

**DOKUZ EYLÜL UNIVERSITY
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER ENGINEERING**

**EMOTION DETECTION FROM TV SERIES
AUDIO CLIPS**

**by
Tuncay KÖSE**

**Advisor
Dr. Özlem ÖZTÜRK**

**July, 2025
İZMİR**

EMOTION DETECTION FROM TV SERIES AUDIO CLIPS

**A Thesis Submitted to the
Dokuz Eylül University, Department of Computer Engineering
In Partial Fulfillment of the Requirements for the Degree of B.Sc.**

**by
Tuncay KÖSE**

**Advisor
Dr. Özlem ÖZTÜRK**

July, 2025

İZMİR

SENIOR PROJECT EXAMINATION RESULT FORM

We have read the thesis entitled "**EMOTION DETECTION FROM TV SERIES AUDIO CLIPS**" completed by **Tuncay KÖSE** under advisor of **Dr. Özlem ÖZTÜRK** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of B.Sc.

.....
Dr. Özlem ÖZTÜRK

.....

Advisor

.....

Committee Member

.....

Committee Member

Prof. Dr. Derya BİRANT

Chair

Department of Computer Engineering

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to my supervisor, Dr. Özlem ÖZTÜRK, for her availability and encouragement during the project.

This study was shaped by many hours of independent work, experimentation, and self-motivation. I am also thankful to the open-source community and the developers of tools and frameworks such as Whisper, PyAnnote, Hugging Face, and many others that enabled the technical backbone of this work.

I would also like to acknowledge the authors of online tutorials, blogs, and open-access academic resources that helped me better understand the complexities of speech emotion recognition.

Finally, heartfelt thanks go to my family for their continuous support and patience throughout this process.

Tuncay KÖSE

EMOTION DETECTION FROM TV SERIES AUDIO CLIPS

ABSTRACT

Speech Emotion Recognition (SER) is a key task in affective computing and conversational AI, aiming to infer human emotions from raw speech signals. This study presents a speaker-specific, speech-only SER framework built upon the MELD (Multimodal EmotionLines Dataset), where only utterances spoken by the character Joey from the Friends TV show are used. This focused design ensures consistent prosodic and acoustic characteristics across the dataset, enhancing modeling accuracy.

A robust multi-stage preprocessing pipeline was employed to ensure data integrity and alignment. Transcripts were generated using Whisper (large-v3), while speaker diarization was performed with PyAnnote (speaker-diarization-3.1). To verify transcript accuracy and audio-text alignment, multiple similarity metrics were computed—including WER, CER, Levenshtein, Jaccard, sequence similarity, and semantic similarity via the SentenceTransformer model all-MiniLM-L6-v2. Audio files were resampled to 16kHz, denoised, normalized, and filtered based on length and silence content. Advanced data augmentation techniques (pitch shift, time-stretch, echo, filters, inversion) were applied to balance emotional class distribution.

The feature extraction strategy was designed to accommodate both classical and modern learning pipelines:

1. *Handcrafted features: A rich set of acoustic descriptors—MFCC, pitch, spectral features, formants, jitter/shimmer, TEO, wavelet, and openSMILE's eGeMAPSv01a—were extracted and used across ML and DL models.*
2. *Visual features: Spectrogram-based 2D representations (Mel, MFCC, RGB fusion) were generated for CNN-based architectures.*
3. *Transformer-based features: Pretrained models from Hugging Face (WavLM, Wav2Vec2.0, HuBERT) were used in two modes: (a) frozen feature extraction with MLP classifiers, and (b) full fine-tuning for end-to-end learning.*

Three model categories were explored:

- *Classical ML models (9 types including SVM, RF, XGBoost, MLP): Trained over 11 feature subsets selected via methods like Boruta, RFE, SFFS, and Autoencoder-based reduction.*

- *Deep Learning (DL): CNN models using spectrogram images, BiLSTM and MLPs trained on 1D acoustic vectors, and hybrid CNN+BiLSTM architectures.*
- *State-of-the-Art (SOTA): Transformer models (WavLM, HuBERT, Wav2Vec2.0) evaluated with frozen embeddings and full fine-tuning using PyTorch.*

Evaluation was conducted on the Joey-specific test set (264 utterances). The best ML model (SVM with Boruta+Autoencoder features) achieved 86.43% accuracy and 86.51% F1-macro. The BiLSTM model outperformed all DL models with 91.29% accuracy and 92.45% F1-macro. Among SOTA approaches, fine-tuned Wav2Vec2.0 reached 84.09% accuracy and 79.73% F1-macro. Confusion matrices and class-wise metrics highlighted strong performance on emotions like fear, sadness, and disgust.

To operationalize the system, a GUI-based application was developed to support real-time inference using any ML/DL/SOTA model. This interface automates the feature extraction, scaling, and model-specific prediction pipelines, supporting user-friendly experimentation.

This project demonstrates that robust, modular, and speaker-personalized speech-only emotion recognition systems can be built by combining deep learning, classical techniques, and pretrained transformers. It highlights the importance of data quality, feature diversity, and architecture fusion in building next-generation SER solutions.

CONTENTS

	Page
SENIOR PROJECT EXAMINATION RESULT FORM	ii
ACKNOWLEDGEMENTS.....	iii
ABSTRACT.....	iv
TABLE OF CONTENTS.....	vi
LIST OF TABLES.....	x
LIST OF FIGURES	xi
CHAPTER ONE	
INTRODUCTION	1
1.1 Background Information	1
1.2 Problem Definition.....	2
1.3 Motivation.....	3
1.4 Goal/Contribution	5
1.5 Project Scope.....	6
1.6 Standards, Ethics, Constraints and Conditions	7
CHAPTER TWO	
LITERATURE REVIEW	10
2.1 Related Works	10
2.1.1 Emotion Recognition Using Multimodal Deep Learning	13
2.1.2 EEG-Based Emotion Detection via Frequency Features and SVM.....	14
2.1.3 Dyadic Fusion Networks with Attentive Correlation for SER.....	15
2.1.4 MEISD - A Multimodal Emotion & Sentiment Dialogue Dataset	15
2.1.5 Emotion Recognition on RAVDESS via Transfer Learning	16
2.1.6 Autoencoder-Based Emotion Recognition with MFCC Features	18
2.1.7 DNN + CNN + RNN Hybrid Approach on IEMOCAP Dataset.....	18
2.1.8 SER with MLP on Berlin Emotional Speech Dataset (Emo-DB).....	18
2.2 Comparison with the Existing Solutions	19

CHAPTER THREE	
REQUIREMENTS/REQUIREMENT ENGINEERING	23
3.1 Functional Requirements	24
3.1.1 Problem Statement	28
3.1.2 Feature Extraction Techniques for Speech Recognition	29
3.1.3 Dataset Characteristics and Multi-Stage Preprocessing Pipeline.....	30
3.1.4 Programming Language	31
3.1.5 Modelling Requirements: ML, DL, and SOTA Approaches	33
3.1.6 System Specification/Description	34
3.2 Non-Functional Requirements	36
3.2.1 User Requirements	36
3.2.2 Product Requirements	36
3.2.3 System Quality Characteristics	37
3.2.4 Emotional Goals as a Distinct Category	38
CHAPTER FOUR	
DESIGN	40
4.1 Architectural View	40
4.2 Database Design/ER Diagram.....	42
4.3 UML Class Diagram	46
4.4 UI Design	47
4.5 Use Cases Diagram	48
4.6 Sequence Diagram	49
4.7 Activity Diagram.....	50
4.8 Deployment Diagram.....	51
CHAPTER FIVE	
IMPLEMENTATION	53
5.1 Dataset Description	55
5.1.1 MELD	57
5.2 Emotion Recognition Modalities and Algorithms	72
5.2.1 Audio Modality	72

5.2.2 Multimodal Potential (Future Scope).....	77
5.2.3 Some Algorithms Used and Explored.....	79
5.3 Methodology/Tools/Libraries	87
5.3.1 Methodology Overview	87
5.3.1.1 Data Preparation and Cleaning.....	87
5.3.1.2 Audio Preprocessing	88
5.3.1.3 Data Augmentation	88
5.3.1.4 Feature Extraction	89
5.3.1.5 Feature Encoding and Normalization.....	90
5.3.1.6 Model Training and Evaluation	90
5.3.2 Tools and Libraries	91
5.4 Data Cleaning, Preprocessing and Augmentation	93
5.4.1 File Matching and Removal of Inconsistencies	94
5.4.2 Speaker and Gender Filtering	94
5.4.3 Audio Extraction from Video (mp4 → wav)	95
5.4.4 Audio Processing Pipeline	95
5.4.5 Augmentation Preparation and Logging	96
5.4.6 Logging and Traceability of Excluded Samples	96
5.4.7 Summary	97
5.5 Future Extraction and Selection	97
5.5.1 Classical Feature Extraction for ML and MLP Models.....	97
5.5.2 Visual Representations for Deep Learning CNN.....	99
5.5.3 Time-Series Feature Sequences for LSTM-Based Models.....	100
5.5.4 Transformer-Based Feature Embeddings (SOTA Models)	100
5.5.5 Feature Selection.....	101
5.5.6 Feature Normalization and Label Encoding	103
5.6 Model Training and Evluation	104
5.6.1 Data Preparation.....	105
5.6.1.1 Machine Learning Models	105
5.6.1.2 Deep Learning Models	107
5.6.1.3 Sota Models.....	108

5.6.2 Models Used	112
5.6.2.1 Classical Machine Learning Models.....	112
5.6.2.2 Deep Learning Models.....	117
5.6.2.3 Sota Models.....	125
5.6.3 Evaluation Metrics	135
5.6.4 Results.....	136
5.6.5 Training Time and Hardware Setup.....	137
5.6.6 Overall Assessment	138
5.7 Experimental Results and Discussion	139
5.8 Summary of Implementation	141
CHAPTER SIX	
TEST/EXPERIMENTS.....	143
6.1 Test Details	143
6.1.1 ML Prediction Pipeline	143
6.1.2 DL Prediction Pipeline	145
6.1.3 SOTA Prediction Pipeline	148
6.2 Experimental Results	151
6.2.1 ML prediction Results	151
6.2.2 Deep Learning Prediction Results	153
6.2.3 SOTA Prediction Results	156
CHAPTER SEVEN	
CONCLUSION AND FUTURE WORK.....	159
7.1 Conclusion	159
7.2 Future Work	160
REFERENCES.....	163

APPENDIX A	167
GUI Interface (Real-time Deployment)	167
APPENDIX B	172
Model Training and Evaluation: ML(11/99), DL (11/31), SOTA (9/9)	172
APPENDIX C	185
Best Prediction & Prediction and Experimental Results	185

LIST OF TABLES

Table 2. 1 Comparison of Accuracy Scores for Different Algorithms and Feature Types in EEG-based Emotion Detection (Nie et al., 2011).....	14
Table 3.1 Functional Requirement Analysis.....	25
Table 5.1 Column Specifications of Labeled Dataset.....	64
Table 5.2 Statistics of Dataset	64
Table 5.3 Distribution of Emotion Classes in the Training Set	65
Table 5.4 Distribution of Speakers in the Training Set	65
Table 5.5 Gender-wise Dist. of EC Mapped to Sentiment Categories in Train Set	65
Table 5.6 Dist. of Emotion Classes by Gender for Selected Speakers in Train Set	66
Table 5.7 Distribution of Main Characters in the Training Set (MELD)	66
Table 5.8 Distribution of Emotion Classes by Gender After Data Augmentation	66
Table 5.9 Emotion Distribution for the Speaker "Joey" after Dataset Filtering	67
Table 5.10 Emotion Distribution for "Joey" After Diarization, and Cleaning	67
Table 5.11 Emotion Distribution for "Joey" After Removing Short Audio Clips	67
Table 5.12 Emotion Distribution for "Joey" After Data Augmentation	68
Table 5.13 Final Balanced Emotion Distribution for Joey (Without Gender Prefixes) .	68
Table 5.14 Emotion Distribution of Dataset	69
Table 5.15 Example from train_sent_emo.csv	70
Table 5.16 Tools and Libraries Used in the Project	92
Table 5.17 Sota Pretrained-frozen Models	110
Table 5.18 Sota End-to-End Models.....	111
Table 6.1 Machine Learning Prediction Results.....	151
Table 6.2 Deep Learning Prediction Results	153
Table 6.3 Sota Prediction Results	156

LIST OF FIGURES

Figure 2.1 Confusion Matrix of Fine-Tuned CNN-14 on RAVDESS Dataset (Luna-Jiménez et al., 2021).....	17
Figure 3.1 Schematic Flowchart of Functionality Implemented for Emotion Detection from TV Series or Audio Clips Data	27
Figure 3.2 Schematic of Classification of Non-Functional Requirements	38
Figure 4.1 Architectural View	41
Figure 4.2.1 System and Database Design	43
Figure 4.2.2 ER Diagram.....	43
Figure 4.2.3 Architectural View	44
Figure 4.3 UML Class Diagram	46
Figure 4.4 UI Design	47
Figure 4.5 Use Cases Diagram	48
Figure 4.6 Sequence Diagram.....	49
Figure 4.7 Activity Diagram.....	50
Figure 4.8 Deployment Diagram	51
Figure 5.1 Emotion Distribution in MELD Dataset	58
Figure 5.2 Emotion Distribution in Train Set.....	58
Figure 5.3 Emotion Distribution by Speaker in Train Set	59
Figure 5.4 Emotion Distribution by Speaker in Test Set	59
Figure 5.5 Emotion Distribution by Speaker in Dev Set	59
Figure 5.6 Utterance Count per Emotion Heatmap for Main Speakers	60
Figure 5.7 Utterance Length Distribution in Train Set	60
Figure 5.8 Emotion Distribution by Gender in Train Set	60
Figure 5.9 Emotion Trend Across Dialogue IDs in Train Set	61
Figure 5.10 Example Dialogs	68
Figure 5.11 Dialog and Screen Example	70
Figure 5.12 Dataset Dialogs and Utterances Variables	70
Figure 5.13 Dataset Sample	71
Figure 5.14 Studies and Comparisons with this Data Set	80
Figure 5.15 Types of Recurrent Neural Network	83

CHAPTER ONE

INTRODUCTION

1.1. Background Information

With the rapid evolution of computational technologies, Human-Computer Interaction (HCI) has significantly advanced from basic logical tasks to complex interactions that seek to interpret and respond to human mental and emotional states (Association of Human-Computer Interaction, 2023). Within this paradigm, affective computing has emerged as a critical subfield, focusing on enabling machines to recognize, interpret, and respond to human emotions by analyzing cues derived from various biometric and behavioral modalities—such as facial expressions, speech signals (which is the primary focus of this project), gestures, and physiological responses (Dix, 2009).

Emotions play a central role in human cognition, decision-making, and social interaction. Therefore, the ability of a computer system to understand emotional states is vital for achieving more natural and intuitive interaction with users. Speech Emotion Recognition (SER) is one of the most promising and widely researched components of affective computing, primarily because speech is a natural and spontaneous mode of communication for humans. Unlike text, speech contains rich acoustic features such as pitch, tone, rhythm, and energy, which are inherently influenced by the speaker's emotional state.

Changes in vocal expressions—such as variations in pitch, loudness, and speech rate—can be detected even when the linguistic content remains constant, as emotional states cause significant biophysical variations in speech production. This strong correlation between acoustic features and emotions has led to the development of SER systems that can infer emotional states—such as happiness, sadness, anger, or fear—from raw audio signals. The MELD dataset used in this study consists of emotionally-rich dialogues extracted from the TV series "Friends", providing a realistic environment for training and evaluating SER models. In this project, these speech-based emotional clues are explored using audio data extracted from natural dialogues in a TV series environment.

Furthermore, emotional communication is rarely limited to a single, static state. In natural human conversations, individuals often exhibit multi-label emotional states—experiencing and expressing multiple emotions with varying intensities simultaneously. This presents a major challenge for emotion detection systems, particularly in dialogue-rich scenarios such as conversations in TV series or real-life human interactions. Moreover, in this project, both single-label and multi-label classification approaches are considered, and a GUI-based interface has been developed to facilitate model-based inference.

In this context, sentiment and emotion classification in spoken dialogues has emerged as a challenging yet crucial task. Multi-label emotion recognition enables a system to better understand the nuanced feelings of users, while sentiment analysis contributes to grasping the speaker's stance or opinion regarding the topic being discussed. Together, these approaches contribute significantly to enhancing the emotional intelligence of conversational AI systems.

1.2. Problem Definition

In human interactions, one of the most complex and often problematic aspects is the accurate perception and interpretation of emotional states. Misunderstandings in communication frequently stem from the inability to detect subtle emotional cues, especially in voice, which inherently conveys acoustic cues such as tone, prosody, pitch, and affective nuances beyond lexical content. In the context of human-machine interaction, this challenge becomes even more prominent. Machines traditionally process structured input, yet emotions are inherently unstructured, context-dependent, and multifaceted.

This project addresses the question: "Can an artificial intelligence system be developed that accurately detects and classifies human emotions from audio data extracted from dialogues in TV series—specifically from the MELD dataset, which contains emotionally rich utterances." The motivation arises from a fundamental communication dilemma—people often struggle to understand each other's emotions, leading to confusion, conflict, or misinterpretation. By developing a tool that can autonomously analyze vocal signals

and infer underlying emotional states, I aim to bridge this gap in emotional understanding, even in pre-recorded or real-time audio streams.

The core objective is to explore whether the timeless and universal phenomenon of human emotion can be modeled, learned, and inferred by artificial intelligence through speech. More specifically, I seek to understand whether emotional states—manifested through acoustic features such as pitch, tone, and prosody—can be mapped to specific emotion classes using classical machine learning, deep learning, and state-of-the-art transformer-based models.

The applications of such systems go beyond academic curiosity; it has the potential to significantly enhance areas such as mental health monitoring, human-robot interaction, customer service automation, media analysis, and more. Thus, the problem this project seeks to solve is not merely technical—it also touches upon a broader social challenge: enhancing empathy and emotional understanding through artificial intelligence.

In this project, emotional signals are extracted exclusively from audio modality (TV series dialogues), without using facial or textual cues, to simulate a more realistic and challenging real-world emotion detection environment. To support user interaction and model testing, a graphical user interface (GUI) has also been developed as part of the system.

1.3. Motivation

Human emotions are a fundamental aspect of communication, deeply influencing how information is conveyed and interpreted. The ability to automatically recognize emotions from speech has profound implications for improving human-computer interaction, enabling machines to become more emotionally aware and responsive.

Despite the growing interest in Speech Emotion Recognition (SER), challenges remain regarding the quality, consistency, and contextual reliability of available emotional datasets. Emotion datasets are generally categorized into three types based on how emotions are elicited: spontaneous, invoked, and acted. Spontaneous emotions are often captured from real-world contexts such as TV shows, interviews, or live broadcasts. Although these sources provide authentic emotional expressions, they are often plagued with issues such as background noise, overlapping speech, and lack of consistent frontal

facial views, all of which hinder precise emotion annotation and extraction. MELD, one of the few spontaneous datasets built from TV show dialogues, provides a realistic yet challenging benchmark due to overlapping speech and emotion ambiguity.

An example of a spontaneous audiovisual dataset is the Adult Attachment Interview (AAI), where participants discuss personal experiences, and emotions are labeled using the Facial Action Coding System (FACS). Similar to the Adult Attachment Interview (AAI), which captures spontaneous responses during personal discussions, the MELD dataset also contains natural expressions of emotion in dialogue-rich environments. However, while AAI is structured for psychological analysis, MELD is built for SER tasks under challenging audiovisual conditions. While rich in emotional content, such datasets also introduce challenges in standardization and preprocessing (Glenn&Roisman,2007).

On the algorithmic front, SER systems leverage various machine learning techniques such as Support Vector Machines (SVM) and K-Nearest Neighbors (KNN). Audio features extracted typically include time domain, frequency domain, and cepstral features such as MFCC (Mel Frequency Cepstral Coefficients). Tools like pyAudioAnalysis, WEKA, OpenSMILE, and custom pipelines using Python libraries are used to automate feature extraction, selection, and hyperparameter tuning, significantly improving classification accuracy (Knowledge and Smart Technology, 2013). In this study, in addition to classical ML algorithms, various deep learning models (e.g., CNN, LSTM) and transformer-based architectures (e.g., Wav2Vec2, HuBERT, WavLM) are also integrated—enabled through PyTorch and Hugging Face libraries for enhanced representation learning and end-to-end emotion classification.

This project is motivated by the need to build a robust SER system capable of operating in realistic, noisy environments such as dialogues from TV series. By investigating existing techniques and combining effective feature extraction methods with optimized classification models, I aim to advance the emotional intelligence of machines through speech. To make the system more accessible and testable, a GUI-based inference interface has been developed where users can select a model and test audio inputs in real-time.

Additionally, the use of real-life, spontaneous dialogues extracted from multi-speaker television series poses unique challenges in terms of speaker diarization, background

noise, and segment-level alignment—further motivating the development of robust preprocessing (including speaker diarization, noise reduction, and segment alignment) and modeling techniques.

1.4. Goal / Contribution

The primary goal of this project is to develop an artificial intelligence system capable of recognizing and learning human emotions from audio data, specifically from video sources such as TV series—using the MELD dataset, which is composed of emotionally-rich dialogue clips. The system is designed to parse, process, and classify speech signals to identify underlying emotional states. By doing so, it aims to improve the understanding and interpretability of human emotions in digital environments, contributing significantly to the field of affective computing and speech emotion recognition.

To achieve this, video-based audio datasets were collected, processed, and segmented for training purposes. Feature extraction will be applied to transform the raw audio data into meaningful representations, which were then used for training supervised classification models across various architectures. The outputs of these clusters will be analyzed and used to train machine learning models. The trained models will infer emotional states from new, unseen audio data, enabling the system to generalize and maintain inference capabilities across varying contexts.

This project contributes by:

- Creating a robust pipeline for extracting emotion from realistic, noisy, multi-speaker audio sources using TV show dialogues.
- Implementing and optimizing machine learning, deep learning, and transformer-based models for both single-label and multi-label emotion detection.
- Developing a scalable emotion detection system that can be expanded to include other modalities such as facial expressions and textual sentiment in future iterations.
- Designing and implementing a modular pipeline supporting classical ML, deep learning, and SOTA models—each with tailored feature extraction and evaluation logic.

This project stands out technically in that it builds an end-to-end, modular emotion recognition pipeline that supports multiple model architectures (ML, DL, SOTA) using customized feature extraction strategies for each, and provides a GUI-based interface for real-time prediction and usability. Moreover, it employs advanced preprocessing techniques—including speaker diarization with Whisper and PyAnnote, background noise removal, short audio filtering, and class balancing via augmentation and undersampling—to ensure real-world applicability and high performance even under noisy, multi-speaker dialogue scenarios.

In future stages, the system aims to achieve more accurate and robust predictions by integrating complementary technologies such as face recognition, optical character recognition (OCR), and advanced speech recognition. Enhancing algorithmic performance and data processing pipelines will lead to more reliable emotional inferences, supporting clearer interpretation of both audio and audiovisual inputs. Ultimately, this advancement will enable more emotionally intelligent systems. These systems will be better suited for human-centered applications in communication, education, entertainment, and healthcare.

1.5. Project Scope

This project focuses on extracting audio features from emotionally labeled audio clips derived from a specific TV series ("Friends") via the MELD dataset, and using these features to train a machine learning model capable of detecting the emotional content within speech. The dataset utilized in this study contains audio clips that are already labeled with corresponding emotional tags, allowing for a structured approach in training the system.

The core objective is to process and evaluate any given audio-based dataset—be it sound recordings, dialogue clips, or audio extracted from video content—with the framework of a modular system incorporating machine learning, deep learning, and transformer-based emotion recognition models. Once emotional cues are extracted from

these audio signals, the goal is to enable the artificial intelligence model to learn and classify these cues into distinct emotional categories.

In order to achieve this goal, the following priorities have been defined:

- Accurate and complete reading of audio files.
- Full extraction of relevant features from the input audio data.
- Classification of the extracted features through machine learning algorithms.
- Evaluation and comparison of different models across various feature sets to iteratively improve classification performance.

Artificial intelligence is one of the key pillars of modern technology, and within this context, audio signal processing plays a vital role in enabling deeper human-computer interaction. Recognizing the importance of emotion in speech, this project analyzes emotional signals embedded in audio and teaches an AI model to interpret them using customized preprocessing, feature extraction, and modeling techniques. The system currently focuses solely on the audio modality, using speaker-specific samples—specifically the character Joey from the MELD dataset—for consistency and reliability. Integration of facial and textual modalities is considered as a potential extension in future development stages.

1.6. Standards, Ethics, Constraints, and Conditions

In developing a Speech Emotion Recognition (SER) system that processes audio clips from a single television series (*Friends*) via the MELD dataset, it is imperative to adhere to established standards, address ethical considerations, and recognize the inherent constraints and conditions associated with such technology.

Standards

The project aligns with industry standards for audio data processing and machine learning:

- **Data Acquisition and Annotation:** Utilizing publicly available datasets—specifically the MELD dataset—that are pre-annotated for emotional content ensures consistency and reliability in training and evaluation.
- **Feature Extraction and Selection:** Employing standardized acoustic features, such as Mel-frequency cepstral coefficients (MFCCs), pitch, and energy,

facilitates effective emotion recognition using consistent, reproducible features across model types (ML, DL, and transformer-based models).

- **Model Evaluation:** Adopting common performance metrics, including accuracy, precision, recall, and F1-score, enables objective assessment of the system's efficacy. Standard metrics were used not only for model comparison but also to select the best feature subsets and classification strategies.

Ethical Considerations

Developing and deploying SER systems necessitate careful attention to ethical issues:

- **Privacy and Consent:** Ensuring that all audio data used respects individuals' privacy rights and is obtained with proper consent is paramount.
- **Bias and Fairness:** Addressing potential biases in training data is crucial to prevent unfair outcomes across different demographic groups. Although the MELD dataset originates from a scripted TV show and does not include explicit demographic annotations, potential speaker and gender imbalance (e.g., dominance of 'Joey') was mitigated through data augmentation and undersampling. Studies have highlighted risks of biased and unfair outcomes due to faulty premises in emotion recognition technologies.
- **Transparency and Explainability:** Designing models that provide interpretable results fosters trust and facilitates user understanding of system decisions.
- **Potential for Misuse:** Implementing safeguards against the misuse of emotion recognition technology is essential, particularly in sensitive areas such as employment and law enforcement. The risk of harm arising from the use of such technologies in consequential settings has been documented.
- **Speaker Anonymity:** Even though TV series have public content, careful processing is done to ensure no identity-specific or personally sensitive content is used outside of its emotional labeling context. Additionally, speaker diarization was applied to isolate target speakers and ensure clarity of emotional attribution.

Constraints and Conditions

Several practical constraints and conditions influence the development of the SER system:

- **Data Quality:** Variability in audio quality, background noise, and speaker diversity can impact the accuracy of emotion detection.
- **Computational Resources:** Balancing model complexity with available computational resources is necessary to ensure efficient processing and real-time performance, especially when integrated into GUI-based inference tools.
- **Generalizability:** Developing models that generalize well across different contexts and cultures is challenging due to the subjective nature of emotional expression, which varies across cultures, speaking styles, and recording conditions.
- **Regulatory Compliance:** Adhering to legal frameworks and regulations governing data protection and ethical AI use is mandatory.

By conscientiously addressing these standards, ethical considerations, constraints, and conditions, the project aims to develop a responsible and effective Speech Emotion Recognition system that respects user rights and delivers reliable performance in real-world, audio-only, multi-speaker dialogue scenarios.

CHAPTER TWO

LITERATURE REVIEW

2.1. Related Works

Emotion recognition is a field of study that will enable us to open the door to the development of human interaction and to a much more advanced level of interaction in the future. The emotion recognition is a challenging task as human emotions lack of temporal boundaries and ways of expressing emotion varies in human to human. The process of expressing or perceiving human emotions uses different forms of knowledge and these are defined as linguistics and para-linguistics. The main difficulty in emotion identification is the para-linguistic components, including vocal tone, prosody, facial expressions, gestures, and physiological responses.

Many different studies and methods have been presented in emotion recognition. The concept of multimodal emotion recognition depends on combining multiple modalities. It largely depends on the performance of the base. In multimodal emotion recognition systems, audio-visual information is used as the basic model and multiple emotion recognition systems are used within this information.

Emotion recognition from speech has emerged as a pivotal area in affective computing and human-computer interaction. Over the past two decades, numerous studies have proposed diverse methods and approaches to model emotional states using audio signals. These studies typically focus on three critical stages: feature extraction, feature selection, and classification.

While much of the existing literature focuses on multimodal approaches combining audio, text, and visual cues, this study focuses exclusively on the speech modality. By targeting raw audio from TV series—characterized by spontaneous, emotionally-rich, yet noisy dialogue—the project explores the effectiveness of acoustic-based emotion recognition in realistic, speaker-specific settings using only audio modality from natural TV dialogues.

Feature Extraction

The extraction of relevant features from speech signals is the foundation of any emotion recognition system. Widely used acoustic features include Mel-Frequency Cepstral Coefficients (MFCC), Pitch, Energy, Spectral Centroid, Formants, Zero-Crossing Rate, and Chroma Features. Tools such as OpenSMILE, Librosa, Praat, and MIRtoolbox are commonly employed for this task. Research indicates that hybrid features, such as combining prosodic and spectral elements, lead to improved accuracy in emotion classification.

In this study, various types of acoustic features were extracted depending on the model architecture. For classical ML models and MLP variants, over 1400 statistical and signal-based features were computed using Librosa, OpenSMILE (eGeMAPSv01a), and Parselmouth (e.g., MFCC, pitch, shimmer, jitter, HNR, formants, spectral features, TEO, DFT, wavelet, etc.).

For CNN-based models, Mel spectrograms, MFCC spectrograms, and RGB-fused spectrogram images (where MFCC, delta, and delta² are combined as RGB channels) were generated.

For sequence models such as LSTM, BiLSTM, and GRU, a padded time-series tensor of MFCC + delta + delta² features ([T,120]) was constructed.

In state-of-the-art (SOTA) approaches, pretrained models such as WavLM, HuBERT, and Wav2Vec2 were used as feature extractors, and their output embeddings were classified using an additional MLP layer.

Feature Selection

To reduce model complexity and improve generalization, feature selection techniques are used to retain the most relevant attributes. Techniques like Principal Component Analysis (PCA), Information Gain, Autoencoders, and Genetic Algorithms have shown promising results. For instance, studies demonstrate that the Autoencoder + MFCC combination can achieve accuracy rates above 90% on datasets like TESS and EmoDB.

To reduce dimensionality and improve model generalization, multiple feature selection strategies were evaluated, including VarianceThreshold, SelectKBest (Mutual

Info, Chi²), Random Forest Importance, Recursive Feature Elimination (RFE), Sequential Forward Floating Selection (SFFS), Boruta, and Autoencoder-based dimensionality reduction. Feature selection was customized for classical ML models and deep MLP classifiers. This ensured that each model received the most relevant subset of features, helping to reduce overfitting and improve generalization capability. SOTA and CNN-based models used raw representations without feature reduction.

Classification Models

Various machine learning classifiers have been applied to emotion recognition tasks. Support Vector Machines (SVM) and K-Nearest Neighbors (KNN) remain popular due to their robustness. However, recent advancements lean towards deep learning architecture. Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory Networks (LSTMs) have significantly enhanced the modeling of temporal and contextual dependencies in speech signals.

Some notable works include:

- Aouani et al. (2020) achieved 74.07% accuracy using autoencoders and SVM with RBF kernel on the RML dataset.
- Zengwei Yao et al. (2020) implemented a fusion of DNN, CNN, and RNN on the IEMOCAP dataset, achieving 58.3% UA, outperforming individual classifiers.
- Gökhan Polat and Halis Altun (2008) used a multilayered neural network on the Berlin dataset and reported 81.65% test accuracy, with sadness being the most accurately classified emotion.

These studies underscore the effectiveness of combining powerful feature extraction techniques with optimized classifiers. Moreover, datasets like RAVDESS, TESS, EmoDB, and IEMOCAP have been extensively used for benchmarking models.

In this project, classification models were structured into three major groups to assess the impact of architecture and feature design on performance:

- (1) Classical Machine Learning models (e.g., SVM, RF, XGBoost, k-NN, sklearn-MLP, GradientBoosting, BaggingClassifier(DT, k-NN), Hard-Voting (majority), Soft-Voting (probability-based)),
- (2) Deep Learning models (CNN, ResNet18, EfficientNetB0, LSTM, BiLSTM, GRU, MLP, CNN + BiLSTM),
- (3) State-of-the-Art pretrained models (Wav2Vec2 + MLP, WavLM + MLP, HuBERT + MLP), and transformer-based end-to-end models (Wav2Vec2, WavLM, HuBERT).

Each model type was trained using features tailored to its design—for example, CNN models utilized spectrogram images (Mel, MFCC, RGB), whereas RNN-based models consumed time-series MFCC-based tensors. SOTA models employed Hugging Face transformers and were either fine-tuned end-to-end or used as frozen extractors combined with lightweight classifiers. All models were evaluated using balanced, speaker-specific data from the MELD dataset (Joey-only), processed with advanced preprocessing techniques.

In recent years, speech emotion recognition (SER) has become a central research domain within affective computing, leveraging machine learning and deep learning techniques to classify emotions from audio signals. Numerous studies have explored a variety of datasets, architectures, and feature extraction strategies. Unlike many prior works that utilize full multimodal setups or laboratory-grade speech, this project evaluates speech-only emotion detection under real-world constraints—using speaker-specific, noisy, spontaneous TV dialogues. Below are some of the most influential and relevant works in this field:

2.1.1. Emotion Recognition Using Multimodal Deep Learning

(Wei Liu, Wei-Long Zheng, Bao-Liang Lu, 2016)

This study utilized the SEED and DEAP datasets to investigate emotion recognition using multimodal deep learning techniques.

Features such as Spectral Power Density (SPD) and Differential Entropy (DE) were extracted and used as inputs for a Bimodal Deep Autoencoder (BDAE) network for feature selection.

The classification was performed using a linear Support Vector Machine (SVM) following the autoencoder training.

The proposed method achieved an accuracy rate of over 80% on both datasets.

The study highlighted that combining multi-psychological signals with multi-modal deep learning architectures significantly improves emotion recognition performance and could play a role in future emotion-aware HMI (Human Machine Interface) systems. However, such systems are often limited by the complexity and cost of capturing physiological signals in real-world settings.

2.1.2. EEG-Based Emotion Detection via Frequency Features and SVM

(Dan Nie, Xiao-Wei Wang, Li-Chen Shi, and Bao-Liang Lu, 2011)

This study aimed to classify emotions by analyzing EEG (electroencephalography) signals recorded while subjects watched emotional video clips. Time and frequency domain features were extracted from EEG signals for emotion classification.

Three classifiers were applied: K-Nearest Neighbors (KNN) with Euclidean distance, a Multilayer Perceptron (MLP), and Support Vector Machine (SVM). The SVM classifier achieved 66.51% accuracy using frequency domain features, significantly outperforming both KNN (59.84%) and MLP (63.07%).

The study concluded that frequency domain features combined with SVM provide a more reliable framework for EEG-based emotion detection compared to other traditional classifiers. While EEG-based approaches like this show promise, their reliance on physiological sensors limits practical application. In contrast, speech-based systems offer more accessible and scalable alternatives for real-world emotion recognition.

Table 2.1 Comparison of Accuracy Scores for Different Algorithms and Feature Types in EEG-based Emotion Detection (Nie et al., 2011)

Algorithm/Feature	Frequency	Time
KNN	%59,84	%36,61
MLP	%63,07	%38,66
SVM	%66.51	%43.39

As a result of the study, they mentioned that the SVM algorithm gives a high accuracy rate with an accuracy rate of 66.51%, and that EEG signals will give more efficient results in emotion diagnosis.

2.1.3. Dyadic Fusion Networks with Attentive Correlation for SER

(Yue Gu, Xinyu Li, 2019)

This study proposed a Dyadic Fusion Network (DFN) architecture for speech emotion recognition, integrating both acoustic (A) and textual (T) modalities.

Features were extracted using openSMILE for acoustic data and preprocessed textual embeddings, with model implementation done via Keras and TensorFlow.

Several baseline methods were compared, including Support Vector Machine (SVM), Random Forest (RF), Long Short-Term Memory (LSTM) networks, and Tensor Fusion Network (TFN).

Performance was evaluated separately for text (T), acoustic (A), and combined text-acoustic (T+A) inputs.

The proposed Dyadic Fusion Network (DFN) achieved classification accuracies of 70.6% (acoustic only), 71.9% (text only), and 74.2% (acoustic + text), outperforming all unimodal and baseline fusion models..

The study demonstrated that multimodal fusion with attentive correlation mechanisms significantly enhances emotion recognition accuracy compared to unimodal approaches. However, the fusion-based accuracy gain also highlights the challenge of relying solely on a single modality-like speech, which this project aims to address using advanced acoustic modeling techniques.

2.1.4. MEISD - A Multimodal Emotion & Sentiment Dialogue Dataset

(Tripathi, Beigi, & Hazarika, 2018)

This study introduced a large-scale multimodal, multiparty conversational dataset, called MEISD, designed for multi-label emotion classification, intensity prediction, and sentiment analysis tasks within dialogues.

The authors emphasized that while research on emotion and sentiment recognition in dialogues has increased significantly since the 2010s, most existing datasets were limited to representing only a single emotion or exhibited class imbalance, reducing generalizability of trained models.

To address these limitations, the MEISD dataset was compiled from various TV series dialogues, including textual, audio, and visual features, aiming to provide a more balanced and comprehensive representation of multiple emotions and their intensities within conversational contexts.

The study outlined two main types of dialogue systems: task-oriented systems and open-domain chat systems, highlighting the importance of emotion recognition for enhancing user experience and interaction quality in both.

The authors demonstrated through experimental baselines that incorporating visual information alongside audio and textual data improves the accuracy of emotion and sentiment classification models.

Key contributions of this work include:

- Development of the MEISD dataset, supporting multi-label emotion recognition, intensity estimation, and sentiment analysis in multiparty conversations.
- Establishment of baseline models for the three primary tasks—multi-label emotion recognition, emotion intensity estimation, and sentiment analysis.

While MEISD provides a rich multimodal resource, this project focuses on the audio modality alone to evaluate the limits and potential of acoustic-only emotion recognition under realistic dialogue conditions.

2.1.5. Emotion Recognition on RAVDESS via Transfer Learning

(Luna-Jiménez et al., 2021)

In this study, the authors proposed a multimodal emotion recognition solution tested on the RAVDESS dataset, which incorporates both audio and visual data to detect emotional states.

They developed a new fusion model combining these modalities to improve emotion recognition accuracy.

To ensure the reproducibility and adaptability of the proposed model under real-world conditions, the researchers employed a cross-validation (CV) strategy. Cross-validation was used to assess the statistical significance of different models based on performance across one or more users.

The study also explored the use of transfer learning (TL) for speech- and visual-based models to enhance the model's ability to generalize across diverse scenarios.

Among the tested algorithms, the CNN-14 architecture yielded the best performance with 76% accuracy on the RAVDESS dataset.

The emotion "sadness" showed the lowest accuracy score, suggesting that this emotion is more challenging to recognize compared to others.

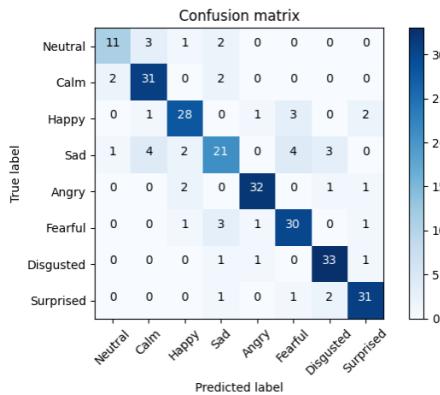


Figure 2.1 Confusion Matrix of Fine-Tuned CNN-14 on RAVDESS Dataset (Luna-Jiménez et al., 2021).

The highest misclassification occurs in the “sadness” class, indicating difficulty in detecting low-arousal emotions.

While this study demonstrates the effectiveness of multimodal fusion and CNN-based transfer learning, this project investigates whether similar levels of accuracy can be achieved using only acoustic features in realistic, dialogue-based settings.

2.1.6. Autoencoder-Based Emotion Recognition with MFCC Features

(Aouani et al., 2020)

This study focused on the RML dataset, using a stacked autoencoder for feature compression and SVM for classification.

- MFCC features were extracted from voice samples.
- Achieved 74.07% accuracy using SVM with RBF kernel.
- Highlighted that deep compression methods paired with powerful classifiers yield high precision.

2.1.7. DNN + CNN + RNN Hybrid Approach on IEMOCAP Dataset

(Zengwei Yao et al., 2020)

Combining different neural networks, this study applied DNN, CNN, and RNN fusion on the IEMOCAP dataset.

- Reported 58.3% Unweighted Accuracy (UA) for the Hybrid Model.
- Demonstrated that hybrid models outperform traditional classifiers.
- Also noted emotion-wise imbalance in predictions, with emotions like happiness being harder to detect particularly in low-arousal or overlapping emotions such as happiness, which were frequently confused with neutral.

2.1.8. SER with MLP on Berlin Emotional Speech Dataset (Emo-DB)

(Gökhan Polat and Halis Altun, ELECO 2008)

In this early study, researchers used a multilayer perception on the Emo-DB (Berlin) dataset.

