TUGAS AKHIR - IF184802

Parallelizing CNN and Transformer Encoders for Audio Based Emotion Recognition in English Language

Adam Satria Adidarma

NRP 05111942000001

Dosen Pembimbing

Kelly Rossa Sungkono, S.Kom., M.Kom.

NIP 1994201912088

Dosen Pembimbing 2

Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D

NIP 1987202012004

Program Studi S1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2022

TUGAS AKHIR – IF184802

Klasifikasi Emosi Manusia untuk Audio Berbahasa Inggris Menggunakan CNN dan Encoder Transformer dengan Teknik Pararel

Adam Satria Adidarma

NRP 05111492000001

Dosen Pembimbing

Kelly Rossa Sungkono, S.Kom., M.Kom.

NIP 1994201912088

Dosen Pembimbing 2

Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D

NIP 1987202012004

HALAMAN JUDUL

Program Studi S1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2022

FINAL PROJECT – IF184802

Parallelizing CNN and Transformer Encoders for Audio Based Emotion Recognition in English Language

Adam Satria Adidarma

NRP 05111942000001

Advisor I

Kelly Rossa Sungkono, S.Kom., M.Kom.

NIP 1994201912088

Advisor II

Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D

NIP 1987202012004

Study Program Bachelor of Informatics

Department of Informatics

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

Surabaya

2022

LEMBAR PENGESAHAN

Klasifikasi Emosi Manusia untuk Audio Berbahasa Inggris Menggunakan CNN dan Encoder Transformer dengan Teknik Pararel

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
Memperoleh gelar Sarjana Komputer pada
Program Studi S-1 Teknik Informatika
Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh : **Adam Satria Adidarma**
NRP. 05111942000001

Disetujui oleh Tim Penguji Tugas Akhir:

1.	Kelly Rossa Sungkono, S.Kom., M.Kom.	Pembimbing
2.	Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D	Ko-pembimbing
3.	Nama dan gelar penguji	Penguji
4.	Nama dan gelar penguji	Penguji
5.	Nama dan gelar penguji	Penguji

SURABAYA
Maret, 2023

APPROVAL SHEET

Parallelizing CNN and Transformer Encoders for Human Emotion Classification for Audio Based Emotion Recognition in English Language

FINAL PROJECT

Submitted to fulfill one of the requirements
for obtaining a degree Bachelor of Computer Science at
Undergraduate Study Program of Informatics
Department of Informatics
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember

By: **Adam Satria Adidarma**
NRP. 05111942000001

Approved by Final Project Examiner Team:

1.	Kelly Rossa Sungkono, S.Kom., M.Kom.	Advisor
2.	Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D	Co-Advisor
3.	Name of Examiner and academic title	Examiner
4.	Name of Examiner and academic title	Examiner
5.	Name of Examiner and academic title	Examiner

SURABAYA
Month, Year

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

Nama mahasiswa / NRP : Adam Satria Adidarma / 05111942000001
Departemen : Teknik Informatika
Dosen Pembimbing / NIP :_____

Dengan ini menyatakan bahwa Tugas Akhir dengan judul "Judul Tugas Akhir" adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah. Bilamana di kemudian hari dietemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima saksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, _____

Mengetahui	
Dosen Pembimbing	Mahasiswa
(_____)	(Adam Satria Adidarma)
NIP.	NRP. 05111942000001

STATEMENT OF ORIGINALITY

The undersigned below:

Name of student / NRP : Adam Satria Adidarma / 05111942000001
Department : Informatics
Advisor / NIP :_____

Hereby declare that Final Project with the title of "Title of Final Project" is the result of my own work, is original, and is written by following the rules of scientific writing.
If in the future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with the provisions that apply at Institut Teknologi Sepuluh Nopember.

Surabaya, _____

Acknowledge	
Advisor	Student
(_____)	(Adam Satria Adidarma)
NIP.	NRP. 05111942000001

Klasifikasi Emosi Manusia untuk Audio Berbahasa Inggris Menggunakan CNN dan Encoder Transformer dengan Teknik Pararel

Nama Mahasiswa / NRP : Adam Satria Adidarma / 05111942000001
Departemen : Teknik Informatika FTEIC- ITS
Pembimbing : Kelly Rossa Sungkono, S.Kom., M.Kom.
Ko-Pembimbing : Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D

ABSTRAK

Abstrak

Kecerdasan artifisial telah berdampak signifikan pada berbagai industri dan sektor masyarakat, dengan adopsi kecerdasan artifisial yang tumbuh 37% dari 2018 hingga 2019, menurut laporan Gartner. Pengenalan emosi bicara (PEB) adalah subbidang kecerdasan artifisial yang fokus pada mengenali aspek emosional manusia saat berbicara, terpisah dari konten semantik. Emosi berperan penting dalam komunikasi manusia dan telah menjadi objek penelitian yang semakin meningkat dalam beberapa tahun terakhir. Meskipun studi saat ini tentang deteksi emosi sering memfokuskan pada modalitas visual, seperti ekspresi wajah, emosi adalah konsep multimodal yang membutuhkan studi terhadap indikator visual, taktil, vokal, dan fisiologis. PEB dapat diterapkan dalam berbagai konteks, termasuk pusat panggilan, pendidikan, pemasaran, psikologi, dan kesehatan.

Penelitian ini mengusulkan pendekatan untuk menerapkan sistem PEB menggunakan model Convolution Neural Network (CNN) yang diparalelkan dengan Transformer Encoder Block dan mengevaluasi kinerjanya pada beberapa dataset emosi audio berbahasa Inggris yang tersedia secara publik, seperti CREMA-D, RAVDESS, dan SAVEE. Untuk mengekstraksi fitur dari suara-suara ini, digunakan Mel Frequency Cepstrum Coefficients (MFCC). Model yang diusulkan dibandingkan dengan berbagai arsitektur kecerdasan arifisial, termasuk CNN LeNet, arsitektur Convolutional Recurrent Neural Network (CRNN), dan model Support Vector Machine (SVM), untuk menentukan pendekatan yang paling efektif untuk PEB. Kinerja setiap model bervariasi tergantung pada dataset yang digunakan. Diantara model yang dibandingkan, model CNN dan Transformer Encoder Block menunjukkan kinerja terbaik secara keseluruhan pada semua dataset. Blok Pengekoder Transformer dan CNN adalah teknik kecerdasan artifisial yang kuat untuk pemrosesan data sekuensial. Transformer Encoder Block efektif dalam memproses urutan data yang panjang dan efektif dalam memproses bahasa alami. Sementara itu, CNN sangat baik dalam menangkap pola lokal dan hubungan dalam data, sehingga cocok untuk menganalisis gambar spektrogram yang dihasilkan dari sinyal audio.

Kata kunci: Kecerdasan Artifisial, CNN, Transformer, PEB, *Self-Attention*, Audio

Parallelizing CNN and Transformer Encoders for Human Emotion Classification for Audio Based Emotion Recognition in English Language

Student Name / NRP : Adam Satria Adidarma / 05111942000001
Department : Teknik Informatika FTEIC- ITS
Advisor : Kelly Rossa Sungkono, S.Kom., M.Kom.
Co-Advisor : Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D

Abstract

Artificial intelligence (AI) has had a significant impact on various industries and sectors of society, with the adoption of AI growing 37% from 2018 to 2019, according to a Gartner report. Speech emotion recognition (SER) is a subfield of AI that focuses on recognizing the emotional aspects of speech, separate from the semantic content. Emotions play a crucial role in human communication and have been the subject of increasing research in recent years. While current studies on emotion detection often focus on visual modalities, such as facial expressions, emotion is a multimodal concept that requires the study of visual, tactile, vocal, and physiological indicators. SER can be applied in various contexts, including call centers, education, marketing, psychology, and healthcare.

This study proposes an approach to implement an SER system using a parallelized Convolutional Neural Network (CNN) model with a Transformer Encoder Block and evaluate its performance on some publicly available English audio emotion dataset such as CREMA-D, RAVDESS, and SAVEE. To extract the features of these sounds, the Mel Frequency Cepstrum Coefficients (MFCC) was used. The transformer encoder block with CNN model is compared to various machine learning architectures, including the original LeNet CNN, a Convolutional Recurrent Neural Network (CRNN) architecture, and a Support Vector Machine (SVM) model, to determine the most effective approach for SER. The performance for each model varies significantly depending on the dataset being used. Among the models being compared, the Transformer Encoder and CNN model showed the best overall performance across all datasets. Both the Transformer Encoder Block and CNN are powerful deep-learning techniques for processing sequential data. The Transformer Encoder Block is effective when processing long sequences of data and is highly effective in natural language processing tasks. Meanwhile, CNNs are excellent at capturing local patterns and relationships in the data, making them ideal for analyzing spectrogram images generated from audio signals.

Keywords: AI, CNN, Transformer, SER, Self-Attention, Audio.

KATA PENGANTAR

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Table of Contents

HALAMAN JUDUL i
LEMBAR PENGESAHAN iii
PERNYATAAN ORISINALITAS v
ABSTRAK vii
KATA PENGANTAR ix
Table of Contents x
List of Figures xii
List of Tables xiii
Chapter I 1
1.1 Background 1
1.2 Problem Statement 2
1.3 Problem Scope 2
1.4 Purpose 2
1.5 Benefit 3
Chapter II 4
2.1 Related Works 4
2.2 Basic Theory 5
2.2.1 Emotion 6
2.2.2 Sound 6
2.2.3 Speech Recognition 7
2.2.4 Feature Extraction 7
2.2.5 Convolutional Neural Networks (CNN) 9
2.2.6 Transformer 12
2.2.7 PyTorch 16
2.2.8 Support Vector Machine (SVM) 16
2.2.9 Convolutional Recurrent Neural Network (CRNN) 17
2.2.10 Confusion Matrix 19
2.2.11 Librosa 20
2.2.12 Normalization 21
Chapter III 23
3.1 Designed Method 23
3.1.1 Supporting Tools 24
3.1.2 Hardware 24

3.1.3 Software 24
3.2 Implementation and Trial Plans 24
3.2.1 Dataset 24
3.2.2 Model Implementation 26
3.2.3 Comparison Model Architecture Design 32
3.2.4 Model Evaluation 36
3.2.5 User Interface 38
Chapter IV 40
4.1 Implementation 40
4.1.1 Data Exploration 40
4.1.2 Pre-Processing 45
4.1.3 Feature Extraction 49
4.2 Experiment Results 50
4.2.1 Support Vector Machine (SVM) 51
4.2.2 Convolutional Based Architecture 54
4.2.3 Pararel Transformer Encoder with CNN Architecture 62
4.3 Method Comparisons 67
Chapter V 69
5.1 Conclusion 69
5.2 Recommendation 69
References 71
APPENDIX 75
BIODATA PENULIS 93

List of Figures

Figure 2.1: Human Emotions (Charlie, 2014). 6
Figure 2.2: Longitudinal Nature of Sound Wave (StudyCorgi, 2022). 7
Figure 2.3: MFCC Block Diagram. 8
Figure 2.4: CNN Architecture. 10
Figure 2.5: 2×2 Convolution Filter. 10
Figure 2.6: 2×2 Max Pooling Layer. 11
Figure 2.7: Transformer Architecture (Vaswani, et al., 2017). 13
Figure 2.8: Multi-Head Attention (Vaswani, et al., 2017). 13
Figure 2.9: Scaled Dot-Product Attention (Vaswani, et al., 2017). 15
Figure 2.10: Encoder Block (KiKaBeN, 2021). 15
Figure 2.11: Decoder Block (KiKaBeN, 2021). 16
Figure 2.12: CRNN Architecture (Shi, Bai, & Yao, 2017). 18
Figure 2.13: Bidirectional LSTM (Ihianle, et al., 2020). 19
Figure 2.14: Confusion Matrix Table. 19
Figure 3.1: Model Architecture 23
Figure 3.2: Pre-Processing Flowchart Diagram. 27
Figure 3.3: SVM Flowchart Diagram. 32
Figure 3.4: Deep Learning Flowchart Diagram. 34
Figure 3.5: Web Design Pre-Audio Input. 38
Figure 3.6: Web Design Post-Audio Input. 38
Figure 4.1: SVM Data Distributions. 48
Figure 4.2: Deep Learning Based Data Distributions. 49
Figure 4.3: CREMA-D Accuracy and Loss Curve During Training Process. 54
Figure 4.4: SAVEE Accuracy and Loss Curve During Training Process. 55
Figure 4.5: RAVDESS Accuracy and Loss Curve During Training Process. 55
Figure 4.6: CREMA-D Accuracy and Loss Curve During Training Process. 59
Figure 4.7: SAVEE Accuracy and Loss Curve During Training Process. 59
Figure 4.8: RAVDESS Accuracy and Loss Curve During Training Process. 59
Figure 4.9: CREMA-D Accuracy and Loss Curve During Training Process. 63
Figure 4.10: SAVEE Accuracy and Loss Curve During Training Process. 63
Figure 4.11: RAVDESS Accuracy and Loss Curve During Training Process. 63

List of Tables

Table 3.1: Transformer Encoder Block Combined with CNN-based Architecture Layer. 31
Table 3.2: SVM Model Parameters. 33
Table 3.3: Deep Learning Model Parameters. 34
Table 3.4: LeNet Architecture Layer. 34
Table 3.5: CRNN Architecture Layer. 35
Table 3.6: Web-based Application Feature Functionality. 39
Table 4.1 CREMA-D Amplitude and Spectogram. 41
Table 4.2 RAVDESS Amplitude and Spectogram 42
Table 4.3: SAVEE Amplitude and Spectogram. 43
Table 4.4: Model Accuracy Comparison on Librosa Data Load Parameters. 45
Table 4.5: SVM Model Accuracy Comparison on Different Split Data Ratio. 47
Table 4.6: Deep Learning Based Model Accuracy Comparison on Different Split Data Ratio. 48
Table 4.7: Speech Signals Features. 50
Table 4.8: SVM CREMA-D Confusion Matrix. 51
Table 4.9: SVM RAVDESS Confusion Matrix. 51
Table 4.10: SVM SAVEE Confusion Matrix. 52
Table 4.11: Comparison of SVM Precision, Recall, and F1 Score for Emotion Classification. 52
Table 4.12: Amplitude Comparison Between Closely Related Emotions in the Dataset. 53
Table 4.13: LeNet CREMA-D Confusion Matrix. 55
Table 4.14: LeNet RAVDESS Confusion Matrix. 56
Table 4.15: LeNet SAVEE Confusion Matrix. 56
Table 4.16: Comparison of LeNet Precision, Recall, and F1 Score for Emotion Classification. 57
Table 4.17: Amplitude Comparison Between Closely Related Emotions in the Dataset. 57
Table 4.18: CRNN CREMA-D Confusion Matrix. 60
Table 4.19: CRNN RAVDESS Confusion Matrix. 60
Table 4.20: CRNN SAVEE Confusion Matrix. 60
Table 4.21: Comparison of CRNN Precision, Recall, and F1 Score for Emotion Classification. 61
Table 4.22: Amplitude Comparison Between Closely Related Emotions in the Dataset. 62
Table 4.23: T. Encoder and CNN CREMA-D Confusion Matrix. 64
Table 4.24: T. Encoder and CNN RAVDESS Confusion Matrix. 64
Table 4.25: T. Encoder and CNN SAVEE Confusion Matrix. 65
Table 4.26: Comparison of T.Encoder and CNN Precision, Recall, and F1 Score for Emotion Classification. 65
Table 4.27: Amplitude Comparison Between Closely Related Emotions in the Dataset. 66
Table 4.28: Model Comparison on Each Trial. 67

Chapter I
Introduction

In this chapter, the research background and context will be explored, including the problem being addressed, the scope of the problem, and the purpose and potential benefits of the research being conducted.

1.1 Background

Artificial Intelligence has emerged in every industry and has a profound impact on every sector of human society. According to Gartner Report (Costello, 2019), artificial intelligence adoption has grown 37% during 2018-2019 because the capabilities of artificial intelligence have matured significantly over the years leading to the adoption of this technology by enterprises around the world. Speech Emotion Recognition (SER) is one of the emerging applications in the context of artificial intelligence. SER is the task of recognizing the emotional aspects of speech independently over the semantic content. Humans can efficiently perform this task as a natural part of our communication, but the ability to do it automatically using a programmable device is still a subject of research (Lech, Stolar, Best, & Bolia, 2020).

In the book of The Media Equation (Ivar, Byron, & Clifford, 1996), Studies in human-computer interaction made the discovery that people often interact with computers as if they were other people and react to similar feedback from humans. Most of these social aspects ranging from politeness to reciprocity have been observed in human-computer interactions. Computer scientists believed that emotions and machines should connect in order to have better and more effective communication. Both data-driven reasoning and emotional perception are crucial for a machine intelligence (Cowie, 2001). Giving machines emotional intelligence, the general user experience, and machine performance will be improved.

Emotions play a big role in human communication. Over the past years, research to understand human emotions was increasing (Jarymowicz & Maria, 2012). There are already a variety of computer systems that uses emotional speech classification such as security systems, psychology and computer vision applications, and interactive computer designs. Current studies on emotion detection mainly focus on visual modalities, including facial expressions, muscle movements, hand posture, body posture, *etc.* (Keltner, Dacher, & Cordaro, 2017). However, emotion is a multimodal concept, and the task to detect emotions requires interdisciplinary studies that include visual modality, tactile communication, vocalization, and physiological indications (Heredia, Cardinale, Dongo, & Díaz-Amado, 2021).

A speech recognition system depends on the selection of a speech multimodal database, the extraction of pertinent features, and the selection of an effective classification algorithm. In the aforementioned works, emotion detection using audio data was chosen because it can be applied to various computer application system that doesn't require visual modalities, such as emotion detection on call center services to analyze customer habits to help improve the quality of service for the provider through sounds. Emotion detection based on audio data can also help learning experience in the field of education to help improve students' mental health by monitoring their emotions through sound. This system can also be used across various applications, such as marketing, psychology, health care, *etc*.

Emotion classification and sound detection using multiple SVM methods, such as linear and nonlinear, have received significant interest recently (Sonawane, Inamdar, & Bhangale, 2017). Some studies also tried to improve the accuracy of this method by using transfer learning on pre-trained deep learning models (Latif, Rana, Younis, Qadir, & Epps, 2018). Their results showed that deep learning-based Convolutional Neural Network (CNN) methods outperformed the handcrafted feature-based SVM method in image classification (Younghak Shin, 2017).

This is because deep learning methods learn categories incrementally through its hidden layer architecture, defining low-level categories first, and then moving to the higher-level categories. The number of datasets can also be a factor in improving the quality of this CNN method (Mahapatra, 2018). In addition, some robust deep learning architectures such as GPT-3 & BERT (Brown, et al., 2020) (Devlin, Chang, Lee, & Toutanova, 2019) are emerging to solve sequential learning problems based on a self-attention mechanism in the Transformer Network (Vaswani, et al., 2017). These architectures are now considered a state-of-the-art technique in the field of NLP (Natural Language Processing).

Based on the above statement, this study aims to implement a deep learning LeNet-based CNN architecture in parallel with a self-attention mechanism Transformer Encoder Block in the process of detecting human emotions with an English audio dataset. This model will be compared against a machine learning-based Support Vector Machine (SVM) and a deep learning convolution-based architecture, which includes the LetNet-based CNN, and the Convolutional Recurrent Neural Network (CRNN) model. By conducting this comparison, the performance and effectiveness of the Pararel Transformer Encoder with CNN model in relation to these alternative machine-learning approaches can be assessed. The objective is to determine the most suitable and accurate method for emotion detection, taking into consideration factors such as expressive feature representation and the ability to predict different emotions based on the overall structure of the Mel Frequency Cepstrum Coefficients (MFCC) plot.

1.2 Problem Statement

From the background stated previously, the problem statement can be expressed as follows:

a) How to detect human emotions from audio data with the Pararel Transformer Encoder with CNN Architecture?

b) How to build the model architecture to give a good accuracy?

c) Which classification methods are more accurate to detect emotion through audio between SVM, LeNet based CNN, CRNN, and the Pararel Transformer Encoder with CNN Architecture?

1.3 Problem Scope

In order to stay true to the issues raised above, this paper includes a number of constraints. The problem in this paper has the following limitations:

a) Audio data is in English;

b) In one voice of the dataset, there is only one emotion;

c) The model can only distinguish between the six emotions of anger, disgust, fear, happy, neutral, and sad;

1.4 Purpose

The purpose of this research is as follows:

a) Find out the process to detect human emotions from audio data with the Pararel Transformer Encoder with CNN Architecture;

b) Determine which method has higher accuracy between SVM model, LeNet based CNN model, CRNN, and the Pararel Transformer Encoder with CNN Architecture for detecting emotions through audio;

c) Determine what architecture is going to give a good accuracy for the Pararel Transformer Encoder with CNN Architecture model;

1.5 Benefit

The purpose of this study is to implement a human emotion detection system through voice for emotional perceptions of a robot machine intelligence. This study will be beneficial in several ways. Firstly, the Pararel Transformer Encoder with CNN Architecture aims to provide more accurate and reliable results compared to existing methods, thereby enhancing the development of artificial intelligence (AI) systems. By improving the detection of human emotions, AI systems can offer more personalized and tailored services, leading to enhanced user experiences and a better understanding of human behaviour. Secondly, the study intends to compare the results of the Pararel Transformer Encoder with CNN Architecture with existing methods, providing valuable insights into its effectiveness in detecting human emotions. This comparison will contribute to the advancement of research in the field and guide future investigations, helping to refine and optimize emotion detection techniques. Furthermore, the application of the Pararel Transformer Encoder with CNN Architecture holds the potential in enabling AI systems to interact more effectively with humans and deliver personalized services. This can have significant implications across various domains, such as customer service and healthcare, where the ability to understand and respond to human emotions is crucial. By integrating emotion detection capabilities, AI systems can provide more empathetic and tailored support, enhancing customer satisfaction and overall service quality. In summary, the Pararel Transformer Encoder with CNN Architecture for human emotion detection through voice offers the potential for improved accuracy, comparative analysis with existing methods, and the development of AI systems that can better interact with humans. These benefits contribute to advancements in AI technology, personalized services, and the understanding of human emotions, with implications across diverse industries and applications.

Chapter II
Literature Review

This chapter will discuss about previous research on this topic and present the foundational theories that guide this study.

2.1 Related Works

In the context of detecting human emotions through audio data, the selection and extraction of audio features are important to understand. The Sequential Minimal Optimization (SMO) algorithm was used as the primary method of sound analysis during the training of SVM models in recent years. In this case, the sound is divided into a number of frames which will then be examined iteratively. There are two emotional characteristics of the voice that can be observed to understand human emotion (Citron, Gray, Critchley, Weekes, & Ferstl, 2014): arousal, which is the level of autonomic activation that an event creates, which ranges from calm to excited. and valence, which is the level of pleasantness that an event generates and is defined along a continuum from negative to positive.

The INTERSPEECH 2013 (Steidl, et al., 2013) introduced various aspects of speech and audio that are connected to emotions which employ the SMO algorithm using a rather 'brute force' method to classify and define audio feature sets. Another research such as (Eyben, et al., 2016) introduced a new method of audio feature extraction using a minimal set of parameters, which implements prosodic, excitation, vocal tract, spectral descriptors, and an extension to the minimalistic set, which contains a small set of cepstral parameters (i.e., MFCC & Spectral Flux).

Emotion recognition from pure speech is a highly advanced and widely used method, and advancements in this area heavily depend on the development of comprehensive collections of emotional speech data. The structure of the emotional speech corpus can be divided into two parts in general. The first part is lab recording, which is a collection of speech datasets that are often recorded in a recording studio using high-quality microphones and accompanied by linguistics experts. Some of the corpora that use this type of structure are EmoDB (Burkhardt, Paeschke, Rolfes, Sendlmeier, & Weiss, 2005), a database of German emotional speech comprising 800 sentences with 10 utterances by 10 different actors that could be used in normal conversation and could be interpreted according to all the emotions employed. IEMOCAP (Busso, et al., 2008), a database consisting of 12 hours worth of audiovisual with multimodal and multispeaker data, including 10 actors both scripted and improvised sessions recorded by SAIL Lab. AESDD of the University of Southern California (Vryzas, Kotsakis, Liatsou, Dimoulas, & Kalliris, 2018), includes Greek language expressions of acted emotional speech and the other controlling spontaneous emotional speech. The second corpus type is non-lab recording. This corpus contains utterances that reflect emotions involuntarily in natural scenarios, such as living spaces, theatrical performances, etc. Some examples that employ this type of corpus are DAPS (Mysore, 2015), this dataset is a collection of aligned recordings of the same speech made on typical consumer devices in real-world settings that consist of approximately 4 and a half hours of data. Freefield1010 (Stowell & Plumbey, 2013), a collection of 7690 excerpts from field recordings throughout the world, was later standardized for research. CHEAVD (Li, Tao, Chao, Bao, & Liu, 2017), containing 140 minutes of emotional segments from movies, TV shows, and talk shows with 238 speakers, ranging from children to the elderly, covers a wide range of speaker diversity.

Studies on different methods of speech representation have been done in recent years with various types of deep-learning architecture. In 2019 (Schneider, Baevski, Collobert, & Auli, 2019), the wav2vec model introduced us to unsupervised learning for speech recognition by

learning representations of unprocessed audio data. Then in 2020 (Baevski, Zhou, Mohamed, & Auli, 2020), the second version of this model was introduced which improves the model even further by employing a self-supervised training method based on contrastive learning for automatic speech recognition. However, in 2021, HuBERT (Hsu, et al., 2021) highlighted many issues with the self-supervised learning approach. These problems include (1) many pronunciation units in the speech, (2) no vocabulary of sound units during the pre-training phase, and (3) the length of sound units being changeable without any segmentation. With these problems, the idea of the HuBERT model is to apply the prediction loss only to masked regions and force the model to learn good high-level representations of unmasked inputs to infer the targets of masked ones correctly. Other studies such as the UniSpeech (Wang, et al., 2021) pointed out a problem in the speech recognition community that some of the successful techniques require thousands of hours of human-annotated speech recordings for training which is not available for a lot of languages spoken worldwide. The UniSpeech model can learn consistent contextual representations using both supervised and unsupervised data. This model consists of convolutional feature extraction, a transformer encoder, and a feature quantizer. UniSpeech is able to perform better than both supervised and unsupervised pre-training on multilingual speech recognition tasks. Furthermore, WavLM (Chen, et al., 2022) was introduced as an extension of HUBERT (Hsu, et al., 2021) to masked speech prediction and denoising modeling, so the pre-trained model performs well on both automatic and non-automatic speech recognition to solve full stack speech processing tasks. This model achieved the best performance on multiple speech datasets.

In a typical speech emotion recognition system, audio data, feature extraction, classification models, and emotional output recognition are all included. Some of the popular classification methods right now for an emotion recognition system include SVM (Sonawane, Inamdar, & Bhangale, 2017), Hidden Markov Model (HMM) (Starner & Pentland, 1995), and Recurrent Neural Network (RNN) (Chamishka, et al., 2022). Speech emotion recognition tasks require an emotion speech database for training the model. In this study, three datasets were utilized for the purpose of human emotion classification. The CREMA-D (Crowd-Sourced Emotional Multimodal Actors Dataset) (Cao, et al., 2014) dataset is used for human emotion classification which includes 7,442 clips of 91 actors, with each actor performing a set of basic emotions (anger, disgust, fear, happiness, sadness, surprise) as well as neutral expressions with a distribution of 48 male and 43 female actors coming from a variety of races and ethnicities (African America, Asian, Caucasian, and Hispanic). The RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song) (Livingstone & Russo, 2018) dataset contains speech and song recordings from 24 professional actors (12 male and 12 female) who performed various emotional expressions. The dataset covers eight basic emotions: calm, happy, sad, angry, fearful, surprised, disgusted, and neutral. The RAVDESS dataset provides both audio and visual data, making it suitable for multimodal emotion recognition studies. The SAVEE (Surrey Audio-Visual Expressed Emotion) (Jackson & Haq, 2015) dataset consists of speech recordings from four male actors who portrayed seven different emotions: anger, disgust, fear, happiness, sadness, surprise, and neutral. The SAVEE dataset is specifically designed for speech-based emotion recognition tasks and provides a valuable resource for studying the acoustic features associated with different emotional expressions.

2.2 Basic Theory

This section will explain the basic theory used as a reference in this study. Among other things, this chapter will explain the literature review, human emotion, voice understanding, speech recognition, feature extraction, neural network convolution, and transformers, as well as a brief explanation of the framework library, used to implement emotion detection in the

human voice in this study, namely PyTorch, touching upon its advantages and capabilities in implementing the emotion detection model and highlighting its suitability for developing and training neural networks for speech-based emotion recognition tasks.

2.2.1 Emotion

Emotion is an aspect of consciousness which are generally understood to represent the synthesis of subjective experience, expressive behavior, and neurochemical activity. Most researchers consider them to be part of the evolutionary legacy of the human species and serve adaptive purposes by supplementing common perception and facilitating social communication. (Solomon, 2009) Emotions come in a variety of forms, and they all have an impact on how humans live and relate to each other. There are times when we may feel as though these emotions are controlling us. Our actions, behaviors, and perceptions are all influenced by the emotions we are experiencing at any given time. According to (Cherry, 2021), psychologist Paul Eckman identifies six fundamental emotions that were shared across all human societies in the 1970s. These emotions include *happiness, sadness, disgust, fear, surprise*, and *fury*. Later, he expanded this list for *pride, humiliation, embarrassment, and enthusiasm*. Figure 2.1 depicts various human emotions nowadays.

Figure 2.1: Human Emotions (Charlie, 2014).

2.2.2 Sound

Sounds are produced by sound waves. Humans could hear it by passing a medium through the ears. All sound is produced by the vibration of molecules. For example, when a person makes a sound, there are vibrations that move through the air molecules. Sound waves travel away from where they originate. When these vibrating air molecules reach the ear, the eardrum also vibrates. The bones in the ear vibrate as if the object that generated the sound waves vibrates. There are three types of continuous mediums which are solids, liquids, and gases. Sound travels faster through a solid medium since the particle here is closer together than in gases or liquid medium. These vibrations let humans hear different things such as music. There are also irregular vibrations called noises. Human beings could make very complex sounds used for talking. A sound wave is a longitudinal wave that has two parts (Compression and Rarefaction). Compression is where air molecules are pushed together. Rarefaction is where the molecules are far apart. Sound is produced by a series of mechanical compressions and

rarefactions of mechanical waves that sequentially propagate through a medium (StudyCorgi, 2022). Figure 2.2 shows a representation of the longitudinal nature of sound waves.

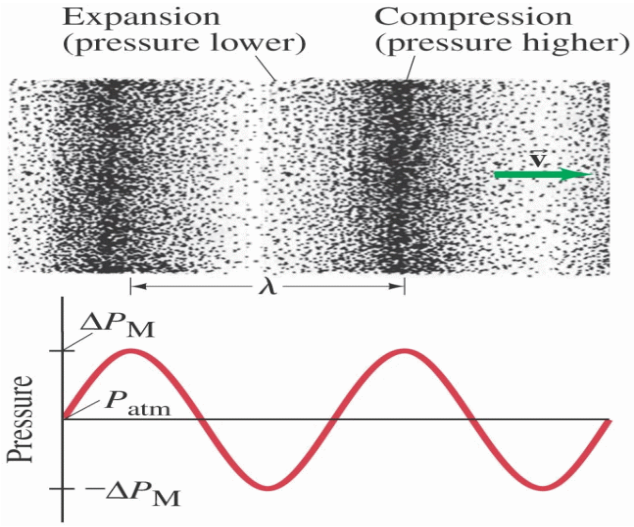


Figure 2.2: Longitudinal Nature of Sound Wave (StudyCorgi, 2022).

2.2.3 Speech Recognition

Speech Recognition is an interdisciplinary subject of computer science and computational linguistics that develops approaches and technology to enable the translation of spoken language into text by computer machines with the main benefit of searchability. It is often referred to as computer voice recognition or automatic speech recognition (ASR). Speech recognition draws on expertise and research from the domains of computer science, linguistics, and computer engineering. Speech recognition systems use computer algorithms to process**,** interpret**,** and convert spoken words into text. A software program converts the sounds picked up by the microphone into characters that computers and humans can understand**.** This program must be able to adapt to the highly variable and context-specific nature of human speech. The software algorithms that process human speech are trained on a variety of speech patterns, speaking styles, language, accents, and idioms. The software also separates speech from the background noises that often accompany the signals (Yu & Deng, 2015).

2.2.4 Feature Extraction

In machine learning, feature extraction is the process of turning raw data into numerical features that can be processed while keeping the information in the original dataset. The amount of redundant data in the dataset is decreased within this process. In the end, the data reduction speeds up the learning and generalization phases of the machine learning process while also enabling the model to be built with less computation power. This study employs one of the most popular feature extraction methods in the context of Speech Emotion Recognition (SER) called the Mel-Frequency Cepstral Coefficient (MFCC) (Kishore & Satish, 2013). The procedure to find MFCCs is mainly with the following steps shown in Figure 2.3:

a. Pre-Emphasis

The design structure of a voice production system causes dampening in high-frequency regions. Pre-Emphasis amplifies high-frequency sections and conducts filtering which is used to offset the spectrums of voiced regions. Widely used pre-emphasis filter is given in Equation 2.1,

$$y[n] = x[n] - a * x[n-1], a \approx 0.95 - 0.97. \quad (2.1)$$

Where:

- $y[n]$ is the output signal at time n.
- $x[n]$ is the input signal at time n.
- a is the pre-emphasis coefficient.
- $x[n-1]$ is the input signal at the previous time step $(n-1)$.

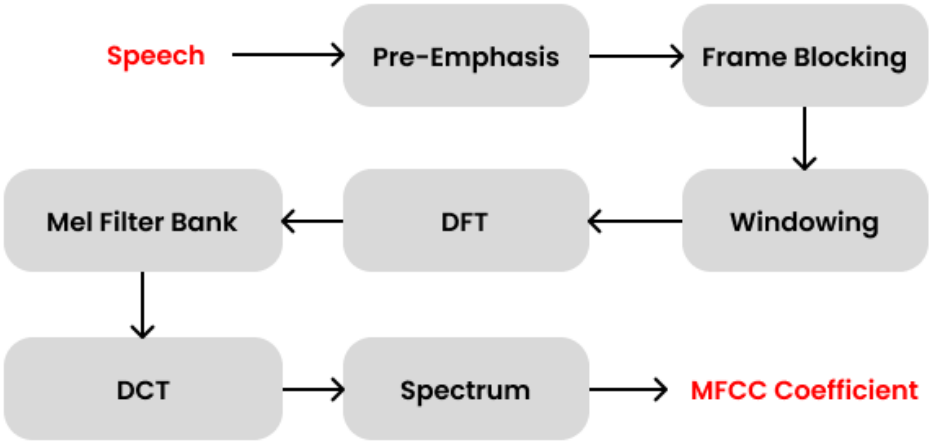


Figure 2.3: MFCC Block Diagram.

b. Frame Blocking

Due to voice signal as a slow time-varying signal, speech analysis over a short enough time span is required for stable acoustic features. Frame blocking entails processing the voice signal at short time intervals to extract the characteristic features in a more stable condition.

c. Windowing

Windowing is the process of splitting an audio signal into segments of specific lengths. This reduces the effect of aliasing or signal discontinuity at the beginning and end of each frame that could occur due to the frame-blocking process.

d. Discrete Fourier Transform (DFT)

Discrete Fourier Transform is one of the most powerful tools in digital signal processing which enables us to find the spectrum of a finite-duration signal. In MFCC, DFTs are used to convert each windowed frame into a magnitude spectrum with Equation 2.2,

$$X(k) = \sum_{n=1}^{\infty} x(n) e^{-j2\pi kn/N}. \quad (2.2)$$

Where:

- $X(k)$ is the k^{th} frequency domain sample, with k ranging from $0\ to\ N-1$.
- $x(n)$ it the n^{th} time domain sample, with n ranging from $0\ to\ N-1$.

- N it the number of samples in the sequence.
- j is the imaginary unit $(sqrt(-1))$.
- π is the mathematical constant $(3.1415\ldots)$.

e. Mel-Frequency Warping

In this process block, the triangle waves that make up the Mel filter bank f frequency in Hz units are used to create the signal. As a result, using this method, the signal value in $M(f)$ frequency units is determined. The MFCC coefficient value is determined by the number of filters in the Mel filter bank. The Mel scale is a nonlinear scale that compresses the higher frequencies, which are more difficult for humans to perceive. The algebraic equation for the process of converting Mel spectrum and FFT frequency values in Hz to Mel frequency units is defined in Equation 2.3 as:

$$M(f) = 2595.\log_{10}(1 + \frac{f}{700}). \quad (2.3)$$

Where:

- $M(f)$ is the frequency of Mel.
- f is the frequency in Hz.
- log_{10} is the logarithm base 10.

f. Discrete Cosine Transform (DCT)

A DCT is applied to the transformed Mel frequency coefficients to produce a set of cepstral coefficients. The Mel spectrum was represented on a log scale which results in a signal in the cepstral domain with frequency peaks corresponding to the pitch on the signal. Since most of the signal information is represented by the first few MFCC coefficients, the system can be made robust by extracting only those coefficients ignoring higher-order DCT components.

g. Mel Cepstrum

The final result of the MFCC block process shown in Figure 2.3 is the coefficient of the Mel frequency cepstrum. A cepstrum representation of the speech spectrum adequately represents the local spectral characteristics of the signal for a given frame analysis.

