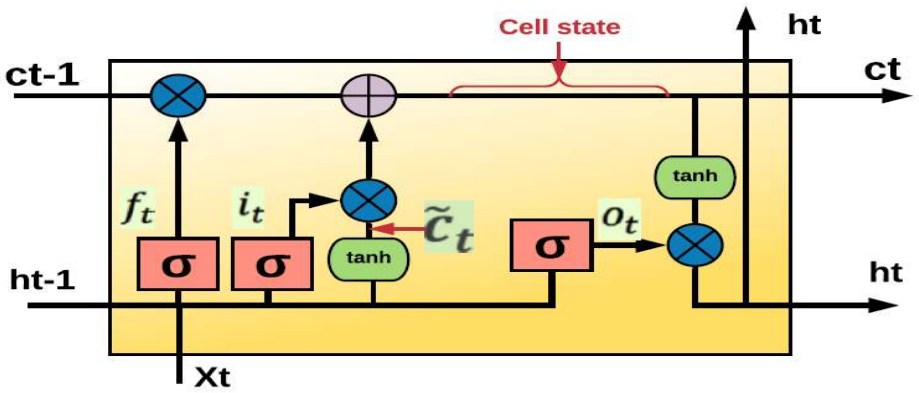

Figure 2.14. Single-cell of LSTM network [70].

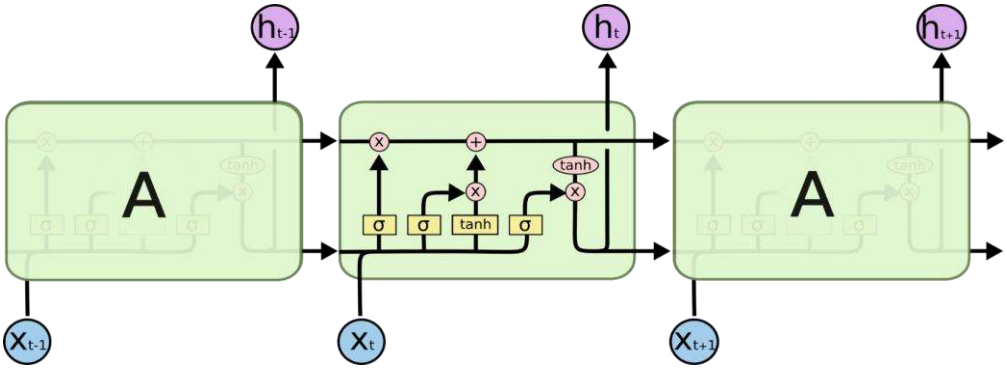

Figure 2.15. The general architecture of the LSTM network [79].

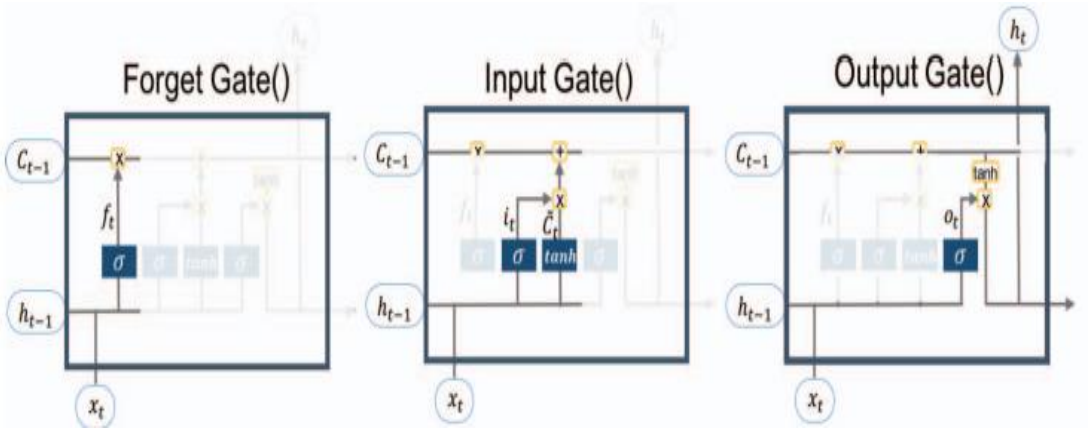

Figure 2.16. LSTM's gates [72].

As shown in Figure 2.16, the forget gate is defined using Equation 2.11. by using this equation, unnecessary information is discarded using entering the output (h_{t-1}) at the previous unit ($t-1$), and adding the current time (t) input (x_t) into the sigmoid function $S_{(t)}$. As can be seen in equation 2.12. Where values between 0 and 1 are generated, as can be seen in Figure 2.18. A value of zero indicates that this information is forgotten, and one indicates that this information is not forgotten [72].

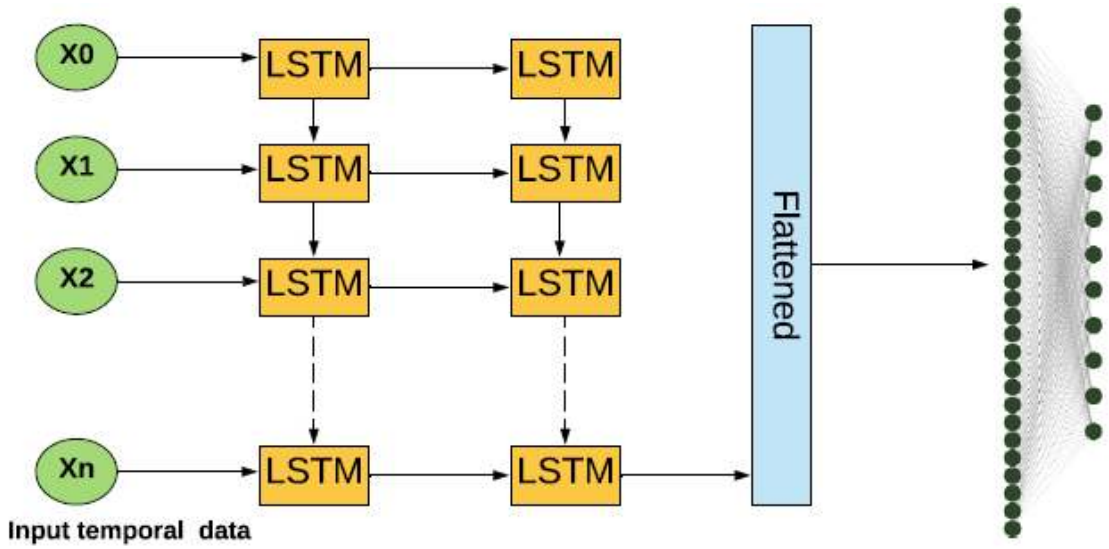

Figure 2.17. The architecture of the LSTM model for human emotion recognition [70].

$$f_t = \sigma(w_f \,.\, [h_{t-1}, x_t] + b_f \tag{2.11}$$

$$S_{(t)} = \frac{1}{1+e^{-t}} \tag{2.12}$$

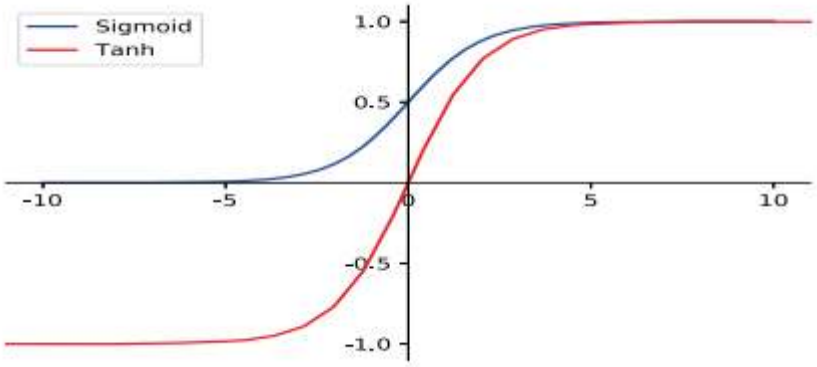

Figure 2.18. The sigmoid and tanh function [72].

As for the input gate, it defines the new information that must be remembered in the cell. In the input gate, equation 2.13 is used to calculate the value of (i_t) to determine the amount of state of the new information cell to remember. The (i_t) value is between 0 and 1. At the same time, the tanh layer gets an election message ($\tilde{c}_t$) using equation 2.14. Then the values of the i_t are multiplied by the values of $\tilde{c}_t$: that were obtained using the Sigmoid function to get updated information, as can be seen in equation 2.15 [72].

$$i_t = \sigma(w_i \,.\, [h_{t-1}, x_t] + b_i \tag{2.13}$$

$$\tilde{c}_t = tanh(w_c \,.\, [h_{t-1}, x_t] + b_c \tag{2.14}$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \tag{2.15}$$

As for the output gate, it determines the information that will be output from the cell. Whereas, Equation 2.16 is used to generate values between 0 and 1 to determine the number of cells of information that need to be output (o_t). Then the result we got in equation 2.17 is multiplied by a value between -1 and 1 using the tanh function. As can be seen in equation 2.17. For the LSTM block at time t, we will get the output (h_t).

$$o_t = \sigma(w_o \,.\, [h_{t-1}, x_t] + b_o \tag{2.16}$$

$$h_t = o_t * tanh\,(c_t) \tag{2.17}$$

2.8. Evaluation Metrics of the Semantic Segmentation

The main objective of the emotion classification based on EEG signals is assigning a class label for each type of emotion in the target. The evaluation process of emotion recognition is performed using global and class-wise metrics such as confusion matrix, sensitivity, specificity, the

area under the carve (AUC), precision, recall, and f1-score. The global metrics measure the overall accuracy without respecting the classic's accuracy individually. These metrics are accurate with applications that have convergent numbers of class labels. While in other metrics, each is usually used with the application that has class imbalance [67]. The following metrics are used to evaluate the performance of the emotion detection based on EEG signals in this thesis:

2.8.1. Confusion Matrix

It is a tool that measures the performance evaluation of the classification model. It contains information about the number of actual and predicted labels. The performance of such systems is commonly evaluated using the data in the matrix. Table 2.1 shows the confusion matrix for a two-class model [80]. The entries in the confusion matrix have the following meaning in the context of this thesis:

- **TP** is the number of **correct predictions** that an instance is positive,
- **FP** is the number of incorrect predictions that an instance positive,
- **FN** is the number of incorrect predictions that an instance is negative, and
- **TN** is the number of **correct predictions** that an instance is negative.

Table 2.1. The Confusion Matrix for Two Class [81].

		Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

2.8.2. Global Accuracy, Sensitivity, and Specificity

Global accuracy is defined as the ratio between the correctly predicted values for all classes to the total number of values, as shown in equation 2.18. sensitivity is the ratio of true positives that are correctly identified over all positive assessments, as shown in equation 2.19 [82]. Specificity is used to calculate the proportion of the true negatives correctly identified over all negative evaluations, as can be seen in equation 2.20 [82].

$$Global\ Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.18}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{2.19}$$

$$Specificity = \frac{TN}{TN + FP} \tag{2.20}$$

Global Accuracy is not effective with a class-imbalanced dataset with a significant disparity between positive and negative labels. The accuracy is calculated for each class separately to overcome this issue, and then the mean accuracy is produced [83]. A sensitivity test is used to identify positive emotions correctly, and This is called the ratio of true positives that are correctly determined by the test. In contrast, the specificity test is used to identify people with negative emotions correctly. It is called the proportion of true negatives, which is correctly determined by the test [84].

2.8.3. Precision, Recall, and F1-Score

Precision is the proportion of correctly classified positive cases from the total number of actual positive cases. Equation 2.21shows how to calculate the precision. Recall (Like sensitivity test) is the ratio of positive cases that were correctly classified to all observations, as shown in equation 2.22. The F1 score is the average of both precision and recall tests. Where false positives and false negatives were taken into account, this test is helpful if the distribution of a class is uneven. While, if the false positives and false negatives have the exact cost, the accuracy test will work best. When a balance needs to seek a balance between precision and recall, F-score will be used, as in Equation 2.23.

$$Precision = \frac{TP}{TP + FP} \tag{2.21}$$

$$recall = \frac{TP}{TP + FN} \tag{2.22}$$

$$F1 - Score = 2 * (\frac{precision * recall}{precision + recall}) \tag{2.23}$$

2.9. Python Software Environment

In this thesis, the anaconda version (4.9.2), Python version (3.8.5), and Jupyter notebook version (6.1.4) software package are used to design the 1D CNN, LSTM, and 1D CNN + LSTM architectures. However, the following toolboxes have been used to prepare the main program and supported function codes.

2.9.1. Deep Learning Toolbox

There are many frameworks for deep learning in the Python environment. These libraries allow researchers and programmers to write code easily. The most popular of these platforms are TensorFlow and Keras [85].

TensorFlow Platform (TF):

The TensorFlow framework is one of the most widely used open-source platforms for deep learning [86]. The TensorFlow platform was developed by Google in November 2015 and supported deployment across CPUs and GPUs. One of the advantages of the TensorFlow platform is that it does not require a graphics processing unit (GPU). It will try to use it automatically. But if there is more than one GPU, you must specify the operations that each GPU will execute by writing the following code [85]:

```
with TensorFlow.device("/gpu:1"):
# Model definition here
```

To install TensorFlow on anaconda, open the Anaconda prompt. And then, type the following command: pip install TensorFlow. The last version of the TensorFlow used is 2.0. You can read more about TensorFlow and its recent developments here: (https://www.tensorflow.org/).

Keras Platform:

A library within the Python environment consists of a high-level neural network written in Python and using TensorFlow in the backend. Keras is relatively more straightforward than TF because it enables you to run quick experiments; for example, it can execute the entire code with less than 15 lines. Also, writing Keras in Python enables it to become a particular framework because the Python community has a larger community of users and supporters. It is very easy to get started with Keras. To install Keras on Anaconda, open the Anaconda prompt. And then, type the following command: pip install Keras. You can read more about Keras and its recent developments here: (https://keras.io/)

2.9.2. Graphic Processing Units (GPUs)

GPUs were created for rendering graphics. However, GPUs have become popular for general-purpose high-performance computation in the last ten years, including medical image processing. GPUs are most likely due to the increased programmability of these devices, combined with low cost and high performance[87].

Recently, the use of GPUs became popular for general-purpose computations that require a highly data-parallel architecture. GPUs architecture consists of many processing cores, each of which has many functional units, such as arithmetic logic units (ALUs). Each available unit includes a set of thread processors under the supervision of a shared control unit. The control unit stimulates the thread processors to implement the same instruction on each pixel of the image in parallel [88].

The rapid advancement in GPU manufacturing with various architectures led NVIDIA to declare its parallel computation platform known as computing unified device architecture (CUDA) in 2006. CUDA is a parallel programming platform, and its instruction set architecture uses a parallel compute engine from NVIDIA GPU to solve large computational problems. CUDA is an open-source and extension of the C programming language. CUDA program contains two phases executed in either host (CPU) or device (GPU). There is no data parallelism in the host code. The phases that exhibit a rich amount of data parallelism are implemented in the device code. A CUDA program uses a NVIDIA C compiler (NVCC) that separates the two phases during the compilation process. The host code is ANSI C code, and the device code is ANSI C code with extended keywords. The windows user can compile the CUDA programming with Microsoft visual studio 2008 onwards using NVIDIA Nsight. Eclipse IDE support for Linux and Mac platforms. Some other languages or programming interfaces to support parallel computing are OpenCL (Open Computing Library), DirectX Compute, and FORTRAN [89].

2.9.3. Collaboratory (Colab):

Collaboratory, Colab, or Google Colaboratory is a cloud service for writing code in Python and execute it on your computer for free. It also allows free access to GPUs and TPU as well. Colab is based on Jupyter notebook technology, an open-source tool that can run locally or on the cloud. It contains multiple cells fully configured for deep learning and free access to a powerful GPU. Colab has basic machine learning and artificial intelligence libraries, such as TensorFlow, Matplotlib, and Keras. Colab also offers to save Jupiter notebook files on the user's Google Drive account. Finally, the Google Colaboratory infrastructure is hosted on the Google Cloud platform [90].

