

FINE GRAINED EMOTION RECOGNITION FROM EEG SIGNAL

A Thesis Submitted in Partial Fulfillment of the Requirements for the
Degree of Bachelor of Science in Computer Science and Engineering of the
University of Asia Pacific

Submitted By

Samina Yasmin

ID: 17101113

Mahmudul Hasan

ID: 17101120

Rokhshana-Nishat-Anzum

ID: 17101137

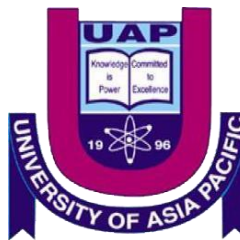
-

Supervised By

Tanmoy Sarkar Pias

Lecturer

Department of Computer Science and Engineering
University of Asia Pacific



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY OF ASIA PACIFIC

June 2021

DECLARATION

We, hereby, declare that the work presented in this Thesis is the outcome of the investigation performed by us under the supervision of Tanmoy Sarkar Pias, Lecturer, Department of Computer Science & Engineering, University of Asia Pacific. We also declare that no part of this Thesis and thereof has been or is being submitted elsewhere for the award of any degree or Diploma.

Countersigned



.....

(Tanmoy Sarkar Pias)

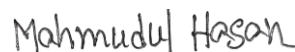
Signature



.....

(Samina Yasmin)

Supervisor



.....

(Mahmudul Hasan)



.....

(Rokhshana-Nishat-Anzum)

Candidates

CERTIFICATE OF APPROVAL

We hereby recommend that the thesis prepared by Samina Yasmin (17101113), Mahmudul Hasan (17101120), Rokhshana-Nishat-Anzum (17101137) entitled “FINE GRAINED EMOTION RECOGNITION FROM EEG SIGNAL” is accepted as fulfilling the part of the requirements for degree of Bachelor of Science in Computer Science and Engineering.



Tanmoy sarkar Pias

Lecturer (On study leave)

Department of Computer Science and Engineering

University of Asia Pacific (UAP)

Chairman of the Committee
(Supervisor)



S M Rafiuddin Rifat

Lecturer

Department of Computer Science and Engineering

University of Asia Pacific (UAP)

Member of the Committee
(External)



Md. Nahiyan Uddin

Lecturer

Department of Computer Science and Engineering

University of Asia Pacific (UAP)

Member of the Committee
(External)

MD. Rajibul Islam

Assistant Professor

Department of Computer Science and Engineering

University of Asia Pacific (UAP)

Head of the Department

ACKNOWLEDGEMENTS

First of all, we would like to declare the greatness of Almighty Allah and express our gratitude for the opportunity to study for a Bachelor's degree in Computer Science and Engineering at a reputed educational institution like the University of Asia Pacific. We are also grateful to the Department of Computer Science and Engineering at our varsity for providing research opportunities on important topics in an academic manner.

We are delighted to have completed our thesis work and the man who deserves the most thanks for the completion of this work is our respected supervisor Tanmoy Sarker Pias. Although he was always busy with a lot of valuable work, in the midst of this busy time we even got him whenever we felt his need. He not only gave us the time when we faced any problem but also gave us the right instructions and advice. His guidance and comments helped a lot in completing this thesis report.

We are grateful to our teachers whose inspiring speeches and teaching inspired us to work in this field. We are also grateful to those who have provided the knowledge and resources to make this thesis a reality.

ABSTRACT

The process of identifying human emotion is known as emotion recognition. Emotions are mental states originating in the human brain and this is closely related to the activities of the nervous system. Electroencephalogram (EEG) is a well-established approach to record neuron activities which is reliable for emotion recognition compared to the non-physiological clues. Emotion recognition is most often used for neuroscience, psychology, cognitive science, computer science, artificial intelligence, and brain-computer interfaces. So far, there have been reports of various searches for active patterns involving different emotions. However, with time, their stability is still not fully investigated. Hence, this paper presents convolutional neural network (CNN) models working on the DEAP dataset and it contains emotional states which are arousal, valence, dominance, and liking where each state is rated from 1-9. We are able to get better accuracy from what has been done about binary classification so far with the DEAP dataset for valence and arousal. Our binary models achieved 96.63% and 96.17% accuracy respectively for valence and arousal. Only four emotion spaces are found with binary classification whereas 8-class classification is worked more precisely with sixty-four emotion spaces. This paper attempts to work with 8-class classification also and get a promising result with the accuracy of 93.83% and 93.79% respectively for valence and arousal. For both cases, Fast Fourier Transformation (FFT) has been used as the feature extraction method and all the four classification models are created under 1D-CNN using the same architecture. The design and application of CNN models depend on CNN's ability to automatically detect new emotion-related EEG features in future work. Furthermore, our study is a testament to the robust neural network classifiers for brain signals that will be even surpassing traditional theoretical learning techniques.

TABLE OF CONTENTS

ABSTRACT	iv
ACKNOWLEDGEMENT	v
TABLE OF CONTENTS	vi-vii
LIST OF FIGURES	viii
LIST OF TABLES	ix

Chapter 1: Introduction	1
--------------------------------	----------

1.1 Motivation.....	2
1.2 Research Problem	2
1.3 Research Statement.....	3
1.4 Contribution	3

Chapter 2: Literature Review	4
-------------------------------------	----------

Chapter 3: Background Study	8
------------------------------------	----------

3.1 Basic Structure of Human Brain.....	8
3.1.1 Brainstem	9
3.1.2 Cerebellum.....	9
3.1.3 Cerebrum	9
3.1.3.1 Frontal Lobe.....	10
3.1.3.2 Parietal Lobe... ..	10
3.1.3.3 Occipital Lobe	10
3.1.3.4 Temporal Lobe.....	11
3.2 Brainwave	11
3.2.1 Delta Wave... ..	11
3.2.2 Theta Wave... ..	12
3.2.3 Alpha Wave	12
3.2.4 Beta Wave.....	12
3.2.5 Gamma Wave	13
3.3 EEG and EEG data acquisition	13

3.4 Machine Learning for EEG.....	14
3.4.1 Long short-term memory (LSTM).....	14
3.4.2 Support Vector Machine (SVM).....	14
3.4.3 Convolutional Neural Network (CNN)	15
3.4.4 Deep Neural Network (DNN).....	15
Chapter 4: Dataset	16
Chapter 5: Methodology	21
4.1 Preprocessing	21
4.2 Feature Extraction.....	21
Chapter 6: Experiment	26
6.1 Experiment-1.....	26
6.2 Experiment-2.....	27
6.3 Experiment-3.....	28
6.4 Experiment-4.....	29
6.5 Experiment-5.....	29
6.6 Experiment-6.....	30
6.7 Binary-Class Classification.....	32
6.8 8-Class Classification.....	33
Chapter 7: Result Analysis	34
7.1 Binary-Class Classification Result	34
7.2 8-Class Classification Result	35
Chapter 8: Conclusion and Future Work	38
Bibliography	39
Appendix A	43

LIST OF FIGURE

3.1 Main parts of the brain...	8
3.2 Various parts of cerebrum.....	10
3.3 Delta wave	11
3.4 Theta wave	12
3.5 Alpha wave	12
3.6 Beta wave.....	12
3.7 Gamma wave	13
3.8 EEG data acquisition procedure.....	13
4.1 Raw EEG signal for one subject	20
5.1 Workflow of EEG data analysis method...	21
5.2 Feature extraction method.....	22
6.1 Model architecture-1	27
6.2 Model architecture-2.....	28
6.3 Model architecture-3	28
6.4 Model architecture-4	29
6.5 Model architecture-5	30
6.6 Model architecture-6.....	31
6.7 Circumplex model for binary-class emotions	32
6.8 Circumplex model for 8-class emotions	33
7.1 Confusion matrix for binary-class classification...	34
7.2 Accuracy-loss graph for binary-class...	35
7.3 Confusion matrix for 8-class classification...	35
7.4 Accuracy-loss graph for 8-class...	36

LIST OF TABLE

2.1 Overview of literature review	5
4.1 Synopsis of the DEAP dataset... ..	16
4.2 EEG channels used in DEAP dataset... ..	17
4.3 Other channels used in DEAP dataset	18
4.4 EEG Emotiv electrodes	18
4.5 Data orientation of each subject.....	19
6.1 Model architecture-1 (experiment for valence)	26
6.2 Model architecture-2 (experiment for valence)	27
6.3 Model architecture-3 (experiment for valence)	28
6.4 Model architecture-4 (experiment for valence)	29
6.5 Model architecture-5 (experiment for valence)	30
6.6 Model architecture-6 (experiment for valence)	31
7.1 Binary-class classification report... ..	34
7.2 8-class classification report.....	36
7.3 Result summary	37
7.4 Result comparison for binary-class.....	37

Chapter 1

Introduction

Emotions are a significant factor of human knowledge, behavior and communication. It is a neural impulse that moves an organism to action, prompting automatic reactive behavior that has been adapted through evolution as a survival mechanism to meet a survival need [1]. Emotion recognition is the process of identifying human emotions. Emotions can be divided into two categories: Positive and Negative [2]. While positive emotions are needed for good health, negative emotions can cause mental health problems, such as depression, stress, and anxiety [3]. Emotions are known to arise from the central and peripheral nervous system and cause temporal movement due to the synchronized execution of neurons [4]. They are expressed by internal signals as well as external expressions like facial expression, speech, body posture, eye blinking, and skin response. If only external expressions are used for emotion measurement, incorrect results may be obtained, because in many cases external expressions can be controlled. That's why internal signals get priority. Electroencephalogram (EEG), Temperature (T), Electrocardiogram (ECG), Electromyogram (EMG), Galvanic Skin Response (GSR), Respiration (RSP) are examples of such internal signals. Excluding others, EEG is selected for its non-invasive, fast and low-cost characteristics, as compared with other physiological signals.

Neuropsychological measurement of electrical activity in the brain is known as Electroencephalography (EEG) and it is recorded by electrodes. Electrodes are normally placed on the scalp, or in special cases, subdurally and in the cerebral cortex. EEG estimates voltage fluctuations that result due to ionic flows inside the neurons of the cerebrum. EEG provides an excellent temporal resolution, even though it has a poor spatial resolution and requires many sensors placed on the scalp. Pure EEG signal is a composition of sub-bands: theta, alpha, beta and gamma [17]. Individual sub-bands are associated with individual relevant physical activities. For instance, Theta wave refers to REM sleep, deep and raw emotions, and cognitive processing. A drowsy state indicates the Alpha waves. It also becomes the cause of relaxation and calmness. Beta points to the conscious state during the thought process. The Gamma waves are available when trying to perceive two different senses at the same time as sound and sight [7].

Yet now, there comes out of extensive research using machine learning to identify states of emotion with EEG. Machine learning-based theoretical methods are often effectively used for emotion classification, while the disadvantage of such methods is that researchers

have to spend a lot of effort to detect and design different emotion-related features from the resulting noisy signals and these features are time-consuming calculations. Various methods of EEG feature extraction have been explored in recent years, although the Fast Fourier Transformation (FFT), Short-time Fourier Transform (STFT) and Discrete Wavelet Transformation (DWT) etc. are most effectively used. Then machine learning and deep learning models are applied to the extracted features so that they are trained for recognizing emotions. Support Vector Machine (SVM), Linear and nonlinear regression, Decision trees and K-nearest neighbor (KNN) are examples of mostly used machine learning model architecture. Convolution Neural Networks (CNN), Long-Short Term Memory (LSTM), Convolution Long-Short Term Memory (CLSTM) are popular model architectures from deep learning for this field. There are so many EEG datasets that are publically available and some researchers use their own dataset. Some famous publically available datasets are DEAP (A Database for Emotion Analysis using Physiological Signals), SEED (SJTU Emotion EEG Dataset) and MAHNOB (MAHNOB-HCI-Tagging database) etc. Among these, the DEAP dataset has been used for this research work.

1.1 Motivation

Rapidly advancing technology is bringing immense facilities into our lives. The emotion recognition concept can play a very important role in the advancement of human-computer interaction technology. Identifying the sensations of mentally challenged or autistic people who are disabled to express themselves. Monitoring the psychological condition of a person who has suffered a serious injury or illness. Determining fatigue while driving and advance warning. ATMs will deny dispensing money when a person feels scared to withdraw. Identify whether e-learners understand their lessons correctly or how much learning strategies need to change. Identify whether an offender is lying in his statement. Since consumers are influenced by emotions when they make a purchase decision, emotion recognition could be a field from which business analysts, content creators and advertisers can take full advantage. We have been motivated to do research on emotion recognition from the idea that emotion recognition can be applied properly in the mentioned fields and the results obtained from it can bring something beneficial for human civilization.

1.2 Research Problem

As we have worked with the DEAP dataset, we have found main data along with four emotion labels (arousal, valence, dominance and liking). From the DEAP data, using main data and only two emotion labels (arousal and valence), it is possible to recognize human

emotion. Each emotion label is divided into two equal segments and a total of four emotional states are created. The classification that follows this method is known as binary classification. The outcome of this binary classification is four compound emotional states where multiple real-life emotions exist in each state. Almost all studies in this field have been done so far following the binary classification method. However, it is possible to further develop the accuracy that they have earned. And this binary classification is unable to detect real-life emotions precisely as it uses only four emotional states.

1.3 Research Statement

In order to increase the accuracy of binary classification, the DEAP dataset needs to be extracted in a good way, then have to apply the models of good architecture on the extracted feature which is able to provide better accuracy from now on. And to recognize emotions more accurately, a different approach is needed where a large number of emotional states may be used.

1.4 Contribution

With this study we have tried to solve the mentioned problems. Significant contributions are:

- We have brought the best accuracy using the binary classification which is the conventional emotion recognition method and for this, we have used a very simple and light 1D-CNN model architecture and FFT as a feature extraction method.
- We have also proposed the eight-class emotion classification method which is able to recognize emotion much more accurately and we have also found a satisfactory classification accuracy for it. For this eight-class classification approach, we get $8*8 = 64$ emotional states that help to recognize emotion more precisely.