2.2.5 Convolutional Neural Networks (CNN)

Convolutional neural networks are a subset of deep learning techniques that have gained prominence in several computer vision applications such as LeNet (LeCun, Bottou, Bengio, & Haffner, 1998) and AlexNet (Krizhevsky, Sutskever, & Hinton, 2012), and are generating attention in many different fields, including speech recognition. CNN was intended to learn spatial hierarchies of characteristics automatically and adaptively, from low to high-level patterns. CNN is a mathematical construct that is usually composed of three types of layers including convolution, pooling, and fully connected layers. Compared to the traditional hand-crafted feature extraction techniques, CNN is far more data-hungry because of its millions of learnable parameters to estimate and is more computationally expensive, resulting in requiring graphical processing units (GPUs) for model training (Albawi, Mohammed, & Al-Zawi, 2017). Figure 2.4 shows a general view of how layers are connected inside a CNN architecture, illustrating how the various layers are interconnected.

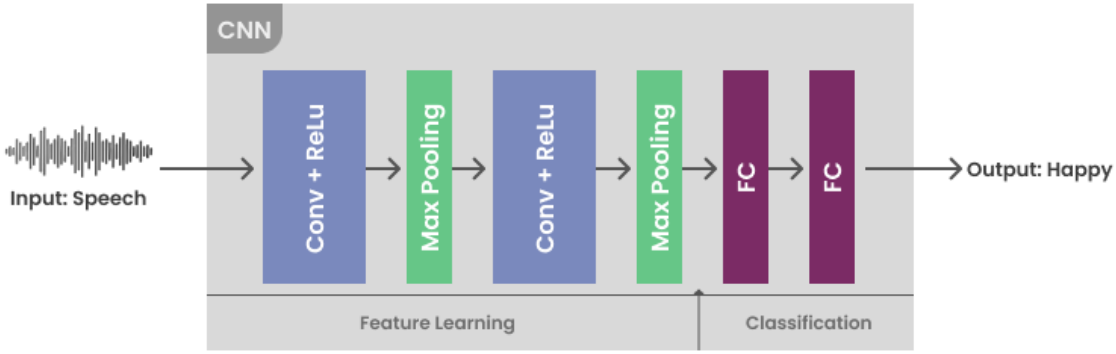


Figure 2.4: CNN Architecture.

a. Convolution

Convolution is a special type of linear operation used in feature extraction, where small numerical arrays (kernels) are applied to the input. This is an array of numbers called a tensor. The element-wise product between each element of the kernel and the input tensor is computed at each position of the tensor and summed to get the output value at the corresponding position of the output tensor, called a feature map, depicted in Figure 2.5. This process is repeated by applying multiple kernels to form any number of feature maps representing different properties of the input tensor. Therefore, different kernels can be viewed as different feature extractors. Two important hyperparameters that define the convolution operation are the size and number of kernels.

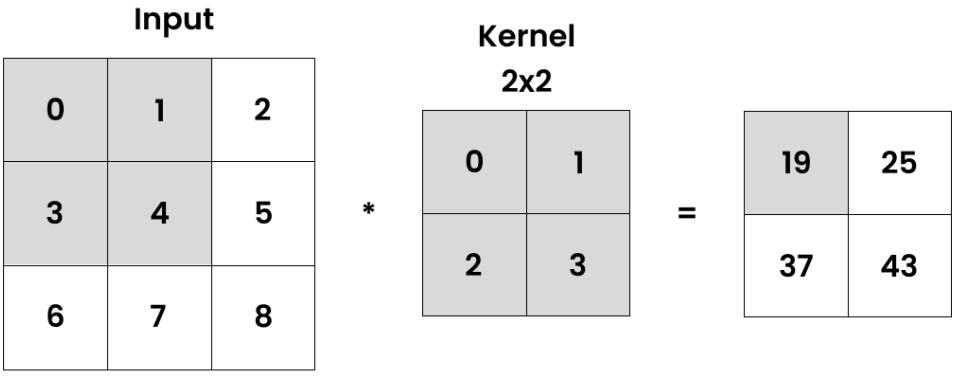


Figure 2.5: 2 × 2 Convolution Filter.

b. Activation Function (AF)

The activation function is the node that is added at the end of each output of the neural network. In the CNN architecture, the activation function is the final calculation of the feature map output, or the generation of feature patterns after the convolution or merging calculation process. Although smooth nonlinear functions like the *sigmoid* or *hyperbolic tangent* (tanh) function have been employed in the past because they are mathematical representations of the behaviour of biological neurons, the present study focuses on utilizing the rectified linear unit (ReLU) (Nair & Hinton, 2010) and the exponential linear unit (ELU). The ReLU has become the state-of-the-art AF due to its simplicity and improved performance. However, the major problem with ReLU-based AF is the under-utilization of negative values leading to a vanishing gradient. In order to cope with these limitations, exponential function-based AFs such as ELU were introduced which utilize the negative values with the help of the exponential function. (Dubey, Singh, & Chaudhuri, 2021) ReLU is a simple function which is the identity function for positive input and zero for negative input presented in Equation 2.4. On the other hand,

Equation 2.5 illustrates the formulation of ELU.

$$f(x) = \max(0, x) = \begin{cases} x, & if\ x \geq 0 \\ 0, & otherwise \end{cases}. \tag{2.4}$$

Where:

- $f(x)$ is the output of the function.
- x is the input to the function.

$$f(x) = \begin{cases} x, & x > 0 \\ \alpha \times (e^x - 1), & x \leq 0 \end{cases}. \tag{2.5}$$

Where:

- $f(x)$ is the output of the function.
- x is the input to the function.
- α is a positive constant controlling the slope of the function for negative inputs.

c. Max Pooling

A pooling layer offers a standard down-sampling method that lowers the feature map in-plane dimensions to introduce translation invariance to slight shifts and distortions and limit the number of ensuing learnable parameters. One of the most popular types of pooling operations is max pooling. The idea behind max pooling is that it preserves the most important information from the input while discarding less important information. This can be particularly useful for classification tasks, where the max pooling layer can help the model focus on the most important features in an audio, such as spectral peaks, spectral roll-off points, and spectral flux. Figure 2.6 shows an example of a max pooling with 2×2 filter on a 4×4 feature map.

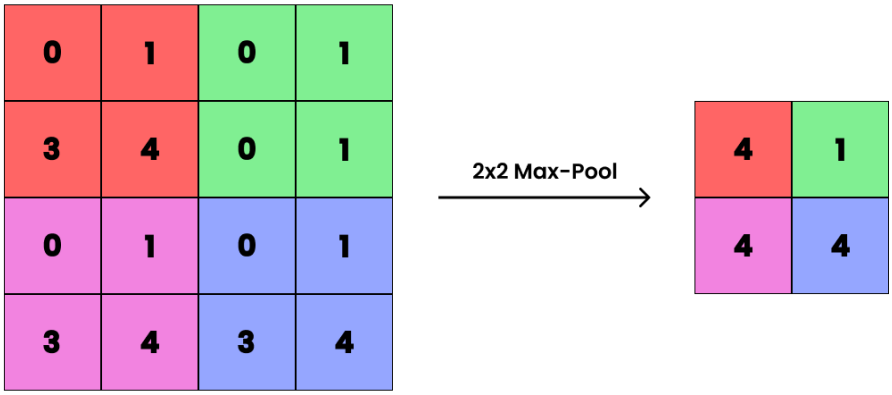


Figure 2.6: 2×2 Max Pooling Layer.

d. Fully Connected Layer

Feature maps generated from the feature extraction layers are still in the form of a multidimensional array. Therefore, these feature maps are typically flattened, or converted into a one-dimensional array of vectors, and connected to one or more fully connected layers, also known as dense layers, in which each input is connected to their outputs by learnable weight resulting in probabilities for each class in the classification tasks. After passing through the fully connected layers, the final layer uses the SoftMax activation function that normalizes real values output from the last fully connected layer to get probabilities of the input being in a particular class (classification) where each value ranges between 0 and 1. The final fully

connected layer usually has as many output nodes as there are classes. This arrangement ensures that the network can produce probability estimates for each class, enabling accurate classification predictions based on the input data.

e. SoftMax

The softmax function is a function that turns a vector of N real values into a vector of N real values that sums to 1. The input values can be positive, negative, zero, or greater than one, but the softmax transforms them into values between 0 and 1 so that they can be interpreted as probabilities. This activation function is particularly useful in multiclass classification problems where the objective is to assign an input sample to one of several predefined classes. This function applies an exponential operation to each element of the input vector and then normalizes the resulting values to ensure they sum up to one. This normalization property makes softmax well-suited for classification tasks, as it allows the model to interpret the output values as class probabilities.

Mathematically, given an input vector $z = [z_1, z_2, \dots, z_n]$, the softmax function computes the output vector $y = [y_1, y_2, \dots, y_n]$, where:

$$y_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}. \tag{2.6}$$

Equation 2.6 above presents the softmax formula where e^{z_i} represents the exponential function applied element-wise to the input vector. The denominator $\sum_{j=1}^{n} e^{z_j}$ sums up the exponentiated values of all elements in the input vector, ensuring the normalization property.

2.2.6 Transformer

The transformer is a deep learning model architecture that is built entirely on the self-attention mechanism to weigh the importance of each part of the input data differently. It is mainly used in the fields of natural language processing (NLP). This architecture is designed to process sequential input data to solve NLP-related tasks such as text translation or summarization. However, unlike Recurrent Networks such as GRU (Gated Recurrent Unit) and LSTM (Long Short-Term Memory), transformers could process the entire input at once. Attention mechanisms provide context for each position in the input sequence which allows for more parallelization than recurrent neural networks and therefore reduces training time. The model of the transformer architecture follows the overall architecture of Figure 2.7 using stacked self-attention and pointwise fully connected layers for both the encoder and decoder shown in the left and right halves of the figure respectively (Vaswani, et al., 2017).

a. Self-Attention

In artificial neural networks, attention is a technique designed to mimic cognitive attention. This effect improves some parts of the input data and reduces others. The motivation for this is that networks need to pay more attention to small but important pieces of data. Learning which parts of the data are more important than others is context-dependent, which is trained by gradient descent. Attention functions can be described as associating a query and a set of key-value pairs with an output. Where query, key, value, and output are all vectors. The output is computed as a weighted sum of the values. The weight assigned to each value is calculated by the query compatibility function using the appropriate key.

Self-attention, also called intra-attention, is an attention mechanism that associates different positions of a single sequence to compute representations of the same sequence. In a self-attention mechanism, each element in the input is represented as a vector, and the model learns

a set of attention weights that determine how much importance each element should be given when producing the output. The attention weights are learned through training and allow the model to selectively focus on certain parts of the input while ignoring others. Self-attention has proven especially useful for machine reading, summarizing summaries, or generating image descriptions.

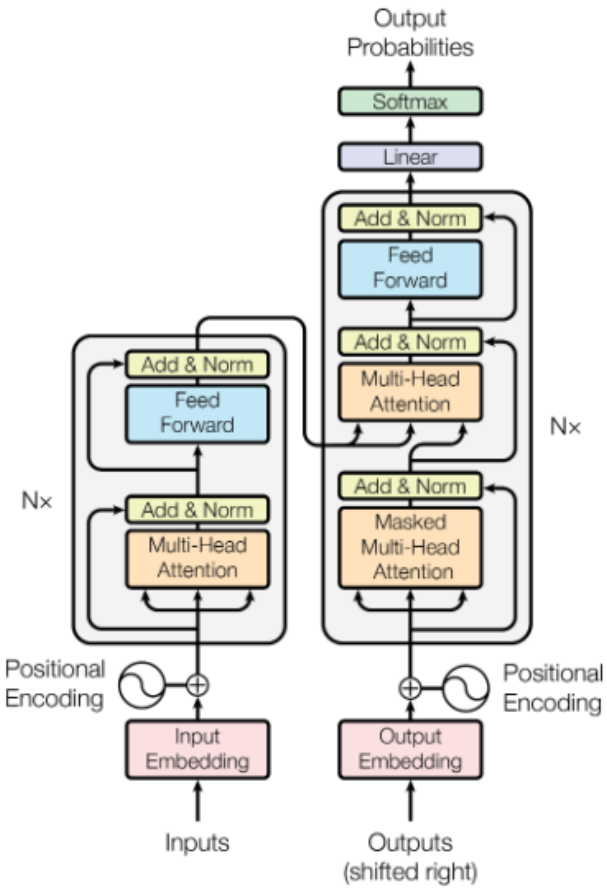


Figure 2.7: Transformer Architecture (Vaswani, et al., 2017).

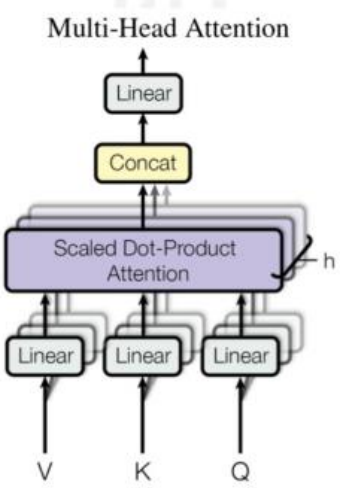


Figure 2.8: Multi-Head Attention (Vaswani, et al., 2017).

b. Multi-Head Self Attention

In Transformer, the Attention module iterates its computation several times in parallel. Each of them is called an attention head. The Attention module splits its query, key, and value parameters N times, passing each split individually through a separate head. All these similar attention calculations are combined to produce a final attention score. This is called multi-headed attention and gives the Transformer greater power to encode multiple relationships and nuances for each word. Multi-head attention allows the model to jointly pay attention to information from different representational subspaces at different positions. In most general form, the multi-head attention mechanism can be represented as shown in Equation 2.7. Figure 2.8 shows that a multi-head attention consists of several attention layers running in parallel.

$$MultiHead(Q, K, V) = concat(head_1, head_2, \ldots, head_h) * W^O. \quad (2.7)$$

Where:

- $Q, K, and\ V$ are matrices of queries, keys, and values respectively.
- $head_1, head_2, \ldots, head_h$ are the attention maps computed by the h different attention heads.
- W^O is a learned projection matrix.
- $concat$ is a function that concatenates the attention maps along the second dimension.

Each attention head computes an attention map using Equation 2.8 below:

$$head_h = attention\left(QW_h^Q, KW_h^K, VW_i^V\right). \quad (2.8)$$

Where:

- $W_i^Q,\ W_i^K,\ and\ W_i^V$ are learned projection matrices for the h^{th} attention head.

c. Scaled Dot-Product Attention

Transformers implement scaled dot product attention depicted in Figure 2.9, that follows the steps of the general attention mechanism. Scaled dot product attention first computes the dot product of each query and every key. Then divide each result by $\sqrt{d_k}$ and apply the softmax function. In doing so, it obtains the weights that are used to scale the values. The formula for scaled dot product attention was defined below in Equation 2.9 as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \quad (2.9)$$

Where:

- $Q, K, and\ V$ are matrices of queries, keys, and values respectively.
- QK^T is the dot product of the queries and keys.
- d_K is the dimensionality of the keys.
- $softmax$ is the SoftMax function, which normalizes the attention weights.

In practice, the computations performed by scaled dot product attention can be efficiently applied to the entire set of queries at once. For this purpose, the matrices $Q, K, and\ V$ are

supplied as inputs to the attention function. The scaling factor $1/\sqrt{d_k}$ is included to help stabilize the attention weights and improve the numerical stability of the model.

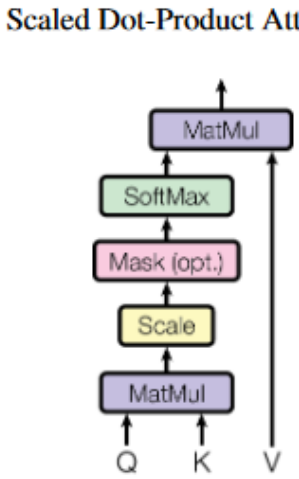


Figure 2.9: Scaled Dot-Product Attention (Vaswani, et al., 2017).

d. Encoder

Figure 2.10 shows the two main components of an encoder block: the self-attention mechanism and a feed-forward neural network. The self-attention mechanism accepts an input encoding from previous encoders and weighs their relevance against each other to produce an output encoding. Then, a feed-forward neural network processes each output code independently. To maintain information flow and aid in gradient propagation, residual connections and normalization layers are employed in each sub-layer, ensuring the stability and effectiveness of the overall encoder block architecture.

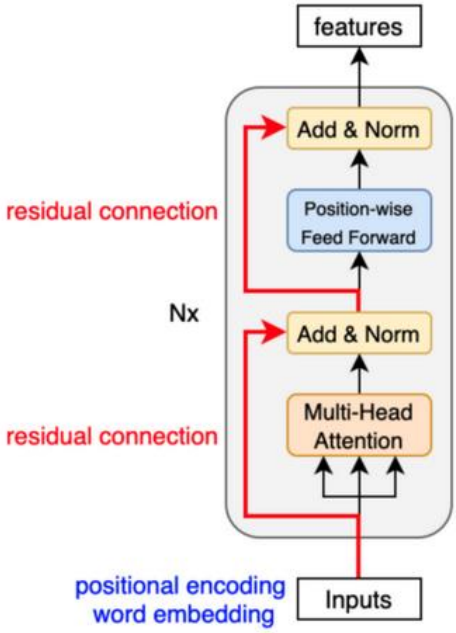


Figure 2.10: Encoder Block (KiKaBeN, 2021).

e. Decoder

The decoder block takes two main encoder components of a self-attention mechanism and a feed-forward neural network and inserts a third sub-layer that performs multi-head attention

over the output of the encoder stack, shown in Figure 2.11. This new sub-layer obtains relevant information from the encoding produced by the encoder block. Like the encoder block, each sub-layer employs a residual connection and a normalization layer.

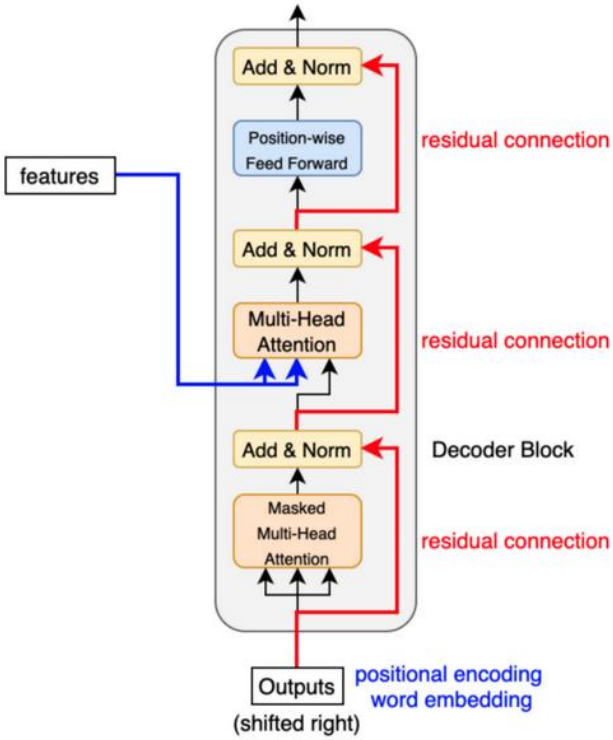


Figure 2.11: Decoder Block (KiKaBeN, 2021).

2.2.7 PyTorch

PyTorch is an open-source machine learning framework based on the Python programming language and the torch library. It is developed primarily by the Meta AI research team and can be used in both Python and C++ programming languages. However, this framework works best with Python. Over 200 and more different mathematical operations are supported by the PyTorch framework and its popularity is still growing because it makes building models for artificial neural networks simpler. Researchers primarily utilize PyTorch for research and applications using artificial intelligence.

Because of the pythonic nature of this framework, PyTorch is able to utilize core python concepts such as classes, structures, and conditional loops making it easy and intuitive to understand. PyTorch is also popular for its dynamic computation graphs, which allow greater flexibility in building complex architectures. This allows neural network developers and scientists to run and test pieces of code in real-time, rather than waiting for the entire program to be written (Paszke, et al., 2019).

2.2.8 Support Vector Machine (SVM)

Support Vector Machines (SVMs) are a type of supervised learning algorithm used for classification and regression analysis. Introduced by (Vapnik & Cortes, 1995) as a binary classifier that could solve two-group classification problems with high accuracy, SVMs have gained popularity due to their ability to handle both linearly and non-linearly separable datasets and their effectiveness in high-dimensional feature spaces.

SVMs work by finding the hyperplane that maximally separates the two classes in the input data. The hyperplane is selected based on the margin, which is the distance between the

hyperplane and the closest data points from each class. The SVM algorithm aims to find the hyperplane that maximizes this margin. The data points closest to the hyperplane are called support vectors, and they are used to define the hyperplane. As SVMs evolved, two main techniques were proposed to enable their use for multi-class classification in a one-vs-all and one-vs-one method (Duan, Rajapakse, & Nguyen, 2007).

In one-vs-all classification, each class is treated as a binary classification problem. A separate SVM model is trained for each class, where the samples of that class are assigned a positive label, and all other samples are assigned a negative label. In one-vs-one classification, all possible pairs of classes are created, and a separate SVM model is trained for each pair. During testing, each sample is classified by each SVM model, and the class with the most votes is assigned to the sample.

Both methods have their advantages and disadvantages. one-vs-all is more straightforward to implement and can handle imbalanced datasets. However, it may not be as accurate as one-vs-one, particularly when the number of classes is large. one-vs-one is more accurate and can handle overlapping classes, but it requires training a large number of SVM models, making it more computationally expensive.

2.2.9 Convolutional Recurrent Neural Network (CRNN)

Convolutional Recurrent Neural Networks (CRNN) have emerged as a powerful architecture in the field of deep learning, combining the strengths of both CNN and RNN models. CRNNs are particularly well-suited for tasks that involve sequential data and require the extraction of meaningful features from input images. With the ability to capture spatial information through convolutional layers and model temporal dependencies through recurrent layers, CRNNs have demonstrated remarkable success in various domains, including computer vision, natural language processing, and speech recognition. The network architecture of CRNN, as shown in Figure 2.12, consists of three main components, including the convolutional layers, the recurrent layers, and the transcription layer (Shi, Bai, & Yao, 2017).

a. Feature Sequence Extraction

In the CRNN architecture, the convolutional layers are responsible for extracting meaningful features from the input sequence. These convolutional layers employ filters to perform local receptive field operations across the sequential data, such as audio signals or texts. By convolving the filters over the input sequence, the convolutional layers capture relevant patterns and structures at different scales. The hierarchical nature of the convolutional layers allows them to learn increasingly abstract and complex features as the layers progress deeper into the network. This process enables the CRNN model to extract both spatial and temporal information from the input sequence, effectively capturing the salient features necessary for subsequent analysis and classification tasks.

This feature extraction process also encodes the temporal aspects of the input sequence. By leveraging the local receptive fields of the convolutional layers, the model can detect temporal patterns and dependencies present in the sequential data. The convolutional layers process the input sequence in a sliding window manner, extracting local features that are influenced by the neighbouring elements in the sequence. Through repeated convolutions, the CRNN model learns to encode sequential dependencies and capture long-range temporal information. This temporal feature encoding is essential for tasks that require an understanding of the context and temporal dynamics of the input sequence, such as speech recognition or natural language processing. By combining spatial and temporal feature extraction, the CRNN model creates a powerful representation of the input sequence that can be further utilized for sequence labelling or transcription tasks.

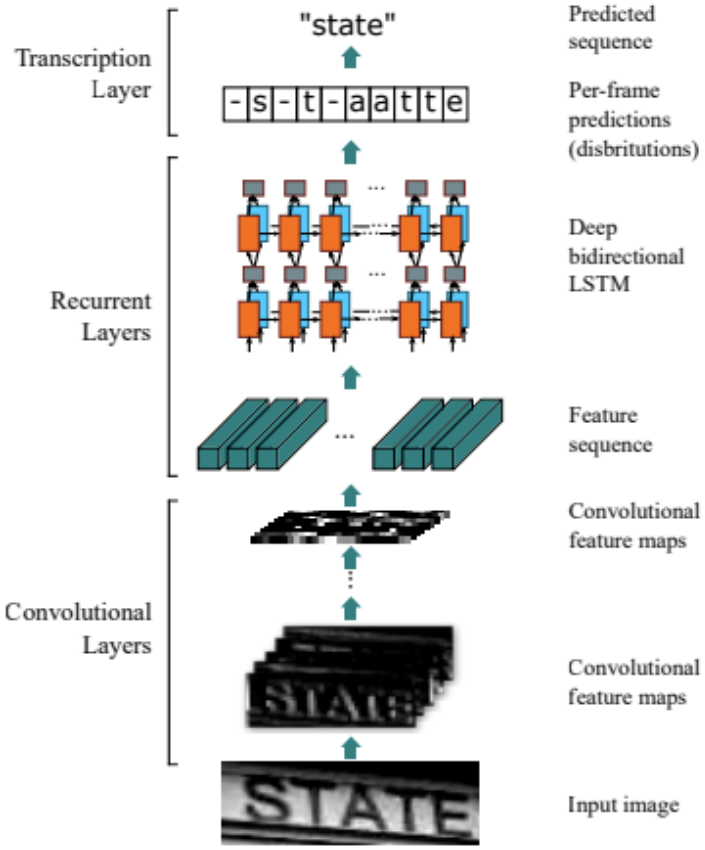


Figure 2.12: CRNN Architecture (Shi, Bai, & Yao, 2017).

b. Sequence Labeling

Sequence labelling is a fundamental task in various applications, such as speech recognition and part-of-speech tagging. In the CRNN model, sequence labelling is achieved through the integration of recurrent layers, such as the bidirectional Long Short-Term Memory (BLSTM) layer (Graves, Mohamed, & Hinton, 2013). Unlike unidirectional recurrent layers, which process the input sequence in one direction, bidirectional layers, as depicted in Figure 2.13 process the sequence in both directions simultaneously. This enables the model to capture not only the past context but also the future context of each element in the sequence. By maintaining hidden states and utilizing gating mechanisms in both the forward and backward directions, the bidirectional layers effectively model the context and sequential information present in the feature sequence.

The bidirectional layers capture the dependencies and relationships between elements in both the past and future contexts. This enhanced temporal contextual understanding is particularly valuable for sequence labelling tasks, where the labels assigned to each element often depend on both the preceding and succeeding context. The bidirectional layers allow the model to consider the entire sequence up to a given time step, enabling it to make informed predictions based on the accumulated contextual information from both directions. This integration of bidirectional recurrent layers further improves the model ability to handle sequential data, capturing the nuanced relationships between elements and enhancing the accuracy of the sequence labelling task. By leveraging the learned temporal dependencies and contextual understanding from both directions, the CRNN model can effectively label each element in the input sequence, facilitating tasks such as speech recognition, where accurately assigning labels to individual audio frames is critical.

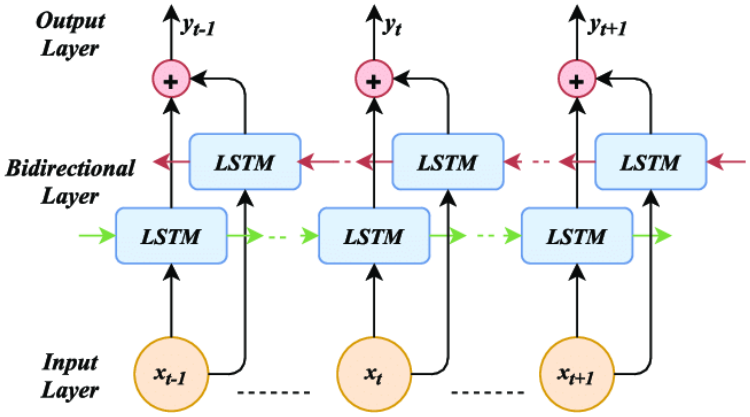


Figure 2.13: Bidirectional LSTM (Ihianle, et al., 2020).

c. Transcription

The transcription stage in the CRNN model converts the learned representations from the preceding layers into meaningful output predictions. After the feature sequence has been processed by the convolutional and recurrent layers, it is passed through a transcription layer. The transcription layer typically consists of one or more fully connected layers followed by an activation function, such as softmax. These fully connected layers map the encoded features to the desired output space, enabling the model to make predictions based on the learned representations. The softmax activation function is used to transform the real-valued outputs into a probability distribution over the possible classes or labels. This normalization ensures that the predicted probabilities sum up to one, allowing for easier interpretation and decision-making in classification tasks. The transcription layer acts as the final stage of the CRNN model, transforming the encoded feature sequence into interpretable and actionable output prediction.

2.2.10 Confusion Matrix

Ground Truth / Model Predict	Positive	Negative
Positive	True Positive (TP)	False Positive (FP)
Negative	False Negative (FN)	True Negative (TN)

Figure 2.14: Confusion Matrix Table.

The confusion matrix, also known as an error matrix, is a fundamental tool in the field of machine learning and statistical analysis. It is primarily used to evaluate the performance of a classification model by providing a comprehensive summary of the model predictions. The matrix is constructed based on the actual and predicted class labels assigned by the model to a given dataset. By visually representing the outcomes of the classification process, an individual could determine the accuracy of the model by observing the diagonal values for measuring the number of accurate classifications (Karimi, 2021).

Figure 2.14 illustrates the configuration of a confusion matrix, which is typically presented as a square matrix. In this representation, the rows correspond to the predicted class labels,

while the columns represent the actual class labels. The diagonal elements of the matrix denote the correctly classified instances as true positives (TP) and true negatives (TN), while the off-diagonal elements signify the misclassified instances as false positives (FP) and false negatives (FN). Each entry within the confusion matrix represents the frequency or count of a specific combination of true and predicted class labels. The confusion matrix can be employed in binary classification scenarios, involving two classes, or extended to accommodate multi-class classification tasks by incorporating additional rows and columns.

a. Accuracy

From the confusion matrix, several performance metrics can be derived to assess the effectiveness of model classification. One such metric is accuracy, which measures the overall correctness of the predictions by calculating the ratio of correctly classified instances to the total number of instances. Equation 2.10 is used to calculate this metric of evaluation.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%. \tag{2.10}$$

b. Precision

Precision focuses on the performance of the model for each individual class which evaluates the proportion of true positive predictions out of all positive predictions as depicted in Equation 2.11. Precision is a useful metric in a case where a false positive is a higher concern than a false negative.

$$Precision = \frac{TP}{TP + FP} \times 100\%. \tag{2.11}$$

c. Recall

Recall assesses the proportion of true positives out of all actual positive instances. It showed how many of the actual positive cases were correctly predicted by the model as shown in Equation 2.12. A higher recall means that most of the positive cases will be labelled as true positives. On the other hand, low recall means that the model has a high number of false negatives.

$$Recall = \frac{TP}{TP + FN} \times 100\%. \tag{2.12}$$

d. F1-score

The F1-score combines precision and recall into a single measure which provides a balance between precision and recall, considering both false positives and false negatives as presented in Equation 2.13. Having a precision or recall value of 0 is not desirable and hence it will give us the F1 score of 0 (lowest). On the other hand, if both the precision and recall value is 1, it'll give us the F1 score of 1 indicating perfect precision-recall values. All the other intermediate values of the F1 score range between 0 and 1.

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \times 100\%. \tag{2.13}$$

2.2.11 Librosa

Librosa is a Python library for audio signal processing and music analysis. It provides a comprehensive set of tools and functions that enable researchers and developers to extract

meaningful information from audio signals. Librosa offers a wide range of audio preprocessing capabilities, including audio loading, feature extraction, spectral analysis, and audio manipulation. Librosa leverages the power of the Fast Fourier Transform (FFT) to convert audio signals from the time domain to the frequency domain. This transformation allows for a detailed analysis of the spectral content of the audio, revealing information about the underlying frequencies and amplitudes.

One of the key features of Librosa is its ability to extract a wide range of audio features that capture different characteristics of the sound. These features include MFCCs, spectral contrast, chroma feature, and rhythmic features. MFCCs are used in the present study for speech signal analysis tasks and provide a compact representation of the spectral envelope of an audio signal. By capturing the distribution of frequencies, MFCCs enable the extraction of valuable insights into the audio content. Their application extends to various tasks in music information retrieval and audio analysis, serving as a foundation for tasks such as emotion recognition and genre classification. Additionally, Librosa supports time-frequency representations such as the spectrogram, which visualizes the changing spectrum of an audio signal over time. This representation is particularly useful for tasks like onset detection, beat tracking, and audio segmentation. By analyzing the spectrogram, important events and structures can be identified in the audio signal and extract temporal information that can be used for further analysis and processing.

Librosa provides a user-friendly and efficient framework for analyzing and manipulating audio signals. Its comprehensive set of features and functions, along with its integration with other Python libraries such as NumPy and SciPy, makes it a valuable tool in the field of audio signal processing such as emotion classification, speech processing, and sound analysis (McFee, et al., 2015).

2.2.12 Normalization

Normalization is an approach which is applied during the preparation of data in order to change the values of numeric columns in a dataset to use a common scale when the features in the data have different ranges. This technique can be helpful in the context of prediction or classification tasks, where multiple approaches can yield significantly different results. Normalization brings these diverse predictions or forecasts closer together, ensuring consistency and reducing the influence of large variations (Gopal Krishna Patro & Sahu, 2015).

One commonly used technique is Standard Scaler (Pedregosa, et al., 2023), also known as Z-score normalization. Standard Scaler is applied as a preprocessing step to transform the input data into a standardized scale. This technique ensures that the features have zero mean and unit variance, resulting in a distribution that is centred around zero and has a consistent scale. Equation 2.14 present the formula for Standard Scaler given by:

$$z = \frac{(x - u)}{s}. \tag{2.14}$$

Where:

- z represent the normalized value of the variable.
- x is the original value.
- u is the average/mean of the variable.
- s is the standard deviation of the training samples.

Additionally, another normalization technique that could speed up the training process of

deep learning architectures explored in the present study is Batch Normalization (Ioffel & Szegedy, 2015). Unlike Standard Scaler, Batch Normalization operates within the network itself, normalizing the activations at each layer during the training process. It addresses the issue of internal covariate shift, which refers to the change in the distribution of network activations as the parameters of the previous layers change during training. Batch Normalization performs normalization on mini-batches of data, calculating the mean and variance for each feature independently. The normalized values are then obtained by applying a linear transformation to the mini-batch. Equation 2.15 showed the formula for Batch Normalization as follows:

$$y = \frac{x - \mu_{batch}}{\sqrt{\sigma_{batch}^2 + \epsilon}} \times \gamma + \beta. \tag{2.15}$$

Where:

- y represent the normalized output.
- x is the input value.
- μ_{batch} and σ_{batch} are the mean and standard deviation of the mini-batch, respectively.
- ϵ is a small constant added for numerical stability.
- γ and β are learnable parameters that allow the network to adapt and scale the normalized values.

The inclusion of both Standard Scaler and Batch Normalization techniques is instrumental in the preprocessing and optimization of data for deep learning architectures. By utilizing Standard Scaler as a preprocessing step, the input data is scaled to a suitable range, allowing for fair comparisons, and preventing dominant features. Simultaneously, the integration of Batch Normalization within the network guarantees that the activations within the model are properly normalized, fostering faster convergence and improved training stability.

Chapter III
Methodology

This chapter will provide an overview of the Pararel Transformer Encoder with CNN Architecture for this study, including the tools and techniques that will be used, as well as plans for implementation and testing.

3.1 Designed Method

This section provides a comprehensive summary of the Pararel Transformer Encoder with CNN architectural model functionality, outlining its key components and mechanisms. The architectural model, devised specifically for the audio emotion recognition task, aims to optimize performance and accuracy in accurately identifying and classifying emotions within audio data. The model architecture consists of two parallel blocks: a CNN block and a Transformer encoder block. To give a clear understanding of the model structure, a detailed diagram is presented in Figure 3.1, illustrating the interconnections and flow of information within the architecture, offering an intuitive glimpse into the intricate workings of the model and aiding in the comprehension of subsequent sections.