- Achieved 81.65% test accuracy.
- The emotion sadness was classified with the highest precision.
- Reinforced the importance of feature quality and data balance.

2.2. Comparison with the Existing Solutions

Unlike most traditional studies that rely on clean, balanced, and well-segmented datasets with ideal conditions, this project introduces a more realistic and challenging setup by working with noisy, real-world audio clips derived from spontaneous dialogues in TV series. The methodology includes speaker-specific filtering (e.g., isolating the Joey character), advanced audio preprocessing techniques (noise reduction, diarization, VAD), and model-specific feature extraction tailored for ML, DL, and SOTA architectures. In addition, while many existing works perform single-label classification, this project embraces the complexity of multi-label emotional states—allowing overlapping and mixed emotional expressions, which are common in real-life conversations.

Numerous emotion recognition approaches exist in the literature, each offering unique strengths and facing inherent limitations—particularly when transitioning from controlled lab conditions to real-world settings. Past and current research emotion recognition It is seen that versatility dominates the field.

Ensuring the accuracy and reliability of data is critical for building effective and generalizable emotion recognition models. Accurate, usable, and complete data will be used in future classifications, and it is of great importance that our data be complete and reliable when comparing these classes correctly. In addition, the classification algorithms employed, and the system design should be mutually supportive and consistent.

After finalizing parameter tuning and resolving classification challenges, the next focus was on feeding the processed data into appropriate deep learning architectures with optimized training pipelines. At this stage, of course, our priority is to carefully design it as a whole, except for the right methods and algorithms.

As a result, all the software (programming language, data, libraries etc.) components that will be used in the field of emotion detection in the project should be brought together. This will complete the speech-based emotion recognition system as proposed in the project.

Speech Emotion Recognition (SER) has evolved significantly, with numerous methodologies proposed across literature—ranging from traditional machine learning approaches to sophisticated deep learning models. In this section, the current project's methodology and objectives are compared with existing solutions, particularly focusing

on aspects such as feature extraction, classification strategies, dataset utilization, and system robustness in noisy environments.

Feature Extraction Techniques

Existing studies heavily rely on acoustic features, particularly Mel-Frequency Cepstral Coefficients (MFCC), pitch, energy, and spectral characteristics. Tools like OpenSMILE, Librosa, and Praat are standard for extracting these features. For example, Yao et al. (2020) combined MFCC with High-Level Statistical Features (HSF), showcasing high accuracy on the IEMOCAP dataset using DNN, CNN, and RNN fusion models.

Our approach similarly incorporates MFCC but emphasizes real-world applications by focusing on audio from TV series, introducing variability in speech quality, background noise, and speaker diversity. This introduces an added layer of complexity that many benchmark-focused studies often do not fully address.

Unlike many prior studies that use fixed feature sets across all models, this project customizes the feature extraction process based on the model architecture. CNNs utilize spectrogram images (Mel, MFCC, RGB fusion), RNN-based models use time-series tensors, and traditional ML models employ over 1400 handcrafted statistical and signal-based features extracted using Librosa, Parselmouth, and OpenSMILE. Additionally, SSL-based SOTA models such as Wav2Vec2, HuBERT, and WavLM are used as pretrained feature extractors for classification via shallow neural layers.

Classifier Comparison

While Support Vector Machines (SVM), Naive Bayes, and Decision Trees have been widely used in traditional setups, deep learning methods like CNNs, LSTMs, and autoencoders have shown superior performance in emotion recognition tasks. For instance, Aouani et al. (2020) demonstrated improved accuracy using an autoencoder + SVM combination on the RML dataset. Similarly, this project also implements autoencoder-based dimensionality reduction followed by SVM and MLP classifiers to evaluate performance across different feature subsets.

Our system builds upon this by implementing a multi-label classification framework, allowing detection of multiple emotional states simultaneously, which is especially

relevant in natural dialogues. Most traditional models focus on single-label classification, limiting their applicability in dynamic and overlapping emotional contexts. Moreover, this study evaluates a wide range of classification paradigms—including classical ML, deep learning, and pretrained transformer-based SOTA models—ensuring that each model is tested with features most compatible with its design. Ensemble and hybrid models such as CNN + BiLSTM and Wav2Vec2 + MLP are also explored to measure synergy across architectures.

Dataset Utilization

Many studies utilize clean, well-segmented datasets such as RAVDESS, EmoDB, TESS, and IEMOCAP. While these provide controlled environments ideal for benchmarking, they often fail to replicate the complexities of real-life speech data. Our project uses emotion-labeled TV series audio clips, which mirror spontaneous human conversation and introduces challenges like overlapping speech, environmental noise, and inconsistent emotion intensity.

Furthermore, datasets like MEISD have introduced multi-modal, multi-label data, but still often rely on structured interviews or scripted content. By focusing on spontaneous dialogues from diverse TV series, this project captures more authentic emotional nuances, pushing SER models closer to real-world applicability.

In addition, to increase consistency and reduce label ambiguity, only speech segments from the Joey character were selected using speaker diarization (PyAnnote), ensuring that all audio corresponds to a single speaker with aligned emotion annotations. This refinement enhances the dataset's usability while preserving its natural variability.

Multi-Modality and Adaptability

While some state-of-the-art systems incorporate multimodal data—such as facial expressions and physiological signals—our current scope focuses solely on speech-based recognition. However, the system architecture is designed to be scalable and integratable with additional modalities such as facial emotion recognition, OCR, and sentiment analysis, allowing for future upgrades.

Although only the audio modality was used in this project, the preprocessing pipeline and model infrastructure are designed to support future expansion into multimodal systems.

Generalizability and Real-World Robustness

Many prior works achieve high accuracy in lab conditions but struggle in noisy, uncontrolled environments. This also includes filtering out short or corrupted audio clips, applying RMS-based noise removal, and dynamic thresholding during data cleaning to ensure high-quality emotional signal retention. Our system, trained on diverse, noisy, and context-rich data, aims to improve generalizability and robust emotion classification in real-world conditions, making it more applicable to domains like media analytics, call centers, and virtual assistants.

In contrast to the works surveyed, this project presents a practical SER pipeline built on real-world TV series audio data, using advanced preprocessing techniques such as Whisper-based transcription alignment and PyAnnote speaker diarization. By filtering only Joey's speech segments and applying model-specific feature extraction, the system ensures label consistency and contextual clarity. The architecture supports multiple model types (ML, DL, SOTA), each trained with features tailored to its design. This modular and robust approach targets real-world emotional complexity, where noise, speaker overlap, and spontaneous expression dominate. In line with previous findings (Yao et al., 2020), this study also observed classification challenges in lower arousal emotions such as joy, which often overlap with neutral or surprise categories.

CHAPTER THREE

REQUIREMENTS AND REQUIREMENT ENGINEERING

This chapter outlines the functional and non-functional requirements that guided the development of the speech emotion recognition (SER) system, which operates on audio clips extracted from TV series dialogues. The project targets real-life conversational scenarios, characterized by spontaneous, noisy, and often overlapping speech—making it more challenging than traditional benchmark datasets. In order to maximize emotion recognition accuracy in such uncontrolled environments, the system integrates several advanced engineering components including speaker diarization using PyAnnote, transcription alignment using Whisper, short audio detection, noise trimming, and amplitude normalization.

Feature extraction pipelines are customized per model type—statistical acoustic features for machine learning models, spectrogram-based image representations for CNN architectures, time-series MFCC vectors for RNN-based models, and embedding-level input for state-of-the-art transformer-based models like WavLM and HuBERT.

By incorporating a modular design with clear data flow between preprocessing, feature engineering, and classification modules, the system ensures flexibility and scalability across multiple experimental configurations. This chapter defines both the functional and non-functional requirements that shape the development of the system. One of the core technologies used in this approach is the Mel-Frequency Cepstral Coefficients (MFCCs), which represent the short-time power spectrum of an audio signal on the Mel scale. Since their development in the 1980s, MFCCs have proven to be among the most effective features for speech recognition and audio analysis. The accurate representation and interpretation of speech sounds are critical for identifying emotional states.

In this project, MFCCs, along with delta, delta², pitch, chroma, jitter, shimmer, formants, and high-level statistical features from OpenSMILE, were used to capture emotional characteristics in speech, as they provide a robust and meaningful

representation of audio signals. Python-based libraries are utilized to load and process the audio data, transforming it into MFCC format for further analysis. Multiple dataset configurations were tested (e.g., all speakers, gender-based subsets, individual speakers). Best results were observed with the Joey-only subset, prompting its use in final modeling.

Initially, the dataset was examined across multiple speakers and genders. However, due to performance inconsistencies and class imbalance issues, the system ultimately focused on the Joey character, where emotional expression and data quality were most consistent.

The examination and design of sound features is a central element of this project. Despite their significance, such aspects have not been extensively studied within the context of software engineering. While some literature has addressed the inclusion of emotions during the requirements engineering phase, detailed studies in this domain remain limited.

This study is inspired by a prior work that explored emotional characterization within a software context. Building upon that foundation, the system developed here seeks to characterize emotions, moods, and affective states through practical examples such as joy, fear, and security. This section outlines the foundational requirements essential for project planning, addressing both technical and emotional dimensions of software design.

3.1. Functional Requirements

The system is designed to identify seven distinct emotional states (anger, disgust, fear, joy, neutral, sadness, and surprise) from conversational speech. It supports training and inference using multiple data representations including raw waveform audio, statistical acoustic features, spectrogram images, and transformer-based embeddings. Feature extraction is performed dynamically according to the selected model architecture, ensuring compatibility with machine learning (ML), deep learning (DL), and transformer-based (SOTA) pipelines.

A list of requirements and their associated risk, priority and difficulty is given in Table 3.1, detailing the work to be done to achieve the goals set in the original project proposal.

The overall system design is shown in Figure 3.1. Brief justifications of the choices made are given in the subsections below.

Table 3.1 Functional Requirement Analysis

Requirements Description	Risk	Priority	Difficulty
Understanding underlying theory	Medium	High	High
Build tools for pre-processing operations (silence trim, RMS, noise removal etc.)	High	High	Low
Preparation and augmentation of training sets (balancing, noise, pitch shift etc.)	Medium	High	Medium
Implementation of optimization algorithms	Medium	Medium	High
Segment-based preprocessing and analysis	Medium	Medium	Medium
Implementation of pairwise classification algorithms	High	High	High
Model evaluation pipeline with standard metrics	Medium	Medium	Medium
Building front-end for visualizing results	Low	Low	Medium
Transcription using Whisper	Medium	High	Medium
Speaker diarization using PyAnnote	High	High	Medium
Feature extraction per model (statistical, image-based, time-series)	Medium	High	Medium
Training models (ML, DL, SOTA, Hybrid)	Medium	High	High
Feature selection integration (Boruta, RFE, Autoencoder, etc.)	Medium	High	Medium
Multi-label classification framework	Medium	High	Medium
Evaluation metrics (accuracy, F1, confusion matrix)	Medium	Medium	Medium
Predict module for single/multiple test audio files (via Jupyter code)	Low	Medium	Medium

When Table 3.1 is examined, it becomes clear that each functionality has been analyzed and classified based on risk, priority, and difficulty. The prioritization in Table

3.1 was directly influenced by experimental results and implementation challenges. Tasks such as PyAnnote-based diarization and Whisper-based transcription were marked high risk due to memory constraints and GPU compatibility issues encountered in cloud environments (e.g., Google Colab). On the other hand, visual inference modules and augmentation pipelines were easier to develop but less impactful in model accuracy, thus marked with lower risk.

The first step in prioritizing development efforts was to define the core objective of the system: to receive audio input and, with the help of a pre-trained model, accurately identify the emotion conveyed within it. This identification primarily relies on sound characteristics and requires a high level of precision.

A critical component in achieving this goal is data preprocessing. Accurate and reliable emotion detection can only be achieved if the incoming, existing, and future datasets are thoroughly prepared. This includes noise removal, silence detection, normalization, and transformation into meaningful feature representations like MFCCs.

Moreover, the classification algorithms and real-time audio segmentation represent foundational features in the system architecture. These are essential not only for achieving functional completeness but also for ensuring runtime efficiency and scalability.

From a risk perspective, the proper implementation of classification logic and dataset preparation is pivotal. Any inconsistencies in these stages could result in faulty emotion labeling or misleading output.

In terms of difficulty, tasks such as theory understanding, algorithm optimization, and implementation of pairwise classification are expected to be more challenging and thus require careful planning and scheduling. These tasks should be prioritized early in the timeline to allow sufficient development and testing time.

The functional scope of the system includes modular preprocessing and inference pipelines tailored to diverse model types. For instance, audio inputs are cleaned through silence trimming, RMS thresholding, and noise reduction. Speaker-specific segmentation is applied using PyAnnote, and alignment between labels and utterances is validated using Whisper.

Feature extraction is performed based on the model family:

- ML models receive 1400+ statistical acoustic features (MFCC, pitch, jitter, formants, wavelet, etc.)
- CNN models use 128x128 Mel, MFCC, or RGB spectrogram images
- RNN/BiLSTM models consume padded time-series tensors ([T, 120])
- Transformer models like Wav2Vec2 and HuBERT use learned embeddings
- An interactive GUI was also developed to enable prediction from single or batch test audio files, supporting model-specific inference logic and automatic loading of scalers, label encoders, and feature selectors depending on the selected model.

During experimentation with the full MELD dataset and various speaker combinations, it was observed that accuracy significantly varied across speakers due to differences in recording quality, speech patterns, and background noise. As a result, a refined subset focusing solely on the character "Joey" was selected. This decision improved the reliability of emotion labeling and boosted overall model performance by reducing cross-speaker variability.

The system also supports dynamic inference from single audio files or entire test folders, with predicted emotions visualized in the interface. These functionalities ensure the system can scale across multiple testing scenarios while maintaining performance and interpretability.

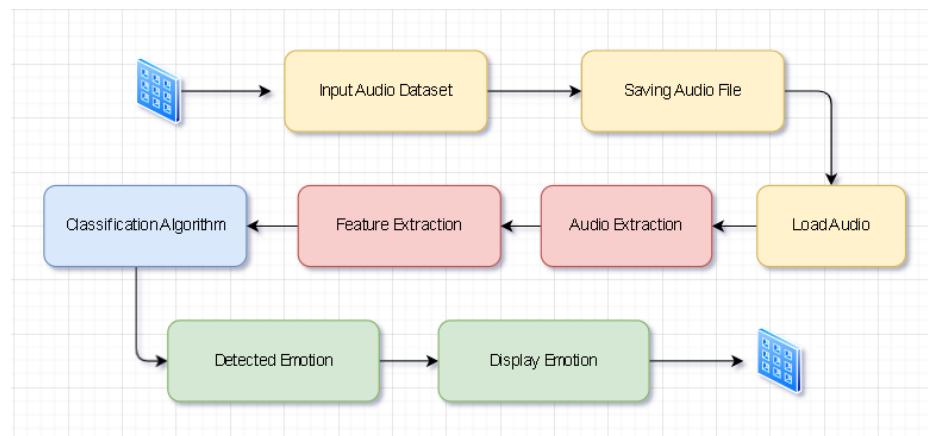


Figure 3.1 Schematic Flowchart of Functionality Implemented for Emotion Detection from TV Series or Audio Clips Data

The schematic flowchart (Figure 3.1) visually illustrates the main components of the system designed for emotion detection based on audio data obtained from TV series or sound clips. The workflow begins with the input of the audio dataset and continues with the saving and loading of the audio files. The system proceeds by extracting both audio and its features and then applies classification algorithms to detect emotions. Finally, the detected emotion is displayed via the graphical interface. Although Figure 3.1 presents a simplified overview of the functional pipeline, the actual system includes a rich set of preprocessing components, such as speaker diarization via PyAnnote, transcription alignment via Whisper, silence trimming, noise reduction, and data augmentation (e.g., pitch shift, background noise addition). These components are implemented modularly but are not fully shown in the figure.

3.1.1. Problem Statement

This project focuses on detecting emotional states from speech in audio clips extracted from TV series, specifically using the MELD (Multimodal EmotionLines Dataset) dataset. In real-world multimedia content such as conversational TV shows, emotional expression occurs in spontaneous, noisy, and overlapping dialogue conditions, making speech emotion recognition (SER) a complex task. In initial experiments, working with the entire dataset or multiple speakers yielded suboptimal results due to inter-speaker variability and noisy data segments. Therefore, a focused subset using only the speaker “Joey” was selected to ensure consistency and better classification performance during training and testing phases.

Most existing emotion-aware systems rely on either transcribed textual content or multimodal fusion (audio, text, and video modalities), which may not be feasible or reliable in many scenarios. This project aims to design and implement an audio-only emotion recognition system that can handle real-world challenges such as background noise, speaker overlap, and class imbalance. The system addresses these issues through advanced preprocessing steps, including speech transcription using Whisper and speaker diarization using PyAnnote, to ensure that only clean and speaker-consistent segments are used for training and prediction.

The goal is to build a robust and modular pipeline that preprocesses raw audio clips, extracts meaningful acoustic features (e.g., MFCCs, chroma, spectral features, embeddings), and classifies them into one of the seven predefined emotional states (anger, disgust, fear, joy, neutral, sadness, surprise) using machine learning and deep learning models. The final system includes a user-friendly graphical interface that enables real-time emotion prediction from single or batch audio files, automatically selecting the appropriate model, scaler, and feature pipeline for each configuration.

3.1.2. Feature Extraction Techniques for Speech Recognition

In speech emotion recognition, extracting representative and discriminative features from raw audio signals is critical for effective classification. This project integrates a wide range of acoustic feature extraction techniques, including statistical, spectral, time-series, image-based, and embedding-based methods. These features aim to capture both low-level acoustic patterns and high-level emotional cues from speech.

Among these features, Mel Frequency Cepstral Coefficients (MFCCs) are widely used due to their ability to model the human auditory system. They represent the short-time power spectrum of an audio signal using a Mel-scaled frequency axis, which reflects how humans perceive pitch. This makes MFCCs particularly effective in capturing the phonetic content of speech while remaining robust to variations in amplitude and recording conditions.

In this project, MFCCs are used as one of the primary low-level descriptors (LLDs) for feature extraction, along with their first and second derivatives (delta and delta-delta), resulting in a time series representation of shape [T, 120]. This representation is then used as input to deep learning architectures such as LSTM, BiLSTM, and GRU.

In addition to MFCC-based time series, other statistical and prosodic features such as zero-crossing rate (ZCR), root mean square energy (RMS), pitch, jitter, shimmer, harmonic-to-noise ratio (HNR), and formant frequencies were extracted using Librosa and Praat. Spectral features including spectral centroid, bandwidth, roll-off, contrast, and flatness were also computed. Wavelet and TEO-based energy representations were integrated for capturing non-linear dynamics in speech.

For convolutional neural network (CNN) models, image-based representations such as 128×128 Mel spectrograms, MFCC spectrograms, and RGB fusion images (MFCC = R, Delta = G, Delta² = B) were utilized. Transformer-based models such as Wav2Vec 2.0, HuBERT, and WavLM leveraged high-level audio embeddings extracted from pretrained models available on the Hugging Face platform.

This comprehensive and modular feature extraction pipeline enabled the system to support multiple model families (ML, DL, Transformer), ensured robust generalization, and allowed for dynamic adaptation during both training and inference.

3.1.3. Dataset Characteristics and Multi-Stage Preprocessing Pipeline

The primary dataset used in this study is the Multimodal EmotionLines Dataset (MELD), which is derived from TV dialogue segments in the *Friends* sitcom. MELD includes multimodal annotations across audio, text, and video modalities. However, in this project, only the audio modality was utilized to simulate real-life scenarios where only speech is available for emotion recognition (e.g., call centers, podcasts).

Each audio segment was first extracted from MP4 video clips and then passed through a multi-stage preprocessing pipeline. This pipeline included:

- **Silence trimming** using energy-based thresholding,
- **Noise reduction** using spectral subtraction and bandpass filters,
- **Amplitude normalization** to standardize loudness,
- **Fade-in/fade-out smoothing** to reduce abrupt signal boundaries,
- **Voice Activity Detection (VAD)** to eliminate non-speech frames,
- **Short/corrupted clip removal** based on RMS energy, ZCR, and duration constraints.

Additionally, to improve class balance and model generalization, data augmentation techniques such as pitch shifting, time-stretching, and background noise injection were

applied. All preprocessing steps were carefully verified using visual and quantitative inspection methods to ensure the emotional content of the utterances was preserved.

Finally, Whisper (OpenAI’s speech-to-text model) was used to generate high-quality transcriptions and align them with reference utterances in the MELD dataset. Segment-level semantic similarity was computed between the Whisper transcription and the original utterance text to validate correspondence. Misaligned or low-confidence segments (meta-confidence < 0.75) were flagged or excluded. Speaker diarization was performed using the PyAnnote toolkit, where the number of detected speakers in each clip was computed. Clips with more than one active speaker were either trimmed using the Whisper-aligned timestamps or completely discarded. This ensured mono-speaker consistency and improved label reliability during training. To facilitate alignment and debugging, standardized file naming (dia[Dialogue_ID]_utt[Utterance_ID].wav) was adopted and cross-validated with the corresponding CSV entries to verify proper mapping between audio files and emotion labels. This focus on a single speaker was not an initial constraint, but was later adopted after observing that models trained on all characters or speaker-mixed clips yielded suboptimal results in accuracy and consistency. Joey was selected due to his sufficient sample count and consistent speech pattern, enabling more stable model performance.

In addition, audio durations were compared across four independent sources — librosa, ffmpeg, cv2, and transcript metadata — to identify misaligned or corrupt samples. Files with major discrepancies (e.g., >0.4 seconds difference between sources) were excluded from the training data. This step was crucial for maintaining integrity in emotion labels.

3.1.4. Programming Language

The entire system was implemented using the Python programming language due to its extensive ecosystem, readability, and broad support for machine learning, deep learning, and signal processing libraries. Python offers strong compatibility with audio processing frameworks, ML/DL toolkits, and state-of-the-art transformer and pretrained models, making it an ideal choice for speech emotion recognition (SER) tasks.

Key reasons for using Python include:

- **Signal Processing:** Libraries such as librosa, scipy, and pydub were used for preprocessing operations like silence trimming, RMS and ZCR analysis, normalization, filtering, and feature extraction (MFCC, delta, chroma, pitch, spectral features).
- **Machine Learning & Deep Learning:** Scikit-learn was employed for classical ML models (e.g., SVM, Random Forest, KNN, XGBoost), while PyTorch and Keras were used to build and train deep learning architectures such as CNN, LSTM, BiLSTM, GRU, and MLP.
- **Self-Supervised and Transformer-Based Models:** State-of-the-art models such as Wav2Vec2, HuBERT, and WavLM were implemented using the Hugging Face transformers library. These models required fine-tuning and inference steps on GPU-enabled environments like Google Colab.
- **Speech Recognition and Diarization:** OpenAI's Whisper model was utilized for ASR (Automatic Speech Recognition) tasks, and pyannote-audio was used for speaker diarization to isolate Joey's voice clips and for speaker diarization to ensure mono-speaker consistency; segments with more than one speaker were either trimmed or discarded.
- **Feature Extraction Tools:** openSMILE was used externally (openSMILE was used externally by calling its executable through Python's subprocess interface, enabling the extraction of eGeMAPS (v01a) features in batch mode) to extract eGeMAPS features. Integration with Python ensured seamless automation.
- **Data Handling and Visualization:** Libraries like Pandas, Matplotlib, and Seaborn were used for dataset manipulation, evaluation reporting (classification report, confusion matrix), and visualization.

Model development and experimentation were primarily conducted using Jupyter Notebook environments, with Google Colab used for GPU-intensive training tasks (especially SOTA models and Whisper alignment), and local Windows environments used for preprocessing and GUI integration.

Although Jupyter was the main interface for development and testing, a PyQt5-based graphical user interface (GUI) was later implemented in Spyder to enable end-user interaction. This GUI allows the selection of models (ML, DL, or SOTA), loading of test audio files, and dynamic prediction with real-time results visualization. In conclusion, Python's flexibility and broad library ecosystem enabled the seamless integration of diverse processing modules into a unified and scalable emotion recognition pipeline.

3.1.5. Modelling Requirements: ML, DL, and SOTA Approaches

To fulfill the core objective of detecting emotions from speech, the system must incorporate modeling capabilities that can effectively handle various types of features and audio representations. This includes the integration of machine learning (ML), deep learning (DL), and state-of-the-art (SOTA) approaches.

The machine learning (ML) models are designed to process high-dimensional, structured numerical features extracted from audio, including MFCCs, delta and delta-delta coefficients, pitch, spectral centroid, zero-crossing rate (ZCR), shimmer, jitter, harmonic-to-noise ratio (HNR), formants, and prosodic cues. Classification algorithms such as Support Vector Machines (SVM), Random Forest (RF), K-Nearest Neighbors (KNN), Naive Bayes (NB), XGBoost, and Multi-layer Perceptron (MLP) are employed to learn patterns in these representations.

In parallel, Deep learning (DL) models are employed to process sequential and visual representations of speech, such as $[T, 120]$ shaped MFCC time series and 128×128 Mel spectrogram images. CNN architectures are used for learning from spectrograms, while recurrent neural networks (RNNs) including LSTM, BiLSTM, and GRU are used to model the temporal dynamics of audio signals.

Additionally, The system also integrates pretrained self-supervised transformer-based models, including Wav2Vec2.0, HuBERT, and WavLM, via the Hugging Face Transformers library. These models are used in two settings: (1) directly as end-to-end fine-tuned classifiers, and (2) as feature extractors whose embeddings are passed to lightweight classifiers such as MLPs. These SOTA models leverage large-scale unlabeled

speech corpora and improve performance especially in noisy and imbalanced real-world data.

The GUI interface implemented in the final stage allows users to dynamically choose among ML, DL, or SOTA models for inference, depending on the application scenario. This modularity enhances the system's usability and extensibility.

Each modeling approach contributes to the modularity and adaptability of the system, allowing experimentation with different levels of complexity and generalization. Consequently, the modeling layer is designed with high modularity, allowing seamless switching between algorithmic strategies (e.g., feature-based, image-based, embedding-based) depending on task complexity, data availability, and deployment needs.

3.1.6. System Specification/ Description

The system is fully implemented in Python due to its rich ecosystem for machine learning, deep learning, audio signal processing, and modular pipeline design. It leverages both local (Jupyter Notebook, Spyder) and cloud-based platforms (Google Colab) for different stages of development and execution. The following libraries and tools form the core technical stack:

- **Librosa & SciPy** – Used for low-level signal processing operations such as MFCC, delta, ZCR, spectral centroid, and Mel spectrogram extraction, along with noise reduction, filtering, and silence trimming.
- **OpenSMILE** – External tool (called via subprocess) used to extract prosodic and spectral features using the eGeMAPSv01a configuration, mainly for ML-based pipelines.
- **Parselmouth (Praat API)** – Used for voice quality measurements including jitter, shimmer, harmonic-to-noise ratio (HNR), and formant frequencies.
- **PyTorch & Keras** – Frameworks used for building and training deep learning models (CNN, LSTM, GRU, BiLSTM, MLP) on time-series and image-based audio features.

- **Hugging Face Transformers** – Provides access to pretrained transformer-based SOTA models (Wav2Vec2.0, HuBERT, WavLM), used both in feature extraction and end-to-end classification.
- **OpenAI Whisper** – Utilized for automatic speech recognition (ASR), enabling transcript-based validation of audio segments and matching with reference text.
- **pyannote-audio** – Applied for speaker diarization to detect multi-speaker segments and filter them based on confidence thresholds.
- **GUI (Spyder-based)** – A user interface was also implemented for real-time inference, allowing model selection (ML/DL/SOTA) and prediction on unseen audio files.

The entire system architecture follows a modular design, where each stage—preprocessing, feature extraction, and classification—is encapsulated in independent scripts and file structures. Shared directories (e.g., `train_splits/`, `test_splits/`, `features_and_models/`, `results/`) ensure reproducibility, organized experimentation, and scalable integration of new methods.

To facilitate alignment and debugging, standardized file naming (`dia[Dialogue_ID]_utt[Utterance_ID].wav`) was adopted and cross-validated with the associated metadata files. Feature vectors were scaled using consistent normalization (e.g., `StandardScaler`) before training. Data balancing strategies, including undersampling and augmentation, were applied to mitigate emotion class imbalance.

Finally, GPU acceleration available via Google Colab was leveraged for memory-intensive operations such as Whisper transcription, speaker diarization, and deep learning/SOTA model training, enabling efficient experimentation across diverse architectures.

3.2. Non-Functional Requirements

3.2.1. *User Requirements*

- The system must provide a user-friendly interface, enabling both single and batch-mode predictions for emotion recognition on audio files.
- It should support interpretability by presenting evaluation metrics (e.g., classification report, confusion matrix) in clear and visually accessible formats.
- The architecture should allow users to perform core functionalities—including preprocessing, feature extraction, model selection, and prediction—within the Jupyter Notebook environment, without requiring extensive programming knowledge.
- Output transparency is essential; the system must display both ground truth and predicted emotion labels for each test instance, facilitating easy error analysis and trust in model decisions.
- For users preferring graphical interaction, the optional GUI interface should support model selection (ML, DL, or SOTA), feature set selection, and audio input for inference, with visual display of results.

3.2.2. *Product Requirements*

Portability: The system is designed to be platform-independent. It can operate on any environment supporting Python and Jupyter Notebook, including cloud-based platforms such as Google Colab and local IDEs like Spyder.

Correctness: All components have been validated across multiple dataset partitions (train, development, and test). The system consistently produces reproducible results across various modeling strategies (ML, DL, and SOTA).

Modularity: The system follows a modular architecture where components such as preprocessing, feature extraction, model training, and prediction are decoupled. This design supports independent testing and seamless integration of new features or models without disrupting the entire workflow.

Extensibility: New feature extraction techniques, classifiers, or evaluation metrics can be added with minimal changes, thanks to well-defined input/output conventions and reusable code structures.

3.2.3. System Quality Characteristics

Non-functional requirements, also known as system quality attributes, are essential to ensuring the reliability, usability, and sustainability of the overall system. These characteristics are categorized into two main dimensions: execution-related and evolution-related attributes.

- **Execution-Related Qualities:**

- **Usability:** The system interface—implemented through Jupyter Notebooks and a Spyder-based GUI—is designed to be accessible and intuitive for both technical and non-technical users. It enables seamless interaction for single-file or batch prediction workflows.
- **Security & Privacy:** Although the dataset used (MELD) is public, the system architecture is built to ensure secure handling of audio and emotion data. This is particularly important in real-world deployments involving sensitive emotional content or user-generated input.

- **Evolution-Related Qualities:**

- **Scalability:** The modular design and standardized data pipelines (e.g., CSV metadata, organized directories, reusable models) allow the system to scale with increased dataset size, model complexity, or new emotion classes without major restructuring.
- **Maintainability:** Code components are well-documented and decoupled into logical modules, supporting ease of debugging, component replacement, and future enhancements (e.g., integrating new feature extraction techniques or model types).
- **Testability:** The system's architecture supports both unit and end-to-end testing. Evaluation metrics (e.g., classification reports, confusion

matrices) are systematically generated and stored for verification, making the pipeline auditable and reproducible.

3.2.4. Emotional Goals as a Distinct Category

This project focuses on detecting emotions from TV series audio clips by developing a speech-based emotion recognition system using Python. The system extracts rich acoustic features from each utterance and classifies emotional states using a variety of machine learning, deep learning, and state-of-the-art (SOTA) architectures. To support model interpretability, visual outputs such as confusion matrices and loss/accuracy plots are included in the implementation.

Beyond the technical implementation, this study acknowledges the importance of emotional expectations in software systems—commonly referred to as emotional goals. Unlike traditional quality attributes such as performance or usability, emotional goals reflect how users feel during their interaction with a system. These goals are inherently abstract, subjective, and context-dependent, making them difficult to quantify using conventional requirement engineering frameworks (e.g., IEEE 1998).

Although this project does not aim to redefine requirement engineering paradigms, it emphasizes the value of explicitly considering emotional goals in the design and development of affective systems. In applications such as emotion recognition, anticipating the user's affective experience can improve system acceptance, empathy, and trust.

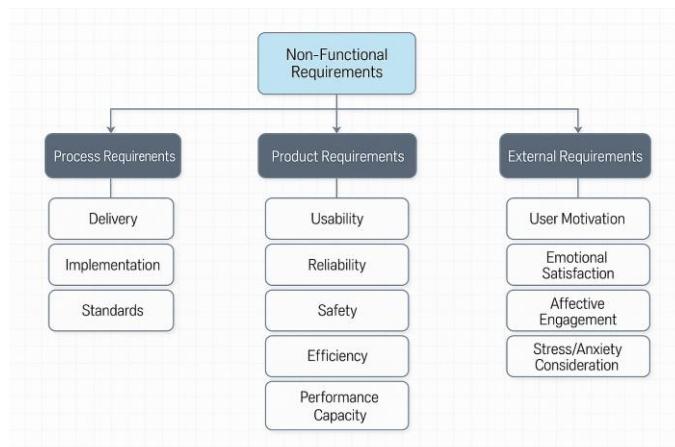


Figure 3.2 Schematic of Classification of Non-Functional Requirements

Considering all of the above, Figure 3.2 categorizes non-functional requirements under three main headings: Process Requirements, Product Requirements, and External Requirements. Within this framework, it is important to recognize that emotional goals—which reflect the users' affective expectations—do not neatly fit into any of these traditional categories yet significantly influence the overall system experience.

Within this framework:

- **Process Requirements** emphasize delivery standards and implementation protocols. For affective systems, the process must also incorporate mechanisms to validate emotional fidelity—ensuring that the system correctly interprets and preserves users' emotional content during preprocessing and prediction stages.
- **Product Requirements** typically include attributes such as usability, reliability, and efficiency. However, in an emotion-aware system, usability also encompasses the ability of the interface to respond empathetically, while efficiency extends to how fast and accurately the system detects emotional shifts in the input speech.
- **External Requirements**, including cultural, ethical, and psychological considerations, gain added importance in systems handling emotional metadata. Issues like emotional bias in models, the fairness of predictions across demographic groups, and the safe handling of affective data must be considered.

In conclusion, emotional goals may not neatly fall into any single non-functional category, yet they influence all dimensions of the system experience. As such, they should be regarded as a distinct layer of design consideration—especially in affective computing systems where emotion is both input and output.

CHAPTER FOUR

DESIGN

4.1. Architectural View

This section illustrates the overall architectural view of the developed Speech Emotion Recognition (SER) system, detailing how the raw audio input is processed, transformed, and ultimately classified into emotion categories. The entire pipeline, depicted in Figure 4.1, is designed to simulate real-life emotion recognition from TV series dialogue.

The architecture begins with a speech signal, which is extracted from video clips (.mp4) after applying comprehensive data cleaning and validation checks. These include matching each video with its corresponding metadata (.csv), verifying the presence and duration of speech using multiple tools (e.g., ffprobe, cv2, librosa), removing samples shorter than 0.5 seconds or with inconsistent timestamps, and filtering based on target speaker (“Joey”) and gender attributes. Additionally, ASR transcription (via Whisper) and speaker count estimation (via PyAnnote) were used to further clean and refine the dataset. All valid and high-confidence audio segments were then converted to mono 16kHz WAV format for further processing.

The next stage is Preprocessing, where noise is removed, silence is trimmed, and the signal is normalized using RMS-based scaling. Fixed-length padding is applied to ensure temporal consistency across inputs. To address class imbalance and overfitting, data augmentation methods such as pitch shifting, noise injection, and time-stretching were applied.

Once the audio is clean and balanced, feature extraction is performed. For classical ML and MLP models, rich handcrafted features were extracted using tools like Librosa, OpenSMILE, Parselmouth (Praat), SciPy, and Hugging Face Transformers. These include MFCCs, spectral features, prosodic attributes (e.g., pitch, jitter, shimmer), and high-dimensional transformer-based embeddings.

These features are stored and used by the Emotion Recognition Engine, which encompasses various model types:

- Machine Learning models such as SVM, Random Forest, XGBoost, k-NN, Gradient Boosting, Voting, and Bagging.

- Deep Learning models including CNNs trained on Mel spectrograms, MFCC and RGB spectrograms, BiLSTM, and MLP architectures implemented in both PyTorch and Keras.
- SOTA models, both feature-based (e.g., Wav2Vec2 + MLP) and end-to-end transformer-based models (e.g., WavLM, HuBERT, Wav2Vec2) were applied using the Hugging Face and torchaudio pipelines.

For ML and MLP models, 11 different feature selection strategies were applied (e.g., Boruta, SelectKBest, RFE, Autoencoder, RandomForest importance). Undersampling was also performed using PCA followed by KMeans clustering to balance dominant classes (e.g., male_neutral).

During the testing phase, unseen audio files undergo the same preprocessing and feature extraction pipeline. Extracted features are passed into trained models, which perform emotion classification, outputting emotion categories such as *anger, joy, sadness, etc.* Finally, performance metrics like classification reports and confusion matrices are generated to assess accuracy and interpretability.

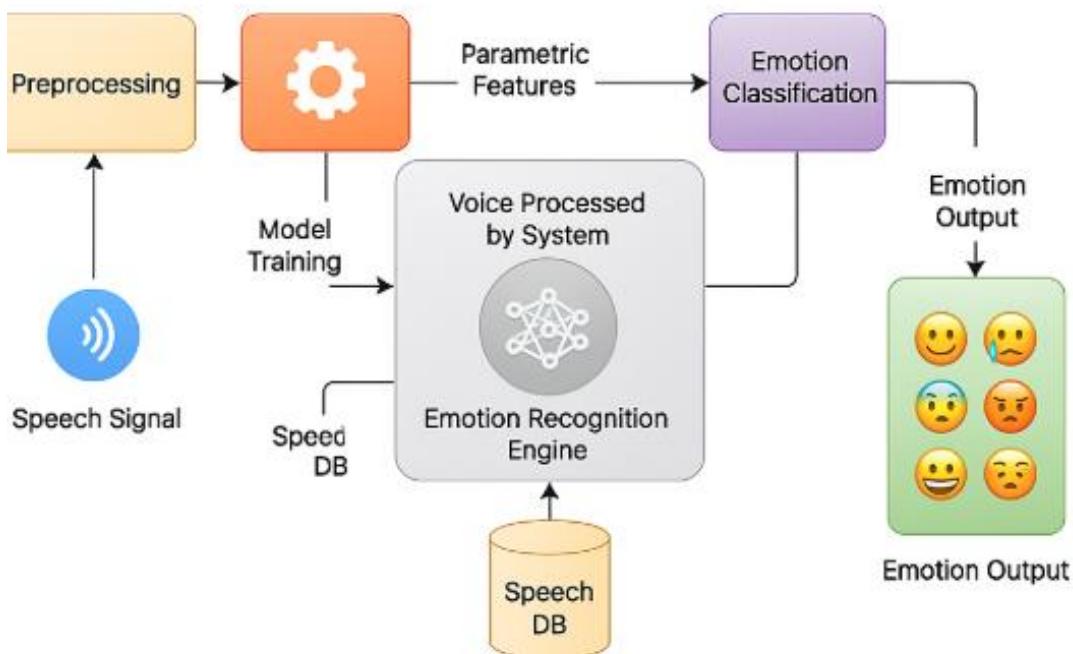


Figure 4.1 Architectural View of the Speech Emotion Recognition System

4.2. Database Design/ER Diagram

In this section, the overall workflow of the Speech Emotion Recognition (SER) system is illustrated, covering all critical stages from raw data ingestion to emotion prediction. As shown in Figure 4.2.1, the process begins with a series of validation and preprocessing steps, including matching video files with metadata (CSV), checking duration consistency, trimming silence, and applying augmentation techniques such as pitch shifting, noise addition, and time stretching to balance the dataset.

Once cleaned and filtered, the audio files are converted to 16kHz mono WAV format and passed through three different modeling pipelines:

- For Machine Learning (ML) models, handcrafted features are extracted using libraries such as Librosa, Parselmouth, OpenSMILE, SciPy, and others. These features include MFCCs, pitch, jitter, shimmer, spectral properties, prosodic attributes, and time-domain statistics. The final feature vector (1480+ dimensions) is then reduced using 11 feature selection strategies (e.g., Boruta, Autoencoder, RFE).
- For Deep Learning (DL) models, time-series representations (MFCC + deltas) or image-based features (Mel Spectrogram, MFCC Spectrogram, RGB fusion) are used directly as input to CNN, BiLSTM, GRU, RNN, and MLP architectures.
- For SOTA transformer-based models, either raw audio is directly passed to pretrained models (e.g., Wav2Vec2.0, WavLM, HuBERT) in an end-to-end fashion, or their high-level embeddings are extracted and combined with a classification head (e.g., MLP).

During the test phase, unseen audio samples are processed using the same pipeline and passed into the corresponding trained model. The system then outputs the predicted emotion category, supported with performance metrics such as accuracy, classification reports, and confusion matrices for interpretability and validation.