Chapter 2

Literature Review

There has been a lot of research on publicly available datasets (DEAP, SEED, MAHNOB, LUMED) for emotion recognition. But some have tried to recognize emotions on their own datasets. Since the DEAP dataset is used in this paper, it would be better to review the contributions and the recent studies using the DEAP dataset. All the research works mentioned in Table 2.1 have been done using the binary class classification approach where arousal and valence have been used as emotion labels.

Rahul Sharmaa Et al. [4] achieved an accuracy of 82.01% with a ten-fold cross-validation technique using Long short-term memory (LSTM) by decomposing with DWT. They worked on only two dimensions, namely arousal and valence, and with four quadrants respectively LaLv (low arousal low valence), HaLv (high arousal low valence), LaHv (low arousal high valence) and HaHv (high arousal high valence).

Zhongke Gao Et al. [9] proposed a model named Channel-fused dense convolutional network (CDCN), consisting of a 1D convolution layer and 1D dense layer. For pre-extracting, they used differential entropy (DE) and worked on four emotions. Their model applied the SEED dataset and DEAP dataset and obtained an accuracy of 90.6% and 92.58% respectively.

Yuling Luo Et al. [10] demonstrated their best performance with Spiking Neural Networks (SNNs) using three pre-extracting methods: DWT, Variance and FFT. They gained their best results with the use of variance, both on SEED and DEAP datasets. The emotion states of arousal, valence, dominance and liking were classified with accuracies of 74%, 78%, 80% and 86.27% for the DEAP dataset, as well as an overall accuracy of 96.67% for the SEED dataset.

Eman A. Abdel-Ghaffar Et al. [13] proposed a two-dimensional emotion model named Log-Euclidean Riemannian Metric (LERM) using Symmetric Positive Definite manifold (SPD). They received an accuracy of 88.3% for HVHA, 84.38% for LVHA, 79.3% for LVLA, and 78.4% for HVLA. The average accuracy for valence was $74.6\% \pm 3.9$, and $72.6\% \pm 6.7$ for arousal.

Fei Wang Et al. [5] used the Electrode-frequency distribution maps (EFDMs) model for classifying and short-time Fourier transform (STFT) for feature extraction. Gradient weighted class Activation mapping (Grad-CAM) is used in their research to obtain a better understanding of their selected features. When they applied their model on the SEED dataset, they obtained 90.59% for accuracy, and on the DEAP dataset, they obtained an accuracy of 82.84%. However, they only worked with valence labels with three classes: negative, neutral, positive.

Kit Hwa Cheah Et al. [8] used two types of CNN models: single-path CNN and two-path CNN model using 4 folds of cross-validation. However, they did not use any manual pre-extraction methods, and instead, worked with only valence and arousal. Each emotional state was divided into three classes. Single-path CNN received an accuracy of 97.59% and 98.4% for 3-class valence and arousal, while two-path CNN received 98.75% and 97.58% for 3-class valence and arousal. Yucel Cimta Et al. [11] did not use any manual pre-extraction methods, and instead, depended on the CNN Deep Learning method. Three datasets were used in their research work: DEAP, SEED and LUMED. In the SEED dataset when studying two classes of emotions, an accuracy of 86.56% was obtained, and 78.34% was obtained for three classes of emotions. When applied to the DEAP dataset, they received 72.81% accuracy for two emotion states: valence and arousal. Guolu Cao Et al. [12] created a CNN model, utilizing Principal Component Analysis (PCA) as the pre-extracting technique. They worked with two classes for arousal and valence with an accuracy of $84.3 \pm 4.0\%$ and $81.2 \pm 3.0\%$ respectively. Xiaolong Zhong Et al. [14] concentrated on the physiological forms of brain waves. Their method was efficient in recognizing emotions, especially in beta and gamma waves. 2D SE_CNN was applied to the DEAP and the MAHNOB-HCI datasets. In their study, the DEAP dataset received an accuracy of 66.23% for valence and 68.50% for arousal, while the MAHNOB-HCI dataset obtained 70.25% for valence and 73.27% for arousal. Yucel Cimtay Et al. [15] used the InceptionResnetV2 CNN model, following a hybrid fusion strategy on the DEAP dataset as well as LUMED-2 datasets with facial expressions and galvanic skin response (GSR). They achieved a maximum of 91.5% accuracy on the DEAP dataset with arousal and valence.

Table 2.1: Overview of literature review

No	Research	Year	Feature extraction	Modeling technique	Performance
1	Rahul Sharmaa Et al. [4]	2020	DWT	LSTM	82.01%

2	Divya Acharya Et al. [3]	2020	FFT	LSTM	89.83%
3	Zhongke Gao Et al. [9]	2020	DE	CDCN	92.58%
4	Yulong Luo Et al. [10]	2020	Variance	SNN	Valence: 78% Arousal: 74%
5	Yucel Cimtay Et al. [11]	2020	Raw Data	CNN	DEAP-72.81%
6	Eman A. Et al. [13]	2020	SPD	LERM	Valence: 74.6% \pm 3.9, Arousal: 72.6% \pm 6.7
7	Xiaolong Zhong Et al. [14]	2020	Down sampling	CNN	Valence : 66.23% Arousal : 68.50%
8	Yucel Cimtay Et al. [15]	2020	Raw Data	CNN	91.5%
9	Guolu Cao Et al. [12]	2019	PCA	CNN	Valance: 81.2 \pm 3.0%* Arousal: 84.3 \pm 4.0%
10	Soheil Rayatdoost Et al. [1]	2018	HOC, PSD, DE, HOS	RF	Valence: 60.86% Arousal: 58.08%
11	Ningjie Liu Et al. [16]	2018	LFCC	KNN, ResNets	Valence: 90.39% Arousal: 89.06%
12	Abeer Al- Nafjan Et al. [22]	2017	PSD	DNN	Valence: 82% Arousal: 82%
13	Samarth Tripathi Et al. [24]	2017	GD	CNN	Valence: 81.41% Arousal: 73.36%
14	Xiang Li Eta1. [21]	2016	CWT	C-RNN	Valence: 72.06% Arousal: 74.12%

15	Wei-Long Zheng Et al. [25]	2016	DE	GELM	69.67%
16	Wei Liu Et al. [23]	2016	PSD,DE	BDAE	Valence: 85.20% Arousal: 80.50%
17	Hyun Joong Yoon Et al. [18]	2013	FFT	Bayes classifier	Valence: 70.9%, Arousal: 70.1%
18	Viktor Rozgić Et al. [19]	2013	PCA	SMV	Valence: 76.9% Arousal: 68.4%
19	Xiaowei Zhang Et al. [20]	2013	Sliding 4- sec windows with a 2-sec overlap	Ontology Reasoning BIO- EMOTION	Valence: 75.19% Arousal: 81.74%

From the above discussion, we can see that almost everyone has worked on binary-classification. But the testing accuracy that has been found so far is not enough and it is possible to improve more. We have been able to achieve the best accuracy through our work. Only four emotional spaces can be found with a binary-classification which is unable to accurately recognize real-life emotions. To overcome this lack, we have introduced in our experiments an 8-class classification technique that can recognize a variety of emotions and has achieved satisfactory accuracy in this method as well.

Chapter 3

Background Study

This paper is a testimony to a study which has been worked on emotion recognition and for this, the EEG signal data has been used. For this, we have learned about human brain structure and brain waves as emotions are related to brain waves. Then EEG, how EEG works and EEG data collection procedures have been focused on. How a machine can learn from data and then make decisions on its own by applying that learning to new data, this knowledge has been gathered by us. For this purpose, Machine Learning, Artificial Neural-Network and Deep Learning have been gone through by us.

3.1 Basic Structure of Human Brain

The brain is one of the most important and complex organs in the human body. Although it weighs only 2% of the total human body, this plays a key role in the nervous system. It is covered with a scalp inside the head and billions of neurons are found there.

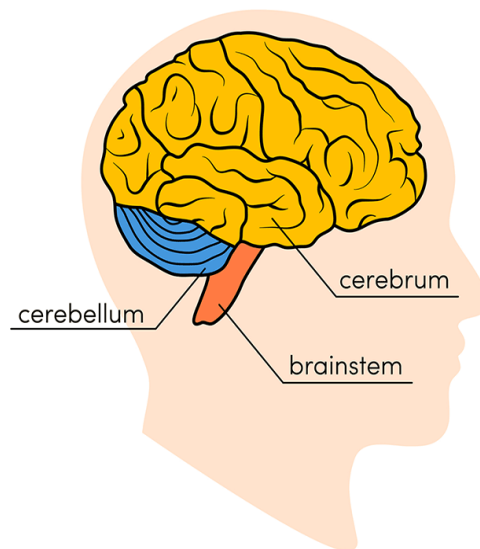


Figure 3.1: Main parts of the brain [31]

These huge numbers of neurons communicate among themselves with the help of trillions of connectivity. This important organ is divided into 3 main parts: Brainstem, Cerebellum and Cerebrum. These are demonstrated in Fig. 3.1.

3.1.1 Brainstem

Brainstem is the interface between the brain and spinal cord which is located under the cerebrum. The brain stem acts as a relay center by connecting the spinal cord between the cerebrum and the cerebellum. Pons, main-brain, medulla-oblongata are the three-part of the Brainstem. This is a composition of white and grey matter. To control the waking and sleeping cycles of the brain it acts as a switch and also regulates the muscle tone. It also monitors many biological functions like oxygen levels, pressure, wheezing, puking, swelling-reflexes, coughing, homeostasis etc.

3.1.2 Cerebellum

Cerebellum is known as the little brain. This is situated at the posterior position of the Brainstem and the inferior position of the Cerebrum. The anterior and the posterior lobes are connected with the help of the vermis at the middle. Its shape is hemispherical and wrinkly. Its main functions are to control motor functions such as balancing, posture and aligning muscle actions. Also Leads to the consistency and efficiency of motor activities like speech, walking and writing.

3.1.3 Cerebrum

The largest part of the brain is known as the cerebrum and it has two identical-looking hemispheres: the left hemisphere and right hemisphere. These hemispheres maintain their connectivity through the corpus callosum fibers. The right part of the body is controlled by the left hemisphere and the left part of the body is controlled by the right hemisphere. Here a grey-colored outer covering called the cerebral cortex is seen and the white core's matter is covered by it. Though corpus callosum fibers transfer data between hemispheres, some functionality in each hemisphere is operated independently. The calculations, writings, speech and cognitions are assigned to the left hemisphere but the imagination, musical aptitude, spatial capacity etc. are controlled by the right hemisphere.

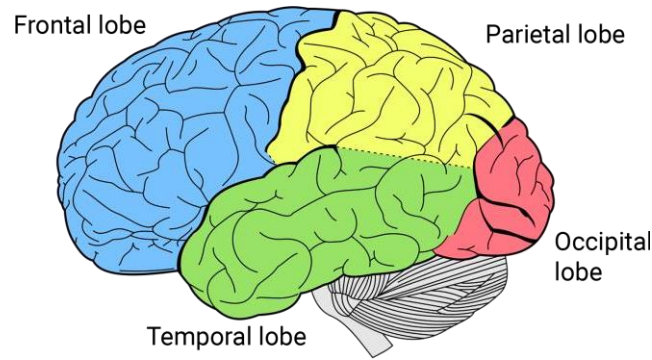


Figure 3.2: Various parts of cerebrum [32]

This largest part of the human brain is important for all emotional activities. The entire area of each hemisphere can be divided into four lobes: temporal lobe, parietal lobe, frontal lobe and occipital lobe. And these are shown in Fig. 3.2.

3.1.3.1 Frontal Lobe

As the name implies, the front part of a hemisphere is the frontal lobe. Speech, planning, intelligence, emotions, judgment, writing, self-awareness and motor abilities are very much related to the frontal lobe.

3.1.3.2 Parietal Lobe

The position of the parietal lobe is on the back of the frontal lobe. This is named because it is close to the parietal bone. The somatosensory cortex is in this parietal lobe and this lobe works for determining sensory information like temperature, the sense of pain and touch.

3.1.3.3 Occipital Lobe

The posterior part of the hemisphere is called the occipital lobe. This lobe contains an important biological receiver named the visual cortex and the visual information that we see by our eyes is received by that receiver. The obtained information is processed for taking decisions about vision, color and perceptions.

3.1.3.4 Temporal Lobe

If we look closely at the hemisphere, we can see that the temporal lobe is situated next to the hemisphere. The auditory cortex is found here and all the auditory information is processed here.

All of the above segments carry a huge number of neurons and they are interconnected. They transmit data in the form of waves through electrochemical pulses. These waves are known as brain waves

3.2 Brainwaves

The central nervous system's electrical pulses are known as brain waves which have repetitive cycle characteristics and analyzable patterns also [26]. Neural information is transmitted for neural oscillation and this transmission is measured in Hz [29]. These pulses may be created from a single neuron or may also be the result of the interaction of multiple neurons. The information is sent in the form of a wave through the nervous system to the target place so that all kinds of activities are done properly. Brain waves vary depending on what kind of information they are carrying [27]. For example, when people are delighted, high frequency is seen in their brain waves, on the contrary when people feel bored, low frequency is seen in brain waves. Brainwaves can be distinguished by frequency range and some important brain waves are:

3.2.1 Delta Wave

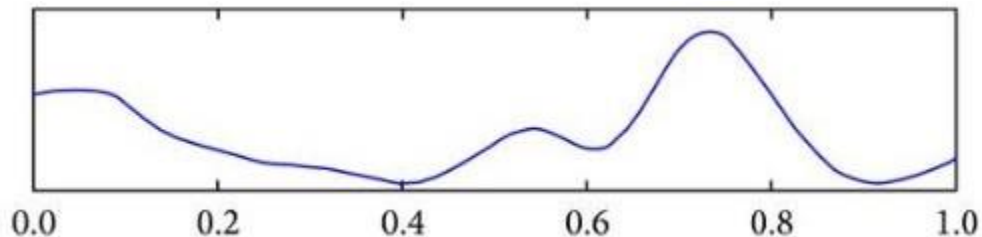


Figure 3.3: Delta wave [30]

Between 0.5 Hz and <3 Hz denotes the Delta waves. Normally it is found when a person is in deep sleep, trance and unconscious [28].