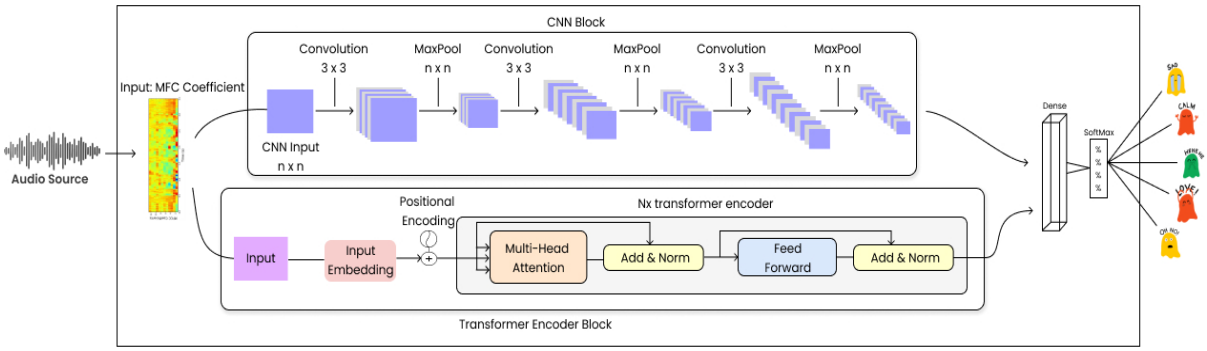


Figure 3.1: Model Architecture.

The CNN architecture in this study is based on recent advancements in image and sequence processing. It includes a series of convolutional and pooling layers, similar to the classic LeNet architecture (LeCun, Bottou, Bengio, & Haffner, 1998), which extract features from the input data and reduce the size of the feature maps through downsampling. The fully-connected layers then process the extracted features to produce the final output of the network, which is transformed into a probability distribution over the possible classes using the SoftMax function. While LeNet is a relatively simple architecture compared to modern CNNs, it has been successful in many classification tasks and has been applied in various domains such as handwritten digit recognition.

The Transformer architecture is precisely as (Vaswani, et al., 2017). However, in this study, only the encoder blocks are employed which is a component of the Transformer architecture that was introduced in the paper. It is used to process the input sequence and extract relevant features that will be passed to the fully-connected layers, which process these features to produce the final output of the network. The fully-connected layers are responsible for mapping the extracted features into the desired output format, such as classification probabilities or predicted values.

The encoder block consists of a self-attention layer followed by a feedforward layer. The self-attention layer uses a dot-product attention mechanism to calculate the attention weights between each pair of input elements. These weights are then used to compute a weighted sum

of the input elements, which is used as the output of the self-attention layer. The feedforward layer consists of two linear transformations with a ReLU activation function in between. It takes the output of the self-attention layer as input and produces the final output of the encoder block.

The success in the use of the parallel deep learning technique of GoogleNet (Szegedy, et al., 2015), also known as Inception-v1, was the inspiration for the parallel architecture of this study, which allows the network to process multiple features concurrently. This could be achieved by using a series of inception modules, which will be concatenated and fed into the fully-connected (dense) layer. This parallel architecture enables GoogleNet to achieve good performance while being relatively efficient in terms of the number of parameters and computation time. It has been widely used in many image classification and object detection tasks.

3.1.1 Supporting Tools

In order to carry out this study, certain tools and equipment will be needed, including both hardware and software. The specific devices that will be used in this research are listed below:

3.1.2 Hardware

The primary tool employed for this study, which entails running complex simulations and analyzing data, is the Lenovo Legion 5 2021 Laptop. This hardware includes the following specifications:

a. AMD Ryzen 7 5800H (8 cores / 3.20GHz)
b. NVIDIA RTX 3070 Laptop GPU
c. 16GB of Random Access Memory (3200MHz)
d. 1TB Solid State Drive (SSD)

3.1.3 Software

To ensure that the models employed in this study performs correctly, certain software tools will be utilized to support this research. The software that will be used in this study includes:

a. Operating System: Windows 11
b. Programming Language: Python 3.10.9
c. Editor: Jupyter Notebook
d. Framework: PyTorch 1.13.1

3.2 Implementation and Trial Plans

This section will explain the dataset used in the study, as well as the stages of model and user interface implementation for the Pararel Transformer Encoder with CNN Architecture, including pseudocode and explanations. The evaluation metrics used to assess the performance of the Pararel Transformer Encoder with CNN Architecture will also be described.

3.2.1 Dataset

There are three English audio datasets used in the experiment, namely the CREMA-D (Cao, et al., 2014), RAVDESS (Livingstone & Russo, 2018), and SAVEE (Jackson & Haq, 2015) datasets. These datasets have been widely adopted in the field of audio emotion recognition and provide valuable resources for training and evaluating models. Each dataset encompasses a unique collection of emotional speech recordings, allowing for comprehensive analysis and

understanding of various aspects of emotional expression in audio data. The utilization of multiple datasets enhances the robustness of the study, as it enables the evaluation of model performance across diverse contexts and ensures a more representative analysis of audio emotion recognition capabilities.

a. Crowd-Sourced Emotional Multimodal Actors Dataset (CREMA-D)

The CREMA-D dataset is a large collection of audio and visual data that was created to aid research in the field of audio-visual scene understanding. The dataset contains 7,442 clips of audio and video recordings of various everyday scenarios, such as people talking, laughing, and singing in various environments. This dataset works with 91 actors (48 male and 43 female) between the ages of 20 and 74 coming from a variety of races and ethnicities (African American, Asian, Caucasian, Hispanic, and Unspecified). The actors spoke from a selection of 12 sentences. The sentences were presented using one of six different emotions (anger, disgust, fear, happy, neutral, and sad) and four different emotion levels (low, medium, high, and unspecified).

The recordings were made in a variety of environments, including homes, offices, parks, and streets. The dataset also includes a wide range of different people, including individuals of different ages, genders, and ethnicities. This diversity makes the dataset particularly useful for training models that can generalize well to real-world scenarios. Additionally, each clip in the dataset is accompanied by detailed annotations that describe the audio and visual content of the clip, as well as information about the people and objects present in the scene. These annotations make it possible for researchers to use the dataset for a wide range of different tasks, such as speech recognition, object detection, and facial recognition.

b. Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS)

The RAVDESS is another publicly available dataset that contains emotional speech recordings. This dataset consists of 1,470 recordings from 24 professional actors, comprising 50% male and 50% female. The actors were asked to portray different emotions such as calm, happy, sad, angry, fearful, and surprised. One of the key strengths of the RAVDESS dataset is its balanced representation of male and female voices. This is important because previous studies have shown that there are gender differences in the perception and expression of emotions. By including an equal number of male and female actors, the RAVDESS dataset provides a more representative sample of emotional expressions across genders. In addition, the RAVDESS dataset also includes recordings of speech in both neutral and accented English. An accent can also affect the perception and expression of emotions. By including accented English recordings, the RAVDESS dataset provides a more diverse set of emotional expressions that better represent real-world situations where emotions are expressed across different accents and cultures. The RAVDESS dataset has been widely used in emotion recognition, speech synthesis, and speech analysis tasks, showcasing its effectiveness in training deep neural networks, generating emotional speech, and analyzing acoustic features of emotional speech.

c. Surrey Audio-Visual Expressed Emotion (SAVEE)

The SAVEE dataset is a well-known and widely used audio dataset that contains a total of 480 recordings of spoken British English sentences, each of which is about 4-5 seconds in length. The dataset features four male speakers, and each speaker expressed seven different emotions: anger, disgust, fear, happiness, sadness, surprise, and neutral. The sentences were chosen from the standard TIMIT corpus and phonetically balanced for each emotion. The dataset was created specifically to aid research in the field of speech emotion recognition, and it has been utilized in numerous studies in this area. The recordings were recorded in a visual

media lab with high-quality audio-visual equipment, processed and labelled, resulting in a high-quality dataset that is well-suited for training and testing emotion recognition models.

3.2.2 Model Implementation

This section will outline the steps involved in conducting the research and explain the progression of the study from beginning to end. The various stages of the research are depicted in Figure 3.1, which provides a visual representation of the Pararel Transformer Encoder with CNN model architecture flow.

a. Pre-Processing

The initial stage in this research is by preprocessing the audio data from the datasets. Preprocessing is a critical step in audio data analysis that involves transforming raw audio signals into a format that is suitable for machine learning algorithms to extract meaningful information. Audio data can be complex, containing a wide range of frequencies, background noise, and other variations that can make it challenging to identify and extract relevant features. Preprocessing involves several steps that help to clean and transform the raw audio data into a format that is more amenable to analysis. In the case of audio emotion recognition, preprocessing is particularly important. It involves transforming the raw audio signals into a format that can be used to train a machine-learning model to recognize different emotions based on the acoustic properties of the speech. This involves a series of steps, including downsampling the audio to a custom target sample rate, truncating the audio to a set number of duration, and removing any silence before the actors start talking. These steps help to ensure consistency in the audio data and remove any irrelevant noise or silence that may affect the accuracy of the emotion recognition model. Algorithm 3.1 presents a pseudocode outlining the audio pre-processing steps used in this experiment.

Algorithm 3.1: Pre-Processing.

Input:
1. target_sample_rate.
2. target_time_duration.
3. offset_value.
4. label_mapping.

Output:
1. preprocessed audio waveforms.

Algorithm:
1. Define function to load and process audio files in the given directory.
2. Load each audio file in the directory using librosa.load function and apply preprocessing. This includes downsampling to the custom target sample rate, truncating to target duration value, and removing silence before the speech starts using the defined offset.
3. Map the emotion labels to numerical values using the defined label mapping.
4. Split the preprocessed audio data with an 80:20 ratio of training and testing data respectively.
5. Return the preprocessed audio data in the form of a tuple containing the training and testing sets.

To preprocess the audio data, the audio files will be loaded using the librosa.load() function

with a custom target sample rate and a fixed duration. This custom sample rate was chosen to standardize the sampling frequency of the audio files and make it consistent with the default sample rate used by most deep learning frameworks. In addition, an offset will be applied to the audio files to remove any silence before the actors start speaking. This is to ensure that only the emotional speech segments were retained for analysis and to avoid any unnecessary background noise or silence that might interfere with the emotion recognition process. Once the audio files were loaded, they were split into training and testing sets evenly across emotions. The training set was used to train the emotion recognition model, while the testing set was used to evaluate the model performance on unseen data. This split was done randomly to ensure that the training and testing sets were representative of the entire dataset and avoid any bias in the model performance. The emotion labels will be mapped numerically into six different categories for training using a mapping of {'angry': 0, 'fear': 1, 'disgust': 2, 'happy': 3, 'neutral': 4, 'sad': 5}. This mapping was done to convert the categorical labels into a format that can be used by the machine learning model for training and evaluation. To provide a clear representation of the process, a flowchart diagram has been included in Figure 3.2, which illustrates the sequence of operations involved in the pre-processing step.

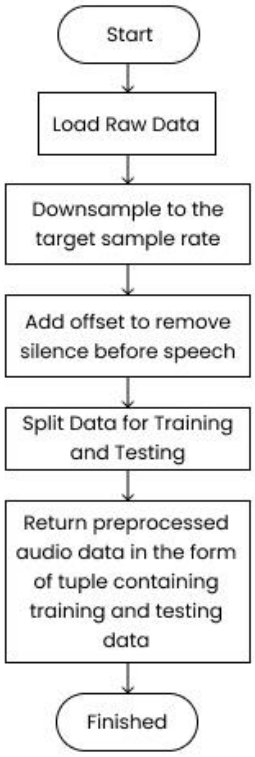


Figure 3.2: Pre-Processing Flowchart Diagram.

b. Feature Extraction

The next stage after preprocessing is to extract the audio features from the input data using the MFCC which is a popular feature extraction technique used in speech and audio processing because it is able to capture the spectral characteristics of an audio signal in a compactly and efficiently. MFCCs are derived from the power spectrum of an audio signal and are based on the Mel-scale, which is a non-linear scale that is based on the perceived frequency of a sound by the human ear. This makes MFCCs well-suited for tasks such as speech recognition and speaker identification, where the human ear is the primary means of perception. The flow of the MFCC feature extraction workflow has been depicted in Figure 2.3. Algorithm 3.2 shows a

pseudocode for extracting MFCC features from an audio signal using the librosa library. Once the MFCC features are extracted, they can be used as inputs for machine learning models for speech emotion recognition tasks, enabling the analysis and classification of emotions expressed through audio signals.

Algorithm 3.2: MFCC Feature Extraction.

```
Input:
   1. n_mfcc.
   2. mels.
   3. window_size.
   4. hop_length.
   5. audio_data.

Output:
   1. Mel frequency cepstrum coefficient (MFC Coefficient).

Algorithm:
   1. Compute the short-time Fourier transform (STFT) of the audio waveform
      using the specified window size and hop length.
   2. Compute the magnitude spectrogram of the STFT.
   3. Apply a mel filterbank to the magnitude spectrogram.
   4. Convert the mel spectrogram to decibel (dB) units.
   5. Compute the discrete cosine transform (DCT) of the log-mel
      spectrogram.
   6. Return the resulting MFCC coefficients as a feature matrix for further
      analysis and emotion recognition.
```

The application employs the librosa library to retrieve the audio signal from a file to extract the MFCCs from an audio waveform that involves several sequential stages. The first step is to compute the Short-Time Fourier Transform (STFT) of the audio waveform using a specified window size and hop length. This step involves dividing the audio signal into short frames of equal length, with a hop length of about half of the frame size. The Hanning windowing function is typically used for each audio frame, which helps to reduce spectral leakage and improve the frequency resolution of the STFT. The next step is to compute the magnitude spectrogram of the STFT. The magnitude spectrogram is then transformed using a Mel filter bank, which approximates the frequency response of the human auditory system. After applying the filter bank, the resulting Mel spectrogram is converted to decibel (dB) units using a logarithmic scale. This step is necessary to compress the magnitude values and better represent the relative loudness of each frequency bin. Finally, the MFCC coefficients are computed by performing a discrete cosine transform of the log-mel spectrogram. This step involves transforming the Mel spectrogram into a sequence of cepstral coefficients that represent the spectral envelope of the signal.

c. Model Architecture Design

After the audio features have been extracted, the next step is to create the model architecture and use the extracted features as input to the model which will allow the model to classify human emotions based on the extracted features. There are two blocks of the deep learning model for the Pararel Transformer Encoder with CNN Architecture, the CNN block and the Transformer block which will be working in parallel with each other. The idea is for the CNN to give spatial feature representation of the input data, and the Transformer block in sequence

encoding to try and model as accurately as possible the temporal relationships between pitch transitions in human emotions. The expansion of CNN filter channels and reduction of feature maps will provide the most expressive feature representation with the lowest computational cost, while the Transformer encoder will learn to predict frequency distributions of different emotions according to the global structure of the MFCC plot of each emotion. The implementation for CNN and Transformer block will be shown in Algorithm 3.3 and Algorithm 3.4, respectively.

Algorithm 3.3: CNN Block.

Input:
1. input audio tensor.

Output:
1. convolution Embedding.

Algorithm:
1. Define the CNN model architecture by initializing the sequential model.
2. Add the convolutional layers to the model and specify the number of filters, filter sizes, and activation function for each layer.
3. Add batch normalization before max pooling to normalize the inputs of each layer.
4. Add a max pooling layer to the model to reduce the dimensionality of the output.
5. Add a flattening layer to the model to convert the 2D matrix output into a 1D vector.
6. Return the convolution embedding from the output of the Flatten layer.

Algorithm 3.3 presents an implementation of the CNN Block for the deep learning model. The first layer is the input layer that takes in audio features with a certain size and number of channels. The second layer is the convolution layer that applies a set of filters to this input data, generating a set of feature maps. In this study, the filters are 3×3 matrices and they are applied to the input data through a process called convolution. The third layer is the batch normalization layer which normalizes the output from the convolution layer, making the model more stable and efficient. Fourth is the activation layer which applies an activation function (i.e., ReLU) to the feature maps generated by the previous layer. This layer introduces non-linearity to the model, allowing it to learn more complex relationships in the data. Finally, the pooling layer down-samples the feature maps by applying a pooling operation (i.e., MaxPooling). This helps reduce the size of the feature maps and, as a result, lowers the computational complexity of the model. This sequence of applying convolutional layers, batch normalization layers, activation layers, and pooling layers is repeated three times in the model architecture.

The Transformer encoder implementation was shown in Algorithm 3.4 which is designed to process sequential data of the audio source. It consists of a series of self-attention layers and feedforward layers, which are used to predict frequency distributions of different emotions according to the global structure of the MFCCs of each emotion. In the implementation, the output sequence is initialized first as an empty list. Then the input sequence is embedded by applying an embedding matrix to each element of the input, resulting in a sequence of embedded vectors. The embedded sequence will then be passed through a series of self-attention layers to compute a sequence of context-aware representations. Each self-attention layer applies the

attention mechanism to the input sequence to compute a weighted sum of the input vectors, where the weights are computed based on the relationships between the input elements. These context-aware representations are then passed through a series of feedforward layers to compute a sequence of transformed representations. Each feedforward layer applies a linear transformation to the input, followed by a nonlinear activation function. Finally, the transformed representations are used to compute the output sequence by applying a linear transformation and an activation function to each element of the transformed representations.

Algorithm 3.4: Transformer Encoder Block.

Input:

1. input audio tensor.

Output:

1. transformer encoding embedding.

Algorithm:

1. Define transformer encoder block by initializing the transformer encoder layer.
2. Add a multi-head attention layer with a specified number of heads, hidden dimensions, and dropout rate.
3. Add a layer normalization layer after the multi-head attention layer.
4. Add a feedforward neural network layer with a specified number of hidden units and activation function.
5. Add another layer normalization layer after the feedforward neural network layer.
6. Return the transformer encoding embedding from the output of the final layer normalization layer.

Algorithm 3.5: Dense Layer Concatination.

Input:

1. cnn embedding input.
2. transformer encoding embedding input.
3. number of classes.

Output:

1. class prediction probabilities.

Algorithm:

1. Concatenate the CNN and Transformer Encoding embeddings along the feature dimension to create a joint embeddings.
2. Pass the joined embeddings through a dense layer to combine the features.
3. Apply a softmax activation function to the output of the dense layer to obtain class probabilities.
4. Return the class prediction probabilities.

The final stage of the Pararel Transformer Encoder with CNN Architecture model is to concatenate both outputs from the CNN model and the Transformer encoder model and pass the resulting tensor through a dense layer with a softmax activation function for prediction. The softmax function is a common choice for the activation function in the final layer of a classification model. It takes a vector of arbitrary real-valued scores and converts it into a

probability distribution, where the probability of each class is given by the corresponding element in the output vector. Algorithm 3.5 outlines the process of combining the outputs of the CNN and Transformer models, passing them through a dense layer, and applying the softmax function to the output of the dense layer to make predictions.

The dense layer implementation of Algorithm 3.5 starts by combining the output tensors of CNN and Transformer which has the shape (batch_size, feature_size) for each of the models, respectively. The combined output is then passed through a linear layer with the number of class units, which is the six different emotional states in the dataset. The dense layer has weights and biases that will be learned during training to transform the combined output into the final prediction. Finally, the SoftMax activation function is applied to the output of the linear layer that will convert the prediction scores into a probability distribution over the classes. The class with the highest probability is taken as the model prediction. The layer architecture of the Pararel Transformer Encoder with CNN Architecture model is presented in Table 3.1.

Table 3.1: Transformer Encoder Block Combined with CNN-based Architecture Layer.

Block	**Layer**	**Output Shape**	**Parameter**
1	Conv2D Layer 1	[16, 40, 80]	160
	ELU	[16, 40, 80]	-
	Batch Normalization	[16, 40, 80]	32
	MaxPool	[16, 20, 40]	-
	Dropout	[16, 20, 40]	-
	Conv2D Layer 2	[32, 20, 40]	4,640
	ELU	[32, 20, 40]	-
	Batch Normalization	[32, 20, 40]	64
	MaxPool	[32, 5, 10]	-
	Dropout	[32, 5, 10]	-
	Conv2D Layer 3	[64, 5, 10]	18,496
	ELU	[64, 5, 10]	-
	Batch Normalization	[64, 5, 10]	128
	MaxPool	[64, 1, 2]	-
	Dropout	[64, 1, 2]	-
2	Conv2D Layer 1	[16, 40, 80]	160
	ELU	[16, 40, 80]	-
	Batch Normalization	[16, 40, 80]	32
	MaxPool	[16, 20, 40]	-
	Dropout	[16, 20, 40]	-
	Conv2D Layer 2	[32, 20, 40]	4,640
	ELU	[32, 20, 40]	-

Block	Layer	Output Shape	Parameter
	Batch Normalization	[32, 20, 40]	64
	MaxPool	[32, 5, 10]	-
	Dropout	[32, 5, 10]	-
	Conv2D Layer 3	[64, 5, 10]	18,496
	ELU	[64, 5, 10]	-
	Batch Normalization	[64, 5, 10]	128
	MaxPool	[64, 1, 2]	-
	Dropout	[64, 1, 2]	-
3	Transformer Encoder Layer 1 – 6	[2, 40]	48,232
	FC Layer 1	[6]	1,782
	SoftMax	[6]	-

Note: "-" indicates that there are no specific parameters associated with the layer or operation.

3.2.3 Comparison Model Architecture Design

This section will provide an overview of the model architectures that were employed for the purpose of comparing the Pararel Transformer Encoder with CNN Architecture. The models that will be compared include a traditional machine learning model in the form of SVM, as well as deep learning models based on Convolutional architectures such as the LeNet-based CNN and the CRNN model.

a. Support Vector Machine (SVM)

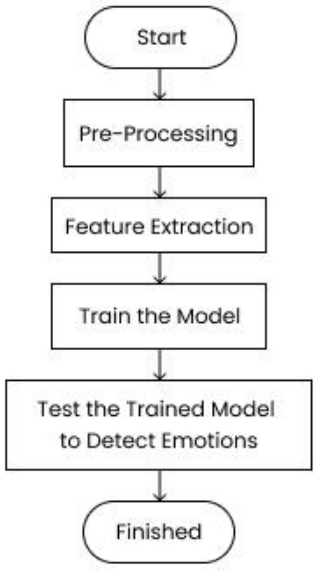


Figure 3.3: SVM Flowchart Diagram.

The design presented here details the process flow for implementing the SVM method. SVM was utilized as a benchmark due to its growing usage in audio emotion recognition systems (Sonawane, Inamdar, & Bhangale, 2017). Additionally, SVM is a classical machine learning model that can handle high-dimensional data and swiftly developed for training, is user-friendly and has been extensively applied in diverse domains, including image and audio classification, and data analysis. The process flow of how SVM is employed in this experiment is illustrated in Figure 3.3.

For this study, the SVM model was employed using the scikit-learn library in Python. To optimize model hyperparameters, the GridSearchCV function was utilized, which performs an exhaustive search over specified parameter values for an estimator. The hyperparameters that were optimized included the regularization parameter C, the kernel coefficient gamma, and the probability setting. A grid of hyperparameters was created by defining a list of possible values for each parameter. The values used for the 'C' parameter were [0.001, 0.01, 0.1, 1, 10], while the values used for the 'gamma' parameter were [0.001, 0.01, 0.1, 1,'scale', 'auto']. The 'kernel' parameter had four possible values: linear, rbf, poly, and sigmoid. Finally, the 'probability' parameter was set to True and False.

GridSearchCV exhaustively searches for the best combination of hyperparameters by training and testing the model with every possible combination of hyperparameters. After running the GridSearchCV method with the given parameters, the best hyperparameters for the SVM model were presented in Table 3.2 which will be used to train and evaluate the SVM model to determine the performance of the trained model.

Table 3.2: SVM Model Parameters.

Parameter	Value
C	10
Gamma	0.01
Kernel	Rbf
Probability	True

b. LeNet-based Convolutional Neural Network (LeNet CNN)

LeNet-based CNN was used as a benchmark model for comparison because the Pararel Transformer Encoder with CNN Architecture model CNN block was derived from the original CNN architecture of LeNet, which comprises several convolutional and pooling layers. LeNet CNN was used for the model because of its simplicity yet highly effective for many classification tasks and has been successfully utilized in diverse domains of classification. The experimental process flow, depicted in Figure 3.4, demonstrated how the deep learning models, including both LeNet-based CNN and the CRNN models, were employed.

The model architecture of LeNet CNN used in this experiment was identical to the original paper (LeCun, Bottou, Bengio, & Haffner, 1998), except for the activation function which was changed from tanh to Elu. The first layer of the CNN used six filters of size 5×5 to process the input audio data, followed by a 2×2 max pooling operation that reduced the dimensionality of the feature maps. The output of the first layer was then processed by the second layer, which used 16 filters of size 5×5. This output was again subsampled using a 2×2 max pooling operation. The final three layers of the CNN architecture were fully connected layers, responsible for capturing complex relationships and producing emotion class predictions. The first fully connected layer consisted of 120 neurons, followed by a second fully connected layer with 84 neurons. The output from the second fully connected layer was subsequently fed into the softmax layer, which outputs probability distribution across emotion classes in the dataset.

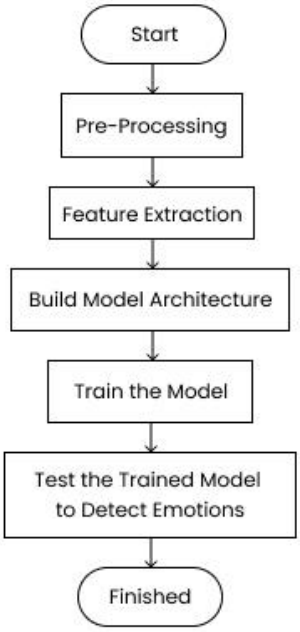


Figure 3.4: Deep Learning Flowchart Diagram.

To efficiently optimize the model during the training process, a minibatch size of 32 was utilized, meaning that 32 samples were processed in each iteration. The training process was set to run for a fixed number of epochs, specifically 250, although it would terminate earlier if convergence was achieved. Stochastic Gradient Descent (SGD) was employed as the optimizer, with a learning rate of 0.001, weight decay of 0.001, and a momentum value of 0.8. These hyperparameters were chosen based on previous experimentation and a thorough review of related literature. During training, the cross-entropy loss function was employed to quantify the dissimilarity between the predicted and actual emotion class labels. The objective was to minimize this loss function, thereby improving the model accuracy in accurately classifying emotions within the audio dataset. A detailed breakdown of the model parameters can be found in Table 3.3, while Table 3.4 provides a comprehensive overview of the architecture layer.

Table 3.3: Deep Learning Model Parameters.

Parameter	Value
Batch Size	32
Epoch Size	300
Optimizer	SGD (learning rate=0.001, decay=0.001, momentum=0.8)
Loss Function	Cross-Entropy Loss

Table 3.4: LeNet Architecture Layer.

Layer	Output Shape	Parameter
Conv2D Layer 1	[6, 36, 76]	156
Elu	[6, 36, 76]	-

Layer	Output Shape	Parameter
MaxPool	[6, 18, 38]	-
Conv2D Layer 2	[16, 14, 34]	2,416
Elu	[16, 14, 34]	-
MaxPool	[16, 7, 17]	-
FC Layer 1	[120]	228,600
FC Layer 2	[84]	10,164
FC Layer 3	[6]	510
SoftMax	[6]	-
Total Parameters		241,846

Note: "-" indicates that there are no specific parameters associated with the layer or operation.

c. Convolutional Recurrent Neural Network (CRNN)

The CRNN architecture was used as a comparison method against the Pararel Transformer Encoder with CNN Architecture model due to its convolutional-based architecture, which is also based on the LeNet CNN with a sequence of convolution and pooling layers. However, the CRNN model is unique in that it incorporates a recurrent network for modelling temporal dependencies within the feature maps. This could be beneficial in the analysis of sequential data such as audio, where features from a given time step depending on the features of the previous time step. The process flow of how CRNN is employed in this experiment was the same as that for the LeNet CNN model, as shown in Figure 3.4.

The architecture of the CRNN model in this experiment was based on the original paper (Shi, Bai, & Yao, 2017). The model comprised of seven convolutional layers with a 2x2 max pooling layer applied after the first, second, fourth, and sixth convolutional layers. The other convolutional layers had no pooling operations applied. The output of the final convolutional layer was fed into a BLSTM layer, which is capable of modelling the temporal dependencies in the feature maps.

The final layer of the CRNN model was a fully connected layer, followed by a softmax layer for classification. The use of a BLSTM layer in the CRNN model allows the model to process the input sequence in both forward and backward directions, which enables the model to capture information from both past and future frames.

The CRNN model used in the experiment had the same model parameter values as the LeNet model, which can be found in Table 3.3. The layer architecture of the CRNN model is presented in Table 3.5.

Table 3.5: CRNN Architecture Layer.

Layer	Output Shape	Parameter
Conv2D Layer 1	[64, 48, 80]	640
ReLU	[64, 48, 80]	-
MaxPool	[64, 24, 40]	-
Conv2D Layer 2	[128, 24, 40]	73,856

Layer	Output Shape	Parameter
ReLU	[128, 24, 40]	-
MaxPool	[128, 12, 20]	-
Conv2D Layer 3	[256, 12, 20]	295,168
ReLU	[256, 12, 20]	-
Conv2D Layer 4	[256, 12, 20]	590,080
ReLU	[256, 12, 20]	-
MaxPool	[256, 6, 20]	-
Conv2D Layer 5	[512, 6, 20]	1,180,160
ReLU	[512, 6, 20]	-
Conv2D Layer 6	[512, 6, 20]	2,359,808
ReLU	[512, 6, 20]	-
MaxPool	[512, 3, 20]	-
Conv2D Layer 7	[512, 2, 19]	1,049,088
ReLU	[512, 2, 19]	-
Sequence Mapping Layer	[2, 64]	65,600
LSTM Layer 1	[2, 512]	659,456
LSTM Layer 2	[2, 512]	1,576,960
FC Layer 1	[2, 6]	3,078
SoftMax	[6]	-
Total Parameters		7,855,942

Note: "-" indicates that there are no specific parameters associated with the layer or operation.

3.2.4 Model Evaluation

Model evaluation is an important step in the development of a deep learning model, as it could assess the performance of the model on unseen data and determine its suitability for a given task. This section will outline some general considerations for evaluating deep learning models for a speech emotion recognition task.

There are several ways to evaluate the performance of deep learning models. This study aims to compare the performance of four different machine learning models on a speech emotion recognition task. These models include the standard machine learning SVM model, a LeNet-based CNN architecture model, the CRNN model, and the Pararel Transformer Encoder with CNN Architecture for this study. A combination of different evaluation metrics will be used to evaluate the performance of these models.

First, the training and validation accuracy of the model will be tracked to ensure that the model is not overfitting the training data. Tracking the training process of a model could help identify and address the overfitting and underfitting in the data. Overfitting occurs when the model performs well on the training data but poorly on the validation or test data, indicating

that it has learned patterns that are specific to the training data and are not generalizable. While underfitting occurs when the model performs poorly on both the training and validation data, indicating that it is not able to learn the underlying patterns in the data. In addition to addressing overfitting and underfitting, tracking the model accuracy on the validation set can also help choose the best hyperparameter values leading to the best model performance by observing and changing the effect on the validation accuracy. In summary, this method will show how well the models can learn the classification task and identify any issues with the optimization process. After training the model, the test set will be used to evaluate their performance using several metrics.

One of the best metrics to evaluate is the test set accuracy for each model to get an idea of how the model performs on unseen data. This metric will give a summary of the model performance on the test set. The test set is a set of data that the model has not seen during training, and therefore provides a more realistic evaluation on the performance of the model. Evaluating the model on the test set accuracy can give a more accurate assessment of the model generalization ability, which is its ability to perform well on unseen data. This is essential for understanding the suitability of the model for deployment and its potential real-world performance.

In addition to measuring accuracy, the inclusion of a confusion matrix for each model offers valuable insights into the types of errors made and potential data imbalances. A confusion matrix is a table that presents the number of true positive, true negative, false positive, and false negative predictions made by the model. True positive predictions indicate when the model correctly identifies instances of the positive class, while true negative predictions occur when the model accurately identifies instances of the negative class. Conversely, false positive predictions represent cases where the model incorrectly identifies instances as belonging to the positive class, and false negative predictions occur when the model incorrectly identifies instances as belonging to the negative class. By leveraging this evaluation metric, it becomes possible to assess the performance of the models across different emotion classes, gaining a detailed understanding of their strengths and weaknesses.

Other metrics such as precision, recall, and the F1 score can be used on the test set for each model. These evaluation metrics are commonly used to assess the performance of deep learning models, particularly for classification tasks. Precision measures the proportion of true positive predictions made by the model among all positive predictions, while recall measures the proportion of true positive predictions made by the model among all actual positive examples. The F1 score is a combination of precision and recall and is calculated as the harmonic mean of the two. The F1 score is useful because it takes into account both the precision and recall of the model and provides a single metric that reflects the overall performance of the model. These metrics of evaluation give a detailed understanding of the model performance and compare them to each of the six different emotion classes in the CREMA-D dataset.

Lastly, the weighted average of the F1 scores could be considered as one of the metrics of evaluation for this study. The weighted average of the F1 score is a variation of the F1 score that is used to evaluate the performance of a classification model when dealing with imbalanced classes. In an imbalanced dataset, the classes are not equally represented, which can make it difficult to accurately evaluate model performance. The weighted average helps address the issue by adding weights in different classes in the calculation of the F1 score. These weights reflect the relative importance of the different classes and can be used to give more emphasis to the performance of the model on a particular class. This metric is useful for evaluating the

model performance with imbalance classes and can be used to compare the overall performance of the three models for the speech emotion recognition task.

3.2.5 User Interface

This study provides a way to share the machine learning models employed for this study using an interactive web-based application that can be used on any device with browser support. This section will explain about the design for the web-based user interface application which will be discussed in detail, highlighting the features and functionalities that will be included in the application. Figure 3.5 and Figure 3.6 provides an overview of the design for the web-based user interface application.

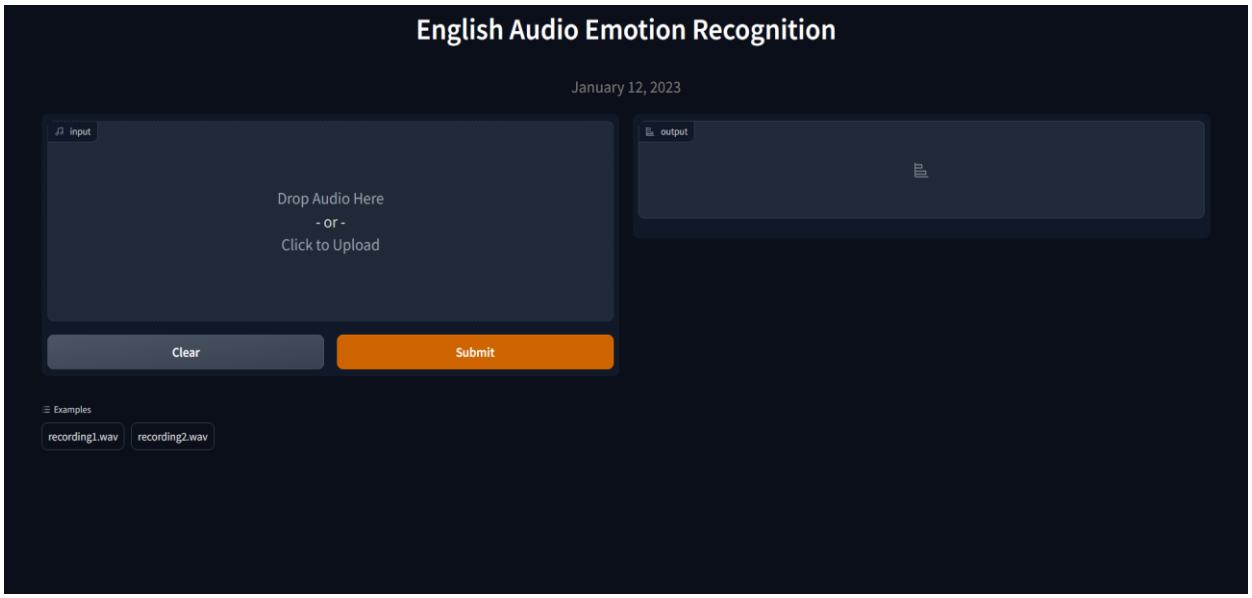


Figure 3.5: Web Design Pre-Audio Input.