3. MATERIAL AND METHODS

3.1. Introduction

Emotion detection based on EEG signals via 1D CNN, LSTM, and 1D CNN + LSTM architectures require several steps regarding the training data set selection framework and network architecture design. In this chapter, the data set used to train and test the models is focused on. Next, this study focused on the feature extraction methods, model steps, and details of the training process. Figure 3.1 shows the flowchart of this thesis study. As can be seen in the flowchart figure, there are five steps in this thesis: data set, feature extraction, feature normalization, building models, and training and testing.

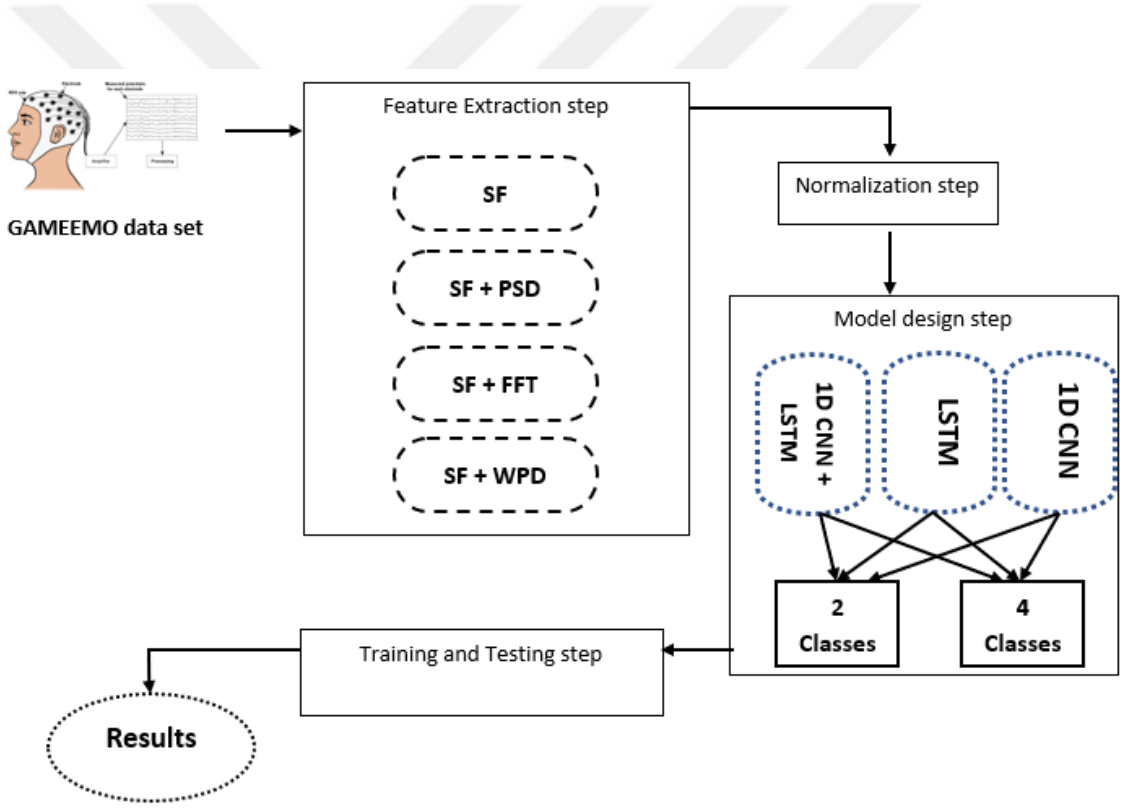

Figure 3.1. The flowchart of this thesis

3.2. GAMEEMO Dataset

It is a physiological database containing EEG signals of 28 subjects using a portable EEG device. Participants are between 20 and 27 years old, without specifying their gender. Four computer games were used as stimuli to arouse the emotions of the participants. For each of the 14 channels, the length of the EEG signal is 38252, and the sampling rate for signals is 128 Hz. Fourteen channels were used to record brain activity: AF3, AF4, F3, F4, F7, F8, FC5, FC6, O1,

O2, P7, P8, T7, and T8. The four stimuli were selected based on the discrete emotions model and the dimensional emotions model. All four games were chosen to be compatible with both models. They gave a symbol for each game: G1 for the first game, G2 for the second game, G3 for the third game, and G4 for the fourth game. In the structure of the discrete emotion model, G1 and G3 express negative emotion, while G2 and G4 express positive feelings. In the structure of the dimensional model, G1 refers to the LANV region, G2 refers to the LAPV region, G3 refers to the HANV region, and G4 refers to the HAPV region. Each participant played each game for 5 minutes. In total, 20 min EEG data were obtained for each subject. Table 3.1 shows the four stimuli used in the dataset and their compatibility with both models of emotions. This dataset contains raw EEG data and pre-processed EEG data. Also, it is available to the public free of charge via the following link (https://data.mendeley.com/datasets/b3pn4kwpmn/3). The main article of the GAMEEMO dataset can be read in Study [5]. In this study, pre-processed EEG data were selected to detect emotions. Figure 3.2 shows the signal amplitude according to the four games and channel F3.

Table 3.1. The stimuli used in the GAMEEMO database

Game symbol	Stimuli type	Discrete models (Positive and negative)	Dimensional model (Arousal-valence)
G1	Boring motion	Negative emotion	LANV region
G2	Calm emotion	Positive emotion	LAPV region
G3	Horror motion	Negative emotion	HANV region
G4	Funny emotion	Positive emotion	HAPV region

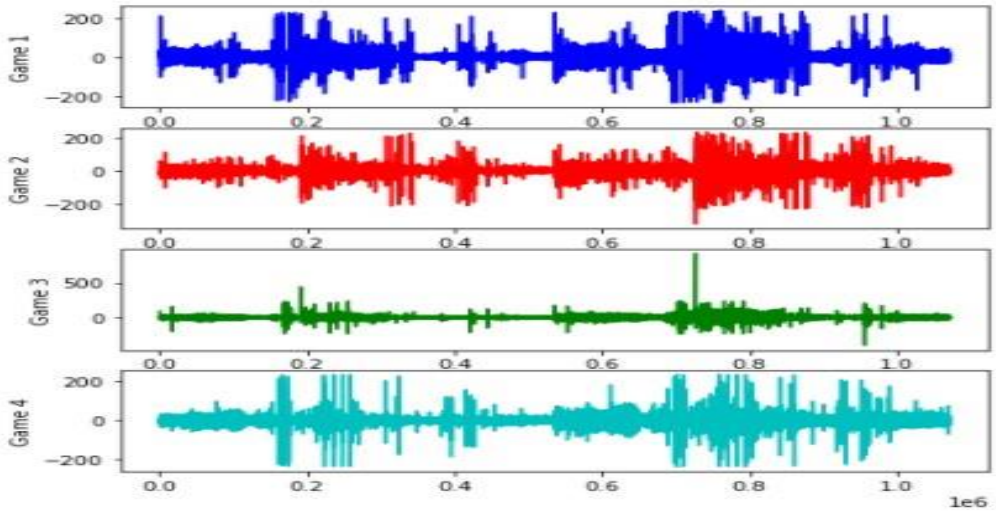

Figure 3.2. An example of EEG signals using the F3 channel.

3.3. Feature Extraction Methods

In this section, four methods of extracting features from the EEG are selected: statistical features (SF), a combination of SF, and power spectral density (PSD) using the Welsh method. Combination of SF and Fast Fourier transform (FFT) and SF and wavelet packet decomposition. We have discussed the details of the four methods in chapter two. The first method used the statistical method to extract four features: mean, standard deviation, minimum value, and maximum value. Ready functions from the NumPy library have been implemented to extract the six features. A total of 6 * 38252 * 28 * 4 features were obtained. The number 6 indicates the number of statistical features extracted. The number 38252 indicates the length of the signal. The number 28 indicates the number of subjects, and the number 4 indicates the number of computer games. If we multiply the last three numbers, we get 4284224 samples. This number refers to the rows in the input matrix, either the columns or the features we get from the six statistical features we extracted. The code that was used in this study is shown in Figure 3.3.

```
# feature extraction using statistical method

x_std=np.array(np.std(dfNor, axis=1))
x_avg=np.array(np.average(dfNor, axis=1))
x_min=np.array(np.min(dfNor, axis=1))
x_max=np.array(np.max(dfNor, axis=1))
x_var=np.array(np.var(dfNor, axis=1))
x_nanmead=np.array(np.median(dfNor, axis=1))

dfStat=np.append(x_std,x_avg ,axis=0)
dfStat=np.append(dfStat,x_min ,axis=0)
dfStat=np.append(dfStat,x_max ,axis=0)
dfStat=np.append(dfStat,x_var ,axis=0)
dfStat=np.append(dfStat,x_nanmead ,axis=0)


dfStat=np.reshape(dfStat, (4284224, 6))
dd=pd.DataFrame(dfStat,columns=['std', 'Avg', 'Min','Max','Variance','Meadian'])
```

Figure 3.3. The code of statistical feature method used in this thesis

The second method of feature extraction (SF + PSD) is divided into two parts: in the first section, the features were extracted by the statistical feature method, and then it was combined with the welch method. The welch method extracts the power spectral density (PSD) from the input signal. If the input signal is a vector, the welch method treats it as a single channel. But if the input signal is an array, the welch method treats each column of the array independently. The input signal

is split into the longest possible segments using the welch method to obtain approximately eight segments with an overlap of not more than 50% [91]. The total of the features obtained using the combination of SF + PSD is (8 + 6) * 38252 * 28 * 4, where the combination 8 + 6 refers to the features obtained using PSD and SF method, respectively. Figure 3.4 shows the code used to extract features using the PSD method.

```
from scipy import signal
# Define sampling frequency and time vector
sf=50
# Define window length (4 seconds)
win = 4 * sf
freqs, psd = signal.welch(dataFG, sf, nperseg=win)
print (psd.shape)
```

Figure 3.4. The PSD method is used to extract features from the EEG signals.

Fast Fourier Transform (FFT) is one of the most commonly used methods of signal processing. The FFT provides frequency information about the signal by converting the signal into individual spectral components. The FFT outputs are real and imaginary parts of positive and negative frequencies [92]. Using SF + FFT has been Extracting (6 + 14) * 28252 * 28 * 4. The number 14 refers to the features extracted from 14 EEG channels consisting of real and imaginary frequencies. The imaginary part was removed, and only the real part was left. Figure 3.5 shows the code used to extract features via the FFT method. And Figure 3.6 shows the signals obtained after applied the FFT with the SF method.

```
from scipy.fft import fft, fftfreq

# applay Fast Fourier Transform method
df_fft = np.fft.fft(dataFG,14)

# Select the real part only
df_real1=df_fft.real


print (df_real1)
print (df_real1.shape)
print ('========================')
```

Figure 3.5. The FFT method used to extract features from EEG signals

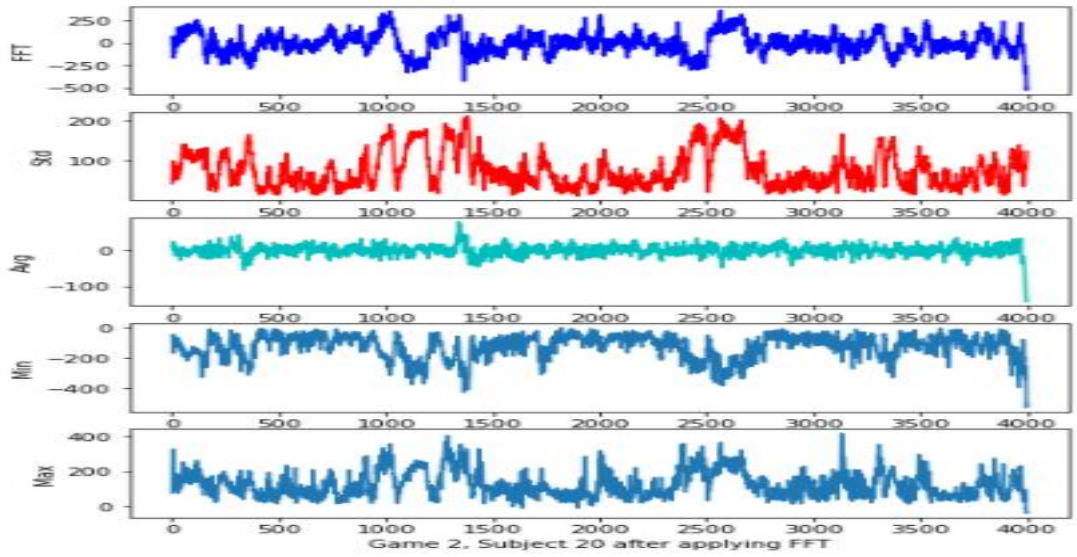

Figure 3.6. The signals were obtained from subject number 20 (according to the G2) using FFT + SF method.

The wavelet packet decomposition (WPD) method is based on wavelet decomposition (WD) [93]. The use of WD to decompose the signal leads to splitting the signal into two complementary parts: low frequency and high frequency. Low frequency is called an approximation part. Where, after applying several decompositions of the signal, a series of approximations can be obtained. Also, the difference in approximation between two successive decompositions is called details [49]. The approximation and detail are half the size of the original signal at the first stage of decomposition. Then the decomposition is repeated in the second stage to produce four signals, each of which is equal to a quarter of the original signal, called: AA (approximation of the approximation), DA (detail of the approximation), AD (approximation of the detail) and DD (detail of the detail). The third stage of decomposition produces eight signals called AAA, DAA, ADA, DDA, AAD, DAD, ADD, and DDD [94]. In this thesis, a third-level decomposition was used, and a total of 14 features were obtained for each EEG channel. Figure 3.7 shows the code for the first of three levels of signal decomposition.

```
#Define Wavelate Transform level 3
x=dataFG
coeffs=pywt.wavedec(x,'db1',mode='periodic',level=3)

CA3 ,CD3,CD2,CD1=coeffs

# make inverse wavelate transform
yx=pywt.waverec(coeffs, 'db1',mode='periodic')
```

Figure 3.7. The WPD method, the first of three levels

3.4. Data Preprocessing and Normalization

Since the data set used is a previously processed data set, we did not implement any of the data processing methods. We just applied the data normalization method to get data with values between 0 and 1. learn. preprocessing.StandardScaler() method from the Scikit-Learn library was used to normalize our data. Data standardization aims to reduce all features to a common domain between 0 and 1 and reduce the training and testing time. learn. preprocessing.StandardScaler() is applied to each feature independently using Equation 3.1:

$$X_{scaled} = \frac{X - X_{mean}}{X_{std}} \tag{3.1}$$

Where X: refers to the input signal, X_{mean}: refers to the mean of the training samples, and X_{std}: refers to the standard deviation of the training samples. The value of X_{mean} equal zero if the with_mean=False, and the value of X_{std} equal one if the with_std=False. Figure 3.8 shows the data normalization method that was used in this thesis. The variable in_Data: refers to the input features.

```
# Data normalization
normalized = preprocessing.StandardScaler().fit(in_Data)
X_scaled = normalized.transform(in_Data)
print('X_scaled=',X_scaled.shape)
```

Figure 3.8. Data normalization method using the Scikit-Learn library.

3.5. Data Partitioning

At this stage, the data that we obtained through various methods of feature extraction was divided into three sets: the first set is the training set with 70% of the total data we obtained, the second set is the testing set with 20% of the total data, and the third set is the Validation set with 10% of the total data. The train_test_split() function from the sklearn.model_selection library was used to initially split the data into two sets (70% and 30%) for both training and testing, respectively. Figure 3.9 shows the code used to split the data. Then the test set was divided into (20% and 10%) for both testing and Validation.

```
x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.3, random_state=42,shuffle=True)
```

Figure 3.9. The code used to make training and testing sets

3.6. Building Proposed Models

In this thesis, two deep learning models are used: ID CNN, LSTM, and a hybrid 1D CNN + LSTM models to detect human emotions based on physiological brain signals. In this section, we will review the architecture of both models used.