3.2.2 Theta Wave

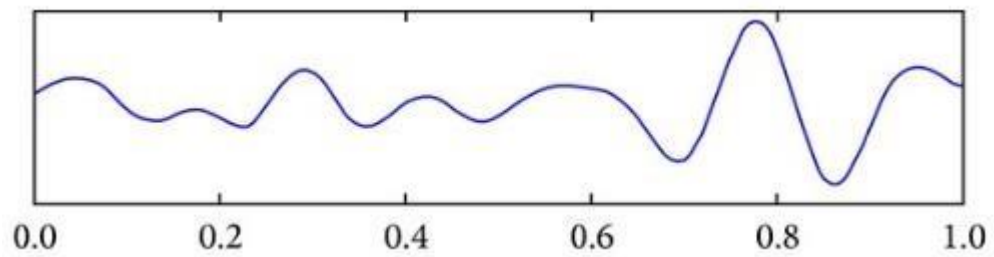


Figure 3.4: Theta wave [30]

Theta waves show a frequency range between 3 Hz to 7 Hz [17]. These waves are found when a person is meditating, daydreaming, fantasizing etc. [28].

3.2.3 Alpha Wave

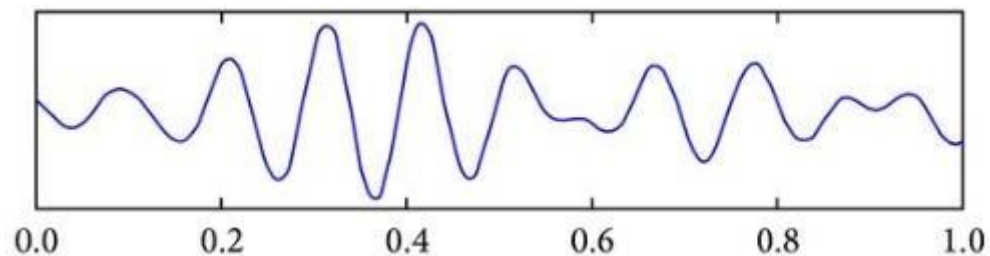


Figure 3.5: Alpha wave [30]

The range of 8 Hz to 13 Hz is allocated for the Alpha brain waves [17]. When a person is in drowsy, calm, relaxation states then the Alpha waves are found [7].

3.2.3 Beta Wave

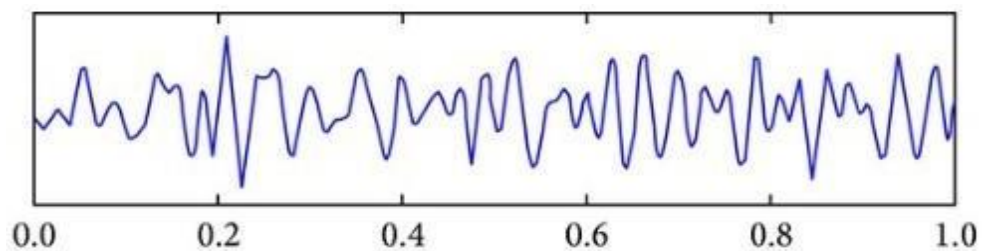


Figure 3.6: Beta wave [30]

The Beta wave occupies the area of 14 Hz to 29 Hz in the frequency range [17]. This points to the conscious state during the thought process [7]. These waves are also seen when a person remains alert or active.

3.2.4 Gamma

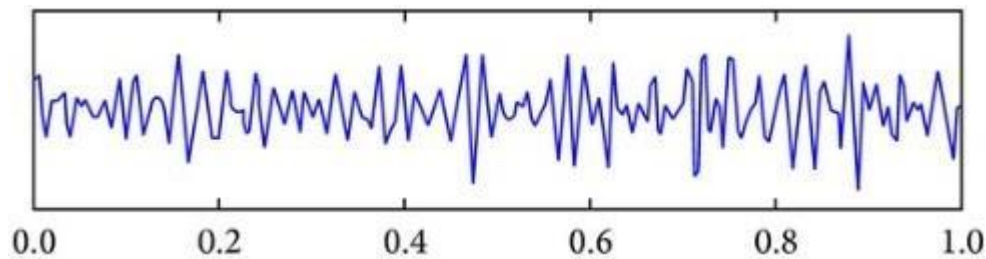


Figure 3.7: Gamma wave [30]

Frequency ranges from 30 Hz to 48 Hz in this area known as the Gamma waves. These waves are available when trying to perceive two different senses at the same time as sound and sight [7].

3.3 EEG and EEG data acquisition

Electroencephalography (EEG) was discovered by Hans Berger in 1929 who was a German psychiatrist. Neuropsychological measurement of electrical activity in the brain is known as Electroencephalography (EEG) and it is recorded by some sensors which are called electrodes.

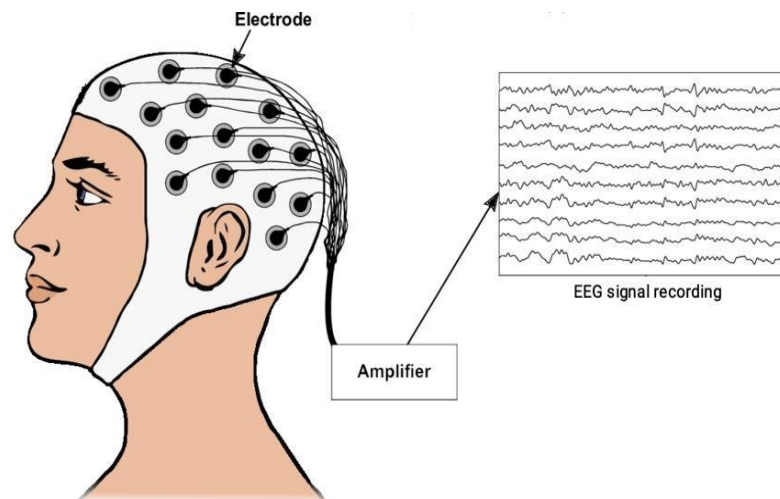


Figure 3.8: EEG data acquisition procedure [33]

When we read about the brain wave, we have known that different types of brain signals come out of our brain for different activities in different situations. The EEG tries to collect and store those brain waves for analysis. The electrodes are placed on the top of the skull so that the electrodes can absorb the brain signals from very close to the brain. Then these absorbed signals are amplified so that they become usable. These amplified signals may contain a lot of artifacts and noise for different reasons. Various artifacts removal and filters are used to purify these signals. Then if anyone wants to extract key features from these signals and work with them then there are so many feature extraction methods to use.

3.4 Machine Learning for EEG

Preprocessed data (artifacts removed and filtered) are used for teaching a machine. For this various machine learning and deep learning, models are available. According to the convenience, different algorithms could be used. Some of the algorithms that have been gone through by us are mentioned below:

3.4.1 Long short-term memory (LSTM)

Long Short Term Memory (LSTM) is a deep learning model architecture and normally this is a Recurrent Neural Network(RNN) architecture. To learn long-term information this architecture is so capable. LSTM was first introduced by Schmidhuber and Hochreiter. If any sequence has a large variety, this algorithm will work there tremendously. To decrease the dependency problem in the long term this architecture is designed.

3.4.2 Support Vector Machine (SVM)

Support vector machine is a machine learning classification algorithm. It maintains a line to classify the points. This line is known as a hyperplane and the closest point from this line on both sides is known as vector points. The distance between two side vector points is maintained the highest through this hyperplane. Here the decision boundary is maintained.

3.4.3 Convolutional Neural Network (CNN)

Convolutional Neural Network is an example of a neural network and this is mostly used in computer vision for image processing. Other types of data can be analyzed and classified with this algorithm. CNN can be used on that data which can be represented by a matrix and the entire matrix will be changed after row interchange or column interchange. This algorithm is able to detect patterns from data. In this algorithm convolution layer, pooling layer and fully connected layer are seen. This algorithm can work with data, without feature extraction and from those data can find out the pattern.

3.4.4 Deep Neural Network (DNN)

The deep Neural Network is an example of the Artificial Neural Network. It has several layers along with input layers and output layers. This layer contains nodes that act like human brain neurons and every node is connected to the next layer node. Thus it creates neural networks like the human brain. The connections between any two layers' nodes with specific weighted values are given.

Chapter 4

Dataset

The DEAP dataset is a publicly available multimodal dataset [17] that includes Electroencephalogram (EEG) signals and used for detecting human emotional states. A team of researchers at the Queen Mary University of London developed this dataset.

Table 4.1: Synopsis of the DEAP dataset

Types of dataset	Multimodal dataset
No. of participant	32
No. of EEG channel	32
Data collection method	Showing one-minute long excerpts of music videos
No. of used data collection resource	40 music videos
Sampling rate	128 Hz
Rating values	Continuous scale 1-9
Rating scales	Arousal and Valence

The DEAP dataset is available in two parts, with the first part containing an online self-assessment of 14-16 volunteers based on arousal, valence, and dominance for 120 one-minute music videos. The second part contains the participant ratings, physiological recordings and face video of an experiment where 32 volunteers watched 40 music videos which are the subset of the previously mentioned 120 music videos. Physiological signals and EEG signals were recorded where each participant also rated the videos following the above procedure. Facial expression at the time of watching videos was also recorded from 22 participants. Individual online self-assessment ratings, a list of the used YouTube music videos, the ratings that the participants gave for the videos, all the answers for the questionnaire of the participants before the experiment and participant's frontal face video recordings as well as raw physiological data recordings in BioSemi .bdf format are in the official dataset. Forty experiments for each of the 32 participants are found in the

dataset. For each of the 40 experiments, the label array for each participant contained ratings of arousal, valence, dominance, and liking. For each participant, 8064 physiological/EEG signals data were collected with 40 different channels for each experiment and put into the data array. Among 40 channels/electrodes 32 are EEG channels/electrodes. Tables 4.2 and Table 4.3 below represents 40 channels/electrodes which have been used in this dataset:

Table 4.2: EEG channels used in DEAP dataset

Channels No.	Electrode Name	Channels No.	Electrode Name
1	Fp1	17	Fp2
2	AF3	18	AF4
3	F3	19	Fz
4	F7	20	F4
5	FC5	21	F8
6	FC1	22	FC6
7	C3	23	FC2
8	T7	24	Cz
9	CP5	25	C4
10	CP1	26	T8
11	P3	27	CP6
12	P7	28	CP2
13	PO3	29	P4
14	O1	30	P8
15	Oz	31	PO4
16	Pz	32	O2

Table 4.3: Other channels used in DEAP dataset

Other Channels	Electrode Name
33	hEOG (horizontal EOG, hEOG1 - hEOG2)
34	vEOG (vertical EOG, vEOG1 - vEOG2)
35	zEMG (Zygomaticus Major EMG, zEMG1 - zEMG2)
36	tEMG (Trapezius EMG, tEMG1 - tEMG2)
37	GSR (values from Twente converted to Geneva format (Ohm))
38	Respiration belt
39	Plethysmograph
40	Temperature

Among 32 EEG channels/electrodes, 14 channels are used for collecting emotive data and they are known as emotive channels. The list of these emotive channels/electrodes with their position on the scalp is given in Table 4.4.

Table 4.4: EEG Emotiv electrodes

Brain's part	Serial no.	EEG channel no.	Electrode	Electrode location
Right part	1	2	AF3	Frontal
	2	3	F3	Frontal
	3	4	F7	Frontal
	4	5	FC5	Frontal
	5	8	T7	Temporal
	6	12	P7	Parietal

	7	14	O1	Occipital
Left part	8	18	AF4	Frontal
	9	20	F4	Frontal
	10	21	F8	Frontal
	11	22	FC6	Frontal
	12	26	T8	Temporal
	13	30	P8	Parietal
	14	32	O2	Occipital

Brain data is collected using electrode caps, EEG signal is collected via 512 Hz sampling frequency. After watching the videos all participants rated those according to a 1-9 scale based on valence, arousal and dominance. The duration of every sampled data is 63s. For experiments normally pre-processed data is used where 128Hz downsampling, electrooculogram (EOG) removal, filtering, segmentation and so on have been used. Two versions of preprocessed data are found on the official website. One of them is the figdata_preprocessed_matlab folder processed with Matlab where files are in .mat format and another is the data_preprocessed_python folder processed with Python (numpy) where files are in .dat format. The preprocessed data folder contains 32 files and each file holds the individual data of each of the 32 subjects. The data format is in Table 4.5.

Table 4.5: Data orientation of each subject

Name of array	Shape of array	Contents of array
data	40 x 40 x 8064	video/trial x channel x data
labels	40 x 4	video/trial x label (valence, arousal, dominance, liking)

Both .dat files and .mat files contain data field with shape 40*40*8064 and label field with shape 40*4, where data field shape 40*40*8064 stands for 40 trials, 40 channels and 8064 refers to 63*128. Here sampling time is 63 seconds and the sampling frequency is 128Hz. In label field shape 40*4 indicates 40 experiments and 4 dimensions respectively for valence, arousal, dominance and liking. Signals were recorded according to the international 10-20 system. From the data_preprocessed_python folder, some parts of a file out of 32 preprocessed files are plotted and shown in Fig 4.1.