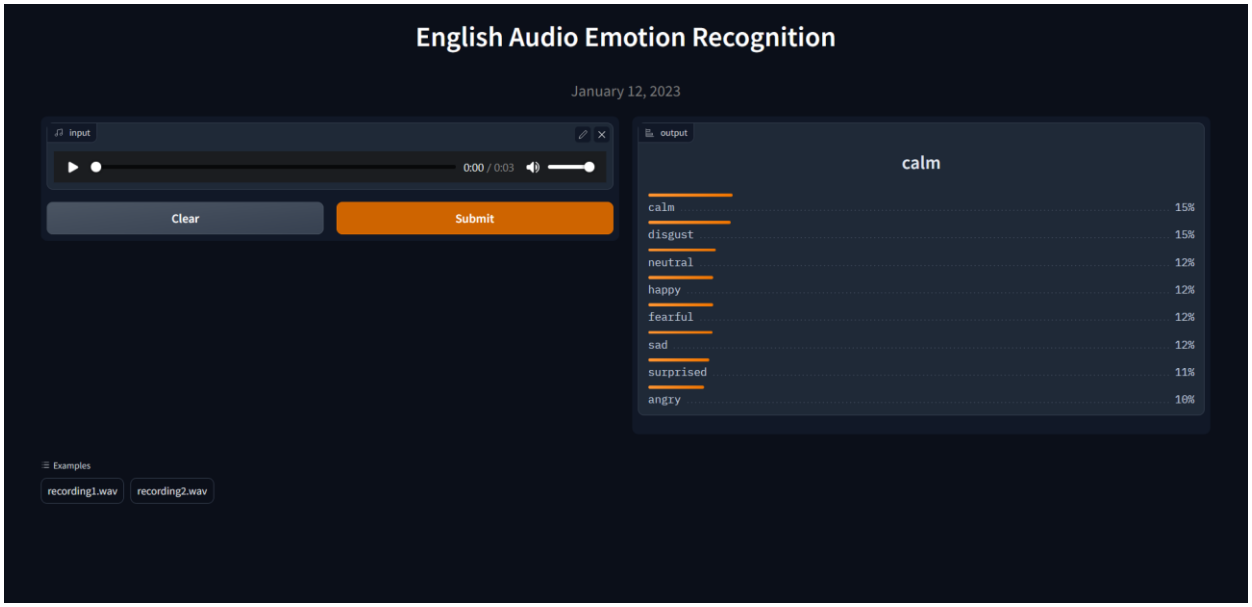
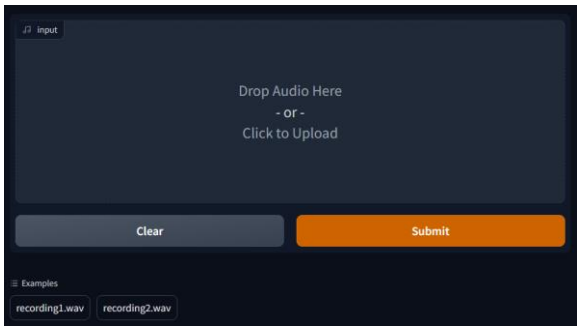
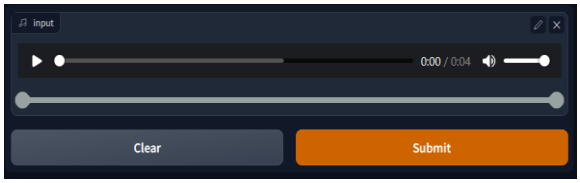


Figure 3.6: Web Design Post-Audio Input.

The design for this web-based application is intended to make the application user friendly

and easy to navigate, allowing users to quickly and easily access the features and test the machine learning models used for this study. Additionally, users can also view the model performance through charts displayed in the output section of the website depicted in Figure 3.6, which shows the probability distribution of emotions classification. Overall, the goal of the design is to create an efficient and effective user experience for the web-based application, specifically for showcasing the machine learning models performance. Table 3.6 include a detailed explanation that will provide a comprehensive overview of how each feature works in order to give a clear understanding of the functionalities of this web-based application.

Table 3.6: Web-based Application Feature Functionality.

Feature	Description
	This is the input section; Users can choose to upload their audio recording in wav format or use the included recording examples.
	Once an audio recording is loaded, a couple of features can be used, such as: • Audio playback; • Volume control; • Snipping tool;
	The output section will give the emotion classification based on the imported audio file along with a classification label of the probability distribution scores for each emotion.

Chapter IV
Result and Discussion

This chapter will discuss the implementations and the results achieved during this research. The performance and effectiveness of each machine learning model will be assessed and presented in achieving its intended objectives. The analysis will provide valuable insights into the challenges and successes encountered during the implementation process, and offer recommendations for further improvement and development.

4.1 Implementation

An outline of the methods employed in this study to compare the audio emotion recognition system was presented in Chapter III. This section will explore the data preparation process of the system, highlighting data exploration, pre-processing, and feature extraction steps.

4.1.1 Data Exploration

The dataset in this study consists of English human voices that have already been labelled. Three distinct datasets were utilized for means of comparison, namely CREMA-D, RAVDESS, and SAVEE datasets, which contain single emotional labels for each sound sample. The CREMA-D dataset comprises approximately 7,442 clips of 91 actors, with an equal distribution of males and females. Each actor was instructed to act out 6 different emotions, including anger, disgust, fear, happiness, neutral, and sad. The clips vary in duration, ranging from 1 to 5 seconds and are recorded at a sampling rate of 16kHz with a resolution of 16 bits. The emotional state of each clip was labelled by multiple crowd-workers on Amazon Mechanical Turk, ensuring a diverse set of labels that represent the emotions expressed in each clip. The RAVDESS dataset consists of approximately 1,440 audio files of actors performing scripted speech and song segments. The audio files are recorded at a sampling rate of 48kHz with a resolution of 16 bits and are labelled with one of eight emotions, including calm, happy, sad, and angry, amongst others. The actors are of diverse ages, genders, and ethnicities, ensuring that the dataset is representative of a wide range of voices and accents. Finally, the SAVEE dataset is a collection of emotional speech recordings from four male speakers. The dataset contains a total of 480 British English utterances, where each speaker expresses seven different emotions: anger, disgust, fear, happiness, sadness, surprise, and neutral. The recordings were made under controlled laboratory conditions, with each speaker uttering the same set of 72 sentences. The audio was recorded with a native sampling rate of 44.1 kHz and saved in 16-bit wav format.

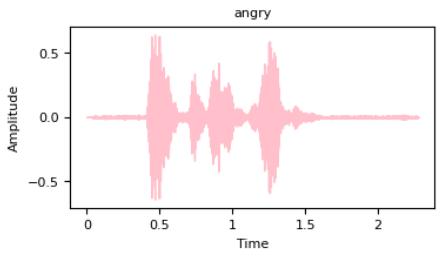
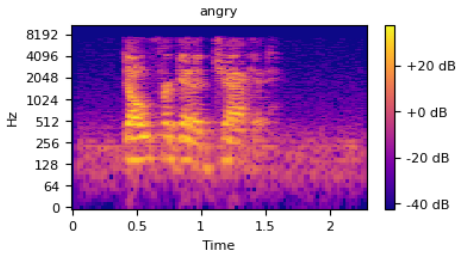
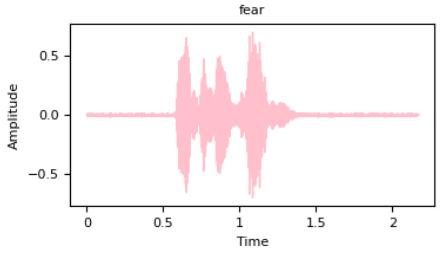
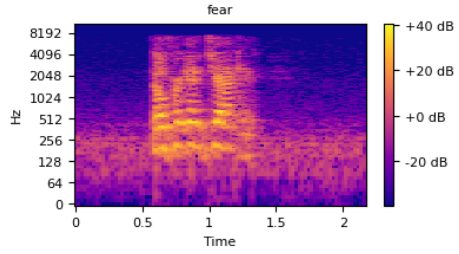
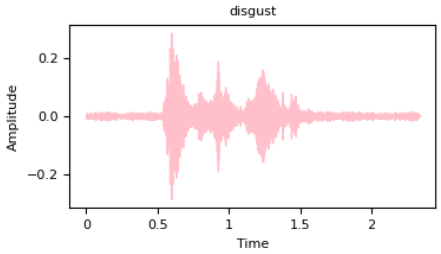
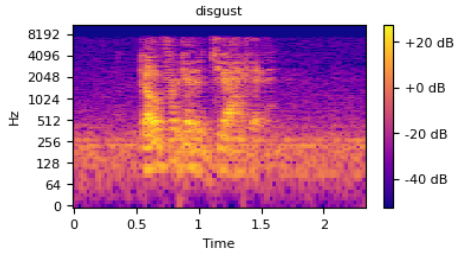
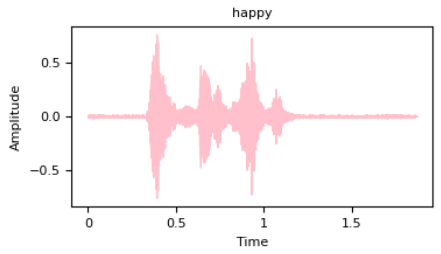
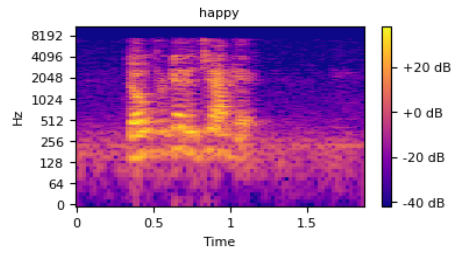
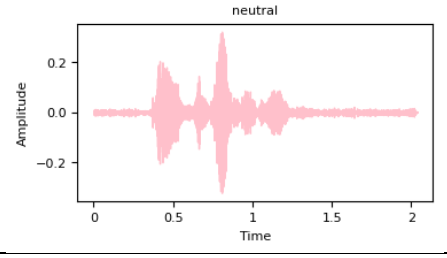
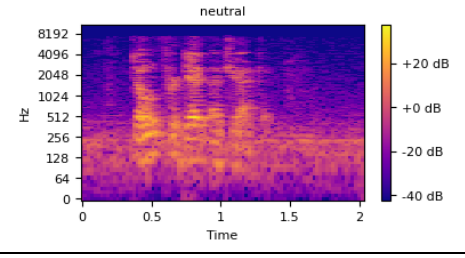
This study utilized the emotions present in the CREMA-D dataset, which consists of 6 different emotion classes including anger, disgust, fear, happiness, neutral, and sad. Using a standardized set of emotions across the dataset allows for a fair comparison of performance between different models and techniques. The use of the same emotion classes across all datasets helps to eliminate any potential biases that may have resulted from variations in labelling, recording quality, or other factors that may impact the effectiveness of emotion recognition models. Overall, by using a standardized set of emotions, the study aims to provide a more accurate and reliable analysis of the performance of different models and techniques for audio emotion recognition.

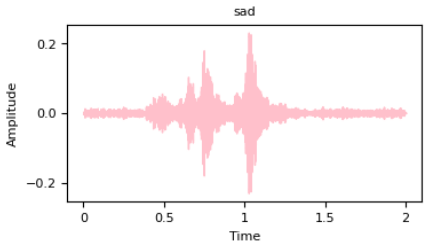
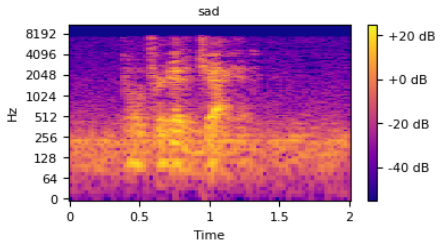
As part of the exploratory data analysis, amplitude and spectrogram plots were generated for these datasets. An amplitude plot shows the variation of sound pressure over time and is useful for identifying the overall loudness of a recording. The spectrogram plot, on the other hand, shows how the energy of the signal is distributed across different frequencies over time and can help identify specific characteristics of the audio signal. Table 4.1 showed examples of amplitude and spectrogram plots representing each of the six emotion categories in the

CREMA-D dataset. Table 4.2 displayed the amplitude and spectrogram plots for the RAVDESS dataset. Finally, Table 4.3 includes the corresponding amplitude and spectrogram plots for the SAVEE dataset.

In addition to generating amplitude and spectrogram plots for the datasets, further analysis was conducted to explore the distribution of emotions within each dataset. The distribution of emotional labels provides insights into the balance and representation of different emotions within the datasets. By examining the frequency of each emotion in the CREMA-D, RAVDESS, and SAVEE datasets, it is possible to identify any imbalances that may exist.

Table 4.1 CREMA-D Amplitude and Spectogram.

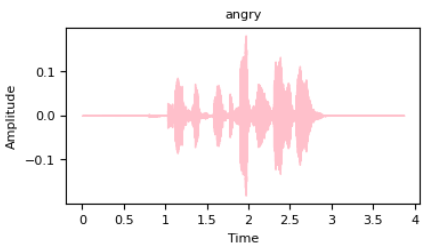
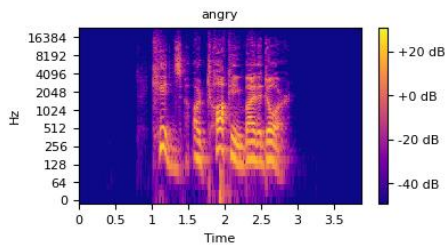
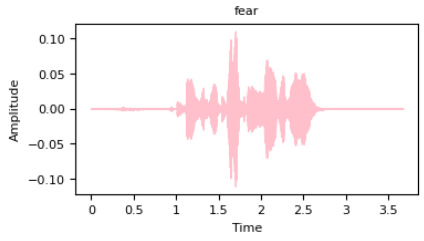
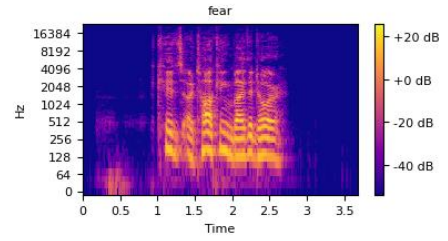
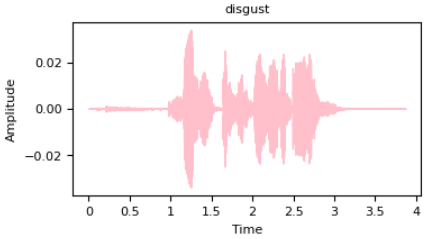
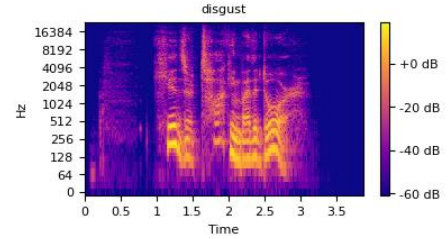
Emotion	**CREMA-D**	
	Amplitude	**Spectogram**
Angry		
Fear		
Disgust		
Happy		
Neutral		

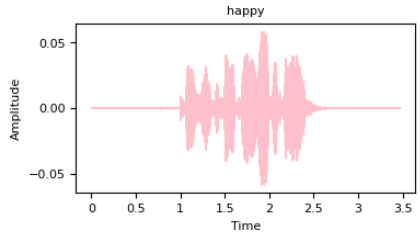
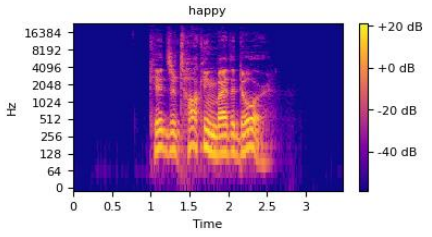
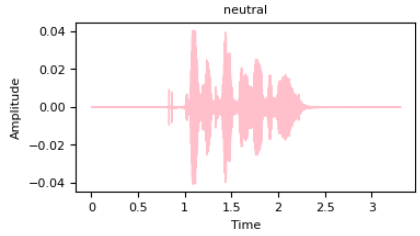
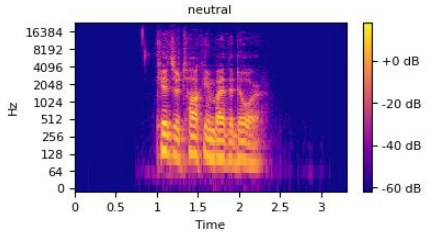
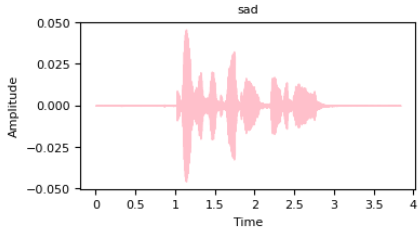
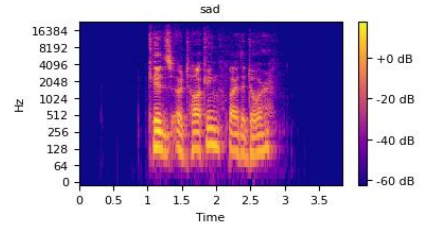
Emotion	CREMA-D	
	Amplitude	Spectogram
Sad		

Sound characteristics of different emotions in the CREMA-D dataset are shown in Table 4.1. The amplitude plot of the dataset reveals that each emotion class has distinct sound characteristics. For example, anger, fear, and joy emotions have higher amplitudes above 0.5, while disgust, neutral, and sad emotions are at lower amplitudes of approximately 0.2. Additional figures illustrating these sound characteristics can be found in Appendix 1. These variations in amplitudes suggests that different emotions have unique patterns of sound frequencies.

Similarly, the spectrogram plot also reveals the distribution of sound frequencies for each emotion class. The distribution of each emotion is different, and several similar emotions show patterns in the CREMA-D dataset. For instance, angry and happy emotions share similar patterns, with high frequencies and a narrow band of energy concentrated around the middle of the frequency range. In contrast, disgust and sad emotions have lower frequencies and more spread-out energy distribution across the frequency range. The spectrogram plot helps to identify these subtle differences in patterns between different emotion classes.

Table 4.2 RAVDESS Amplitude and Spectogram

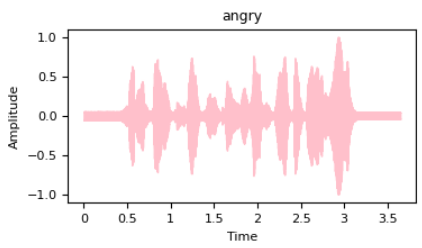
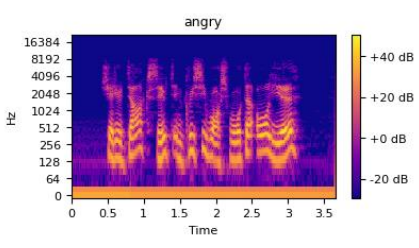
Emotion	RAVDESS	
	Amplitude	Spectogram
Angry		
Fear		
Disgust		

Emotion	RAVDESS	
	Amplitude	**Spectogram**
Happy		
Neutral		
Sad		

Compared to the CREMA-D dataset, the RAVDESS dataset has a much greater variety of sound characteristics shown in Table 4.2. For a more comprehensive understanding, additional figures showcasing these sound characteristics can be found in Appendix 2. The amplitude plots highlight the distinct amplitudes associated with each emotion in the RAVDESS dataset. Each emotion exhibits a unique range of amplitudes, further emphasizing the diversity of sound characteristics within the dataset. The spectrograms of the RAVDESS dataset also clearly displayed distinct characteristics for each emotion. The significant differences in amplitude and spectrogram patterns for each emotion can be used as reliable indicators of emotional states.

To illustrate, a neutral emotion can be identified in an audio signal by observing a spectrogram that displays a uniform pattern in contrast to the patterns observed for other emotions. This discovery carries significance because it implies that the RAVDESS dataset could be a valuable resource for researchers investigating emotional expression in speech. The RAVDESS dataset includes a greater range of audio characteristics that could facilitate a more nuanced analysis of emotional expression, as well as increase the accuracy of emotion recognition models.

Table 4.3: SAVEE Amplitude and Spectogram.

Emotion	SAVEE	
	Amplitude	**Spectogram**
Angry		

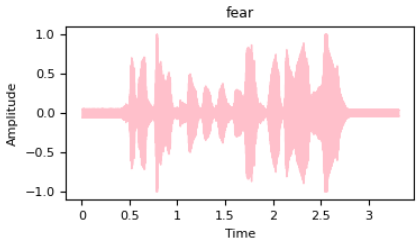
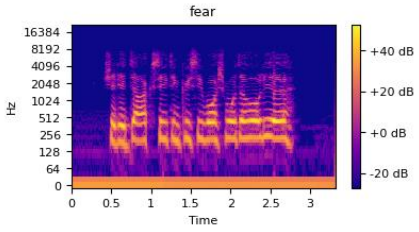
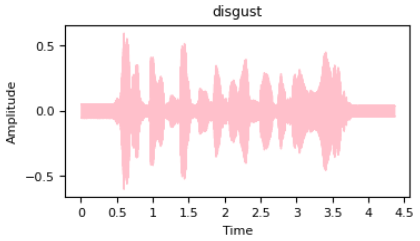
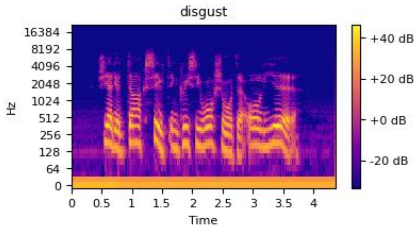
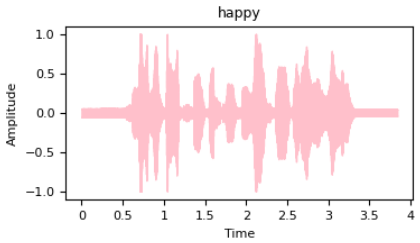
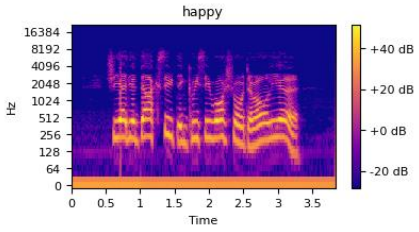
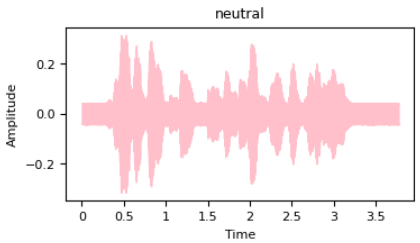
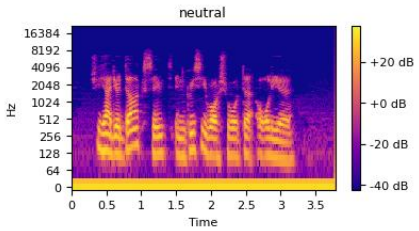
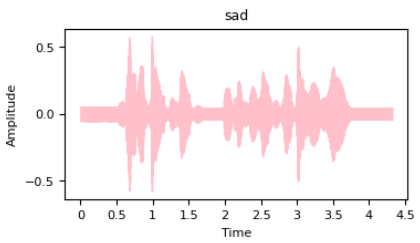
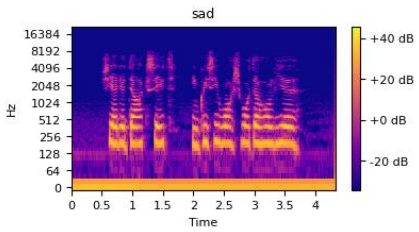
Emotion	SAVEE	
	Amplitude	Spectogram
Fear		
Disgust		
Happy		
Neutral		
Sad		

Table 4.3 provide the amplitude and spectrogram plots of the SAVEE dataset for each emotion class which can reveal important insights into the acoustic characteristics of different emotions. For a more detailed examination, additional figures illustrating these characteristics can be found in Appendix 3. The amplitude plots could provide information about the overall shape and amplitude of the audio signal over time. The fact that anger, fear, and happy emotions have a maximum amplitude of 1.0, while disgust and sad have a maximum amplitude of 0.5, and neutral has a maximum amplitude of 0.2, could indicate that these emotions have different intensities and energy levels. This suggests that emotions such as anger, fear, and happiness are generally more intense and louder than emotions such as sadness, disgust, and neutrality.

On the other hand, the spectrogram plots provide insights into the frequency content and distribution of energy over time. The variations in frequency content and energy distribution can reveal distinct characteristics of each emotion class. For example, high-frequency energy

levels are greater in anger and fear emotions, while low-frequency energy levels are higher in sadness and disgust emotions. These differences could be useful in identifying different emotions based on their acoustic features.

Overall, these plots demonstrate that the use of amplitude and spectrogram plots can provide valuable insights into the unique acoustic properties of various emotions in the datasets examined in this study.

4.1.2 Pre-Processing

Audio data pre-processing is a critical step in the development of any audio analysis system. It involves a series of techniques aimed at cleaning, transforming, and preparing raw audio data to make it suitable for use in machine learning algorithms. Some of the key pre-processing techniques employed in this study include loading and splitting the dataset, feature extraction, and feature scaling. By applying these techniques, machine learning models can improve the accuracy and reliability of their audio analysis systems and ensure that they can extract meaningful insights from their data.

a. Load Data

The data loading process was critical in preparing the audio data for use in the machine learning models. To ensure that the audio data was suitable for training the models, several data loading parameters were tested using the librosa library. The main parameters tested were the target duration, target sample rate, and offset. The target durations were set based on the minimum, maximum, and average durations of the dataset. The CREMA-D dataset has a target duration of 1.28 seconds, 5 seconds, and 2.54 seconds. The RAVDESS dataset has a target duration of 3.07 seconds, 5.27 seconds, and 3.73 seconds. Lastly, the SAVEE dataset has a target duration of 1.63 seconds, 7.14 seconds, and 3.84 seconds.

As for the target sample rate parameter, two different values were tested, the native sample rate (16kHz for CREMA-D, 48kHz for RAVDESS, and 44.1kHz for SAVEE) and the default sample rate from librosa, which is 22.5 kHz. This facilitated the assessment of the comparative effectiveness of various machine learning models under different data loading sample rates.

Finally, to determine the impact of removing silence or noise on the performance of the model, the offset parameter was tested. The average offset value was 0.3 seconds for the CREMA-D dataset, 0.8 seconds for the RAVDESS dataset, and 0.5 seconds for the SAVEE dataset. The inclusion and exclusion of the offset parameter were tested in the models in order to determine how the removal of any silence or noise before speech in the dataset impacted model performance. The average value offsets of speech in the dataset can be observed in Table 4.1, Table 4.2, and Table 4.3 for the CREMA-D, RAVDESS, and SAVEE datasets, respectively.

Table 4.4: Model Accuracy Comparison on Librosa Data Load Parameters.

Dataset	Target Duration	Target Sample Rate	Offset	Accuracy			
				SVM	LeNet	CRNN	T. Encoder and CNN
CREMA-D	1.28	16 kHz	0	42.85%	39.62%	41.53%	46.94%
CREMA-D	1.28	22.5 kHz	0	45.00%	40.90%	40.99%	45.74%
CREMA-D	1.28	16 kHz	0.3	47.88%	40.09%	45.30%	52.05%
CREMA-D	5	16 kHz	0	**51.91%**	44.46%	**56.32%**	**57.02%**
CREMA-D	5	22.5 kHz	0	51.85%	43.59%	53.63%	56.35%
CREMA-D	5	16 kHz	0.3	51.51%	45.60%	54.84%	56.48%

Dataset	Target Duration	Target Sample Rate	Offset	Accuracy			
				SVM	LeNet	CRNN	T. Encoder and CNN
CREMA-D	2.54	16 kHz	0	50.91%	45.33%	53.76%	56.15%
CREMA-D	2.54	22.5 kHz	0	51.71%	44.86%	52.15%	54.60%
CREMA-D	2.54	16 kHz	0.3	51.85%	**46.07%**	52.02%	54.13%
RAVDESS	3.07	48 kHz	0.8	69.81%	61.32%	65.09%	75.47%
RAVDESS	3.07	48 kHz	0	69.81%	63.21%	61.32%	73.58%
RAVDESS	3.07	22.5 kHz	0	68.87%	60.38%	52.83%	74.53%
RAVDESS	5.27	48 kHz	0.8	68.87%	64.15%	66.04%	70.28%
RAVDESS	5.27	48 kHz	0	**72.64%**	59.43%	**66.98%**	**78.30%**
RAVDESS	5.27	22.5 kHz	0	65.09%	61.32%	62.26%	71.70%
RAVDESS	3.73	48 kHz	0.8	70.75%	**66.98%**	63.21%	75.47%
RAVDESS	3.73	48 kHz	0	71.70%	63.21%	61.32%	72.64%
RAVDESS	3.73	22.5 kHz	0	66.98%	53.68%	62.26%	68.87%
SAVEE	1.63	44.1 kHz	0.5	63.10%	57.14%	42.86%	64.29%
SAVEE	1.63	44.1 kHz	0	63.10%	50.00%	57.14%	71.43%
SAVEE	1.63	22.5 kHz	0	66.67%	59.52%	57.14%	69.05%
SAVEE	3.84	44.1 kHz	0.5	76.19%	**64.29%**	69.05%	69.05%
SAVEE	3.84	44.1 kHz	0	69.05%	52.38%	54.76%	59.52%
SAVEE	3.84	22.5 kHz	0	75.00%	61.90%	59.52%	73.81%
SAVEE	7.14	44.1 kHz	0.5	73.81%	57.14%	54.76%	66.67%
SAVEE	7.14	44.1 kHz	0	**78.57%**	50.00%	**69.05%**	**76.19%**
SAVEE	7.14	22.5 kHz	0	76.19%	54.76%	57.14%	66.67%

This study evaluated the performance of various machine learning algorithms for audio-based speech recognition for audio by means of assessment. The algorithms that were used include SVM, LeNet-based CNN, CRNN, and the Pararel Transformer Encoder with CNN Architecture. The primary evaluation metric used in this study was accuracy. The accuracy of the models was compared when they were trained on different combinations of data loading parameters that were tested in this study.

Based on the results, it is evident that data loading parameters have a crucial role in the performance of machine learning models for audio speech recognition. Table 4.4 showed that the optimal data loading parameters varied across different machine learning architectures. Specifically, the SVM, CRNN, and the Pararel Transformer Encoder with CNN Architecture achieved their highest accuracy with a longer duration, a native sample rate, and no offset. On the other hand, for the LeNet CNN, an average duration, a native sample rate, and an offset proved to be the optimal parameters. Notably, these findings were consistent across all tested datasets. Within the table, three values will be bolded for each machine learning architecture, indicating the highest accuracy achieved for each of the tested datasets.

The variability in optimal data loading parameters suggests that each machine learning architecture is sensitive to specific aspects of the input data. This can be attributed to the fact that each machine-learning architecture has its own unique characteristics and requirements. For example, the optimal data loading parameters for SVM, CRNN, and the Pararel Transformer Encoder with CNN Architecture have the same optimal data loading parameters. This similarity in optimal data loading parameters suggests that these models may be sensitive to the duration of audio samples. Longer audio samples may provide more context for the network to analyze, which can lead to improved accuracy. Additionally, using the native sample

rate may preserve the quality of the audio signal, which may also contribute to better performance. On the other hand, LeNet is an old architecture designed for digit image classification, so it may be better suited to audio samples with a shorter duration and a lower sample rate.

This experiment emphasizes the importance of carefully selecting appropriate data loading parameters in the preparation of audio speech recognition datasets. The optimal data loading parameters should be chosen based on the characteristics of the dataset and the machine learning architecture being used to obtain the best performance.

b. Dataset Splitting

The process of data splitting involves partitioning a dataset into two or more subsets, with the aim of training and assessing machine learning models. During this process, a subset of the dataset is designated for training the model, while the remaining portion is reserved for testing or validation. This step plays a crucial role in model development, as it enabled the evaluation of the model performance on previously unseen data.

This section aims to evaluate the performance of different machine learning models by testing three different data splitting ratios using the optimal data loading parameters for each model shown in Table 4.4. Like the preceding table, the evaluation tables incorporate three bolded values for each machine learning architecture. These emphasized values serve as indicators of the highest achieved accuracy for each of the datasets subjected to testing. The ratios tested for the machine learning SVM model include 80:20, 90:10, and 70:30, where the first number represents the percentage of data used for training, and the second number represents the percentage of data used for testing. The results of the evaluation of the SVM model were presented in Table 4.5.

Table 4.5: SVM Model Accuracy Comparison on Different Split Data Ratio.

Dataset	Ratio (%)		Accuracy
	Train	Test	SVM
CREMA-D	80 (5953)	20 (1489)	51.91%
CREMA-D	90 (6697)	10 (745)	**53.96%**
CREMA-D	70 (5209)	30 (2233)	50.02%
RAVDESS	80 (844)	20 (212)	70.28%
RAVDESS	90 (950)	10 (106)	**72.64%**
RAVDESS	70 (739)	30 (317)	68.77%
SAVEE	80 (336)	20 (84)	73.81%
SAVEE	90 (378)	10 (42)	**78.57%**
SAVEE	70 (294)	30 (126)	67.46%

In Table 4.6, apart from the SVM model, the assessment of deep learning models such as CRNN, LeNet, and Parallel Transformer Encoder with CNN Architecture was conducted, taking into account various data splitting ratios. Differing from the SVM model, the dataset for these models was divided into three distinct parts: training, testing, and validation. The evaluation encompassed three different ratios for data splitting, namely 80:10:10, 90:5:5, and 70:15:15. This comprehensive analysis explores the performance of these deep learning models in comparison to the SVM model under different data partitioning strategies, providing valuable insights into their effectiveness for emotion recognition tasks. By considering various data splitting ratios and conducting a comprehensive evaluation, this experiment examines the

performance of these deep learning models and also learn the impact of different data partitioning strategies on their effectiveness for accurate emotion recognition.

Table 4.6: Deep Learning Based Model Accuracy Comparison on Different Split Data Ratio.

Dataset	Ratio (%)			Accuracy		
	Train	Test	Validation	LeNet	CRNN	T. Encoder and CNN
CREMA-D	80 (5953)	10 (744)	10 (745)	49.46%	54.30%	57.39%
CREMA-D	90 (6697)	5 (372)	5 (373)	**52.96%**	**55.38%**	**59.68%**
CREMA-D	70 (5209)	15 (1116)	15 (1117)	48.66%	53.58%	54.57%
RAVDESS	80 (844)	10 (106)	10 (106)	62.26%	62.26%	74.53%
RAVDESS	90 (950)	5 (53)	5 (53)	**64.15%**	**69.81%**	**79.25%**
RAVDESS	70 (739)	15 (158)	15 (159)	57.59%	64.56%	70.25%
SAVEE	80 (336)	10 (42)	10 (42)	64.29%	66.67%	76.19%
SAVEE	90 (378)	5 (21)	5 (21)	**66.67%**	**71.43%**	**83.33%**
SAVEE	70 (249)	15 (63)	15 (63)	58.73%	68.25%	71.43%

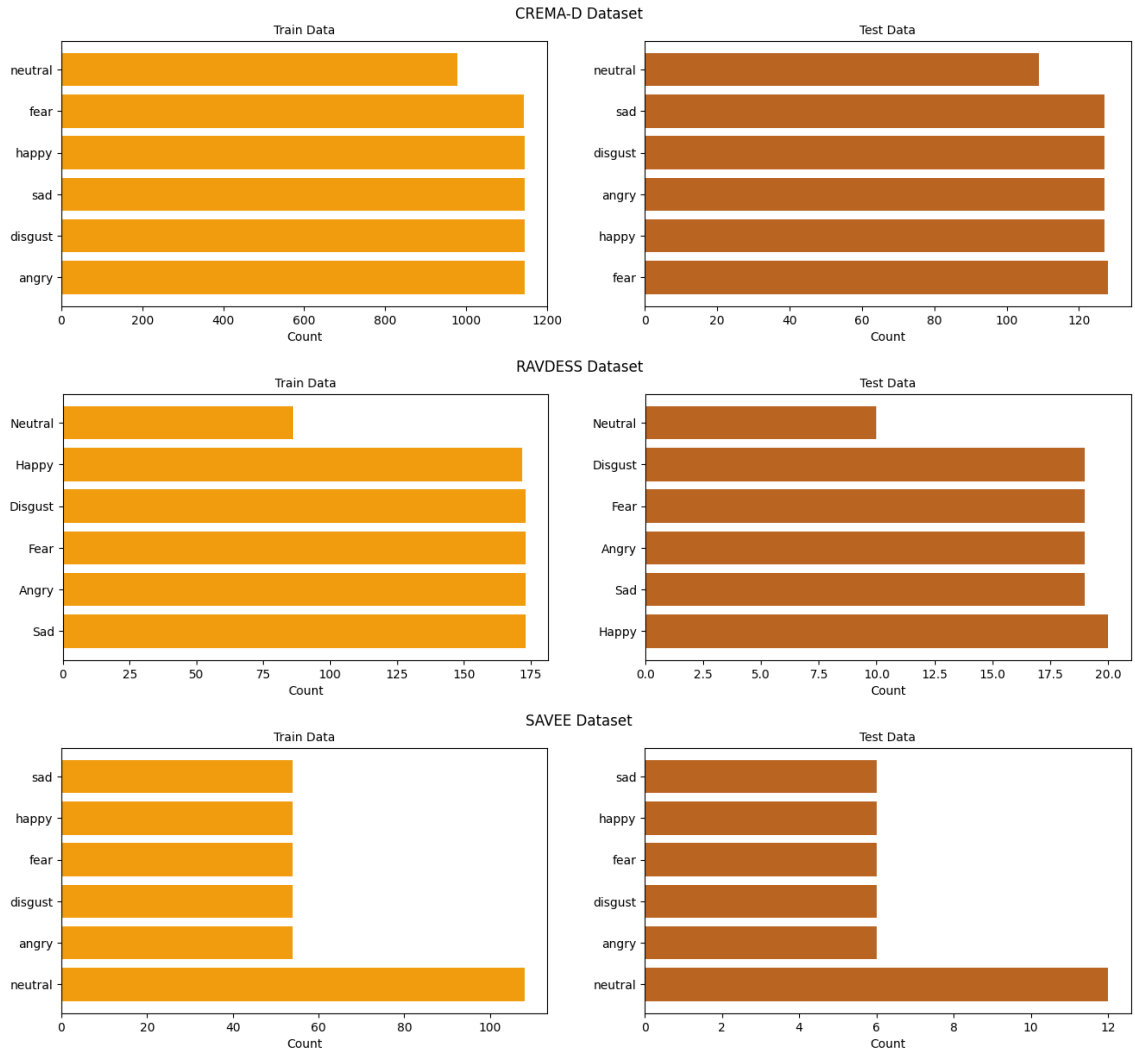


Figure 4.1: SVM Data Distributions.