3.6.1. 1D CNN Architecture

CNNs are widely used in human activity recognition studies. 1D CNN is one of the algorithms helpful in analyzing EEG signals in the literature [69]. The 1D CNN architecture proposed in this thesis is composed of two convolutional layers. Each convolutional layer consists of 16 filters and three kernel sizes. The feature map size was 16 x 1 at the output of max pooling using the SF method, 16 x 17 using the SF + FFT method, 16 x 7 using the SF + PSD method, and 16 x 17 using the SF + WPD method. A dropout of 0.2 has been implemented to prevent overfitting. Figure 3.10 shows the 1D CNN architecture proposed in this thesis. Table 3.2 shows the layer's parameters using multi-classification (arousal-valence model) with three methods: SF, SF + PSD, and SF +WPD. Also, APPENDIX- 1 shows the main code of 1D CNN using the SF method. And Appendix- 2 shows the testing process for the same model.

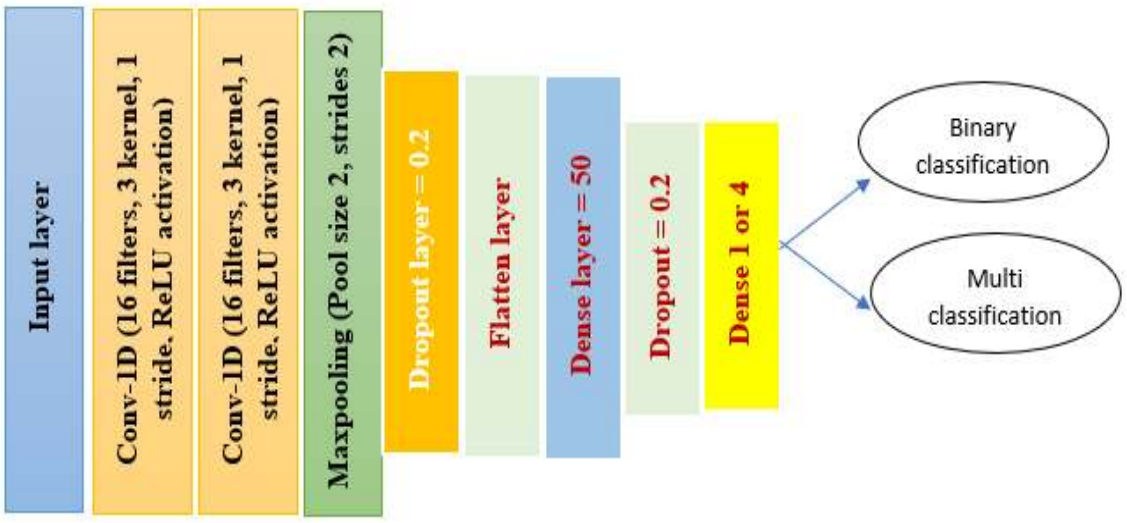

Figure 3.10. The 1D CNN architecture is used in this thesis.

Table 3.2. The layer parameters of the 1D-CNN model using SF, SF+PSD, and SF + WPD methods, multi-class classification

Layer	SF method		SF + PSD method		SF + WPD method	
	Output	Param	Output	Param	Output	Param
Input	(None, 6, 1)	-	(None, 14, 1)	-	(None, 35, 1)	-
Conv1D	(None, 6, 16)	64	(None, 14, 16)	64	(None, 35, 16)	64
Conv1D	(None, 3, 16)	784	(None, 14, 16)	784	(None, 18, 16)	784
Max Pooling	(None, 1, 16)	0	(None, 7, 16)	0	(None, 9, 16)	0
Batch normalization	(None, 1, 16)	64	(None, 7, 16)	64	(None, 9, 16)	64
Flatten	(None, 16)	0	(None, 112)	0	(None, 144)	0
Dense	(None, 50)	850	(None, 50)	5650	(None, 50)	7250
Dropout	(None, 50)	0	(None, 50)	0	(None, 50)	0
Dense	(None, 4)	204	(None, 4)	204	(None, 4)	204
Total params		1966		6766		8366
Trainable params		1934		6766		8334
Non-trainable params		32		0		32

3.6.2. LSTM Architecture

LSTM networks are part of RNNs that learn patterns from time-series data. LSTM networks are used in many applications such as speech recognition, weather forecasting, or stock market forecasting [70]. Also, LSTMs networks have been used extensively in classifying ECG and EEG signals [95]. For example, Tan et al. [96], introduced an approach based on a combination of LSTM and CNN to detect coronary artery disease (CAD) using ECG signals. LSTM differs from RNN in solving the vanishing gradient problem. The architecture of LSTM networks consists of several cells, and each cell consists of three gates, namely forget, input, and output gates. Each cell acts as a memory to store, pass, or erase information. In this thesis, two-layer LSTM networks are consisting of 16 units with the ReLU activation function. Then, it is followed by a Dropout layer of 0.1 to prevent overfitting. After that, it is followed by a flattening layer. Then the dense layer consists of 100 neurons and then the output layer with sigmoid activation in binary classification and SoftMax in multi-classification. Figure 3.11 shows the LSTM model used in this thesis. Also, Table 3.3 lists the parameters of all layers implemented using the SF method and multi-classification.

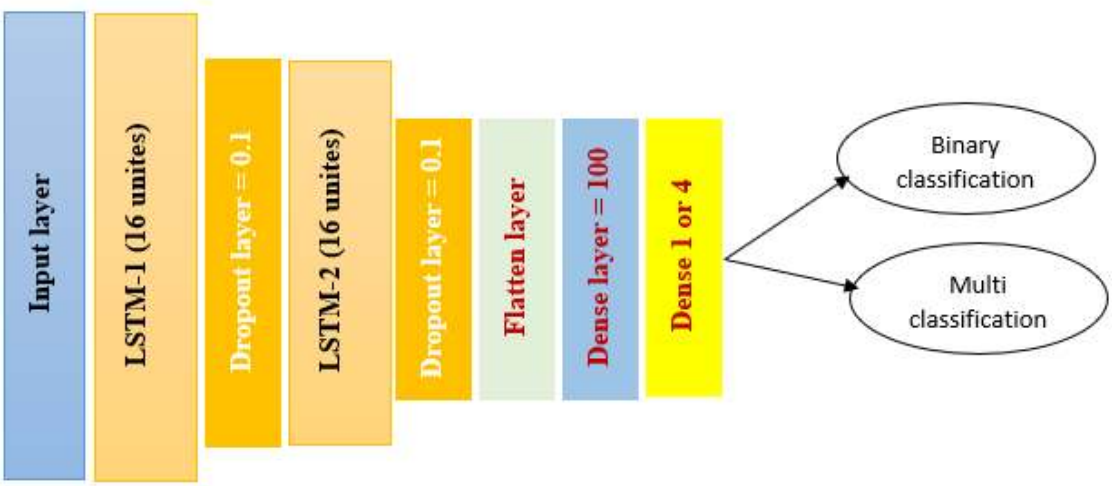

Figure 3.11. The LSTM model is used in this thesis.

Table 3.3. The layer parameters of the LSTM model using SF method, multi-class classification

Layer	SF method	
	Output	Param
Input	(None, 6, 1)	-
LSTM 1	(None, 6, 16)	1152
Dropout	(None, 6, 16)	0
LSTM 2	(None, 16)	2112
Dropout	(None, 16)	0
Flatten	(None, 16)	0
Dense	(None, 100)	1700
Dense	(None, 4)	404
Total params		5368
Trainable params		5368
Non-trainable params		0

3.6.3. Hybrid Model: 1D CNN + LSTM

A hybrid model was built by combines CNN and RNN networks to recognize human emotions. The model relies on two 1D CNN layers followed by two LSTM layers to extract features from the input data. Each 1D CNN layer consists of 32 filters, nine kernels, and a ReLU activation function. Also, each LSTM layer consists of 32 units and a ReLU activation function. Then, a fully

connected layer, then a Dense layer consisting of 100 neurons, an output layer with an activating function sigmoid in binary classification, and SoftMax in multiclassification. Figure 3.12 shows the hybrid model that was used in this thesis. And Table 3.4 shows the params of all layers using the SF method.

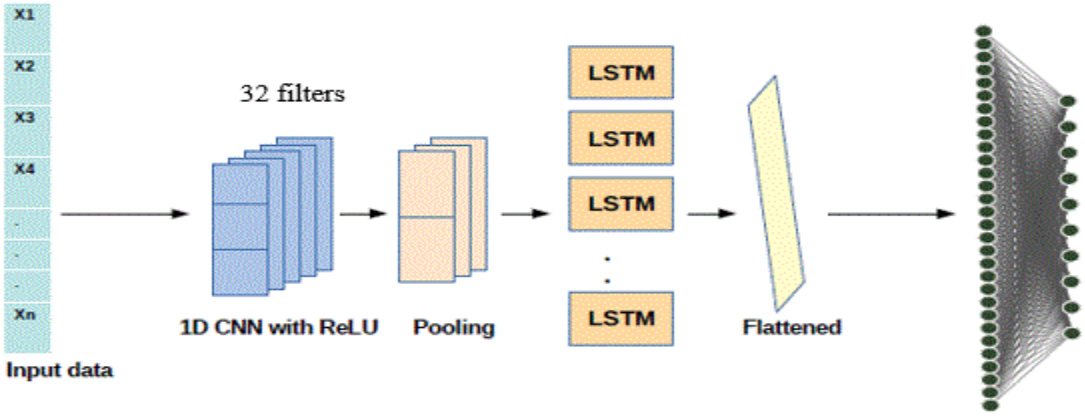

Figure 3.12. The hybrid model (1D CNN + LSTM) architecture [70].

Table 3.4. The total params for the hybrid model using the SF method.

Layer	**SF method Multi-class classification**		**SF method Binary-class classification**	
	Output	Param	Output	Param
Input	(None, 6, 1)	-	(None, 6, 1)	-
Conv1D	(None, 6, 32)	320	(None, 6, 32)	320
Conv1D	(None, 6, 32)	9248	(None, 6, 32)	9248
Max Pooling	(None, 3, 32)	0	(None, 3, 32)	0
Batch normalization	(None, 3, 32)	128	(None, 3, 32)	128
LSTM	(None, 3, 32)	8320	(None, 3, 32)	8320
LSTM	(None, 3, 32)	8320	(None, 3, 32)	8320
Flatten	(None, 96)	0	(None, 96)	0
Dense	(None, 100)	9700	(None, 100)	9700
Dropout	(None, 100)	0	(None, 100)	0
Dense	(None, 4)	404	(None, 1)	101
Total params		36440		36137
Trainable params		36376		36073
Non-trainable params		64		64

3.7. Convolutional Layers

As shown in Figures 3.10 and 3.12, the convolutional layers have kernels of sizes 3. Consequently, the receptive field size is (3×3×1). One refers to the depth of the signal input matrix. The convolutional layer function includes the same border mode parameter, which adds (zero) padding around the input to produce the same-sized feature map as layer output. Figure 3.13 shows that a single kernel (3×3) produces a two-dimensional feature map. For every layer deeper into the network, a larger kernel is used. Deeper layers extract more complex and class-specific information, which requires a larger number of trainable parameters.

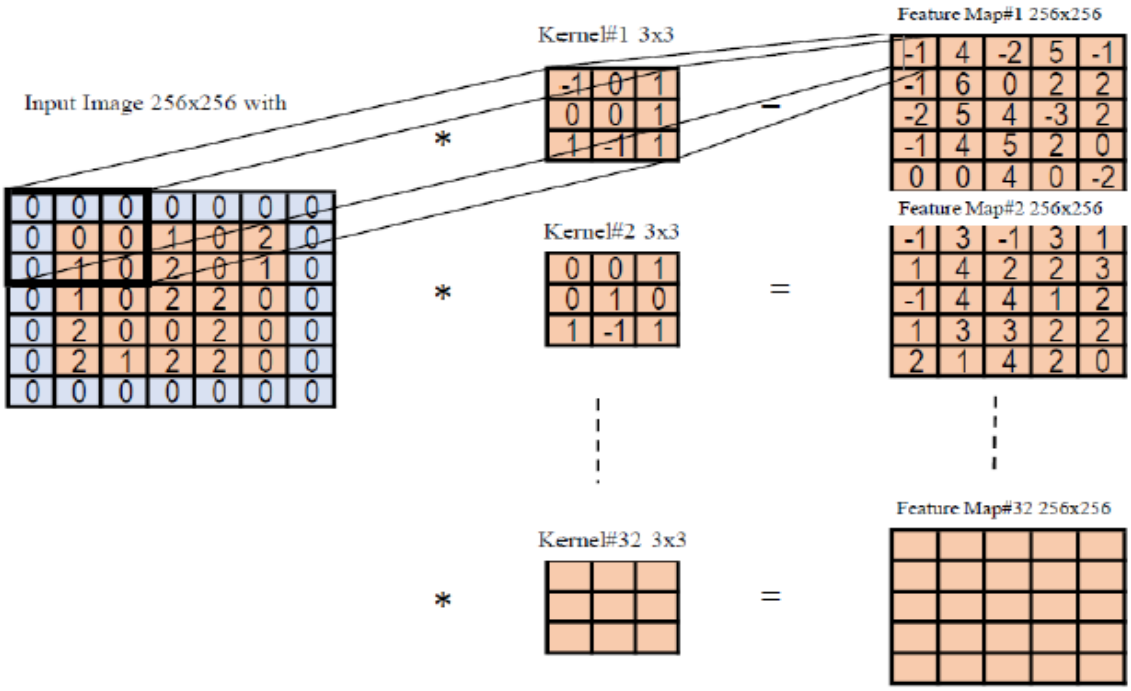

Figure 3.13. Feature Mapping from Input using 3x3 kernels.

A kernel is a matrix of trainable weights; hence the total number of weights depends on its size. A single kernel of size (3×3×1) in the first convolutional layer contains nine weights. In this convolutional layer, 32 kernels, therefore, have 9×32 = 288 weights. Hence every kernel adds additional trainable weights to the neural network. The number of weights for each convolutional layer is calculated using the equation 3.2:

$$No.of\ weights = (IF * K * D) + B \tag{3.2}$$

Where IF is the number of input features, K is the number of the kernels, D is the dimension of the kernel, and (B) is the bias value.

3.8. ReLU Activation Function Layers

Each convolutional layer in CNN is followed by a nonlinear activation function (or a piece-wise linear) function. The main objective of the activation function is to squash the real-valued input to produce a non-linear output within a small range such as [0; 1] and [-1; 1]. Applying a nonlinear function after the weight layers is highly important since it allows a neural network to learn nonlinear mappings. A stacked network of weight layers is equivalent to a linear mapping from the input domain to the output domain without nonlinearities.

The nonlinear activation function used in the network is real. A simple and quick computation activation function maps the input to a 0 if negative and keeps its value unchanged if it is positive, as shown in equation 3.3 and Figure 3.14.

$$y = \begin{cases} 0 & if\ x < 0 \\ x & if\ x > 0 \end{cases} \tag{3.3}$$

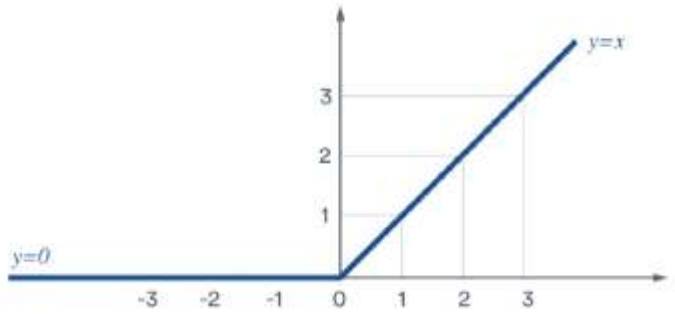

Figure 3.14. ReLU activation function

3.9. Max-Pooling Layers

A pooling layer is a type of neural network layer, which has no learnable parameters. As in convolutional layers, the max-pooling operation needs to specify the size of the pooled region and the stride. In the modified 1D CNN and hybrid 1D CNN + LSTM, we use a max-pooling window 2x2 with a stride of 2. It down-samples the input features spatial size to half by selecting the maximum values within the pooling window. The max-pooling layers make network performance invariant to the small translations. Together with a convolutional layer, they are basic components of convolutional neural networks. Figure 3.15 illustrates how an input feature 4x4 is minimized to half by using the max-pooling function.