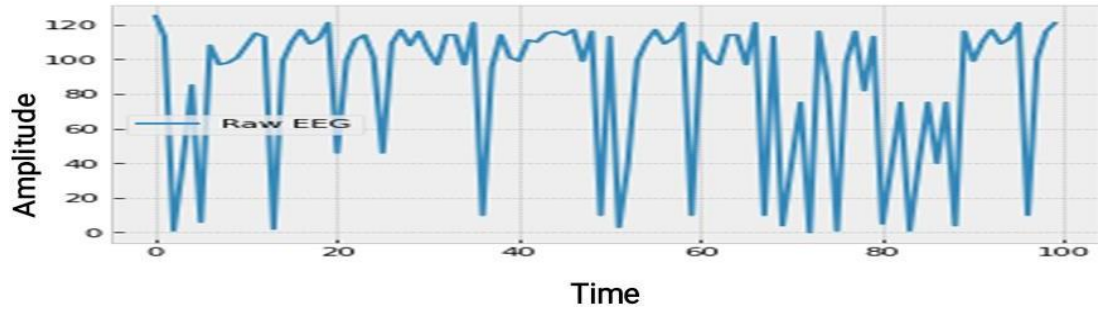


Figure 4.1: Raw EEG signal for one subject

Chapter 5

Methodology

The procedure for EEG Data Analysis is shown below through Fig 5.1. Firstly, the dataset is collected then raw data is cleaned using the various pre-processing techniques. Next, this cleaned raw data is segmented in the feature extraction step. Then those extracted features are used to train a model for getting a better classification result.



Figure 5.1: Workflow of EEG data analysis method

5.1 Preprocessing

The DEAP EEG signal has been recorded with a good instrument and as a result, the chance of artifacts are minimized. The DEAP dataset's EEG signals are downsampled to 128 Hz first, so that the data content is collected in a good way between 0-48 Hz. Then the electromyogram (EMG) and electrooculogram (EOG) has been removed from the downsampled data. A bandpass filter has been applied to separate the delta waves from the analysis process. A blind source separation technique has been used removing eye artifacts. Using Common Average Reference (CAR) the data has been averaged. Each recording data has been partitioned into 60 seconds segments and a pre-trial baseline of 3 seconds has been removed.

5.2 Feature Extraction

In the research field, emotion classification potentiality depends on two factors: feature extraction and classification. Feature extraction reduces the initial dataset by identifying key features of data and later these features are used for classification. Distinguishing property, recognizable measurement, and functional components obtained from a section of a pattern are represented by features. A better classification accuracy comes if extracting

features from a dataset are used instead of the original dataset. By Feature extraction, various advantages could be found like Minimizing the loss of important signals, decreasing the risk of over-fitting, improving the visualization of data, and reducing the implementation complexity.

Three types of features were found [7]:

- **Time-domain features:** used for statistical features.
- **Frequency-domain features:** decomposition of preprocessed signal data into sub-bands.
- **Time-Frequency domain features:** used for non-stationary waveform signals [16].

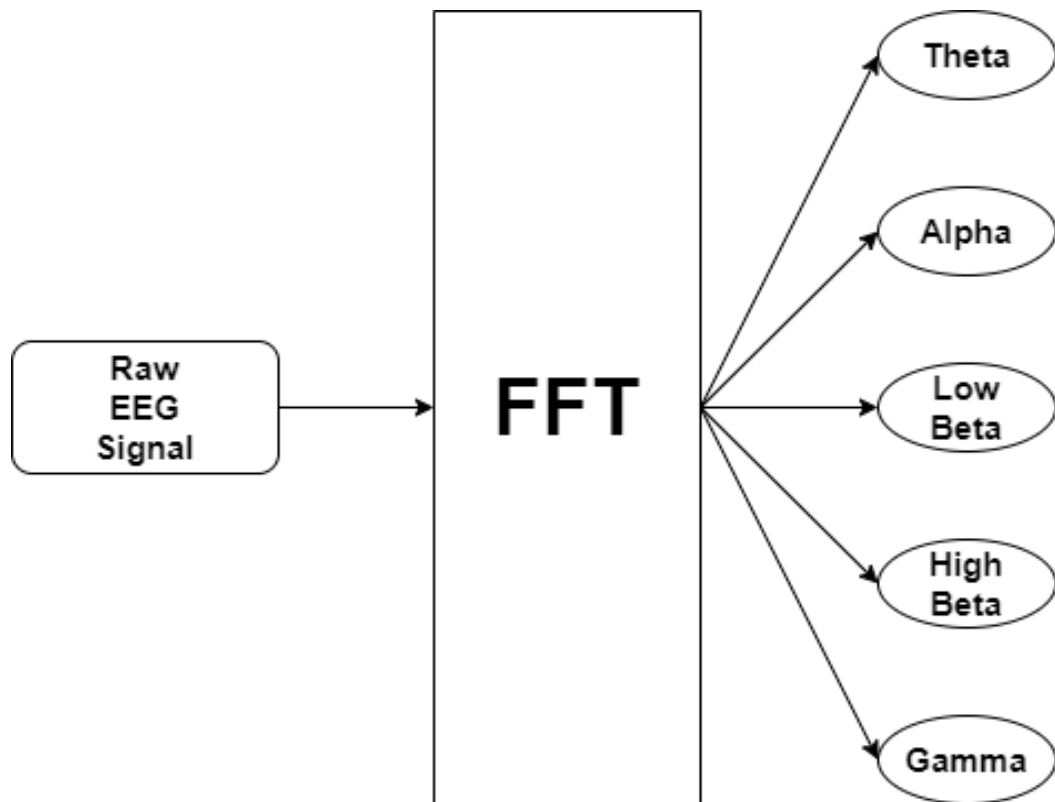


Figure 5.2: Feature extraction method

Frequency-domain features are used for this work Fig 5.2.

Different methods are used for EEG feature extraction, including FFT, Wavelet Transform (WT), Time-Frequency Distribution (TFD), Equivocator methods (EM), Auto-Regressive methods (ARM), etc. From these methods, FFT is ultimately applied to the preprocessed EEG datasets by using a python module named PyEEG, which is formatted in NumPy.

Fourier analysis is commonly used for signal processing to convert time-domain signals into frequency domain signals. Thus, it was used to decompose the EEG signal into frequency components. To compute the Fourier transformation, the most popular algorithm is FFT, which computes the Discrete Fourier Transform (DFT) of a sequence [3].

$$X_k = \sum_{i=0}^{N-1} x_i(n) e^{\frac{-j2\pi i k}{N}} \quad \text{for } k = 0, 1, 2 \dots N-1 \quad (1)$$

In equation (1), $k = 0, 1, 2 \dots N-1$ X_k is the coefficient of discrete Fourier, length available data is N and $x_i(n)$ is the time domain input signal.

The purpose is to extract key features from pre-processed data and use them to classify emotions through a CNN-based deep learning model. The Fast Fourier Transform method is used when employing mathematical tools to extract the EEG features. By using power spectral density (PSD) estimation, the characteristics of the EEG signal are computed. The EEG spectrum of wave characteristics is then divided into four frequency bands. Through the approximate auto-correlation sequence of the Fourier converter, PSD can be counted accurately. When extracting features by using FFT for EEG, there are mainly two types of techniques: Periodogram Method and Welch's method. Calculating PSD with a periodogram is the easiest way. Frequency decomposition is included by Period gram and the modulus squared of the Fourier transform of the signal is expressed as:

$$\tilde{P}_{xx}(f) = \frac{\Delta t}{N} \left| \sum_{i=0}^{N-1} x_i(n) e^{\frac{-j2\pi i k}{N}} \right|^2 \quad (2)$$

In equation (2), t refers to space between the samples, available data length is ' N ', $x_i(n)$ is the time domain input signal and $\tilde{P}_{xx}(f)$ denotes PSD for $x_i(n)$. Welch's method is another PSD assessment strategy that is used to improve the modified periodogram accuracy. This method is established through the use of signals in overlapping windows where for each window a periodogram is calculated and then to calculate the PSD those periodograms are constructed. Supposing signals $x(n)$ have finite length, then the relationship with power spectral density is estimated as:

$$x_i(n) = x(n + iD), \quad n = 0, 1, 2 \dots, M-1 \quad (3)$$

while $i = 0, 1, 2 \dots, L-1$

In equation (3), ' iD ' represents the start of the i^{th} sequence and ' L ' is the length of the formatted data segment. The subsequent outcome periodograms are given:

$$\tilde{P}_{xx}^{(i)}(f) = \frac{1}{MU} \left| \sum_{n=0}^{M-1} x_i(n) w(n) e^{-j2\pi f n} \right|^2$$

In equation (4), 'U' represents the normalization factor of the power in the window function and is expressed such that

$$U = \frac{1}{M} \sum_{n=0}^{M-1} w^2(n), \quad (4)$$

In equation (5), $w(n)$ indicates the window function. The mean of these modified periodograms presents Welch's power spectrum which is considered as:

$$P_{xx}^W = \frac{1}{L} \sum_{i=0}^{L-1} \tilde{P}_{xx}^{(i)}(f) \quad (5)$$

In this paper 14 channels, 5 bands, window size = 256, step size = 16, sample rate = 128 are used for per subject's data in feature extraction.

Here,

- **Channel:** The difference between the electrode and the weighted mean of contiguous electrodes is considered as an EEG channel.
- **Band:** A fixed range of wave frequencies and amplitudes over a time scale is called Frequency Bands in EEG.
- **Windows Size:** The length of a cutout (sliding) of a time sequence of data is known as the window size.
- **Step Size:** During the training period amounts of weights are needed to update, that is called step Size.
- **Sample Rate:** From a non-digital or continuous signal to create a digital or discrete signal how many samples are taken per second, this number is known as the sampling rate.

After applying FFT, the entire dataset is split into training and testing segments following the 7:1 ratio. Then encoding is subsequently applied for labeling to avoid over-fitting. This dataset contains four label columns: arousal, valence, dominance and liking. But only arousal and valence labels are used for categorizing with the help of categorical function. Then normalization has been used to bring the different ranges of data between 0 and 1. Sometimes normalization helps to increase the accuracy of the models. Standard Scalar is

one of the techniques for normalization. Initially, 2D arrays are found in the DEAP dataset but used models need 3D data as input. That's why 2D data is converted into 3D using reshaping.

Chapter 6

Experiment

In this research, we have worked with many 1D-CNN model architectures for the experiments. Since our dataset has the same range of values in four labels and the same type of main data, we have tested all the model architectures on one label. Because if we used the same architecture for the same range and the same type of data, much difference would not be seen among the accuracy. Our goal was to find out a model architecture that would give the best result and then it would be used for other labels. We have started the experiment as planned. We have label values from 1 to 9 in our dataset and we start experimenting by expecting 10 classes for these values. We have tried to increase the accuracy of our models in various ways.

6.1 Experiment-1

In this experiment, model architecture-1 has been used and this naming convention for all the model architecture in this experiment chapter is given by us. Here used model architecture is 1D CNN and the architecture has been unchanged for the whole experiment. Architecture has been mentioned in Fig 6.1. Completing the experiment, seven times the model has been run and each time a different feature extraction method has been used where sample rate, window size, step size have been changed. Only 86.91% accuracy has been found from this experiment and it has not been acceptable to us. All the experiment reports have been mentioned in Table 6.1.

Table 6.1: Model architecture-1 (experiment for valence)

Window Size	Step Size	Sample Rate	Batch Size	Epoch	Train accuracy	Test accuracy
256	16	128	256	579	78.42%	86.91%
512	16	128	256	498	77.89%	80.81%
128	16	128	256	293	43.74%	44.84%
256	32	128	256	553	58.82%	57.69%

256	8	128	256	219	60.70%	69.41%
256	16	256	256	308	44.58%	46.58%
256	16	64	256	423	54.99%	59.07%

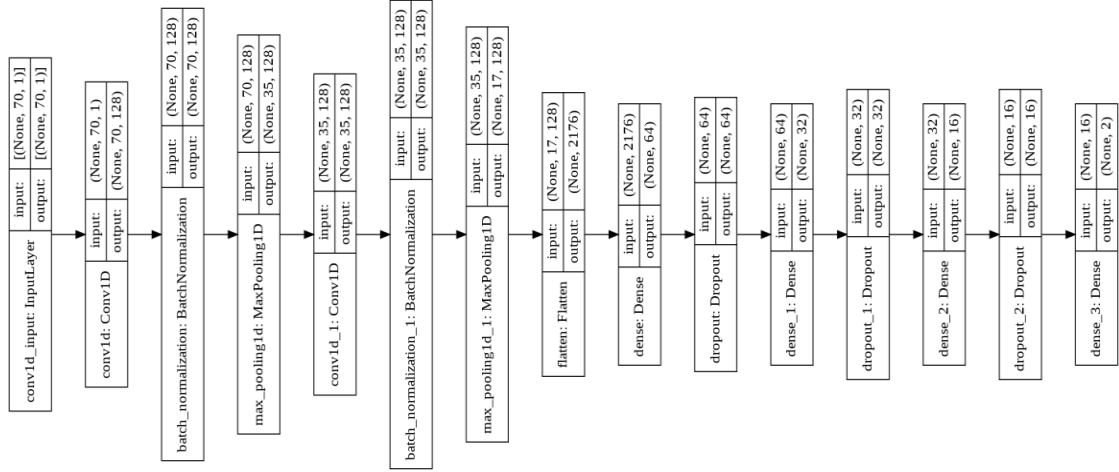


Figure 6.1: Model architecture-1

6.2 Experiment-2

In this experiment, the sample rate, window size, step size remained the same as the previous experiment for feature extraction. Then model architecture-2 has been applied on the extracted features and this model architecture-2 has been demonstrated in Fig 6.2. In this experiment this architecture has been run for only once and has been assumed that this architecture accuracy would not improve according to expectation. And the experiment's details have been shown in Table 6.2.