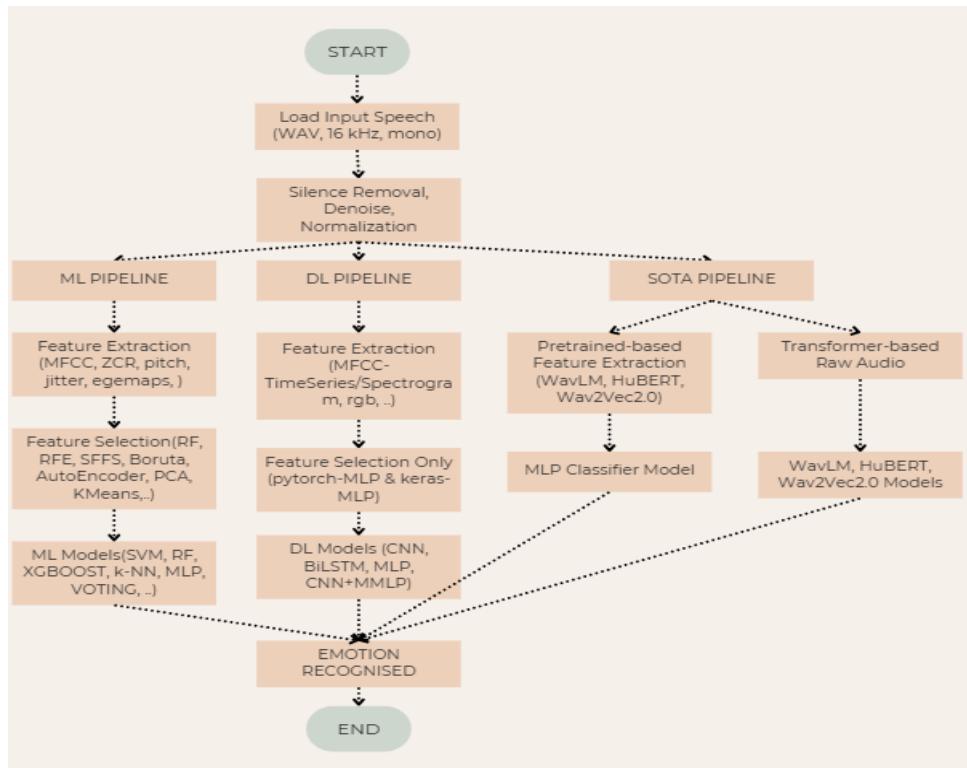


Figure 4.2.1 System and Database Design

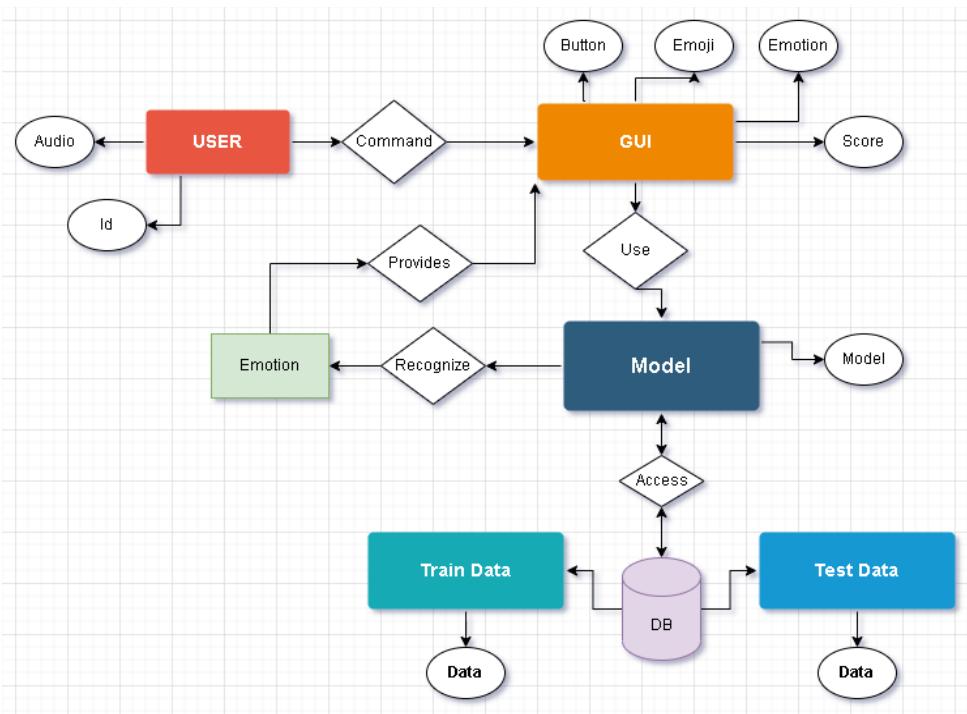


Figure 4.2.2 ER Diagram

The Entity-Relationship diagram illustrates the interaction between system components, particularly the user, graphical interface, model, and datasets. Although a formal database is not used, the system relies on structured CSV files and preprocessed WAV audio data. The user initiates emotion recognition through a GUI, which communicates with trained models. These models access training and test datasets stored as CSVs and audio files, producing an emotion label as output. The output is then visualized through the GUI. This diagram effectively captures the logical flow and data dependencies within the Speech Emotion Recognition system.

Since the system is structured on two bases—training and testing—the specific steps each of these datasets undergo can be observed in Figure 4.2.3.

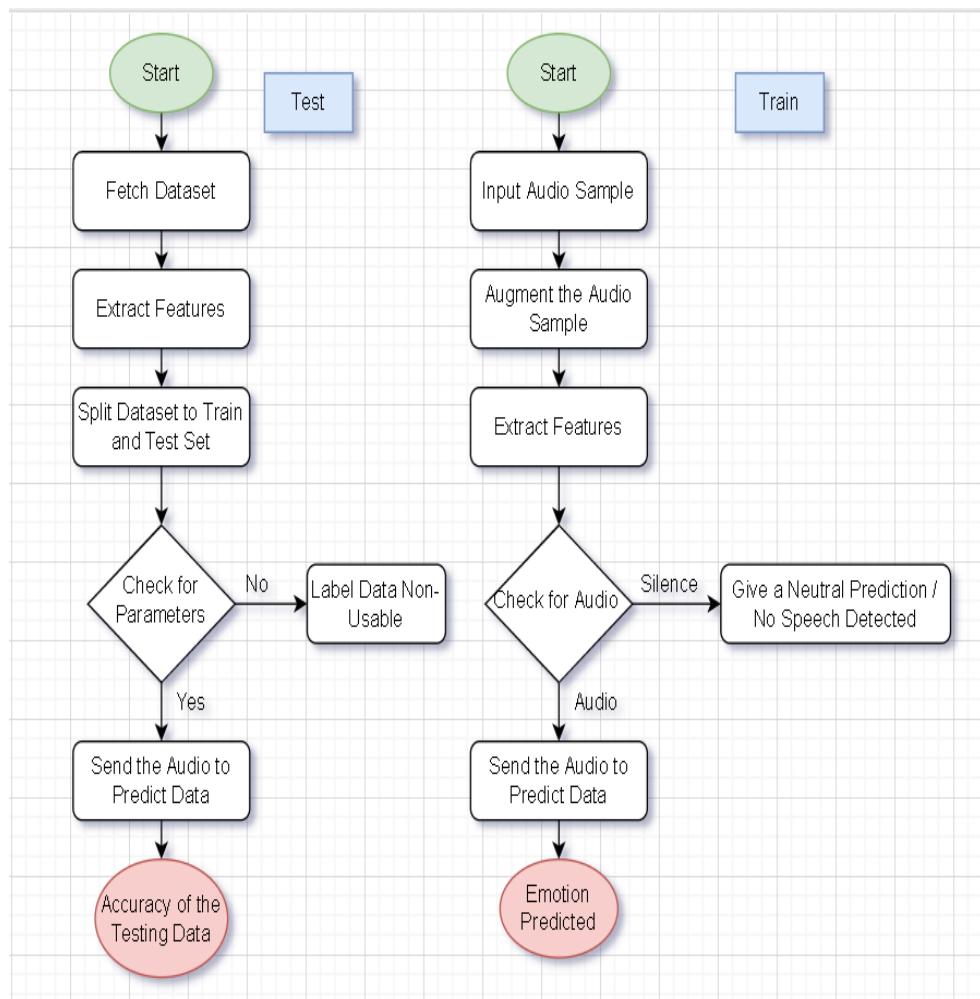


Figure 4.2.3 Architectural View

The overall system is divided into two major pipelines: training and testing. For training, audio samples are first augmented using techniques such as pitch shift or noise addition. Features are then extracted from the enhanced samples, and a silence detection step is performed. Only voiced samples proceed to emotion modeling. In the testing pipeline, the system fetches the audio dataset and applies the same feature extraction process. After checking for usability (e.g., valid labels, non-silent segments), valid samples are fed into the trained model to obtain predictions. If the sample is detected as silence or fails parameter checks, it is either labeled as non-useable or assigned a neutral emotion class. This architectural view clearly distinguishes the processes for training and inference while ensuring consistency in preprocessing and prediction flow.

4.3. UML Class Diagram

The UML Class Diagram in Figure 4.3 illustrates the core class structures and their interactions within the Speech Emotion Recognition (SER) system. Five primary classes were defined based on system responsibilities: User, GUI, Model, train_data, and test_data.

- The User class handles user interactions such as recording audio, viewing prediction results (e.g., F1-score, text), and initiating commands via the GUI.
- The GUI class serves as the front-end interface. It is responsible for rendering the predicted emotion, corresponding emoji visuals, and performance metrics (e.g., F1-score) through user-initiated actions.
- The Model class encapsulates all core functionalities, including data preprocessing, feature extraction, audio recognition, emotion prediction, and model evaluation.
- The train_data and test_data classes represent structured inputs for supervised learning tasks, supplying x (features) and y (labels) as needed for model training or inference.

Each class maintains a clear role and interacts with one another through well-defined functions and data dependencies. This structure supports modularity, extensibility, and ease of debugging during development.

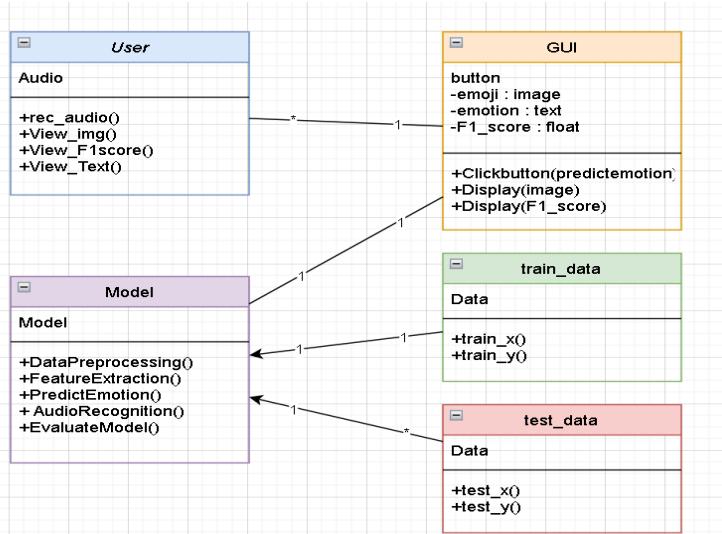


Figure 4.3 UML Class Diagram

4.4. UI Design

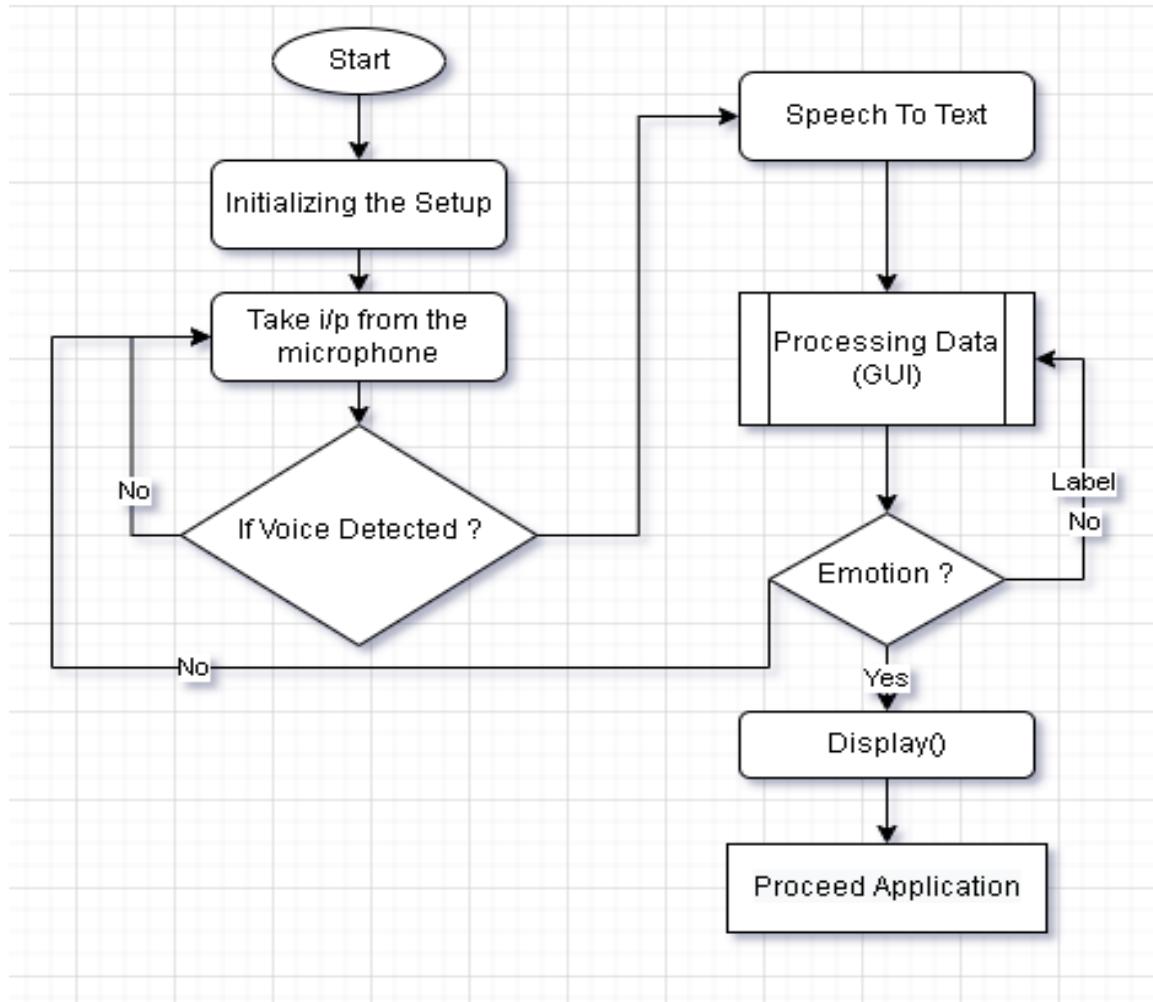


Figure 4.4 UI Design

The UI Design flow in Figure 4.4 outlines the process that occurs on the client side of the Speech Emotion Recognition system. The process begins with setting up the system and capturing input via the user's microphone. Upon detecting voice activity, the speech is converted to text and passed to the GUI layer for further processing.

If an emotion is recognized, the GUI displays relevant outputs including emotion labels and visual indicators such as emojis or performance scores. Otherwise, the system either waits for further input or returns a neutral output. This ensures that the application remains responsive and context-aware throughout user interaction.

4.5. Use Cases Diagram

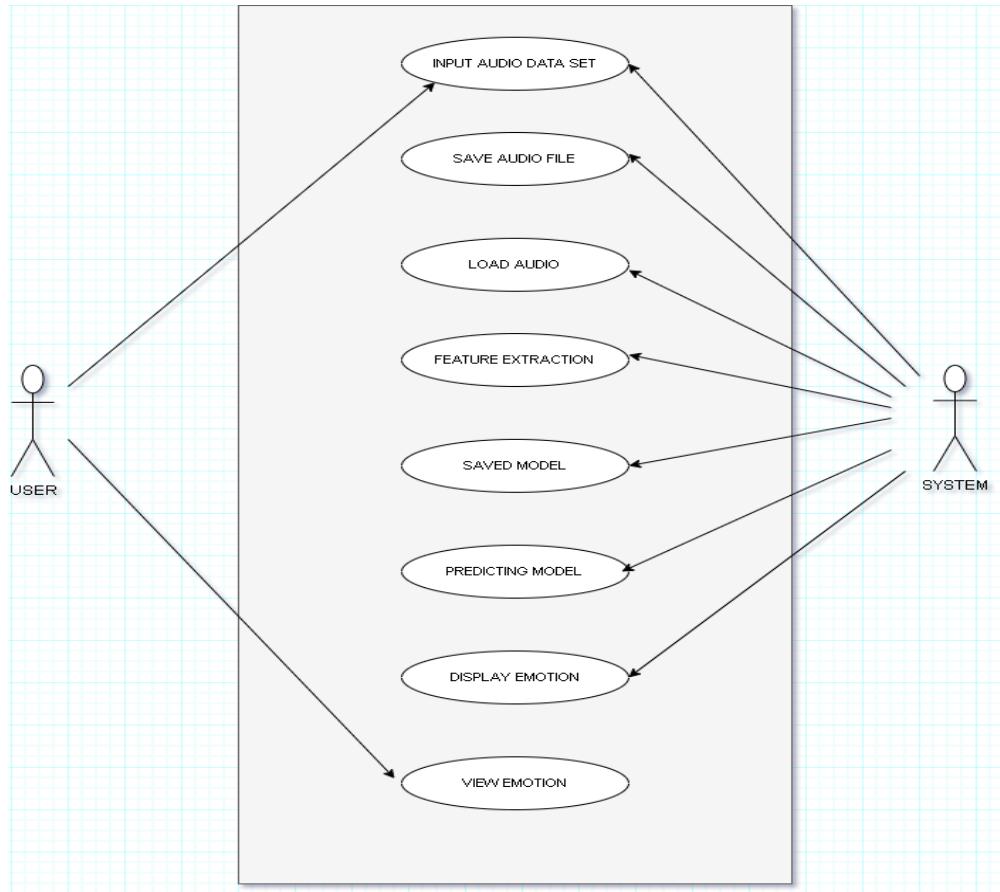


Figure 4.5 Use Cases Diagram

Figure 4.5 illustrates the Use Case Diagram for the Speech Emotion Recognition (SER) system, capturing the primary interactions between the user and the system. Two actors are defined:

- **User (Left):** Responsible for initiating actions such as providing audio input, starting emotion analysis, and viewing results.
- **System (Right):** Performs internal tasks in response to user actions, including audio handling, feature extraction, model prediction, and result display.

Each use case encapsulates a system function that contributes to the overall workflow. The diagram demonstrates how the user can input an audio dataset, trigger feature extraction, run predictions using a pre-trained model, and visualize the detected emotional state.

4.6. Sequence Diagram

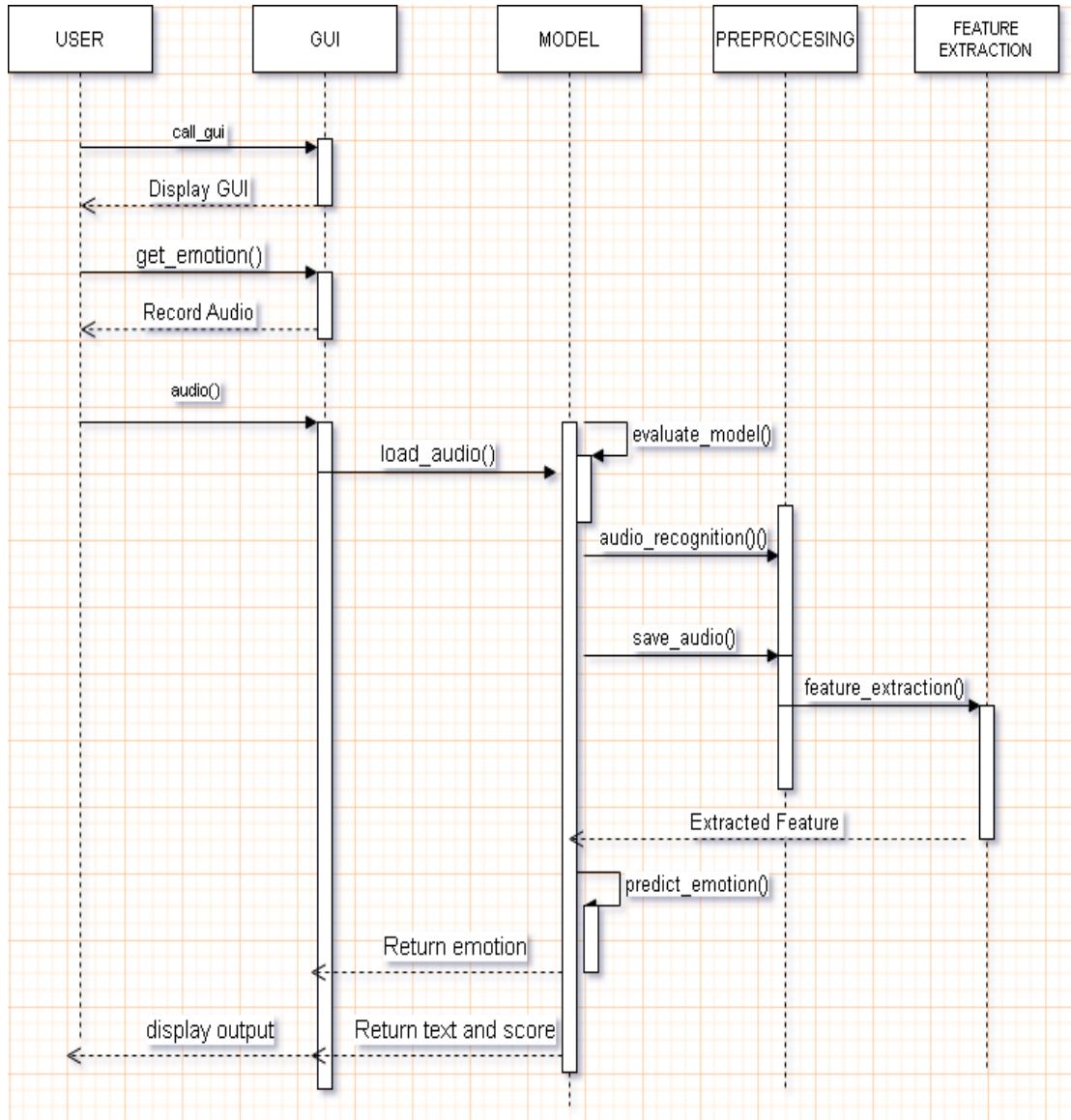


Figure 4.6 Sequence Diagram

The Sequence Diagram illustrates the order in which all system operations are executed, along with the specific layers responsible for each task. Additionally, it provides insights into the timing and duration of these operations. This diagram offers a comprehensive view of how data flows and is processed at each stage, from initial data entry to emotion F-score analysis.

4.7. Activity Diagram

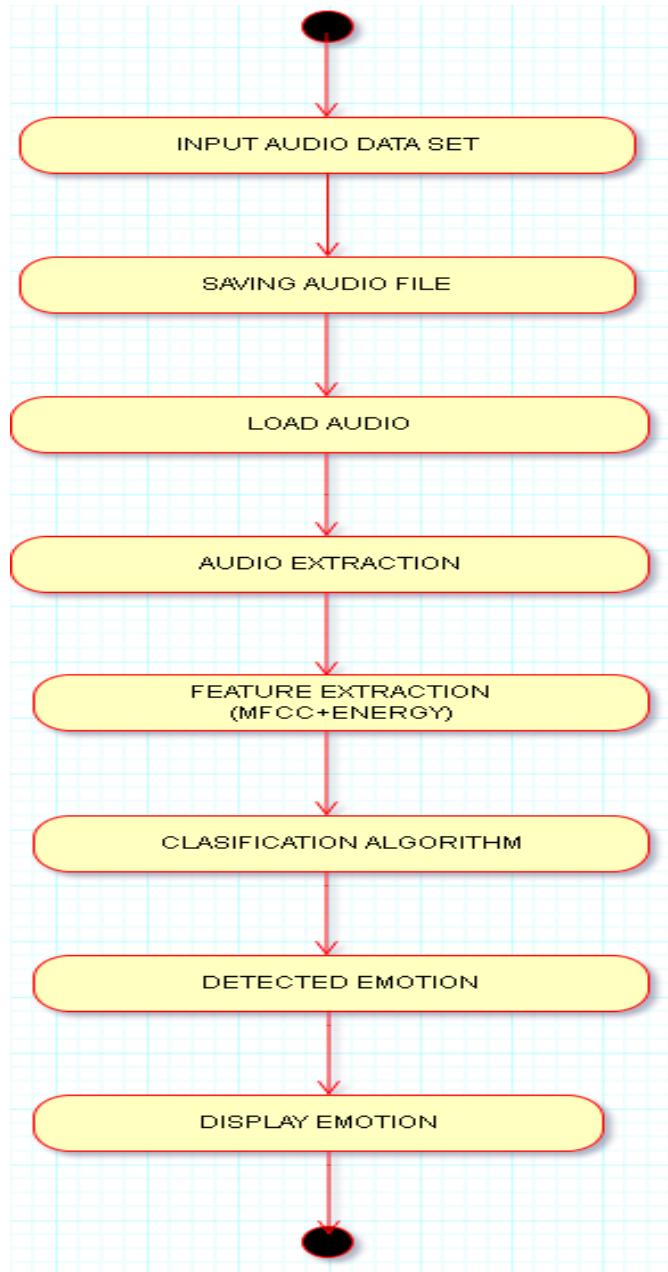


Figure 4.7 Activity Diagram

The Activity Diagram represents the system's behavior in terms of workflow. Conditional and parallel actions are illustrated in a sequential manner, providing a clear view of the system's operational logic. These actions are organized in the order in which they occur, starting from data entry and continuing through to the emotion display stage.

4.8. Deployment Diagram

The Deployment Diagram is a structural diagram that presents an overview of the modeled system by illustrating the connections and distribution relationships between components. In this diagram, the working environment of the system is defined, and communication between various components is clearly depicted.

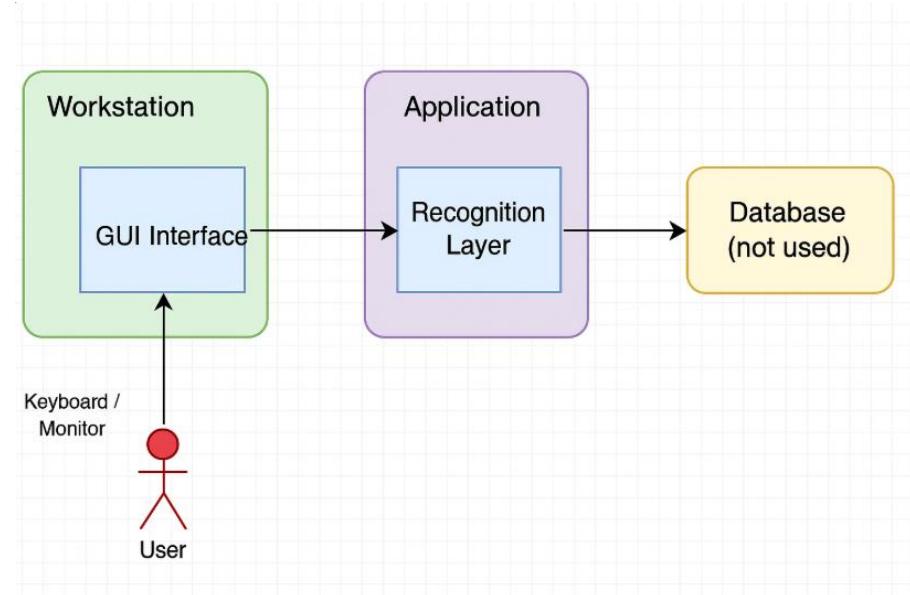


Figure 4.8 Deployment Diagram

Figure 4.8 illustrates the Deployment Diagram of the proposed Speech Emotion Recognition (SER) system. Unlike traditional distributed systems, this project is fully executed on the user's local machine (Workstation) and does not depend on any external database servers.

The deployment includes the following key components:

- Workstation Node: Represents the user's environment where the Graphical User Interface (GUI) is launched. The GUI manages all user interactions including audio file input, model selection, and displaying predicted emotions.
- Application Layer: Embedded within the same workstation, this layer includes the Recognition Module, which performs feature extraction, model loading, emotion classification, and result visualization.

- Although a Database Server component is visually represented in the diagram, it is not implemented in this project. Instead, all essential data (models, scalers, encoders, etc.) are stored and accessed as local files (e.g., .pkl, .pt, .h5) on the local file system.
- Optional components such as log files can be used for tracking predictions or debugging purposes, and these are also managed locally.
- The system operates in a standalone and offline mode, requiring no internet or networked database connectivity. The only exception is during the optional usage of pre-trained models (e.g., via Hugging Face), which may require internet access for initial downloads.

CHAPTER FIVE

IMPLEMENTATION

Emotion Recognition in Conversation (ERC) systems aim to automatically identify human emotions from spoken interactions, which is a critical component in developing empathetic, human-centric artificial intelligence. In this study, we specifically focus on audio-based emotion detection using dialogue segments extracted from television series, adopting the MELD (Multimodal EmotionLines Dataset) as the core dataset for our analysis and experiments.

One of the primary motivations behind selecting TV series dialogues, especially from a sitcom like *Friends*, is that they simulate real-life, emotionally rich, and spontaneous conversations far more authentically than traditional scripted datasets. Unlike acted speech datasets, MELD offers natural prosody, interruptions, overlapping speech, and varying background conditions—factors that significantly increase the complexity but also the realism of the problem space. This makes MELD a challenging yet meaningful benchmark for developing practical emotion recognition systems. Beyond its multimodal richness, MELD comprises over 13,000 utterances across training, development, and test sets, spoken by six main characters and various supporting speakers. Each utterance is labeled with one of seven emotions: anger, disgust, fear, joy, sadness, surprise, or neutral. As part of our analysis, we performed extensive data profiling on MELD—including utterance length statistics, word diversity, speaker-emotion distributions, and temporal emotion trends across dialogues—to better understand the dataset's structure and challenges. These analyses revealed significant class imbalance, with “neutral” being heavily overrepresented, and non-uniform distribution of emotions across speakers.

Although MELD is a multimodal dataset that provides aligned audio, text, and visual data, this project purposefully concentrates solely on the speech (audio) modality. The rationale behind this decision lies both in resource constraints and in the desire to deeply understand and maximize performance in one modality before expanding to others. Through extensive experimentation on the audio modality, our models were able to

achieve up to 65% accuracy, which is considered promising given the challenges of working with spontaneous, noisy real-world data.

However, working with MELD also exposed several limitations and inconsistencies that are well-documented in existing literature. For example, MELD’s emotion labels are annotated considering the utterance as a whole—its textual content, vocal tone, and facial expression. This leads to potential mismatches when models are trained on a single modality. In some cases, the audio clip does not perfectly align with the transcript in the CSV file (e.g., the transcript may say “*Yes. I know*”, while the audio says “*Yes. I know, but I’m not going*”). We also empirically verified some of these inconsistencies, such as mismatched transcripts and truncated audio clips, using alignment checks, audio duration comparison, and Whisper-based transcription validation. Similarly, the video frames may show emotional cues (like a smile) that contradict the labeled emotion (e.g., an utterance labeled as *sad* may show a smiling face), and often multiple characters appear in the same frame, making speaker-face matching ambiguous. Furthermore, many video clips lack detectable faces, introducing additional complexity in any visual emotion analysis.

From a researcher’s perspective, these inconsistencies are not only common but expected in multimodal datasets. Prior studies using MELD have also highlighted these issues. For instance, Hazarika et al. (2018) and Poria et al. (2019) mention that perfect modality alignment is not guaranteed and propose various techniques to mitigate misalignment.

Despite these challenges, this project successfully addresses them in the audio domain by applying meticulous preprocessing (noise reduction, silence trimming, normalization) and data augmentation (e.g., pitch shift, time stretch, white noise). Additionally, to counter class imbalance—particularly in underrepresented emotions such as *fear*, *disgust*, and *sadness*—targeted augmentation strategies were deployed. In addition to these textual inconsistencies, our exploratory analysis involved visualizing word distributions, utterance length histograms, and emotion shifts across dialogue timelines. These visual tools helped identify noise and guided decisions in filtering, normalization, and augmentation steps.

Although the current focus is on the audio modality, the design of this project is forward-compatible with future multimodal fusion approaches, where audio, text, and

visual features can be integrated to achieve greater performance. However, given the scope, timeline, and resource limitations of a single-researcher undergraduate thesis, the decision to begin with a deep dive into the speech modality is both justified and strategic. Initially, our experiments involved using all six main characters from the dataset. However, classification accuracy remained relatively low, and emotional tone varied significantly across speakers. As a result, we strategically filtered the dataset to include only one speaker—Joey—allowing more consistent vocal patterns, prosody, and timbre characteristics. This speaker-specific focus led to improved classification performance and more stable feature representation, forming a key turning point in our pipeline design.

In the following sections, we detail the entire implementation pipeline: from dataset description, preprocessing, and feature engineering to model training, evaluation, and discussion of results, culminating in a structured, modular, and reproducible system for speech-based emotion recognition from TV series dialogues—particularly filtered to Joey’s utterances to ensure consistent acoustic representation.

5.1. Datasets Description

In this study, the primary dataset used is MELD (Multimodal EmotionLines Dataset), which contains audio, text, and visual data extracted from the Friends TV series. The main motivation behind using this dataset is its realistic and context-rich conversations that closely simulate real-life interpersonal communication, unlike many controlled datasets with scripted dialogues. MELD includes multi-party conversations with emotional annotations, making it particularly valuable for developing models aimed at conversational AI systems. The dataset comprises over 13,000 utterances in total, spoken by six main characters (Joey, Ross, Chandler, Rachel, Monica, Phoebe) and several secondary speakers. Each utterance is annotated with one of seven primary emotions (anger, disgust, fear, joy, sadness, surprise, neutral), making it a valuable benchmark for evaluating emotion recognition systems in real-world, emotionally dynamic dialogues.

The dataset is split into training, development, and testing subsets, and each utterance is annotated with both an emotion (e.g., anger, sadness, joy, etc.) and a sentiment (positive, neutral, negative).

The inclusion of gender and speaker metadata further enhances its potential for granular analysis. Prior to model development, extensive exploratory analysis was performed on MELD to better understand the structure and limitations of the dataset. These analyses included: unique word distributions, utterance length statistics, dialogue-emotion mappings, speaker-emotion heatmaps, temporal emotion trends, and speaker-specific class imbalance. Visual tools such as bar charts, word clouds, heatmaps, and histograms guided strategic decisions throughout data preprocessing and model design.

In this project:

- The MELD dataset has been thoroughly cleaned and preprocessed.
- Audio segments are extracted from .mp4 video clips using ffmpeg.
- Gender metadata is preserved in the dataset to support future analysis of gender-emotion correlations. Initially, gender was fused into emotion labels (e.g., “male_joy”, “female_anger”) for exploratory analysis, but this was later abandoned to avoid inflating the number of emotion classes and to maintain consistency in speaker-specific modeling.
- Augmentation techniques are used to balance emotion classes with lower frequency.
- The dataset was filtered to include only utterances spoken by the character Joey. This decision was based on experimental results showing significantly improved classification performance when focusing on a single speaker. Joey was selected due to having the highest number of emotion-labeled utterances among all characters, with relatively balanced distribution across emotion categories. Focusing on a single voice also helped standardize prosodic and acoustic variability, which improved model consistency and generalization. This ensured consistency in speaker identity, prosody, and vocal traits, and allowed focused modeling on a single-speaker subset.

To improve the dataset's integrity and ensure alignment between audio and transcript data, several cleaning steps were applied. These included:

- Removing audio files shorter than 0.5 seconds

- Discarding clips where the duration difference between the actual audio file and CSV timing exceeded 0.4 seconds
- Verifying transcript and speech alignment via Whisper
- Removing clips with no detectable speech or missing audio

This preprocessing ensures that the dataset is well-structured for training both traditional machine learning and deep learning models, focusing initially on speech-only emotion recognition with plans for future multimodal expansion. Although only audio modality was used in this study, the cleaned and preprocessed dataset has been structured in a way that remains compatible with future experiments involving text and visual modalities. All cleaning, filtering, speaker-based slicing, and file-matching operations were logged and validated via automated scripts.

5.1.1. MELD

The Multimodal EmotionLines Dataset (MELD) is a comprehensive corpus derived from the TV series *Friends*, designed specifically for emotion recognition tasks in multi-party conversations (Poria et al., 2019). Unlike scripted or controlled datasets such as RAVDESS or IEMOCAP, MELD offers a more realistic simulation of natural dialogue, making it ideal for training systems that aim to function in real-world scenarios. In this project, MELD was selected due to its emotionally rich content and the availability of audio modality, which aligns with the study’s mono-modal speech emotion recognition (SER) objective. Unlike many traditional datasets, MELD reflects a diverse range of spontaneous expressions, overlapping speech, and real-life dialogue patterns across multiple speakers.

MELD provides synchronized audio, textual transcripts, and video clips for over 13,000 utterances across more than 1,400 dialogues, covering a wide spectrum of seven emotion labels:

- Anger, Disgust, Sadness, Joy, Neutral, Surprise, Fear

As shown in Figure 5.1, the distribution of these emotion classes is highly imbalanced, with the “neutral” category significantly more frequent than others across all dataset

splits. In the training set specifically (Figure 5.2), “neutral” samples outnumber minority emotions like “fear” or “disgust” by more than a factor of ten. This imbalance necessitated augmentation and class balancing strategies discussed in later sections.