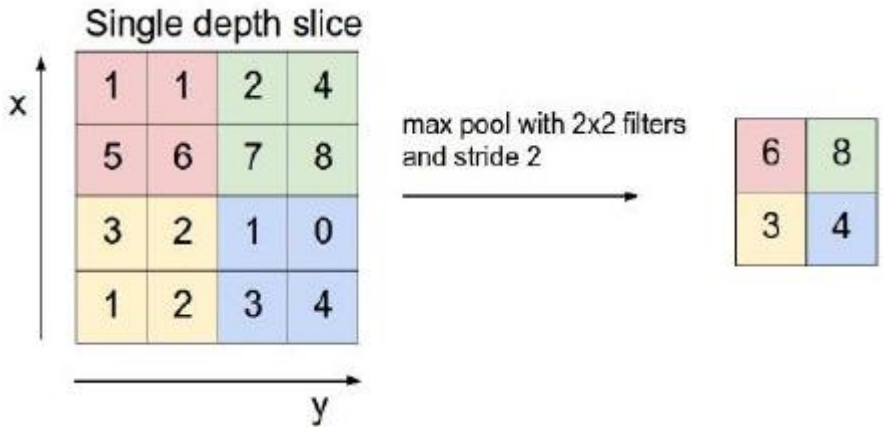

Figure 3.15. Max pooling layer

3.10. Training Our Models

In this thesis, Python with Jupyter notebook software packages, deep learning, and parallel computing toolboxes are used to train and test the modified 1D CNN, LSTM, and hybrid 1D CNN + LSTM models' architecture to recognize human emotions using EEG signals. Besides, Python is installed on the HP workstation with the Core i7 Central Processing Unit (CPU), 8.0 GB RAM., and Intel HD Graphics Family 2.0 GB. Sometimes we used the colab cloud platform if the training process was out of memory. The training process included the following steps. Before training the models, many training options were identified until we reached the options shown in Table 3.5, which are the best options for training the proposed models.

Table 3.5. Training Options of the Modified models.

Parameter	Value
Optimization Algorithm	Adam
Initial Learn Rate	1e-3
Max Epochs	100
Min Epochs	30
Learning rate	0.001
Shuffle	True
Batch size	42
metrics	Accuracy
Loss function for binary	binary_crossentropy
Loss function for multi	sparse_categorical_crossentropy

4. RESULTS AND DISCUSSION

4.1. Introduction

In this section, we focused on the results we obtained using the modified three models. We calculated the confusion matrix. The performance of the models is measured in terms of global accuracy, sensitivity, specificity, precision, recall, and f1-score. Finally, we make a comparison between the modified models and other related works.

4.2. Results

4.2.1. Global Accuracy, Sensitivity, and Specificity

The performance of the modified models is tested using 30% data samples from the total samples data set. Table 4.1 shows the global accuracy, sensitivity, and specificity metrics for the 1D CNN model we obtained using all methods. We observe from Table 4.1 that the best binary classification (discrete emotion model) accuracy obtained is 99.11% using the SF+WPD method. Then, 86.60% using the SF+PSD method. The best multi-classification (arousal-valence emotion model) accuracy is 98.50% using the SF+WPD method as well. It was followed by 82.87% using the SF+FFT method. The best sensitivity in binary classification is 99.66% using the SF method. The worst sensitivity in binary classification is 91.22% using the SF+FFT method.

Table 4.1. The accuracy, sensitivity, and specificity parameters for the 1D CNN model.

Method	Classes	Accuracy (%)	Sensitively (%)_	Specificity (%)
Statistical Feature (SF)	Binary	82.93	99.66	66.20
	Multi	74.84	79.79	91.68
SF + Welsh Power spectral density (PSD)	Binary	86.60	91.80	81.40
	Multi	73.96	74.31	91.30
SF + Fast Fourier Transform (FFT)	Binary	85.22	91.22	79.12
	Multi	82.87	83.09	94.30
SF + Wavelet Packet Decommission (WPD)	Binary	99.11	99.18	99.03
	Multi	98.50	98.52	99.50

Tables 4.2 and 4.3 show the global accuracy, sensitivity, and specificity parameters for the LSTM and the hybrid 1D CNN +LSTM models, respectively. We observe from Table 4.2 that the best binary classification accuracy obtained is 98.81% using the SF+WPD method. Then, 85.77% using the SF+PSD method. The best multi-classification accuracy is 98.44% using the SF+WPD method as well. Then, 82.06% using the SF+FFT method. The best sensitivity in binary

classification is 99.31% using the SF+WPD method. The worst sensitivity in binary classification is 90.25% using the SF+PSD method.

As for Table 4.3, the best accuracy for binary classification is 98.96% using the SF+WPD method. And the worst accuracy is 73% using the SF+PSD method. The best sensitivity for binary classification is 99.29%. The best sensitivity for multi-class classification is 98.40%. And the worst sensitivity is 83%.

Figures 4.1 and 4.2 show the performance of binary-class classification and multi-class classification with all models and methods. As shown in Figure 4.1, the hybrid model is more efficient than the rest of the models in the binary-class classification, as the accuracy, sensitivity, and specificity are more than 80% using all methods. Figure 4.2 shows the hybrid model is almost more stable as well.

Table 4.2. The accuracy, sensitivity, and specificity parameters for the LSTM model.

Method	**Classes**	**Accuracy (%)**	**Sensitively (%)**	**Specificity (%)**
Statistical Feature (SF)	Binary	82.81	94.96	70.67
	Multi	69.72	72.7	89.94
SF + Welsh Power spectral density (PSD)	Binary	85.77	90.25	81.28
	Multi	73	73.63	91
SF + Fast Fourier Transform (FFT)	Binary	84.23	91.26	76.43
	Multi	82.06	82.15	94
SF + Wavelet Packet Decommission (WPD)	Binary	98.81	99.31	98.31
	Multi	98.44	98.47	99.48

Table 4.3. The accuracy, sensitivity, and specificity parameters for the hybrid 1D CNN + LSTM model.

Method	**Classes**	**Accuracy (%)**	**Sensitively (%)_**	**Specificity (%)**
Statistical Feature (SF)	Binary	82.88	82.10	83.65
	Multi	75	83.11	92
SF + Welsh Power spectral density (PSD)	Binary	85.38	91	80
	Multi	73	73.8	91
SF + Fast Fourier Transform (FFT)	Binary	85.17	88.72	81.60
	Multi	82.78	83	94.28
SF + Wavelet Packet Decommission (WPD)	Binary	98.96	99.29	98.64
	Multi	98.37	98.40	99.46

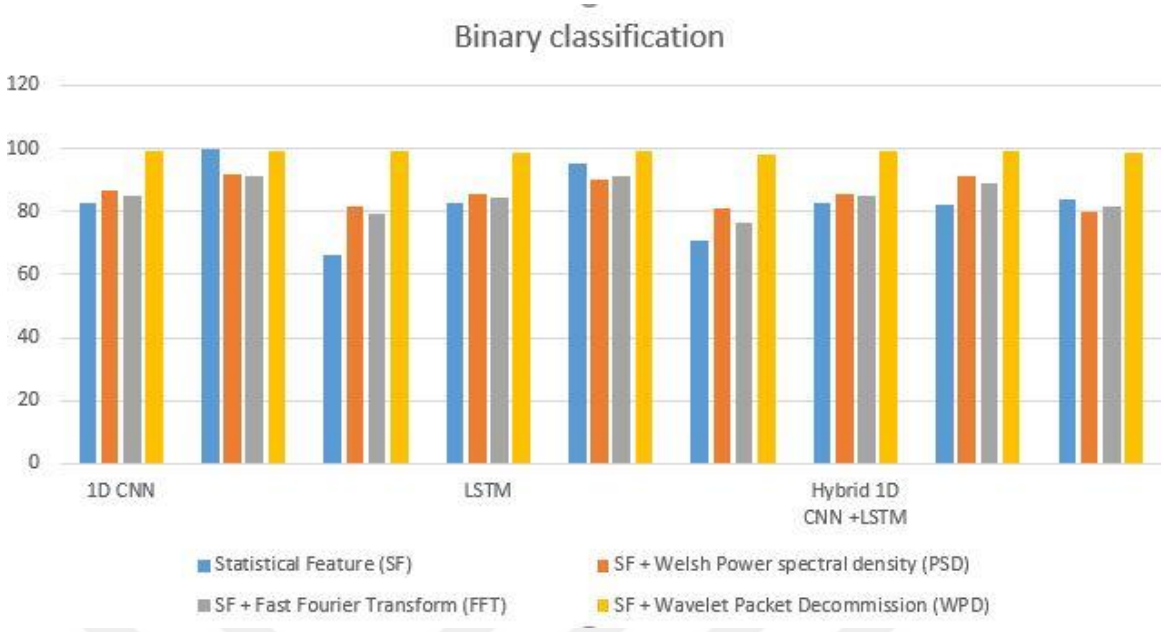

Figure 4.1. The performance of models through all methods using binary classification.

Figure 4.2. The performance of models through all methods using multi-class classification.

4.2.2. The Precision, Recall, and F1-Score

In this thesis, we calculate the precision, recall, and f1-score metrics to find the accuracy of each class. Table 4.4 shows the precision, recall, and f1-score parameters for binary classification (discrete model) with all feature extraction methods, and Tables 4.5- 4.8 show the precision, recall, and f1-score parameters for multi-class classification (arousal-valence model) with all feature

extraction methods. As shown in Table 4.4, the best accuracy for classifying negative and positive emotions is 99% using the F1-Score parameter and the SF+WPD method. Also, Table 4.4 shows that SF+WPD and SF+PSD methods are more effective than the other two methods. The accuracy of classifying negative and positive emotions for both methods is more than 80% with all models and all parameters, as shown in Figure 4.3.

Table 4.4. The precision, recall, and f1-score parameters for binary classification (discrete model) with all feature extraction methods

Method	model	Negative class			Positive class		
		Precision (%)	Recall (%)	F1-Score (%)	Precision (%)	Recall (%)	F1-Score (%)
SF	1D CNN	100	66	80	75	100	85
	LSTM	93	81	85	83	90	86
	1D CNN+ LSTM	82	84	83	83	82	83
SF+PSD	1D CNN	91	81	86	83	92	87
	LSTM	89	81	85	83	90	86
	1D CNN+ LSTM	90	80	85	82	91	86
SF+FFT	1D CNN	90	79	84	81	91	86
	LSTM	90	76	83	79	91	85
	1D CNN+ LSTM	88	82	85	83	89	86
SF+WPD	1D CNN	99	99	99	99	99	99
	LSTM	99	98	99	98	99	99
	1D CNN+ LSTM	99	99	99	99	99	99

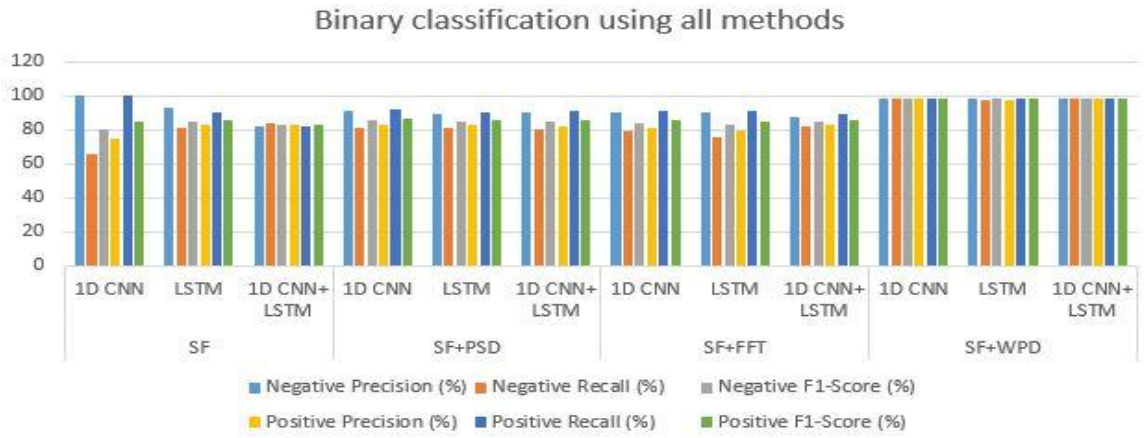

Figure 4.3. The precision, recall, and f1-score for binary classification with all methods.

Table 4.5. The precision, recall, and f1-score for multi-class classification using the Statistical feature method.

Classes	1D CNN			LSTM			1D CNN+LSTM		
	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %
LANV	69	94	79	64	75	69	70	93	79
LAPV	73	86	79	71	90	79	70	92	79
HANV	76	87	81	68	73	71	67	96	79
HAPV	82	53	64	76	54	63	94	54	68

Table 4.6. The precision, recall, and f1-score for multi-class classification using SF + PSD method.

Classes	1D CNN			LSTM			1D CNN+LSTM		
	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %
LANV	74	70	72	76	67	71	73	70	71
LAPV	75	85	80	74	87	80	74	86	80
HANV	77	77	77	78	75	76	75	77	76
HAPV	70	65	67	64	66	65	71	63	67

Table 4.7. The precision, recall, and f1-score for multi-class classification using SF + FFT method.

Classes	1D CNN			LSTM			1D CNN+LSTM		
	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %
LANV	85	78	81	78	80	79	85	77	81
LAPV	83	89	86	84	87	86	82	90	86
HANV	87	83	85	86	83	84	87	83	85
HAPV	78	82	80	80	79	79	77	83	80

Table 4.8. The precision, recall, and f1-score for multi-class classification using SF + WPD method.

Classes	1D CNN			LSTM			1D CNN+LSTM		
	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %	Precision %	Recall %	F1-Score %
LANV	100	99	99	100	99	99	100	99	99
LAPV	99	96	98	99	96	98	99	96	98
HANV	98	98	98	98	100	99	98	100	99
HAPV	97	99	98	99	99	99	96	100	98

As can be seen in the tables above, the four classes named LANV, LAPV, HANV, and HAPV represent the four zones in the arousal-valence model. We note in the four tables above that the classification of the LAPV zone achieved the best performance using all methods. Achieve an accuracy of f1-score between 79% and 98%, using the SF method and SF+WPD method,

respectively. Figures 4.4- 4.7 show the difference in the performance of all four zones with the different models used. Also, the classification of the HAPV zone achieved the worst performance using all methods almost. Achieve an accuracy of f1-score between 63% and 99%, using the SF method and SF+WPD method, respectively.

Also, we noticed that the SF method is not stable in all zones, As shown in Table 4.5. For example, in a 1D CNN model, LANV zone, the accuracy value jumped from 69% using the precision parameter to 94% using the recall parameter. In an LSTM model, LAPV zone, the accuracy value jumped from 71% using the precision parameter to 90% using the recall parameter. At the same time, the value decreased from 82% to 53% in the 1D CNN model, HAPV zone.

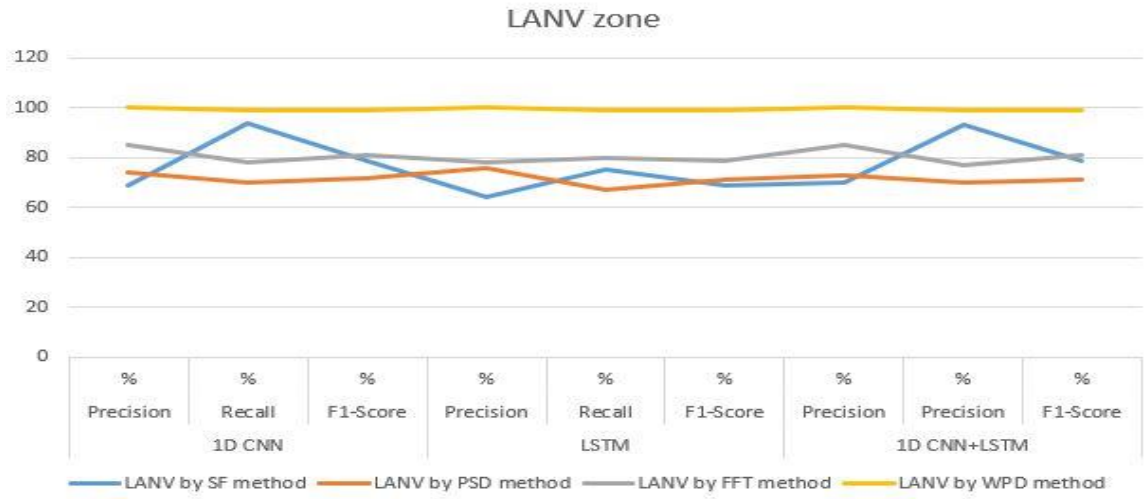

Figure 4.4. The performance of the LANV zone among all metrics and all methods.