Table 6.2: Model architecture-2 (experiment for valence)

Window Size	Step Size	Sample Rate	Batch Size	Epoch	Train accuracy	Test accuracy
256	16	128	256	530	84.29%	69.38%

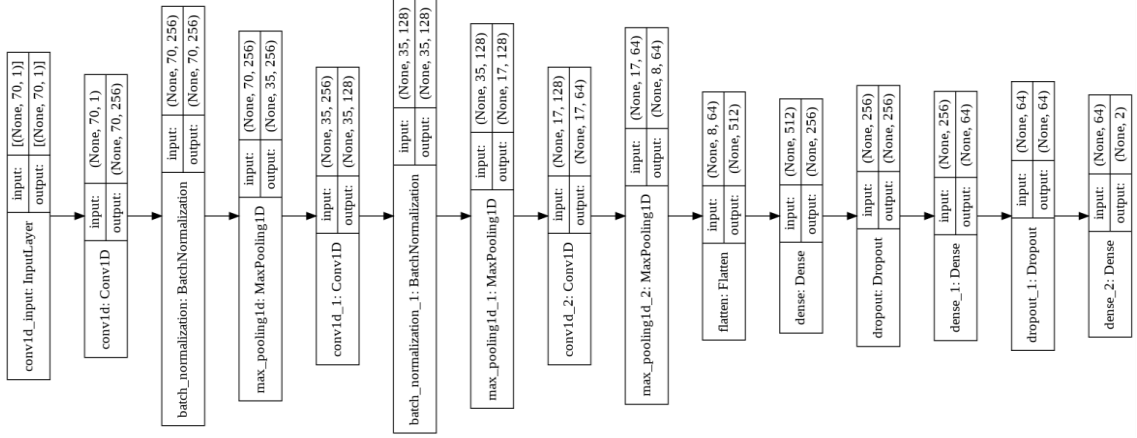


Figure 6.2: Model architecture-2

6.3 Experiment-3

This is another experiment where the expected accuracy has not been found. For this model architecture-3 has been used which has been demonstrated in Fig 6.3. This experiment has not been shown to have better accuracy. A better result has not been shown by this architecture and the result details are given in Table 6.3.

Table 6.3: Model Architecture-3 (experiment for valence)

Window Size	Step Size	Sample Rate	Batch Size	Epoch	Train accuracy	Test accuracy
256	16	128	256	494	66.13%	70.39%

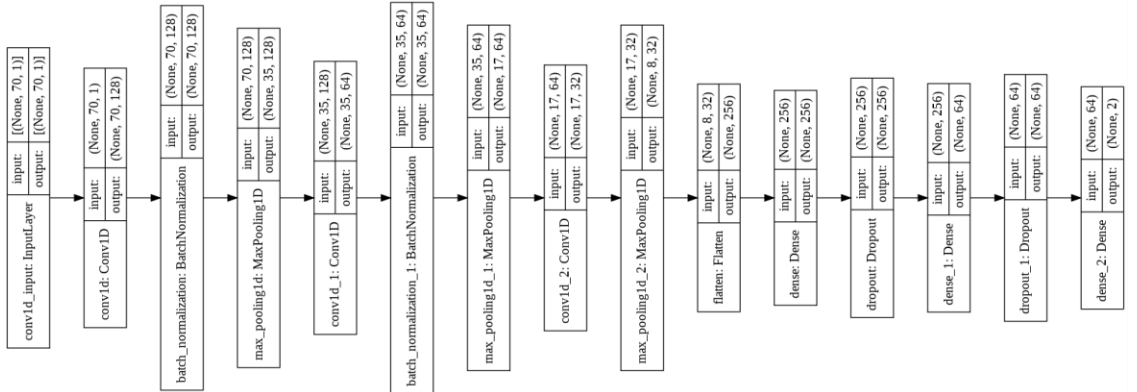


Figure 6.3: Model architecture-3

6.4 Experiment-4

A better performance than experiment-2 and experiment-3 has been found in this experiment. But it has not surpassed the result of experiment-1. The model architecture-4 has been used in this step and the details about model architecture are shown in Fig 6.4. For this experiment a model has been run for three times and each of the time the batch size has been changed but for this any increment of the accuracy has not been seen. All the necessary details are shown in Table 6.4.

Table 6.4: Model architecture-4 (experiment for valence)

Window Size	Step Size	Sample Rate	Batch Size	Epoch	Train accuracy	Test accuracy
256	16	128	256	284	88.25%	86.78%
256	16	128	128	350	88.96%	72.93%
256	16	128	512	372	91.95%	73.79%

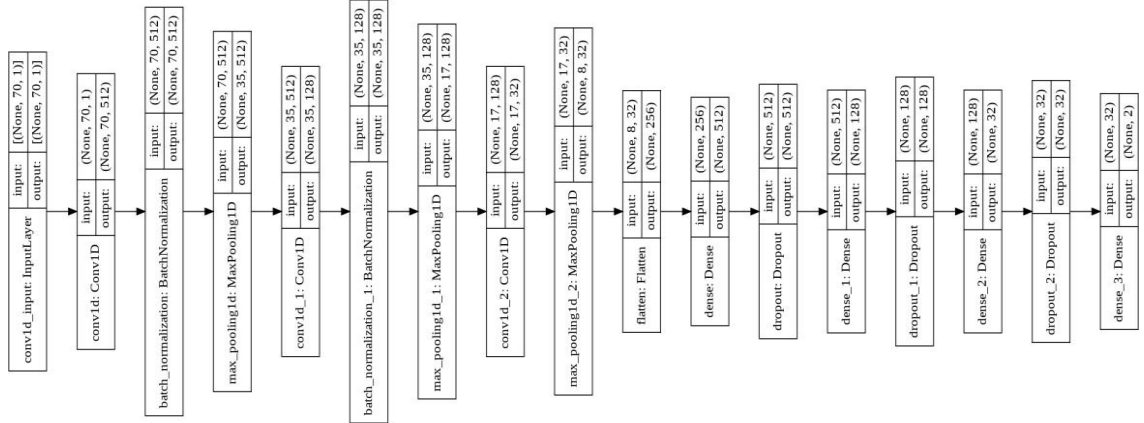


Figure 6.4: Model architecture-4

6.5 Experiment-5

This is the worst model architecture ever used by us. The architecture details have been shown in Fig 6.5. Although it has run the least epoch for each time. All other architectures

have much better accuracy when running such epochs. Due to this dilapidated state of the architecture, it has not been taken further. For this experiment a total of five times have been run models. And all the details of this experiment are displayed in Table 6.5.

Table 6.5: Model architecture-5 (experiment for valence)

Window Size	Step Size	Sample Rate	Batch Size	Epoch	Train accuracy	Test accuracy
256	16	128	256	50	64.99%	65.22%
256	16	128	512	100	64.77%	65.05%
256	16	128	128	100	67.23%	66.75%
512	16	128	256	100	65.90%	66.00%
512	16	128	128	100	67.27%	66.4%

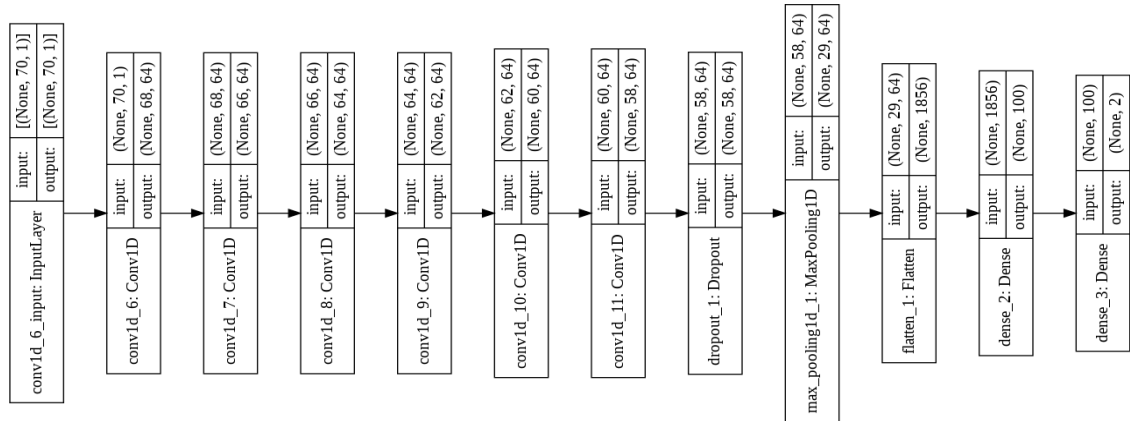


Figure 6.5: Model architecture-5

6.6 Experiment-6

After testing five models of architecture, model number six has satisfied us. This desired architecture is shown in Fig: 6.6. The outcome of this experiment shows a better accuracy than before five experiments and it indicates that it will give better performance for binary classification. Table 6.6 carries the important details of this architecture.

Table 6.6: Model architecture-6 (experiment for valence)

Window Size	Step Size	Sample Rate	Batch Size	Epoch	Train accuracy	Test accuracy
256	16	128	256	319	99.36%	94.02%

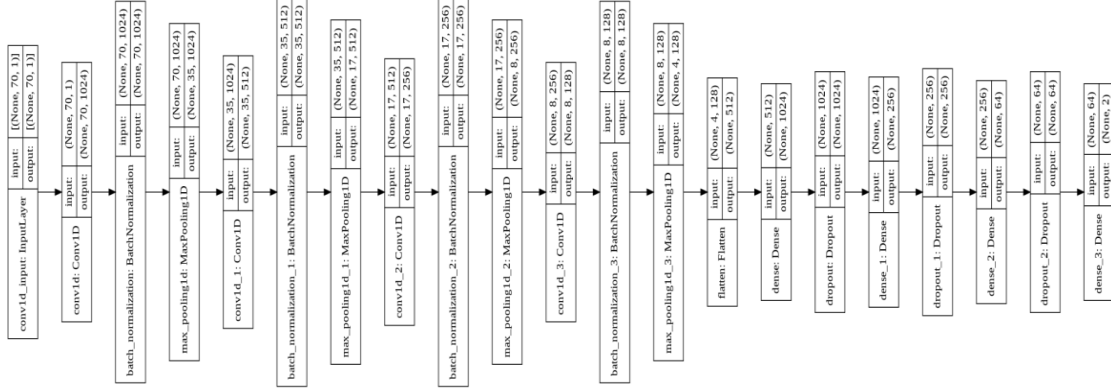


Figure 6.6: Model architecture-6

Looking at all the mentioned experiments, it is understood that it would be better to select model architecture-6 for our work and that's why this model architecture has been selected. This 1D-CNN has been used whose hidden layer can be changed to increase the accuracy. A segment is created with a 1D Convolution layer, a batch normalization layer as well as a 1D max-pooling layer and this segment has been found a total of four times at the starting of the model architecture. The fourth segment's result is converted into a 1D array with the help of flattening. Then three connected layers (dense layers) are applied and after every connected layer a drop-out layer is also given to avoid the overfitting problem. Then the output layer has been constructed including the number of classes and the softmax activation function.

The extracted features contain the main data as well as the emotion state labels. This research strives to better identify emotions with two different approaches. One of them is the conventional binary classification method and another is our proposed eight-class classification method. But in both classification methods, the same 1D-CNN model architecture has been used which has already been selected by us. This architecture is much more simple and light. Total four models are applied for the classification. Two models for the binary classification and the other two models for the eight-class classification, here all the model works on arousal and valence emotional states. For architecture selection, the

label values were divided into 10 classes for testing purposes. But here the same values have been used according to need so that the classifications can be effective and avoid different complexities. The label array contains floating values from 1 to 9 for both valence and arousal. But among all the values, the amount of 9 is very poor. For the convenience of reducing calculation and space-complexity, we converted all the 9 into 8.99. In this case, the difference between 9 and 8.99 is very negligible and it does not create any impact.

6.7 Binary-Class Classification

In a two-dimensional emotion recognition system binary class classification is found as a conventional method. All the values of the label array are divided into two classes where 1-4.99 for one class and 5-8.99 for other classes. Then the binary arousal classification model and the binary valence classification model have been trained on all the data. It provides better results than all the previous work mentioned in the literature review and these results are shown in the performance measurement section.

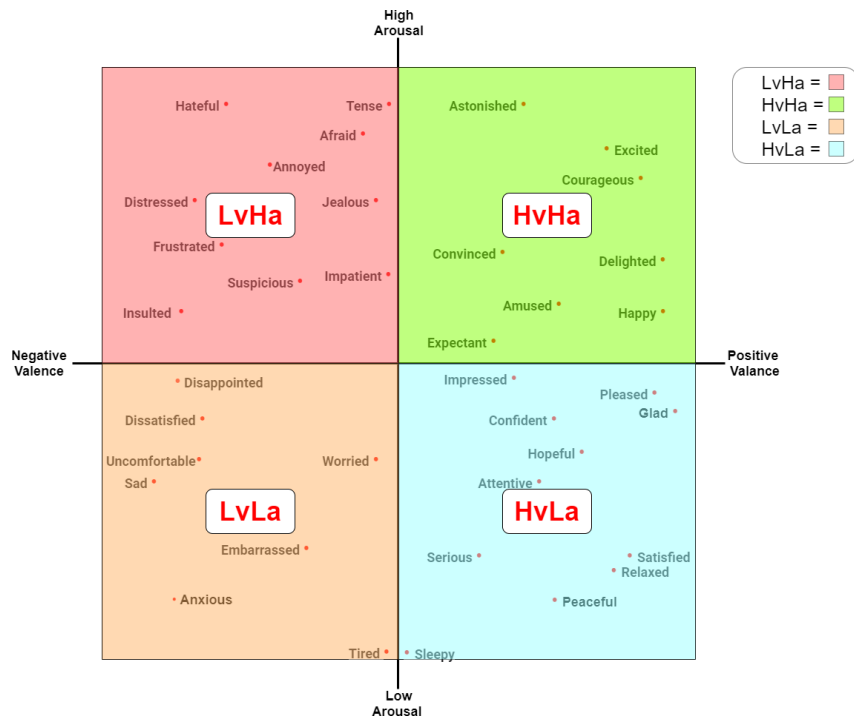


Figure 6.7: Circumplex model for binary-class emotions

Binary classification divides the entire emotion space into four classes and each of the four classes is the combination of multiple real-life emotions. These four compound emotions are expressed by HaHv, LaHv, LaLv and HaLv and these are demonstrated in Fig 6.7.