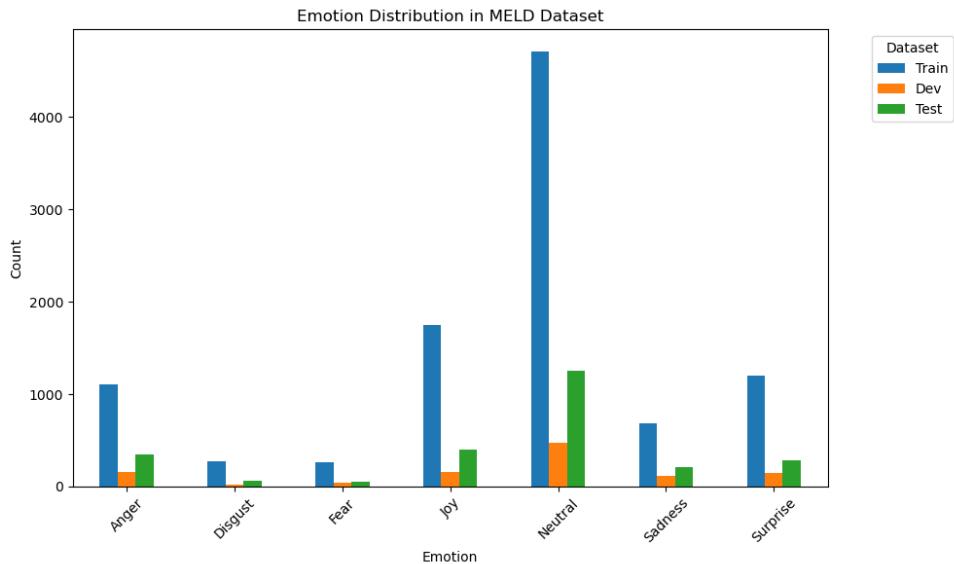


Figure 5.1 Emotion Distribution in MELD Dataset

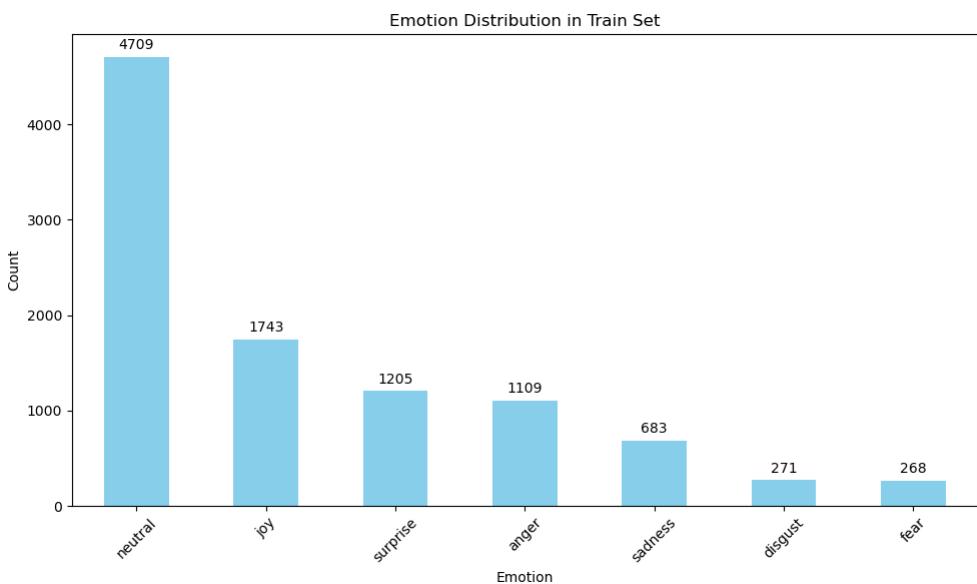


Figure 5.2 Emotion Distribution in Train Set

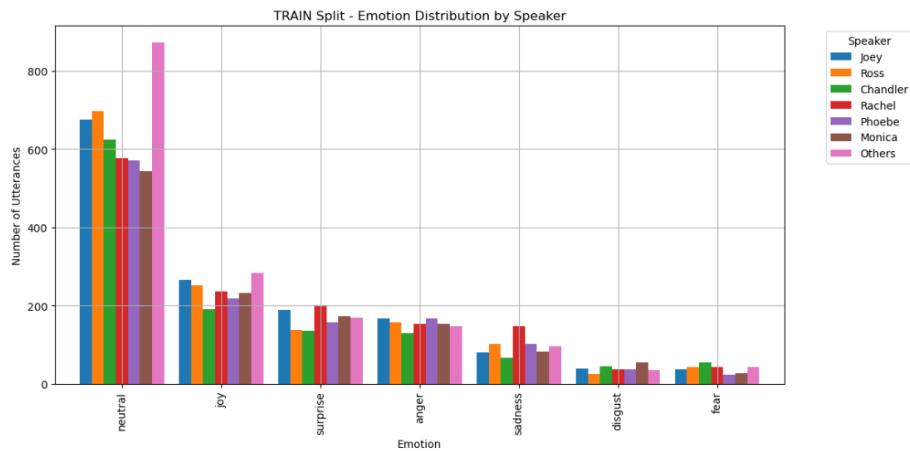


Figure 5.3 Emotion Distribution by Speaker in Train Set

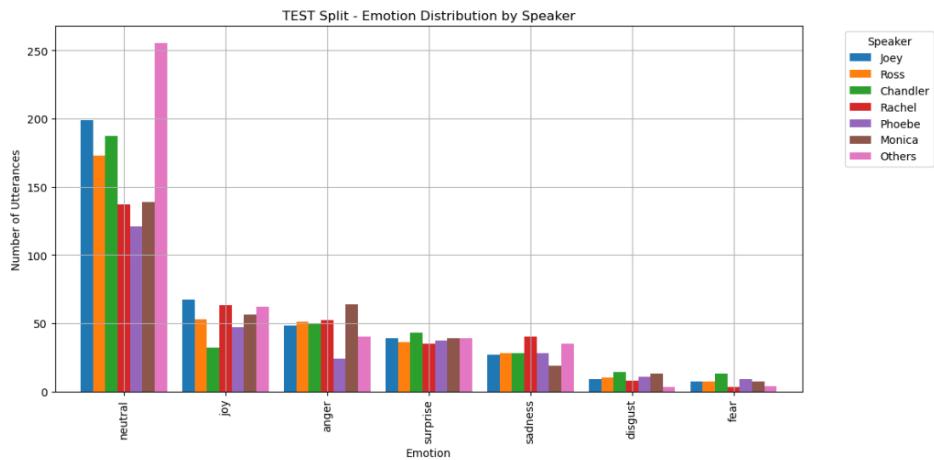


Figure 5.4 Emotion Distribution by Speaker in Test Set

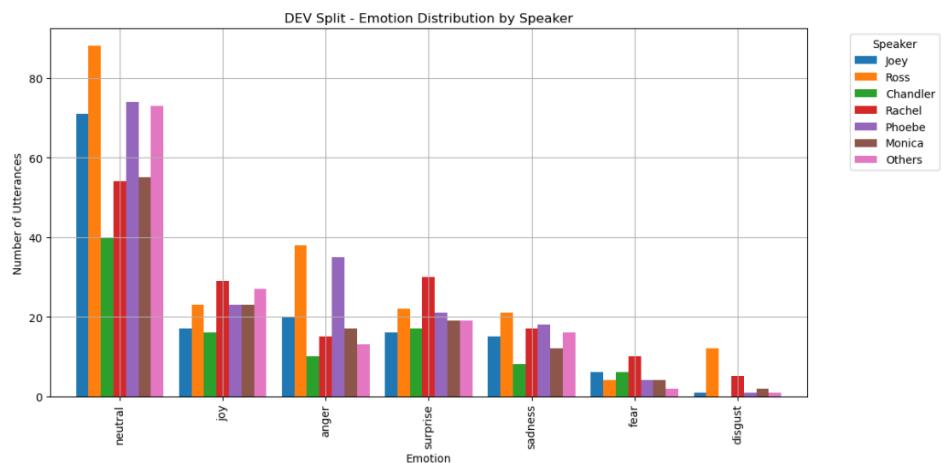


Figure 5.5 Emotion Distribution by Speaker in Dev Set

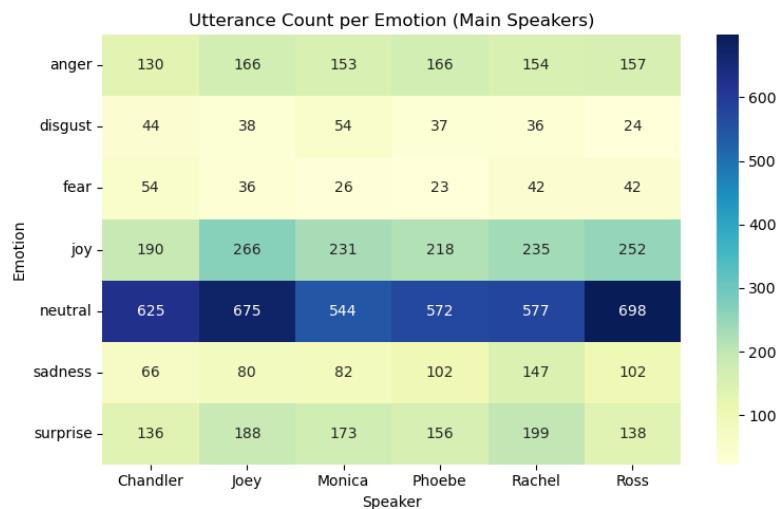


Figure 5.6 Utterance Count per Emotion Heatmap for Main Speakers

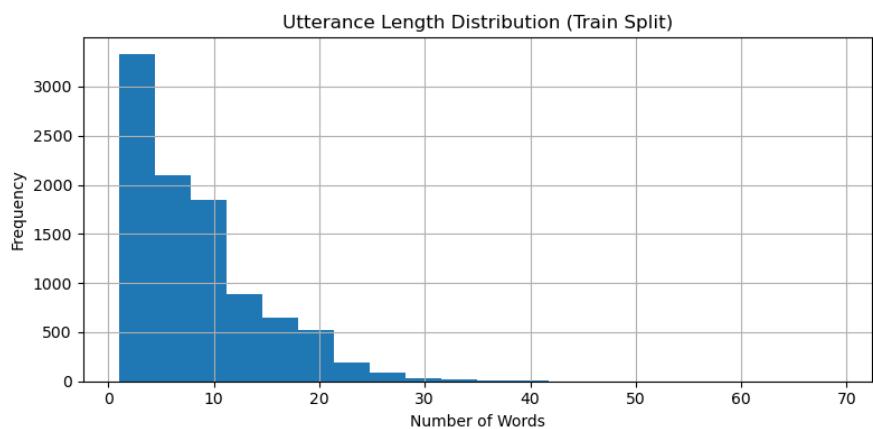


Figure 5.7 Utterance Length Distribution in Train Set

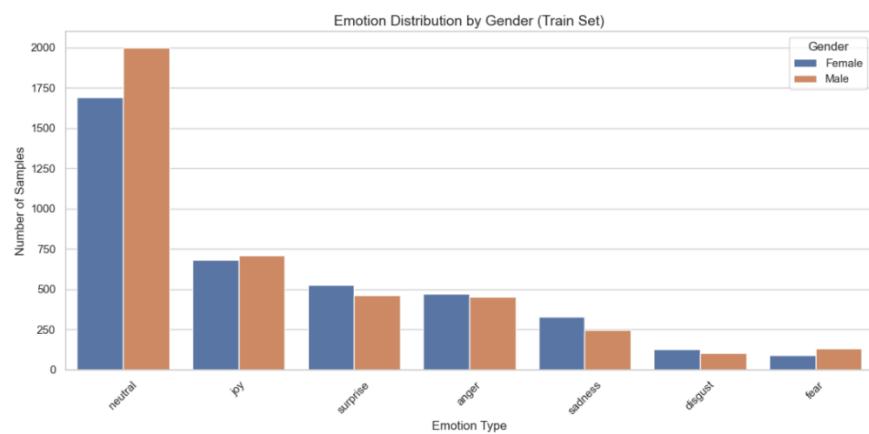


Figure 5.8 Emotion Distribution by Gender in Train Set

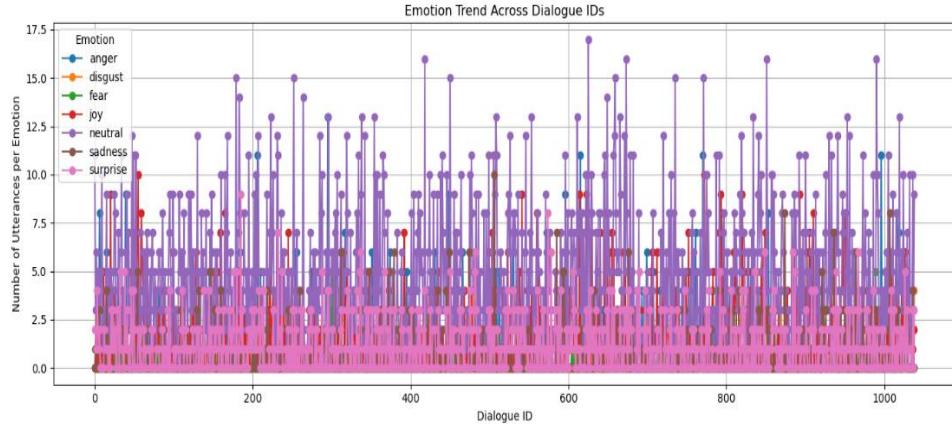


Figure 5.9 Emotion Trend Across Dialogue IDs in Train Set

In addition to these emotion labels, each utterance is also annotated with a sentiment polarity: positive, negative, or neutral. However, in this study, sentiment labels were not used, as the focus remained exclusively on categorical emotion recognition using the audio signal. Future extensions may incorporate sentiment-aware modeling or multi-task learning with both emotion and sentiment targets. This dual labeling structure allows for flexible experimentation across both emotion recognition and sentiment analysis tasks.

Each utterance is annotated considering not only the spoken content but also prosodic (tone, pitch, rhythm) and contextual cues such as facial expressions and conversational dynamics. This means that the emotion labels reflect the holistic expression of the utterance, not merely the audio or text alone. However, in this project, only the audio modality is used for classification purposes.

Note: MELD, being a multimodal dataset, presents inherent challenges in aligning all modalities. Several utterances have discrepancies between the transcript and audio content, or between annotated emotion and observable visual expressions. For instance, the CSV might include an utterance like “*Yes. I know*”, while the actual audio includes more content (e.g., “*Yes. I know, but I'm not going.*”). These inconsistencies were verified through Whisper-based transcription comparisons, audio duration calculations, and manual sampling. Utterances with extreme mismatch between CSV and audio, or missing content altogether, were excluded from the final dataset. Likewise, video clips might show faces expressing emotions inconsistent with the assigned labels. These misalignments have been carefully handled during preprocessing.

Data Organization

The dataset is organized into three main subsets:

- train_sent_emo.csv, dev_sent_emo.csv, test_sent_emo.csv

Each CSV file contains labeled utterances along with metadata such as speaker identity, emotion, sentiment, timing, and episode-related fields. Table 5.1 provides an overview of the columns in the dataset.

Table 5.1 Column Specifications of Labeled Dataset

Column Name	Description
Sr No.	Serial numbers of the utterances mainly for referencing the utterances in case of different versions or multiple copies with different subsets
Utterance	Individual utterances from EmotionLines as a string.
Speaker	Name of the speaker associated with the utterance.
Emotion	The emotion (neutral, joy, sadness, anger, surprise, fear, disgust) expressed by the speaker in the utterance.
Sentiment	The sentiment (positive, neutral, negative) expressed by the speaker in the utterance.
Dialogue_ID	The index of the dialogue starting from 0.
Utterance_ID	The index of the particular utterance in the dialogue starting from 0.
Season	The season no. of Friends TV Show to which a particular utterance belongs.
Episode	The episode no. of Friends TV Show in a particular season to which the utterance belongs.
StartTime	The starting time of the utterance in the given episode in the format 'hh:mm:ss,ms'.
EndTime	The ending time of the utterance in the given episode in the format 'hh:mm:ss,ms'.
Gender (added)	Speaker gender metadata (added during preprocessing)
FileName (added)	Audio filename in the format diaX_uttY.wav for linking with audio data

The video and audio files corresponding to each utterance are stored in structured directories (train_splits/, dev_splits/, test_splits/) using a consistent naming format:

- dia<Dialogue_ID>_utt<Utterance_ID>.mp4

This naming convention ensures precise synchronization between CSV metadata and audiovisual content. During preprocessing, each utterance was matched with its audio segment based on these indices, and the filenames were retained in a FileName column for easier mapping.

Preprocessing and Enhancements

To ensure data quality, consistency, and effective emotion recognition from speech signals, several preprocessing steps were applied to the MELD dataset:

- **Audio Extraction:** Each .mp4 video clip was processed using ffmpeg to extract the audio channel, converting it to .wav format at 16kHz sampling rate and mono-channel configuration suitable for speech processing tasks.
- **Gender Annotation:** An auxiliary gender mapping file (gender_map.csv) was integrated to annotate each speaker as either male or female. Although gender-specific composite emotion labels (e.g., female_fear, male_disgust) were initially explored, this fusion was later discarded. The study proceeded with original categorical emotion labels (e.g., joy, sadness, anger), but the gender column was retained in the dataset for future exploratory analysis.
- **File Consistency Checks:** Several validation steps were performed to ensure alignment between metadata and actual audio/video files:
 - Mismatches between .csv records and corresponding audio files were identified and removed.
 - Utterances with audio duration under 0.5 seconds were filtered out.
 - Clips with significant discrepancies (e.g., >0.4s) between annotated timestamps and actual audio length were discarded.
 - Alignment between audio content and transcript was verified using Whisper-based transcription.
- **Speaker Filtering:** Initially, the dataset was filtered to include only utterances spoken by main characters (e.g., Joey, Ross, Monica, etc.) to ensure consistency in vocal traits and pronunciation patterns. Later, to focus the modeling task and

reduce speaker variability, the dataset was further narrowed down to include only utterances spoken by the character Joey.

- **FileName Column Addition:** A new FileName column was added to each dataset split (train/dev/test), formatted as dia<Dialogue_ID>_utt<Utterance_ID>.wav, to facilitate efficient mapping between CSV metadata and corresponding audio files.

These preprocessing steps collectively improved the dataset’s integrity and suitability for training robust speech emotion recognition models. The remaining metadata fields, including gender and FileName, were preserved for transparency and reproducibility.

Dataset Statistics

This section provides a comprehensive overview of the dataset’s distribution at three critical stages of preprocessing and data refinement: the original MELD dataset, after cleaning, gender mapping, and filtering for main characters, and after narrowing the dataset to include only utterances spoken by Joey.

Initial Dataset Statistics (Before Any Processing):

Table 5.2 Statistics of Dataset

Statistics	Train	Dev	Test
# of modality	{a,v,t}	{a,v,t}	{a,v,t}
# of unique words	10,643	2,384	4,361
Avg. utterance length	8.03	7.99	8.28
Max. utterance length	69	37	45
Avg. # of emotions per dialogue	3.30	3.35	3.24
# of dialogues	1039	114	280
# of utterances	9989	1109	2610
# of speakers	260	47	100
# of emotion shift	4003	427	1003
Avg. duration of an utterance	3.59s	3.59s	3.58s

Table 5.3 Distribution of Emotion Classes in the Training Set

Emotion	
neutral	4709
joy	1743
surprise	1205
anger	1109
sadness	683
disgust	271
fear	268

Table 5.4 Distribution of Speakers in the Training Set

Speaker	
Joey	1510
Ross	1457
Rachel	1435
Phoebe	1321
Monica	1299
...	
Phoebe/Waitress	1
Vince	1
Gary Collins	1
Hold Voice	1
Front Desk Clerk	1

Post-Cleaning & Main Character Filtering:

Table 5.5 Gender-wise Distribution of Emotion Classes Mapped to Sentiment Categories in Train Set

Sentiment	negative	neutral	positive
Emotion			
female_anger	473	0	0
female_disgust	127	0	0
female_fear	91	0	0
female_joy	0	0	684
female_neutral	0	1693	0
female_sadness	331	0	0
female_surprise	251	0	277
male_anger	453	0	0
male_disgust	106	0	0
male_fear	132	0	0
male_joy	0	0	708
male_neutral	0	1998	0
male_sadness	248	0	0
male_surprise	246	0	216

Table 5.6 Distribution of Emotion Classes by Gender for Selected Speakers in Train Set

Emotion	
male_neutral	1998
female_neutral	1693
male_joy	708
female_joy	684
female_surprise	528
female_anger	473
male_surprise	462
male_anger	453
female_sadness	331
male_sadness	248
male_fear	132
female_disgust	127
male_disgust	106
female_fear	91

Table 5.7 Distribution of Main Characters in the Training Set (MELD)

Speaker	
Joey	1449
Ross	1413
Rachel	1390
Phoebe	1274
Monica	1263
Chandler	1245

Table 5.8 Distribution of Emotion Classes by Gender After Data Augmentation

Emotion	
male_neutral	1971
female_neutral	1657
female_sadness	1643
female_surprise	1551
male_fear	1439
female_disgust	1415
female_anger	1410
male_joy	1398
male_surprise	1359
female_joy	1354
male_anger	1341
male_sadness	1229
male_disgust	1184
female_fear	1007

Speaker-Specific Subset: Joey Only:

Table 5.9 Emotion Distribution for the Speaker "Joey" after Dataset Filtering

Emotion	
male_neutral	675
male_joy	266
male_surprise	188
male_anger	166
male_sadness	80
male_disgust	38
male_fear	36

Table 5.10 Emotion Distribution for "Joey" After Diarization, Trimming, and Cleaning

male_anger	116
male_disgust	26
male_fear	30
male_joy	176
male_neutral	487
male_sadness	52
male_surprise	126

Table 5.11 Emotion Distribution for "Joey" After Removing Short Audio Clips

Emotion	
male_neutral	463
male_joy	166
male_anger	113
male_surprise	109
male_sadness	52
male_fear	29
male_disgust	25

Table 5.12 Emotion Distribution for “Joey” After Data Augmentation

Emotion	
male_neutral	463
male_surprise	300
male_anger	300
male_joy	300
male_fear	300
male_disgust	300
male_sadness	300

Table 5.13 Final Balanced Emotion Distribution for Joey (Without Gender Prefixes)

Emotion	
neutral	463
surprise	300
anger	300
joy	300
fear	300
disgust	300
sadness	300

However, class imbalance is a major challenge in MELD, especially for classes like *disgust*, *fear*, and *sadness*. These were addressed later using data augmentation techniques (detailed in Section 5.5).

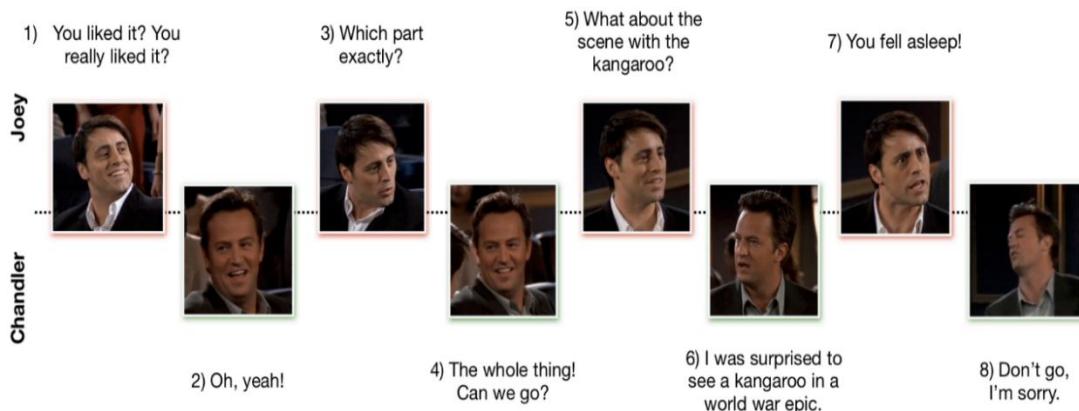


Figure 5.10 Example Dialogs

Audio video and text-based data were taken as modelling. These data contain unique expressions that are independent of each other. Preliminary information about the data such as the average and maximum length of these expressions, the emotions per dialog, the speakers that make up the expressions and the number of expressions, the average expression values are listed as seen in table 5.2.

Table 5.14 Emotion Distribution of Dataset

	Train	Dev	Test
Anger	1109	153	345
Disgust	271	22	68
Fear	268	40	50
Joy	1743	163	402
Neutral	4710	470	1256
Sadness	683	111	208
Surprise	1205	150	281

Beca In Table 5.14, the list of emotion labels in the data shows how it is distributed when grouped according to train develop and test sets.

Train, Test and Development Files and Descriptions

- **/data/MELD/train_sent_emo.csv** - contains the utterances in the training set along with Sentiment and Emotion labels.
- **/data/MELD/dev_sent_emo.csv** - contains the utterances in the dev set along with Sentiment and Emotion labels.
- **/data/MELD/test_sent_emo.csv** - contains the utterances in the test set along with Sentiment and Emotion labels.

Description of Raw Data

There are three folders in the form of .tar.gz files—train, dev, and test—each corresponding to the video clips referenced in the associated .csv files.

- In any folder, each video clip in the raw data corresponds to an expression in the corresponding .csv file.
- Video clips are named as: **diaX1_uttX2.mp4**

- X1 is the Dialogue_ID and X2 is the Utterance_ID that represents a specific expression provided in the corresponding .csv file.

Table 5.15 Example from train_sent_emo.csv

Dialogue_ID	Utterance_ID	Utterance	Speaker	Emotion	Sentiment
6	1	You liked it? You really liked it?	Monica	Surprise	Positive

This utterance is linked to the clip dia6_utt1.wav, which contains Monica's corresponding audio excerpt. Each such utterance forms the basic unit for emotion classification.

For example; For the dia6_utt1.mp4 video clip, the expression corresponding to this video clip can be specified in the Train_sent_emo.csv file with Dialogue_ID=6 and Utterance_ID=1,'

- *You liked it? You really liked it?'*



Figure 5.11 Dialog and Screen Example

```
-->
dia0_utt0:
Dialogue_ID: '0'
Emotion: sadness
EndTime: 00:21:00,049
Episode: '7'
Season: '4'
Sentiment: negative
Speaker: Phoebe
SrNo: '1'
StartTime: 00:20:57,256
Utterance: Oh my God, he's lost it. He's totally lost it.
Utterance_ID: '0'
dia0_utt1:
Dialogue_ID: '0'
Emotion: surprise
EndTime: 00:21:03,261
Episode: '7'
Season: '4'
Sentiment: negative
Speaker: Monica
SrNo: '2'
StartTime: 00:21:01,927
Utterance: What?
Utterance_ID: '1'
dia100_utt0:
Dialogue_ID: '100'
Emotion: neutral
```

Figure 5.12 Dataset Dialogs and Utterances Variables

A	B	C	D	E	F	G	H	I	J	K	L
Sr No.	Utterance	Speaker	Emotion	Sentiment	Dialogue_Utterance	Season	Episode	StartTime	EndTime		
1	1 also I was the point person on my company's transition from the KL-5 to GR-6 syste	Chandler	neutral	neutral	0	0	8	21 00:16:16,059	00:16:21,731		
2	2 You mustâ€™ve had your hands full.	The Interviewer	neutral	neutral	0	1	8	21 00:16:21,940	00:16:23,442		
3	3 That I did.	Chandler	neutral	neutral	0	2	8	21 00:16:23,442	00:16:26,389		
4	4 So letâ€™s talk a little bit about your duties.	The Interviewer	neutral	neutral	0	3	8	21 00:16:26,820	00:16:29,572		
5	5 My duties? All right.	Chandler	surprise	positive	0	4	8	21 00:16:34,452	00:16:40,917		
6	6 Now youâ€™ll be heading a whole division, so youâ€™ll have a lot of duties.	The Interviewer	neutral	neutral	0	5	8	21 00:16:41,126	00:16:44,337		
7	7 I see.	Chandler	neutral	neutral	0	6	8	21 00:16:48,800	00:16:51,886		
8	8 But thereâ€™ll be perhaps 30 people under you so you can dump a certain amount on	The Interviewer	neutral	neutral	0	7	8	21 00:16:48,800	00:16:54,514		
9	9 Good to know.	Chandler	neutral	neutral	0	8	8	21 00:16:59,477	00:17:00,478		
10	10 We can go into detail.	The Interviewer	neutral	neutral	0	9	8	21 00:17:00,478	00:17:02,719		
11	11 No donâ€™t i beg of you!	Chandler	fear	negative	0	10	8	21 00:17:02,856	00:17:04,858		
12	12 All right then, weâ€™ll have a definite answer for you on Monday, but I think I can say	The Interviewer	neutral	neutral	0	11	8	21 00:17:05,025	00:17:13,324		
13	13 Really?	Chandler	surprise	positive	0	12	8	21 00:17:13,491	00:17:16,536		
14	14 Absolutely. You can relax	The Interviewer	neutral	neutral	0	13	8	21 00:17:17,579	00:17:20,707		
15	15 But then who? The waitress I went out with last month?	Joey	surprise	negative	1	0	9	23 00:36:40,364	00:36:42,824		
16	16 You know? Forget it!	Rachel	sadness	negative	1	1	9	23 00:36:44,368	00:36:46,578		
17	17 No-no-no-no, no! Who, who were you talking about?	Joey	surprise	negative	1	2	9	23 00:36:44,368	00:36:49,122		
18	18 No, I-I-I don't, I actually don't know	Rachel	fear	negative	1	3	9	23 00:36:49,290	00:36:51,791		
19	19 OK!	Joey	neutral	neutral	1	4	9	23 00:36:52,376	00:36:53,543		
20	20 All right, well...	Joey	neutral	neutral	1	5	9	23 00:36:53,545	00:36:55,000		
21	22 Yash circu	Rachel	neutral	neutral	1	6	9	23 00:36:59,475	00:37:01,139		

Figure 5.13 Dataset Sample

Labelling

To facilitate model training and evaluation, all categorical labels were converted into one-hot encoded format. Each label corresponds to a specific index in the vector representation. The mappings are as follows:

- Emotion Labels
 - The emotion categories are mapped to the following indices:
 {'neutral': 0, 'surprise': 1, 'fear': 2, 'sadness': 3, 'joy': 4, 'disgust': 5, 'anger': 6}
 - Example – *joy*:
 → One-hot encoded: [0., 0., 0., 0., 1., 0., 0.]
- Sentiment Labels
 - The sentiment annotations are mapped as follows:
 Example – *positive*:
 → One-hot encoded: [0., 1., 0.]

Initially, the study focused on sentiment classification, but the main emphasis later shifted towards fine-grained emotion recognition using the 7 emotion categories.

5.2. Emotion Recognition Modalities and Algorithms

Emotion recognition from speech is a crucial subfield of affective computing and human-computer interaction, as audio signals inherently carry both linguistic and paralinguistic cues. In this study, we focus solely on the audio modality, extracted from a filtered version of the MELD (Multimodal EmotionLines Dataset) corpus. Specifically, all utterances are constrained to those spoken by the character Joey from the *Friends* TV series. This speaker-specific subset allows for more consistent acoustic modeling by minimizing inter-speaker variability and preserving uniform prosodic patterns.

Although MELD is a multimodal dataset comprising audio, video, and text, this project intentionally emphasizes speech-only emotion recognition. This approach reflects real-world constraints where visual or textual cues may not always be available, such as in telephone calls, smart assistants, or privacy-sensitive environments. Notably, the MELD emotion annotations were made with access to all three modalities, which introduces a challenge for audio-only recognition — for example, an utterance labeled as “sad” might sound neutral in tone when voice is isolated.

To mitigate such challenges, the audio pipeline was rigorously cleaned and aligned using Whisper for transcript verification and PyAnnote for speaker diarization. Additional filters removed audio shorter than 0.5 seconds or those with excessive mismatch between annotated and actual duration. Silence trimming, noise normalization, and extensive data augmentation (pitch shift, time stretch, background noise injection) were also applied to address class imbalance and acoustic variance.

Despite these mono-modal constraints, the system achieved up to 86% accuracy in emotion classification using advanced models — including deep learning (CNN, LSTM, MLP), transformer-based architectures (e.g., WavLM, HuBERT, Wav2Vec2.0), and robust classical ML pipelines. This performance illustrates that speech alone, when enhanced with high-quality preprocessing and appropriate modeling strategies, can yield highly competitive results even when benchmarked against multimodal standards.

5.2.1. Audio Modality

The audio-only emotion recognition pipeline in this project is composed of multiple stages, meticulously designed to optimize signal fidelity, robustness, and model

compatibility. Each step has been tailored to suit both traditional machine learning (ML) and advanced deep learning (DL) and transformer-based (SOTA) models.

A. Preprocessing and Cleaning

To ensure data integrity and quality before feature extraction, a comprehensive audio preprocessing pipeline was applied:

- Speaker diarization and alignment verification were performed using Whisper and PyAnnote.
- Filtering and removal of problematic audio files, including:
 - Files with duration less than 0.5 seconds
 - Files with mismatched duration between .mp4 and annotated timestamps (threshold: 0.4s)
 - Files with missing or undetectable speech
- Resampling: All audio signals were converted to 16 kHz, 16-bit PCM format.
- Silence trimming: Initial and trailing silences were removed using librosa.effects.trim.
- Noise reduction: Applied using noisereduce spectral gating.
- RMS normalization: To ensure amplitude consistency across samples.
- Padding: Short clips were padded to 1 second to allow fixed-length input for some DL models.

B. Data Augmentation

To mitigate class imbalance (especially dominant neutral class), multiple audio augmentation techniques were applied selectively using librosa, scipy, and audiomentations:

- Additive white noise
- Pitch shifting (\pm steps)
- Time stretching

- Echo and reverb
- Bandpass / lowpass filtering
- Volume modulation
- Waveform inversion (reversing)
- Random clipping and shifting

Each minority class was augmented to reach 300 samples, ensuring a balanced dataset.

C. Feature Extraction:

Three main paradigms were followed depending on the model type:

1. Classical ML / DL (Only MLP) Features:

Used with ML and some DL (MLP) models. Feature extraction was performed using librosa, parselmouth, and OpenSMILE. Key features include:

- MFCCs (40 coefficients + delta + delta²) → summarized with statistical metrics (mean, std, skewness, kurtosis)
- Chroma STFT
- Mel-Spectrogram
- Spectral Features: centroid, bandwidth, contrast, rolloff
- Prosodic: Zero-Crossing Rate (ZCR), Root Mean Square (RMS), pitch (F0), jitter, shimmer, Harmonics-to-Noise Ratio (HNR), formants
- Advanced Representations: DFT, TEO, Wavelet (PyWavelets), LPC, LPCC
- OpenSMILE eGeMAPS v01a: Affective speech descriptors

These features were concatenated into a high-dimensional vector (~1480 dimensions), later processed through feature selection.

2. DL Features:

Used visual inputs for CNN-based DL models:

- Mel Spectrogram Images (128×128 , 224×224)
- MFCC Spectrogram Images (128×128 , 224×224)
- RGB Fusion: MFCC (R) + Delta (G) + Delta² (B)
- Multi-Input (Mel + MFCC): Combined CNN branches
- Transfer Learning Inputs: ResNet18 (Mel), EfficientNetB0 (RGB Fusion)

Spectrograms were generated using `librosa.display.specshow()` and stored in image folders.

3. SOTA Embeddings Features:

Used with transformer-based pretrained models:

Raw waveform input fed into models like:

- microsoft/wavlm-large, microsoft/wavlm-base-plus
- facebook/wav2vec2-large-960h-lv60, ehcalabres/wav2vec2-lg-xlsr-en-speech-emotion-recognition
- facebook/hubert-base-ls960, facebook/hubert-large-ls960-ft

These models generated 1024/768-dimensional embeddings, which were used as inputs to shallow classifiers (e.g., MLP), or fine-tuned end-to-end.

D. Feature Selection & Normalization:

To prevent overfitting and reduce dimensionality, various feature selection techniques were employed:

- Variance Threshold, SelectKBest (Mutual Information, Chi-Square)
- Random Forest and XGBoost feature importance
- RFE, Sequential Forward Floating Selection (SFFS)
- Autoencoder-based dimensionality reduction
- Boruta feature selection
- PCA, LDA (optional evaluation)

Additionally, K-Means clustering-based undersampling was applied specifically to the overrepresented neutral class, offering a smarter alternative to random or DBSCAN undersampling. This ensured diversity within neutral samples while reducing the class to 300 examples.

E. Classification Algorithms:

A wide range of classifiers were implemented, tailored to the nature of the input features:

- *Traditional Machine Learning Models:*

- Support Vector Machines (SVM)
- Random Forest (RF)
- k-Nearest Neighbors (KNN)
- Logistic Regression
- Naive Bayes
- Decision Trees
- Gradient Boosting
- Bagging (KNN or DT)
- Hard / Soft Voting Ensembles

All ML models used the same feature extraction pipeline and tested with 11 different feature selection sets.

- *Deep Learning Models:*

- CNN variants: Mel, MFCC, RGB, Multi-input, Transfer Learning (ResNet, EfficientNet)
- Sequential: LSTM, BiLSTM, GRU, CNN-BiLSTM hybrids
- Fully Connected: Torch-MLP, Keras-MLP

Each DL model used tailored input format: either spectrograms or time-series features (e.g., [T, 120] MFCC + delta + delta²)

- *SOTA Models:*
 - Frozen Feature Extractors + Classifier:
 - WavLM-Large / Base-Plus (Pretrained (frozen)) + MLP
 - Wav2Vec2.0-Large / Emotion-Plus (Pretrained (frozen)) + MLP
 - HuBERT-Large / Base-Plus (Pretrained (frozen)) + MLP
 - End-to-End Fine-Tuned Transformers:
 - WavLM (HuggingFace Trainer)
 - HuBERT (HuggingFace Trainer)
 - Wav2Vec2.0 (HuggingFace Trainer)

Transformers received raw audio input and output emotion probabilities directly or via MLP.

F. Evaluation

All models were evaluated using consistent metrics:

- Accuracy
- F1-score (macro and per-class)
- Precision, Recall
- Confusion Matrix

Both test-time inference and batch evaluation pipelines were implemented, and GUI-based real-time prediction support was added for each model.

5.2.2. Multimodal Potential (Future Scope)

While this project focuses exclusively on audio-based emotion recognition, the true strength of the MELD dataset lies in its multimodal structure, combining audio, text, and visual cues. These modalities, when used in synergy, can significantly improve the robustness and accuracy of emotion classification models:

Text Modality:

- MELD includes transcripts (via the *Utterance* column), which can be utilized with NLP techniques such as.
 - Tokenization and vectorization via pretrained language models like Word2Vec, GloVe, or BERT.
 - Sentiment and emotion classification through transformer-based encoders.

Visual Modality:

- Original mp4 video files allow rich visual analysis, including:
 - Frame extraction and face detection (e.g., RetinaFace)
 - Face tracking (e.g., DeepSORT)
 - Speaker diarization (PyAnnote)
 - Face-emotion recognition (e.g., using CNN or FER+)
 - Multimodal alignment between audio, text, and face signals

This pipeline would enable frame-level emotion extraction and comparison against MELD labels, allowing per-modality reliability assessments.

Fusion Techniques (Future Work):

To fully utilize MELD's multimodal nature, the following fusion strategies are planned for future development:

- **Early Fusion:** Concatenating low- or mid-level features from audio, text, and video before feeding into the model.
- **Late Fusion:** Merging independent modality predictions at decision level (e.g., weighted average, majority voting).
- **Hybrid Fusion:** Leveraging attention-based or context-aware architectures (e.g., multimodal transformers, gated units) to dynamically weigh modalities.

By combining multiple signals, these methods can help bridge the gap between holistically annotated labels and single-modality limitations, improving emotion recognition performance and generalizability in complex real-world scenarios.

5.2.3. Some Algorithms Used and Explored

In terms of algorithms, both classical machine learning (ML) and deep learning (DL) models were implemented and evaluated. Below is a summary of the algorithms explored during the project:

A. Classical Machine Learning Algorithms:

i. Support Vector Machine (SVM)

Support Vector Machines (SVMs) are supervised learning models used for classification and regression analysis (Cortes & Vapnik, 1995). In the context of speech emotion recognition (SER), SVMs are effective in distinguishing between emotional states using extracted acoustic features such as MFCCs, pitch, energy, and spectral descriptors. The key advantage of SVMs lies in their ability to create optimal decision boundaries (hyperplanes) that maximize the margin between different emotion classes, even in high-dimensional feature spaces.

SVM can handle linear and non-linear classification problems using kernel functions, such as the Radial Basis Function (RBF) or polynomial kernel, which project the input space into higher dimensions to make data linearly separable.

ii. Decision Tree

Decision Trees are supervised learning algorithms used for both classification and regression tasks. They operate by recursively splitting the dataset into subsets based on feature values, aiming to maximize information gain at each split. In the context of speech emotion recognition, decision trees can effectively classify emotions based on acoustic features such as MFCCs, pitch, and energy (Quinlan, 1986).

The decision tree's hierarchical structure provides interpretability and makes it suitable for analyzing how specific features contribute to emotion detection.

iii. Naive Bayes

Naive Bayes is a probabilistic classifier based on Bayes' Theorem, with a strong (naive) assumption that all features are conditionally independent given the class. Despite

this simplification, it has proven to be a robust and efficient classifier, especially in text and speech-based emotion recognition, where features like MFCCs, pitch, and energy are used (Chauhan, 2020).

In SER, Naive Bayes can be employed to estimate the probability of an emotional class given an observed set of acoustic features. It is computationally efficient and works well with high-dimensional data, making it suitable for real-time applications or baseline models.

iv. K- Nearest Neighbor (K-NN)

A general version of the nearest neighbor technique bases the classification of an unknown sample on the “votes” of K of its nearest neighbor rather than on only it’s on single nearest neighbor. If the costs of error are equal for each class, the estimated class of an unknown sample is chosen to be the class that is most represented in the collection of its K nearest neighbors (Cover & Hart, 1967).

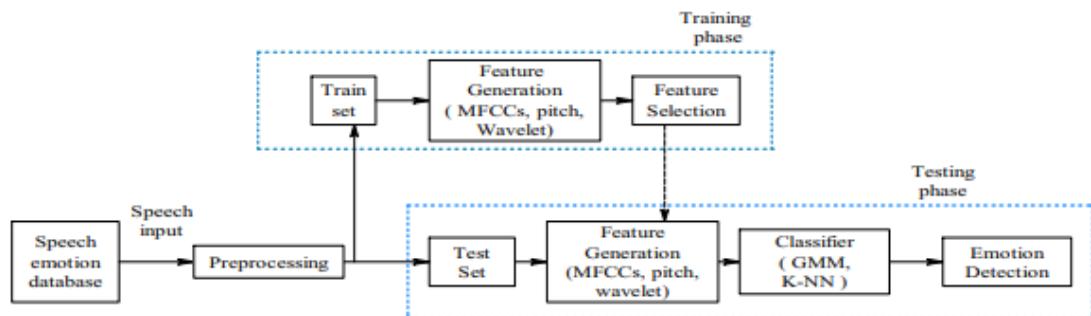


Figure 5.14 Studies and Comparisons with this Data Set

v. Random Forest

An ensemble-based decision tree algorithm that operates by constructing multiple decision trees during training and outputting the mode of their predictions. It is known for robustness against overfitting and handles high-dimensional feature spaces effectively.

vi. XGBoost

An optimized implementation of gradient boosting that supports parallel processing, regularization, and missing value handling. It is widely used for structured data and provides superior performance in many classification tasks.

vii. Gradient Boosting

A sequential ensemble method that builds weak learners (usually decision trees) in a stage-wise manner, where each subsequent learner tries to correct the errors of the previous ones. It is particularly powerful for complex, non-linear data patterns.

viii. Bagging Classifier

A bootstrap aggregation technique that trains multiple base learners (e.g., decision trees or KNN) on random subsets of the data and aggregates their predictions, typically by majority voting. It reduces variance and improves generalization.

ix. Voting Ensemble

An ensemble strategy that combines the predictions of multiple diverse models. Hard voting uses majority rule, while soft voting averages the predicted probabilities. This technique leverages the strengths of various classifiers to improve overall accuracy.

x. MLP (Scikit-learn)

A feedforward artificial neural network implemented using scikit-learn's MLPClassifier, trained using backpropagation. It can model complex non-linear relationships and serves as a baseline deep learning model within the traditional ML framework.

B. Deep Learning Algorithms:

xi. Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are a class of deep neural networks most commonly used for image processing tasks (Albawi et al., 2017). However, in recent years, they have also shown strong performance in speech emotion recognition (SER) when applied to spectrograms or Mel-frequency cepstral coefficients (MFCCs) — visual representations of audio signals. By treating these representations as images, CNNs can effectively learn spatial and temporal patterns corresponding to emotional cues in speech.

CNNs operate by applying convolutional filters over input data to capture local patterns, followed by pooling layers to reduce dimensionality and extract dominant features. These features are then passed through fully connected layers for classification.

xii. Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is an advanced type of Recurrent Neural Network (RNN) specifically designed to learn long-term dependencies in sequential data. Traditional RNNs suffer from the vanishing gradient problem, which limits their ability to retain information over long sequences. LSTMs address this limitation through a gated cell architecture that allows the network to selectively retain or forget information over time (Huang et al., 2019).