The evaluation results presented in Table 4.5 and Table 4.6 offer valuable insights into the optimal data split ratios for the machine learning algorithm SVM as well as the deep learning architectures, including LeNet CNN, CRNN, and the Pararel Transformer Encoder with CNN Architecture. Interestingly, it was observed that the highest model accuracy for SVM was achieved with a data split ratio of 90% training data and 10% testing data. Similarly, for the deep learning architectures, the best-split ratio was determined to be 90% training data, 5% testing data, and 5% validation data. These findings provide valuable guidance in selecting the appropriate data split ratio for achieving optimal performance. Additionally, Figure 4.1 and Figure 4.2 illustrate the distribution of data across each emotion class, based on the identified best data splitting ratio for each machine learning and deep learning algorithm. These plots offer visual representations of the data distribution, providing further insights into the composition of the dataset for each emotion class.

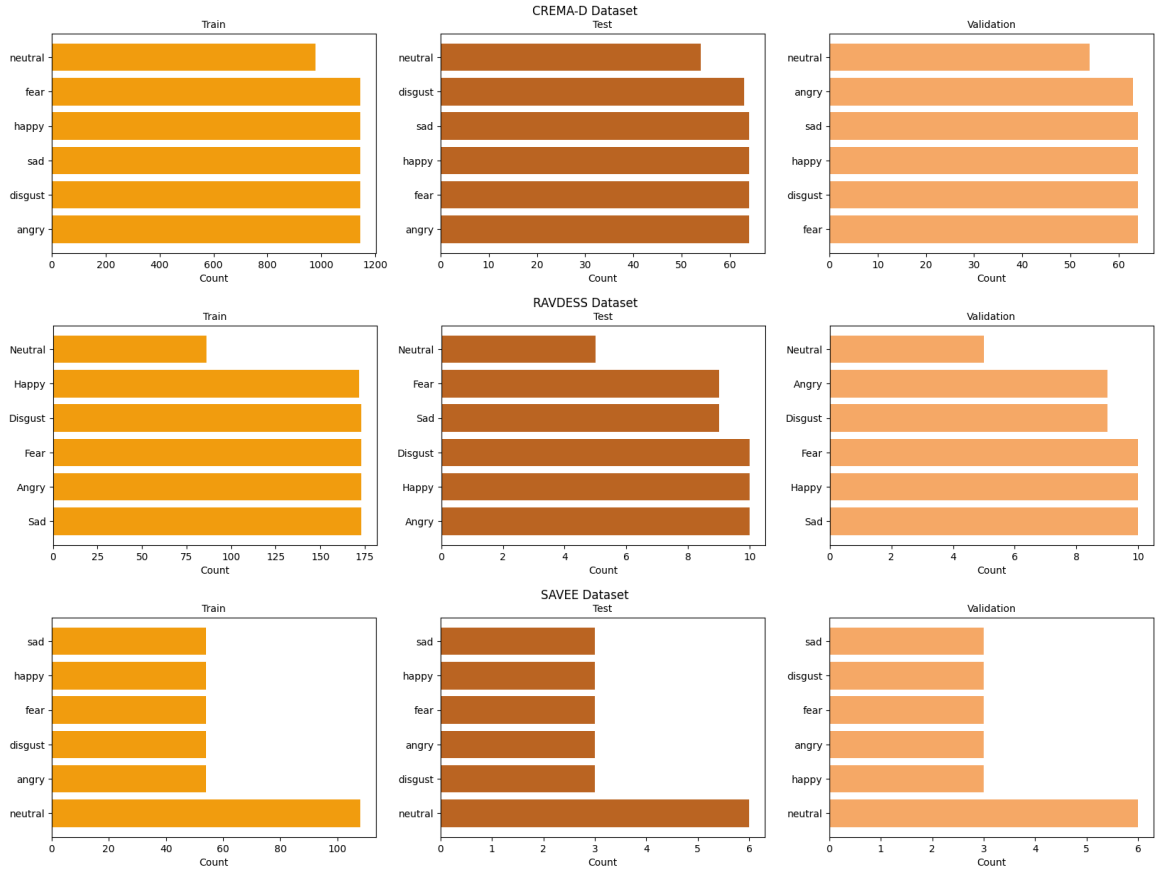


Figure 4.2: Deep Learning Based Data Distributions.

4.1.3 Feature Extraction

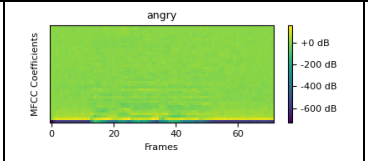
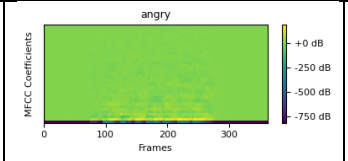
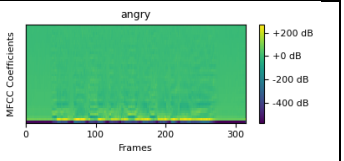
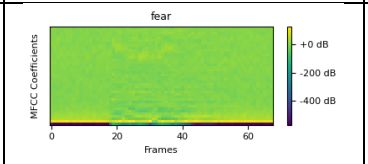
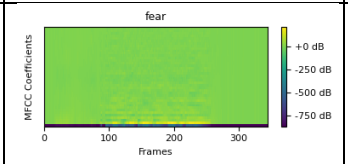
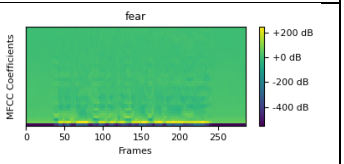
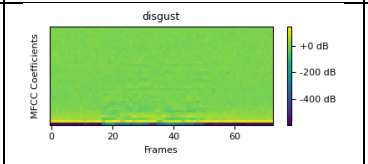
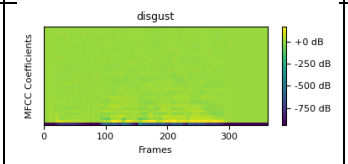
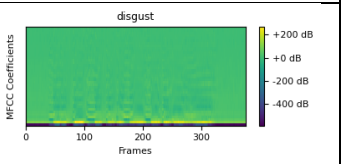
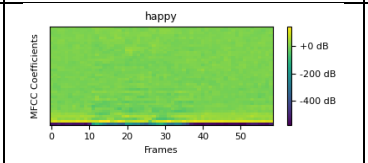
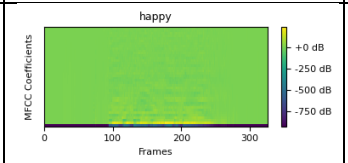
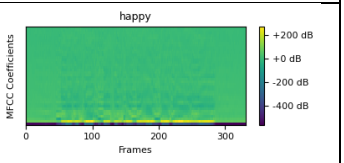
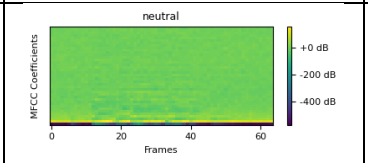
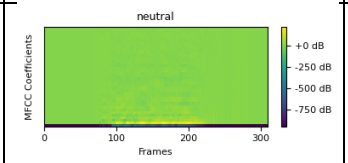
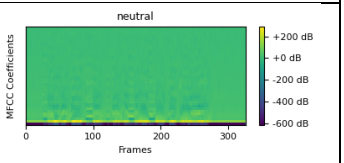
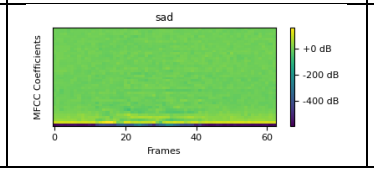
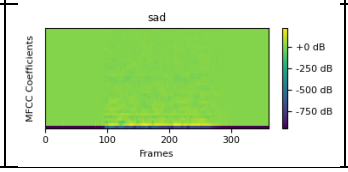
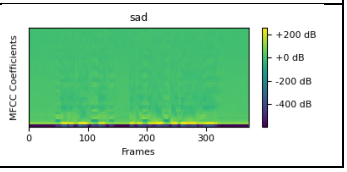
Audio feature extraction plays a pivotal role in numerous audio processing applications, particularly in the realm of speech recognition systems. Within the realm of audio features, MFCCs have been widely used due to their effectiveness in capturing the spectral characteristics of audio signals. In this study, the librosa library was employed to extract MFCCs from the audio signals. Optimal parameter values were carefully chosen for the MFCC extraction, which included an MFCC value of 40, an FFT window length of 1024, a window size of 512, a Hanning windowing function, and 128 Mel frequency bands.

The choice of 40 MFCC coefficients aligns with established practices in speech and music processing literature. This number of coefficients adequately captures the spectral characteristics of audio signals while maintaining computational efficiency. The FFT window

length strikes a balance between frequency resolution and time resolution, whereas the window size is employed to reduce the computational time required for STFT calculations. Application of the Hanning windowing function serves to mitigate spectral leakage arising from windowing and enhance segmentation accuracy. Additionally, the utilization of Mel frequency bands allows for a more faithful representation of the frequency response of the human auditory system, achieved by mapping the frequency domain to the Mel scale. The selection of the number of Mel frequency bands is guided by well-established psychoacoustic principles rooted in human hearing.

Furthermore, Table 4.7 provides the signal characteristics represented by the MFCC plot for each of the emotion classes in the CREMA-D, RAVDESS, and SAVEE datasets using the aforementioned parameters. For a more comprehensive understanding of these signal characteristics, additional figures illustrating the MFCC plots can be found in the Appendix. These MFCC plots offer valuable insights, revealing that different emotions possess distinct spectral characteristics that can be effectively captured through the utilization of MFCCs.

Table 4.7: Speech Signals Features.

Emotion	CREMA-D	RAVDESS	SAVEE
Angry			
Fear			
Disgust			
Happy			
Neutral			
Sad			

4.2 Experiment Results

This section will present the results of the experiments conducted to evaluate the performance of the Pararel Transformer Encoder with CNN Architecture for sound emotion recognition system. Several machine learning models are tested on the dataset, including the

SVM model and the Convolutional based architecture models to compare with the Pararel Transformer Encoder with CNN Architecture model. Each model will have a report of the overall accuracy, confusion matrix, and F1 scores for each emotion class in the dataset. The performance of each model will be compared to identify the strengths and weaknesses of each approach.

4.2.1 Support Vector Machine (SVM)

The first evaluated model is the performance of the SVM model on the sound emotion recognition task. For this experiment, the model used the previously tested optimal data preprocessing steps from 4.1, which include loading the audio data with the librosa library with the optimal parameters of the longest duration, a native sample rate, and without any offsets, and a 90:10 data split for training and testing the SVM model. In addition, the MFCCs feature extraction was applied to the training and testing set for the SVM model.

To determine the optimal hyperparameters for the SVM model in this experiment, the GridSearch technique was utilized. The GridSearch approach explores and tests various combinations of hyperparameters within the model. As a result, the optimal set of hyperparameters was identified in Table 3.2, which included a value of 10 for C, 0.01 for gamma, 'rbf' for the kernel, and a value of True for the probability parameter.

Table 4.8, Table 4.9, and Table 4.10 contain detailed confusion matrices that provide a comprehensive overview of the performance of the SVM model in classifying emotions within the CREMA-D, RAVDESS, and SAVEE datasets, respectively. These confusion matrices, created using the hyperparameters mentioned earlier, offer a breakdown of the actual and predicted classes. Each row in the confusion matrix corresponds to the actual emotion classes, while each column represents the predicted emotion classes. By examining the values within the matrices, potential misclassifications or patterns in the classification process can be identified.

Table 4.8: SVM CREMA-D Confusion Matrix.

Class	Falsely Predicted Emotion						Total Truth	Total False
	Angry	Fear	Disgust	Happy	Neutral	Sad		
Angry	87	6	8	19	6	1	**87**	40
Fear	12	55	13	11	18	18	55	72
Disgust	10	13	49	13	14	29	49	**79**
Happy	15	7	14	69	17	5	69	58
Neutral	2	13	6	17	58	13	58	51
Sad	0	8	17	2	16	84	84	43

Table 4.9: SVM RAVDESS Confusion Matrix.

Class	Falsely Predicted Emotion						Total Truth	Total False
	Angry	Fear	Disgust	Happy	Neutral	Sad		
Angry	16	2	1	0	0	0	**16**	3
Fear	1	12	2	1	0	3	12	**7**
Disgust	0	2	16	0	1	0	**16**	3
Happy	1	0	3	14	1	1	14	6
Neutral	0	1	0	0	5	4	5	5
Sad	0	2	0	2	1	14	14	5

Table 4.10: SVM SAVEE Confusion Matrix.

Class	Falsely Predicted Emotion						Total Truth	Total False
	Angry	Fear	Disgust	Happy	Neutral	Sad		
Angry	3	0	1	2	0	0	3	**3**
Fear	0	6	0	0	0	0	6	0
Disgust	1	0	4	0	1	0	4	2
Happy	2	0	0	4	0	0	4	2
Neutral	0	0	0	0	11	1	**11**	1
Sad	0	0	0	0	1	5	5	1

The confusion matrices presented in Table 4.8, Table 4.9, and Table 4.10 provide valuable insights into the performance of the sound emotion recognition system developed in this study using SVM. The results indicate that the system struggled to distinguish between certain emotions in the datasets. In addition to providing an overall evaluation, these tables also feature bolded values that represent the highest true and false values for a certain emotion in each dataset, further enhancing the interpretation of the results. In the case of the CREMA-D dataset, the confusion matrix in Table 4.8 shows that the system found it challenging to differentiate between angry and happy, as well as between disgusted and sad. This is reflected in the number of incorrect predictions for these emotion classes. Likewise, for the RAVDESS dataset, the confusion matrix in Table 4.9 suggests that the system had difficulty distinguishing between neutral and sad. Furthermore, Table 4.10 depicts the confusion matrix for the SAVEE dataset, which suggests that the system had challenges in distinguishing between angry and happy emotions.

In addition to the confusion matrices, the precision, recall, and F1 score are used as metrics to evaluate the performance of the model in terms of how accurately it classifies the different emotions in the dataset. It is worth noting that in the tables, the bolded values represent the highest value for individual emotions for each dataset. By examining these metrics, it is possible to draw conclusions about the model performance in accurately classifying different emotions. Precision measures the proportion of correctly predicted positive cases out of all predicted positive cases, while recall measures the proportion of correctly predicted positive cases out of all actual positive cases. The F1 score is a weighted average of precision and recall and provides a measure of the overall accuracy of the model in classifying the different emotions. Table 4.11 presents these evaluation metrics for each of the emotion classes in the CREMA-D, RAVDESS, and SAVEE datasets.

Table 4.11: Comparison of SVM Precision, Recall, and F1 Score for Emotion Classification.

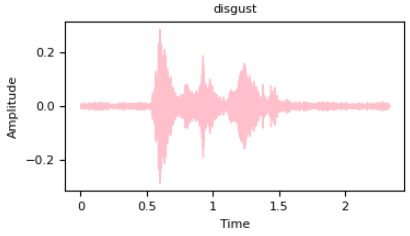
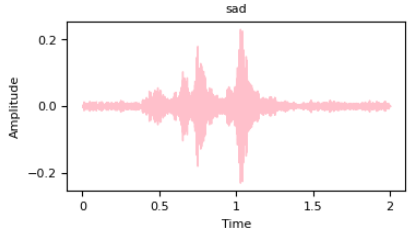
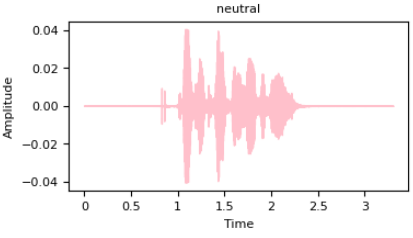
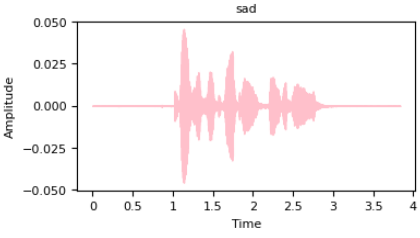
Emotion	Precision			Recall			F1-Score		
	Crema	RAVDESS	SAVEE	Crema	RAVDESS	SAVEE	Crema	RAVDESS	SAVEE
Angry	**0.69**	**0.89**	0.50	**0.69**	**0.84**	0.50	**0.69**	**0.86**	0.50
Fear	0.46	0.73	**1.00**	0.38	**0.84**	**1.00**	0.42	0.78	**1.00**
Disgust	0.54	0.63	0.80	0.43	0.63	0.67	0.48	0.63	0.73
Happy	0.53	0.82	0.67	0.54	0.70	0.67	0.53	0.76	0.67
Neutral	0.45	0.62	0.85	0.53	0.50	0.92	0.49	0.56	0.88
Sad	0.56	0.64	0.83	0.66	0.74	0.83	0.61	0.68	0.83

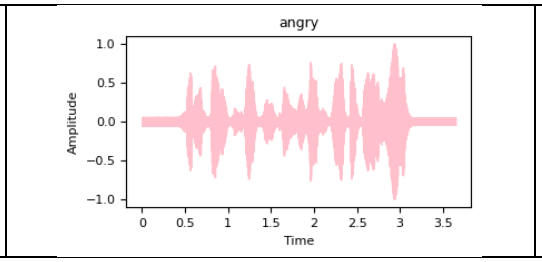
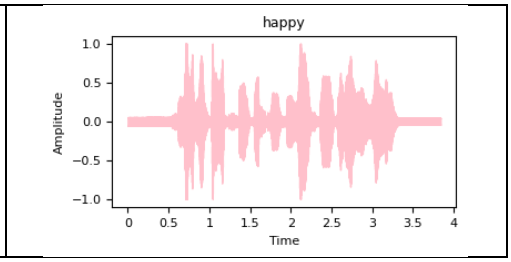
The results of the SVM model performance on the datasets, as measured by F1 scores, showed that the SVM model exhibited superior performance in the RAVDESS dataset compared to the CREMA-D dataset. Specifically, the SVM model achieved an average accuracy of 72.64% in the RAVDESS dataset, while the CREMA-D dataset yielded a comparatively lower average accuracy of 53.96%. These results suggest that the RAVDESS dataset may be more suitable for emotion classification tasks, given its higher accuracy and potentially more distinct emotional expressions. Furthermore, the SVM model also demonstrated favorable performance on the SAVEE dataset, achieving an average accuracy score of 78.57%. These outcomes highlight the effectiveness of the SVM model in accurately classifying emotions across various datasets and emphasize the potential benefits of utilizing the RAVDESS and SAVEE datasets for emotion classification research and applications.

Moreover, the F1 scores for individual emotions also varied between datasets. For the RAVDESS dataset, the model achieved high F1 scores for Angry (0.86), Fear (0.78), and Sad (0.68), indicating good performance in classifying these emotions. However, the model struggled to classify Neutral (0.56) and Disgust (0.63) emotions in this dataset. Conversely, the CREMA-D dataset demonstrated the highest F1 score for Angry (0.69) emotion, but Fear (0.42) and Disgust (0.48) emotions were more challenging to classify. The SAVEE dataset showed excellent performance in classifying Fear (1.00) and Neutral (0.88) emotions but struggled to accurately classify Angry (0.50), Happy (0.67), and Disgust (0.73) emotions. These results indicate that the SVM model ability to classify emotions may differ depending on the specific emotion being classified and the dataset used.

Table 4.12 also includes amplitude plots that highlight the emotions which the SVM model struggles to differentiate from one another. Within the CREMA-D dataset, the SVM model faces challenges in distinguishing between the emotions of disgust and sadness. The amplitude plots reveal that these emotions share a maximum amplitude value of approximately 0.2, accompanied by similar wave structures that appear to mirror each other. Similarly, the RAVDESS dataset poses difficulties in distinguishing between the emotions of neutrality and sadness. Both emotions exhibit average maximum amplitude values ranging from 0.04 to 0.05, and their wave structures display striking resemblances, further exacerbating the challenge of differentiation using the SVM model. Likewise, in the case of the SAVEE dataset, the SVM model encounters difficulty in distinguishing between the emotions of anger and happiness. These emotions exhibit a maximum amplitude of 1.0, and their wave structures demonstrate notable similarities, contributing to the complexities in differentiation using the SVM model.

Table 4.12: Amplitude Comparison Between Closely Related Emotions in the Dataset.

CREMA-D		
RAVDESS		

SAVEE		

4.2.2 Convolutional Based Architecture

In this section, the results of the experiment conducted on the Convolution based architecture model will be explained, which includes LeNet CNN and the Convolutional Recurrent Neural Network, used in this study.

a. LeNet-based Convolutional Neural Network (CNN)

The LeNet CNN model architecture used in the study was based on the architecture proposed by (LeCun, Bottou, Bengio, & Haffner, 1998). Table 3.4 provides a detailed overview of the architecture of LeNet CNN model used in the study, including the number of filters, kernel size, pooling size, and activation functions used in each layer. The model was trained on the CREMA-D, RAVDESS, and SAVEE datasets. The training process involved feeding the model with the training dataset and optimizing the model parameters to minimize the loss function. The loss function was used to calculate the error between the predicted output and the actual output of the model. The optimization process used the backpropagation algorithm to adjust the weights and biases of the model, which contributed to the reduction in the loss function.

The training process of the LeNet CNN model architecture was monitored using a loss curve and an accuracy curve. The loss curve displayed the changes in the loss function over time as the training progressed, with the objective of minimizing the loss function to improve the predictions of the model. The accuracy curve displayed the changes in accuracy over time, with the objective of maximizing accuracy to ensure the model makes correct predictions.

Both the loss curve and accuracy curve during the training process of the LeNet CNN architecture model were shown in Figure 4.3, Figure 4.4, and Figure 4.5 for each of the datasets used in this study. The loss curves showed a decreasing trend, which implies that the model was learning from the training dataset and making better predictions. However, it appears that the performance of the model on the three datasets differs significantly in terms of the number of epochs required to converge and the achieved accuracy.

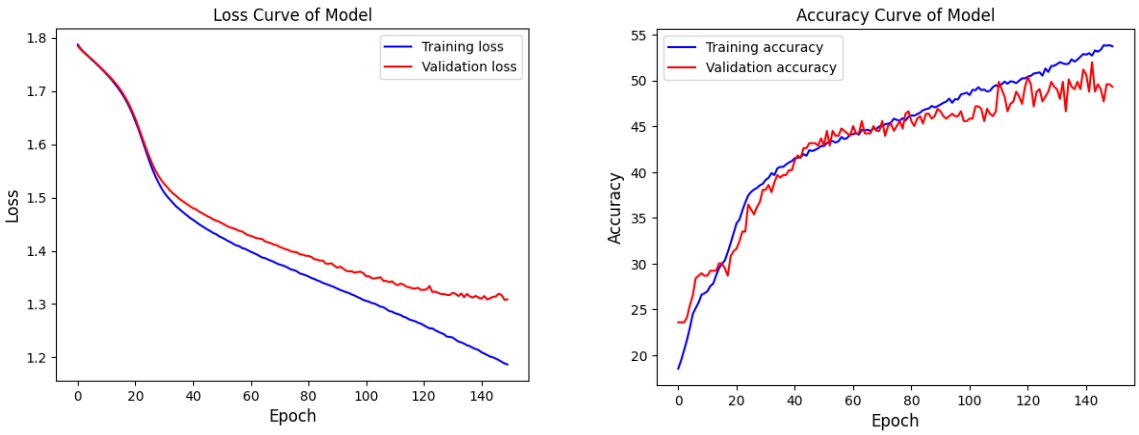


Figure 4.3: CREMA-D Accuracy and Loss Curve During Training Process.

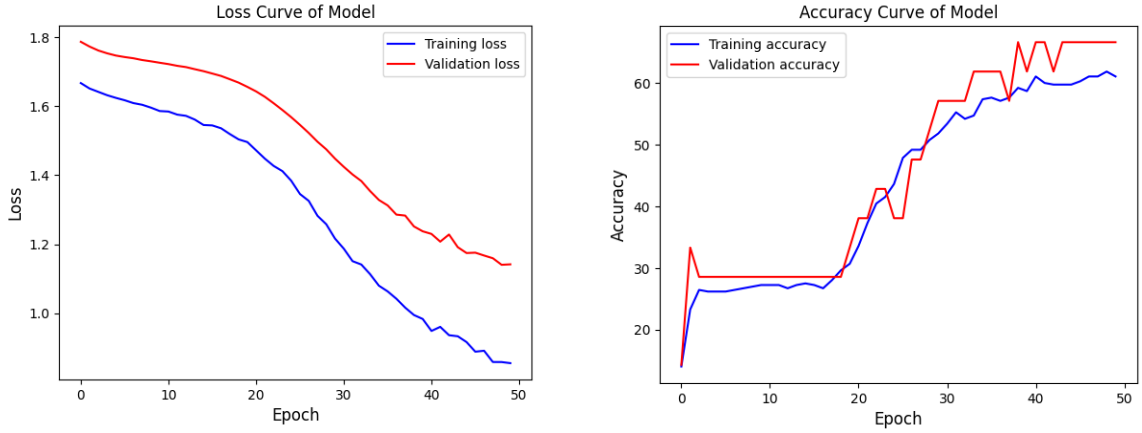


Figure 4.4: SAVEE Accuracy and Loss Curve During Training Process.

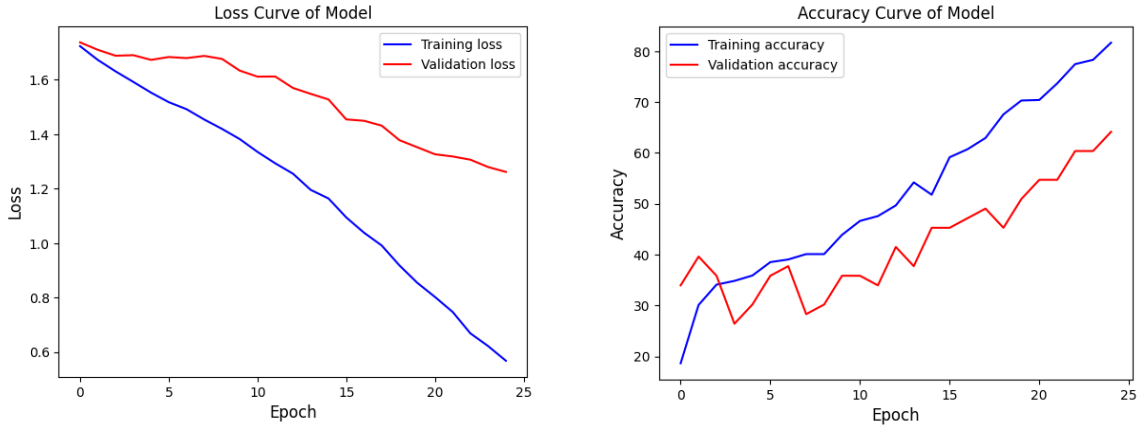


Figure 4.5: RAVDESS Accuracy and Loss Curve During Training Process.

The number of epochs needed for a model to converge or start overfitting can vary depending on the complexity of the dataset and the architecture of the neural network. The results show that the model performs best on the SAVEE dataset, achieving an accuracy of 67% with 50 epochs before overfitting occurs. The SAVEE dataset is relatively small and less complex compared to the other datasets, making it easier for the model to learn the features and generalize them to new data. On the other hand, the model requires more epochs to converge on the more complex datasets, such as CREMA-D and RAVDESS, achieving accuracies of 52% and 64%, respectively.

The performance of the trained model will be evaluated by testing the model against the test set data of each dataset. To assess the performance of the LeNet CNN architecture model on the CREMA-D, RAVDESS, and SAVEE datasets, confusion matrices are presented in Table 4.13, Table 4.14, and Table 4.15, respectively. Notably, within these tables, bolded values indicate the highest true and false values corresponding to a specific emotion in each dataset. The confusion matrix contains rows that represent the actual classes and columns that represent the predicted classes.

Table 4.13: LeNet CREMA-D Confusion Matrix.

Class	Falsely Predicted Emotion						Total Truth	Total False
	Angry	Fear	Disgust	Happy	Neutral	Sad		
Angry	43	3	6	8	4	0	**45**	21
Fear	6	30	6	6	5	11	30	34

Class	Falsely Predicted Emotion						Total Truth	Total False
	Angry	Fear	Disgust	Happy	Neutral	Sad		
Disgust	5	5	36	2	9	7	36	28
Happy	11	12	6	22	8	5	22	**42**
Neutral	3	2	9	4	24	12	24	30
Sad	0	11	4	1	8	39	39	24

Table 4.14: LeNet RAVDESS Confusion Matrix.

Class	Falsely Predicted Emotion						Total Truth	Total False
	Angry	Fear	Disgust	Happy	Neutral	Sad		
Angry	5	0	1	3	0	0	5	4
Fear	0	5	1	1	0	2	5	4
Disgust	0	1	9	0	0	0	**9**	1
Happy	1	1	0	7	1	0	7	3
Neutral	0	0	0	0	4	1	4	1
Sad	0	2	1	2	1	4	4	**6**

Table 4.15: LeNet SAVEE Confusion Matrix.

Class	Falsely Predicted Emotion						Total Truth	Total False
	Angry	Fear	Disgust	Happy	Neutral	Sad		
Angry	1	0	0	0	1	1	1	**2**
Fear	0	2	0	0	0	1	2	1
Disgust	1	0	1	0	1	0	1	**2**
Happy	0	0	0	3	0	0	3	0
Neutral	0	0	0	0	6	0	**6**	0
Sad	0	0	0	0	2	1	2	1

Table 4.13, Table 4.14, and Table 4.15. These matrices serve as a valuable tool for obtaining insightful performance metrics of the model. The results show that the model performance varied across different datasets. In the case of the CREMA-D dataset, the confusion matrix revealed that the model had a high level of confusion between happy emotion and all other classes. On the other hand, the model trained on the RAVDESS dataset had difficulty distinguishing between sad and fear emotions. Finally, for the SAVEE dataset, due to the small size of the testing set, the model was able to accurately classify certain emotions, such as happy and neutral, for all the test data. The confusion matrix showed that all three instances of happy emotion were correctly predicted, and there were no incorrect predictions for this class. Similarly, all six instances of neutral emotion were classified accurately. However, there were instances where other emotions, such as sad, were wrongly classified as neutral by the model.

The performance of the model in correctly classifying emotions in the dataset can be evaluated using precision, recall, and F1-score in addition to the confusion matrices. Analyzing these metrics can provide insights into the model ability to accurately identify different emotions. While precision measures the proportion of correctly predicted positive cases out of all predicted positive cases, recall measures the proportion of correctly predicted positive cases out of all actual positive cases. The F1-score, which is a weighted average of precision and recalls, can be used to evaluate the overall accuracy of the model in classifying emotions. Table 4.16 illustrates the precision, recall, and F1-score for each emotion class in the CREMA-D, RAVDESS, and SAVEE datasets. It is important to note that in the tables, there are bolded

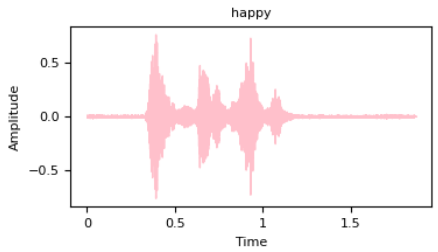
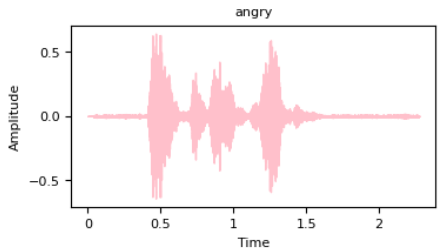
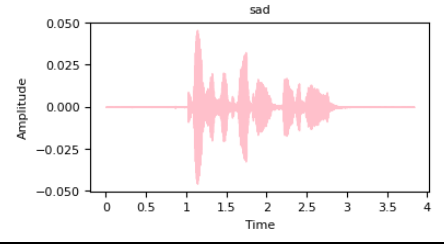
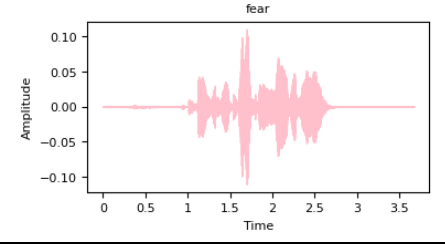
values which highlight the highest values corresponding to individual emotions within each dataset. The bolded entries provide a clear visual representation of the exceptional outcomes achieved by the model in accurately classifying specific emotions.

Table 4.16: Comparison of LeNet Precision, Recall, and F1 Score for Emotion Classification.

Emotion	Precision			Recall			F1-Score		
	Crema	RAVDESS	SAVEE	Crema	RAVDESS	SAVEE	Crema	RAVDESS	SAVEE
Angry	**0.63**	**0.83**	0.50	**0.67**	0.56	0.33	**0.65**	0.67	0.40
Fear	0.48	0.56	**1.00**	0.47	0.56	0.67	0.47	0.56	0.80
Disgust	0.54	0.75	**1.00**	0.56	**0.90**	0.33	0.55	**0.82**	0.50
Happy	0.51	0.54	**1.00**	0.34	0.70	**1.00**	0.41	0.61	**1.00**
Neutral	0.41	0.67	0.60	0.44	0.80	**1.00**	0.43	0.73	0.75
Sad	0.53	0.57	0.33	0.62	0.40	0.33	0.57	0.47	0.33

The utilization of F1 scores in evaluating the model performance provides valuable insights into its precision and recall capabilities, offering a comprehensive assessment of its ability to accurately classify both positive and negative cases. The results presented in Table 4.16 highlight the model performance on each dataset. Within the CREMA-D dataset, the LeNet model achieved an average accuracy of 52%, with the highest F1 score of 0.65 observed for angry emotions. Moving to the RAVDESS dataset, the average accuracy improved to 64%, accompanied by the highest F1 scores of 0.67 for angry emotions and an impressive 0.82 for disgusted emotions. Similarly, in the case of the SAVEE dataset, the LeNet model achieved an average accuracy of 67%, with a notable F1 score of 1.0 for happy emotions. However, the model encountered challenges in precisely classifying certain emotions across all three datasets, particularly sad emotions. This is evident from the comparatively lower F1 scores for sad emotions, which were 0.57, 0.47, and 0.33 for the CREMA-D, RAVDESS, and SAVEE datasets, respectively. These findings showed the limitations of the LeNet model in accurately identifying and classifying specific emotional states, particularly for emotions that prove to be more challenging to discern accurately.

Table 4.17: Amplitude Comparison Between Closely Related Emotions in the Dataset.

CREMA-D		
RAVDESS		

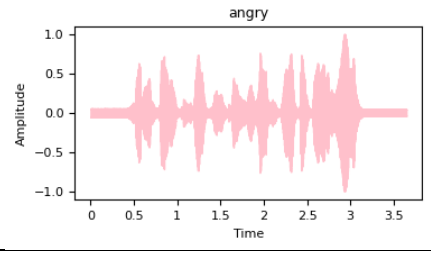
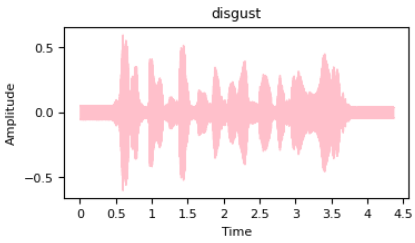
SAVEE		

Table 4.17 presents closely related emotions for each of the datasets used in this experiment, which provides a clear visualization of the similarities in the waveform structure of these emotions. The closely related emotions identified in the experiment results are happy and angry for the CREMA-D dataset, while the RAVDESS dataset shows similarities between sadness and fear. Furthermore, the SAVEE dataset demonstrates challenges in differentiating between the emotions of anger and disgust. These emotions are challenging to differentiate from each other due to the similarity in their acoustic features. These acoustic features include variations in pitch, vocal intensity, speech rate, and spectral energy distribution, which can overlap and contribute to confusion in emotion classification.

In the case of happy and angry emotions in the CREMA-D dataset, both emotions can have similar loudness and pitch, making it difficult to distinguish between them. Similarly, in the RAVDESS dataset, sad and fear emotions have similar low-frequency features, which can make them difficult to differentiate. Finally, for the SAVEE dataset, the similarity in the acoustic features of angry and disgust emotions, such as their low-frequency energy, can also make it challenging to distinguish between them. In short, the closely related emotions have similar acoustic characteristics, making them difficult for the LeNet CNN model to differentiate.

b. Convolutional Recurrent Neural Network (CRNN)

The CRNN model architecture for this study was based on a combination of a CNN and RNN, which is particularly suited for processing sequential data such as speech signals. Table 3.5 provides a detailed overview of the architecture of the CRNN model used in this study, including the number of filters, kernel size, pooling size, and activation functions used in each layer. The model was also trained on the CREMA-D, RAVDESS, and SAVEE datasets using the same training process as the LeNet CNN model. The training process involved feeding the model with the training dataset and optimizing the model parameters to minimize the loss function using the backpropagation algorithm.