6.8 8-Class Classification

This technique works precisely to recognize emotion properly using the DEAP datasets values. Many real-life emotions can be recognized using the eight-class classification technique where the binary class classification is able to find out only four compound emotions and each of these compound emotions consists of multiple real-life emotions. Using eight-class classifications those emotions are possible to recognize, a small number of them are mentioned in Fig 6.8.

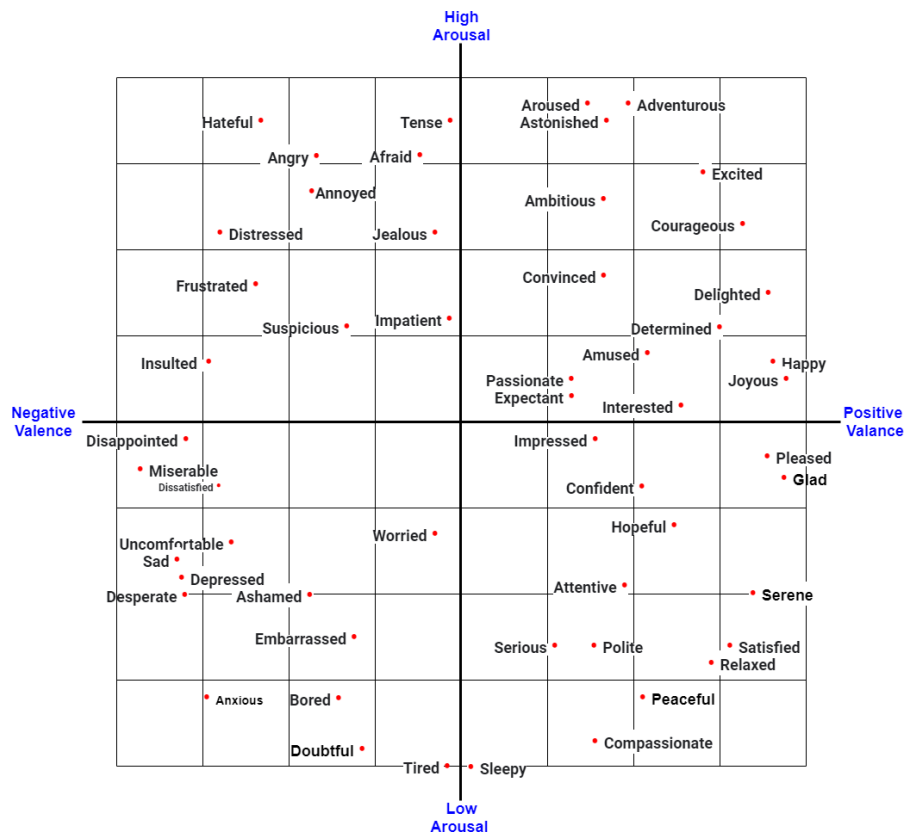


Figure. 6.8: Circumplex model for 8-class emotions

To be successful in this classification technique, arousal and valence states label array values are needed to divide into eight segments. Eight label segmentation have been created maintaining 1 - 1.99, 2 - 2.99, 3 - 3.99, 4 - 4.99, 5 - 5.99, 6 - 6.99, 7 - 7.99, 8 - 8.99 procedure. Then the eight-class arousal classification model and the eight-class valence classification model have been applied to all the data for training. These models have given us satisfactory results also and we have discussed them in the performance measurement section.

Chapter 7

Result Analysis

7.1 Binary-Class Classification Result

The binary classifiers get 96.63% accuracy for arousal and 96.17% accuracy for valence. The binary arousal classifier achieved 99.65% training accuracy and 96.17% test accuracy in 131 epochs and at epoch 126, it provides the best accuracy.

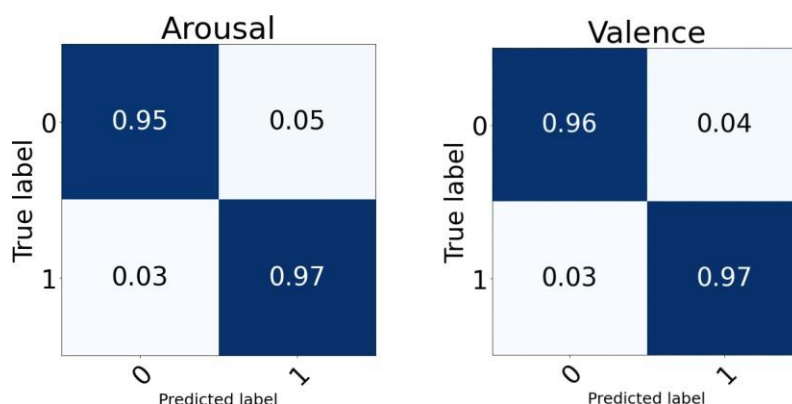


Figure 7.1: Confusion matrix for binary-class classification

On the other hand, the binary valence classifier provides 96.63% test accuracy when the training accuracy is 99.73% after 163 epochs but it shows the best test accuracy at 150th epoch.

Table 7.1: Binary-class classification report

Arousal				Valence			
Class	Precision	Recall	F1-score	Class	Precision	Recall	F1-score
0	0.96	0.95	0.96	0	0.96	0.96	0.96
1	0.96	0.97	0.97	1	0.97	0.97	0.97

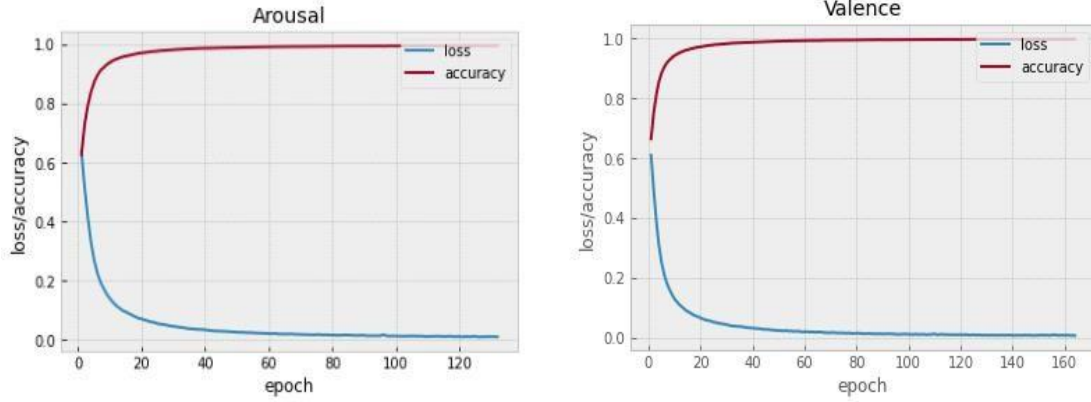


Figure 7.2: Accuracy-loss graph for binary-class

Fig 7.1 demonstrates the normalized confusion matrix of arousal and valence where arousal is divided into two classes: high-arousal and low-arousal. And valence is also divided into two classes: positive valence and negative valence. For arousal and valence in both cases, the condition of true positive and true negative is looking better. Table 7:1 contains the classification report for both arousal and valence which shows the precision, recall and f1-score very clearly. Accuracy and losses graphs for arousal and valence are shown in Fig 7.2.

7.2 8-Class Classification Result

The eight-class arousal classifier gets 98.83% training accuracy after running 151 epochs and gets 93.79% best test accuracy at 126th epoch.

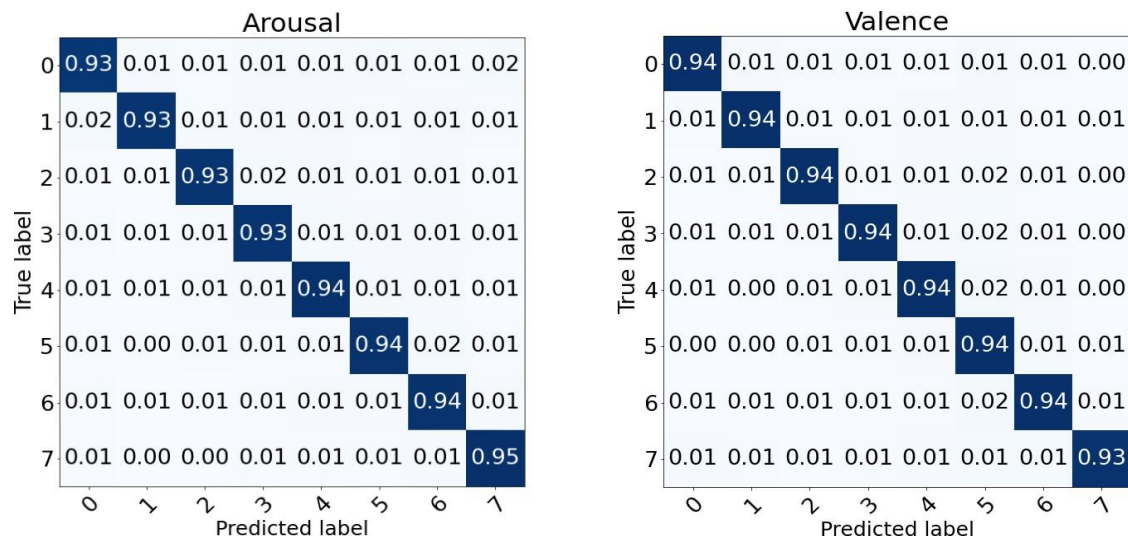


Figure 7.3: Confusion matrix for 8-class classification

The eight-class valence classifier shows the best test accuracy of 93.83% at 120th epoch where it takes 160 epochs to get 98.92% train accuracy.

Table 7.2: 8-class classification report

Arousal				Valence			
Class	Precision	Recall	F1-score	Class	Precision	Recall	F1-score
0	0.93	0.92	0.92	0	0.94	0.94	0.94
1	0.95	0.93	0.94	1	0.94	0.93	0.94
2	0.95	0.93	0.94	2	0.95	0.94	0.94
3	0.95	0.93	0.94	3	0.94	0.94	0.94
4	0.94	0.94	0.94	4	0.95	0.93	0.94
5	0.94	0.94	0.94	5	0.94	0.94	0.94
6	0.95	0.94	0.94	6	0.94	0.94	0.94
7	0.94	0.95	0.94	7	0.94	0.93	0.93

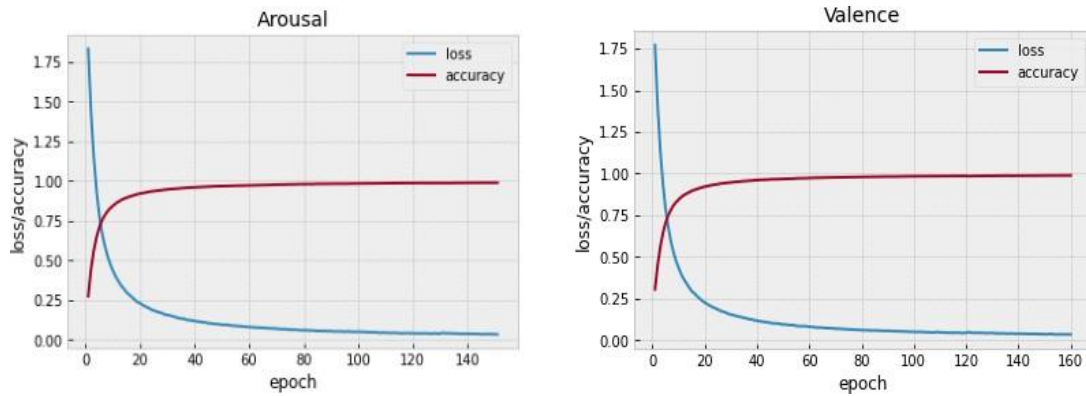


Figure 7.4: Accuracy-loss graph for 8-class

Fig 7.3 presents the confusion matrix for eight class arousal and eight class valence confusion matrix. Here both of the normalized confusion matrices show the difference between true values and the predicted values. And Table 6.2 presents the classification report of arousal and valence. Fig 7.4 has been used for showing accuracy and loss.

Table 7.3: Result summary

Type	Arousal	Valence
Binary-class	96.18%	96.63%
Eight-class	93.79%	93.83%

Table 7.4: Result comparison for binary-class

No.	Modeling technique	No. of class	Accuracy
1	CNN [15]	2-class	91.5%
2	LSTM [3]	2-class	89.83%
3	CDCN [9]	2-class	92.58%
4	KNN, ResNets [16]	2-class	Valence:90.39% Arousal:89.06%
5	Our Model: 1D-CNN	2-class	Valence:96.63% Arousal:96.18%

Using the DEAP dataset those model architectures are on the top list for better emotion recognition accuracy, the best accuracy models from each of them are demonstrated in Table 7.4. Yucel Cimtay Et al. [15] have worked with raw data for binary classification and get the highest accuracy among the reviewed CNN models with an accuracy of 91.%. Many researchers have worked with the famous LSTM model for binary classification and also got good results. According to the literature review, Divya Acharya Et al. [3] have got 89.83% accuracy and this is the best accuracy for LSTM. To get this best LSTM binary classification accuracy have used FFT as feature extraction. Analyzing the results of our reviewed papers, CDCN is also found in the top category for accuracy of binary class emotion recognition and Zhongke Gao Et al. [9] have worked with this model. They have got 92.58% accuracy where DE is the extraction method. Ningjie Liu Et al. [16] have applied their KNN, ResNets on DEAP dataset to get valence: 90.39% and arousal: 89.06% accuracy. But in the same field using FFT with the help of two 1D-CNN models we got 96.07% accuracy for valence and 96.33% accuracy for arousal which is the highest compared to all other models of Table 7.4.