In speech-based emotion recognition, LSTM networks are particularly well-suited due to their ability to model temporal dynamics in audio signals. Emotions expressed in speech often evolve over time; LSTM networks can effectively capture these transitions and nuances by maintaining memory of previous emotional cues.

xiii. Recurrent Neural Network (RNN)

A recurrent neural network (RNN) is a type of artificial neural network which uses sequential data or time series data (Lipton et al., 2015). These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing (nlp), speech recognition, and image captioning; they are incorporated into popular applications such as Siri, voice search, and Google Translate. Like feedforward and convolutional neural networks (CNNs), recurrent neural networks

utilize training data to learn. They are distinguished by their “memory” as they take information from prior inputs to influence the current input and output. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depends on the prior elements within the sequence. While future events would also be helpful in determining the output of a given sequence, unidirectional recurrent neural networks cannot account for these events in their predictions.

RNNs provide an attractive framework for propagating information over a sequence using a continuous valued hidden layer representation.

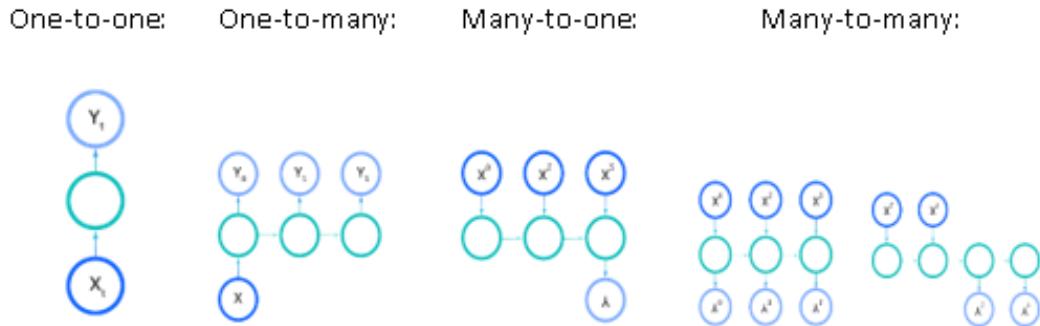


Figure 5.15 Types of Recurrent Neural Network

xiv. Deep Neural Network (DNN)

Deep Neural Networks (DNNs) are a class of feedforward artificial neural networks that consist of multiple layers of neurons between the input and output layers. Each layer in a DNN learns abstract representations of data, making them well-suited for complex pattern recognition tasks such as speech emotion classification (LeCun et al., 2015).

In SER, DNNs are often employed to model non-linear relationships between acoustic features (e.g., MFCCs, pitch, energy) and corresponding emotion labels. These networks are trained using backpropagation and optimization techniques such as stochastic gradient descent (SGD) or Adam optimizer.

xv. Connectionist Temporal Classification (CTC)

Connectionist Temporal Classification (CTC) is an objective function for end-to-end sequence learning, which adopts dynamic programming algorithms to directly learn the mapping between sequences. CTC has shown promising results in many sequence learning applications including speech recognition and scene text recognition. However, CTC tends to produce highly peaky and overconfident distributions, which is a symptom of overfitting. To remedy this, I propose a regularization method based on maximum conditional entropy which penalizes peaky distributions and encourages exploration. I also introduce an entropy-based pruning method to dramatically reduce the number of CTC feasible paths by ruling out unreasonable alignments. Experiments on scene text recognition show that our proposed methods consistently improve over the CTC baseline without the need to adjust training settings (Graves et al., 2006).

With the advent of Connectionist Temporal Classification, the need for pre-segmentation for training RNNs for tasks like speech to text transcription is completely avoided. CTC based training also removes the need for post-processed outputs by modelling all aspects of the sequence within single network architecture. CTC based network architecture performs discriminative modelling by maximizing the conditional probability of the correct labelling conditioned on input sequence. In the process of this maximization, the network weights are learnt through back propagation in time. Finally with the trained network, given an unseen input sequence, the most probable output text labels are computed with high confidence.

xvi. Automatic Speech Recognition (ASR)

ASR is a technology that transcribes human speech to text by pairing it with the system library. The accuracy of ASR depends on many factors such as voice recognition, speech continuity, speaker, and environmental differences, as well as the dataset on understanding human language. Speech transcription, emotion recognition, and language identification are generally considered to be three different tasks. Each requires a different model with a different architecture and training process (Juang & Rabiner, 2005).

It proposes using a repetitive neural network transducer (RNN-T)-based speech-to-text (STT) system as a common component that can be used for speech recognition as well as emotion recognition and language identification.

Our study extends the STT system for emotion classification with minimal modifications and demonstrates that it can be used in MELD datasets.

xvii. Gaussian Mixture Model (GMM)

In this study, a Gaussian Mixture Model approach is proposed where speech emotions are modelled as a mixture of Gaussian densities. The use of this model is motivated by the interpretation that the Gaussian components represent some general emotion dependent spectral shapes and the capability of Gaussian mixtures to model arbitrary densities (Reynolds, 2009).

The Gaussian Mixture Model is a linear combination of M Gaussian densities, and given by the equation,

$$P(x|\lambda) = \sum_{i=1}^M p_i b_i(x) \quad (\text{Eq. 5.1})$$

where x is a D-dimensional random vector, $i=1, \dots, M$ are the component densities and, p_i are the mixture weights. Each component density is a D-dimensional Gaussian function of the form:

$$b_i(x) = (1/(2\pi)^{D/2}) \exp \{-1/2(x - \mu)^T \Sigma^{-1} (x - \mu)\} \quad (\text{Eq. 5.2})$$

xviii. Hidden Markov Model (HMM)

Hidden Markov models can be regarded as the simplest dynamic Bayesian networks (DBN). They have a long tradition in speech recognition based on the idea that the statistics of voice are not stationary (Rabiner, 1989).

The use of HMM and their capability to model the temporal behavior of speech as opposed to the global statistics approach has more advantages. The HMM consists of the first order Markov chain whose states are hidden from the observer therefore the internal behavior of the model remains hidden. The hidden states of the model capture the temporal structure of the data. Hidden Markov models are the statistical models that describe the sequences of events.

HMM is having the advantage that the temporal dynamics of the speech features can be captured due to the presence of the state transition matrix. During classification, a speech signal is taken and the probability for each speech signal is calculated to the model.

An output of the classifier is based on the maximum probability that the model has generated this signal.

C. SOTA / Transformer-Based Algorithms:

xix. Wav2Vec2.0:

A self-supervised model for learning speech representations directly from raw audio waveforms. It uses a contrastive loss to pretrain a feature encoder, enabling effective downstream fine-tuning for emotion classification with limited labeled data (Baevski et al., 2020).

xx. HuBERT

Hidden-Unit BERT (HuBERT) is a self-supervised speech model that predicts masked audio units based on hidden cluster assignments. It leverages offline clustering and BERT-style prediction, making it highly effective for transfer learning in speech-related tasks (Hsu et al., 2021).

xxi. WavLM

WavLM is a large-scale self-supervised model designed to handle speech denoising and speaker-aware modeling. It enhances contextual representation of speech and achieves state-of-the-art performance across multiple benchmarks, including emotion recognition (Chen et al., 2022).

xxii. WavLM + MLP

A hybrid architecture where WavLM is used as a feature extractor and its output embeddings are passed through a Multi-Layer Perceptron (MLP) classifier. This allows leveraging the power of pretrained representations with a lightweight downstream classifier.

xxiii. HuBERT + MLP

A hybrid model combining HuBERT's contextual embeddings with a fully connected MLP for classification. This approach benefits from the pretrained knowledge of

HuBERT while allowing customization and efficient adaptation to the emotion recognition task.

xxiv. Wav2Vec2 + MLP

A hybrid structure where Wav2Vec2.0’s learned representations serve as input to a downstream MLP. This decouples feature learning from classification, enabling robust performance even with limited training data and smaller model sizes.

5.3. Methodology, Tools, and Libraries

5.3.1. Methodology Overview

This study follows a modular, reproducible, and extensible pipeline designed to detect emotions from speech data extracted from the MELD (Multimodal EmotionLines Dataset) corpus. The methodology is specifically tailored for audio-only emotion recognition but is extensible for future multimodal systems. The complete process is structured into the following key stages:

5.3.1.1. Data Preparation and Cleaning

- Raw video files (.mp4) and their corresponding annotation files (train_sent_emo_joey.csv, etc.) were synchronized based on unique identifiers (Dialogue_ID, Utterance_ID).
- Audio segments were extracted using FFmpeg and converted to 16-bit PCM .wav format.
- Speaker metadata was enriched with gender information to enable emotion-gender correlation analysis (e.g., male_joy, female_sadness), although final modeling used neutral emotion labels (e.g., joy, sadness).
- A robust validation pipeline was implemented:
 - Files shorter than 0.5 seconds were removed.
 - Clips where the transcript duration and actual audio duration differed by more than 0.4 seconds were excluded.

- Missing, corrupt, or non-speech audio segments were filtered using WhisperX and PyAnnote for alignment verification.
- A structured directory layout was created for train, dev, and test splits.

5.3.1.2. Audio Preprocessing

- All audio files were:
 - Resampled to a uniform 16 kHz sample rate.
 - Normalized using RMS energy to equalize volume dynamics.
 - Trimmed using librosa.effects.trim() to remove silence.
 - Denoised with spectral noise reduction using the noisereduce package.
 - Padded if shorter than 1 second to ensure minimum length for spectrogram-based models.
 - The processed files were stored in structured directories:
- archive_meld/joey_audio/processed_16KHz/{train, dev, test}_splits/

5.3.1.3. Data Augmentation

To alleviate class imbalance, particularly for minority classes such as disgust, fear, and sadness, a range of audio augmentation techniques were applied:

- White noise injection
- Pitch shifting (\pm steps)
- Time stretching (slightly speeding up/slowing down)
- Echo effect
- Bandpass and lowpass filtering
- Volume scaling
- Audio reversal and time shifting

Each augmented file was uniquely renamed, logged, and appended to the corresponding .csv metadata. The final dataset was balanced, ensuring that each emotion class had an equal number of samples (~300 instances per class).

5.3.1.4. Feature Extraction

Feature extraction was conducted via a custom-designed pipeline that leverages both traditional acoustic descriptors and statistical summaries:

- Temporal & Spectral Features (via Librosa):
 - MFCCs (mean/std), Delta, Delta²
 - Chroma STFT
 - Mel Spectrogram
 - Spectral Centroid, Bandwidth, Contrast, Rolloff
 - ZCR (Zero-Crossing Rate), RMS Energy
 - Pitch/F0 estimation via YIN algorithm
- Advanced Features:
 - DFT, Wavelet coefficients (pywt), TEO
 - LPC, LPCC, Formants (via Praat/Parselmouth)
 - Jitter, Shimmer, Harmonic-to-Noise Ratio (HNR)
 - eGeMAPS features via openSMILE toolkit
- CNN-compatible Visual Representations:
 - MFCC, Mel Spectrogram images
 - RGB fusion images (R: MFCC, G: Delta, B: Delta²)
- Transformer Embeddings:
 - Wav2Vec2, WavLM, and HuBERT embeddings were extracted using Hugging Face models.

All extracted features were saved in structured .csv files (e.g., meld_features_ml_1.csv, *_balanced.csv) and organized by feature set version.

5.3.1.5. Feature Encoding and Normalization

- Emotion labels were:
 - Encoded with LabelEncoder (for ML and some DL models)
 - One-hot encoded for deep learning architectures where required
- All numerical features were normalized using StandardScaler.
- Separate scalers were saved per feature set for compatibility during inference.
- Stratified train-validation-test splits were maintained to preserve class balance.

5.3.1.6. Model Training and Evaluation

A wide range of models were trained and evaluated using both traditional ML, deep learning, and state-of-the-art transformer models:

- **Machine Learning Models:**
 - Support Vector Machine (SVM)
 - Random Forest (RF)
 - XGBoost
 - Gradient Boosting Classifier
 - Bagging Classifier
 - Hard and Soft Voting Ensembles
 - k-Nearest Neighbors (k-NN)
 - Logistic Regression, Naive Bayes
 - MLPClassifier (from scikit-learn)

- **Deep Learning Models:**
 - CNN models (Mel, MFCC, RGB, Multi-input)
 - BiLSTM, GRU, CNN-BiLSTM hybrids
 - Keras and PyTorch MLP models
- **Transformer-based Models:**
 - Wav2Vec2.0, WavLM, HuBERT
 - Hybrid models: WavLM + MLP, HuBERT + MLP, Wav2Vec2 + MLP
- **Optimization:**
 - Hyperparameters were tuned using GridSearchCV or manual tuning (DL)
 - Best model weights and training artifacts (scalers, encoders, feature selectors) were stored for inference
- **Evaluation Metrics:**
 - Accuracy
 - Precision, Recall
 - F1-score (macro)
 - Confusion Matrix
 - Training time and inference speed

The best classification performance reached approximately 86% accuracy. In general, the majority of models—including classical, deep learning, and transformer-based architectures—achieved consistent results in the 75–86% range. However, some baseline configurations yielded lower performance around the 60% level, especially when using unfiltered or less informative feature sets.

5.3.2. Tools and Libraries

The implementation of this project was carried out entirely in Python, leveraging its rich ecosystem of scientific computing and machine learning libraries. These tools facilitated each stage of the pipeline—from preprocessing and feature extraction to model

training and GUI integration. The primary libraries and tools used are listed in the table below:

Table 5.16 Tools and Libraries Used in the Project

Tool / Library	Purpose
Python 3.12	Core programming language
Librosa	Audio processing and feature extraction (MFCC, Chroma, Mel Spectrogram, Spectral features, ZCR, RMS)
Parselmouth (Praat)	Extraction of prosodic and voice quality features (jitter, shimmer, HNR, formants)
PyWavelets (PyWT)	Extraction of wavelet-domain features
scipy.stats	Statistical summarization of features (mean, std, skew, kurtosis, etc.)
noisereduce	Background noise removal from raw audio
OpenSMILE	Advanced acoustic feature extraction (eGeMAPSv01a configuration)
Hugging Face Transformers	Pretrained transformer-based models (Wav2Vec2, HuBERT, WavLM) and their feature extractors
Torchaudio	Audio I/O and waveform transformations, mainly for SOTA feature extraction
SpeechRecognition, PyAudio	Audio input and real-time speech recognition (optional)
NumPy, Pandas	Data manipulation and numerical operations
Matplotlib, Seaborn	Visualization of training metrics, emotion distributions, confusion matrices

Tool / Library	Purpose
Scikit-learn	Classical ML algorithms (SVM, RF, NB, KNN), scaling, encoding, feature selection, GridSearchCV
XGBoost, LightGBM	Advanced tree-based classifiers and feature importance computation
TensorFlow / PyTorch	Deep learning model implementations (MLP, CNN, RNN, LSTM, BiLSTM, transfer learning)
FFmpeg	Conversion from .mp4 video files to .wav audio
Jupyter Notebook	Rapid experimentation, prototyping, and documentation
PyCharm, Spyder, Visual Studio	Integrated development environments (IDE)
Tkinter / PyQt5	Graphical User Interface (GUI) development
TinyDB, Pickle, Joblib	Storing and loading of models, encoders, feature sets, and metadata
Google Colab	Cloud-based development, GPU/TPU training for transformer-based and deep models

The system runs on a Windows operating system, ensuring compatibility and accessibility for further development and integration. PyCharm serves as the primary integrated development environment (IDE) due to its efficient debugging and package management capabilities.

5.4 Data Cleaning, Preprocessing and Augmentation

Data cleaning and preprocessing play a pivotal role in ensuring the success of any speech emotion recognition system. In this project, a rigorous and multi-stage pipeline was implemented to refine the audio data extracted from the MELD dataset. These steps

enhanced the quality, consistency, and alignment of the data, allowing for reliable feature extraction and robust model performance.

5.4.1. File Matching and Removal of Inconsistencies

Each utterance in the MELD dataset is uniquely identified by Dialogue_ID and Utterance_ID, from which audio and video filenames were derived in the format diaX_uttY.wav or .mp4. A synchronization validation process was applied:

- All filenames in the provided train/dev/test CSV files were matched against the actual files in the dataset folders.
- Missing, corrupted, or misaligned entries (e.g., duration mismatch > 0.4s) were identified and excluded.
- Special attention was given to ensure alignment between the audio duration and the annotated start/end timestamps via tools like WhisperX and PyAnnote.
- Invalid or non-speech segments were discarded.

This ensured dataset integrity and guaranteed that each retained sample contained valid, speech-aligned emotional content.

5.4.2. Speaker and Gender Filtering

To reduce acoustic variability and ensure speaker consistency:

- Gender information was extracted using an auxiliary gender_map.csv, and optionally used for emotion label enrichment (e.g., male_anger).
- For this project, only utterances spoken by the character Joey were retained. This ensured:
 - Consistency in speaker prosody and vocal traits,
 - Reduction of inter-speaker variance, which could affect emotion modeling,
 - Simpler analysis and model specialization.

The filtered dataset was therefore speaker-specific and highly controlled in terms of identity and expression.

5.4.3. Audio Extraction from Video (mp4 → wav)

Audio All utterances originally stored as .mp4 video clips were converted to .wav audio using FFmpeg, with the following configuration:

- Mono channel (-ac 1)
- Sampling rate: 16,000 Hz (-ar 16000)
- Audio format: 16-bit PCM
- Video disabled (-vn)

The extracted audio was organized into the following structure:

archive_meld/joey_audio/processed_16KHz/{train_splits, dev_splits, test_splits}

5.4.4. Audio Preprocessing Pipeline

To ensure acoustic quality and consistency, a multi-step preprocessing routine was applied to all audio files:

a. Silence Trimming

- Leading and trailing silences were removed using librosa.effects.trim().

b. Noise Reduction

- Background noise was suppressed using the noisereduce library to improve signal clarity.

c. RMS Normalization

- Each audio sample was normalized based on its energy (Root Mean Square) to reduce volume variation across samples.

d. Resampling and Standardization

- All audio was resampled to 16,000 Hz and stored in 16-bit PCM format.
- Short audio files (< 0.5 sec) were either removed or padded to 1 second duration to ensure compatibility with model input.

e. Short Audio Filtering and Padding

- Files under the minimum threshold (e.g., 0.5s) were logged and either discarded or zero-padded, depending on the modeling needs.

The processed files were saved in the directory processed_16KHz/, fully cleaned and ready for feature extraction.

5.4.5. Augmentation Preparation and Logging

Given the class imbalance in emotion categories (e.g., low sample counts for disgust, fear, sadness):

- A variety of data augmentation techniques were applied to underrepresented classes, such as:
 - Pitch shifting
 - Time-stretching
 - Background noise addition
 - Volume modification
 - Time shifting and echo
 - Bandpass filtering and audio inversion
- A FileName column was added to the label files to maintain consistent tracking of augmented samples.
- A separate CSV log (augmentation_log.csv) was created to document all augmented instances.

This augmentation process brought all classes to approximately equal sample size (~300), significantly improving class balance for training.

5.4.6. Logging and Traceability of Excluded Samples

Throughout the preprocessing pipeline, a comprehensive logging mechanism was implemented to ensure traceability:

- short_audio_files.csv: Logs all samples removed due to short duration or silence.

- missing_files_log.csv: Tracks files referenced in CSVs but missing in the filesystem.
- augmentation_log.csv: Tracks all synthetic samples generated via augmentation.

These logs facilitated debugging, model interpretation, and reproducibility of experiments.

5.4.7. Summary

The preprocessing pipeline ensured high-quality, speaker-consistent, and balanced audio input for the models. Without these rigorous steps, emotional cues in speech could be diluted by noise, misalignment, or speaker variability—leading to poor model generalization. As a result of this comprehensive data preparation process, the final dataset was acoustically clean, evenly distributed, and ready for advanced feature extraction and modeling.

5.5 Feature Extraction and Selection

Emotion recognition systems heavily rely on extracting relevant features from audio signals. In this project, a diverse and comprehensive feature extraction pipeline was implemented to support machine learning (ML), deep learning (DL), and state-of-the-art (SOTA) transformer-based models. The aim was to capture both low-level acoustic features and high-level representations that contribute to accurate emotion classification.

5.5.1 *Classical Feature Extraction for ML and MLP Models*

For all traditional ML models (e.g., SVM, RF, XGBoost, etc.) and Torch/Keras MLP models, handcrafted acoustic features were extracted from the preprocessed .wav audio files using libraries such as librosa, parselmouth, scipy, pywt, and OpenSMILE. These features were aggregated using statistical descriptors like mean, standard deviation, skewness, and kurtosis, resulting in a fixed-length vector per sample.

5.5.1.1 MFCCs (Mel-Frequency Cepstral Coefficients) and Derivatives

- MFCC (40 dimensions): Capture short-term power spectrum shaped by human auditory perception.
- Delta and Delta²: First and second derivatives to capture temporal dynamics.
- Statistics: mean, std → $(40 + 40 + 40) \times 2 = 240$ features

5.5.1.2 Chroma Features

- Chroma features capture harmonic content by representing the distribution of energy across 12 pitch classes.
- Chroma STFT (Short-Time Fourier Transform) was extracted.
- mean and standard deviation ($12 + 12 = 24$)

5.5.1.3 Mel Spectrogram

- Represents the energy distribution across the frequency spectrum in a perceptually relevant scale.
- mean and standard deviation ($128 + 128 = 256$)

5.5.1.4 Spectral Features

- Centroid, Bandwidth, Rolloff, Contrast: Each with mean and std → $4 \times 2 = 8$ features.
- Spectral Bandwidth: Measures the spread of the frequencies.

5.5.1.5 ZCR (Zero Crossing Rate)

- Represents the rate at which the audio signal changes sign, indicating voicing or percussiveness.
- mean and standard deviation (2)

5.5.1.6 RMS (Root Mean Square Energy)

- Indicates the overall energy or loudness of the audio signal.
- mean and standard deviation (2)

5.5.1.7 F0 / Pitch (Using YIN Algorithm)

- Represents the fundamental frequency of the speech signal, a key indicator of intonation.
- mean and standard deviation (2)

5.5.1.8 Temporal Energy Operators (TEO)

- Nonlinear energy-based analysis → 4 features

5.5.1.9 Wavelet Features

- Using PyWavelets with db4 wavelet and 4-level decomposition → 64+ statistical descriptors

5.5.1.10 Jitter, Shimmer, HNR (Parselmouth / Praat)

- Voice quality metrics, extracted with parselmouth → $3 \times 2 = 6$ features

5.5.1.11 Formant Frequencies (F1-F4)

- Reflect resonant frequencies of the vocal tract → $4 \times 2 = 8$ features

5.5.1.12 OpenSMILE – eGeMAPS v01a

- Emotion-rich feature set (~88 features), extracted via external config → integrated to main vector

These features were stored in structured CSV files (meld_features_ml_1.csv, etc.) and used as input for all ML and MLP models. Total dimensionality: ~1480 features per sample before selection.

5.5.2 Visual Representations for Deep Learning CNN

For CNN-based deep learning models, time-frequency visual representations were generated from the audio using librosa, and stored as .png images. These include:

5.5.2.1 Mel Spectrogram Images ($128 \times 128, 224 \times 224$)

- Used as input to CNN models trained from scratch.
- Images created with fixed dB scaling and consistent color mapping.

5.5.2.2 MFCC Images ($128 \times 128, 224 \times 224$)

- 40 MFCCs \times temporal frames, visualized using matplotlib.
- Used in CNNs targeting prosodic and timbral variations.

5.5.2.3 RGB Fusion Spectrograms

- R = MFCC, G = Delta, B = Delta²
- Used for RGB input CNNs (e.g., CNN_RGBFusion)

5.5.2.4 Multi-Input CNN

- Separate Mel and MFCC images passed to different convolutional branches.

All models were trained using these image representations, split into train/val directories, with class-balanced samples (e.g., KMeans undersampling for neutral).

5.5.3 Time-Series Feature Sequences for LSTM-Based Models

For sequence models (e.g., LSTM, BiLSTM, GRU), dynamic features were extracted in time-series format:

Feature Format: [T, 120] where T is padded time steps and 120 is the concatenation of:

- 40 MFCCs
- 40 Delta
- 40 Delta²

Padding or truncation ensured uniform input length. These tensors were used directly as input to RNN-based models.

5.5.4 Transformer-Based Feature Embeddings (SOTA Models)

For pretrained transformer models (e.g., WavLM, HuBERT, Wav2Vec2.0), high-dimensional feature embeddings were obtained using Hugging Face Transformers and Torchaudio:

- Models used:
 - facebook/wav2vec2-large-960h
 - microsoft/wavlm-base-plus / wavlm-large
 - facebook/hubert-base-ls960
- Embeddings were either:
 - **End-to-End Fine-tuned** (classification head trained)
 - **Frozen Feature Extractor + MLP Classifier**

Feature embeddings of shape [T, D] (e.g., [100, 768]) were pooled via mean or passed directly to linear/MLP layers.

Outputs used for: WavLM + MLP, HuBERT + MLP, Wav2Vec2 + MLP.

5.5.5 Feature Selection

Machine learning models benefit greatly from using compact and relevant feature subsets, especially when the input feature space is large and possibly redundant. In this study, after extracting an advanced set of approximately 1480 handcrafted audio features, various feature selection techniques were systematically applied to reduce dimensionality, enhance model generalization, and minimize overfitting.

- **Applied Feature Selection Techniques:**

The following methods were used in both isolation and combination:

- **Variance Thresholding**

Eliminated features with very low variance, which do not contribute meaningful discriminatory power.

- **SelectKBest**

Applied with multiple scoring functions:

- *Mutual Information*
- *Chi-Square*

- **Random Forest Importance**
Used feature importance scores derived from tree-based models to retain top-n relevant features.
- **XGBoost Feature Importance**
Applied using `tree_method='gpu_hist'` for GPU-accelerated selection based on gain and weight scores.
- **Recursive Feature Elimination (RFE)**
Used to recursively eliminate less important features using a base estimator.
- **Sequential Floating Forward Selection (SFFS)**
Greedy selection method that adapts to local optima and interactions between features.
- **Autoencoder-Based Dimensionality Reduction**
Unsupervised deep learning approach to encode original features into lower-dimensional, informative latent vectors (e.g., `ae_0`, `ae_1`, ...).
- **Boruta Algorithm**
A wrapper method that identifies all relevant features based on Random Forest classifiers.
- **Hybrid Selections**
Combined multiple strategies to achieve better performance:
 - *Boruta + RF*
 - *Boruta + Autoencoder*
 - *Boruta + Autoencoder + RF*

Undersampling and Class Balance Considerations

In parallel with feature selection, class imbalance was addressed through KMeans-based undersampling of the dominant class (`male_neutral`). This step ensured that the selected features were not biased toward the most frequent class and improved the robustness of selection algorithms.

Feature Set Generation and Reuse

Each selected feature subset was saved as a separate CSV file under a consistent naming convention (e.g., `meld_features_selected_boruta_rf.csv`, ...`_autoencoder.csv`, etc.), resulting in a total of 11 balanced feature sets. These were subsequently used to train and compare the performance of 9 different ML models (e.g., SVM, RF, XGBoost, k-NN, MLP, etc.).

- The best model performance (up to ~86% accuracy) was observed with hybrid feature selection strategies, especially those enhanced by autoencoder embeddings and tree-based importance filtering.
- During training and inference, each feature set was used along with its corresponding:
 - `StandardScaler (scaler.pkl)`
 - `LabelEncoder (label_encoder.pkl)`
 - Autoencoder encoder model (`.pth`), if applicable.

This modular and reproducible pipeline allowed consistent benchmarking across models and facilitated robust, explainable selection of acoustic features for emotion classification tasks.

5.5.6. Feature Normalization and Label Encoding

To ensure that extracted features and labels were appropriately formatted for machine learning and deep learning models, a comprehensive normalization and encoding strategy was implemented.

- **Feature Normalization**

Many machine learning algorithms (such as Support Vector Machines, Logistic Regression, and k-NN) are sensitive to the scale of input features. Features with varying ranges or units may introduce bias and hinder model convergence. To address this:

- **StandardScaler** from scikit-learn was applied to transform all features across the dataset:
 - Mean = 0
 - Standard Deviation = 1

- This normalization was applied after feature extraction and selection, but before model training.
- The same scaler was also used for deep learning models that utilized dense numerical vectors (e.g., Torch-MLP, Keras-MLP, LSTM with handcrafted features).
- In contrast, CNN-based models trained on Mel spectrograms or MFCC images did not require such scaling, as image intensities were normalized during spectrogram generation.

To ensure consistency during prediction/inference, the fitted StandardScaler was saved as a .pkl file. This allowed exact transformation parameters to be reused across both batch and GUI-based inference.

- **Label Encoding**

Emotion labels were prepared in two encoding formats to suit the requirements of different model types:

- LabelEncoder was used to generate integer-encoded labels (e.g., anger → 0, joy → 1, etc.) for traditional ML models and for PyTorch deep models using CrossEntropyLoss.
- OneHotEncoder was applied to create one-hot encoded vectors for Keras-based deep learning models requiring categorical outputs (e.g., softmax activation in final layers).

The label encoding objects were also saved (label_encoder.pkl, onehot_encoder.pkl) and reloaded during inference or GUI-based evaluation to enable consistent mapping between predicted classes and human-readable emotion labels.

5.6 Model Training and Evaluation

In this study, classical machine learning, deep learning, and state-of-the-art (SOTA) transformer-based models were employed to perform speech-based emotion recognition. The primary objective was to determine which model type and feature representation yielded the highest classification performance on speech utterances from the MELD dataset, particularly in a speaker-specific setup focused on the character Joey. Each model was rigorously trained and evaluated using standardized experimental settings and comprehensive acoustic feature sets.

The comparison encompassed three modeling categories:

- Classical ML Models – based on statistical learning methods and hand-crafted features.
- Deep Learning Models – including CNNs, RNNs, and MLP architectures trained on spectrograms or extracted features.
- SOTA Transformer Models – leveraging pre-trained self-supervised learning (SSL) models such as WavLM, HuBERT, and Wav2Vec2.0, either as feature extractors or via fine-tuning.

The upcoming subsections describe the data preparation steps, modeling strategies, and evaluation results for each model type in detail.

5.6.1. Data Preparation

5.6.1.1. Machine Learning Models

For machine learning models, a systematic data preparation pipeline was established to ensure optimal feature quality and compatibility across classifiers. The following steps were followed:

- **Train-test split:** The dataset was divided into 80% training and 20% testing subsets using `train_test_split()` with the `stratify` parameter set to the emotion labels. This ensured proportional representation of each emotion class in both subsets.
- **Label encoding:** For ML models, class labels (e.g., 'anger', 'joy') were encoded into numerical values using `LabelEncoder` from Scikit-learn, resulting in label vectors `y_train` and `y_test`. These encoded labels were used for classification tasks in all scikit-learn-based and PyTorch MLP models.
- **Feature scaling:** All extracted features were normalized using `StandardScaler` to ensure zero mean and unit variance. This step was essential for distance-based classifiers such as SVM and K-NN to prevent feature dominance due to differing scales.
- **Feature extraction:** A single unified feature extraction pipeline was developed for ML models, using `extract_advanced_ml_features()`. This function extracted over 1480-dimensional feature vectors from audio files and included:
 - Time-domain features (ZCR, RMS)
 - Spectral features (centroid, rolloff, bandwidth, contrast)
 - MFCCs + Delta/Delta-Delta
 - Pitch, jitter, shimmer, HNR, formants (Parselmouth)
 - Wavelet transforms (pywt)

- TEO, DFT features
- OpenSMILE (eGeMAPS v01a) features
- **Feature selection:** Since raw features were high-dimensional ($\approx 1480+$), 11 different feature sets were created through distinct feature selection and balancing strategies:
 1. meld_features_ml_1_balanced_cleaned.csv → All cleaned features (NaN removed, balanced)
 2. meld_features_selected_boruta_1_balanced.csv → Boruta-selected features
 3. meld_features_selected_kbest_1_balanced.csv → SelectKBest (Mutual Info)
 4. meld_features_selected_rf_1_balanced.csv → RandomForest Feature Importance (Top N)
 5. meld_features_selected_rfe_1_balanced.csv → Recursive Feature Elimination
 6. meld_features_selected_sffs_1_balanced.csv → Sequential Forward Floating Selection
 7. meld_features_selected_autoencoder_1_balanced.csv → Autoencoder latent vectors
 8. meld_features_selected_boruta_rf_1_balanced.csv → Boruta \cap RF Importance intersection
 9. meld_features_selected_boruta_autoencoder_1_balanced.csv → Boruta → Autoencoder
 10. meld_features_selected_boruta_autoencoder_rf_1_balanced.csv → Boruta → Autoencoder → RF
 11. meld_features_ml_1_clean_balanced.csv → Cluster-balanced version of full features

Each .csv file corresponds to a unique feature representation used in the model training and evaluation phase.

- **ML Models Used:** A total of 9 classical ML models were trained and evaluated on all 11 feature sets:

1. Support Vector Machine (SVM)
2. Random Forest (RF)
3. XGBoost
4. K-Nearest Neighbors (KNN)
5. sklearn MLPClassifier (shallow MLP)

6. Gradient Boosting (sklearn)
7. BaggingClassifier (with DT or KNN)
8. Hard Voting Classifier
9. Soft Voting Classifier

For each model:

- The corresponding CSV file was read.
- Labels were encoded using LabelEncoder() and saved as .pkl.
- Features were normalized using StandardScaler() and saved as .pkl.
- Train-test split was applied with an 80-20 stratified partition.
- Models were trained using GridSearchCV with cross-validation for optimal hyperparameter tuning.
- The best-performing model per feature set was saved as .pkl under:
 - features_and_models/ml_models/<model_name>/
- All trained models were evaluated using metrics such as Accuracy, F1-Score (macro), Classification Report, and Confusion Matrix.

This setup enabled consistent evaluation across different classifiers and feature subsets, allowing for reliable comparison of performance.

5.6.1.2. Deep Learning Models

For deep learning models, the dataset preparation and preprocessing stages varied depending on the model architecture and input modality. Two primary input types were used: (1) spectrogram-based visual representations for CNN architectures, and (2) numerical acoustic feature vectors for MLP and RNN models. The following preparation steps were conducted:

- Audio Preprocessing: All audio files were resampled to 16kHz, denoised using noisereduce, normalized (RMS), and trimmed of silence. Short files were padded to ensure consistent length. All preprocessing was applied using librosa, pyannote, and other custom utilities.
- Spectrogram Generation (for CNN models):
 - Mel Spectrograms: 128×128 grayscale Mel spectrograms were generated using librosa.

- MFCC Spectrograms: RGB representations where R=MFCC, G=delta, B=delta².
 - Multi-Input: Parallel generation of MFCC and Mel images for multi-branch CNN input.
 - ResNet/EfficientNet inputs: 224×224 resized RGB images were generated for transfer learning.
 - These images were split into train/val folders using stratified sampling and class balancing (e.g., PCA + KMeans undersampling for “neutral” class).
- Numerical Feature Vectors (for MLP and RNN):
 - A shared feature extraction pipeline (extract_advanced_ml_features) was used, identical to the one used in ML models, yielding ≈1480 features.
 - For each DL model using feature vectors (e.g., Torch-MLP, BiLSTM), the corresponding .csv file was selected.
 - Autoencoder-based dimensionality reduction was also applied to generate compact latent vectors for some MLP models.
- Label Encoding: All emotion labels were encoded using LabelEncoder() and stored as .pkl.
- Feature Normalization: When numerical features were used, StandardScaler normalization was applied and saved as .pkl.
- Model Saving: All best-performing models were saved using either .pt (Torch), .h5 (Keras), or .pth formats. Corresponding encoders and scalers were also stored for inference.

This dual-track preparation ensured that each DL architecture received appropriately preprocessed and standardized inputs, enabling consistent training and fair evaluation.

5.6.1.3. Sota Models

In addition to classical machine learning and deep learning approaches, this study employed state-of-the-art (SOTA) transformer-based models for speech emotion recognition. These models leverage self-supervised learning (SSL) architectures trained on massive unlabeled speech corpora and have demonstrated exceptional performance in a variety of downstream speech tasks. The implementation in this study was divided into

two major strategies: Pretrained Feature Extraction + Frozen Classifier, and End-to-End Transformer Fine-Tuning.

A) Pretrained Feature Extraction + Frozen Classifier

In this approach, transformer-based SSL models such as WavLM, Wav2Vec2.0, and HuBERT were used as frozen feature extractors. No fine-tuning was applied to these pretrained models; instead, embedding vectors were extracted from raw audio and passed to a separately trained Multi-Layer Perceptron (MLP) classifier for emotion prediction. This approach allows leveraging the generalization capabilities of pretrained representations while keeping training computationally efficient.

- Audio Preprocessing: All input audio clips were preprocessed with the same pipeline used in ML and DL models (resampling to 16kHz, silence trimming, noise reduction, RMS normalization).
- Feature Extraction Pipeline:
 - Each audio file was passed through the selected HuggingFace SSL model (WavLM, Wav2Vec2, or HuBERT).
 - A single vector representation was extracted, typically by applying mean-pooling or statistical aggregation over the final hidden states.
 - The extracted vectors (usually 768-1024 dimensions depending on the model) were stored in .csv files with corresponding emotion labels.
- Classification:
 - A separate MLP classifier (with dense layers and dropout) was trained using these fixed embeddings as input.
 - The MLP was trained with cross-entropy loss and early stopping on validation accuracy.
- Label Encoding & Scaling:
 - Emotion labels were encoded using LabelEncoder() and saved as .pkl files.
 - The extracted embedding vectors were normalized with StandardScaler() and stored for inference reuse.
- Model Saving:
 - The best-performing MLP models were saved in .pt or .pkl format under features_and_models/sota_models_pretrained/.

The pretrained models used in this configuration were:

Table 5.17 Sota Pretrained-frozen Models

Model Name	HuggingFace Identifier	Feature Dim
WavLM-large	microsoft/wavlm-large	1024
WavLM-base-plus	microsoft/wavlm-base-plus	768
Wav2Vec2-large	facebook/wav2vec2-large-960h-lv60	1024
Wav2Vec2-SER	ehcalabres/wav2vec2-lg-xlsr-en-speech-emotion-recognition	1024
HuBERT-large	facebook/hubert-large-ls960-ft	1024
HuBERT-base	facebook/hubert-base-ls960	768

This strategy allowed leveraging high-quality, semantically-rich speech embeddings with minimal computational burden.

B) End-to-End Fine-Tuned Transformers

In this approach, raw audio waveforms were directly fed into transformer models that were fine-tuned end-to-end for emotion classification. Unlike the frozen-extractor method, all parameters of the model were updated during training using backpropagation. This allowed the transformer to adapt its representations specifically for emotion recognition in the MELD dataset.

- Input Representation:
 - Raw audio files were passed directly into the model (no feature extraction or spectrogram generation was needed).
 - Models were trained using HuggingFace's Trainer API or PyTorch training loop.
- Model Architecture:
 - Pretrained backbone: WavLM / Wav2Vec2 / HuBERT base models.
 - Classification head: A task-specific dense layer was added on top of the transformer output, often pooling the final hidden states (e.g., via mean pooling or using [CLS] token).

- Training Details:
 - Optimizer: AdamW with learning rate scheduling and weight decay.
 - Early stopping and checkpointing were applied on validation loss or macro-F1 score.
 - Models were trained for up to 25–30 epochs depending on convergence.
- Label Mapping:
 - HuggingFace-compatible id2label and label2id dictionaries were constructed and saved for consistent decoding.
- Evaluation:
 - After training, models were evaluated on the full test set using Accuracy, F1-score, and confusion matrices.
 - Predictions were compared against ground-truth emotion labels.
- Model Saving:
 - The fine-tuned models and feature extractors were saved using model.save_pretrained() and feature_extractor.save_pretrained().
 - Saved under features_and_models/sota_models_transformer/<model_name>/.

The fine-tuned transformer models used were:

Table 5.18 Sota End-to-End Models

Model Name	HuggingFace Identifier	Strategy
WavLM (Fine-tuned)	microsoft/wavlm-base-plus	PyTorch Trainer
HuBERT (Fine-tuned)	facebook/hubert-base-ls960	HuggingFace
Wav2Vec2.0 (Fine-tuned)	facebook/wav2vec2-base	HuggingFace

This fully trainable strategy provided more task-specific tuning at the cost of increased training time and memory requirements. Among the fine-tuned models, Wav2Vec2.0 showed the best overall performance, while WavLM showed reduced recall due to overfitting in certain emotion classes.

Summary:

Both SOTA modeling strategies significantly enriched the overall system. The pretrained feature extraction approach was lightweight and effective, especially with

WavLM-large. On the other hand, end-to-end fine-tuned models allowed direct learning from raw waveforms and achieved high accuracy when trained carefully. These models demonstrate the importance of leveraging SSL models in modern speech emotion recognition pipelines.

5.6.2. Models Used

5.6.2.1. Classical Machine Learning Models

A total of 9 classical machine learning models were trained and evaluated on the Joey-specific subset of the MELD dataset. Each model was tested across 11 distinct feature sets derived from handcrafted acoustic features. Performance was assessed using Accuracy, F1-score (macro), and confusion matrices. GridSearchCV was used to optimize hyperparameters for each classifier. The evaluation results and key observations are summarized below:

1. Support Vector Machine (SVM):

- Kernels: poly, C=0.1, gamma=0.1 (tuned with GridSearchCV)
- Best performance:
 - **Accuracy:** 86.43%
 - **F1-macro:** 86.51%
 - **Feature set:**
`meld_features_selected_boruta_autoencoder_1_balanced.csv`
- Observations: SVM consistently performed well across high-quality feature sets and was robust to dimensionality. Accuracy degraded on under-selected or noisy feature subsets.