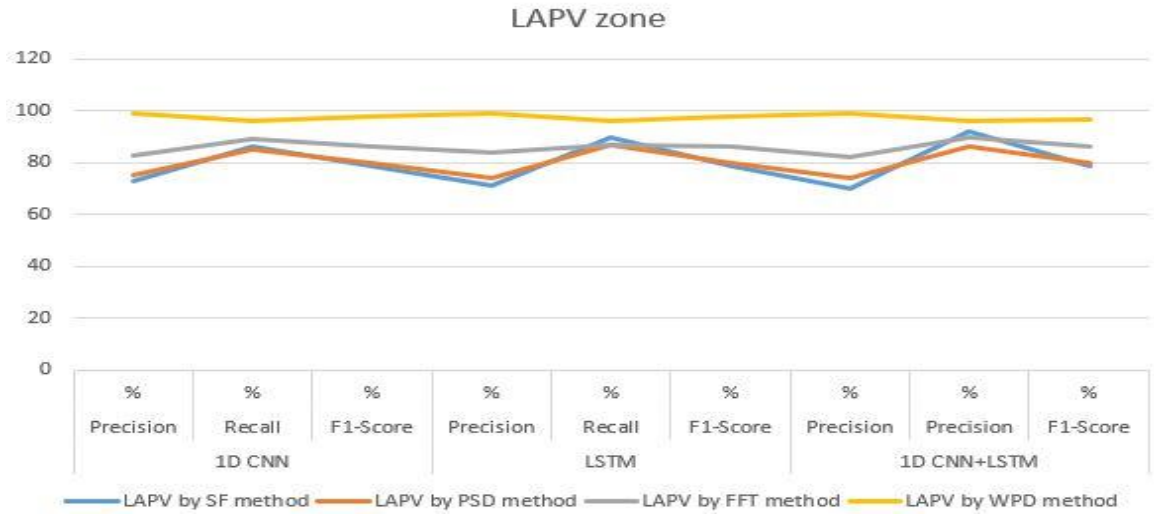

Figure 4.5. The performance of the LAPV zone among all metrics and all methods.

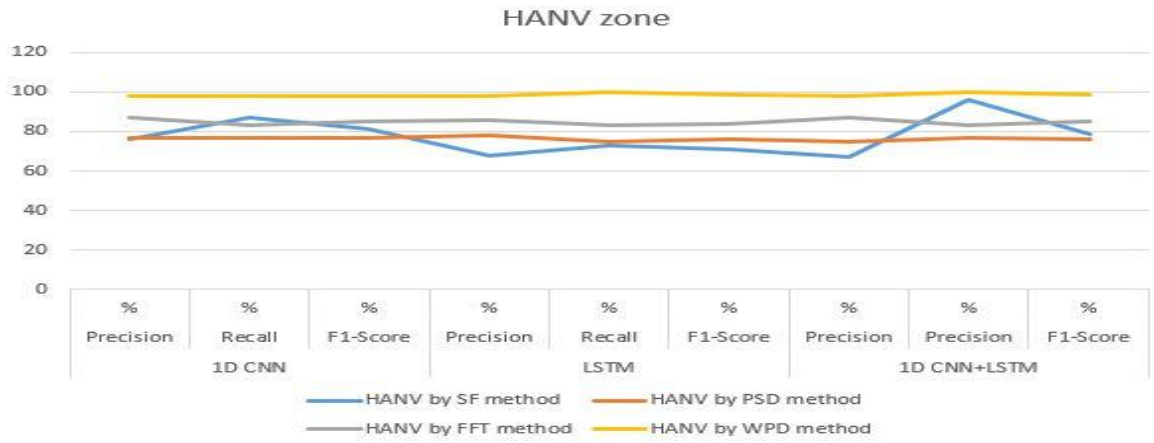

Figure 4.6. The performance of the HANV zone among all metrics and all methods.

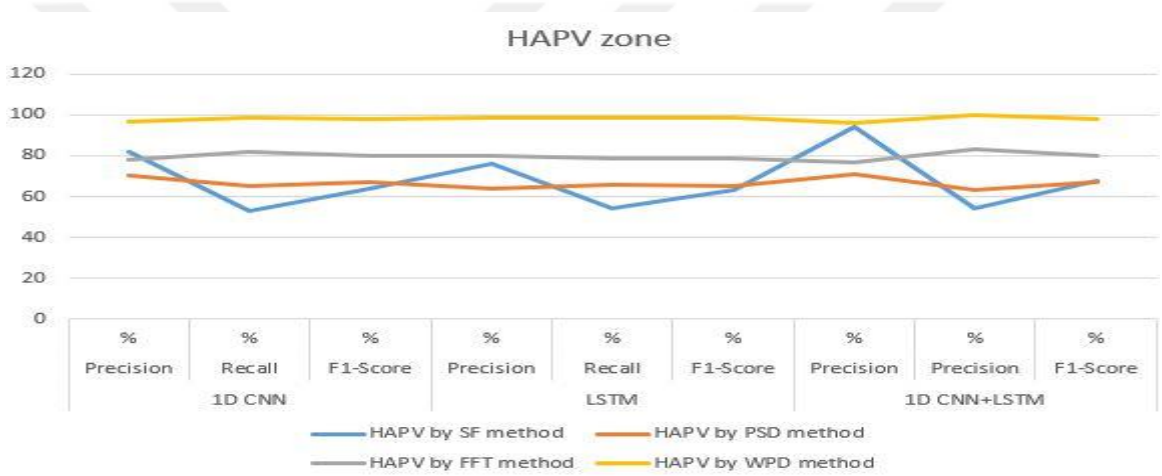

Figure 4.7. The performance of the HAPV zone among all metrics and all methods.

4.2.3. Confusion Matrix Metric

In this thesis, we compute the confusion matrix as another measurement parameter to confirm our testing results. Figure 4.8 shows the confusion matrix matric for the 1D CNN model. Whereas, Figure 4.8 (a) and Figure 4.8(b) represent the confusion matrix of the SF method, binary-class, and multi-class classification. Figure 4.8 (c) and Figure 4.8(d) represent the confusion matrix of the PSD method, binary-class, and multi-class classification. Figure 4.8 (e) and Figure 4.8(f) represent the confusion matrix of the FFT method, binary-class, and multi-class classification. Figure 4.8 (g) and Figure 4.8(h) represent the confusion matrix of the WPD method, binary-class, and multi-class classification.

The same for Figures 4.9 and 4.10. Where Figure 4.9 represents the confusion matrix that applied using the LSTM model. And Figure 4.10 means the confusion matrix implemented using the hybrid model (1D CNN + LSTM).

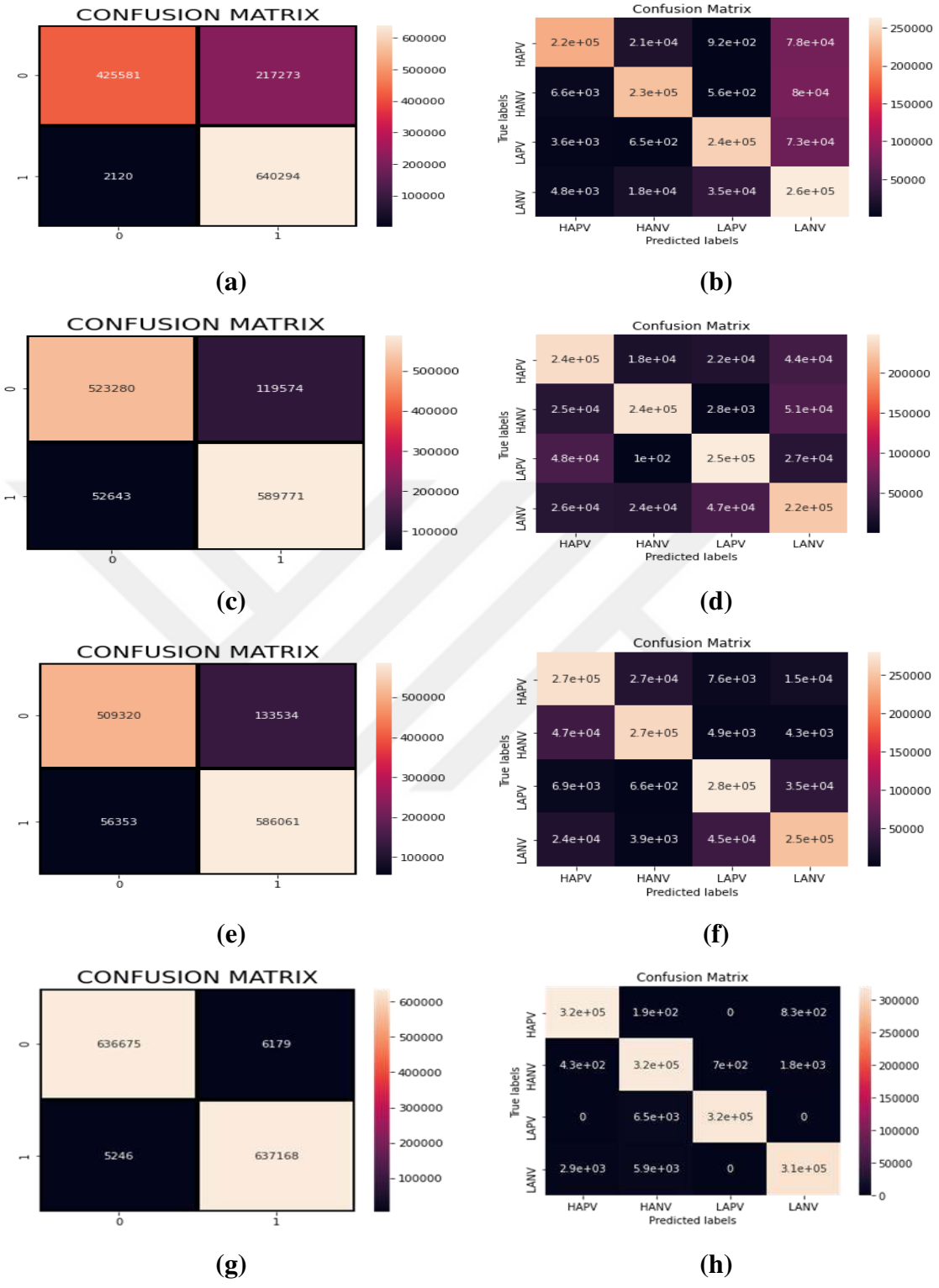

Figure 4.8. The confusion matrix metric for the 1D CNN model. (a) represents the SF method with binary classification. (b) describes the SF method with multi-classification. (c) illustrates the PSD method with binary classification. (d) represents the PSD method with multi-classification. (e) describes the FFT method with binary classification. (f) illustrates the FFT method with multi-classification. (g) represents the WPD method with binary classification. (h) describes the WPD method with multi-classification.

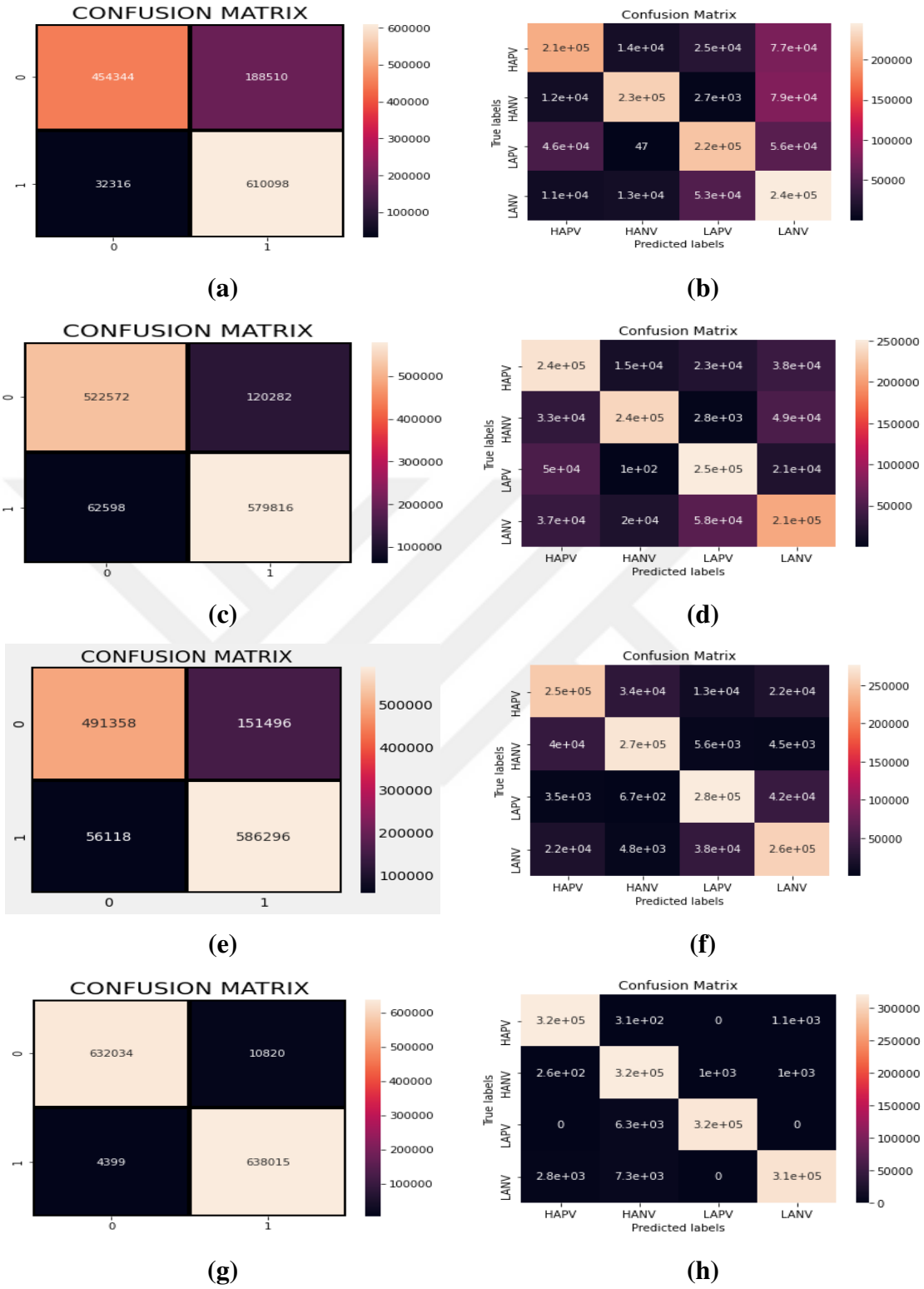

Figure 4.9. The confusion matrix metric for the LSTM model. (a) represents the SF method with binary classification. (b) describes the SF method with multi-classification. (c) illustrates the PSD method with binary classification. (d) represents the PSD method with multi-classification. (e) describes the FFT method with binary classification. (f) illustrates the FFT method with multi-classification. (g) represents the WPD method with binary classification. (h) describes the WPD method with multi-classification.