Chapter 8

Conclusion and Future Work

The models that have been used are very simple and light in terms of architecture. Our proposed CNN models are more effective for emotion recognition and outperform previous research in terms of accuracy. These are able to effectively classify preprocessed EEG data. For arousal-valence binary classification accuracy exceeds the same benchmark activities that shows a noticeable difference and introduces a much more precise eight-class classification approach which provides a satisfactory result also.

For future work, we would like to work with our methodology on real-time data so that the emotions of mentally challenged and autistic people can be expressed easily. We will also focus on how to make our recognizing models more efficient and portable.

Bibliography

1. Rayatdoost, Soheil, and Mohammad Soleymani. "Cross-corpus EEG-based emotion recognition." In 2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP), pp. 1-6. IEEE, 2018.
2. Acharya, Divya, Shivani Goel, Rishi Asthana, and Arpit Bhardwaj. "A novel fitness function in genetic programming to handle unbalanced emotion recognition data." *Pattern Recognition Letters* 133 (2020): 272-279.
3. Acharya, Divya, Shivani Goel, Harshit Bhardwaj, Aditi Sakalle, and Arpit Bhardwaj. "A long short term memory deep learning network for the classification of negative emotions using EEG signals." In 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1-8. IEEE, 2020.
4. Sharma, Rahul, Ram Bilas Pachori, and Pradip Sircar. "Automated emotion recognition based on higher order statistics and deep learning algorithm." *Biomedical Signal Processing and Control* 58 (2020): 101867.
5. Wang, Fei, Shichao Wu, Weiwei Zhang, Zongfeng Xu, Yahui Zhang, Chengdong Wu, and Sonya Coleman. "Emotion recognition with convolutional neural network and EEG-based EFDMs." *Neuropsychologia* 146 (2020): 107506.
6. Hassan, Reshad, Sakib Hasan, Md Jubaer Hasan, Md Rafat Jamader, David Eisenberg, and Tanmoy Pias. "Human Attention Recognition with Machine Learning from Brain-EEG Signals." In 2020 IEEE 2nd Eurasia Conference on Biomedical Engineering, Healthcare and Sustainability (ECBIOS), pp. 16-19. IEEE, 2020.
7. Nath, Debarshi, Mrigank Singh, Divyashikha Sethia, Diksha Kalra, and S. Indu. "An Efficient Approach to EEG-Based Emotion Recognition using LSTM Network." In 2020 16th IEEE International Colloquium on Signal Processing & Its Applications (CSPA), pp. 88-92. IEEE, 2020.
8. Cheah, Kit Hwa, Humaira Nisar, Vooi Voon Yap, and Chen-Yi Lee. "Short-time-span EEG-based personalized emotion recognition with deep convolutional neural network." In 2019 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), pp. 78-83. IEEE, 2019.

9. Gao, Zhongke, Xinmin Wang, Yuxuan Yang, Yanli Li, Kai Ma, and Guanrong Chen. "A channel-fused dense convolutional network for EEG-based emotion recognition." *IEEE Transactions on Cognitive and Developmental Systems* (2020).
10. Luo, Yuling, Qiang Fu, Juntao Xie, Yunbai Qin, Guopei Wu, Junxiu Liu, Frank Jiang, Yi Cao, and Xuemei Ding. "EEG-based emotion classification using spiking neural networks." *IEEE Access* 8 (2020): 46007-46016.
11. Cimtay, Yucel, and Erhan Ekmekcioglu. "Investigating the use of pretrained convolutional neural network on cross-subject and cross-dataset EEG emotion recognition." *Sensors* 20, no. 7 (2020): 2034.
12. Cao, Guolu, Yuliang Ma, Xiaofei Meng, Yunyuan Gao, and Ming Meng. "Emotion recognition based on CNN." In *2019 Chinese Control Conference (CCC)*, pp. 8627-8630. IEEE, 2019.
13. Abdel-Ghaffar, Eman A., and Mohamed Daoudi. "Emotion Recognition from Multidimensional Electroencephalographic Signals on the Manifold of Symmetric Positive Definite Matrices." In *2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 354-359. IEEE, 2020.
14. Zhong, Xiaolong, Zhong Yin, and Jianhua Zhang. "Cross-Subject emotion recognition from EEG using Convolutional Neural Networks." In *2020 39th Chinese Control Conference (CCC)*, pp. 7516-7521. IEEE, 2020.
15. Cimtay, Yucel, Erhan Ekmekcioglu, and Seyma Caglar-Ozhan. "Cross-subject multimodal emotion recognition based on hybrid fusion." *IEEE Access* 8 (2020): 168865-168878.
16. Liu, Ningjie, Yuchun Fang, Ling Li, Limin Hou, Fenglei Yang, and Yike Guo. "Multiple feature fusion for automatic emotion recognition using EEG signals." In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 896-900. IEEE, 2018.
17. Koelstra, Sander, Christian Muhl, Mohammad Soleymani, Jong-Seok Lee, Ashkan Yazdani, Touradj Ebrahimi, Thierry Pun, Anton Nijholt, and Ioannis Patras. "Deap: A database for emotion analysis; using physiological signals." *IEEE transactions on affective computing* 3, no. 1 (2011): 18-31.

18. Yoon, Hyun Joong, and Seong Youb Chung. "EEG-based emotion estimation using Bayesian weighted-log-posterior function and perceptron convergence algorithm." *Computers in biology and medicine* 43, no. 12 (2013): 2230-2237.
19. Rozgić, Viktor, Shiv N. Vitaladevuni, and Rohit Prasad. "Robust EEG emotion classification using segment level decision fusion." In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 1286-1290. IEEE, 2013.
20. Zhang, Xiaowei, Bin Hu, Jing Chen, and Philip Moore. "Ontology-based context modeling for emotion recognition in an intelligent web." *World Wide Web* 16, no. 4 (2013): 497-513.
21. Li, Xiang, Dawei Song, Peng Zhang, Guangliang Yu, Yuexian Hou, and Bin Hu. "Emotion recognition from multi-channel EEG data through convolutional recurrent neural network." In *2016 IEEE international conference on bioinformatics and biomedicine (BIBM)*, pp. 352-359. IEEE, 2016.
22. Al-Nafjan, Abeer, Manar Hosny, Areej Al-Wabil, and Yousef Al-Ohali. "Classification of human emotions from electroencephalogram (EEG) signal using deep neural network." *Int. J. Adv. Comput. Sci. Appl* 8, no. 9 (2017): 419-425.
23. Liu, Wei, Wei-Long Zheng, and Bao-Liang Lu. "Emotion recognition using multimodal deep learning." In *International conference on neural information processing*, pp. 521-529. Springer, Cham, 2016.
24. Tripathi, Samarth, Shrinivas Acharya, Ranti Dev Sharma, Sudhanshi Mittal, and Samit Bhattacharya. "Using deep and convolutional neural networks for accurate emotion classification on DEAP dataset." In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 4746-4752. 2017.
25. Zheng, Wei-Long, Jia-Yi Zhu, and Bao-Liang Lu. "Identifying stable patterns over time for emotion recognition from EEG." *IEEE Transactions on Affective Computing* 10, no. 3 (2017): 417-429.
26. Horlings, Robert, Dragos Datcu, and Leon JM Rothkrantz. "Emotion recognition using brain activity." In *Proceedings of the 9th international conference on computer systems and technologies and workshop for PhD students in computing*, pp. II-1. 2008.

27. Mühl, Christian, Brendan Allison, Anton Nijholt, and Guillaume Chanel. "A survey of affective brain computer interfaces: principles, state-of-the-art, and challenges." *Brain-Computer Interfaces* 1, no. 2 (2014): 66-84.
28. The Science of Brainwaves – the Language of the Brain, [Accessed: 03- Jun- 2021], 2021. [Online]. Available: <https://nhahealth.com/brainwaves-the-language/>
29. Matlovic, Tomas, Peter Gaspar, Robert Moro, Jakub Simko, and Maria Bielikova. "Emotions detection using facial expressions recognition and EEG." In *2016 11th international workshop on semantic and social media adaptation and personalization (SMAP)*, pp. 18-23. IEEE, 2016.
30. Electroencephalography, [Accessed: 03-Jun-2021], 2021. [Online]. Available:<https://en.wikipedia.org/wiki/Electroencephalography>
31. Brain Anatomy and How the Brain Works, [Accessed: 03- Jun- 2021], 2021. [Online]. Available:<https://www.hopkinsmedicine.org/health/conditions-and-diseases/anatomy-of-the-brain>
32. Right Hand, Human Brain, [Accessed: 03-Jun-2021], 2021. [Online]. Available: <https://benjaminckinney.com/right-hand-human-brain/>
33. Nagel, Sebastian. "Towards a home-use BCI: fast asynchronous control and robust non-control state detection." PhD diss., Eberhard Karls Universität Tübingen, 2019.

Appendix A

Code Segment (Python)

Installed pyeeg

```
pip install git+https://github.com/forrestbao/pyeeg.git
```

Mounted with google drive

```
from google.colab import drive  
drive.mount('/content/drive')
```

Import necessary library packages

```
import numpy as np  
import pyeeg as pe  
import pickle as pickle  
import pandas as pd  
import math  
  
from sklearn import svm  
from sklearn.preprocessing import normalize  
  
import os  
#import tensorflow as tf  
import time
```

Path set and define variable

```
channel = [1,2,3,4,6,11,13,17,19,20,21,25,29,31] #14 Channels chosen to fit Emotiv Epoch+  
band = [4,8,12,16,25,45] #5 bands  
window_size = 256 #Averaging band power of 2 sec  
step_size = 16 #Each 0.125 sec update once  
sample_rate = 128 #Sampling rate of 128 Hz  
  
subjectList=[]  
for i in range(1,33):  
    if i<10:  
        subjectList.append(f"{i:02d}")  
    else:  
        subjectList.append(f"{i:2d}")
```

```

print(subjectList)

#List of subjects
path_to_dataset_2 = '/content/drive/MyDrive/Thesis/DEAP-
dataset/Datasets/dat_File_Folder/'
path_to_dataset = '/content/drive/MyDrive/Thesis/DEAP-dataset/Datasets/model_5_Khosru/'

```

Plot raw EEG signal

```

dat_file_path = path_to_dataset_2 + "s01.dat"
s01_np = np.fromfile(dat_file_path, dtype='byte')

```

```

from matplotlib import pyplot as plt
plt.style.use('bmh')

```

```

t = np.arange(0, 100, 1)
EEG_s01 = s01_np[:100]

```

```

plt.figure()
plt.plot(t, EEG_s01, label="Raw EEG")
plt.xlabel("time")
plt.ylabel("amplitude")
plt.legend(loc="center left")

```

Feature extraction function define

```

def FFT_Processing (sub, channel, band, window_size, step_size, sample_rate):
    """
    arguments: string subject
               list channel indice
               list band
               int window size for FFT
               int step size for FFT
               int sample rate for FFT
    return:    void
    """
    meta = []
    with open(path_to_dataset_2+'s'+ sub + '.dat', 'rb') as file:

        subject = pickle.load(file, encoding='latin1') #resolve the python 2 data problem by enc
        oding : latin1

        for i in range (0,40):
            # loop over 0-39 trails

```

```

data = subject["data"][i]
labels = subject["labels"][i]
start = 0;

while start + window_size < data.shape[1]:
    meta_array = []
    meta_data = [] #meta vector for analysis
    for j in channel:
        X = data[j][start : start + window_size] #Slice raw data over 2 sec, at interval of
0.125 sec
        Y = pe.bin_power(X, band, sample_rate) #FFT over 2 sec of channel j, in seq of
theta, alpha, low beta, high beta, gamma
        meta_data = meta_data + list(Y[0])

    meta_array.append(np.array(meta_data))
    meta_array.append(labels)

    meta.append(np.array(meta_array))
    start = start + step_size

meta = np.array(meta)
#np.save('C:/Users/faizan/Downloads/data_preprocessed_python/data_preprocessed_py
thon/s' + sub, meta, allow_pickle=True, fix_imports=True)
np.save(path_to_dataset+'s' + sub, meta, allow_pickle=True, fix_imports=True)

```

Feature extraction function call

```

for subjects in subjectList:
    FFT_Processing (subjects, channel, band, window_size, step_size, sample_rate)

```

Modify the default parameters of np.load

```

import numpy as np
# save np.load
np_load_old = np.load

np.load = lambda *a,**k: np_load_old(*a, allow_pickle=True, **k)

```

Splitting data into test set and training set

```

data_training = []
label_training = []
data_testing = []

```

```

label_testing = []

for subjects in subjectList:

    with open(path_to_dataset + 's' + subjects + '.npy', 'rb') as file:
        sub = np.load(file)
        for i in range (0,sub.shape[0]):
            if i % 8 == 0:
                data_testing.append(sub[i][0])
                label_testing.append(sub[i][1])
            else:
                data_training.append(sub[i][0])
                label_training.append(sub[i][1])

```

Restore np.load for future normal usage

```
np.load = np_load_old
```

Training and test data save in google drive

```

np.save(path_to_dataset + 'data_training', np.array(data_training), allow_pickle=True, fix_imports=True)
np.save(path_to_dataset + 'label_training', np.array(label_training), allow_pickle=True, fix_imports=True)
print("training dataset:", np.array(data_training).shape, np.array(label_training).shape)

np.save(path_to_dataset + 'data_testing', np.array(data_testing), allow_pickle=True, fix_imports=True)
np.save(path_to_dataset + 'label_testing', np.array(label_testing), allow_pickle=True, fix_imports=True)
print("testing dataset:", np.array(data_testing).shape, np.array(label_testing).shape)

```

Training data and label load (arousal)

```

with open(path_to_dataset + 'data_training.npy', 'rb') as fileTrain:
    X = np.load(fileTrain)

with open(path_to_dataset + 'label_training.npy', 'rb') as fileTrainL:
    Y = np.load(fileTrainL)

X = normalize(X)
Z = np.ravel(Y[:, [0]])