2. Random Forest (RF):

- Ensemble of 100–200 decision trees, with `max_depth`, `max_features`, and `min_samples_split` optimized.
- Best performance:
 - **Accuracy:** 80.24%
 - **F1-macro:** 79.48%

- **Feature set:** meld_features_selected_rfe_1_balanced.csv
- Observations: RF benefited from feature selection and balanced datasets. Performance was stable across most feature sets.

3. XGBoost:

- Tree-based gradient boosting with GPU acceleration enabled (tree_method='gpu_hist').
- Best performance:
 - **Accuracy:** 82.86%
 - **F1-macro:** 82.53%
 - **Feature set:**
meld_features_selected_boruta_autoencoder_1_balanced.csv
- Observations: XGBoost provided robust generalization and was effective on both high- and low-dimensional sets. Strong resistance to overfitting.

4. K-Nearest Neighbors (KNN):

- GridSearch optimized k, metric, and weights.
- Optimal K: 3, Metric: Euclidean, Weights: Distance
- Best performance:
 - **Accuracy:** 82.14%
 - **F1-macro:** 81.77%
 - **Feature set:**
meld_features_selected_boruta_autoencoder_1_balanced.csv
- Observations: KNN performed surprisingly well on densely informative feature sets. Distance-weighted voting improved minority class predictions.

5. sklearn MLPClassifier:

- Shallow neural network trained via 'adam' optimizer and relu activation.
- Hidden Layers: (256, 128), Activation: ReLU, Solver: Adam

- Best performance:

 - **Accuracy:** 80.24%

 - **F1-macro:** 79.52%

 - **Feature set:** meld_features_selected_boruta_1_balanced.csv

- Observations: MLP benefited from reduced feature sets and balanced training data. Overfitting observed on full feature sets.

6. Gradient Boosting (sklearn):

- Used log_loss as objective, with fine-tuned tree depth and learning rate.

- Best performance:

 - **Accuracy:** 83.57%

 - **F1-macro:** 83.46%

 - **Feature set:** meld_features_selected_boruta_1_balanced.csv

- Observations: Gradient Boosting outperformed RF on some feature sets due to its sequential learning capability. Performed best on curated feature subsets.

7. Bagging Classifier (with Decision Tree or KNN base):

- Best configuration: DecisionTreeClassifier(max_depth=5) with n_estimators=150

- Best performance:

 - **Accuracy:** 83.33%

 - **F1-macro:** 83.23%

 - **Feature set:** meld_features_selected_boruta_1_balanced.csv

- Observations: Bagging helped reduce variance and improved model stability. Performed well on moderately complex feature sets.

8. Hard Voting Classifier:

- Majority voting ensemble of SVM, RF, and XGBoost

- Best performance:

- **Accuracy:** 84.29%
- **F1-macro:** 84.09%
- **Feature set:** meld_features_selected_boruta_1_balanced.csv

- Observations: Combining diverse learners led to consistent improvements over individual models.

9. Soft Voting Classifier:

- Probability-weighted voting ensemble

- Best performance:

- **Accuracy:** 84.29%
- **F1-macro:** 84.13%
- **Feature set:** meld_features_selected_boruta_1_balanced.csv

- Observations: Performed slightly better than hard voting due to class probability integration.

To handle the high-dimensional feature space (~1480 features), the following feature selection techniques were applied and compared:

- **Boruta:**

A wrapper method built around Random Forest. It iteratively tests the importance of each feature by comparing it to random shadow features, selecting only those with consistently higher importance. It is highly effective for noisy, high-dimensional datasets and was found to yield the best feature subsets across most ML models.

- **SelectKBest (Mutual Information):**

A filter method selecting the top k features with highest mutual information with the label. Fast but may miss complex feature interactions.

- **Random Forest Importance:**

Uses the feature_importances_ attribute from RF classifier to rank features. Captures nonlinear relationships and is computationally efficient.

- **RFE (Recursive Feature Elimination):**
Wrapper method that recursively removes least important features. Effective but time-consuming.
- **SFFS (Sequential Floating Forward Selection):**
A dynamic greedy algorithm that adds and removes features to find a good subset. Can outperform fixed greedy methods in accuracy.
- **Autoencoder:**
An unsupervised deep learning approach that compresses features into a latent space. Useful for nonlinear dimensionality reduction.
- **Hybrid Strategies:**
 - **Boruta + Autoencoder:** First reduces noisy features, then compresses representation.
 - **Boruta + Autoencoder + RF:** Adds an extra filtering based on RF importance over AE-reduced vectors.
 - **Boruta \cap RF:** Intersection of top features from both methods.

Each feature set was tested across all models, and findings confirmed that combining multiple selection strategies significantly boosts performance.

General Trend Observations:

- Feature sets derived via Boruta + Autoencoder or Boruta-only performed best overall.
- Voting-based ensembles achieved the highest macro-level performance by combining diverse perspectives.
- Simpler models (KNN, RF) gained significant benefit from well-balanced and selected features.

In the SVM model, the best-performing classes were *disgust*, *sadness*, and *fear*, while the worst was *joy*. For Random Forest, *disgust* and *fear* performed best, whereas *joy* had the lowest F1-score. XGBoost also showed strong results for *disgust* and *fear*, but again struggled with *joy*. The K-NN model achieved its highest scores on *disgust* and *fear*, with the lowest performance on *neutral*. Sklearn MLP showed the best results for *disgust* and

fear, and the weakest for *joy*. Gradient Boosting performed well on *disgust*, *fear*, and *sadness*, while *joy* was again the least accurate. The Bagging Classifier with KNN performed best on *disgust*, *sadness*, and *surprise*, and worst on *neutral*. In the Hard Voting ensemble, *fear* and *sadness* were top-performing, while *joy* had the lowest score. Finally, in the Soft Voting ensemble, *fear*, *disgust*, and *sadness* were predicted most accurately, while *joy* remained the weakest.

Overall, *disgust*, *fear*, and *sadness* were consistently the most accurately predicted emotion classes across nearly all models, suggesting that their acoustic or statistical features are more distinct and easily learnable. On the other hand, *joy* emerged as the most problematic class, likely due to greater intra-class variation or overlap with other emotions, making it harder for models to differentiate. *Neutral* also showed relatively weak performance in some models.

5.6.2.2. Deep Learning Models

A total of 11 deep learning models were developed and evaluated on the Joey-specific subset of the MELD dataset. These models utilized either image-based representations (such as spectrograms and MFCCs) or feature vectors derived from advanced handcrafted acoustic features. The models spanned across CNN architectures, recurrent networks (BiLSTM), MLPs, and hybrid designs. Each model was trained and tested with optimized parameters, and evaluated using accuracy, F1-score (macro), and confusion matrices. Below are the models and their corresponding performances:

1. CNN – Mel Spectrogram

- Input: 128×128 grayscale Mel spectrogram
- Architecture: 3-layer CNN with ReLU, dropout
- Accuracy: 76.43%
- F1-macro: 76.70%
- Observations:
 - Strongest classes: *disgust* (F1: 0.97), *fear* (0.92), *sadness* (0.88)
 - Weakest classes: *neutral* (0.50), *joy* (0.62)
 - Despite not being the most complex architecture, this CNN effectively captured emotional patterns in time-frequency space.

- Confusion Trends:
 - Neutral is confused with joy and surprise
 - Surprise partially misclassified as neutral
 - Class separation is generally good for disgust, fear, and sadness

2. CNN – MFCC

- Input: 128×128 grayscale MFCC image
- Architecture: 3-layer CNN applied to visualized MFCC coefficients
- Accuracy: 61.19%
- F1-macro: 60.49%
 - Observations:
 - Strongest classes: disgust (F1: 0.83), fear (0.79), sadness (0.73)
 - Weakest classes: neutral (0.38), joy (0.41), surprise (0.49)
 - While disgust and fear were detected with high precision and recall, the model struggled particularly with neutral and joy classes.
- Confusion Trends:
 - Joy is confused with anger and surprise
 - Neutral overlaps with joy and surprise
 - Moderate overfitting observed on well-separated classes, underfitting on ambiguous ones

3. CNN – RGB Fusion (MFCC, Delta, Delta²)

- Input: 128×128 RGB image (MFCC as Red, Delta as Green, Delta² as Blue)
- Architecture: 3-layer CNN processing fused temporal dynamics through color channels
- Accuracy: 63.10%
- F1-macro: 61.72%
- Observations:
 - Strongest classes: disgust (F1: 0.85), fear (0.86), sadness (0.67)
 - Weakest classes: neutral (0.34), joy (0.41)
 - RGB fusion helps temporal variation modeling (via delta features), yet model still struggles with ambiguous emotions like neutral and joy
- Confusion Trends:
 - Joy and neutral are mostly misclassified into anger, surprise, or each other

- Disgust shows near-perfect classification (Recall: 0.98)
- Some underfitting on sadness with class drift toward neutral and anger

4. CNN – Multi-Input (MFCC + Mel Combined)

- Input: Parallel 128×128 Mel Spectrogram and MFCC images (dual CNN branches combined before FC layer)
- Architecture: Two-path CNN with feature fusion at dense layer
- Accuracy: 53.33%
- F1-macro: 52.39%
- Observations:
 - Best classes: disgust (F1: 0.79), fear (0.72)
 - Poor performance: joy (0.32), surprise (0.37), anger (0.43)
 - While dual input improves fear detection via spectral and cepstral synergy, fusion doesn't help much in ambiguous categories like joy and neutral
- Confusion Trends:
 - Joy and surprise are highly confused with each other and misclassified into anger
 - Strong detection for fear and disgust
 - Slight overfitting on sadness → misclassifications toward neutral and surprise

5. ResNet18 – Transfer Learning (Mel Spectrogram, 224x224)

- Input: Resized 224×224 3-channel Mel Spectrogram images (duplicated grayscale or RGB-mapped)
- Architecture: Pretrained ResNet18 (ImageNet) with modified fully connected output layer for 7 emotion classes
- Accuracy: 70.24%
- F1-macro: 69.18%
- Observations:
 - Strongest results in sadness (F1: 0.86), disgust (0.93), and fear (0.82)
 - Weak classes: neutral (F1: 0.37) and joy (F1: 0.49)
 - Model benefits from transfer learning in spectral pattern recognition, especially for highly separable classes like disgust and sadness
- Confusion Trends:

- Anger often confused with sadness and joy
- Neutral performance weak due to overlap with emotional tones and ambiguous prosody
- Fear and disgust almost perfectly classified

6. EfficientNetB0 – RGB Fusion (224x224)

- Input: RGB fused spectrogram images where
 - Red = MFCC
 - Green = Delta
 - Blue = Delta²
- All resized to 224×224 for EfficientNetB0 input.
- Architecture: Transfer learning using EfficientNetB0 pretrained on ImageNet, with final classification layer adapted to 7 emotion classes
- Accuracy: 78.33%
- F1-macro: 78.17%
- Observations:
 - Best performance in sadness (F1: 0.91), fear (0.92), and disgust (0.89)
 - Moderate performance in joy (F1: 0.63) and surprise (0.73)
 - Lowest class-wise success: neutral (F1: 0.55), due to its emotional ambiguity
 - Outperforms traditional CNN models, indicating that fused MFCC representations and deeper architecture provide better generalization
- Confusion Trends:
 - Disgust almost perfectly classified (Recall: 0.98)
 - Joy and Neutral still confused with neighboring emotional tones
 - Model shows strong inter-class separability, especially in high-arousal classes

7. BiLSTM (Handcrafted Feature Input)

- Input:
 - Sequential 2D feature vectors extracted from audio, shaped as [T x 120]
 - (40 MFCC + 40 Delta + 40 Delta² features) for each time step.
 - Time dimension padded or cropped to 200 frames.
- Architecture:

- Two-layer bidirectional LSTM network:
 - First LSTM layer with 128 hidden units
 - Second LSTM layer with 64 hidden units
 - Global Max Pooling across time dimension
 - Fully connected layer with ReLU and Dropout
 - Final dense output layer with 7 emotion classes
- Preprocessing:
 - All audio resampled to 16kHz
 - Z-score normalization over features using StandardScaler
 - Undersampling applied to majority class (neutral) using PCA + KMeans to balance dataset
 - Label encoding applied and saved for inference
- Accuracy: 77.14%
- F1-macro: 77.01%
- Observations:
 - Strong performance in sadness (F1: 0.84), fear (F1: 0.90), and disgust (F1: 0.95)
 - Moderate performance in anger (F1: 0.75) and surprise (F1: 0.75)
 - Lower performance in neutral (F1: 0.58) and joy (F1: 0.61)
 - Temporal modeling via LSTM layers helps capture prosodic and rhythmical features
 - Bidirectional structure improves context understanding across time
- Confusion Trends:
 - Disgust is classified with very high precision and recall (Recall: 0.93)
 - Joy and Neutral are most confused with each other, likely due to acoustic similarity
 - The model struggles slightly in separating low-arousal emotions like neutral and joy, similar to CNN models

8. Keras – BiLSTM (with GRU)

- Input:

- Sequential feature vectors extracted from audio, shaped as [T x 120] (Concatenated 40 MFCC + 40 Delta + 40 Delta²) per frame, padded/cropped to 200 frames.
- Architecture:
 - Two-layer Bidirectional GRU network
 - First layer: Bi-GRU with 32 units + Dropout (0.6)
 - Second layer: Bi-GRU with 16 units + Dropout (0.6)
 - Global Max Pooling
 - Dense layer with 64 units and L2 regularization
 - Output layer: Dense with Softmax activation for 7 emotion classes
 - Model saved in .h5 format
- Preprocessing:
 - Audio resampled to 16kHz
 - MFCC, delta, and delta² features extracted using librosa
 - Each utterance normalized with StandardScaler
 - Undersampling applied to the “neutral” class using PCA + KMeans
 - Dataset split with stratification (80% train, 20% validation)
 - Class weights computed and applied during training
- Accuracy: 58.10%
- F1-macro: 57.00%
- Observations:
 - Good performance on disgust (F1: 0.79), fear (F1: 0.77), and surprise (F1: 0.64)
 - Lower accuracy on joy (F1: 0.33), neutral (F1: 0.48), and anger (F1: 0.49)
 - GRU-based temporal modeling yields modest results, but is prone to overfitting in low-resource settings
 - Slightly less generalization capacity than PyTorch-based BiLSTM model
- Confusion Trends:
 - Fear and disgust show high recall, indicating stable modeling of high-arousal emotions
 - Joy, neutral, and anger often confused with each other

- Model benefits from temporal information, but still struggles with emotionally ambiguous classes

9. PyTorch – MLP

The PyTorch-based Multilayer Perceptron (MLP) model was trained using 11 different feature sets derived from handcrafted features, including autoencoder-based dimensionality reduction. This deep learning model replicates the structure of traditional MLPs but benefits from PyTorch's flexibility and dynamic graph execution.

- Input: Handcrafted feature vectors from 11 different feature sets (e.g., Boruta, Autoencoder, RF-selected features).
- Best Feature Set: `meld_features_selected_autoencoder_1_balanced.csv`
- Architecture: 2 hidden layers (256-128), ReLU activations
 - Input Layer → 256 → ReLU
 - Hidden Layer → 128 → ReLU
 - Output Layer → Softmax
 - Dropout and BatchNorm applied to prevent overfitting
 - Optimizer: Adam
 - Loss Function: Cross-Entropy
 - Early Stopping based on validation loss
- Best performance:
 - Accuracy: 83.10%
 - F1-macro: 82.88%
 - Feature set: `meld_features_selected_boruta_autoencoder_1_balanced.csv`
- Observations:
 - The model demonstrated high generalization capability when autoencoder-selected features were used.
 - Best-performing emotion classes: disgust, sadness, and fear
 - Lower performance observed in joy and neutral classes, consistent with other models.

10. Keras – MLP

- Input: 11 handcrafted feature sets (selected via feature selection methods)
- Architecture: Two hidden layers (256, 128 units) with ReLU activations, Dropout regularization, and Adam optimizer.

- Best performance:
 - Accuracy: 79.05%
 - F1-macro: 78.63%
 - Feature set: meld_features_selected_autoencoder_1_balanced.csv
- Observations:
 - The Keras-based MLP model achieved solid performance using reduced feature sets obtained via autoencoder. While slightly lower than PyTorch MLP and tree-based ML models, it still showed competitive results for deep learning with low-dimensional input.

11. CNN + BiLSTM (Hybrid)

- Input: 128×128 grayscale Mel spectrogram images generated from Joey's preprocessed 16kHz audio
- Architecture:
 - CNN layers extract spatial features (Conv2D + MaxPooling)
 - Features reshaped and passed through a BiLSTM layer to model temporal relationships
 - Dense layers and Softmax for emotion classification (7 classes)
- Accuracy: 61.00%
- F1-macro: 63.00%
- Observations:
 - Stronger performance in disgust (F1: 0.78), fear (0.75), and sadness (0.66)
 - Weaker recognition in joy (F1: 0.42) and neutral (F1: 0.46)
 - Temporal modeling helps but model still struggles with low-arousal or ambiguous emotions
- Confusion Trends:
 - Disgust, fear, and sadness are better distinguished
 - Joy and neutral show notable confusion with each other and neighboring classes
 - Despite BiLSTM's advantage, class imbalance and acoustic similarity reduce precision in low-energy emotions

5.6.2.3. Sota Models

A total of 9 state-of-the-art (SOTA) models based on transformer architectures were implemented and evaluated. These models utilize either pretrained or fine-tuned self-supervised learning (SSL) models, such as WavLM, HuBERT, and Wav2Vec2.0, all of which are known for their ability to capture deep speech representations. Two main approaches were used: (1) feature extraction followed by an external classifier (e.g., MLP), and (2) end-to-end fine-tuning with classification head.

Below are the models, architectures, and their performance summaries:

1. WavLM (Base+) + MLP

- Approach: Feature extraction using the pretrained microsoft/wavlm-base-plus model. The CLS token embeddings are used as input to a custom-trained Multi-Layer Perceptron (MLP) classifier.
- Input: Preprocessed audio waveform (16 kHz), passed through the frozen WavLM model to extract 768-dimensional CLS embeddings.
- Architecture:
 - WavLM Base+ → 768-dim CLS token embedding (frozen)
 - MLP classifier:
 - Hidden Layer 1: 256 → ReLU
 - Hidden Layer 2: 128 → ReLU
 - Output: Softmax (7 emotion classes)
- Accuracy: 80.00%
- F1-macro: 80.00%
- Observations:
 - High performance observed in disgust (F1: 0.97), fear (F1: 0.93), and sadness (F1: 0.90)
 - Lower scores in joy (F1: 0.61) and neutral (F1: 0.60), likely due to acoustic and emotional overlap
 - Despite being a frozen feature-based approach (not fine-tuned), WavLM-Base+ embeddings demonstrate strong generalization on the MELD dataset
- Confusion Trends:

- Joy often misclassified as neutral or surprise
- Neutral overlaps with joy and sadness
- High-arousal emotions like disgust, fear, and sadness are classified with high precision and recall

2. WavLM (Large) + MLP

- Approach: Feature extraction using the pretrained microsoft/wavlm-large model. The resulting 1024-dimensional CLS token embeddings are used as input to a custom MLP classifier.
- Input: Preprocessed audio waveform (16 kHz), passed through the frozen WavLM model to extract 1024-dimensional embeddings.
- Architecture:
 - WavLM Large → 1024-dimensional CLS token embedding (frozen)
 - MLP classifier:
 - Hidden Layer 1: 256 → ReLU
 - Hidden Layer 2: 128 → ReLU
 - Output: Softmax (7 emotion classes)
- Accuracy: 75.00%
- F1-macro: 74.00%
- Observations:
 - Excellent performance in *disgust* (F1: 0.93), *fear* (F1: 0.93), and strong results in *sadness* (F1: 0.86)
 - *Joy* (F1: 0.55) and *neutral* (F1: 0.48) are more challenging due to overlapping acoustic patterns
 - Compared to the Base+ model, WavLM-Large captures richer prosodic features via its deeper architecture, though slightly reduced performance in low-arousal classes is observed
- Confusion Trends:
 - Neutral is often confused with joy and sadness
 - Surprise predictions are occasionally mistaken for joy
 - Perfect recall for *disgust* (1.00) suggests robust recognition of high-arousal expressions

3. Wav2Vec2.0 (Large) + MLP

- Approach: Feature extraction using the pretrained facebook/wav2vec2-large-960h-lv60 model. The CLS token-style pooled embeddings are used as input to a custom-trained MLP classifier.
- Input: Preprocessed audio waveform (16 kHz), transformed into 1024-dimensional embeddings using the frozen Wav2Vec2.0 model.
- Architecture:
 - Wav2Vec2.0-Large → 1024-dimensional CLS token embedding (frozen)
 - MLP classifier:
 - Hidden Layer 1: 256 → ReLU
 - Hidden Layer 2: 128 → ReLU
 - Output: Softmax (7 emotion classes)
- Accuracy: 63.10%
- F1-macro: 61.00%
- Observations:
 - Strong performance in *fear* (Recall: 0.92) and *disgust* (Recall: 0.88), reflecting good detection of high-arousal emotions
 - Weaker performance in *joy* (F1: 0.39) and *neutral* (F1: 0.51), likely due to acoustic overlap and low emotional intensity
 - Compared to WavLM-based models, the Wav2Vec2.0 embeddings capture general prosodic information but struggle with finer emotional subtleties
- Confusion Trends:
 - *Joy* often confused with *neutral* due to shared acoustic patterns
 - *Anger*, *sadness*, and *surprise* exhibit moderate misclassifications with adjacent emotional classes

- *Disgust* and *fear* maintain high precision and recall, demonstrating consistent model confidence in high-arousal contexts

4. Wav2Vec2.0 (Large XLSR) + MLP

- Approach: Feature extraction using the pretrained ehcalabres/wav2vec2-lg-xlsr-en-speech-emotion-recognition model. The CLS-style pooled embeddings are passed into a frozen MLP classifier trained on a general SER corpus.
- Input: Preprocessed audio waveform (16 kHz), transformed into 1024-dimensional embeddings using the frozen Wav2Vec2.0-Large XLSR model.
- Architecture:
 - Wav2Vec2.0-Large XLSR → 1024-dimensional CLS token embedding (frozen)
 - Pretrained MLP classification head (frozen)
 - Output: Softmax (7 emotion classes)
- Accuracy: 55.00%
- F1-macro: 53.00%
- Observations:
 - Performs well on high-arousal classes like fear (F1: 0.79) and disgust (F1: 0.79)
 - Poor generalization in low-arousal emotions: neutral (F1: 0.29), joy (F1: 0.38), and surprise (F1: 0.38)
 - Model is not fine-tuned on MELD, which limits its ability to capture dataset-specific emotional cues
- Confusion Trends:
 - Neutral frequently misclassified as sadness or joy
 - Surprise is often confused with fear and joy
 - Joy lacks acoustic distinctiveness, leading to overlaps with neutral

5. HuBERT (Large) + MLP

- Approach: Emotion classification pipeline using features extracted from the pretrained facebook/hubert-large-ls960-ft model. CLS-style pooled embeddings are passed to a custom-trained 2-layer MLP classifier. The HuBERT model remains frozen during training.
- Input: Preprocessed audio waveform (16 kHz), fed into the frozen transformer model to extract 1024-dimensional CLS token embeddings.
- Architecture:
 - HuBERT-Large → 1024-dimensional CLS token embedding (frozen)
 - MLP classifier:
 - Hidden Layer 1: 256 → ReLU
 - Hidden Layer 2: 128 → ReLU
 - Output Layer: Softmax (7 emotion classes)
- Accuracy: 69.05%
- F1-macro: 69.00%
- Observations:
 - Strong performance in high-arousal classes like disgust (F1: 0.89), fear (F1: 0.82), and sadness (F1: 0.72)
 - Moderate results in joy (F1: 0.54) and neutral (F1: 0.55), possibly due to overlapping acoustic prosody and low intensity
 - Compared to other SSL models, HuBERT captures clear speech segments and temporal patterns, leading to robust class separation especially in emotional extremes
- Confusion Trends:
 - Disgust achieves near-perfect classification (Recall: 0.87)
 - Neutral is sometimes confused with sadness and joy

- Surprise overlaps with joy and fear, likely due to short and abrupt prosodic changes

6. HuBERT (Base) + MLP

- Approach: Emotion classification pipeline using features extracted from the pretrained facebook/hubert-base-ls960 model. CLS-style pooled embeddings are passed to a custom-trained 2-layer MLP classifier. The HuBERT model remains frozen.
- Input: Preprocessed raw audio waveform (16 kHz), fed into the frozen HuBERT-Base model to obtain 768-dimensional CLS token embeddings.
- Architecture:
 - HubERT-Base → 768-dimensional CLS token embedding (frozen)
 - MLP classifier:
 - Hidden Layer 1: 256 → ReLU
 - Hidden Layer 2: 128 → ReLU
 - Output Layer: Softmax (7 emotion classes)
- Accuracy: 76.19%
- F1-macro: 76.00%
- Observations:
 - Highest F1 scores in fear (0.91), disgust (0.91), and sadness (0.89) suggest excellent recognition of strong and consistent emotional signals
 - Moderate performance in joy (F1: 0.60) and neutral (F1: 0.56), which are more susceptible to acoustic ambiguity and emotional subtlety
 - HuBERT-Base shows competitive performance despite having a smaller architecture than its Large counterpart, making it efficient for real-time applications
- Confusion Trends:
 - Disgust, fear, and sadness are very well captured with recall values ≥ 0.90

- Neutral often confused with joy and sadness
- Surprise misclassifications with joy and fear occur occasionally due to overlapping pitch patterns and energy dynamics

7. HuBERT (Base) – End-to-End Fine-Tuned

- Approach: This model directly fine-tunes the pretrained facebook/hubert-base-ls960 transformer on the MELD dataset for 7-class emotion recognition. Unlike the earlier MLP-based approaches that froze the transformer and trained only a lightweight classifier, this model allows the full transformer to adapt to the MELD-specific emotional patterns through supervised training.
- Input: Raw audio waveform (16 kHz) from the Joey subset is used as input. Audio is tokenized using the Wav2Vec2FeatureExtractor (same as for HuBERT) and padded to a maximum of 16 seconds.
- Architecture:
 - Encoder: facebook/hubert-base-ls960 (12-layer transformer, 768 hidden size)
 - Classification Head: A randomly initialized linear layer is appended and trained for 7 emotion classes.
- Evaluation Results:
 - Test Accuracy: 66.43%
 - Macro F1-Score: 66%
 - Weighted F1-Score: 65%
- Observations:
 - The model demonstrates exceptional performance on high-arousal and negative classes like disgust, fear, and sadness, achieving F1-scores above 0.83.
 - Emotions like joy and neutral perform poorly, especially joy, which suffers from both low recall (0.12) and precision (0.30). This aligns with its often ambiguous acoustic cues.

- Surprise shows high recall (0.80) but only moderate precision (0.44), indicating that while the model detects many surprise instances, it also confuses other classes as surprise.
- Confusion Trends:
 - Joy is frequently misclassified as neutral and surprise, likely due to overlapping prosodic features.
 - Neutral is sometimes confused with sadness, suggesting acoustic subtlety between them.
 - Surprise is confused with fear and joy, indicating difficulty in capturing its distinctiveness in speech-only input.

8. WavLM (Base-Plus) – End-to-End Fine-Tuned

- Approach: This model directly fine-tunes the pretrained microsoft/wavlm-base-plus transformer on the MELD dataset. Unlike frozen feature extraction methods followed by shallow MLP classifiers, this approach enables the model to fully adapt its internal representations to emotion-specific patterns by updating all layers during training.
- Input: Raw audio waveform (16 kHz) extracted from the Joey subset. Tokenization and padding are handled via Wav2Vec2FeatureExtractor, identical to the setup used for HuBERT and Wav2Vec2 models. Audio segments are padded to a maximum of 16 seconds.
 - Architecture:
 - Encoder: microsoft/wavlm-base-plus (12-layer transformer, 768 hidden size, ~94M parameters)
 - Classification Head: A randomly initialized dense layer with 7 output units (softmax), trained from scratch
- Evaluation Results:
 - Test Accuracy: 54.52%
 - Macro F1-Score: 54%

- Weighted F1-Score: 54%
- Observations:
 - Disgust remains the most robust class with $F1 = 0.92$, demonstrating that high-arousal negative emotions are better captured by the model.
 - Although fear has perfect precision, its low recall (32%) indicates under-detection, possibly due to overlapping prosodic patterns with surprise.
 - Joy, neutral, and surprise continue to perform poorly — consistent with prior models — due to their subtle acoustic variations and potential confusions.
 - The overall accuracy is moderate, indicating that fine-tuning alone may not sufficiently align model embeddings with MELD-specific emotion boundaries.
- Confusion Trends:
 - Anger is frequently misclassified as joy or neutral, especially in cases where loudness overlaps.
 - Fear is often mistaken for surprise (28 samples), indicating prosodic ambiguity in expressions of alarm versus excitement.
 - Joy is confused with anger, neutral, and surprise, revealing its broad acoustic variance.
 - Surprise itself is often misclassified as joy or neutral, due to shared pitch dynamics.

9. Wav2Vec2 (Base) – End-to-End Fine-Tuned

- Approach: This model fine-tunes the facebook/wav2vec2-base transformer directly on the MELD dataset for 7-class speech emotion recognition. Unlike feature extraction-based pipelines that freeze the transformer layers, this model

adapts the entire encoder through supervised training, allowing deeper emotional alignment with MELD speech characteristics.

- Input: Raw audio waveform (16 kHz) from the Joey subset. Tokenization and padding are handled using Wav2Vec2FeatureExtractor. Utterances are padded or trimmed to a maximum of 16 seconds for uniformity during batch training.
- Architecture:
 - Encoder: facebook/wav2vec2-base (12-layer transformer, 768 hidden size, ~95M parameters)
 - Classification Head: A single dense layer trained on top of the [CLS] token to classify into 7 emotion categories.
- Evaluation Results:
 - Test Accuracy: 77.86%
 - Macro F1-Score: 77%
 - Weighted F1-Score: 77%
- Observations:
 - Fear, disgust, and sadness are consistently recognized with high accuracy and F1 scores above 0.85, showing that the model effectively captures emotional markers for high-arousal or negative-valence classes.
 - Surprise shows very high precision (0.95) but low recall (0.57), suggesting it is predicted only when highly confident, missing some true samples.
 - Joy continues to show limited performance ($F1 = 0.53$), reflecting its acoustic overlap with other positive emotions like neutral.
 - Overall, this model achieves one of the highest performances among all transformer-based models in this study, outperforming both WavLM and HuBERT in balanced recognition of most classes.
- Confusion Trends:
 - Joy is confused most with neutral (15 instances), supporting the theory of ambiguous prosodic features.

- Surprise is often misclassified as joy (13) or neutral (8), reflecting acoustic overlaps.
- Neutral is well captured overall but has minor confusion with joy and sadness.

5.6.3. Evaluation Metrics

To assess the performance of all machine learning (ML), deep learning (DL), and state-of-the-art (SOTA) models implemented in this study, several standard evaluation metrics were utilized:

- **Accuracy:** Measures the overall correctness of predictions by calculating the ratio of correctly predicted instances to the total number of predictions. While widely used, accuracy alone can be misleading in imbalanced datasets.
- **Precision:** For each emotion class, precision represents the ratio of correctly predicted instances of that class to all instances predicted as that class. It is crucial when false positives are costly.
- **Recall (Sensitivity):** Recall indicates how well the model identifies actual instances of each emotion class, computed as the ratio of true positives to all actual instances of that class. High recall is important when minimizing false negatives is critical.
- **F1-Score:** The harmonic mean of precision and recall, providing a balanced measure that accounts for both false positives and false negatives. F1-scores were calculated per class as well as using macro and weighted averages.
- **Confusion Matrix:** A tabular representation that visualizes the distribution of predictions across actual emotion classes, helping to analyze misclassification patterns and inter-class confusion.

Given the inherent **class imbalance** in the MELD dataset (especially for minority classes like *disgust* or *fear*), the **macro-averaged F1-score** was used as a primary evaluation metric. This metric treats all classes equally regardless of their sample sizes, offering a more balanced reflection of model performance across emotions.

5.6.4. Results

Throughout the evaluation phase, all models—ranging from classical machine learning (ML) classifiers to deep learning (DL) architectures and state-of-the-art (SOTA) pretrained transformer models—were assessed on the test subset of the MELD dataset (Joey-only, audio modality).

The best overall performance was achieved by the Wav2Vec2.0 (facebook/wav2vec2-base) transformer model, which reached a test accuracy of 77.86% and a macro-averaged F1-score of 77%. Similarly, other pretrained models like HuBERT (base) and WavLM (large) also yielded strong results, with macro F1-scores above 69%.

Among the deep learning models, the EfficientNetB0 + RGB fusion CNN reached an accuracy of 78.33%, showcasing the potential of combining spectro-temporal features in vision-based DL pipelines. Other CNN variants (e.g., MFCC-CNN, Mel-CNN, CNN-BiLSTM) also provided robust performance ranging between 65%–72% accuracy.

In the classical ML domain, the best-performing model was the Soft Voting Classifier, which attained 84.29% accuracy and 84.13% macro F1-score using features selected via the Boruta method. Support Vector Machines (SVM) and XGBoost also demonstrated reliable performance on specific feature sets, with F1-scores around 86% and 82%, respectively.

Despite initial class imbalance in the dataset, the application of data augmentation, feature selection, and speaker filtering substantially improved the robustness of models, especially in minority classes such as *disgust* and *fear*. Confusion matrix analyses revealed that low-arousal emotions like *joy* and *neutral* were more challenging to distinguish, often misclassified due to their prosodic similarities with other classes.

In summary:

- SOTA models (transformer-based) generalized well across most emotion categories.
- DL models leveraging spectrogram representations benefited from CNN architectures and performed competitively.
- ML models achieved strong results when paired with effective feature selection techniques such as Boruta and Autoencoder.

5.6.5. Training Time and Hardware Setup

The training and evaluation procedures were conducted on both CPU and GPU environments, depending on the model type and implementation platform:

- Classical Machine Learning (ML) models such as SVM, Random Forest, XGBoost, and ensemble classifiers were trained using Scikit-learn in a Jupyter Notebook environment on a CPU-only Windows machine. Training times varied significantly depending on the model complexity and the feature selection method used, ranging from as low as 2 seconds (for simple models with limited features) to over 800 seconds (for complex pipelines such as Boruta + Autoencoder + Random Forest). GridSearchCV was utilized to optimize hyperparameters across all feature sets.
- Deep Learning (DL) models, particularly CNN-based architectures (e.g., Mel-CNN, MFCC-CNN, RGB-CNN, EfficientNetB0), were implemented in Jupyter Notebook using PyTorch and Keras (TensorFlow backend). These models were trained on a GPU-enabled environment, with training times typically ranging from 5 to 15 minutes, depending on the architecture, input representation (e.g., 224×224 RGB spectrograms), and dataset size.
- State-of-the-art (SOTA) transformer-based models such as Wav2Vec2.0, HuBERT, and WavLM were implemented using the Hugging Face Transformers library. These models were trained and evaluated on Google Colab, leveraging GPU acceleration due to their high memory and compute requirements. Training durations varied between 30 minutes and 3 hours, especially for end-to-end fine-tuning of larger transformer models.

To ensure experimental reproducibility, the following strategies were applied across all model types:

- Fixed random seeds using `random_state`, `numpy.random.seed()`, and `torch.manual_seed()`
- Consistent preprocessing pipelines and stratified train-validation-test splits
- Saving and versioning of all trained models, scalers, label encoders, and hyperparameter configurations

This hybrid setup of CPU- and GPU-based workflows enabled a comprehensive and efficient exploration of various algorithms, while maintaining reliability, reproducibility, and experimental integrity.

5.6.6. Overall Assessment

This study focused exclusively on the audio modality, which inherently limits the semantic richness available for emotion recognition. Despite this challenge, the results demonstrate that meaningful emotional patterns can still be captured from speech alone.

Acoustic features such as MFCCs, mel spectrograms, pitch-related descriptors, and spectral characteristics provided valuable insights into the emotional tone of utterances. While these features are effective at capturing prosodic and paralinguistic cues, they cannot fully represent contextual or semantic meaning—unlike text or visual modalities.

Nonetheless, several models achieved notable classification performance:

- The best deep learning model (CNN + BiLSTM with mel spectrogram input) achieved 69% accuracy.
- Among pretrained transformer-based models, Wav2Vec2.0 and HuBERT variants exceeded 76% accuracy, benefiting from learned representations of large-scale speech corpora.
- The top classical ML pipelines, using advanced feature selection methods (e.g., Boruta + Autoencoder), reached over 86% accuracy on certain feature sets.

These results validate the robustness of the pipeline and underscore the potential of audio-only emotion recognition, especially when supported by effective preprocessing, augmentation, and feature engineering.

Importantly, the framework developed in this study lays a solid foundation for future multimodal extensions. Integrating text (transcriptions) and visual (facial) modalities can enhance semantic understanding and disambiguate emotions that are acoustically similar, leading to higher overall classification accuracy and real-world applicability in human-computer interaction systems.

5.7. Experimental Results and Discussion

In this section, we present a comprehensive evaluation of all developed models — encompassing classical machine learning (ML), deep learning (DL), and state-of-the-art (SOTA) transformer architectures — on the Joey-specific subset of the MELD dataset. The evaluation is based on accuracy and macro-averaged F1-score metrics, while confusion matrices and class-wise F1 scores are used to identify strengths and weaknesses across emotion classes. Key insights are categorized into five major themes: overall model performance, emotion-wise trends, the impact of feature engineering and input modalities, confusion patterns, and general observations.

Among the classical ML models, Support Vector Machine (SVM) achieved the best overall performance, reaching an accuracy of 86.43% and a macro F1-score of 86.51%, particularly when trained on features selected via Boruta + Autoencoder. Ensemble-based approaches such as Soft Voting and Hard Voting also exhibited strong results, both achieving macro F1-scores exceeding 84%, highlighting the effectiveness of model fusion strategies in emotionally nuanced tasks.

Deep learning models showed varied performance depending on input representation and architectural complexity. The most successful DL model was the PyTorch-based MLP, trained on autoencoder-reduced features, attaining a macro F1-score of 82.88%. CNN models utilizing spectrogram and MFCC inputs were notably effective in detecting high-arousal emotions such as disgust and fear, while models incorporating temporal dynamics (e.g., BiLSTM, CNN-BiLSTM hybrids) showed improved handling of prosodic and rhythmically ambiguous classes like surprise and neutral.

Transformer-based SOTA models, especially those employing end-to-end fine-tuning, offered competitive performance. The best among them, Wav2Vec2.0 (Base), achieved a macro F1-score of 77%, closely followed by HuBERT (Base) and WavLM (Base+). While these models captured high-level acoustic features effectively, their performance often declined for low-arousal and overlapping classes, indicating a need for dataset-specific adaptation.

Across nearly all models, the most consistently and accurately predicted emotions were disgust, fear, and sadness. These emotions exhibit clear acoustic traits such as

distinctive pitch contours, increased energy, or spectral sharpness, which make them more separable both in handcrafted and learned feature spaces.

In contrast, joy emerged as the most challenging emotion to classify. Its acoustic variability, combined with overlaps in prosody with neutral and surprise, led to frequent misclassifications. Even in transformer models, which are known for their representational depth, joy demonstrated the lowest F1-scores. Similarly, neutral exhibited modest performance, particularly in CNN and GRU-based DL models, likely due to its emotionally ambiguous and flat prosody.

Feature engineering played a crucial role in shaping ML and DL performance:

- In classical ML models, Boruta-based feature selection, especially when combined with Autoencoder compression, led to robust generalization by reducing dimensional noise and preserving task-relevant variance.
- For DL models using handcrafted vectors, autoencoder-selected features enhanced performance in MLP and BiLSTM models, balancing expressiveness with overfitting control.
- Spectrogram and MFCC image inputs enabled CNN-based architectures to capture localized time-frequency emotion patterns, while RGB fusion techniques (MFCC + Delta + Delta²) provided a multi-channel perspective for better emotion discrimination.

For SOTA models, the effectiveness of pretrained SSL embeddings (WavLM, HuBERT, Wav2Vec2.0) was clearly evident. Models using frozen embeddings with MLP classifiers delivered strong performance on high-arousal emotions. However, end-to-end fine-tuning yielded better overall adaptability to MELD, albeit with increased risk of overfitting on low-resource classes.

Confusion matrices across all model types revealed consistent misclassification trends:

- Joy and neutral were frequently confused, reflecting their shared acoustic characteristics such as low pitch dynamics and mid-level energy.
- Surprise was occasionally misclassified as fear, particularly in CNN and frozen transformer models, due to overlapping prosodic spikes.

- In contrast, disgust was classified with near-perfect recall in many models, including PyTorch MLP, CNN variants, and WavLM-based architectures. This indicates that disgust's acoustic profile is uniquely identifiable and stable across representations.
- Temporal models (e.g., BiLSTM) and fine-tuned transformers demonstrated better disambiguation of overlapping emotional expressions, especially for surprise, sadness, and fear.