Throughout the training process of the CRNN model, the evaluation of its performance involved monitoring both the loss curve and the accuracy curve. The loss curve depicted the changes in the loss function as the model underwent training, aiming to minimize the loss function and enhance the accuracy of predictions. Simultaneously, the accuracy curve displayed the progression of accuracy over time, striving to maximize accuracy and ensure precise predictions. These curves played a crucial role in determining the optimal number of epochs for training the CRNN model, striking a balance to prevent overfitting while achieving the highest possible accuracy. The analysis of these evaluation curves not only helped in determining the optimal training duration but also provide valuable insights into the learning dynamics and convergence behavior of the CRNN model. This iterative training approach empowered the model to effectively capture intricate patterns and nuances within speech signals, bolstering its capacity to generalize and generate reliable predictions on unseen data. Consequently, the utilization of loss and accuracy curves significantly contributed to the robustness and performance of the CRNN model in audio emotion recognition tasks.

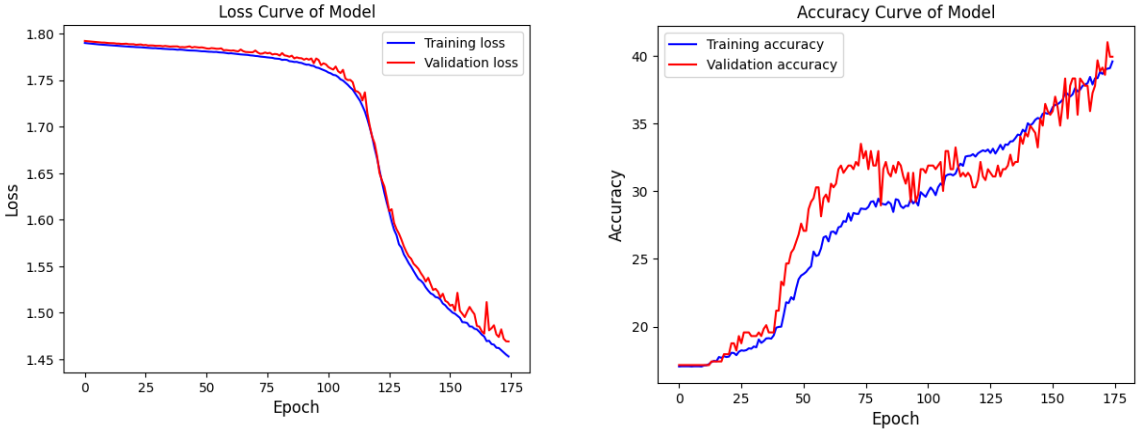


Figure 4.6: CREMA-D Accuracy and Loss Curve During Training Process.

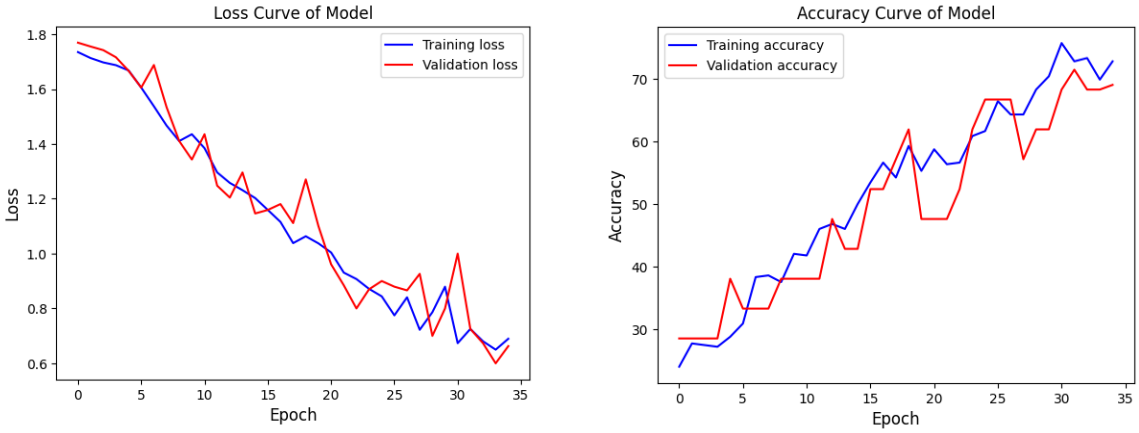


Figure 4.7: SAVEE Accuracy and Loss Curve During Training Process.

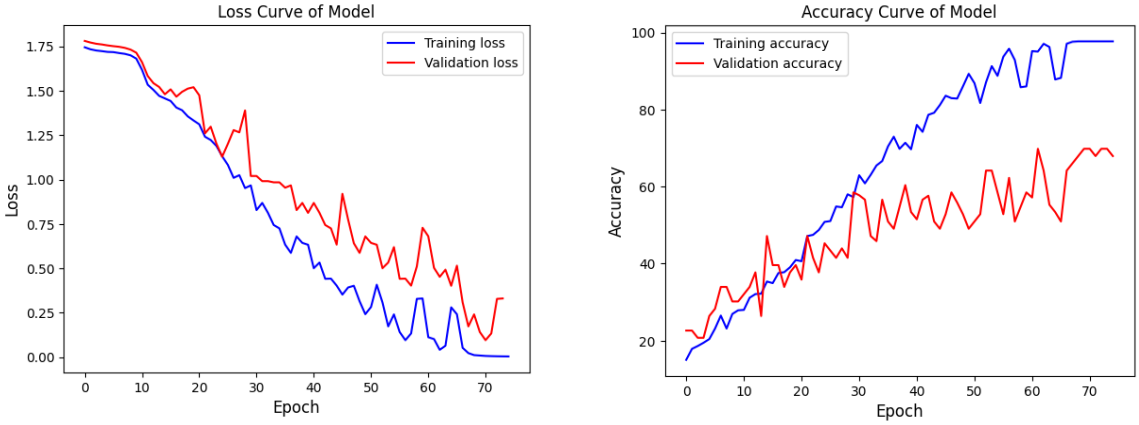


Figure 4.8: RAVDESS Accuracy and Loss Curve During Training Process.

The loss and accuracy curves for each dataset were plotted and shown in Figure 4.6, Figure 4.7, and Figure 4.8. The results indicate that the CRNN model architecture exhibits varying performance across the three datasets. The loss curves of the model showed a decreasing trend during the training process, indicating that the model was effectively learning from the training data. However, the number of epochs required for the model to converge and start overfitting the data varied across datasets. The model trained on the SAVEE dataset required the least number of epochs with a value of 35 to converge, followed by the RAVDESS dataset, which converged on epoch 75. Conversely, CREMA-D required the highest epoch value of 175 to

converge. The accuracy of the CRNN model architecture also differed significantly across the three datasets. The model achieved the highest accuracy of 71% on the SAVEE dataset, followed by the RAVDESS dataset with an accuracy of 69%, and the CREMA-D dataset with an accuracy of 55%. These results suggest that the performance of the model is influenced by the complexity and variability of the datasets.

Furthermore, to evaluate the performance of the CRNN model architecture on the three datasets, the model will be evaluated on their respective test sets. The confusion matrices, which provide a visualization of the model performance in terms of correctly and incorrectly classified samples for each emotion category, will be used in for the CREMA-D, SAVEE, and RAVDESS datasets. These confusion matrices will enable a more detailed analysis of the model ability to distinguish between different emotions and its overall accuracy on the test set. Notably, these tables highlight the highest true and false values for a specific emotion in each dataset through the use of bolded entries. This emphasis on bolded values facilitates a clear identification of the exceptional performance of the model in accurately classifying certain emotions and its potential challenges in accurately classifying others. Moreover, the confusion matrices allow for the identification of patterns in misclassifications, such as recurring confusion between specific pairs of emotions.

Table 4.18: CRNN CREMA-D Confusion Matrix.

Class	Falsely Predicted Emotion						Total Truth	Total False
	Angry	Fear	Disgust	Happy	Neutral	Sad		
Angry	41	0	8	12	3	0	**41**	23
Fear	0	34	4	9	7	10	34	30
Disgust	5	6	34	4	7	8	34	30
Happy	3	10	5	35	9	2	35	29
Neutral	0	5	8	3	31	7	31	23
Sad	0	13	6	2	11	31	31	**32**

Table 4.19: CRNN RAVDESS Confusion Matrix.

Class	Falsely Predicted Emotion						Total Truth	Total False
	Angry	Fear	Disgust	Happy	Neutral	Sad		
Angry	7	0	0	2	0	0	7	2
Fear	0	4	0	3	0	2	4	**5**
Disgust	0	0	7	3	0	0	7	3
Happy	0	0	0	8	0	2	8	2
Neutral	0	0	0	0	2	3	2	3
Sad	0	0	0	1	0	9	**9**	1

Table 4.20: CRNN SAVEE Confusion Matrix.

Class	Falsely Predicted Emotion						Total Truth	Total False
	Angry	Fear	Disgust	Happy	Neutral	Sad		
Angry	1	0	2	0	0	0	1	**2**
Fear	0	2	0	1	0	0	2	1
Disgust	0	0	2	0	0	1	2	1
Happy	1	0	0	2	0	0	2	1
Neutral	0	0	0	0	6	0	**6**	0
Sad	0	0	0	0	1	2	2	1

The confusion matrix of the three datasets evaluated on the CRNN model architecture depicted in Table 4.18, Table 4.19, and Table 4.20 reveals interesting insights into the model performance on different emotional expressions. The CREMA-D dataset exhibits a high level of confusion between fear and sadness, which suggests that the model may not be effectively distinguishing between these two emotions. Similarly, the RAVDESS dataset shows a significant degree of misunderstanding between neutral and sad, suggesting that the model would find it difficult to distinguish minute variations between these two emotional states. The SAVEE dataset, on the other hand, showed a high level of confusion between anger and disgust, which may be due to the similarity in the acoustic features of these two emotions.

In addition to the confusion matrix, precision, recall, and F1-score are also used as evaluation metrics to assess the performance of the CRNN model in correctly classifying emotions in a given dataset. The results obtained from these metrics complement the insights obtained from the confusion matrix, and together they provide a comprehensive evaluation of the model performance. Table 4.21 presents the precision, recall, and F1-score for each emotion class in the CREMA-D, RAVDESS, and SAVEE datasets, providing a detailed analysis of the model accuracy in classifying emotions. It is important to note that the bolded values within the table signify the highest achieved values for individual emotions in each dataset. These bolded entries highlight exceptional accuracy of the model in classifying specific emotions, augmenting the overall analysis of its performance.

Table 4.21: Comparison of CRNN Precision, Recall, and F1 Score for Emotion Classification.

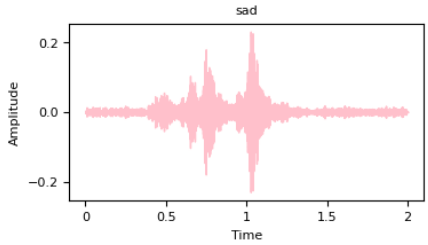
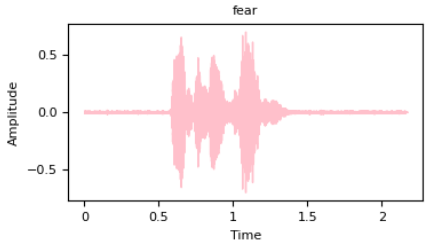
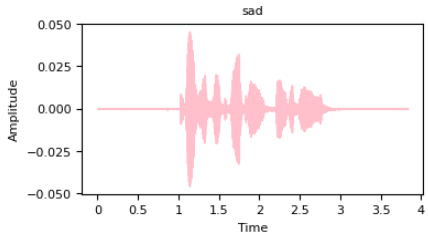
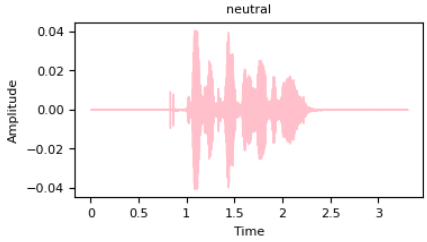
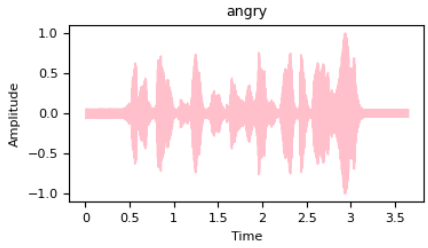
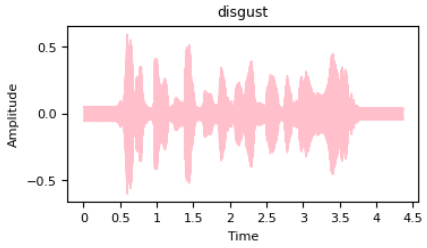
Emotion	Precision			Recall			F1-Score		
	Crema	RAVDESS	SAVEE	Crema	RAVDESS	SAVEE	Crema	RAVDESS	SAVEE
Angry	**0.84**	**1.0**	0.50	**0.64**	0.78	0.33	**0.73**	**0.88**	0.40
Fear	0.50	**1.0**	**1.00**	0.53	0.44	0.67	0.52	0.62	0.80
Disgust	0.52	**1.0**	0.50	0.53	0.70	0.67	0.53	0.82	0.57
Happy	0.54	0.47	0.67	0.55	0.80	0.67	0.54	0.59	0.67
Neutral	0.46	**1.0**	0.86	0.57	0.40	**1.00**	0.51	0.57	**0.92**
Sad	0.53	0.56	0.67	0.49	**0.90**	0.67	0.51	0.69	0.67

The F1 scores results in Table 4.21 demonstrated that the CRNN model accuracy varies across different datasets and emotions. The CREMA-D dataset has an average accuracy of 55% with the highest F1 score of 0.73 for the angry emotion. The RAVDESS dataset, on the other hand, has an average accuracy of 70%, with the highest F1 scores of 0.88 and 0.82 for angry and disgusted emotions, respectively. Finally, the SAVEE dataset achieved an average accuracy of 71%, with the highest F1 score of 0.92 for neutral emotion. The model performed well for some emotions, such as anger and neutral, but encountered challenges in accurately classifying specific emotions across all three datasets, especially for sad emotions, as demonstrated by the lower F1 scores for these emotions. For instance, the F1 scores for sad emotions were 0.51, 0.69, and 0.67 for the CREMA-D, RAVDESS, and SAVEE datasets, respectively. Overall, these results suggest that the CRNN model performance is dependent on the dataset and emotion being classified.

The comparison of waveforms between closely related emotions in each dataset depicted in Table 4.22 provides insights into the challenges encountered by the CRNN model in accurately classifying these emotions. In the CREMA-D dataset, the waveforms of sad and fear emotions showed some similarities, such as the presence of low-frequency components in both. However, there were also noticeable differences, with the fear waveform showing a more rapid rise and

fall in amplitude compared to the more gradual changes observed in the sad waveform. Similarly, in the RAVDESS dataset, the waveforms of sad and neutral emotions also showed a similar pattern of low-frequency components, but there were also significant differences in their spectral content. Specifically, the neutral waveform had a more uniform spectral distribution across frequencies, while the sad waveform had more pronounced energy in the lower frequency range. Finally, in the SAVEE dataset, the waveforms of angry and disgust emotions showed some similarities in the way that both emotions have high amplitude and sharp transitions.

Table 4.22: Amplitude Comparison Between Closely Related Emotions in the Dataset.

CREMA-D		
RAVDESS		
SAVEE		

4.2.3 Pararel Transformer Encoder with CNN Architecture

The Pararel Transformer Encoder with CNN model architecture in this study was based on a combination of the Transformer Encoder block and CNN, which is particularly suited for processing sequential data such as speech signals. The CNN filter block provides expressive feature representations at a low computational cost, while the Transformer Encoder block provides an additional layer of sophistication to the architecture. This block enables the model to learn and predict the frequency distributions of various emotions by leveraging the global structure of the MFCC plots associated with each emotion. By considering the broader context and interdependencies within the MFCC plot, the model gains a holistic understanding of the temporal patterns and relationships, contributing to more accurate emotion classification. The hybrid nature of this architecture, with its combined Transformer Encoder and CNN components, empowers the model to effectively process and interpret the sequential nature of speech signals, capturing both local and global dependencies. This comprehensive approach ensures that the model can leverage both the fine-grained details and the broader context of the input data, ultimately leading to improved emotion classification performance.

Table 3.1 provides a detailed overview of the architecture of the model used in this study to give a comprehensive understanding of the model architecture, including the number of filters,

kernel size, pooling size, and activation functions used in each layer which highlights the configuration made to optimize the model performance. Similar to the other deep learning models evaluated in this study, the Pararel Transformer Encoder with CNN Architecture was trained on the CREMA-D, RAVDESS, and SAVEE datasets. The training process followed a standard procedure, involving the feeding of the model with the training dataset and iteratively optimizing the model parameters through the utilization of the backpropagation algorithm to minimize the loss function.

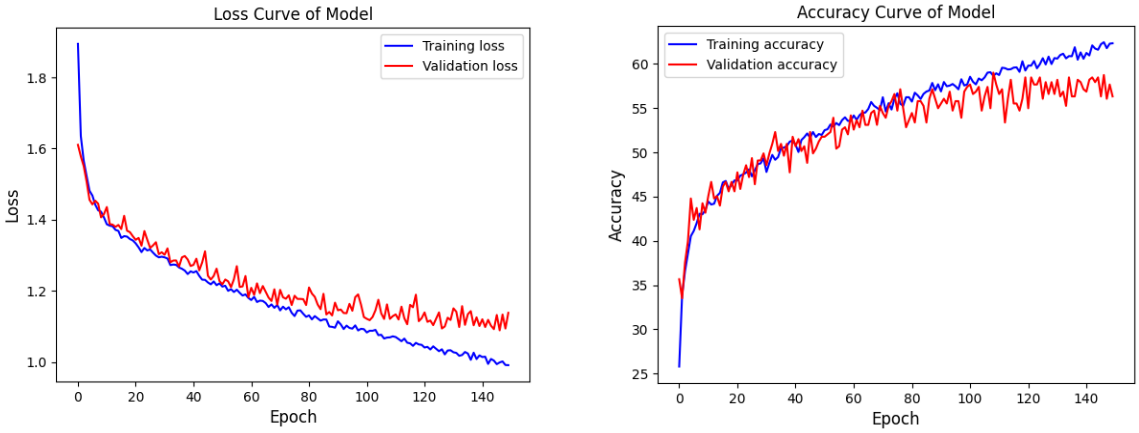


Figure 4.9: CREMA-D Accuracy and Loss Curve During Training Process.

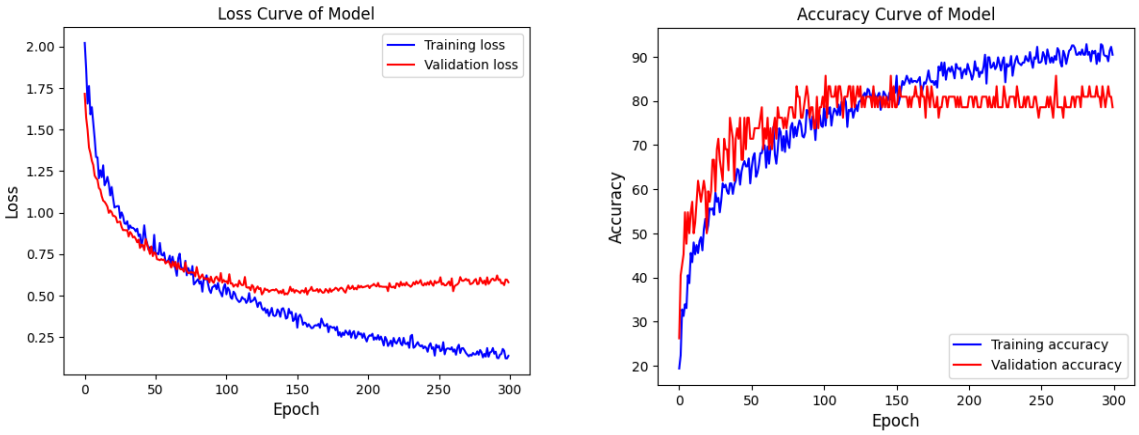


Figure 4.10: SAVEE Accuracy and Loss Curve During Training Process.

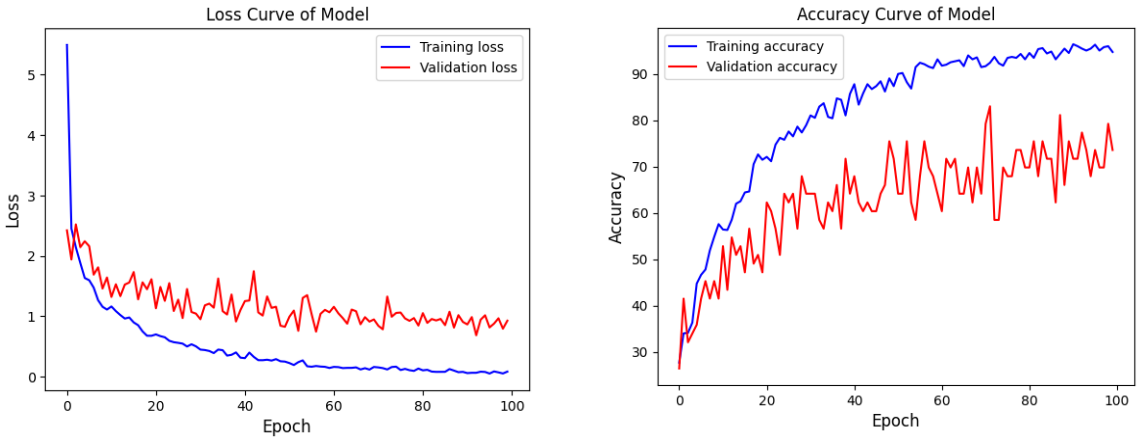


Figure 4.11: RAVDESS Accuracy and Loss Curve During Training Process.

During the training process, continuous monitoring of the loss and accuracy curves was conducted to ensure that the model was effectively optimizing the loss function and achieving high accuracy on the validation set. An optimal number of epochs for training was determined by analyzing these curves, preventing overfitting, and ensuring that the model attained the highest possible accuracy. By carefully considering these factors, the model strives to maximize its predictive capabilities and enhance the accuracy of emotion recognition in speech signals.

The loss and accuracy curves for the model trained on the CREMA-D, RAVDESS, and SAVEE datasets are presented in Figure 4.9, Figure 4.10, and Figure 4.11, respectively. The plots demonstrate that the model performance varies significantly across the different datasets. The loss curves showed a decreasing trend across all three datasets, indicating effective learning. It is evident from the plot that the number of epochs required for the model to converge and start overfitting the training data varied across these datasets. The model trained on the CREMA-D dataset achieved the highest accuracy of 59%, stopping at epoch 145. Conversely, the model trained on the RAVDESS dataset achieved a peak accuracy of 79% and converged faster, stopping at epoch 100. For the SAVEE dataset, a lower learning rate was set to avoid overshooting the optimal solution and to achieve a more stable training process, which resulted in a longer training time, with the model stopping at epoch 300 and achieving a maximum accuracy of 83%.

In addition to analyzing the loss and accuracy curves, confusion matrices will be used to evaluate the performance of this model on the CREMA-D, RAVDESS, and SAVEE datasets. The confusion matrices will provide detailed information about the model ability to classify emotions and identify areas for improvement. Specifically, the confusion matrices will be used to evaluate the true positive, true negative, false positive, and false negative predictions made by the model on each emotion category. These results will be presented in tables for each dataset, allowing for easy comparison between the models' performances. These tables include bolded values, which represent the highest true and false values for each emotion within the respective dataset. By highlighting these values, the tables accompanying each dataset offer a clear and straightforward means of comparing the models' performances. Thus, the utilization of confusion matrices in evaluating the model adds a deeper understanding of its strengths and weaknesses in accurately classifying specific emotions.

Table 4.23: T. Encoder and CNN CREMA-D Confusion Matrix.

Class	Falsely Predicted Emotion						Total Truth	Total False
	Angry	Fear	Disgust	Happy	Neutral	Sad		
Angry	41	3	8	10	2	0	**41**	23
Fear	2	40	6	2	6	8	40	24
Disgust	4	3	34	3	10	10	34	30
Happy	4	9	9	31	9	2	31	**33**
Neutral	1	3	6	1	39	4	39	15
Sad	1	4	9	1	13	35	35	28

Table 4.24: T. Encoder and CNN RAVDESS Confusion Matrix.

Class	Falsely Predicted Emotion						Total Truth	Total False
	Angry	Fear	Disgust	Happy	Neutral	Sad		
Angry	8	0	1	0	0	0	**8**	1
Fear	0	8	0	0	0	1	**8**	1
Disgust	0	1	8	1	0	0	**8**	2

Class	Falsely Predicted Emotion						Total Truth	Total False
	Angry	Fear	Disgust	Happy	Neutral	Sad		
Happy	0	2	0	8	0	0	**8**	2
Neutral	0	0	0	0	4	1	4	1
Sad	0	1	0	0	3	6	6	**4**

Table 4.25: T. Encoder and CNN SAVEE Confusion Matrix.

Class	Falsely Predicted Emotion						Total Truth	Total False
	Angry	Fear	Disgust	Happy	Neutral	Sad		
Angry	3	1	0	0	1	1	3	**3**
Fear	0	4	0	0	0	2	4	2
Disgust	0	0	6	0	0	0	6	0
Happy	1	0	0	5	0	0	5	1
Neutral	0	0	0	0	12	0	**12**	0
Sad	0	0	0	0	1	5	5	1

Table 4.23, Table 4.24, and Table 4.25 showed the confusion matrices for the CREMA-D, RAVDESS, and SAVEE datasets, respectively. The confusion matrices served as one of the metrics of evaluation analyzed to gain a better understanding of the performance of the Transformer Encoder block combined with CNN model architecture. The results showed that the model had different levels of confusion across the different datasets. For the CREMA-D dataset, the model shows high levels of confusion between sad and neutral, and the happy emotion had the lowest true positive rate, indicating that the model inaccurately classified happy samples to other emotion classes. Similarly, for the RAVDESS dataset, the model also showed a high level of confusion between the sad and neutral emotion classes. In contrast, for the SAVEE dataset, the model had high levels of confusion between fear and sad emotions. Based on these results, the performance of the model displays notable variations depending on the dataset on which it was evaluated. The varying levels of confusion between emotion classes in the confusion matrices highlight the model strengths and weaknesses for each tested dataset.

Moreover, in addition to the confusion matrix, precision, recall, and F1-score are employed as evaluation metrics to assess the model ability to accurately classify emotions in the dataset under consideration. The results obtained from these metrics complement the insights gained from the confusion matrix, providing a comprehensive evaluation of the model performance in emotion classification. Table 4.26 presents a detailed analysis of the model accuracy in emotion classification by providing the precision, recall, and F1-score for each emotion category in the tested datasets. This table presents the bolded values as the highest recorded values for each emotion category across the respective datasets, emphasizing the model standout performance in those instances.

Table 4.26: Comparison of T.Encoder and CNN Precision, Recall, and F1 Score for Emotion Classification.

Emotion	Precision			Recall			F1-Score		
	Crema	RAVDESS	SAVEE	Crema	RAVDESS	SAVEE	Crema	RAVDESS	SAVEE
Angry	**0.77**	**1.00**	0.75	0.64	**0.89**	0.50	**0.70**	**0.94**	0.60
Fear	0.65	0.67	0.80	0.62	**0.89**	0.67	0.63	0.76	0.73
Disgust	0.47	0.89	**1.00**	0.53	0.80	**1.00**	0.50	0.84	1.00

Emotion	Precision			Recall			F1-Score		
	Crema	RAVDESS	SAVEE	Crema	RAVDESS	SAVEE	Crema	RAVDESS	SAVEE
Happy	0.65	0.89	**1.00**	0.48	0.80	0.83	0.55	0.84	0.91
Neutral	0.49	0.57	0.86	**0.72**	0.80	**1.00**	0.59	0.67	**0.92**
Sad	0.59	0.75	0.62	0.56	0.60	0.83	0.57	0.67	0.71

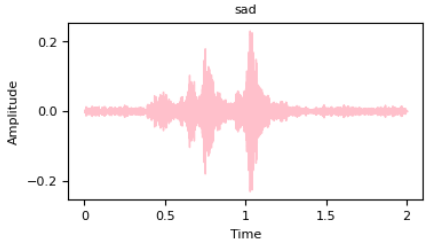
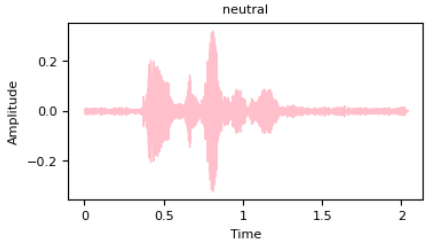
The presented F1 scores in Table 4.26 for emotion classification on the CREMA-D, RAVDESS, and SAVEE datasets reveal varying levels of accuracy for the Transformer Encoder combined with the CNN model. The results show that the model performed relatively well in the RAVDESS and SAVEE datasets, with an average accuracy of 79% and 83%, respectively. Moreover, the SAVEE dataset achieved the highest average accuracy with a perfect f1 score for the disgust emotion, indicating the model exceptional performance in recognizing this emotion. However, the model performance was relatively poor in the CREMA-D dataset, with an average accuracy of 59%.

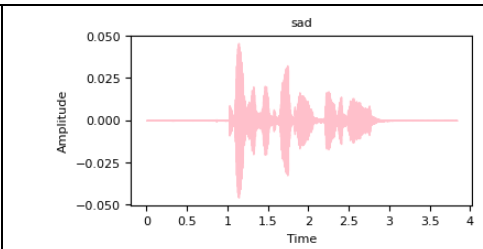
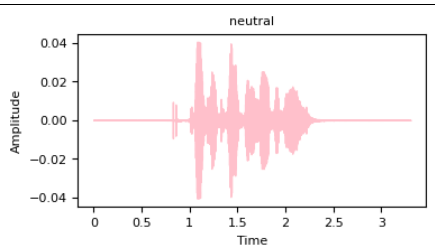
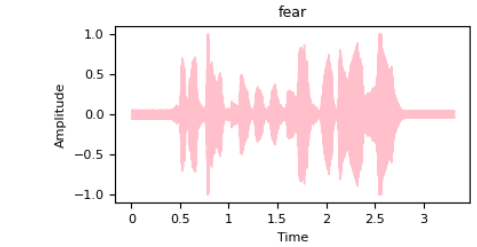
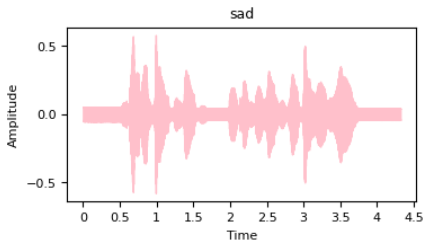
In terms of individual emotion classification, the angry emotion had the highest f1 score in both the CREMA-D and RAVDESS datasets, with f1 scores of 0.70 and 0.94, respectively. On the other hand, the sad emotion had the lowest f1 score for all tested datasets, with scores of 0.57, 0.67, and 0.71 for CREMA-D, RAVDESS, and SAVEE, respectively, indicating the difficulty in accurately recognizing this emotion for this model.

Table 4.27 presents the closely related emotions identified for each dataset used in this study. The plots in this table provide insight into the waveform structure and amplitude patterns for these emotions, highlighting their similarities and differences. In the CREMA-D and RAVDESS datasets, the closely related emotions are sad and neutral, which implies that audio clips in both datasets exhibit similar characteristics in terms of their amplitude profiles. Specifically, both emotions show relatively low amplitude levels, with sad emotion displaying a slightly more pronounced pattern of lower amplitude values. This similarity in amplitude patterns can be seen in the plots, which have overlapping areas that may have contributed to the model confusions in differentiating between the two emotions. This confusion is also reflected in the relatively low F1 scores for sad and neutral emotions compared to the other emotions in both datasets.

On the other hand, for the SAVEE dataset, the closely related emotions are fear and sadness. The amplitude patterns for these emotions show similar variations, with both emotions having a high-frequency content and sudden bursts of energy followed by a quick decrease in intensity. This similarity in amplitude patterns could be attributed to the similar physiological responses associated with these emotions. The model may have struggled to differentiate between these emotions due to their similar amplitude patterns, which is evident from their relatively low F1 scores.

Table 4.27: Amplitude Comparison Between Closely Related Emotions in the Dataset.

CREMA-D		

RAVDESS		
SAVEE		

4.3 Method Comparisons

In this section, the results of various machine learning models on the experiment of human emotion recognition from audio data will be compared. The models being compared include SVM, LeNet CNN, CRNN, and the Pararel Transformer Encoder with CNN Architecture. The models were evaluated on three different datasets: CREMA-D, RAVDESS, and SAVEE. For each dataset, the data was split into training, validation, and testing sets, with a ratio of 90:5:5, respectively. The data preprocessing steps varied for each model and were chosen based on the optimal data loading parameters that were tested. However, all models used the same feature extraction steps, which involved MFCC and normalization. The results of each experiment are displayed in Table 4.28. The bolded values in the table indicate the highest accuracy achieved by the models for each specific dataset, as well as the overall highest accuracy value attained across all datasets.

Table 4.28: Model Comparison on Each Trial.

Model	Datasets	Accuracy	Average
SVM	CREMA-D	54%	68%
	RAVDESS	72%	
	SAVEE	**78%**	
LeNet	CREMA-D	52%	61%
	RAVDESS	64%	
	SAVEE	**67%**	
CRNN	CREMA-D	55%	65%
	RAVDESS	70%	
	SAVEE	**71%**	
Transformer Encoder and CNN	CREMA-D	59%	**74%**
	RAVDESS	79%	
	SAVEE	**83%**	

The accuracy of the models varies significantly depending on the dataset being used. For

instance, the Transformer Encoder and CNN model showed the highest accuracy on the SAVEE dataset, with an accuracy of 83%. However, on the CREMA-D dataset, the same model achieved an accuracy of only 59%. This same pattern is observed in all the other models examined in this study, which indicates that the performance of the models is highly dependent on the characteristics of the dataset being used.

Among the models being compared, the Pararel Transformer Encoder with CNN Architecture model showed the best overall performance, with an average accuracy of 73.66% across all datasets. This is likely due to the fact that this model combines the strengths of both the Transformer Encoder Block and CNN, which are both powerful deep-learning techniques for processing sequential data. The Transformer Encoder Block is particularly effective in processing long sequences of data and has been shown to be highly effective in natural language processing tasks. Meanwhile, CNNs are excellent at capturing local patterns and relationships in the data, making them ideal for analyzing spectrogram images generated from audio signals.

The SVM model also performed relatively well, achieving an average accuracy of 68% across all datasets. SVM is a classic machine learning algorithm that is known to perform well in classification tasks, particularly with high-dimensional data. SVM models work by identifying an optimal hyperplane that separates the different classes in the data. In this study, the SVM model appears to have been effective in identifying patterns in the audio data that correspond to different emotions. In contrast, the CRNN model and LeNet CNN model achieved lower accuracy compared to the other models. Although CRNNs have been shown to be particularly useful in processing sequential data, in this study, the CRNN model showed an average accuracy of 65.33%, while LeNet CNN model achieved an average accuracy of 61%. These results indicate that the combination of both convolutional and recurrent neural network layers did not lead to significant improvements in accuracy for the task of emotion recognition in audio data.

Chapter V
Conclusions and Recommendations

5.1 Conclusion

The study addresses the problem of detecting human emotions from audio data using and draws the following conclusions:

- The process of detecting human emotions from audio data using the Pararel Transformer Encoder with CNN Architecture, which involves three primary steps. The first step involved preprocessing the loaded data using the optimal data loadings parameters such as durations, sample rates, and offsets, which could vary for each model. The second step involved extracting audio features using MFCC and normalization. The final step involved building the model that fits each dataset in the experiment. Through these preprocessing steps, the study found that the accuracy of the models varied significantly depending on the dataset being used. However, the combination of the Transformer Encoder and CNN model showed the best overall performance, with an average accuracy of 73.66% across all datasets.
- The Pararel Transformer Encoder with CNN Architecture used one block of Transformer Encoder consisting of self-attention layers, followed by a feedforward layer that consisted of two linear transformations with a ReLU activation in between. In addition, two identical blocks that resemble the CNN-based architecture of LeNet were utilized to extract features from the input data. The architecture of these blocks included a 3x3 convolution layer followed by an ELU activation function, which introduces non-linearity to the feature maps. Additionally, batch normalization was used to normalize the output and enhance the stability and efficiency of the model. The pooling layer was also employed to down-sample the feature maps by applying a pooling operation and reducing their size. These two blocks worked in parallel with the Transformer Encoder block and a final dense layer. The output tensors of these blocks were combined by the dense layer before passing through a linear layer with the number of class units. For the last layer, the SoftMax activation function was applied to convert the prediction scores into a probability distribution over the six different emotional states in the dataset.
- This study aimed to determine which classification methods were more accurate in detecting emotions through audio between SVM, LeNet-based CNN, CRNN, and the Pararel Transformer Encoder with CNN Architecture. The results indicated that the Pararel Transformer Encoder with CNN Architecture model was the most accurate, followed by the SVM model, while the CRNN and the LeNet CNN models achieved lower accuracies.