```

```
Arousal_Train = np.ravel(Y[:, [0]])
Valence_Train = np.ravel(Y[:, [1]])
Domain_Train = np.ravel(Y[:, [2]])
```

```
Like_Train = np.ravel(Y[:, [3]])
```

```
for i in range(len(Z)):
    if Z[i] == 9:
        Z[i] = 8.99
```

Training data and label load (valence)

```
with open(path_to_dataset + 'data_training.npy', 'rb') as fileTrain:
```

```
X = np.load(fileTrain)
```

```
with open(path_to_dataset + 'label_training.npy', 'rb') as fileTrainL:
```

```
Y = np.load(fileTrainL)
```

```
X = normalize(X)
```

```
Z = np.ravel(Y[:, [1]])
```

```
Arousal_Train = np.ravel(Y[:, [0]])
Valence_Train = np.ravel(Y[:, [1]])
Domain_Train = np.ravel(Y[:, [2]])
Like_Train = np.ravel(Y[:, [3]])
```

```
for i in range(len(Z)):
    if Z[i] == 9:
        Z[i] = 8.99
```

Label value segmentation for binary classification (training)

```
count_0 = 0
count_1 = 0
for i in range(len(Z)):
    if Z[i] >= 1 and Z[i] <= 4.99:
        Z[i] = 0
        count_0 = count_0 + 1
    else:
        Z[i] = 1
        count_1 = count_1 + 1
print(count_0, count_1)
```


Import necessary some another libraries

```
import pandas as pd
import keras.backend as K
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from keras.models import Sequential
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
#from keras.utils import to_categorical
from keras.utils.np_utils import to_categorical
from keras.layers import Flatten
from keras.layers import Dense
import numpy as np
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from keras import backend as K
from keras.models import Model
import timeit
from keras.models import Sequential
from keras.layers.core import Flatten, Dense, Dropout
from keras.layers.convolutional import Convolution1D, MaxPooling1D, ZeroPadding1D
from keras.optimizers import SGD
#import cv2, numpy as np
import warnings
warnings.filterwarnings('ignore')
```

Label data categorized (training)

```
from keras.utils.np_utils import to_categorical
y_train = to_categorical(Z)
y_train = y_train[:,1:]
y_train[:10]
```

Train data convert into numpy array format

```
x_train = np.array(X[:])
```

Tasting data and label load (arousal)

```
with open(path_to_dataset + 'data_testing.npy', 'rb') as fileTrain:
```

```

M = np.load(fileTrain)

with open(path_to_dataset + 'label_testing.npy', 'rb') as fileTrainL:
    N = np.load(fileTrainL)

M = normalize(M)
L = np.ravel(N[:, [0]])

Arousal_Test = np.ravel(N[:, [0]])
Valence_Test = np.ravel(N[:, [1]])
Domain_Test = np.ravel(N[:, [2]])
Like_Test = np.ravel(N[:, [3]])

for i in range(len(L)):
    if L[i] == 9:
        L[i] = 8.99

```

Tasting data and label load (valence)

```

with open(path_to_dataset + 'data_testing.npy', 'rb') as fileTrain:
    M = np.load(fileTrain)

with open(path_to_dataset + 'label_testing.npy', 'rb') as fileTrainL:
    N = np.load(fileTrainL)

M = normalize(M)
L = np.ravel(N[:, [1]])

Arousal_Test = np.ravel(N[:, [0]])
Valence_Test = np.ravel(N[:, [1]])
Domain_Test = np.ravel(N[:, [2]])
Like_Test = np.ravel(N[:, [3]])

for i in range(len(L)):
    if L[i] == 9:
        L[i] = 8.99

```

Label value segmentation for binary classification (testing)

```

count_0 = 0
count_1 = 0
for i in range(len(L)):
    if L[i] >= 1 and L[i] <= 4.99:

```

```

L[i] = 0
count_0 = count_0 + 1
else:
    L[i] = 1
    count_1 = count_1 + 1
print(count_0, count_1)

```

Test data convert into numpy array format

```

x_test = np.array(M[:])
x_test

```

Label data categorized (testing)

```

from keras.utils.np_utils import to_categorical
y_test = to_categorical(L)
y_test = y_test[:,1:]
y_test[:10]

```

Fit the train and test data with StandarScaler

```

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)

```

Reshape train and test data

```

x_train = x_train.reshape(x_train.shape[0], x_train.shape[1], 1)
x_test = x_test.reshape(x_test.shape[0], x_test.shape[1], 1)

```

Model parameter define

```

batch_size = 256
num_classes = 8
epochs = 120
input_shape=(x_train.shape[1], 1)

```

Import another library packages

```

from keras.layers import Convolution1D, ZeroPadding1D, MaxPooling1D, BatchNormalizat
ion, Activation, Dropout, Flatten, Dense
from keras.regularizers import l2

```

Model architecture-1

```
model = Sequential()
input_shape=(x_train.shape[1], 1)
model.add(Conv1D(128, kernel_size=3,padding = 'same',activation='relu', input_shape=input_shape))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=(2)))
model.add(Conv1D(128,kernel_size=3,padding = 'same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=(2)))
#model.add(Conv1D(64,kernel_size=3,padding = 'same', activation='relu'))
#model.add(MaxPooling1D(pool_size=(2)))
model.add(Flatten())
model.add(Dense(64, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(32, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(16, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
```

Model architecture-2

```
model = Sequential()
input_shape=(x_train.shape[1], 1)
model.add(Conv1D(256, kernel_size=3,padding = 'same',activation='relu', input_shape=input_shape))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=(2)))
model.add(Conv1D(128,kernel_size=3,padding = 'same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=(2)))
model.add(Conv1D(64,kernel_size=3,padding = 'same', activation='relu'))
model.add(MaxPooling1D(pool_size=(2)))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
```

Model architecture-3

```

model = Sequential()
input_shape=(x_train.shape[1], 1)
model.add(Conv1D(128, kernel_size=3,padding = 'same',activation='relu', input_shape=input_shape))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=(2)))
model.add(Conv1D(64,kernel_size=3,padding = 'same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=(2)))
model.add(Conv1D(32,kernel_size=3,padding = 'same', activation='relu'))
model.add(MaxPooling1D(pool_size=(2)))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
model.summary()

```

Model architecture-4

```

model = Sequential()
input_shape=(x_train.shape[1], 1)
model.add(Conv1D(512, kernel_size=9,padding = 'same',activation='relu', input_shape=input_shape))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=(2)))
model.add(Conv1D(128,kernel_size=6,padding = 'same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=(2)))
model.add(Conv1D(32,kernel_size=3,padding = 'same', activation='relu'))
model.add(MaxPooling1D(pool_size=(2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
model.summary()

```

Model architecture-5

```

model = Sequential()

```

```

model.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=input_shape))
model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
model.add(Dropout(0.5))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(num_classes, activation='hard_sigmoid'))

initial_learning_rate = 0.001
lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate,
    decay_steps=100000,
    decay_rate=0.96,
    staircase=True)

```

Model architecture-6

```

model = Sequential()

input_shape=(x_train.shape[1], 1)

model.add(Conv1D(1024, kernel_size=9,padding = 'same',activation='relu', input_shape=input_shape))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=(2)))

model.add(Conv1D(512,kernel_size=6,padding = 'same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=(2)))

model.add(Conv1D(256,kernel_size=6,padding = 'same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=(2)))

model.add(Conv1D(128,kernel_size=6,padding = 'same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=(2)))

model.add(Flatten())

model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))

```

```

model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(num_classes, activation='softmax'))
model.summary()

```

Compile the model

```

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer='adam',
              metrics=['accuracy'])

```

Plot model architecture specification

```

from keras.utils.vis_utils import plot_model
plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)

```

Save checkpoint details

```

# check points and early stopping
from keras.callbacks import ModelCheckpoint, EarlyStopping
model_name = ""
filepath="/content/drive/MyDrive/Thesis/DEAP-
dataset/Saved_checkpoints_2/Copy_Valance_Check_point_2/" + model_name + "weights-
improvement-{epoch:02d}-{accuracy:.4f}.hdf5"
print(filepath)
checkpoint = ModelCheckpoint(filepath, monitor='accuracy', verbose=1, save_best_only=True,
                             mode='max')
es = EarlyStopping(monitor='accuracy', mode='max', verbose=1, patience=15)
callbacks_list = [es, checkpoint]

```

Fit the model

```

H = model.fit(x_train, y_train,
              batch_size=batch_size,
              epochs=epochs,
              verbose=1,
              callbacks=callbacks_list)

```

Find the accuracy after completing the model train

```
score = model.evaluate(x_test, y_test, verbose=1)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

Manual save

```
base_path_model = "/content/drive/My Drive/Google_Colab/Autoencoder/saved_models/"
accuracy = "_89"
path_model = base_path_model + model_name + accuracy + ".h5"
model.save(path_model)
```

Find the accuracy from a save point

```
from keras.models import load_model
model_loaded = load_model('/content/drive/MyDrive/Thesis/DEAP-
dataset/Saved_checkpoints_2/Copy_Valance_Check_point_2/weights-improvement-120-
0.9857.hdf5')
score = model_loaded.evaluate(x_test, y_test, verbose=1)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

Training loss graph

```
N = num_classes
EPOCHS = 120
# construct a plot that plots and saves the training history
import matplotlib.pyplot as plt
N = np.arange(0, EPOCHS)
plt.style.use("ggplot")
plt.figure()
plt.plot(N, H.history["loss"], label="train_loss")
# plt.plot(N, H.history["val_loss"], label="val_loss")
plt.title("Training Loss Graph")
plt.xlabel("Epoch #")
plt.ylabel("Loss")
plt.legend(loc="upper left")
plt.show()
```

Training accuracy graph

```
N = num_classes
EPOCHS = 120
# construct a plot that plots and saves the training history
import matplotlib.pyplot as plt
N = np.arange(0, EPOCHS)
plt.style.use("bmh")
```



```
plt.figure()
plt.plot(N, H.history["accuracy"], label="Train_Accuracy")
#plt.plot(N, H.history["val_loss"], label="val_loss")
plt.title("Training Accuracy Graph")
plt.xlabel("Epoch #")
plt.ylabel("Accuracy")
plt.legend(loc="lower left")
plt.show()
```

Training Loss & Training Accuracy graph

```
# summarize history for loss
plt.plot(H.history['loss'])
plt.plot(H.history['accuracy'])
plt.title("Training Loss & Training Accuracy graph")
plt.ylabel('loss/accuracy')
plt.xlabel('epoch')
plt.legend(['loss', 'accuracy'], loc='upper right')
plt.show()
```

Confusion matrix plotting function

```
from sklearn.utils.multiclass import unique_labels
from matplotlib import pyplot as plt
def plot_confusion_matrix(y_true, y_pred, classes,
                          normalize=False,
                          title=None,
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if not title:
        if normalize:
            title = 'Normalized confusion matrix'
        else:
            title = 'Confusion matrix, without normalization'

    # Compute confusion matrix
    cm = confusion_matrix(y_true, y_pred)
    # Only use the labels that appear in the data
    classes = classes[unique_labels(y_true, y_pred)]
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
```

```

else:
    print('Confusion matrix, without normalization')

print(cm)

fig, ax = plt.subplots()
im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
ax.figure.colorbar(im, ax=ax)
# We want to show all ticks...
ax.set(xticks=np.arange(cm.shape[1]),
       yticks=np.arange(cm.shape[0]),
       # ... and label them with the respective list entries
       xticklabels=classes, yticklabels=classes,
       title=title,
       ylabel='True label',
       xlabel='Predicted label')

# Rotate the tick labels and set their alignment.
plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
         rotation_mode="anchor")

# Loop over data dimensions and create text annotations.
fmt = '.2f' if normalize else 'd'
thresh = cm.max() / 2.
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        ax.text(j, i, format(cm[i, j], fmt),
                ha="center", va="center",
                color="white" if cm[i, j] > thresh else "black")
fig.tight_layout()
return ax

np.set_printoptions(precision=2)

```

Confusion matrix plotting function call

```

import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (30,10)
plt.rcParams.update({'font.size': 27})

from sklearn.metrics import confusion_matrix

y_pred = model_loaded.predict(x_test)
y_test_argmax = y_test.argmax(axis=1)
y_pred_argmax = y_pred.argmax(axis=1)

```

```

class_names = np.array([0,1, 2, 3, 4, 5, 6, 7])

# Plot non-normalized confusion matrix
plot_confusion_matrix(y_test_argmax, y_pred_argmax, classes=class_names,
                      title='Confusion matrix, without normalization')
# Plot normalized confusion matrix
plot_confusion_matrix(y_test_argmax, y_pred_argmax, classes=class_names, normalize=True,
                      title='Valence')
#fig = plt.figure(figsize=(20,20), dpi=300)
plt.xlabel('Predicted label', fontsize=25)
plt.show()
#plt.figure(figsize=(40, 40))

```

Classification report

```

y_true = np.array(y_test)

y_pred = np.squeeze(model.predict(x_test))
y_pred = np.array(y_pred >= 0.5, dtype=np.int)

from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))

```

Load model from a save point

```

from keras.models import load_model
model_loaded = load_model('/content/drive/MyDrive/Thesis/DEAP-
dataset/Saved_checkpoints_2/Copy_Valance_Check_point_2/weights-improvement-120-
0.9857.hdf5')

H = model_loaded.fit(x_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_split = 0.2,
                    callbacks= callbacks_list)

```

Test accuracy from a saving point

```

from keras.models import load_model

```

```
model_loaded = load_model('/content/drive/MyDrive/Thesis/DEAP-  
dataset/Saved_checkpoints_2/Copy_Valance_Check_point_2/weights-improvement-115-  
0.9854.hdf5')  
score = model_loaded.evaluate(x_test, y_test, verbose=1)  
print('Test loss:', score[0])  
print('Test accuracy:', score[1])
```