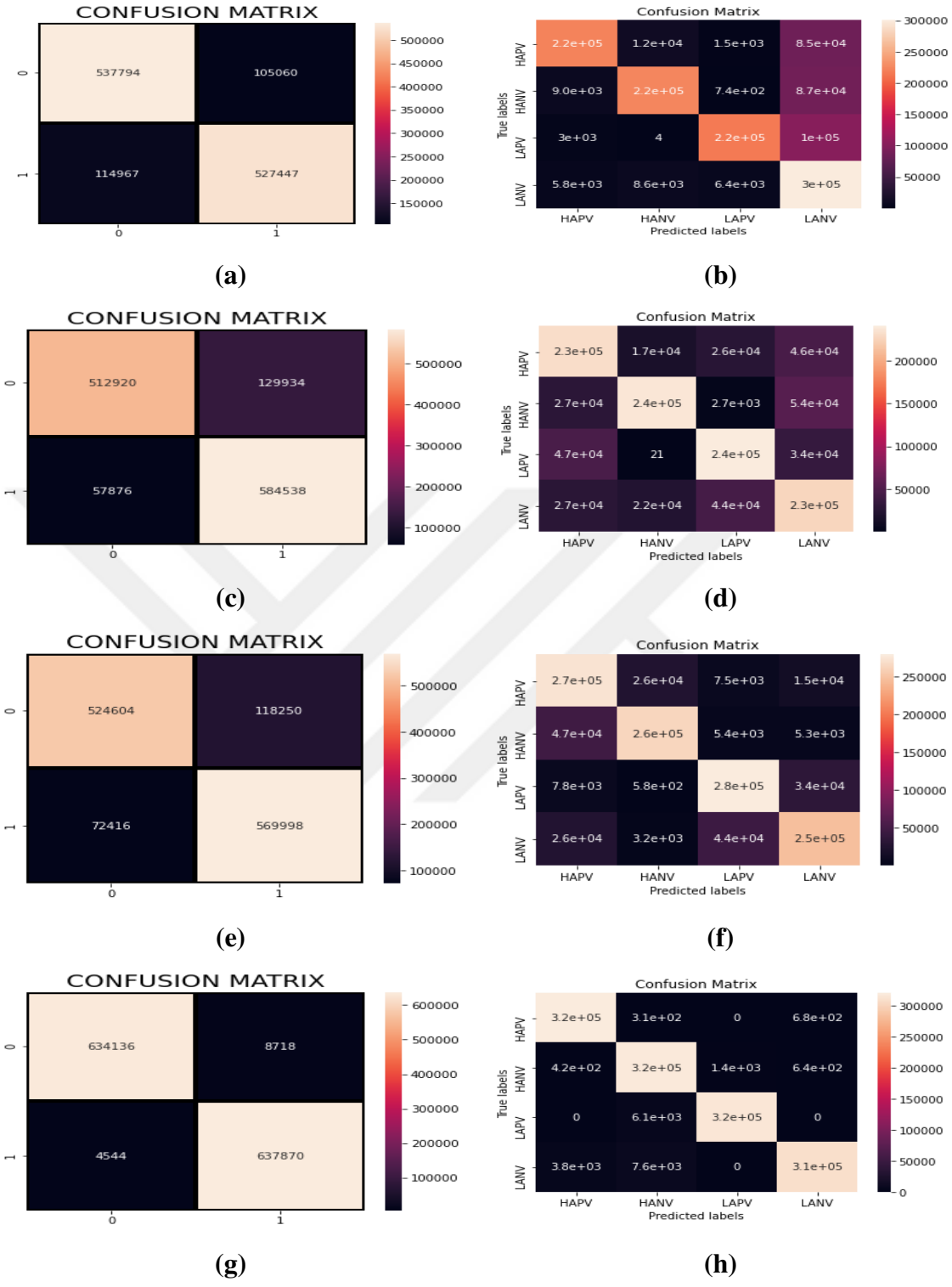

Figure 4.10. The confusion matrix metric for the hybrid model. (a) represents the SF method with binary classification. (b) illustrates the SF method with multi-classification. (c) describes the PSD method with binary classification. (d) represents the PSD method with multi-classification. (e) describes the FFT method with binary classification. (f) illustrates the FFT method with multi-classification. (g) represents the WPD method with binary classification. (h) describes the WPD method with multi-classification.

4.2.4. Training Progress Plot

After defining the training options, it's time to train the modified models using the model. fit() function as shown in Figure 4.11. Training the networks requires four predefined parameters. Model architecture (training data: input and output), number of epochs, number of batch_size, and the value of validation_data. Figures 4.12- 4.17 represent the training curve and the loss function curve for samples of the models proposed in this thesis.

```
# Compiling the CNN
opt=keras.optimizers.Adam(
    learning_rate=0.001)
model.compile(loss='sparse_categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
# Fitting to the training set
history = model.fit(x_train,y_train,epochs=50,batch_size=42,validation_data=(x_test, y_test))
```

Figure 4.11. Compiling and fitting 1D CNN model

Figure 4.12. Training Progress Plot for 1D CNN model using SF method and multi-classification.

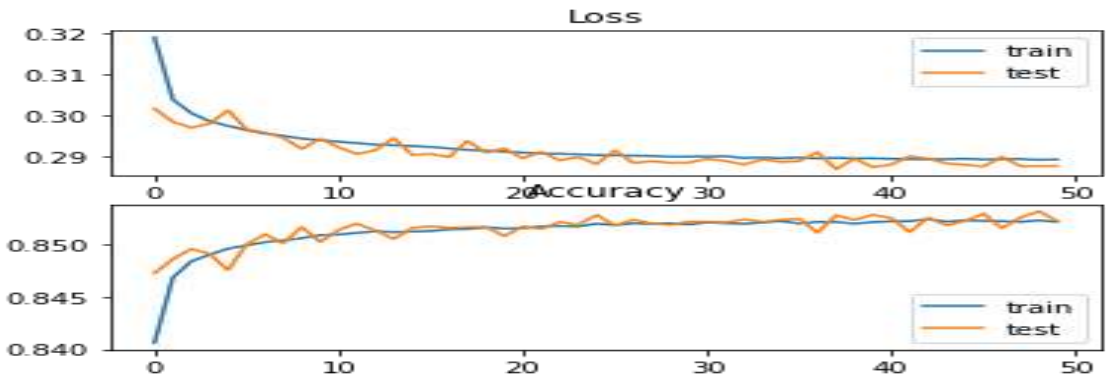

Figure 4.13. Training Progress Plot for 1D CNN model using FFT method and binary classification.

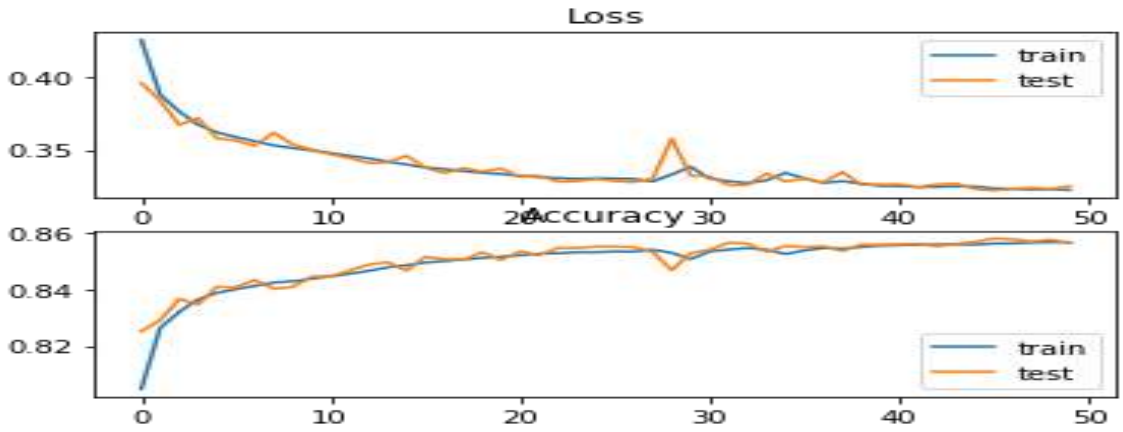

Figure 4.14. Training Progress Plot for LSTM model using PSD method and multi-classification.

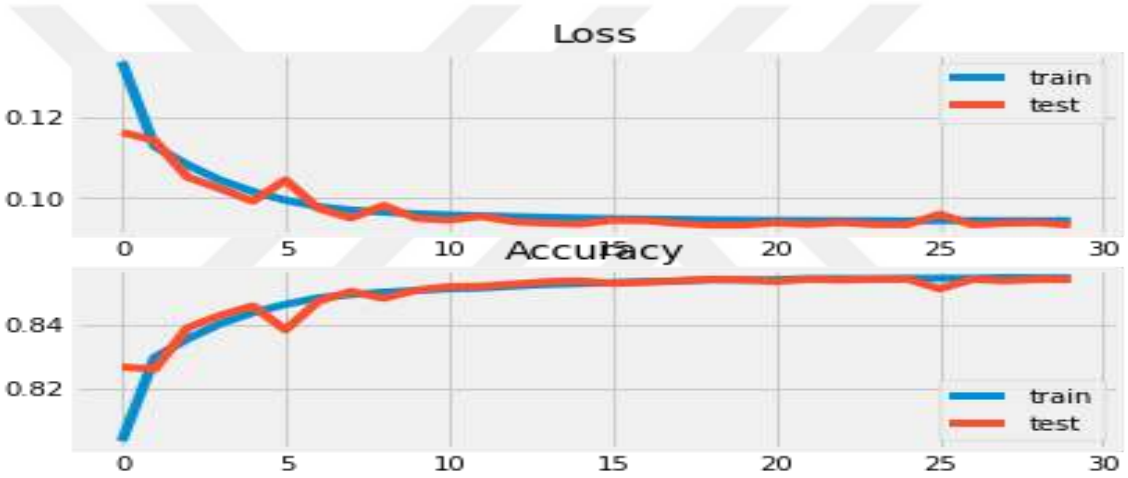

Figure 4.15. Training Progress Plot for LSTM model using FFT method and binary classification.

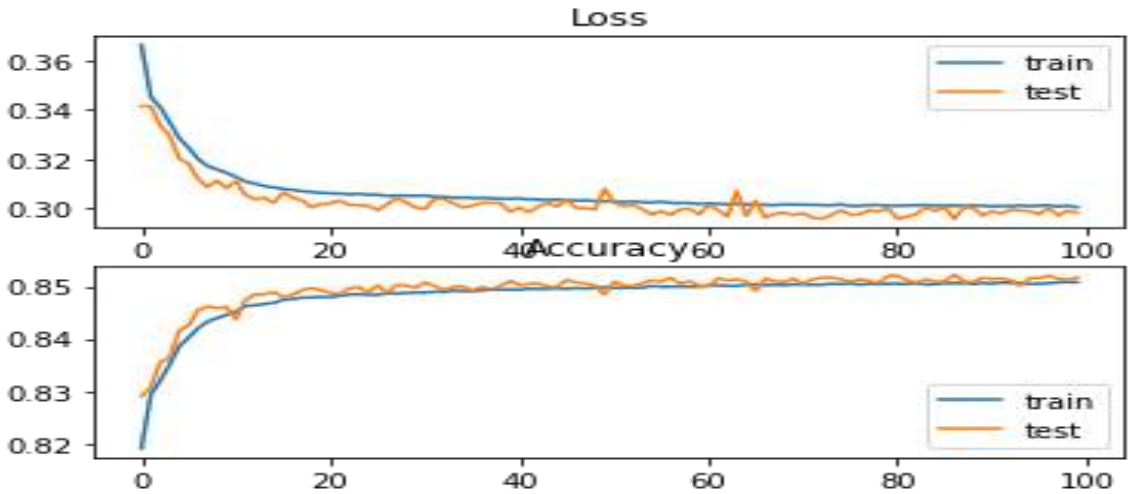

Figure 4.16. Training Progress Plot for the hybrid model using the FFT method and binary classification.

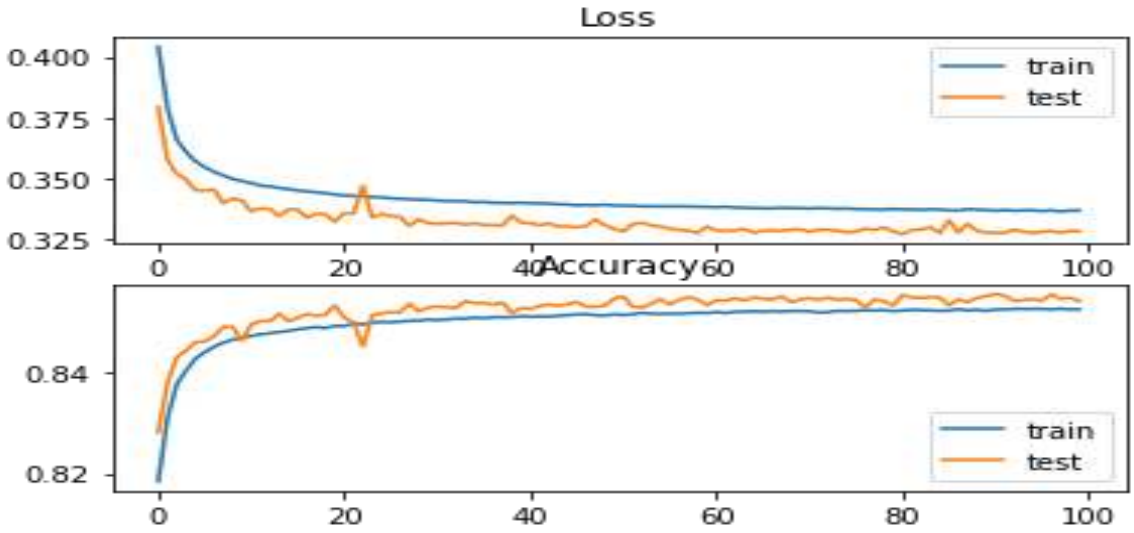

Figure 4.17. Training Progress Plot for the hybrid model using PSD method and multi-classification.

4.3. Comparison with Other Related Models

We compare the performance of the proposed models with many relevant techniques, as shown in Table 4.9. These results indicate that recognize human emotion using the proposed models achieves higher performance than other recent researches.

Table 4.9. The Performance Comparison Between the Proposed Modified models and Some Other Related Work.

Paper	Classes	Method	Model	Global Accuracy
Tarnowski, et al. [97]	3	FFT	K-NN, SVM	88%
Al-Nafjan, et al. [17]	2	PSD	DNN	82%
Ang, et al. [98]	2	DWT	ANN	81.8%
Tripathi, et al. [99]	2	SF	DNN, CNN	73.36%
T. B. Alakus and I. Turkoglu [100]	2	spectral entropy	Bidirection LSTM	76.91%
Salama, et al. [101]	4	3D-CNN, Mask-RCNN	3D-CNN, Mask-RCNN, and SVM	96.13%
George, et al. [102]	2	FFT, SF	SVM	92.36%
Proposed model	**2**	**WPD**	**1D CNN**	**99.11%**
Proposed model	**4**	**WPD**	**1D CNN**	**98.50%**

5. CONCLUSION AND FUTURE WORKS

5.1. Conclusion

The main task of this thesis is to improve the 1D CNN model performance by reducing the number of feature extraction kernels at each convolution layer of the standard or combine the 1D CNN with the LSTM model to make a hybrid model. The modified models have been trained and tested using the GAMEEMO data set. This data set is an EEG signals dataset. This data set contains both the discrete emotion model and the dimensional emotion model. This thesis study consisted of four stages: EEG signals were obtained from the data set in the first stage. Then, EEG signals were decomposed into sub-signals using four methods to extract more features from the mother signal: SF, SF+PSD, SF+FFT, and SF+WPD. After the feature extraction process, the features were classified using three modified models: 1D CNN, LSTM, and 1D CNN + LSTM hybrid model. Finally, the performances of the proposed models were measured by several metrics: global accuracy, sensitivity, and specificity. It was observed that the proposed method performed best in both binary-class classification and multi-class classification. LSTM model using the PSD method yielded the worst accuracy results in multi-class classification. Also, the LSTM model using the SF method yielded the worst accuracy results in binary-class classification.

In contrast, 1D CNN was more effective than these two models and reached high accuracy. The best performance of 99.11% was achieved using the 1D CNN model and WPD method for the binary-class classification. Also, this model got the best accuracy for the multi-class classification using the same method with an accuracy of 98.50%.

Many observations can be concluded from these three modified models. The experimental results emphasize the possibility of improving the CNN model using the feature extraction traditional methods to extract more features from the signal. Also, reducing the number of feature extraction kernels in deep learning models may positively impact model performance. This proposed approach is considered the most accurate classification of GAMEEMO datasets, using all proposed models. Also, we believe that compared to the feature extraction methods used in the literature, these methods presented easy and highly effective signal decomposition methods.