5.2 Recommendation

To further improve and advance the outcomes of this study, the following suggestions can be considered for further research:

- Explore data augmentation techniques to improve the accuracy of the models. Data augmentation can be used to artificially increase the size of the dataset, which can lead to better generalization and performance of the models. Techniques such as pitch shifting, time stretching, and noise addition can be applied to the audio data to create new and diverse samples.

- Combining the three tested datasets into a larger and more diverse dataset can improve the generalization and robustness of the models. The combination of datasets can be done by ensuring that the emotional labels are consistent across all datasets.
- Explore different feature extraction techniques for audio data. While this study used MFCC as the primary feature extraction technique, there are several other techniques that can be explored, such as Gammatone frequency cepstral coefficients (GFCC) and Mel-scaled spectrograms. Additionally, different normalization techniques and combinations of features can be investigated to determine the best approach for emotion recognition from audio data.

References

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). 2017 International Conference on Engineering and Technology. *Springer Open*(9), 1-6.

Baevski, A., Zhou, H., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. *arXiv*(3).

Brown, Mann, T. a., Ryder, B. a., Subbiah, N. a., Kaplan, M. a., Dhariwal, J. D., . . . Gretch. (2020). Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems* (pp. 1877-1901). Curran Associates, Inc.

Burkhardt, F., Paeschke, A., Rolfes, M., Sendlmeier, W. F., & Weiss, B. (2005). A database of German emotional speech. *INTERSPEECH 2005 - Eurospeech, 9th European Conference on Speech Communication and Technology*, (pp. 1517-1520). Lisbon.

Busso, C., Bulut, M., Lee, C., Kazemzadeh, A., Mower, E., Kim, S., . . . Narayanan, S. (2008). IEMOCAP: Interactive emotional dyadic motion capture database. *Language Resources and Evaluation, 42*, 335-359.

Cao, Cooper, H. a., Keutmann, D. G., Gur, M. K., Nenkova, R. C., Verma, A. a., & Ragini. (2014). CREMA-D: Crowd-Sourced Emotional Multimodal Actors Dataset. *IEEE Transactions on Affective Computing, 13*(5), 377-390.

Chamishka, S., Madhavi, I., Nawaratne, R., Alahakoon, D., Silva, D. D., Chilamkurti, N., & Nanayakkara, V. (2022). A voice-based real-time emotion detection technique using recurrent neural network empowered feature modelling. *SpringerLink*, *81*, 35173–35194.

Charlie. (2014, July 14). *It's No Disgrace To Use Your Face!* (The SAVI Singing Actor) Retrieved December 2022, 17 from https://www.savisingingactor.com/its-no-disgrace-to-use-your-face/

Chen, S., Wang, C., Chen, Z., Wu, Y., Liu, S., Chen, Z., . . . Wei, F. (2022). WavLM: Large-Scale Self-Supervised Pre-Training for Full Stack Speech Processing. *IEEE Journal of Selected Topics in Signal Processing*, *16*, 1505-1518.

Cherry, K. (2021, April 5). *The 6 Types of Basic Emotions and Their Effect on Human Behavior* . (verywellmind) Retrieved November 20, 2022 from https://www.verywellmind.com/an-overview-of-the-types-of-emotions-4163976

Citron, F., Gray, M. A., Critchley, H., Weekes, B., & Ferstl, E. C. (2014). Emotional valence and arousal affect reading in an interactive way: Neuroimaging evidence for an approach-withdrawal framework. *Neuropsychologia, 56,100*, 79–89.

Costello, K. (2019, January 21). *Gartner Survey Shows 37 Percent of Organizations have Implemented AI in Some Form*. (Gartner) Retrieved October 11, 2022 from https://www.gartner.com/en/newsroom/press-releases/2019-01-21-gartner-survey-shows-37-percent-of-organizations-have

Cowie, R. (2001). Emotion Recognition in Human-Computer Interaction. *IEEE, 18*(1), 32-80.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv*.

Duan, K.-B., Rajapakse, J. C., & Nguyen, M. N. (2007). One-Versus-One and One-Versus-All Multiclass SVM-RFE for Gene Selection in Cancer Classification. *Springer, Berlin, Heidelberg*.

Dubey, S. R., Singh, S. K., & Chaudhuri, B. B. (2021). A Comprehensive Survey and Performance Analysis of Activation Functions. *CoRR* .

Eyben, Scherer, F. a., Schuller, K. R., Sundberg, B. W., André, J. a., Busso, E. a., . . . P., K. (2016). The Geneva Minimalistic Acoustic Parameter Set (GeMAPS) for Voice Research and Affective Computing. *IEEE Transactions on Affective Computing, 7*(2), 190-202.

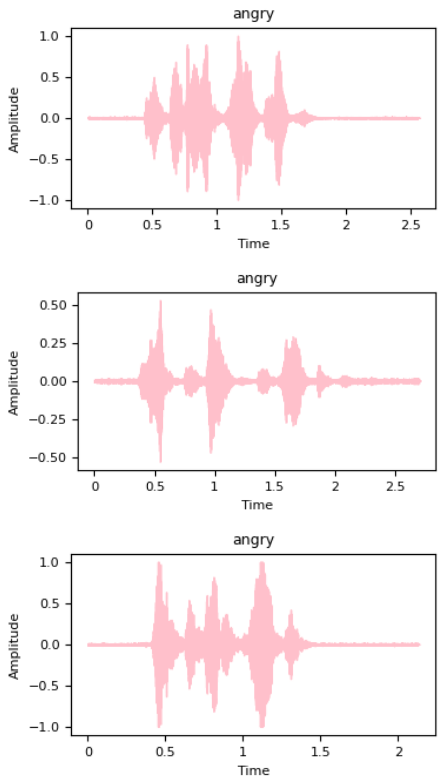
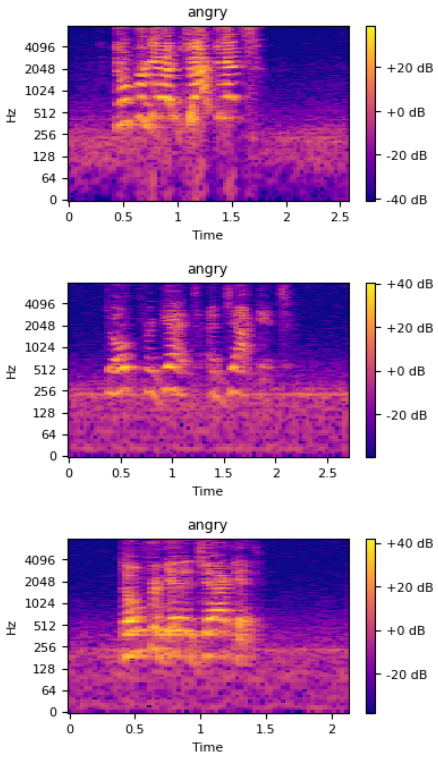
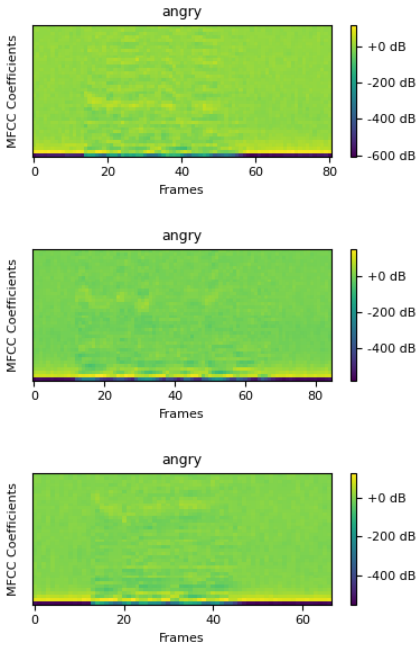
Gopal Krishna Patro, S., & Sahu, K. K. (2015). Normalization: A Preprocessing Stage. *arXiv*.
Graves, A., Mohamed, A. r., & Hinton, G. E. (2013). Speech Recognition with Deep Recurrent Neural Networks. *CoRR*.
Heredia, J., Cardinale, Y., Dongo, I., & Díaz-Amado, J. (2021). A Multi-modal Visual Emotion Recognition Method to Instantiate an Ontology. *16th International Conference on Software Technologies*, 453-464.
Hsu, W.-N., Bolte, B., Tsai, Y.-H. H., Lakhotia, K., Salakhutdinov, R., & Mohamed, A. (2021). HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 29*, 3451-3460.
Ihianle, I., Nwajana, A., Ebenuwa, S., Otuka, R., Owa, K., & Orisatoki, M. (2020). A Deep Learning Approach for Human Activities Recognition From Multimodal Sensing Devices. *IEEE Access*, 179028-179038.
Ioffel, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv*.
Ivar, Byron, R., & Clifford, N. (1996). *The media equation: How people treat computers, television, and new media like real people and places.* Cambridgeshire, England: Center for the Study of Language and Inf.
Jackson, P., & Haq, S. (2015, April 2). *Surrey Audio-Visual Expressed Emotion (SAVEE) Database.* From http://kahlan.eps.surrey.ac.uk/savee/
Jarymowicz, & Maria. (2012). Understanding Human Emotions. *Journal of Russian & East European Psychology, 50*(3), 9-25.
Karimi, Z. (2021, 10). Confusion Matrix. *Research Gate*. From Research Gate.
Keltner, Dacher, & Cordaro, D. T. (2017). Understanding Multimodal Emotional Expressions: Recent Advances in Basic Emotion Theory. In *The Science of Facial Expression* (pp. 57-76). New York: Social Cognition and Social Neuroscience.
KiKaBeN. (2021, December 13). *Transformer's Encoder-Decoder: Let's Understand The Model Architecture*. Retrieved December 18, 2022 from https://kikaben.com/transformers-encoder-decoder/
Kishore, K., & Satish, K. (2013). Emotion recognition in speech using MFCC and wavelet features. *2013 3rd IEEE International Advance Computing Conference (IACC)*, 842-847.
Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 84 - 90.
Latif, S., Rana, R., Younis, S., Qadir, J., & Epps, J. (2018). Transfer Learning for Improving Speech Emotion Classification Accuracy. *arXiv*(4).
Lech, M., Stolar, M., Best, C., & Bolia, R. (2020). Real-Time Speech Emotion Recognition Using a Pre-trained Image Classification Network: Effects of Bandwidth Reduction and Companding. *Frontiers in Computer Science, 2*, 14.
LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*(11), 2278-2324.
Li, Y., Tao, J., Chao, L., Bao, W., & Liu, Y. (2017). CHEAVD: a Chinese natural emotional audio–visual database. *Journal of Ambient Intelligence and Humanized Computing, 8*, 913-924.
Livingstone, S. R., & Russo, F. A. (2018). The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. *PLoS ONE*, 13.
Mahapatra, S. (2018, March 22). *Why Deep Learning over Traditional Machine Learning?* (Towards Data Science) Retrieved October 14, 2022 from

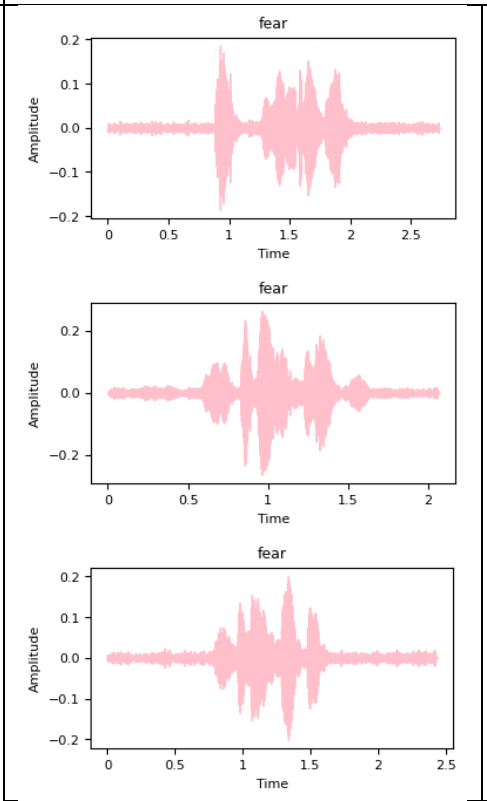
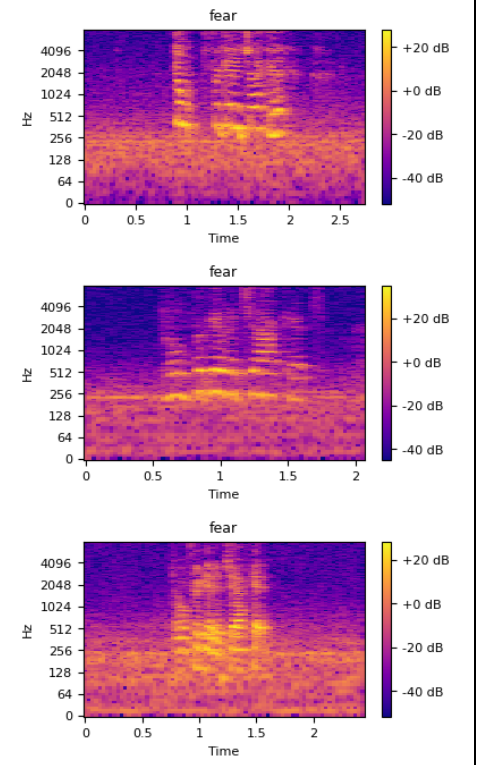
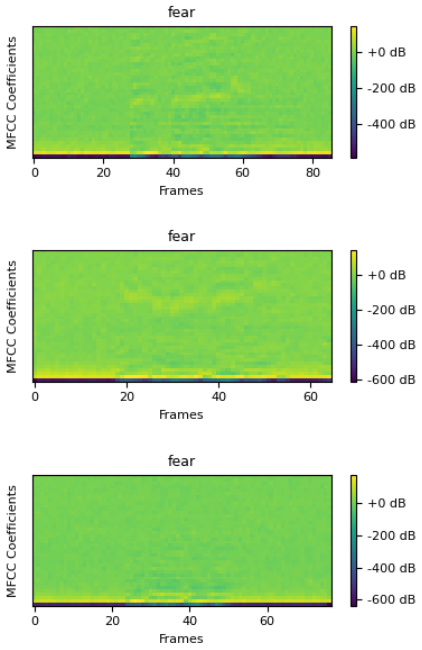
https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063
McFee, B., Raffel, C., Liang, D., Ellis, D., Mcvicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and Music Signal Analysis in Python. *Python in Science Conference*, 18-24.
Mysore, G. J. (2015). Can We Automatically Transform Speech Recorded on Common Consumer Devices in Real-World Environments into Professional Production Quality Speech? - A Dataset, Insights, and Challenges. *IEEE Signal Processing Letters.*
Nair, V., & Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. *Omnipress*, 807–814.
Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., . . . Chilamkurthy, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in neural information processing systems, 32*.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2023, 5 9). *Scikit-learn*. From StandardScaler: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler
Schneider, S., Baevski, A., Collobert, R., & Auli, M. (2019). wav2vec: Unsupervised Pre-training for Speech Recognition. *arXiv*(4).
Shi, B., Bai, X., & Yao, C. (2017). An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2298-2304.
Solomon, R. C. (2009, July 29). *emotion*. (Encyclopedia Britannica) Retrieved November 20, 2022 from https://www.britannica.com/science/emotion
Sonawane, A., Inamdar, M. U., & Bhangale, K. B. (2017). Sound based human emotion recognition using MFCC & multiple SVM. *IEEE*, 1-4.
Starner, T., & Pentland, A. (1995). Real-time American Sign Language recognition from video using hidden Markov models. *Proceedings of International Symposium on Computer Vision - ISCV*, 265-270.
Steidl, B. S., Batliner, A., Vinciarelli, A., Scherer, K., Ringeval, a., Chetouani, M., . . . Kim, S. (2013). The INTERSPEECH 2013 computational paralinguistics challenge: Social signals, conflict, emotion, autism. *Proceedings INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association.* Lyon, France.
Stowell, D., & Plumbey, M. D. (2013). An open dataset for research on audio field recording archives: freefield1010. *arXiv*.
StudyCorgi. (2022, June 25). Retrieved December 17, 2022 from https://studycorgi.com/the-characteristics-of-sound/
Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., . . . Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1-9.
Vapnik, V., & Cortes, C. (1995). Support-vector networks. *SpringerLink*, 273-297.
Vaswani, Shazeer, A. a., Parmar, N. a., Uszkoreit, N. a., Jones, J. a., Gomez, L. a., . . . Illia. (2017). Attention is All you Need. In *Advances in Neural Information Processing Systems.* Curran Associates, Inc.
Vryzas, N., Kotsakis, R., Liatsou, A., Dimoulas, C., & Kalliris, G. (2018). Speech Emotion Recognition for Performance Interaction. *Journal of the Audio Engineering Society. Audio Engineering Society, 66*(6), 457-467.
Wang, Wu, C. a., Qian, Y. a., Kumatani, Y. a., Liu, K. a., Wei, S. a., . . . Xuedong. (2021). UniSpeech: Unified Speech Representation Learning with Labeled and Unlabeled

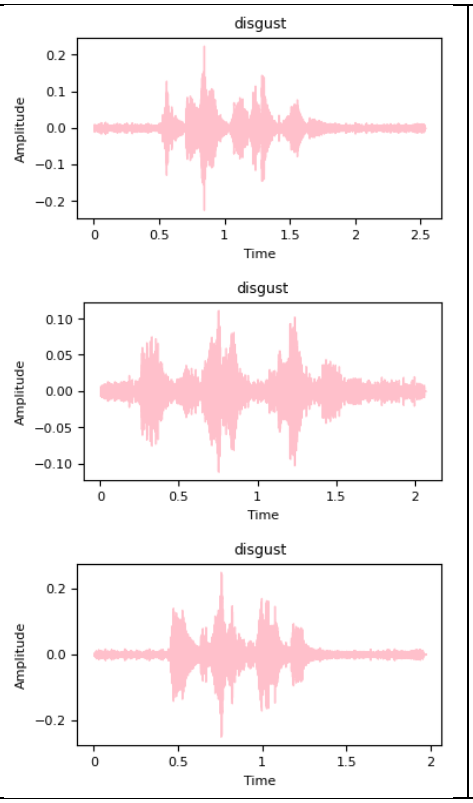
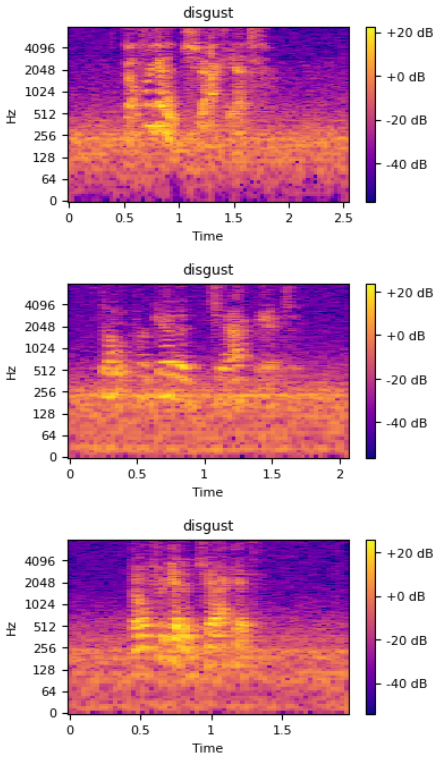
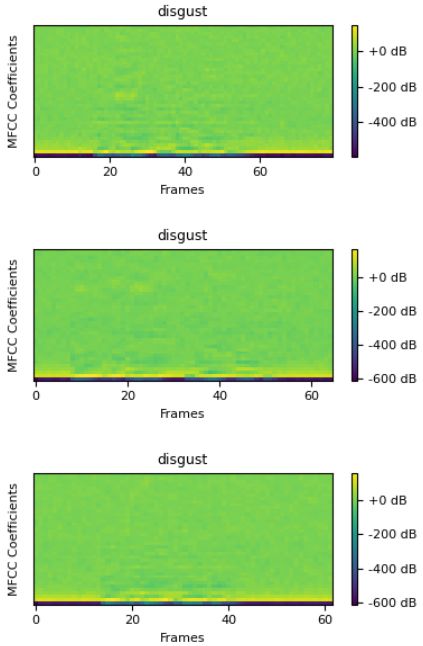
Data. *Proceedings of the 38th International Conference on Machine Learning, 139*, 10937-10947.
Younghak Shin, I. B. (2017). Comparison of hand-craft feature based SVM and CNN based deep learning framework for automatic polyp classification. *IEEE*, 3277-3280.
Yu, D., & Deng, L. (2015). *Automatic Speech Recognition A Deep Learning Approach.* London: Springer London.

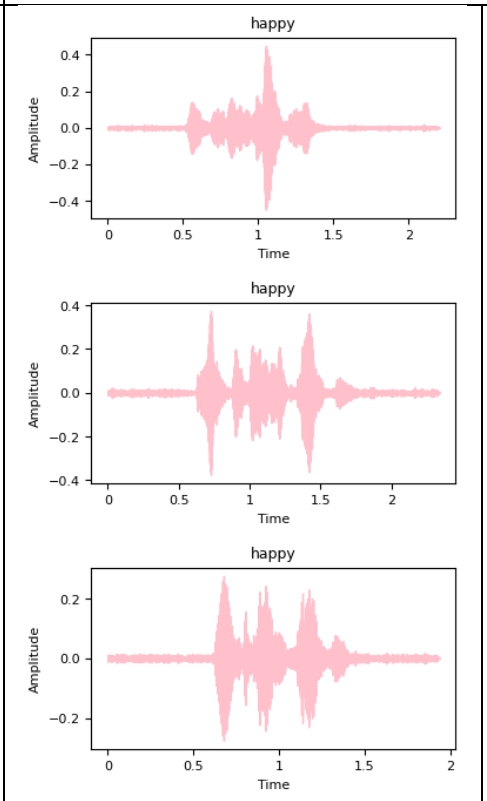
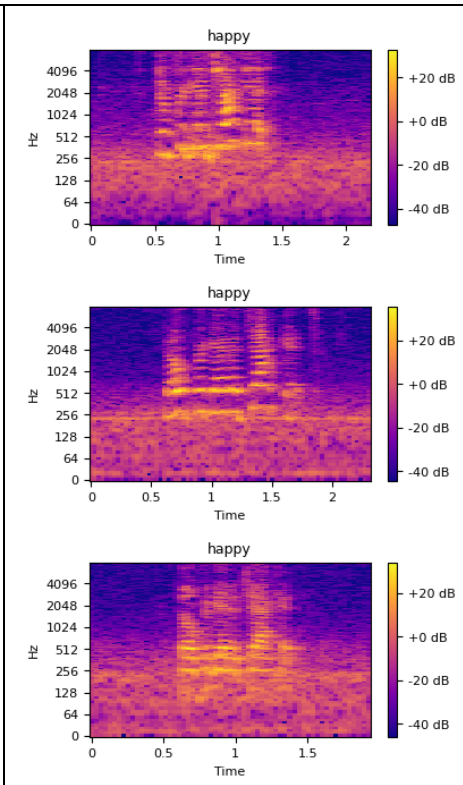
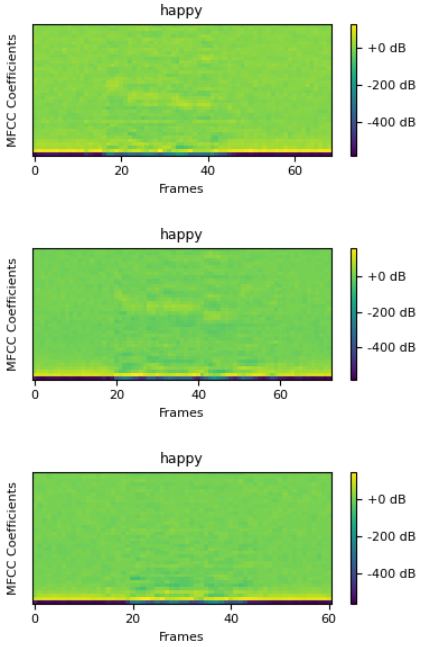
APPENDIX

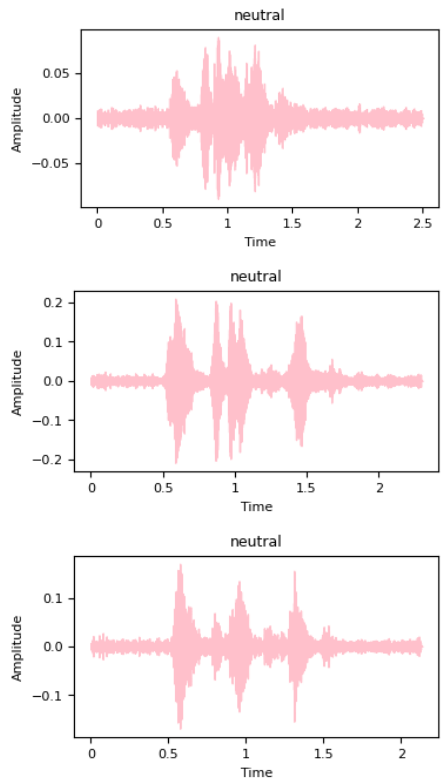
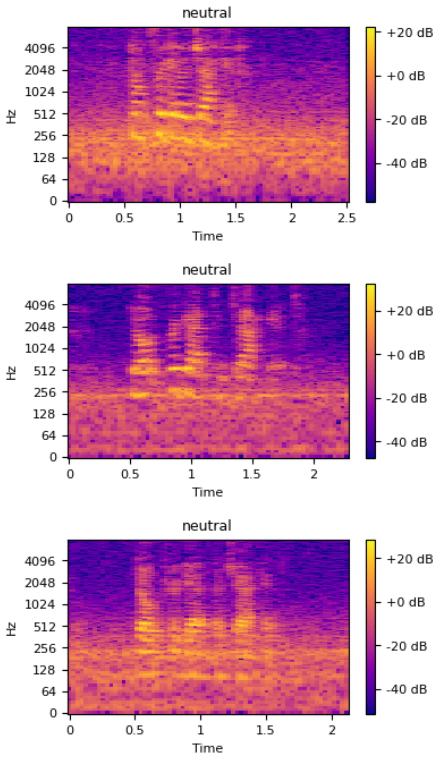
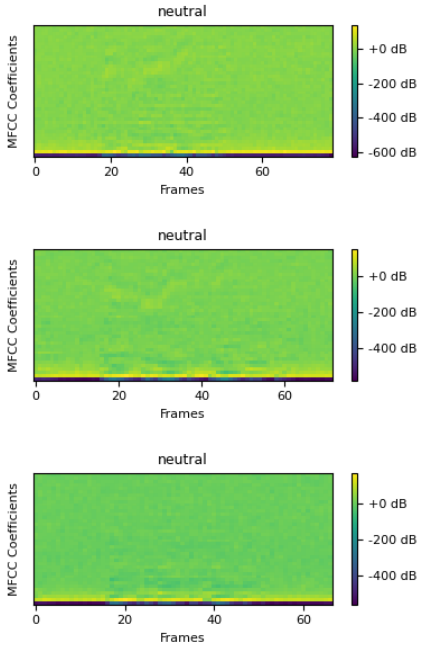
Appendix 1: CREMA-D Data Exploration Table.

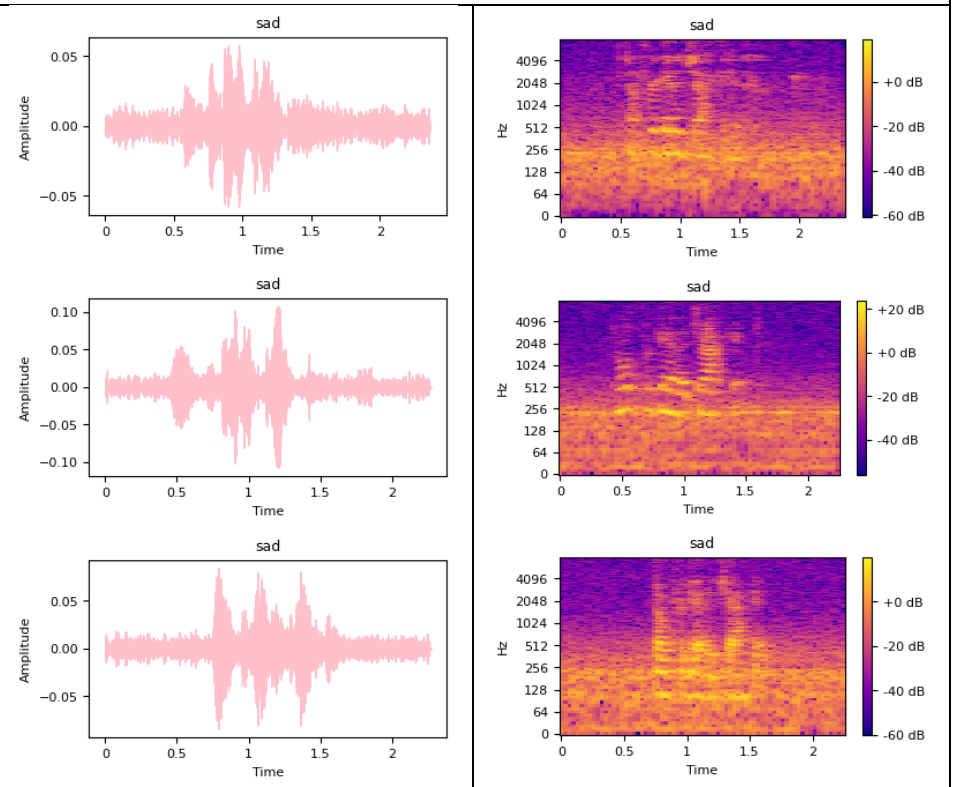
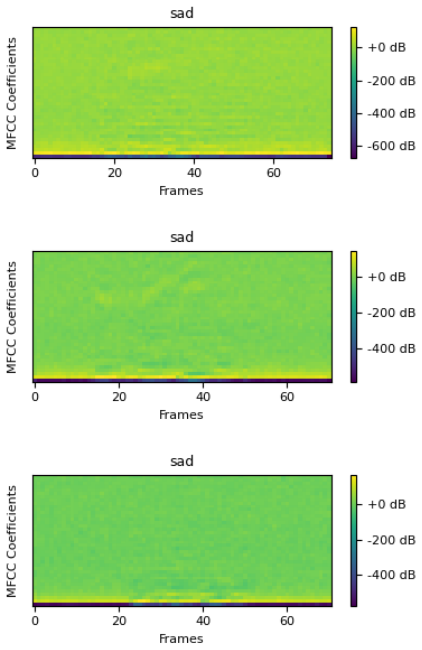
<table>
<tr><th>Emotion</th><th colspan="2">CREMA-D</th></tr>
<tr><td rowspan="3">Angry</td><td>Amplitude</td><td>Spectogram</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td colspan="2">MFCC</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Fear</td><td>Amplitude</td><td>Spectogram</td></tr>
</table>

Emotion	CREMA-D	
		
	MFCC	
		
Disgust	**Amplitude**	**Spectogram**

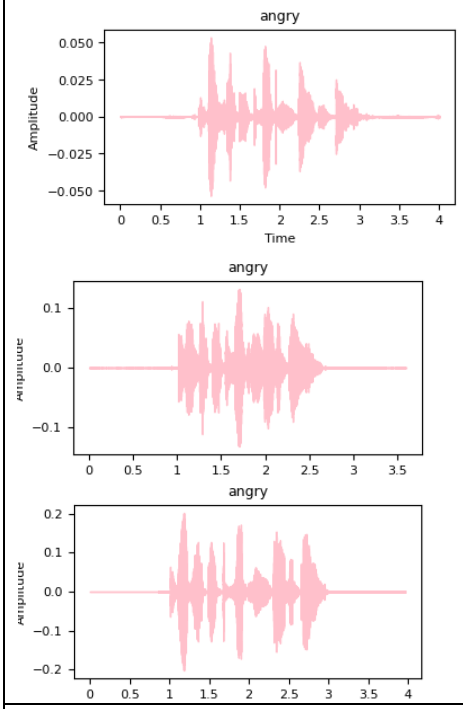
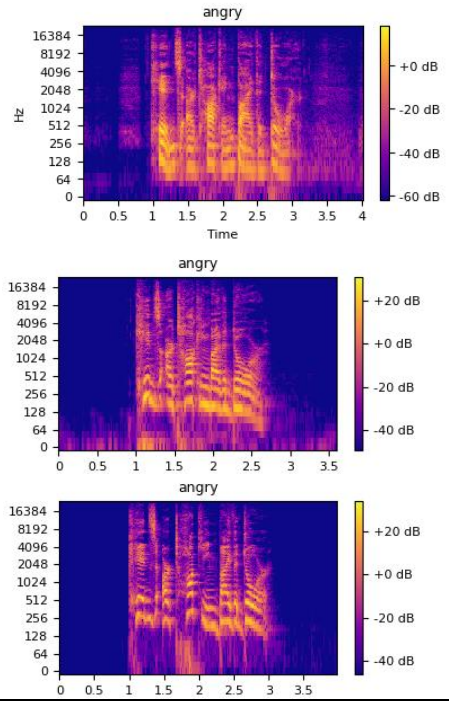
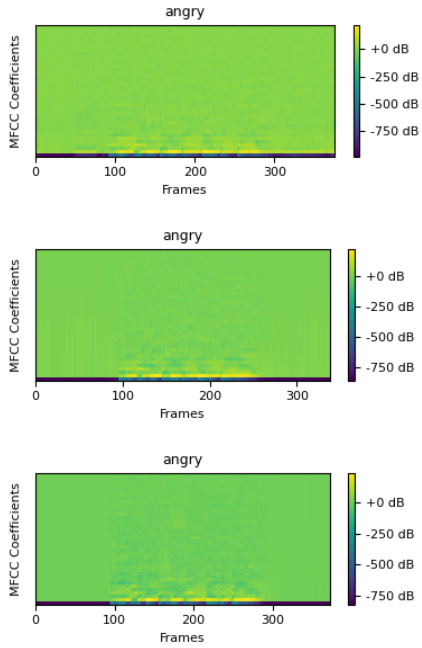
Emotion	CREMA-D	
		
	MFCC	
		
Happy	**Amplitude**	**Spectogram**

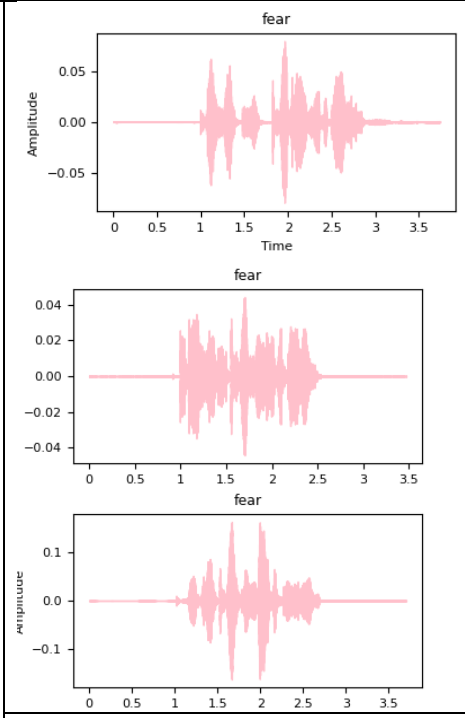
Emotion	CREMA-D	
		
	MFCC	
		
Neutral	**Amplitude**	**Spectogram**

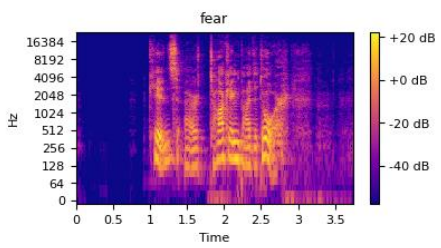
Emotion	CREMA-D	
		
	MFCC	
		
Sad	**Amplitude**	**Spectogram**

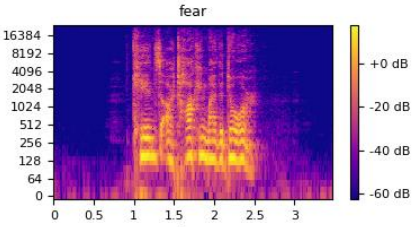
Emotion	CREMA-D
	
	MFCC
	

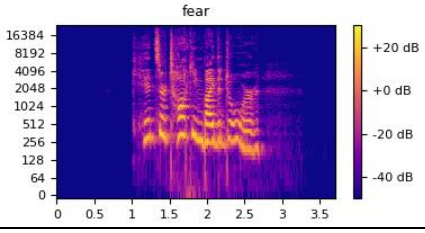
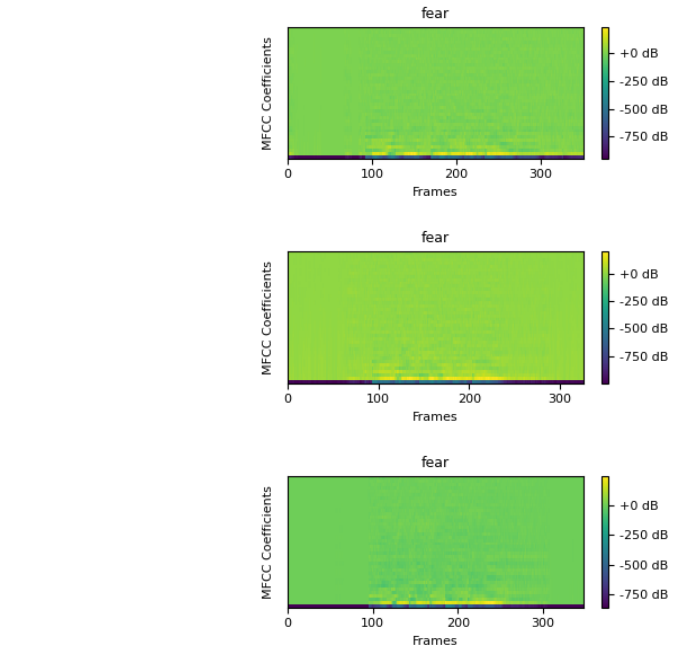
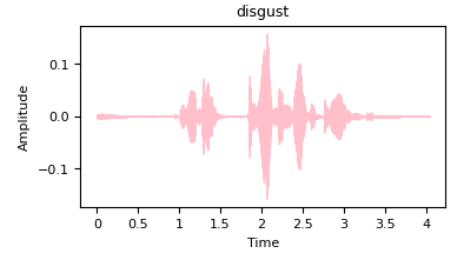
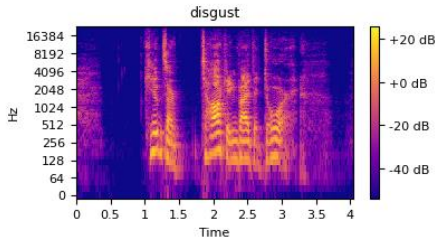
Appendix 2: RAVDESS Data Exploration Table.