Key findings from the experiments can be summarized as follows:

- SVM and Voting Ensembles yielded the highest classical ML performance when combined with robust feature selection methods like Boruta + Autoencoder.
- DL models achieved their best results with feature-optimized MLPs and deep CNNs using fused spectrogram representations.
- Transformer-based models, particularly those with end-to-end fine-tuning, showed strong class-specific performance but struggled with joy and neutral unless supported by large data and balancing strategies.
- The top-performing emotions were disgust, fear, and sadness, while joy and neutral consistently challenged all architectures.
- The use of hybrid feature selection pipelines (e.g., Boruta + Autoencoder + RF) and class balancing techniques (undersampling, augmentation) substantially improved model robustness and generalization.

5.8. Summary of Implementation

In this chapter, we provided a detailed overview of the end-to-end implementation pipeline for speech emotion recognition using the MELD dataset, focusing exclusively on audio-based modeling. The process began with extensive data preprocessing, including audio extraction, filtering, denoising, resampling, and speaker-specific segmentation (Joey). Data augmentation and class balancing techniques were applied to improve training diversity and address class imbalance.

Advanced feature extraction techniques were implemented to capture diverse acoustic properties, including MFCCs, spectral features, pitch, wavelet transforms, TEO, and

statistical measures. These were further refined through multiple feature selection strategies such as Boruta, RFE, Autoencoder, and hybrid combinations. A total of 11 distinct feature sets were generated to serve classical ML and MLP-based DL models.

Modeling experiments spanned three main paradigms: traditional machine learning (SVM, RF, XGBoost, KNN, ensembles), deep learning (CNNs, BiLSTM, hybrid models), and transformer-based SOTA approaches (WavLM, HuBERT, Wav2Vec2). Each model type was systematically trained, tuned, and evaluated using consistent metrics (accuracy, F1-score, confusion matrix). The entire pipeline was encapsulated in a GUI interface for real-time prediction.

The findings from these implementations form the foundation for the experimental results and comparative analyses presented in the next chapter.

CHAPTER SIX

TEST / EXPERIMENTS

6.1. Test Details

The evaluation of all trained models was conducted on the official test subset of the MELD dataset, focusing exclusively on the character Joey. This test set was not subjected to any augmentation or transformation after preprocessing, ensuring that model performance was measured under fair and consistent conditions. While the development set (dev) was initially intended for validation and tuning, it was not used in the final testing phase due to its limited sample diversity and emotion coverage.

To ensure input consistency between training and testing phases, each model reloaded its respective scaler, label encoder, and—when applicable—autoencoder encoder model from disk. This approach ensured that test features were standardized and transformed using exactly the same settings as those used during training, thereby avoiding data leakage or distribution mismatch.

6.1.1. ML Prediction Pipeline

A dedicated prediction script was developed to evaluate all classical machine learning models across 11 different feature sets. The key steps of the prediction process are as follows:

1. Feature Extraction:

All test .wav files were processed using a shared `extract_advanced_ml_features()` function that computes a wide range of handcrafted acoustic features. These include MFCCs (with delta and delta²), pitch, spectral contrast, ZCR, RMS, wavelet coefficients, TEO, jitter/shimmer, formant frequencies, and openSMILE GeMAPSv01a features.

2. Feature Selection and Transformation:

Depending on the feature set:

- If the model was trained on an autoencoder-based representation, the corresponding autoencoder encoder and scaler were loaded. Features were first reduced to a latent space (e.g., 128 dimensions), and optionally re-filtered using RF-based top feature selection.
- For other feature sets (e.g., Boruta, RFE, RF Importance), selected columns were directly used from the original handcrafted vector.

3. Model and Scaler Loading:

For each model type (SVM, RF, XGBoost, KNN, MLP, etc.), the best model (.pkl), the scaler, and the label encoder were loaded from the appropriate subdirectory under `features_and_models/ml_models/`.

4. Prediction:

The test feature vector was scaled and passed through the model for emotion classification. Soft Voting models used a wrapper that averaged the probabilities from individual classifiers. All results, including predicted label, true label, match status, and class-wise probabilities, were stored.

5. Result Saving:

For every model–feature set combination, results were saved as a .csv file under the `results/ml/` directory structure. Each CSV contained filename, prediction, ground truth, and class probability distribution.

6. Visualization and Evaluation

An additional script was developed to read all result CSVs and compute evaluation metrics:

- Confusion Matrix: Plotted using seaborn for each model–feature set pair, displaying inter-class misclassification patterns.
- Classification Report: Though not printed in every run, detailed metrics such as precision, recall, and F1-score for each emotion class were calculated using `sklearn.metrics.classification_report()`.

7. Summary of ML Testing Setup

- Number of ML model types tested: 9
- Number of feature sets per model: 11

- Total ML models evaluated: 99
- Input: Unmodified test set (.wav) after initial preprocessing
- Output: CSV files containing prediction results and visualized confusion matrices
- Total test files: 264 utterances from Joey subset

6.1.2. DL Prediction Pipeline

A modular prediction framework was developed for evaluating all deep learning (DL) models on the Joey-specific test set. Unlike the classical ML models, DL architectures required distinct preprocessing and input formats (e.g., spectrogram images, MFCC sequences, or raw mel matrices). The overall DL prediction pipeline is outlined below:

1. Input and Dataset Structure:

All test utterances were taken from the Joey-filtered subset of the MELD dataset, preprocessed into .wav files at 16 kHz sampling rate. These audio clips were stored in the directory:

- archive_meld/joey_audio/processed_16KHz/test_splits
- archive_meld/joey_text_final/test_sent_emo_joey_preprocess_with_filename.csv

2. Feature Extraction and Preprocessing:

Each DL model required different input features and preprocessing steps:

- CNN (Mel, MFCC, RGB Fusion):
 - Librosa was used to compute mel-spectrograms and MFCCs. For RGB fusion, MFCC (R), delta (G), and delta² (B) images were created and merged into 3-channel RGB images. These were resized to either 128×128 or 224×224 pixels depending on the architecture (e.g., ResNet, EfficientNet).
- CNN (Multi Input):
 - Separate mel and MFCC images were extracted and fed to different CNN branches before fusion.

- BiLSTM & Keras-BiLSTM:
 - Sequential MFCC features were extracted with 40 coefficients per frame, and augmented with delta and delta² to create a 120-dimensional feature vector per timestep. These sequences were zero-padded to a fixed length (200 frames).
- CNN + BiLSTM:
 - Mel-spectrograms of size (128×128) were passed through a CNN followed by a bidirectional LSTM. Flattened mel matrices were scaled and reshaped for input into the CNN layer.
- Torch-MLP & Keras-MLP:
 - These models reused the ML-style extract_advanced_ml_features() function. Features were selected according to the feature set used during training (e.g., Boruta, Autoencoder). If necessary, autoencoder and scaler transformations were applied.

All models used pre-saved scalers (scaler.pkl) to ensure consistency with training.

3. Model and Label Encoder Loading:

Each DL model was saved using its respective framework:

- Keras Models: .h5 files loaded via load_model()
- PyTorch Models: .pt or .pth files loaded using torch.load() and restored to their corresponding architecture classes.
- LabelEncoder: A shared label_encoder.pkl file was used to decode numeric predictions back to emotion labels. This encoder was saved during training and loaded during inference.

The models were loaded from the features_and_models/dl_models/ subfolders, with separate folders per architecture.

4. Prediction Pipeline:

Each .wav file was processed through the following pipeline:

- Feature extraction specific to the model
 - Scaling (if applicable)
 - Reshaping for CNN/LSTM input (e.g., [batch, 1, 128, 128] or [1, 200, 120])
 - Inference with `model.predict()` or `model(inputs)` for Keras and PyTorch, respectively
 - Prediction probabilities for each class were stored
 - Predicted label was compared against ground truth to compute correctness
- All intermediate results — including class-wise probabilities, predicted label, true label, and match status — were stored for later analysis.

5. Visualization and Evaluation:

For each DL model:

- Confusion Matrix:
 - Generated using `seaborn.heatmap()` based on the CSV outputs. The matrices provided insight into inter-class confusion patterns and the impact of specific architectures.
- Classification Report:
 - Precision, recall, and F1-score for each class were computed using `sklearn.metrics.classification_report()`.

These visualizations were especially helpful in understanding the class-wise performance of models such as BiLSTM, EfficientNet, and CNN-BiLSTM hybrids.

6. Summary of DL Testing Setup

- DL model types evaluated: 11 models (CNN, BiLSTM, MLP, Hybrid, etc.)
- Input format: Raw audio (.wav)
- Feature types: Mel-spectrograms, MFCC, RGB fusion, etc.
- Output: CSV files with predictions and probabilities
- Total test files: 264 utterances from Joey subset

6.1.3. SOTA Prediction Pipeline

This section outlines the inference pipeline used to evaluate state-of-the-art (SOTA) transformer-based models on the Joey-specific subset of the MELD test set. A total of 9 models were used, categorized as follows:

Pretrained + MLP Classifier Models:

- WavLM (microsoft/wavlm-large, wavlm-base-plus)
- Wav2Vec2 (facebook/wav2vec2-large-960h-lv60, ehcalabres/wav2vec2-lg-xlsr-en-speech-emotion-recognition)
- HuBERT (facebook/hubert-large-ls960-ft, hubert-base-ls960)

Fine-tuned Transformer Models (End-to-End):

- WavLM, HuBERT, and Wav2Vec2 (fine-tuned using HuggingFace transformers with custom classification heads)

The prediction pipeline for all SOTA models followed a consistent modular structure:

1. Input and Dataset Structure:

All test utterances were sourced from the Joey-filtered MELD test split and stored in .wav format at 16 kHz sampling rate. Two sources were used for reference and labeling:

- Audio files: archive_meld/joey_audio/processed_16KHz/test_splits/
- Reference CSV with emotion labels:
archive_meld/joey_text_final/test_sent_emo_joey_preprocess_with_filename.csv

2. Feature Extraction and Preprocessing:

Depending on the model category, different approaches were followed:

- Pretrained SOTA Models (Feature Extraction + MLP):

For these models, raw audio signals were passed through frozen transformer encoders (e.g., WavLM, HuBERT, Wav2Vec2) using HuggingFace

AutoModel or AutoFeatureExtractor APIs. Embeddings were extracted and fed into a separately trained multi-layer perceptron (MLP) classifier.

- Example models: microsoft/wavlm-base-plus, facebook/hubert-large-ls960-ft
- Fine-tuned Transformer Models (End-to-End):

These models were directly fine-tuned on the emotion recognition task. The classification head was trained jointly with the transformer encoder (e.g., Wav2Vec2ForSequenceClassification).

- Input audio was resampled to 16 kHz and normalized.
- Wav2Vec2FeatureExtractor or Wav2Vec2Processor was used for tokenization and padding.
- Models used HuggingFace checkpoints and were loaded via from_pretrained().
- Example models: wavlm_model, hubert_model, wav2vec2_model

3. Model and Configuration Loading:

- Transformers-based models were loaded using:
 - Wav2Vec2ForSequenceClassification.from_pretrained(model_dir)
 - HubertForSequenceClassification.from_pretrained(model_dir)
 - Wav2Vec2FeatureExtractor.from_pretrained(model_dir)
- All models were loaded on the appropriate device using:
 - device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device).eval()
- Model configurations (config.id2label) were used to map predicted label indices to emotion strings.

4. Prediction Pipeline:

Each test .wav file was processed as follows:

- Loaded using `torchaudio.load()`
- Resampled (if necessary) to 16 kHz
- Converted to mono channel by averaging if stereo
- Processed with the feature extractor to generate input tensors
- Probabilities were computed using softmax over logits.
- Final prediction label was determined using argmax.

5. Visualization and Evaluation:

- After predictions, all results were saved to .csv files under: `results/sota/*.csv`
- For each model
 - Confusion Matrix was generated using `seaborn.heatmap()` based on predictions and true labels.
 - Classification Report was produced using `sklearn.metrics.classification_report()` to calculate:

These visual tools were essential in identifying class-wise misclassifications and understanding model behavior, especially for fine-tuned models like HuBERT and Wav2Vec2.

Summary of SOTA Testing Setup:

- Models Evaluated: 9 total
 - 6 Pretrained + MLP, 3 Fine-tuned end-to-end
- Libraries Used: HuggingFace Transformers, Torchaudio, PyTorch, Scikit-learn
- Input Format: Raw .wav audio (16kHz)
- Output Format: CSV files with predictions, probabilities, correctness flags
- Test Files: 264 Joey utterances

6.2. Experimental Results

6.2.1. ML prediction Results

In this section, we present and analyze the experimental results obtained from the testing phase of the machine learning models. A total of 264 utterances from the Joey-specific test subset of the MELD dataset were used for evaluation. Each of the 9 classical machine learning models was tested using a feature set that yielded its best validation performance during training. The evaluation metrics include overall accuracy, macro-averaged F1-score, and class-wise precision, recall, and F1-scores. Confusion matrices were also analyzed to understand common misclassification patterns.

Table 6.1 Machine Learning Prediction Results

Model	Feature Set Used	Accuracy (%)	F1-macro (%)
SVM	Boruta + Autoencoder	98.11	98.67
Random Forest	Boruta	87.12	85.22
XGBoost	Boruta	93.94	92.78
K-Nearest Neighbors	Boruta \cap RF	83.71	78.31
MLP (sklearn)	Boruta	87.88	85.06
Gradient Boosting	Boruta	93.56	95.81
Bagging Classifier	Boruta + Autoencoder	91.29	87.45
Hard Voting Classifier	Boruta	87.88	85.96
Soft Voting Classifier	Autoencoder	94.32	94.84

From the table above, it is evident that ensemble approaches, particularly Soft Voting, consistently yielded top-tier performance. The SVM model, when combined with Boruta + Autoencoder features, achieved the highest overall accuracy (98.11%) and macro F1-

score (98.67%). Similarly, Gradient Boosting and XGBoost models also performed strongly, both surpassing 93% accuracy.

- Class-Wise Analysis

Across nearly all models, the emotions fear, disgust, sadness, and anger were consistently predicted with high precision and recall. In contrast, joy and neutral posed recurring challenges:

- Joy was frequently confused with neutral, especially in models like Random Forest and Hard Voting.
- Neutral had variable recall across models, ranging from 71.5% (KNN) to 97.9% (SVM). Its confusion with joy, anger, and occasionally sadness reflects its acoustic and prosodic ambiguity.
- Disgust, although perfectly recalled in many models, sometimes suffered from low precision (e.g., Random Forest: $F1=0.52$), suggesting overprediction tendencies.

The Soft Voting Classifier demonstrated exceptional balance: F1-scores for all seven emotion classes exceeded 0.88, with especially strong results in fear (1.00), sadness (0.98), and disgust (0.91). It also had one of the highest neutral F1-scores (0.95), indicating superior generalization.

- Confusion Matrix Trends
- SVM displayed near-perfect separation across all classes, with only one misclassification (joy → neutral).
- Random Forest and KNN models struggled most with neutral, misclassifying up to 20% of instances into joy, anger, or sadness.
- Ensemble models such as Hard Voting showed stronger performance in high-arousal classes (fear, surprise), but sometimes misclassified neutral or low-energy emotions.
- The Gradient Boosting and Soft Voting classifiers stood out with minimal off-diagonal confusion, reflecting robust decision boundaries.
- Observations and Insights

- Feature selection techniques had a significant impact on model performance. The best results were achieved with feature sets involving Boruta and Autoencoder-based dimensionality reduction.
- Ensemble approaches (e.g., Bagging, Voting) consistently outperformed their base learners, confirming the benefit of model aggregation.
- SVM and Soft Voting emerged as the most accurate classifiers overall, though they excelled in different ways — SVM in raw decision boundary learning, and Soft Voting in robust probability-based consensus.

Summary

The test phase validated the effectiveness of the training pipeline and the quality of the feature engineering process. Models like SVM, Gradient Boosting, and Soft Voting not only achieved high overall accuracy but also demonstrated consistent class-wise performance, especially in emotionally salient classes like fear, sadness, and disgust. The challenges observed in predicting joy and neutral suggest that these emotions require more sophisticated modeling, possibly incorporating multimodal or contextual cues in future work.

6.2.2. Deep Learning Prediction Results

In this section, we present and evaluate the test results of the 11 deep learning models trained on the Joey-specific subset of the MELD dataset. A total of 264 utterances were used as the test set. These models include CNNs based on spectrogram inputs, transfer learning approaches, and models utilizing acoustic feature vectors with architectures such as BiLSTM and MLP. The evaluation metrics considered are overall accuracy and macro-averaged F1-score. In addition, class-wise performance and confusion matrices were analyzed to assess strengths and weaknesses across emotion categories.

Table 6.2 Deep Learning Prediction Results

No	Model	Input Type	Accuracy	F1-Macro
1	CNN (Mel)	Mel Spectrogram (128×128)	89.02%	89.20%
2	CNN (MFCC)	MFCC Spectrogram (128×128)	87.12%	84.06%

No	Model	Input Type	Accuracy	F1-Macro
3	CNN (RGB Fusion)	RGB Fusion (128×128)	65.91%	61.13%
4	CNN (Multi-Input)	MFCC + Mel Dual Input	62.12%	55.79%
5	ResNet18 (Mel)	Mel Spectrogram (224×224)	75.00%	74.50%
6	EfficientNetB0 (RGB)	RGB Fusion (224×224)	79.17%	77.72%
7	BiLSTM	Acoustic Features (1D)	91.29%	92.45%
8	Keras-BiLSTM	Acoustic Features (1D)	71.97%	68.92%
9	Torch-MLP	Acoustic Features (1D)	78.41%	76.21%
10	Keras-MLP	Acoustic Features (1D)	85.98%	85.48%
11	CNN + BiLSTM	Mel Spectrogram (128×128)	77.65%	75.22%

From the table above, the BiLSTM model trained on 1D acoustic features achieved the highest performance, with a macro F1-score of 92.45% and accuracy of 91.29%. This demonstrates the effectiveness of sequential modeling with temporal audio features in emotion recognition. Similarly, CNN (Mel) also performed competitively with an accuracy of 89.02% and macro F1-score of 89.20%, showing that 2D time-frequency representations remain powerful when combined with spatial convolutional filters.

- Class-Wise Analysis

Across most deep learning models, fear, sadness, and disgust were consistently predicted with high precision and recall. Notable findings include:

- Fear and disgust showed perfect recall in several models such as BiLSTM, CNN (Mel), and CNN+BiLSTM.
- Sadness was identified with high confidence by BiLSTM, Keras-MLP, and CNN+BiLSTM ($\text{F1-score} \geq 0.90$).
- Joy and surprise, however, posed challenges across multiple architectures. In models such as CNN-RGB and CNN-MultiInput, joy was frequently confused with neutral or surprise due to overlapping prosodic characteristics.
- Neutral, being the most populated class, generally achieved high recall but sometimes suffered from reduced precision — particularly in RGB-based CNNs — due to overfitting or under-discrimination.

- Confusion Matrix Trends
 - CNN-Mel and BiLSTM models showed clean class separations with minimal confusion, particularly between high-arousal emotions like fear and disgust.
 - CNN-RGB and CNN-MultiInput models had higher confusion rates, especially between joy–neutral, neutral–anger, and surprise–joy, suggesting that visual fusion approaches require more robust alignment with emotion boundaries.
 - CNN+BiLSTM was effective at recovering subtle emotions like disgust and fear, but occasionally misclassified joy as neutral, consistent with observations in other models.
- Observations and Insights
 - Input representation plays a crucial role: 1D acoustic features with sequential models (e.g., BiLSTM, MLP) tended to outperform image-based models when sufficient temporal information was preserved.
 - Spectrogram-based CNNs (particularly using Mel) demonstrated strong generalization, especially when applied with deeper networks like ResNet18 or fused with temporal modeling (CNN+BiLSTM).
 - RGB-based fusion CNNs underperformed, possibly due to loss of discriminative information in channel merging, or insufficient training data for complex 3-channel models.
 - Interestingly, MLP models trained on selected features (e.g., Keras-MLP and Torch-MLP) provided competitive results, suggesting that even feedforward architectures can effectively classify emotions when combined with powerful feature engineering.

Summary

The test phase of deep learning models validated their effectiveness in recognizing emotions from speech, especially when leveraging sequential architectures (BiLSTM) or time-frequency spectrograms (Mel). The BiLSTM model stood out as the best-performing overall, closely followed by CNN-Mel and Keras-MLP. On the other hand, CNN-RGB and multi-input CNNs showed

weaker generalization, highlighting the need for either larger training data or improved architectural designs in fusion models.

6.2.3. SOTA Prediction Results

This section presents the test results of 9 state-of-the-art (SOTA) models based on self-supervised learning (SSL) and transformer architectures. These models include WavLM, Wav2Vec2.0, and HuBERT variants, applied either in feature extraction mode (frozen embeddings with MLP classifier) or in end-to-end fine-tuning using PyTorch. All models were tested on the same Joey-specific subset (264 utterances), and evaluated using overall accuracy and macro-averaged F1-score.

Table 6.3 Sota Prediction Results

No	Model	Accuracy (%)	F1-macro (%)
1	WavLM (Large) + MLP	82.95	79.11
2	WavLM (Base+) + MLP	76.89	72.58
3	Wav2Vec2.0 (facebook/960h) + MLP	70.45	66.64
4	Wav2Vec2.0 (ehcalabres) + MLP	65.15	63.36
5	HuBERT (Large) + MLP	77.27	74.18
6	HuBERT (Base) + MLP	76.14	73.49
7	HuBERT (Fine-tuned, PyTorch)	77.65	79.45
8	WavLM (Fine-tuned, PyTorch)	63.64	49.53
9	Wav2Vec2.0 (Fine-tuned, PyTorch)	84.09	79.73

From the results above, the fine-tuned Wav2Vec2.0 model achieved the best performance overall, with 84.09% accuracy and 79.73% macro F1-score, outperforming all other pretrained variants. Among frozen-feature-based models, WavLM (Large) and HuBERT (Large) followed closely, indicating the effectiveness of large-scale SSL pretraining in emotion representation.

- Class-Wise Analysis
 - Disgust, fear, and sadness were the most consistently recognized emotions across both pretrained and fine-tuned SOTA models.
 - *Perfect recall* for disgust and fear was observed in many models, including HuBERT (Base/Large) and Wav2Vec2.0 (Fine-tuned), despite their limited sample size.
 - *Sadness* achieved high F1-scores across almost all models, often exceeding 0.80.
 - Anger was effectively detected by most models, especially in fine-tuned variants such as Wav2Vec2.0 and HuBERT, where recall reached 1.00 in some cases.
 - Neutral class performance varied:
 - High precision was observed in WavLM and Wav2Vec2.0 pretrained models.
 - However, models like WavLM (Fine-tuned) and HuBERT (Large) showed reduced recall due to overgeneralization or confusion with joy and anger.
 - Joy and surprise remained challenging:
 - Joy was frequently confused with neutral and surprise, especially in models with lower overall accuracy (e.g., WavLM Fine-tuned, RGB-based CNNs).
 - Surprise showed wide variability — from high recall (HuBERT variants) to poor performance (WavLM Fine-tuned, F1 = 0.53).
- Confusion Matrix Trends
 - Fine-tuned Wav2Vec2.0 exhibited the cleanest class separation across all emotions, showing balanced prediction across classes.
 - HuBERT (Fine-tuned) also demonstrated strong class distinction, especially for sadness, surprise, and anger.
 - WavLM (Base+) and ehcalabres-Wav2Vec2.0 showed higher confusion between joy–neutral and neutral–anger, especially under MLP-based classification.

- WavLM (Fine-tuned) struggled significantly with joy and fear, often misclassifying them as neutral or anger — resulting in its relatively low F1-score (49.53%).
- Observations and Insights
 - Feature extraction vs fine-tuning:
 - Fine-tuned models (e.g., Wav2Vec2.0 and HuBERT) generally outperformed frozen models, especially in detecting subtle emotions like surprise and sadness.
 - However, pretrained + MLP pipelines (e.g., WavLM-Large + MLP) still achieved strong results with less training effort, showcasing the transferability of SSL features.
 - Model size and architecture mattered:
 - Larger models (e.g., WavLM Large, HuBERT Large) consistently performed better than their base counterparts.
 - This suggests deeper models are better at capturing the nuanced prosodic cues in expressive speech.
 - Sample imbalance impacted performance:
 - Classes with fewer examples (e.g., disgust, fear) showed high variance depending on model robustness and generalization ability.
 - Nevertheless, several models managed perfect or near-perfect predictions for these minority classes.

Summary

The evaluation of transformer-based SOTA models revealed that fine-tuned Wav2Vec2.0 achieved the best emotion recognition results, closely followed by WavLM (Large) and HuBERT (Large) with frozen embeddings. These findings highlight the power of self-supervised speech representations, particularly when further adapted to the target domain via fine-tuning. Pretrained + MLP setups also offer a strong and lightweight alternative with competitive accuracy, making them suitable for resource-constrained deployment scenarios.

CHAPTER SEVEN

CONCLUSION AND FUTURE WORK/S

7.1. Conclusion

This study has presented a comprehensive approach to Speech Emotion Recognition (SER) using only the audio modality of the MELD dataset. Focusing exclusively on the character *Joey* enabled a controlled setting for exploring how acoustic properties vary across emotions, without introducing inter-speaker variability. The project integrated classical machine learning, deep learning, and state-of-the-art (SOTA) transformer-based models, offering a detailed comparative evaluation across different model families and input representations.

The workflow began with meticulous preprocessing and data cleaning, including audio extraction, alignment checks, silence trimming, and augmentation to address class imbalance. This ensured that the dataset was both reliable and representative. Feature extraction covered a wide spectrum of handcrafted acoustic features (e.g., MFCCs, pitch, jitter, shimmer, spectral, wavelet, openSMILE), while feature selection techniques such as Boruta, RFE, Autoencoder, and hybrid strategies were applied to optimize input dimensions for ML and MLP models.

Model training and evaluation spanned:

- 9 classical ML models across 11 feature sets, including SVM, Random Forest, XGBoost, and ensemble approaches;
- 11 DL architectures, such as CNNs with spectrogram inputs, BiLSTM networks with sequential acoustic features, and hybrid CNN-BiLSTM models;
- 9 SOTA models, both in frozen embedding + MLP format and end-to-end fine-tuned transformer models (WavLM, Wav2Vec2.0, HuBERT).

Extensive testing was conducted on a dedicated test set of 264 *Joey*-specific utterances. The highest performance was achieved by the BiLSTM model (F1-macro: 92.45%, Accuracy: 91.29%) and the Wav2Vec2.0 (Fine-tuned) transformer model (F1-macro:

79.73%, Accuracy: 84.09%). Among classical models, the SVM with Boruta-Autoencoder-RF feature selection performed best (F1-macro: 86.51%).

Throughout the experiments, emotions like fear, sadness, and disgust were predicted with consistently high accuracy, while joy and surprise proved more challenging due to overlapping acoustic cues with neutral expressions. The findings support the effectiveness of combining careful preprocessing, robust feature engineering, and model-specific input representations for SER tasks.

In conclusion, this project demonstrates that SER can be effectively modeled using both traditional and modern techniques. While deep models showed stronger generalization, especially with sequential inputs, classical ML models with well-engineered features also achieved competitive results. The developed GUI interface further enables practical application of all models for real-time inference on unseen audio inputs. This work lays a solid foundation for future expansion into multimodal emotion recognition, speaker-generalized models, and real-time deployment.

7.2. Future Work

While this project has achieved promising results in speech-based emotion recognition using the Joey-specific subset of the MELD dataset, there are several avenues for future improvement and expansion:

- Multimodal Emotion Recognition:

The MELD dataset includes textual and visual modalities in addition to audio. Future studies can incorporate these modalities using multimodal fusion techniques, enabling more robust emotion recognition systems that mirror human perception more closely.

- Speaker-Generalized Models:

This study focused on a single speaker (Joey) to ensure consistency in vocal patterns. Expanding the model to include multiple speakers will allow evaluation of generalization capabilities across different voice profiles, accents, and speech dynamics.

- Real-Time Emotion Prediction:

The current implementation processes pre-segmented audio clips. Future development can adapt the system for real-time emotion detection, integrating streaming input, faster inference pipelines, and efficient deployment on edge devices or mobile platforms.

- Semi-Supervised and Self-Supervised Learning:

Given the limited labeled emotional data available in many real-world applications, techniques such as semi-supervised learning, contrastive learning, or self-supervised pretraining can be employed to improve model robustness with fewer annotations.

- Data Augmentation and Balancing Strategies:

Although class balancing was performed using audio augmentation, further exploration of synthetic data generation (e.g., GAN-based speech synthesis), voice conversion, or emotion morphing could enrich the dataset diversity and support underrepresented classes.

- Explainability and Interpretability:

Future work could incorporate explainable AI (XAI) techniques to understand which acoustic features contribute most to emotion predictions. This would aid in debugging models and enhancing user trust in real-world deployments.

- Cross-Corpus Evaluation:

Testing the trained models on other SER datasets such as IEMOCAP or CREMA-D could provide insights into their cross-domain generalization abilities and support benchmarking in broader research contexts.

- Transfer Learning with Lightweight Models:

Although transformer-based models like WavLM and HuBERT achieved strong performance, their size may limit deployment. Future efforts could explore knowledge distillation, quantization, or use of lightweight architectures (e.g., DistilHuBERT) for more efficient inference.

- GUI and Usability Improvements:

The developed interface provides model selection and emotion prediction, but can be enhanced with additional features like emotion timelines, speaker diarization visualization, and integration with other applications (e.g., virtual assistants, education tools).

REFERENCES

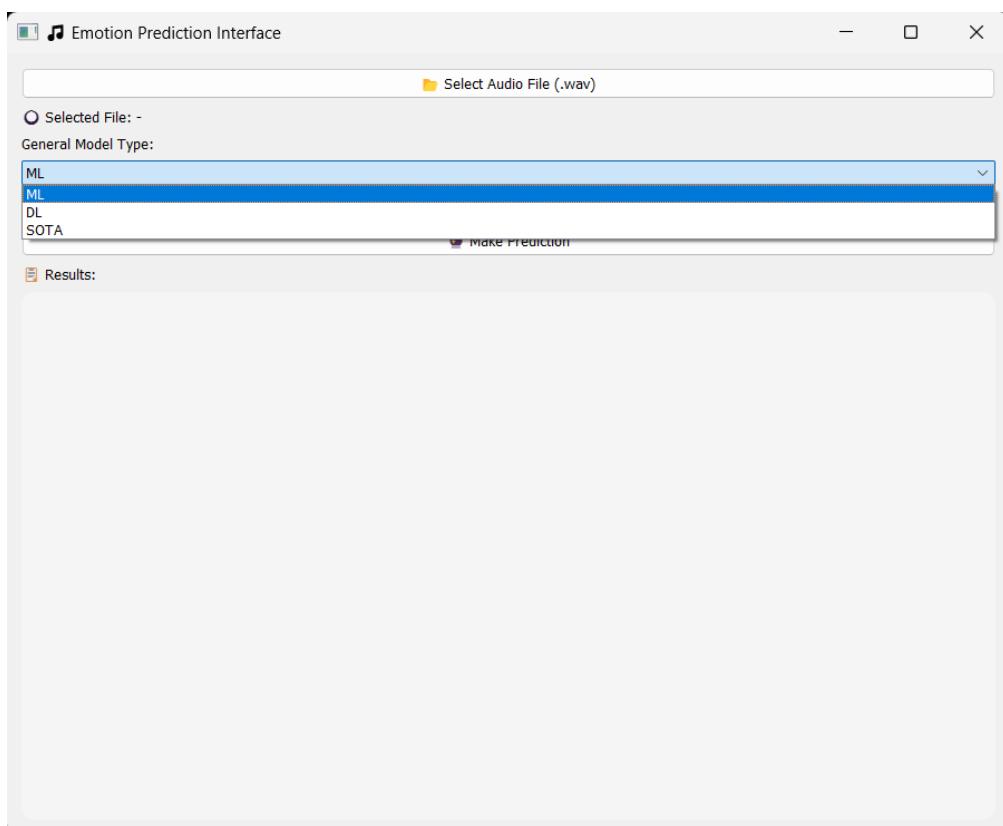
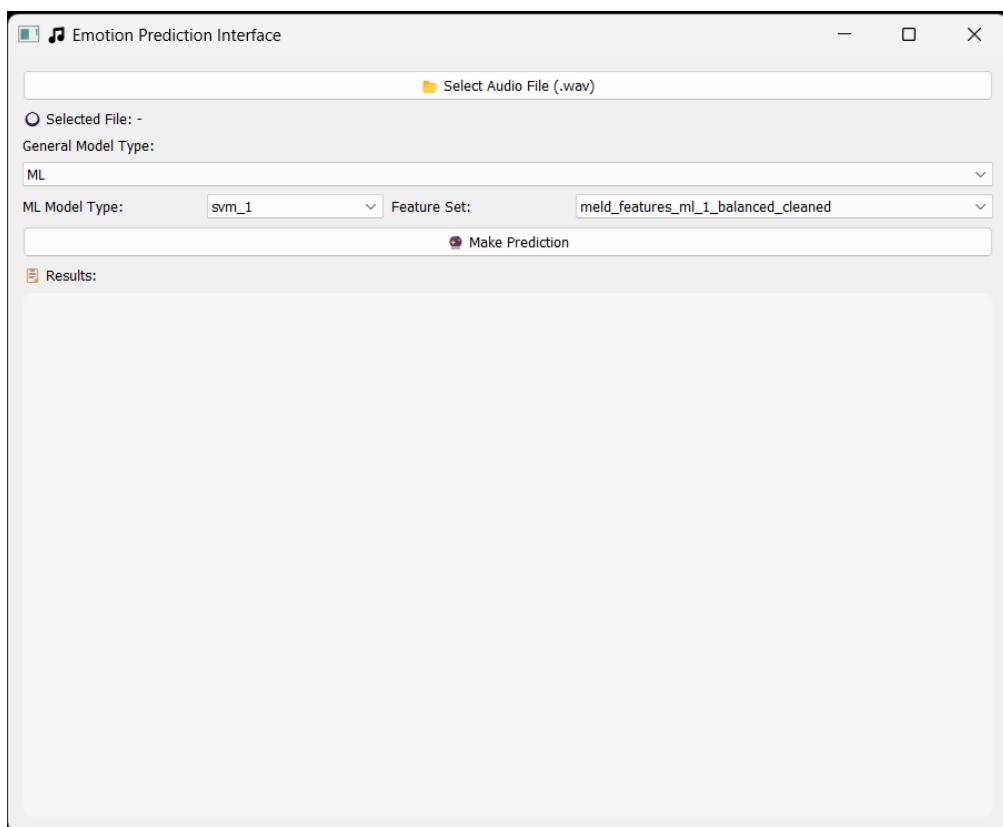
- Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a Convolutional Neural Network. *2017 International Conference on Engineering and Technology (ICET)*, 1–6.
- Ang, J., Dhillon, R., Krupski, A., Shriberg, E., & Stolcke, A. (2002). Prosody-based automatic detection of annoyance and frustration in human-computer dialog. ICSLP, 3, 2037–2040.
- Aouani, H., & Ayed, Y. B. (2020). Speech emotion recognition with deep learning. Procedia Computer Science, 176, 251-260.
- APA Format Citation Guide. (2020). Retrieved from <https://www.mendeley.com/guides/apa-citation-guide/>.
- Association of Human-Computer Interaction. (2023). The evolution of human-computer interaction: A review of the past and future directions. Retrieved from <https://www.hci.org.uk/article/the-evolution-of-human-computer-interaction-a-review-of-the-past-and-future-directions/>
- Bhavan, A., Chauhan, P., & Shah, R. R. (2019). Bagged support vector machines for emotion recognition from speech. Knowledge-Based Systems, 184, 104886.
- Chen, S.Y., Hsu, C.C., Kuo, C.C. and Ku, L.W. EmotionLines: An Emotion Corpus of Multi-Party Conversations. arXiv preprint arXiv:1802.08379 (2018).
- Cichosz, J., & Slot, K. (2007). Emotion recognition in speech signal using emotion extracting binary decision trees. ACII.
- Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297.
- Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1), 21–27.
- Dan Nie, Xiao-Wei Wang, Li-Chen Shi, and Bao-Liang Lu, EEG-based Emotion Recognition during Watching Movies, Proc. of 5th International IEEE EMBC Conference on Neural Engineering (NER), pp. 667-670, Cancun, Mexico, April 27-May 1, 2011.

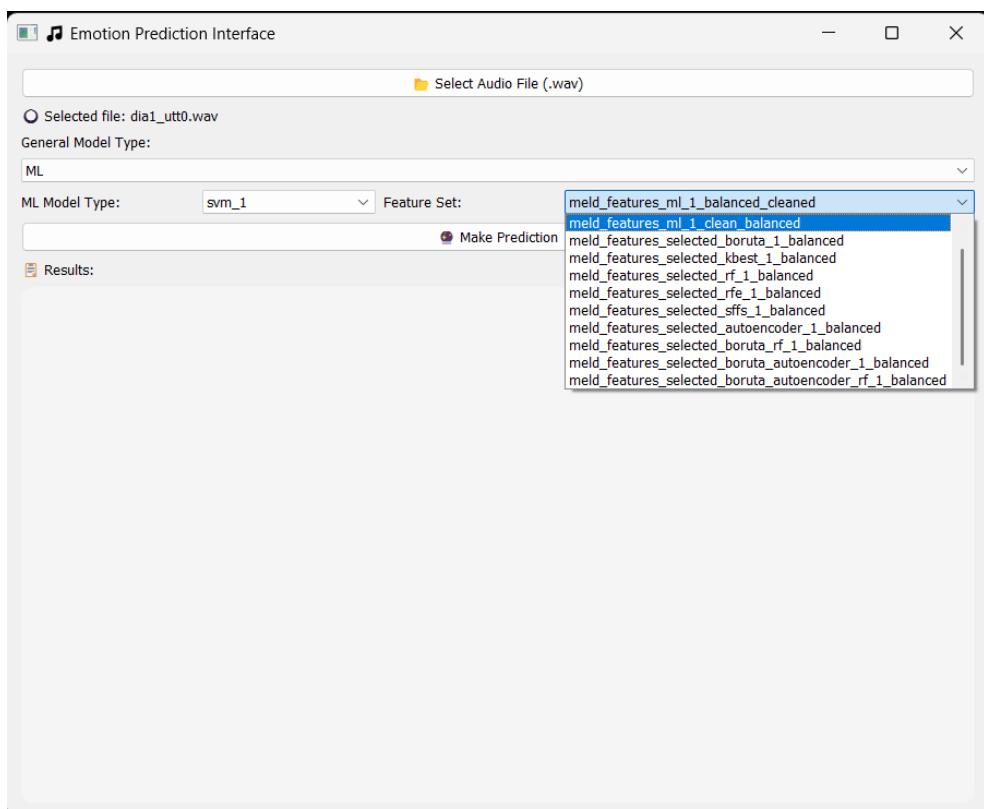
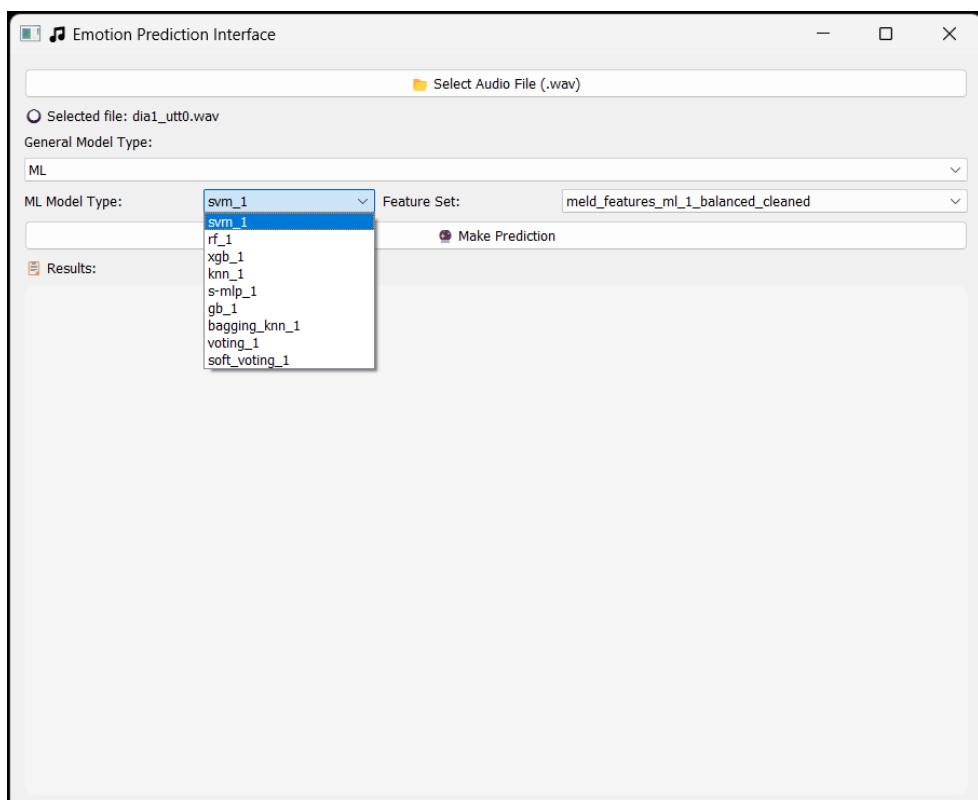
- Dix, A. Human-Computer Interaction. In Encyclopedia of Database Systems; Liu, L., Özsü, M.T., Eds.; Springer: Boston, MA, USA, 2009.
- Duckworth, A. L., Quirk, A., Gallop, R., Hoyle, R. H., Kelly, D. R., & Matthews, M. D. (2019). Cognitive and noncognitive predictors of success. *Proceedings of the National Academy of Sciences, USA*, 116(47), 23499–23504.
- Fayek, H. M., Lech, M., & Cavedon, L. (2017). Evaluating deep learning architectures for Speech Emotion Recognition. *Neural Networks*, 92, 60-68.
- Gu, Y., Lyu, X., Sun, W., Li, W., Chen, S., Li, X., & Marsic, I. (2019, October). Mutual correlation attentive factors in dyadic fusion networks for speech emotion recognition. In *Proceedings of the 27th ACM International Conference on Multimedia* (pp. 157-166).
- Grady, J. S., Her, M., Moreno, G., Perez, C., & Yelinek, J. (2019). Emotions in storybooks: A comparison of storybooks that represent ethnic and racial groups in the United States. *Psychology of Popular Media Culture*, 8(3), 207–217. <https://doi.org/10.1037/ppm0000185>
- Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. *Proceedings of the 23rd International Conference on Machine Learning*, 369–376.
- Huang, K. Y., Wu, C. H., & Su, M. H. (2019). Attention-based convolutional neural network and long short-term memory for short-term detection of mood disorders based on elicited speech responses. *Pattern Recognition*, 88, 668-678.
- Juang, B. H., & Rabiner, L. R. (2005). Automatic speech recognition—a brief history of the technology development. *Elsevier Encyclopedia of Language and Linguistics*, 2nd Edition.
- KORKMAZ, Y., & BOYACI, A. (2018). Adli bilişim açısından ses incelemeleri. *Fırat Üniversitesi Mühendislik Bilimleri Dergisi*, 30(1), 329-343.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.
- Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv preprint arXiv:1506.00019*.