5.2. Future Works Suggestions

Finally, many important topics are suggested for future works:

1. Check whether the proposed models classify the other EEG data set with the same performance or not.
2. Use the proposed models on the biomedical signals data set, such as the EEG data set of epilepsy patients.

3. Focus on the hybrid model. The results showed that the combination of CNN and RNN might show promising results.
4. Utilizing the other software framework to train the models, such as MATLAB.
5. Use SF + WPD feature extraction method in some other signal processing fields. To measure the efficiency of combining the statistical features and the wavelet packet decomposition.

REFERENCES

[1] R. Plutchik, A general psychoevolutionary theory of emotion, in: Theor. Emot., Elsevier, 1980: pp. 3–33.

[2] S. Alhagry, A. Aly, R. A., Emotion Recognition based on EEG using LSTM Recurrent Neural Network, Int. J. Adv. Comput. Sci. Appl. 8 (2017). https://doi.org/10.14569/ijacsa.2017.081046.

[3] M. Naji, M. Firoozabadi, P. Azadfallah, Emotion classification during music listening from forehead biosignals, Signal, Image Video Process. 9 (2015) 1365–1375.

[4] E. Diener, M.Y. Chan, Happy people live longer: Subjective well-being contributes to health and longevity, Appl. Psychol. Heal. Well-Being. 3 (2011) 1–43.

[5] T.B. Alakus, M. Gonen, I. Turkoglu, Database for an emotion recognition system based on EEG signals and various computer games – GAMEEMO, Biomed. Signal Process. Control. (2020). https://doi.org/10.1016/j.bspc.2020.101951.

[6] H. Chao, L. Dong, Y. Liu, B. Lu, Emotion recognition from multiband eeg signals using capsnet, Sensors (Switzerland). 19 (2019). https://doi.org/10.3390/s19092212.

[7] M. Zhao, F. Adib, D. Katabi, Emotion recognition using wireless signals, Commun. ACM. 61 (2018) 91–100. https://doi.org/10.1145/3236621.

[8] A. Turnip, A.I. Simbolon, M.F. Amri, P. Sihombing, R.H. Setiadi, E. Mulyana, Backpropagation neural networks training for EEG-SSVEP classification of emotion recognition, Internetworking Indones. J. 9 (2017) 53–57.

[9] L. Cai, Y. Hu, J. Dong, S. Zhou, Audio-textual emotion recognition based on improved neural networks, Math. Probl. Eng. 2019 (2019).

[10] I. Lasri, A.R. Solh, M. El Belkacemi, Facial emotion recognition of students using convolutional neural network, in: 2019 Third Int. Conf. Intell. Comput. Data Sci., IEEE, 2019: pp. 1–6.

[11] S. Wioleta, Using physiological signals for emotion recognition, in: 2013 6th Int. Conf. Hum. Syst. Interact., IEEE, 2013: pp. 556–561.

[12] R. V Sharan, S. Berkovsky, R. Taib, I. Koprinska, J. Li, Detecting Personality Traits Using Inter-Hemispheric Asynchrony of the Brainwaves, in: 2020 42nd Annu. Int. Conf. IEEE Eng. Med. Biol. Soc., IEEE, 2020: pp. 62–65.

[13] T.H. Priya, P. Mahalakshmi, V.P.S. Naidu, M. Srinivas, Stress detection from EEG using power ratio, Int. Conf. Emerg. Trends Inf. Technol. Eng. Ic-ETITE 2020. (2020) 1–6. https://doi.org/10.1109/ic-ETITE47903.2020.401.

[14] T. Matlovic, P. Gaspar, R. Moro, J. Simko, M. Bielikova, Emotions detection using facial expressions recognition and EEG, Proc. - 11th Int. Work. Semant. Soc. Media Adapt. Pers. SMAP 2016. (2016) 18–23. https://doi.org/10.1109/SMAP.2016.7753378.

[15] S. Yenisoy-karakaş, M. Dörter, M. Odabasi, Jo ur na of, Sci. Total Environ. (2020) 138028. https://doi.org/10.1016/j.scitotenv.2020.138028.

[16] E.S. Salama, R.A. El-Khoribi, M.E. Shoman, M.A. Wahby Shalaby, A 3D-convolutional neural network framework with ensemble learning techniques for multi-modal emotion recognition, Egypt. Informatics J. (2020). https://doi.org/10.1016/j.eij.2020.07.005.

[17] A. Al-Nafjan, M. Hosny, A. Al-Wabil, Y. Al-Ohali, Classification of Human Emotions from Electroencephalogram (EEG) Signal using Deep Neural Network, Int. J. Adv. Comput. Sci. Appl. 8 (2017) 419–425. https://doi.org/10.14569/ijacsa.2017.080955.

[18] H.A. Gonzalez, S. Muzaffar, J. Yoo, I.M. Elfadel, BioCNN: A Hardware Inference Engine for EEG-Based Emotion Detection, IEEE Access. 8 (2020) 140896–140914. https://doi.org/10.1109/ACCESS.2020.3012900.

[19] H. Jebelli, M. Mahdi Khalili, S. Lee, A Continuously Updated, Computationally Efficient Stress Recognition Framework Using Electroencephalogram (EEG) by Applying Online Multitask Learning Algorithms (OMTL), IEEE J. Biomed. Heal. Informatics. 23 (2019) 1928–1939. https://doi.org/10.1109/JBHI.2018.2870963.

[20] A. Raheel, M. Majid, S.M. Anwar, DEAR-MULSEMEDIA: Dataset for emotion analysis and

recognition in response to multiple sensorial media, Inf. Fusion. 65 (2021) 37–49. https://doi.org/10.1016/j.inffus.2020.08.007.

[21] A.R. Aslam, T. Iqbal, M. Aftab, W. Saadeh, M.A. Bin Altaf, A10.13uJ/classification 2-channel Deep Neural Network-based SoC for Emotion Detection of Autistic Children, Proc. Cust. Integr. Circuits Conf. 2020-March (2020) 8–11. https://doi.org/10.1109/CICC48029.2020.9075952.

[22] H. Ullah, M. Uzair, A. Mahmood, M. Ullah, S.D. Khan, F.A. Cheikh, Internal Emotion Classification Using EEG Signal with Sparse Discriminative Ensemble, IEEE Access. 7 (2019) 40144–40153. https://doi.org/10.1109/ACCESS.2019.2904400.

[23] X. Xu, Y. Zhang, M. Tang, H. Gu, S. Yan, J. Yang, Emotion recognition based on double tree complex wavelet transform and machine learning in internet of things, IEEE Access. 7 (2019) 154114–154120. https://doi.org/10.1109/ACCESS.2019.2948884.

[24] S. Gannouni, A. Aledaily, K. Belwafi, H. Aboalsamh, Adaptive Emotion Detection Using the Valence-Arousal-Dominance Model and EEG Brain Rhythmic Activity Changes in Relevant Brain Lobes, IEEE Access. 8 (2020) 67444–67455. https://doi.org/10.1109/ACCESS.2020.2986504.

[25] T.B. ALAKUŞ, İ. TÜRKOĞLU, Feature Selection with Sequential Forward Selection Algorithm from Emotion Estimation based on EEG Signals, Sak. Univ. J. Sci. 23 (2019) 1096–1105. https://doi.org/10.16984/saufenbilder.501799.

[26] M.S. Özerdem, H. Polat, Emotion recognition based on EEG features in movie clips with channel selection, Brain Informatics. 4 (2017) 241–252. https://doi.org/10.1007/s40708-017-0069-3.

[27] A. Chatterjee, U. Gupta, M.K. Chinnakotla, R. Srikanth, M. Galley, P. Agrawal, Understanding Emotions in Text Using Deep Learning and Big Data, Comput. Human Behav. 93 (2019) 309–317. https://doi.org/10.1016/j.chb.2018.12.029.

[28] R. Ramirez, Z. Vamvakousis, Detecting emotion from EEG signals using the Emotive Epoc device, Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 7670 LNAI (2012) 175–184. https://doi.org/10.1007/978-3-642-35139-6_17.

[29] D.J. Hemanth, EEG signal based Modified Kohonen Neural Networks for Classification of Human Mental Emotions, J. Artif. Intell. Syst. 2 (2020) 1–13. https://doi.org/10.33969/ais.2020.21001.

[30] P. Ekman, An Argument for Basic Emotions, Cogn. Emot. 6 (1992) 169–200. https://doi.org/10.1080/02699939208411068.

[31] P.J. Lang, The emotion probe: studies of motivation and attention., Am. Psychol. 50 (1995) 372.

[32] M.Z. Soroush, K. Maghooli, S.K. Setarehdan, A.M. Nasrabadi, A review on EEG signals based emotion recognition, Int. Clin. Neurosci. J. 4 (2017) 118.

[33] L. Shu, J. Xie, M. Yang, Z. Li, Z. Li, D. Liao, X. Xu, X. Yang, A review of emotion recognition using physiological signals, Sensors (Switzerland). 18 (2018). https://doi.org/10.3390/s18072074.

[34] Y.J. Liu, M. Yu, G. Zhao, J. Song, Y. Ge, Y. Shi, Real-time movie-induced discrete emotion recognition from EEG signals, IEEE Trans. Affect. Comput. 9 (2018) 550–562. https://doi.org/10.1109/TAFFC.2017.2660485.

[35] A. Mehrabian, Comparison of the PAD and PANAS as models for describing emotions and for differentiating anxiety from depression, J. Psychopathol. Behav. Assess. 19 (1997) 331–357. https://doi.org/10.1007/BF02229025.

[36] M.M. Bradley, P.J. Lang, International affective digitized sounds (IADS): Stimuli, instruction manual and affective ratings (Tech. Rep. No. B-2), Gainesville, FL Cent. Res. Psychophysiology, Univ. Florida. (1999).

[37] P.J. Lang, M.M. Bradley, B.N. Cuthbert, International affective picture system (IAPS): Technical manual and affective ratings, NIMH Cent. Study Emot. Atten. (1997) 39–58.

[38] D. Raval, M. Sakle, A Literature review on Emotion Recognition System Using Various Facial Expression, 17 (2015) 326–329.

[39] M. Islam, T. Ahmed, S.S. Mostafa, M.S.U. Yusuf, M. Ahmad, Human emotion recognition using frequency & statistical measures of EEG signal, 2013 Int. Conf. Informatics, Electron. Vision, ICIEV 2013. (2013). https://doi.org/10.1109/ICIEV.2013.6572658.

[40] K.K. Parhi, M. Ayinala, Low-complexity Welch power spectral density computation, IEEE Trans. Circuits Syst. I Regul. Pap. 61 (2013) 172–182.

[41] L.W. Power, Spectral Density Computation, 61 (2014) 172–182.

[42] C.W.N.F.C.W. Fadzal, W. Mansor, L.Y. Khuan, N.B. Mohamad, Z. Mahmoodin, S. Mohamad, U.T. Mara, S. Alam, Welch Power Spectral Density of EEG Signal Generated from Dyslexic Children, (2014) 560–562.

[43] F. El-Hawary, T. Richards, A systolic computer architecture for spectrum analysis, in: Proc. Ocean., IEEE, 1989: pp. 1061–1065.

[44] T.-H. Yu, C.-H. Yang, D. Cabric, D. Markovic, A 7.4-mW 200-MS/s wideband spectrum sensing digital baseband processor for cognitive radios, IEEE J. Solid-State Circuits. 47 (2012) 2235–2245.

[45] Y. Park, L. Luo, K.K. Parhi, T. Netoff, Seizure prediction with spectral power of EEG using cost-sensitive support vector machines, Epilepsia. 52 (2011) 1761–1770.

[46] R.J. Shakshi, Brain wave classification and feature extraction of EEG signal by using FFT on lab view, Int. Res. J. Eng. Technol. 3 (2016) 1208–1212.

[47] H.J. Nussbaumer, The fast Fourier transform, in: Fast Fourier Transform Convolution Algorithms, Springer, 1981: pp. 80–111.

[48] W. Ting, Y. Guo-zheng, Y. Bang-hua, S. Hong, EEG feature extraction based on wavelet packet decomposition for brain computer interface, Meas. J. Int. Meas. Confed. 41 (2008) 618–625. https://doi.org/10.1016/j.measurement.2007.07.007.

[49] Z. Ye, B. Wu, A. Sadeghian, Current Signature Analysis of Induction Motor Mechanical Faults by Wavelet Packet Decomposition, IEEE Trans. Ind. Electron. 50 (2003) 1217–1228. https://doi.org/10.1109/TIE.2003.819682.

[50] R. Venkatesan, B. Li, Convolutional neural networks in visual computing: a concise guide, CRC Press, 2017.

[51] O. Ahmed, A. Brifcani, Gene Expression Classification Based on Deep Learning, in: 2019 4th Sci. Int. Conf. Najaf, IEEE, 2019: pp. 145–149.

[52] C.C. Aggarwal, Neural networks and deep learning, Springer. 10 (2018) 973–978.

[53] A.K. Jaiswal, P. Tiwari, S. Kumar, D. Gupta, A. Khanna, J.J.P.C. Rodrigues, Identifying pneumonia in chest X-rays: A deep learning approach, Measurement. 145 (2019) 511–518.

[54] S. Khan, H. Rahmani, S.A.A. Shah, M. Bennamoun, A guide to convolutional neural networks for computer vision, Synth. Lect. Comput. Vis. 8 (2018) 1–207.

[55] M.S. Mazloom, F. Rezaei, A. Hemmati-Sarapardeh, M.M. Husein, S. Zendehboudi, A. Bemani, Artificial intelligence based methods for asphaltenes adsorption by nanocomposites: Application of group method of data handling, least squares support vector machine, and artificial neural networks, Nanomaterials. 10 (2020) 890.

[56] J.-F. Toubeau, J. Bottieau, F. Vallée, Z. De Grève, Improved day-ahead predictions of load and renewable generation by optimally exploiting multi-scale dependencies, in: 2017 IEEE Innov. Smart Grid Technol., IEEE, 2017: pp. 1–5.

[57] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Networks. 61 (2015) 85–117.

[58] A.P. James, Deep Learning Classifiers with Memristive Networks, Springer, 2020.

[59] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F.E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing. 234 (2017) 11–26.

[60] Z. Akkus, A. Galimzianova, A. Hoogi, D.L. Rubin, B.J. Erickson, Deep learning for brain MRI segmentation: state of the art and future directions, J. Digit. Imaging. 30 (2017) 449–459.

[61] S. Kiranyaz, T. Ince, O. Abdeljaber, O. Avci, M. Gabbouj, 1-d convolutional neural networks for signal processing applications, in: ICASSP 2019-2019 IEEE Int. Conf. Acoust. Speech Signal Process., IEEE, 2019: pp. 8360–8364.

[62] J. Xu, B. Wang, J. Li, C. Hu, J. Pan, Deep learning application based on embedded GPU, in: 2017 First Int. Conf. Electron. Instrum. Inf. Syst., IEEE, 2017: pp. 1–4.

[63] Y. Ren, S. Li, C. Chen, C.-C.J. Kuo, A coarse-to-fine indoor layout estimation (cfile) method, in: Asian Conf. Comput. Vis., Springer, 2016: pp. 36–51.

[64] M. Yani, Application of transfer learning using convolutional neural network method for early detection of terry's nail, in: J. Phys. Conf. Ser., IOP Publishing, 2019: p. 12052.

[65] R. Yamashita, M. Nishio, R.K.G. Do, K. Togashi, Convolutional neural networks: an overview and application in radiology, Insights Imaging. 9 (2018) 611–629. https://doi.org/10.1007/s13244-018-

0639-9.

[66] W. Rawat, Z. Wang, Deep convolutional neural networks for image classification: A comprehensive review, Neural Comput. 29 (2017) 2352–2449.

[67] S. Toraman, Automatic recognition of preictal and interictal EEG signals using 1D-capsule networks, Comput. Electr. Eng. 91 (2021) 107033.