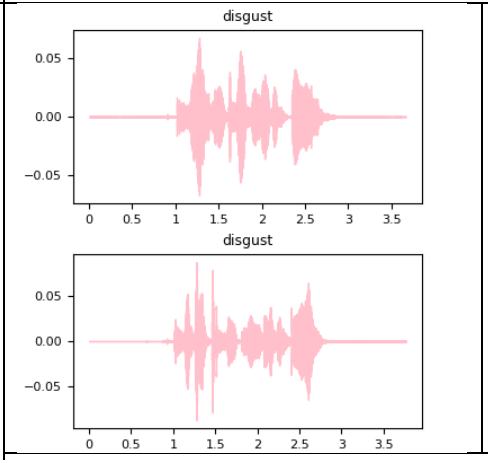
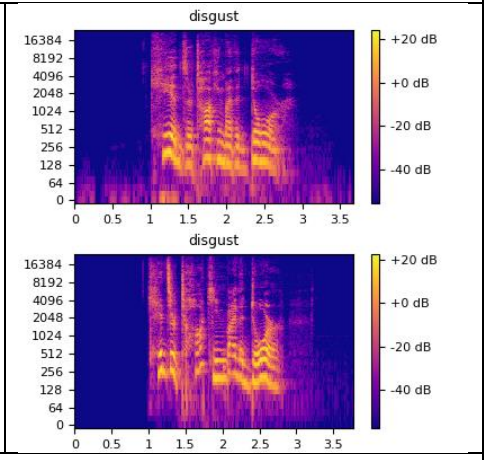
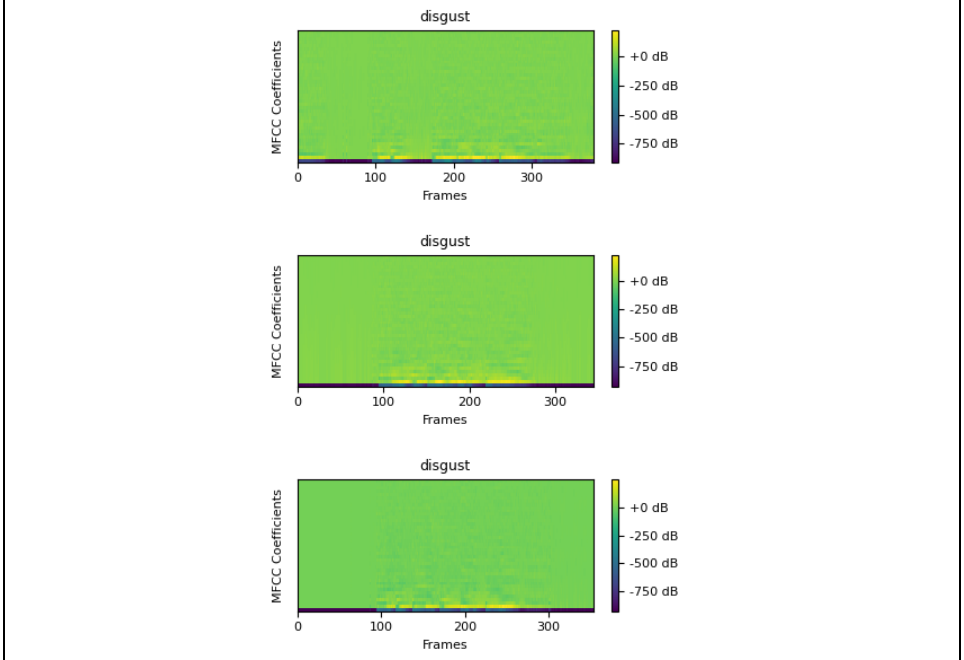
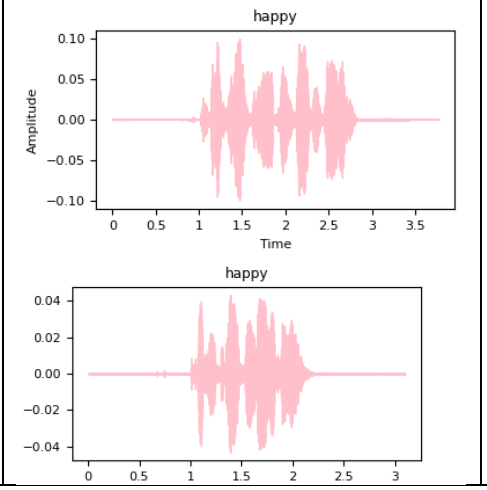
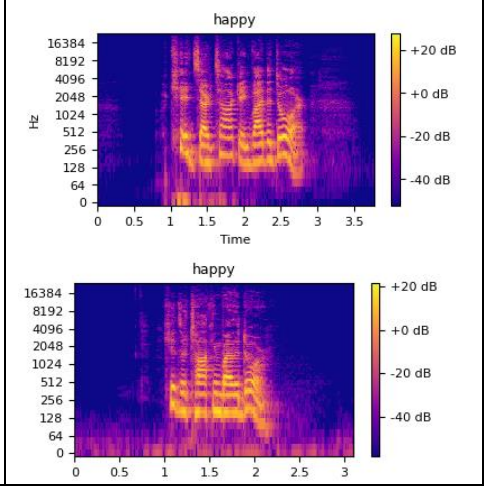
Emotion	RAVDESS	
	Amplitude	**Spectogram**
Angry		
	MFCC	
		
Fear	**Amplitude**	**Spectogram**

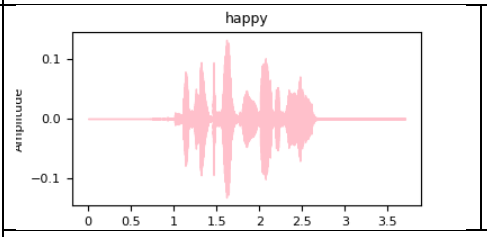
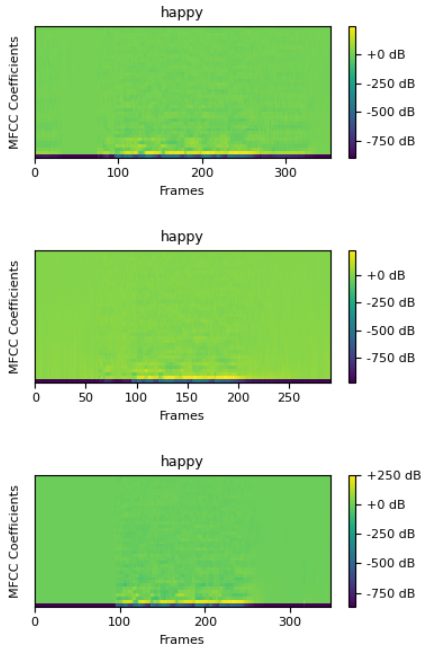
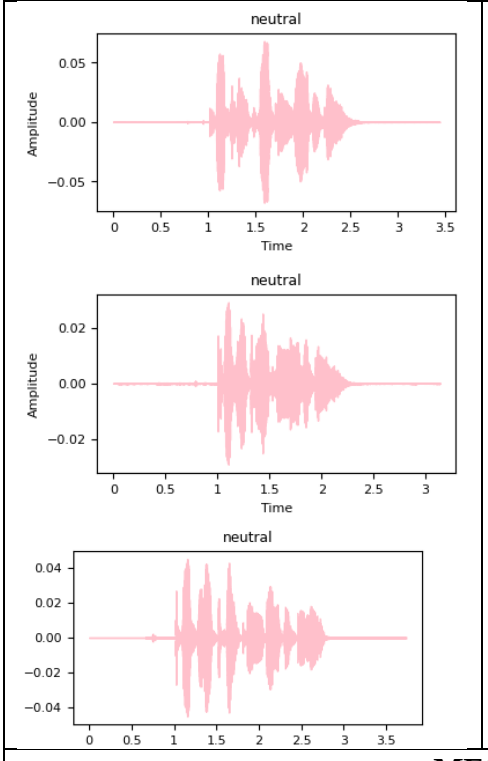
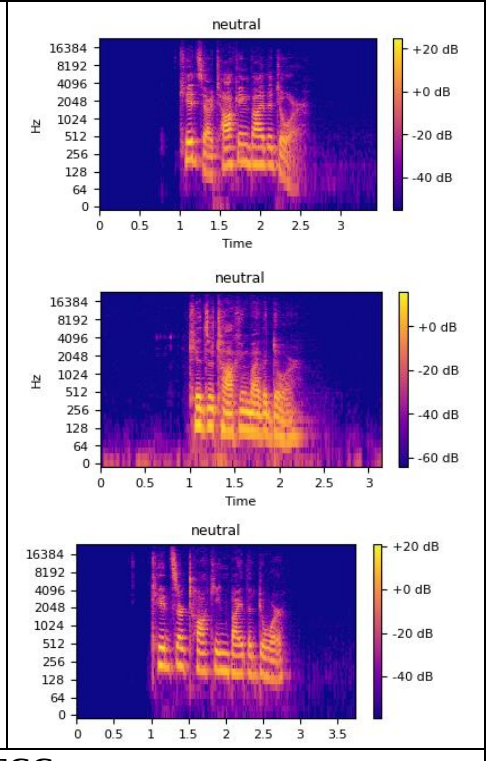
<table>
<tr><th>Emotion</th><th colspan="2">RAVDESS</th></tr>
<tr><td rowspan="2"></td><td> 

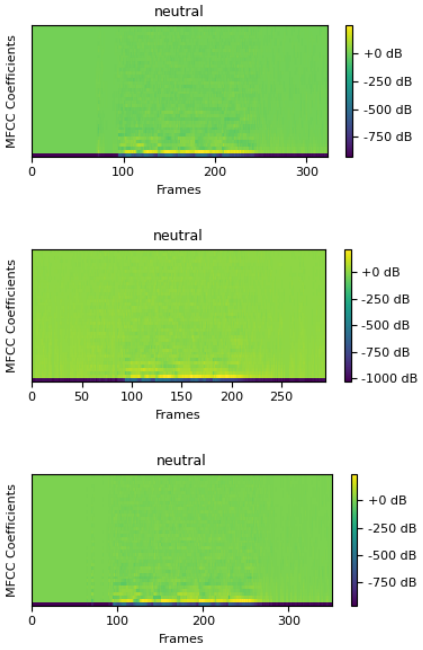
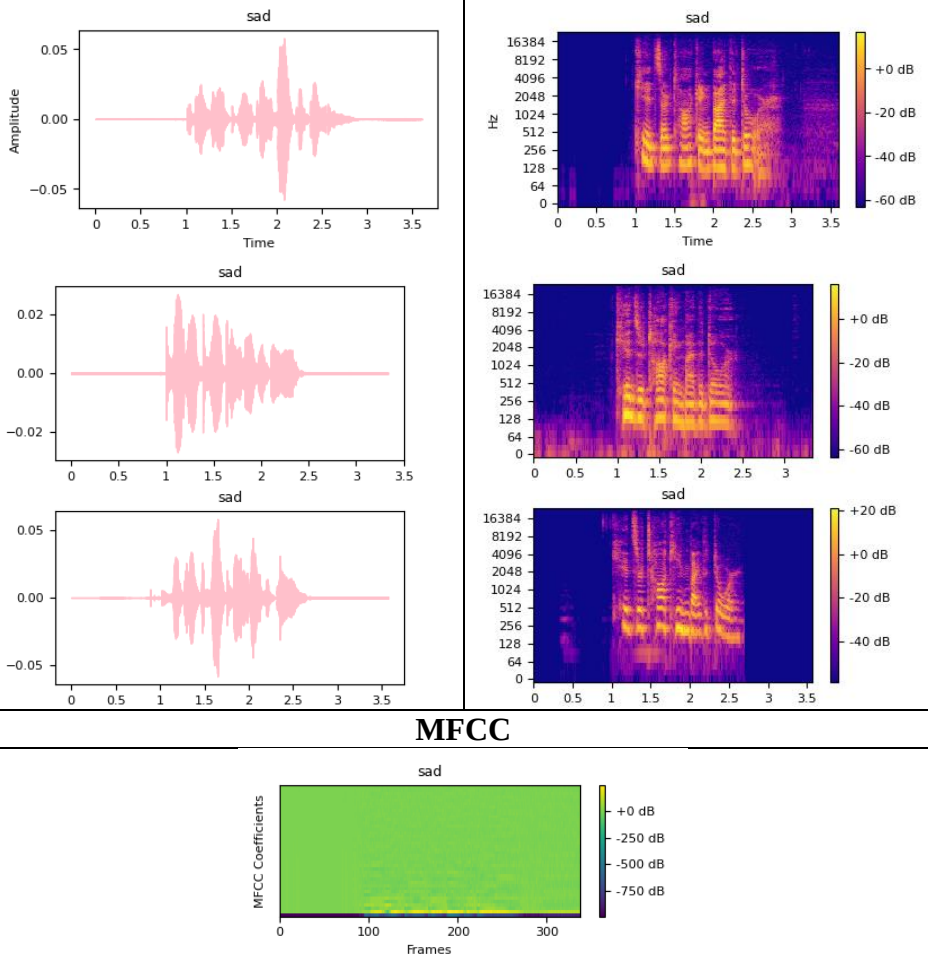
</td><td> 

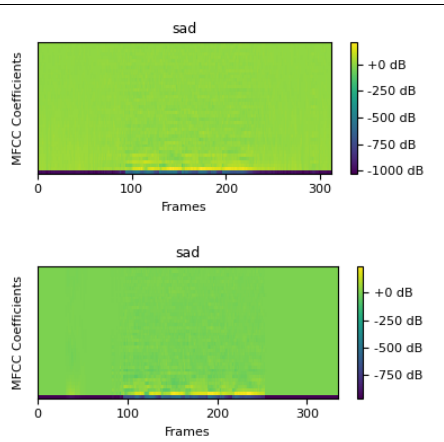



</td></tr>
<tr><td colspan="2">MFCC
 
</td></tr>
<tr><td>Disgust</td><td>Amplitude
 
</td><td>Spectogram
 
</td></tr>
</table>

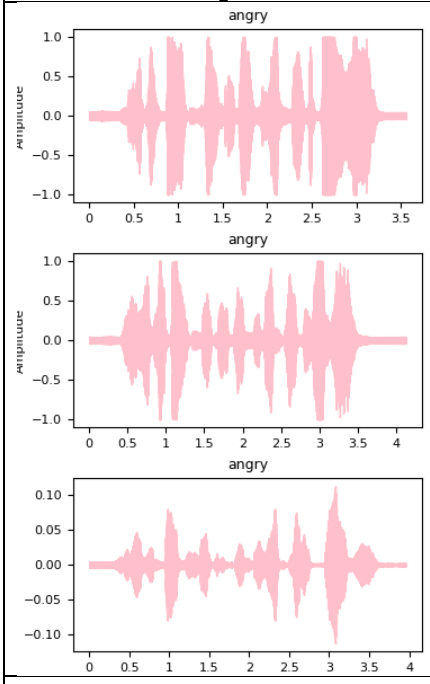
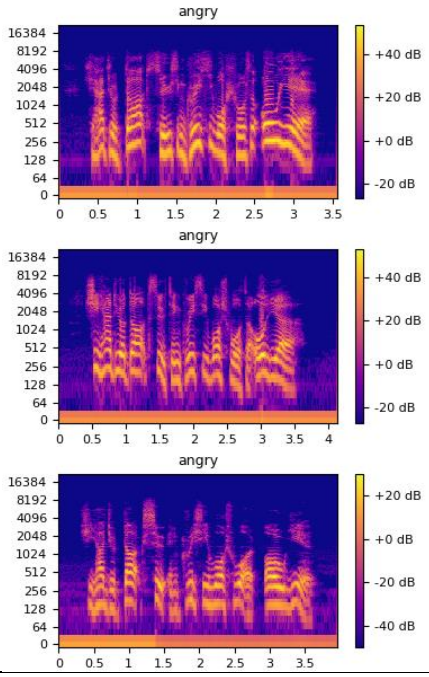
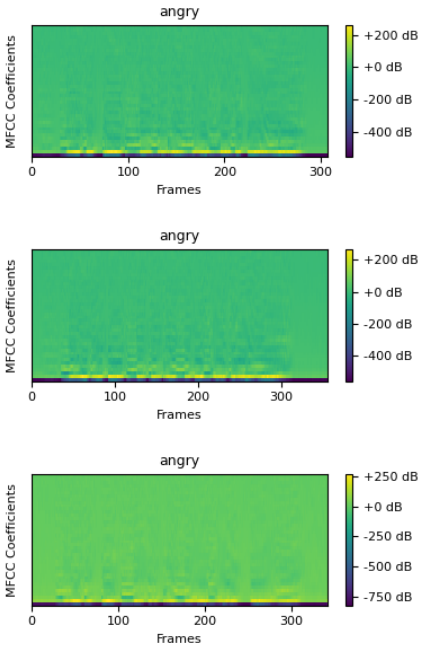
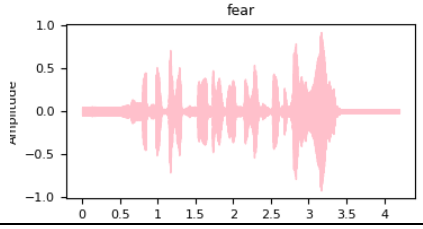
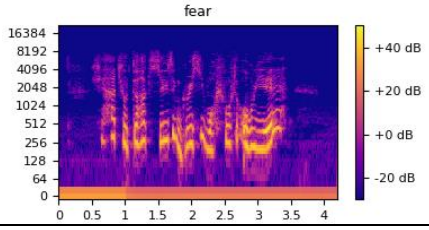
Emotion	RAVDESS	
		
	MFCC	
		
Happy	**Amplitude**	**Spectogram**
		

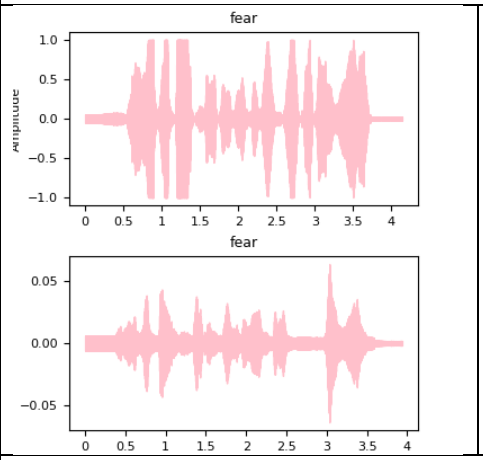
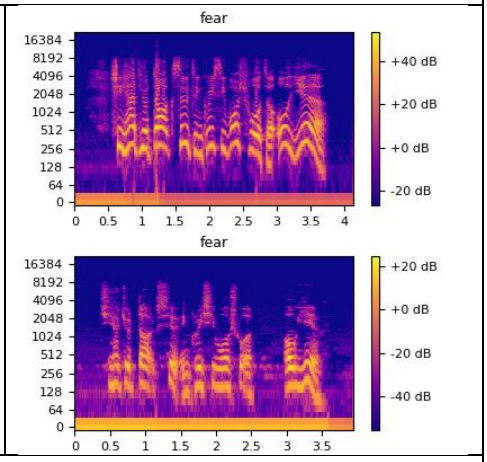
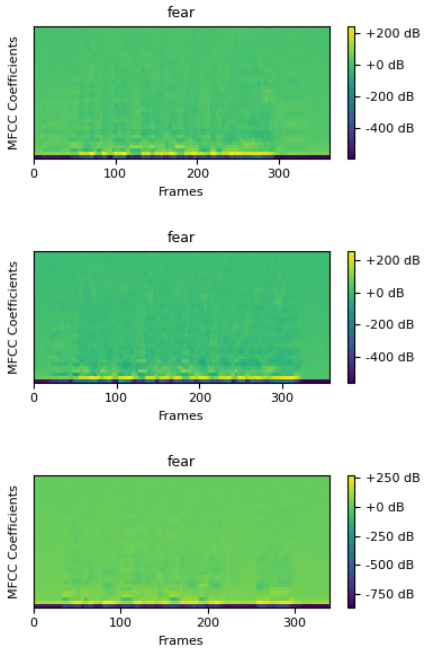
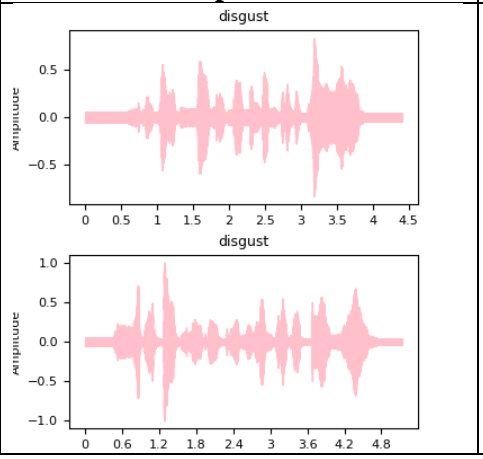
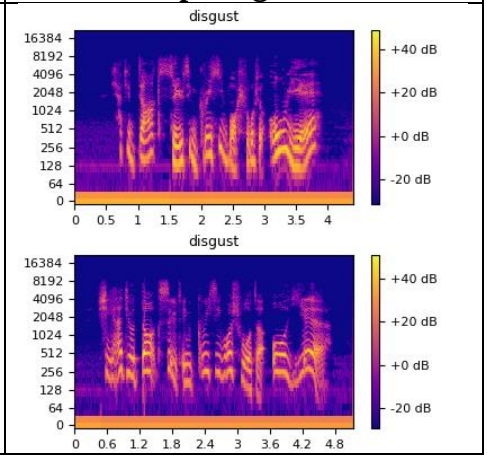
Emotion	RAVDESS	
		
	MFCC	
		
Neutral	Amplitude	Spectogram
		
	MFCC	

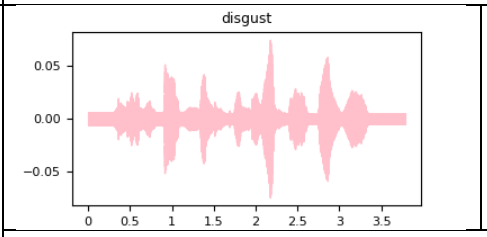
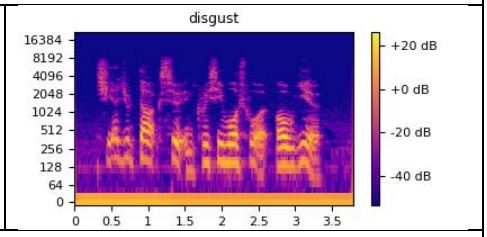
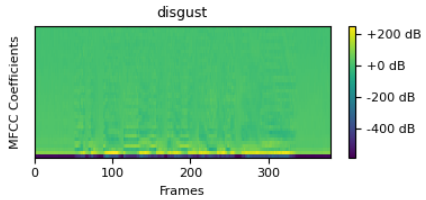
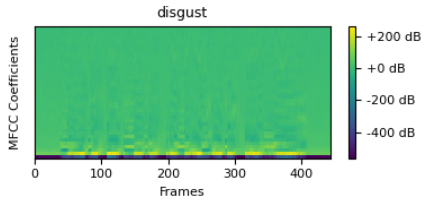
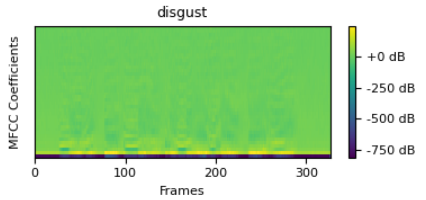
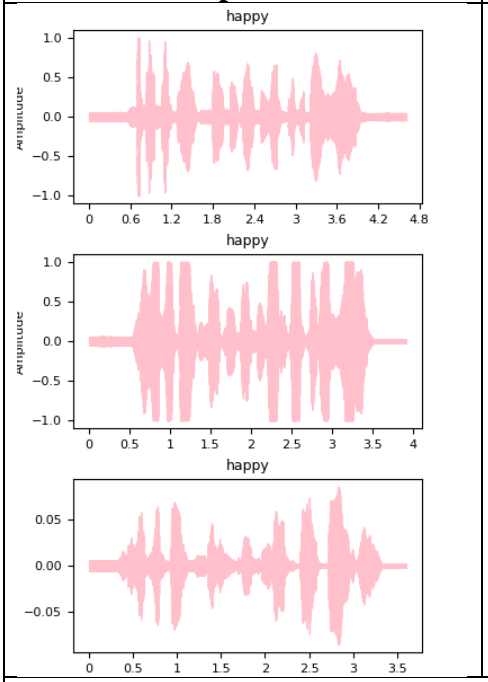
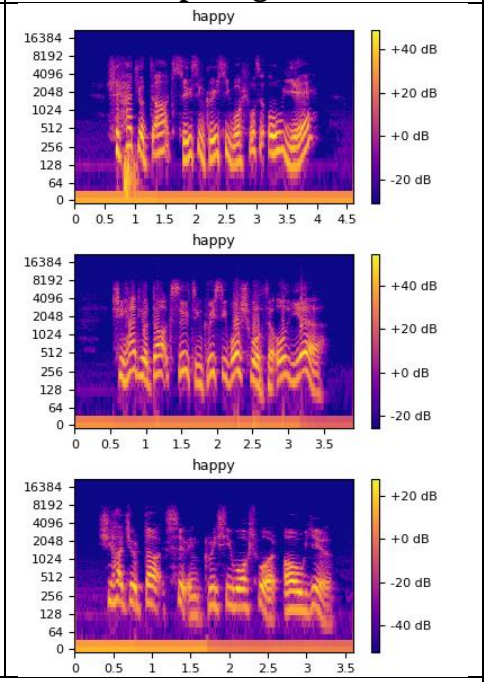
Emotion	RAVDESS
	
Sad	**Amplitude** / **Spectogram** / **MFCC** 

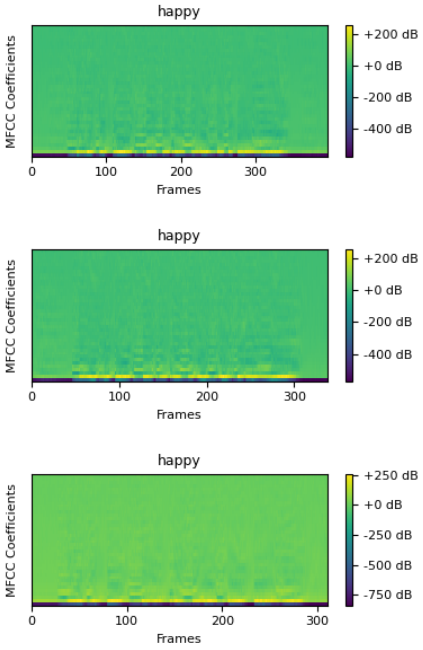
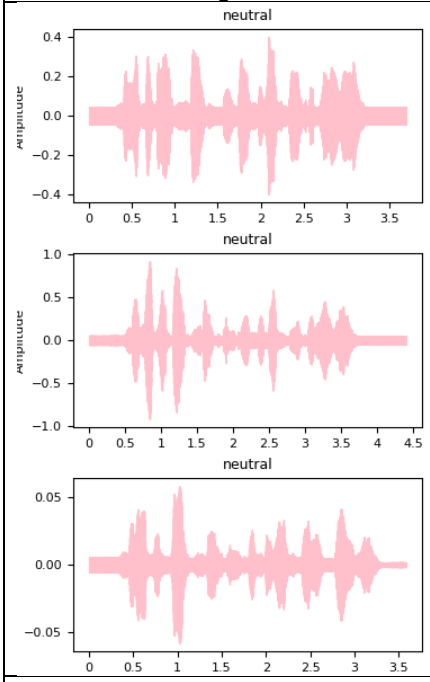
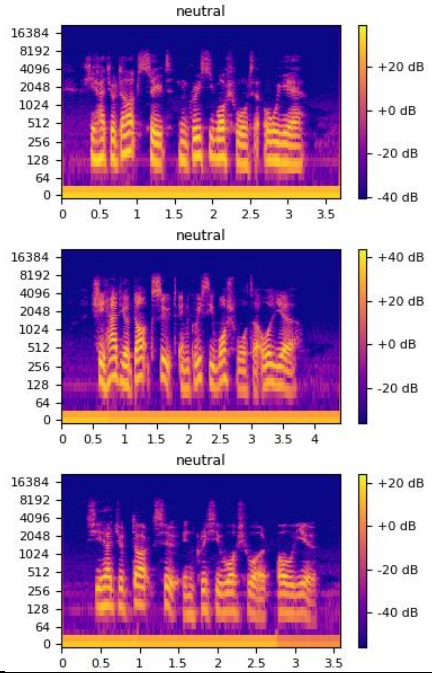
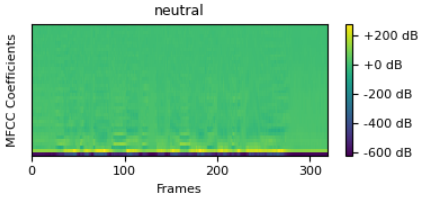
Emotion	RAVDESS
	

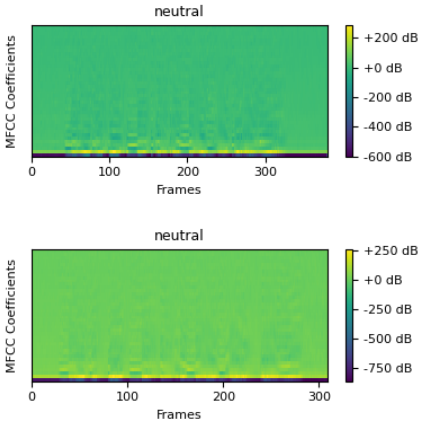
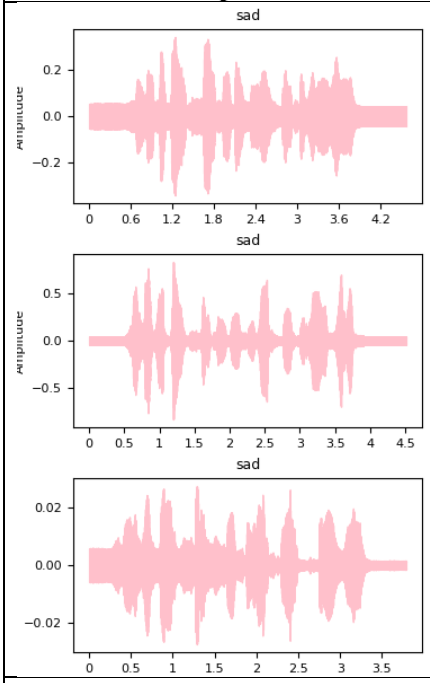
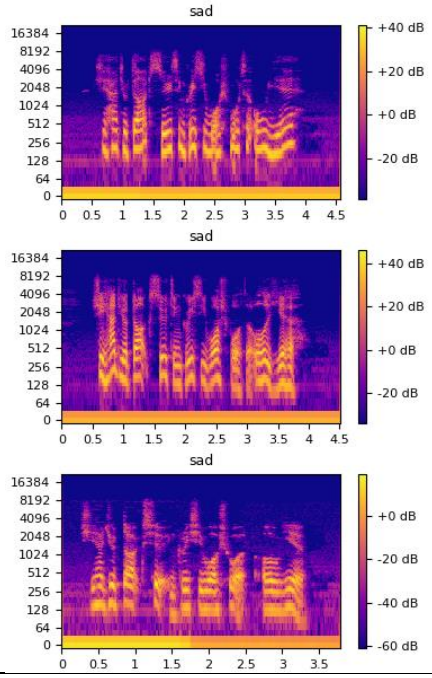
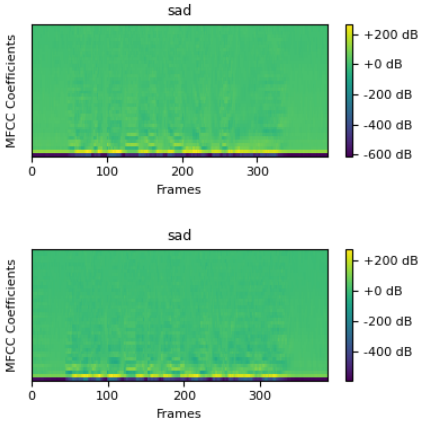
Appendix 3: SAVEE Data Exploration Table.

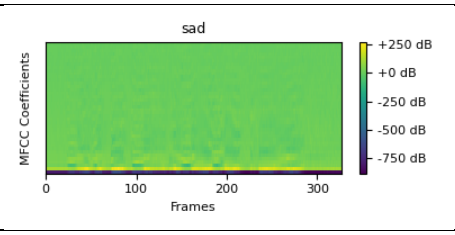
Emotion	SAVEE	
	Amplitude	**Spectogram**
Angry		
	MFCC	
		
Fear	**Amplitude**	**Spectogram**
		

Emotion	SAVEE	
		
	MFCC	
		
Disgust	Amplitude	Spectogram
		

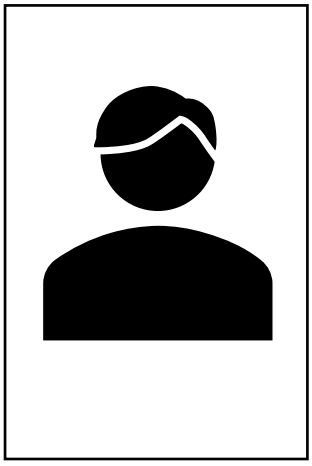
Emotion	SAVEE	
		
	MFCC	
	  	
Happy	Amplitude	Spectogram
		
	MFCC	

Emotion	SAVEE	
		
Neutral	**Amplitude**	**Spectogram**
		
	MFCC	
		

<table>
<tr><th>Emotion</th><th colspan="2">SAVEE</th></tr>
<tr><td></td><td colspan="2"> 
</td></tr>
<tr><td rowspan="2">Sad</td><td>Amplitude</td><td>Spectogram</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td colspan="2">MFCC</td></tr>
<tr><td colspan="2"> 
</td></tr>
</table>

Emotion	SAVEE
	

BIODATA PENULIS



Penulis dilahirkan di Madiun, 29 Januari 1985, merupakan anak pertama dari 4 bersaudara. Penulis telah menempuh pendidikan formal yaitu di TK ABA 18 Madiun, SDN Beteng 1 Madiun, SMPN 2 Madiun dan SMAN 2 Madiun. Setelah lulus dari SMAN tahun 2020, Penulis mengikuti SBMPTN dan diterima di Departemen Teknik Mesin FTIRS - ITS pada tahun 2020 dan terdaftar dengan NRP 02112040000130.

Di Departemen Teknik Mesin Penulis sempat aktif di beberapa kegiatan Seminar yang diselenggarakan oleh Departemen, Himpunan Mahasiswa Teknik Mesin (HMM) dan aktif sebagai Asisten Praktikum Mesin Konversi Enersi maupun Grader mata kuliah Termodinamika.