[68] T. Kim, J. Lee, J. Nam, Sample-level cnn architectures for music auto-tagging using raw waveforms, in: 2018 IEEE Int. Conf. Acoust. Speech Signal Process., IEEE, 2018: pp. 366–370.

[69] S. Abdoli, P. Cardinal, A.L. Koerich, End-to-end environmental sound classification using a 1D convolutional neural network, Expert Syst. Appl. 136 (2019) 252–263.

[70] R.A. Hamad, L. Yang, W.L. Woo, B. Wei, Joint Learning of Temporal Models to Handle Imbalanced Data for Human Activity Recognition, Appl. Sci. 10 (2020) 5293.

[71] A. Akbari Asanjan, T. Yang, K. Hsu, S. Sorooshian, J. Lin, Q. Peng, Short-term precipitation forecast based on the PERSIANN system and LSTM recurrent neural networks, J. Geophys. Res. Atmos. 123 (2018) 12–543.

[72] Y.-T. Tsai, Y.-R. Zeng, Y.-S. Chang, Air pollution forecasting using RNN with LSTM, in: 2018 IEEE 16th Intl Conf Dependable, Auton. Secur. Comput. 16th Intl Conf Pervasive Intell. Comput. 4th Intl Conf Big Data Intell. Comput. Cyber Sci. Technol. Congr. (DASC/PiCom/DataCom/CyberSciTech, IEEE, 2018: pp. 1074–1079.

[73] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, S. Khudanpur, Recurrent neural network based language model, in: Elev. Annu. Conf. Int. Speech Commun. Assoc., 2010.

[74] A. Graves, A. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: 2013 IEEE Int. Conf. Acoust. Speech Signal Process., Ieee, 2013: pp. 6645–6649.

[75] N. Kalchbrenner, P. Blunsom, Recurrent continuous translation models, in: Proc. 2013 Conf. Empir. Methods Nat. Lang. Process., 2013: pp. 1700–1709.

[76] W. Zaremba, I. Sutskever, O. Vinyals, Recurrent neural network regularization, ArXiv Prepr. ArXiv1409.2329. (2014).

[77] L. Tai, M. Liu, Deep-learning in mobile robotics-from perception to control systems: A survey on why and why not, ArXiv Prepr. ArXiv1612.07139. (2016).

[78] M. Sundermeyer, R. Schlüter, H. Ney, LSTM neural networks for language modeling, in: Thirteen. Annu. Conf. Int. Speech Commun. Assoc., 2012.

[79] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780.

[80] G. Csurka, D. Larlus, F. Perronnin, F. Meylan, What is a good evaluation measure for semantic segmentation?., in: BMVC, 2013: pp. 10–5244.

[81] E. Fernandez-Moral, R. Martins, D. Wolf, P. Rives, A new metric for evaluating semantic segmentation: leveraging global and contour accuracy, in: 2018 Ieee Intell. Veh. Symp., IEEE, 2018: pp. 1051–1056.

[82] W. Zhu, N. Zeng, N. Wang, Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS implementations, NESUG Proc. Heal. Care Life Sci. Balt. Maryl. 19 (2010) 67.

[83] F.B. Jada, A.M. Aibinu, A.J. Onumanyi, Performance Metrics for Image Segmentation Techniques: A Review, No. Oct. (2015) 344–348.

[84] H.B. Wong, G.H. Lim, Measures of diagnostic accuracy: sensitivity, specificity, PPV and NPV, Proc. Singapore Healthc. 20 (2011) 316–318.

[85] I. Vasilev, D. Slater, G. Spacagna, P. Roelants, V. Zocca, Python Deep Learning: Exploring deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow, Packt Publishing Ltd, 2019.

[86] J. Moolayil, J. Moolayil, S. John, Learn Keras for Deep Neural Networks, Springer, 2019.

[87] E. Smistad, T.L. Falch, M. Bozorgi, A.C. Elster, F. Lindseth, Medical image segmentation on GPUs–A comprehensive review, Med. Image Anal. 20 (2015) 1–18.

[88] H.L. Khor, S.-C. Liew, J.M. Zain, A review on parallel medical image processing on GPU, in: 2015 4th Int. Conf. Softw. Eng. Comput. Syst., IEEE, 2015: pp. 45–48.

[89] T. Kalaiselvi, P. Sriramakrishnan, K. Somasundaram, Survey of using GPU CUDA programming model in medical image analysis, Informatics Med. Unlocked. 9 (2017) 133–144.

[90] T. Carneiro, R.V.M. Da Nóbrega, T. Nepomuceno, G.-B. Bian, V.H.C. De Albuquerque, P.P. Reboucas Filho, Performance analysis of google colaboratory as a tool for accelerating deep learning applications, IEEE Access. 6 (2018) 61677–61685.

[91] I.M. Rooney, J.R. Buck, Spatial power spectral density estimation using a Welch coprime sensor array processor, J. Acoust. Soc. Am. 145 (2019) 2350–2362.

[92] P. Heckbert, Fourier transforms and the fast fourier transform (fft) algorithm, Comput. Graph. (ACM). 2 (1995) 15–463.

[93] W. Ting, Y. Guo-Zheng, Y. Bang-Hua, S. Hong, EEG feature extraction based on wavelet packet decomposition for brain computer interface, Measurement. 41 (2008) 618–625.

[94] H. Ocak, K.A. Loparo, F.M. Discenzo, Online tracking of bearing wear using wavelet packet decomposition and probabilistic modeling: A method for bearing prognostics, J. Sound Vib. 302 (2007) 951–961.

[95] O. Yildirim, U.B. Baloglu, R.-S. Tan, E.J. Ciaccio, U.R. Acharya, A new approach for arrhythmia classification using deep coded features and LSTM networks, Comput. Methods Programs Biomed. 176 (2019) 121–133.

[96] J.H. Tan, Y. Hagiwara, W. Pang, I. Lim, S.L. Oh, M. Adam, R. San Tan, M. Chen, U.R. Acharya, Application of stacked convolutional and long short-term memory network for accurate identification of CAD ECG signals, Comput. Biol. Med. 94 (2018) 19–26.

[97] P. Tarnowski, M. Kołodziej, A. Majkowski, R.J. Rak, Combined analysis of GSR and EEG signals for emotion recognition, 2018 Int. Interdiscip. PhD Work. IIPhDW 2018. (2018) 137–141. https://doi.org/10.1109/IIPHDW.2018.8388342.

[98] A.Q.-X. Ang, Y.Q. Yeong, W. Wee, Emotion classification from EEG signals using time-frequency-DWT features and ANN, J. Comput. Commun. 5 (2017) 75–79.

[99] S. Tripathi, S. Acharya, R.D. Sharma, S. Mittal, S. Bhattacharya, Using Deep and Convolutional Neural Networks for Accurate Emotion Classification on DEAP Dataset., in: Twenty-Ninth IAAI Conf., 2017.

[100] T.B. Alakus, I. Turkoglu, Emotion recognition with deep learning using GAMEEMO data set, Electron. Lett. (2020).

[101] E.S. Salama, R.A. El-Khoribi, M.E. Shoman, M.A.W. Shalaby, A 3D-convolutional neural network framework with ensemble learning techniques for multi-modal emotion recognition, Egypt. Informatics J. (2020).

[102] F.P. George, I.M. Shaikat, P.S. Ferdawoos, M.Z. Parvez, J. Uddin, Recognition of emotional states using EEG signals based on time-frequency analysis and SVM classifier., Int. J. Electr. Comput. Eng. 9 (2019).

APPENDICES

APPENDIX- 1: THE PYTHON MAIN CODE FOR THE 1D CNN MODEL USING THE SF METHOD

```
# Import libraries
import os
import numpy as np
import scipy.io
from keras.models import Model, Input, Sequential
from keras.activations import relu
import matplotlib.pyplot as plt
from keras.optimizers import Adam
from sklearn.metrics import accuracy_score, precision_score, recall_score,f
1_score, confusion_matrix
from keras.layers import Activation, BatchNormalization, Conv1D, Dense, Max
Pooling1D, Flatten, Dropout
import glob
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
# Data Handling: Load Game 1 .CSV
filePath= '/content/drive/MyDrive/Awf files/all channels G1'
read_files= glob.glob(os.path.join(filePath,"*.csv"))
np_array=[]
for files in read_files:
    filesData= pd.read_csv(files, header=0)
    np_array.append(filesData)
merge_value=np.vstack(np_array)
df1=pd.DataFrame(merge_value)
df1.columns=['AF3', 'AF4', 'F3','F4','F7','F8','FC5','FC6','O1','O2','P7','
P8','T7','T8']
print ('df1 =',df1.shape)
# Data Handling: Load Game 2 .CSV
filePath2= '/content/drive/MyDrive/Awf files/AllChannels G2'
read_files2= glob.glob(os.path.join(filePath2,"*.csv"))
np_array2=[]
for files in read_files2:
    filesData2= pd.read_csv(files, header=0)
    np_array2.append(filesData2)
merge_value=np.vstack(np_array2)
df2=pd.DataFrame(merge_value)
df2.columns=['AF3', 'AF4', 'F3','F4','F7','F8','FC5','FC6','O1','O2','P7','
P8','T7','T8']
```

```
# Data Handling: Load Game 3 .CSV
filePath3= '/content/drive/MyDrive/Awf files/All channels G3'
read_files3= glob.glob(os.path.join(filePath3,"*.csv"))
np_array3=[]
for files in read_files3:
    filesData3= pd.read_csv(files, header=0)
    np_array3.append(filesData3)
merge_value=np.vstack(np_array3)
df3=pd.DataFrame(merge_value)
df3.columns=['AF3', 'AF4', 'F3','F4','F7','F8','FC5','FC6','O1','O2','P7','
P8','T7','T8']
print ('df3=',df3.shape)
# Data Handling: Load Game 4 .CSV
filePath4= '/content/drive/MyDrive/Awf files/All chanels G4'
read_files4= glob.glob(os.path.join(filePath4,"*.csv"))
np_array4=[]
for files in read_files4:
    filesData4= pd.read_csv(files, header=0)
    np_array4.append(filesData4)
merge_value=np.vstack(np_array4)
df4=pd.DataFrame(merge_value)
df4.columns=['AF3', 'AF4', 'F3','F4','F7','F8','FC5','FC6','O1','O2','P7','
P8','T7','T8']
print ('df4 =',df4.shape)
# collect all 4 Games' data in one array
dataFG=np.append(df3,df4,axis=0)
dataFG=np.append(df2,dataFG,axis=0)
dataFG=np.append(df1,dataFG,axis=0)
dd=pd.DataFrame(dataFG,columns=['AF3', 'AF4', 'F3','F4','F7','F8','FC5','FC
6','O1','O2','P7','P8','T7','T8'])
print('dataFG=',dataFG.shape)
# feature extraction method (Statistical feature)
x_std=np.array(np.std(dataFG, axis=1))
x_avg=np.array(np.average(dataFG, axis=1))
x_min=np.array(np.min(dataFG, axis=1))
x_max=np.array(np.max(dataFG, axis=1))
x_var=np.array(np.var(dataFG, axis=1))
x_nanmead=np.array(np.nanmedian(dataFG, axis=1))
# append all feature at one array
dfStat=np.append(x_std,x_avg ,axis=0)
dfStat=np.append(dfStat,x_min ,axis=0)
dfStat=np.append(dfStat,x_max ,axis=0)
dfStat=np.append(dfStat,x_var ,axis=0)
dfStat=np.append(dfStat,x nanmead ,axis=0)
```

```
dfStat=np.reshape(dfStat, (4284224, 6))
print (dfStat.shape)
dd=pd.DataFrame(dfStat,columns=['std', 'Avg', 'Min','Max','Variance','NanMe
adian'])
print (dfStat.shape)
# normalize the data
normalized = preprocessing.StandardScaler().fit(dfStat)
X_scaled = normalized.transform(dfStat)
print('X_scaled=',X_scaled.shape)
# load labels
filePath= '/content/drive/MyDrive/Awf files/G1'
read_files= glob.glob(os.path.join(filePath,"*.csv"))
np_array=[]
for files in read_files:
    filesData= pd.read_csv(files, header=0)
    y_G1 = filesData
    np_array.append(y_G1)
merge_value=np.vstack(np_array)
yG1=pd.DataFrame(merge_value)
filePath2= '/content/drive/MyDrive/Awf files/G2'
read_files2= glob.glob(os.path.join(filePath2,"*.csv"))
np_array2=[]
for files in read_files2:
    filesData2= pd.read_csv(files, header=0)
    y_G2 = filesData2
    np_array2.append(y_G2)
merge_value2=np.vstack(np_array2)
yG2=pd.DataFrame(merge_value2)
filePath3= '/content/drive/MyDrive/Awf files/G3'
read_files3= glob.glob(os.path.join(filePath3,"*.csv"))
np_array3=[]
for files in read_files3:
    filesData3= pd.read_csv(files, header=0)
    y_G3 = filesData3
    np_array3.append(y_G3)
merge_value3=np.vstack(np_array3)
yG3=pd.DataFrame(merge_value3)
y=yAllCh
print(y)
```

```
filePath4= '/content/drive/MyDrive/Awf files/G4'
read_files4= glob.glob(os.path.join(filePath4,"*.csv"))
np_array4=[]
for files in read_files4:
    filesData4= pd.read_csv(files, header=0)
    y_G4 = filesData4
    np_array4.append(y_G4)
merge_value4=np.vstack(np_array4)
yG4=pd.DataFrame(merge_value4)
yAllCh=np.append(yG3,yG4,axis=0)
yAllCh=np.append(yG2,yAllCh,axis=0)
yAllCh=np.append(yG1,yAllCh,axis=0)
y=yAllCh
print(y)
x= X_scaled
# Split dataset into training set and test set
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, rand
om_state=42,shuffle=True)
#Convert the x_train and x_test into numpy array
x_train=np.array(x_train)
x_test=np.array(x_test)
x_train = np.reshape(x_train, (x_train.shape[0],x.shape[1],1))
x_test = np.reshape(x_test, (x_test.shape[0],x.shape[1],1))
model = Sequential()
model.add(Conv1D(filters=16, kernel_size=3, strides=1, input_shape=(6, 1), ac
tivation='relu', padding='same'))
model.add(Conv1D(filters=16, kernel_size=3, strides=1, activation='relu', pad
ding='same'))
model.add(MaxPooling1D(pool_size=2, strides=2 ))
model.add(BatchNormalization())
model.add(Flatten())
model.add(Dense(50, activation='relu' ))
model.add(Dropout(0.2))
model.add(Dense(4))
model.add(Activation('softmax'))
print(model.summary())
# Compiling the CNN
opt=keras.optimizers.Adam( learning_rate=0.001)
model.compile(loss='sparse_categorical_crossentropy', optimizer=opt, metrics=
['accuracy'])
# Fitting to the training set
history = model.fit(x_train,y_train,epochs=50,batch_size=42,validation_data=(
x_test, y_test))
```

APPENDIX- 2: TESTING PROCESS FOR THE 1D CNN MODEL USING SF

```
ModelLoss, ModelAccuracy = model.evaluate(x_test, y_test)
print('Test Loss is {}'.format(ModelLoss))
print('Test Accuracy is {}'.format(ModelAccuracy ))
from matplotlib import pyplot

# plot loss during training
pyplot.subplot(211)
pyplot.title('Loss')
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
# plot accuracy during training
pyplot.subplot(212)
pyplot.title('Accuracy')
pyplot.plot(history.history['accuracy'], label='train')
pyplot.plot(history.history['val_accuracy'], label='test')
pyplot.legend()
pyplot.show()
from sklearn.metrics import confusion_matrix

#Confusion_Matrix for model Analysis
ytest_pred_cnn = []
for rows in model.predict(x_test):
    ytest_pred_cnn.append(np.argmax(rows))

from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,ytest_pred_cnn)
import seaborn as sns

ax= plt.subplot()
sns.heatmap(cm, annot=True, ax = ax); #annot=True to annotate cells

# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
ax.xaxis.set_ticklabels(['HAPV', 'HANV','LAPV', 'LANV']); ax.yaxis.set_tickla
bels(['HAPV', 'HANV','LAPV', 'LANV']);
```

CURRICULUM VITAE

Awf Abdulrahman

[redacted]