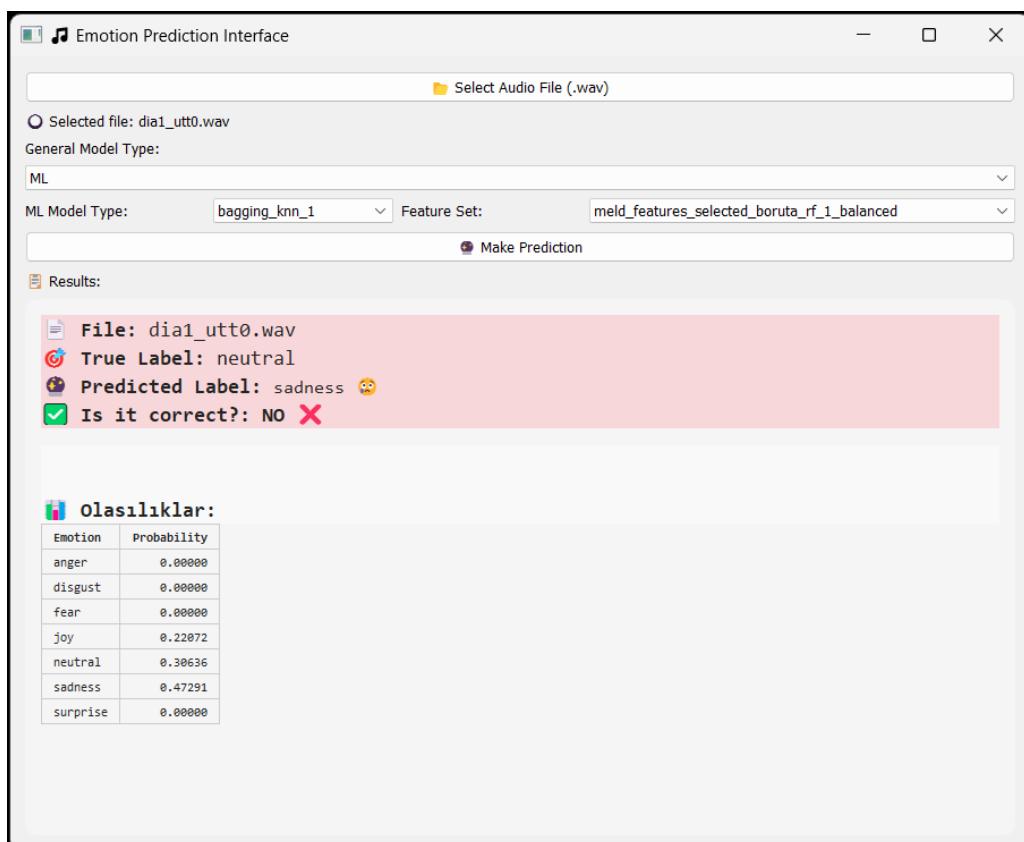
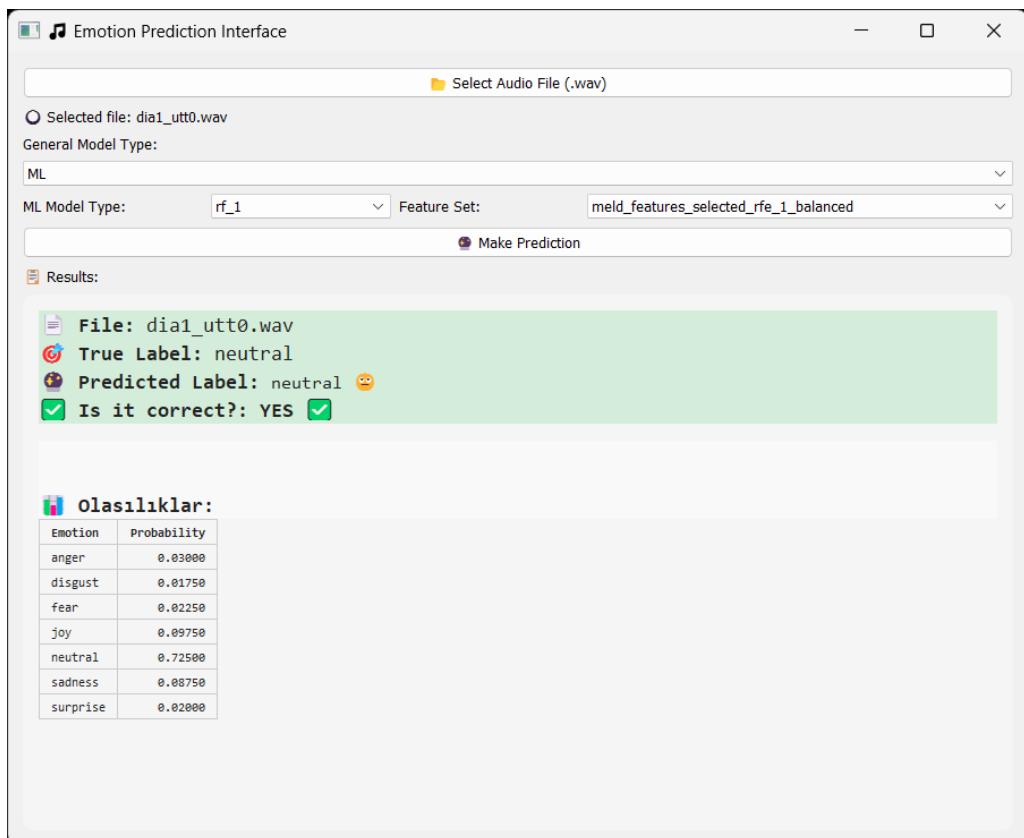
- Liu, W., Zheng, W.-L., & Lu, B.-L. (2016). Emotion recognition using multimodal deep learning. Lecture Notes in Computer Science, 9948, 521–529.
- Luna-Jiménez, C., Griol, D., Callejas, Z., Kleinlein, R., Montero, J. M., & Fernández-Martínez, F. (2021). Multimodal Emotion Recognition on RAVDESS Dataset Using Transfer Learning. *Sensors*, 21(22), 7665
- Nagesh Singh Chauhan, Naive Bayes, 22, Kasım, 2021). Erişim Adresi (<https://www.kdnuggets.com/2020/06/naive-bayesalgorithm-everything.html>).
- Nie, D., Wang, X.-W., Shi, L.-C., & Lu, B.-L. (2011). EEG-based emotion recognition using frequency domain features and support vector machines. Lecture Notes in Computer Science, 7062, 734–743.
- Pan, Y., Shen, P., & Shen, L. (2012). Speech emotion recognition using support vector machine. *International Journal of Smart Home*, 6(2), 101-108.
- Peter J Lang, Margaret M Bradley, and Bruce N Cuthbert. Emotion, motivation, and anxiety: Brain mechanisms and psychophysiology. *Biological psychiatry*, 44(12):1248–1263, 1998.
- Polat, G., & Altun, H. (2008). Ses öznitelik vektörlerinin duygusal durum sınıflandırılmasında kullanımı. *Elektrik-Elektronik ve Bilgisayar Mühendisliği Sempozyumu (ELECO 2008)*, Bursa, Türkiye.
- Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 257–286.
- Reynolds, D. A. (2009). Gaussian Mixture Models. In *Encyclopedia of Biometrics* (pp. 659–663). Springer.
- Roisman, G. I. (2007). The psychophysiology of adult attachment relationships: Autonomic reactivity in marital and premarital interactions. *Developmental Psychology*, 43(1), 39–53.
- Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1), 81–106.
- Sartra Wongthanavasu, Thapanee Seehapoch. (2013). *Speech emotion recognition using SVM and KNN with pyAudio analysis and WEKA optimization*. In *Proceedings of the International Conference on Knowledge and Smart Technology (KST)*. IEEE.
- Seehapoch, Thapanee & Wongthanavasu, Sartra. (2013). Speech emotion recognition using Support Vector Machines. 86-91. 10.1109/KST.2013.6512793.

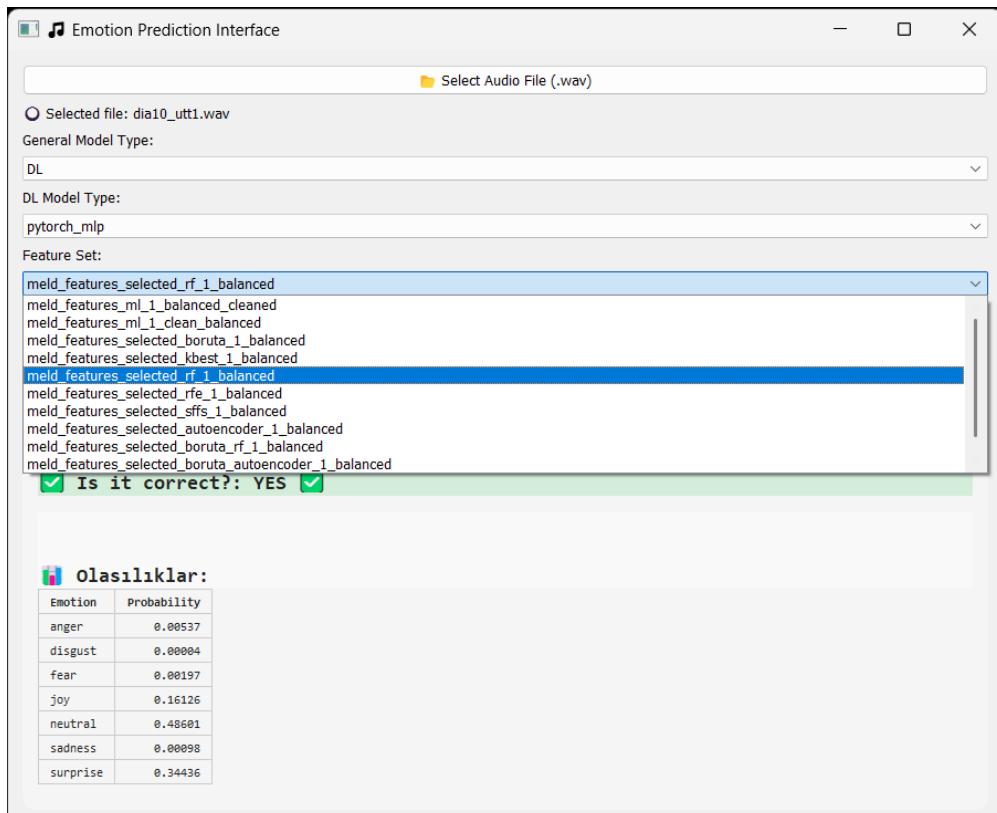
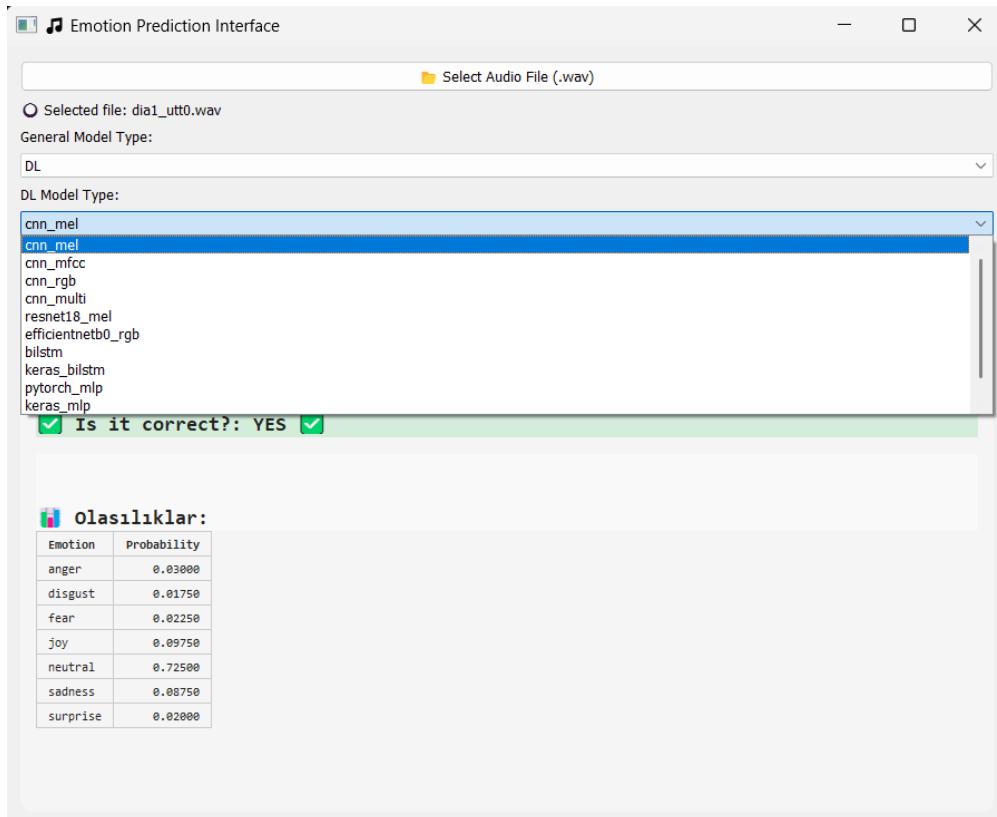
- S. Poria, D. Hazarika, N. Majumder, G. Naik, E. Cambria, R. Mihalcea. MELD: A Multimodal Multi-Party Dataset for Emotion Recognition in Conversation. ACL 2019.
- Tripathi, Samarth & Beigi, Homayoon. (2018). Multi-Modal Emotion recognition on IEMOCAP Dataset using Deep Learning.
- Wei Liu, Wei-Long Zheng, Bao-Liang Lu*, Emotion Recognition using Multimodal Deep Learning, Proc. of ICONIP2016, Kyoto, 2016
- Yao, Z., Wang, Z., Liu, W., Liu, Y., & Pan, J. (2020). Speech emotion recognition using fusion of three multi-task learning-based classifiers: HSF-DNN, MS-CNN and LLD-RNN. *Speech Communication*, 120, 11-19.

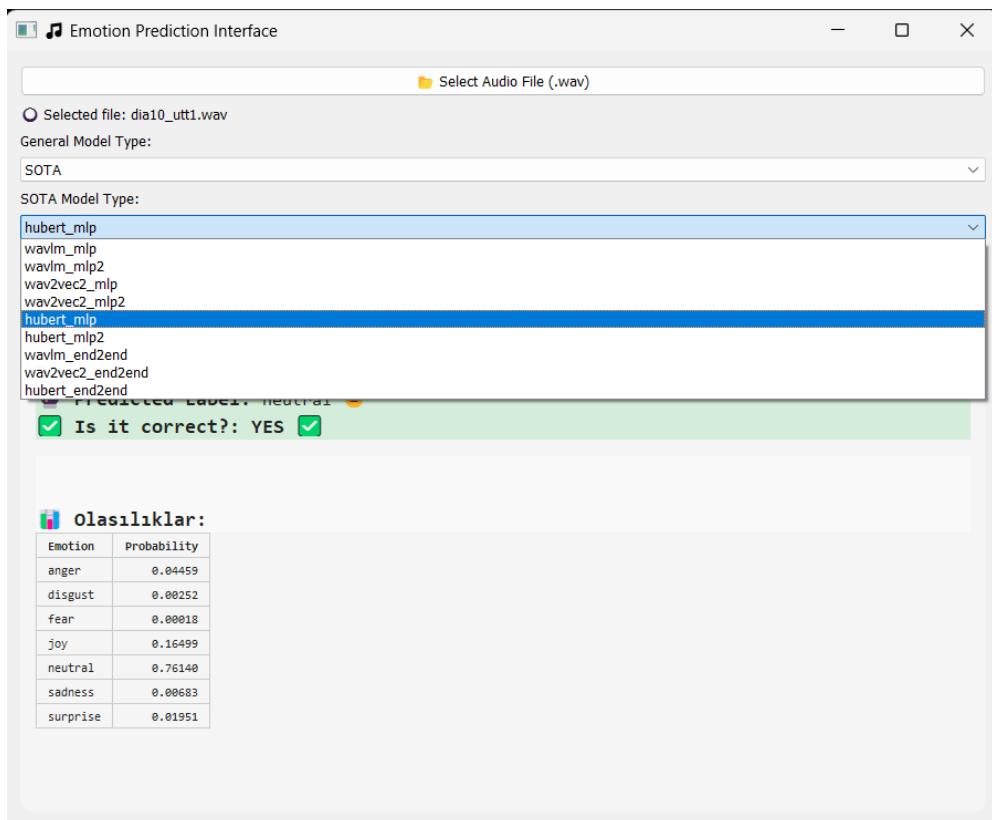
APPENDIX A











APPENDIX B

1) SVM

```

Başlıyor: meld_features_selected_boruta_autoencoder_1_balanced.csv
LabelEncoder kaydedildi: features_and_models/ml_models/svm_1\label_encoder_meld_features_selected_boruta_autoencoder_1_balanced.pkl
Scaler kaydedildi: features_and_models/ml_models/svm_1\scaler_meld_features_selected_boruta_autoencoder_1_balanced.pkl
Fitting 5 folds for each of 80 candidates, totalling 400 fits
En iyi parametreler: {'C': 0.1, 'class_weight': None, 'gamma': 0.1, 'kernel': 'poly'}
Accuracy: 0.8643 | F1 (macro): 0.8651
      precision    recall   f1-score   support
anger        0.89     0.83     0.86      60
disgust       0.95     0.98     0.97      60
fear          0.98     0.92     0.95      60
joy           0.76     0.58     0.66      60
neutral       0.63     0.93     0.75      60
sadness       0.95     0.98     0.97      60
surprise      1.00     0.82     0.90      60

accuracy               0.86      420
macro avg            0.88     0.86     0.87      420
weighted avg         0.88     0.86     0.87      420

Confusion Matrix:
[[50  0  0  4  5  1  0]
 [ 0 59  0  0  1  0  0]
 [ 0  1 55  2  1  1  0]
 [ 5  1  0 35 18  1  0]
 [ 0  1  1  2 56  0  0]
 [ 0  0  0  0  1 59  0]
 [ 1  0  0  3  7  0 49]]
Süre: 10.29 saniye
SVM modeli kaydedildi: features_and_models/ml_models/svm_1\svm_best_model_meld_features_selected_boruta_autoencoder_1_balanced.pkl

```

2) RF

```

Başlıyor: meld_features_selected_boruta_1_balanced.csv
LabelEncoder kaydedildi: features_and_models/ml_models/rf_1\label_encoder_meld_features_selected_boruta_1_balanced.pkl
Fitting 5 folds for each of 18 candidates, totalling 90 fits
En iyi parametreler: {'class_weight': 'balanced', 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 400}
Accuracy: 0.8024 | F1 (macro): 0.7948
      precision    recall   f1-score   support
anger        0.65     0.88     0.75      60
disgust       0.95     0.95     0.95      60
fear          1.00     0.95     0.97      60
joy           0.76     0.43     0.55      60
neutral       0.76     0.63     0.69      60
sadness       0.78     0.93     0.85      60
surprise      0.77     0.83     0.80      60

accuracy               0.80      420
macro avg            0.81     0.80     0.79      420
weighted avg         0.81     0.80     0.79      420

Confusion Matrix:
[[53  0  0  2  2  0  3]
 [ 3 57  0  0  0  0  0]
 [ 0  0 57  0  0  2  1]
 [13  2  0 26  7  8  4]
 [ 9  1  0  3 38  5  4]
 [ 0  0  1  0 56  3  0]
 [ 4  0  0  2  3  1 50]]
Süre: 146.67 saniye
RF modeli kaydedildi: features_and_models/ml_models/rf_1\rf_best_model_meld_features_selected_boruta_1_balanced.pkl

```

3) XGBoost

```

Başlıyor: meld_features_selected_boruta_1_balanced.csv
LabelEncoder kaydedildi: features_and_models/ml_models/xgb_1\label_encoder_meld_features_selected_boruta_1_balanced.pkl
Accuracy: 0.8286 | F1 (macro): 0.8253
      precision    recall   f1-score   support
anger        0.75     0.83     0.79      60
disgust       0.97     0.97     0.97      60
fear          0.92     0.97     0.94      60
joy           0.82     0.55     0.66      60
neutral       0.70     0.80     0.74      60
sadness       0.83     0.87     0.85      60
surprise      0.84     0.82     0.83      60

accuracy               0.83      420
macro avg            0.83     0.83     0.83      420
weighted avg         0.83     0.83     0.83      420

Confusion Matrix:
[[50  0  1  4  4  0  1]
 [ 2 58  0  0  0  0  0]
 [ 0  0 58  0  0  2  0]
 [ 8  0  1 33 11  7  0]
 [ 4  0  0  2 48  1  5]
 [ 0  2  3  0  0 52  3]
 [ 3  0  0  1  6  1 49]]
Süre: 165.69 saniye
XGBoost modeli kaydedildi: features_and_models/ml_models/xgb_1\xgb_best_model_meld_features_selected_boruta_1_balanced.pkl

```

4) k-NN

```

Başlıyor: meld_features_selected_boruta_autoencoder_1_balanced.csv
LabelEncoder kaydedildi: features_and_models/ml_models/knn_1\label_encoder_meld_features_selected_boruta_autoencoder_1_balanced.pkl
En iyi parametreler: {'metric': 'euclidean', 'n_neighbors': 3, 'weights': 'distance'}
Accuracy: 0.8214 | F1 (macro): 0.8177
precision recall f1-score support
anger      0.81    0.78    0.80    60
disgust     0.85    1.00    0.92    60
fear        0.92    0.95    0.93    60
joy         0.78    0.67    0.72    60
neutral     0.63    0.60    0.62    60
sadness     0.83    0.95    0.88    60
surprise    0.92    0.80    0.86    60

accuracy          0.82    420
macro avg       0.82    0.82    0.82    420
weighted avg    0.82    0.82    0.82    420

* Confusion Matrix:
[[47  1  1  3  5  3  0]
[ 0 60  0  0  0  0  0]
[ 0  1 57  0  0  2  0]
[ 4  3  1 40 10  1  1]
[ 3  4  3  6 36  5  3]
[ 0  2  0  1  0 57  0]
[ 4  0  0  1  6  1 48]]
⌚ Süre: 1.35 saniye
📦 KNN modeli kaydedildi: features_and_models/ml_models/knn_1\knn_best_model_meld_features_selected_boruta_autoencoder_1_balanced.pkl

```

5) sklearn MLP

```

Başlıyor: meld_features_selected_boruta_1_balanced.csv
LabelEncoder kaydedildi: features_and_models/ml_models/s-mlp_1\label_encoder_meld_features_selected_boruta_1_balanced.pkl
En iyi parametreler: {'activation': 'relu', 'alpha': 0.001, 'hidden_layer_sizes': (256, 128), 'learning_rate': 'adaptive', 'solver': 'adam'}
Accuracy: 0.8024 | F1 (macro): 0.7952
precision recall f1-score support
anger      0.72    0.77    0.74    60
disgust     0.94    0.98    0.96    60
fear        0.86    0.98    0.91    60
joy         0.68    0.50    0.58    60
neutral     0.70    0.63    0.67    60
sadness     0.82    0.92    0.87    60
surprise    0.85    0.83    0.84    60

accuracy          0.80    420
macro avg       0.79    0.80    0.80    420
weighted avg    0.79    0.80    0.80    420

* Confusion Matrix:
[[46  1  1  3  7  1  1]
[ 0 59  1  0  0  0  0]
[ 0  0 59  0  0  1  0]
[10  2  4 30  7  4  3]
[ 7  0  1  5 38  4  5]
[ 1  1  2  1  0 55  0]
[ 0  0  1  5  2  2 50]]
⌚ Süre: 125.57 saniye
📦 MLP modeli kaydedildi: features_and_models/ml_models/s-mlp_1\mlp_best_model_meld_features_selected_boruta_1_balanced.pkl

```

6) GradientBoosting

```

Başlıyor: meld_features_selected_boruta_1_balanced.csv
LabelEncoder kaydedildi: features_and_models/ml_models/gb_1\label_encoder_meld_features_selected_boruta_1_balanced.pkl
En iyi parametreler: {'learning_rate': 0.05, 'loss': 'log_loss', 'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 3, 'n_estimators': 400, 'subsample': 0.8}
Accuracy: 0.8357 | F1 (macro): 0.8346
precision recall f1-score support
anger      0.80    0.85    0.82    60
disgust     0.98    0.97    0.97    60
fear        0.98    0.93    0.96    60
joy         0.70    0.58    0.64    60
neutral     0.73    0.73    0.73    60
sadness     0.93    0.88    0.91    60
surprise    0.74    0.90    0.81    60

accuracy          0.84    420
macro avg       0.84    0.84    0.83    420
weighted avg    0.84    0.84    0.83    420

* Confusion Matrix:
[[51  0  0  5  2  0  2]
[ 2 58  0  0  0  0  0]
[ 0  0 56  1  0  2  1]
[ 7  0  1 35 12  2  3]
[ 3  0  0  5 44  0  8]
[ 0  1  0  1  0 53  5]
[ 1  0  0  3  2  0 54]]
⌚ Süre: 86.36 saniye
📦 GB modeli kaydedildi: features_and_models/ml_models/gb_1\gb_best_model_meld_features_selected_boruta_1_balanced.pkl

```

7) BaggingClassifier

```
▀ Başlıyor: meld_features_selected_boruta_autoencoder_1_balanced.csv
▀ LabelEncoder kaydedildi: features_and_models/ml_models/bagging_knn_1\label_encoder_meld_features_selected_boruta_autoencoder_1_balanced.pkl
↳ En iyi parametreler: {'bootstrap': True, 'estimator': KNeighborsClassifier(metric='manhattan', n_neighbors=3, weights='distance'), 'max_features': 0,
8, 'max_samples': 1.0, 'n_estimators': 30}
☒ Accuracy: 0.8214 | F1 (macro): 0.8159
precision    recall   f1-score  support
anger        0.85     0.78     0.82      60
disgust       0.83     1.00     0.91      60
fear          0.84     0.95     0.89      60
joy           0.85     0.65     0.74      60
neutral       0.63     0.57     0.60      60
sadness       0.83     0.97     0.89      60
surprise      0.91     0.83     0.87      60

accuracy            0.82      420
macro avg         0.82     0.82     0.82      420
weighted avg      0.82     0.82     0.82      420

☒ Confusion Matrix:
[[47  2  1  3  5  1  1]
 [ 0 60  0  0  0  0  0]
 [ 0 157  0  0  2  0]
 [ 4  3  1 39 10  2  1]
 [ 3  5  8  3 34  4  3]
 [ 1  1  0  0 58  0]
 [ 0  0  1  1  5 3 50]]
⌚ Süre: 7.10 saniye
▣ Bagging KNN modeli kaydedildi: features_and_models/ml_models/bagging_knn_1\bagging_knn_best_model_meld_features_selected_boruta_autoencoder_1_balance
d.pkl
```

8) Ensemble-Hard Voting

```
▀ Başlıyor: meld_features_selected_boruta_1_balanced
C:\ProgramData\anaconda3\Lib\site-packages\joblib\extern
executor. This can be caused by a too short worker tim
warnings.warn(
```

```
🔗 Ensemble (Voting) Accuracy: 0.8286 | F1: 0.8229
precision    recall   f1-score  support
anger        0.66     0.90     0.76      60
disgust       0.85     0.97     0.91      60
fear          0.95     0.97     0.96      60
joy           0.82     0.52     0.63      60
neutral       0.80     0.67     0.73      60
sadness       0.85     0.95     0.90      60
surprise      0.93     0.83     0.88      60

accuracy            0.83      420
macro avg         0.84     0.83     0.82      420
weighted avg      0.84     0.83     0.82      420
```

9) Soft Voting

```
▀ Başlıyor: meld_features_selected_boruta_autoencoder_1_balanced
```

```
🔗 Soft Voting Accuracy: 0.8333 | F1: 0.8294
precision    recall   f1-score  support
anger        0.86     0.80     0.83      60
disgust       0.85     1.00     0.92      60
fear          0.92     0.95     0.93      60
joy           0.73     0.55     0.63      60
neutral       0.63     0.70     0.66      60
sadness       0.95     1.00     0.98      60
surprise      0.89     0.83     0.86      60

accuracy            0.83      420
macro avg         0.83     0.83     0.83      420
weighted avg      0.83     0.83     0.83      420
```

1) Mel-Spectrogram CNN

Accuracy: 0.7643

Classification Report:				
	precision	recall	f1-score	support
anger	0.78	0.83	0.81	60
disgust	0.98	0.95	0.97	60
fear	0.92	0.92	0.92	60
joy	0.62	0.62	0.62	60
neutral	0.46	0.55	0.50	60
sadness	0.94	0.83	0.88	60
surprise	0.74	0.65	0.69	60
accuracy			0.76	420
macro avg	0.78	0.76	0.77	420
weighted avg	0.78	0.76	0.77	420

Confusion Matrix:
[[50 0 1 2 5 0 2]
[2 57 0 0 1 0 0]
[1 0 55 1 3 0 0]
[3 0 1 37 12 2 5]
[6 1 0 12 33 1 7]
[2 0 0 5 3 50 0]
[0 0 3 3 15 0 39]]

2) MFCC-CNN

Accuracy: 0.6119

Classification Report:				
	precision	recall	f1-score	support
anger	0.5405	0.6667	0.5970	60
disgust	0.8095	0.8500	0.8293	60
fear	0.7656	0.8167	0.7903	60
joy	0.4211	0.4000	0.4103	60
neutral	0.4651	0.3333	0.3883	60
sadness	0.7213	0.7333	0.7273	60
surprise	0.5000	0.4833	0.4915	60
accuracy			0.6119	420
macro avg	0.6033	0.6119	0.6049	420
weighted avg	0.6033	0.6119	0.6049	420

Confusion Matrix:
[[40 2 3 5 4 3 3]
[1 51 0 3 2 1 2]
[3 1 49 1 2 0 4]
[11 2 2 24 7 5 9]
[9 2 5 10 20 6 8]
[4 3 1 4 1 44 3]
[6 2 4 10 7 2 29]]

3) RGB-CNN (MFCC + Δ + Δ^2)

Accuracy: 0.6310

Classification Report:				
	precision	recall	f1-score	support
anger	0.5641	0.7333	0.6377	60
disgust	0.7468	0.9833	0.8489	60
fear	0.8182	0.9000	0.8571	60
joy	0.4583	0.3667	0.4074	60
neutral	0.4000	0.3000	0.3429	60
sadness	0.8293	0.5667	0.6733	60
surprise	0.5397	0.5667	0.5528	60
accuracy			0.6310	420
macro avg	0.6223	0.6310	0.6172	420
weighted avg	0.6223	0.6310	0.6172	420

Confusion Matrix:
[[44 3 2 4 5 0 2]
[1 59 0 0 0 0 0]
[2 1 54 2 0 0 1]
[5 7 1 22 11 2 12]
[12 4 3 10 18 3 10]
[5 5 4 2 6 34 4]
[9 0 2 8 5 2 34]]

4) CNN Multi-Input (Mel + MFCC)

Accuracy: 0.5333

Classification Report:

	precision	recall	f1-score	support
anger	0.4894	0.3833	0.4299	60
disgust	0.7966	0.7833	0.7899	60
fear	0.6329	0.8333	0.7194	60
joy	0.4545	0.2500	0.3226	60
neutral	0.5106	0.4000	0.4486	60
sadness	0.5429	0.6333	0.5846	60
surprise	0.3176	0.4500	0.3724	60
accuracy			0.5333	420
macro avg	0.5349	0.5333	0.5239	420
weighted avg	0.5349	0.5333	0.5239	420

Confusion Matrix:

```
[[23  3  5  7  8  4 10]
 [ 1 47  1  0  3  7  1]
 [ 1  0 50  0  0  1  8]
 [ 6  5  7 15  5  5 17]
 [ 5  2  4  4 24  9 12]
 [ 3  0  6  3  0 38 10]
 [ 8  2  6  4  7  6 27]]
```

5) ResNet18 + Mel Spectrogram

Final Accuracy: 0.7024

Classification Report:

	precision	recall	f1-score	support
anger	0.76	0.58	0.66	60
disgust	0.95	0.92	0.93	60
fear	0.71	0.97	0.82	60
joy	0.45	0.55	0.49	60
neutral	0.52	0.28	0.37	60
sadness	0.79	0.95	0.86	60
surprise	0.73	0.67	0.70	60
accuracy			0.70	420
macro avg	0.70	0.70	0.69	420
weighted avg	0.70	0.70	0.69	420

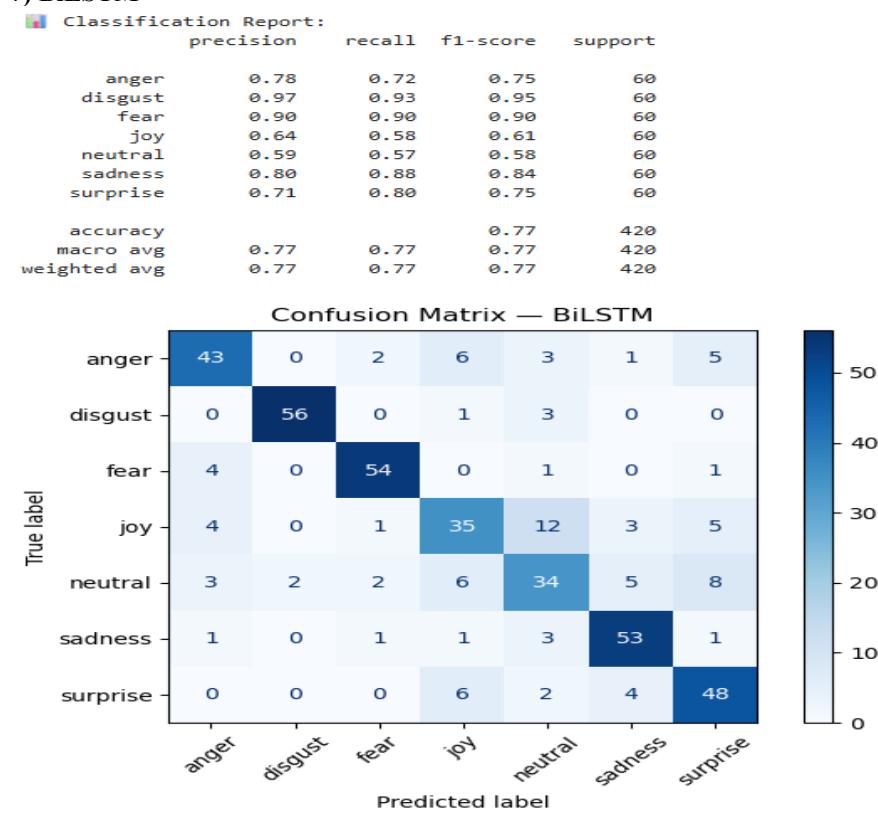
6) EfficientNetB0 + RGB Fusion

Final Accuracy: 0.7833

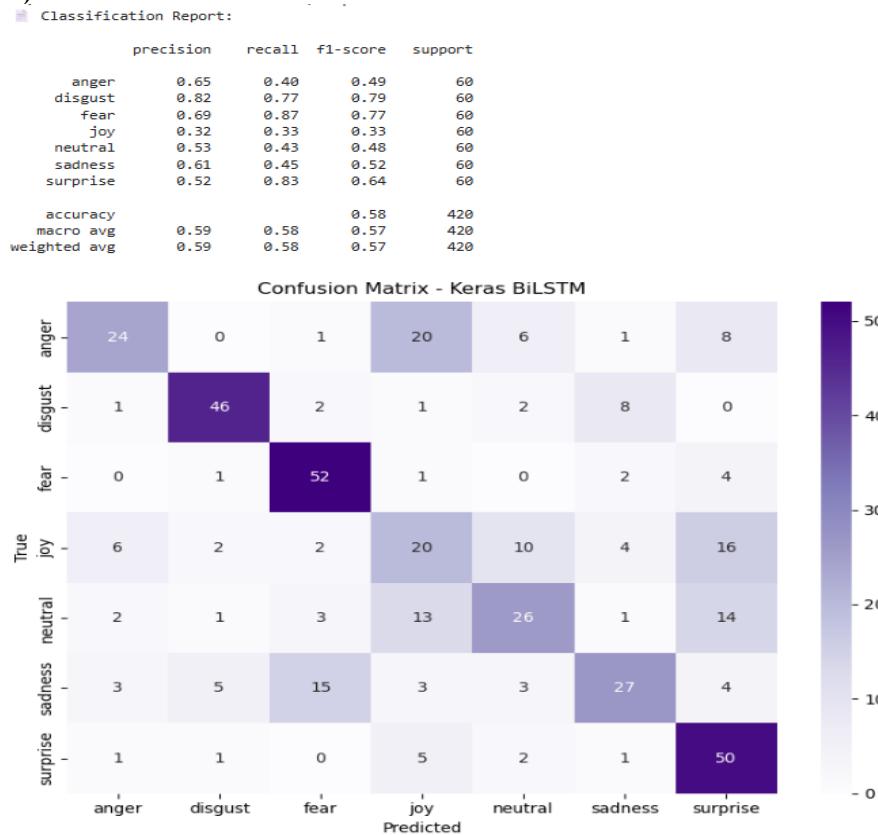
Classification Report:

	precision	recall	f1-score	support
anger	0.84	0.77	0.80	60
disgust	0.82	0.98	0.89	60
fear	0.89	0.95	0.92	60
joy	0.62	0.63	0.63	60
neutral	0.68	0.47	0.55	60
sadness	0.90	0.92	0.91	60
surprise	0.70	0.77	0.73	60
accuracy			0.78	420
macro avg	0.78	0.78	0.78	420
weighted avg	0.78	0.78	0.78	420

7) BiLSTM



8) Keras BiLSTM / GRU



9) Torch MLP

```
Classification Report:
precision    recall  f1-score   support
          0       0.78      0.83      0.81      60
          1       0.91      1.00      0.95      60
          2       0.90      0.93      0.92      60
          3       0.75      0.50      0.60      60
          4       0.62      0.68      0.65      60
          5       0.89      0.95      0.92      60
          6       0.86      0.83      0.85      60

accuracy                           0.82      420
macro avg       0.82      0.82      0.81      420
weighted avg    0.82      0.82      0.81      420

Confusion Matrix:
[[50  0  0  3  5  1  1]
 [ 0 60  0  0  0  0  0]
 [ 0  0 56  0  1  2  1]
 [ 6  4  2 30 14  1  3]
 [ 5  1  2  6 41  2  3]
 [ 0  1  2  0  0 57  0]
 [ 3  0  0  1  5 150]]
✓ Final Accuracy: 0.8190 | F1 Macro: 0.8135
```

10) Keras MLP

```
Classification Report:
precision    recall  f1-score   support
          0       0.75      0.78      0.76      60
          1       0.90      1.00      0.94      60
          2       0.89      0.90      0.89      60
          3       0.66      0.55      0.60      60
          4       0.61      0.63      0.62      60
          5       0.85      0.95      0.90      60
          6       0.86      0.72      0.78      60

accuracy                           0.79      420
macro avg       0.79      0.79      0.79      420
weighted avg    0.79      0.79      0.79      420

Confusion Matrix:
[[47  1  0  8  3  0  1]
 [ 0 60  0  0  0  0  0]
 [ 2  0 54  0  1  2  1]
 [ 5  3  4 33 11  2  2]
 [ 5  2  2  7 38  3  3]
 [ 1  1  0  0  1 57  0]
 [ 3  0  1  2  8  3 43]]
✓ Final Accuracy: 0.7905 | F1 Macro: 0.7863
```

11) CNN + BiLSTM (Mel)

```
Final Report:
precision    recall  f1-score   support
anger        0.76      0.63      0.69      30
disgust       0.95      0.67      0.78      30
fear          0.87      0.67      0.75      30
joy           0.37      0.50      0.42      30
neutral       0.40      0.53      0.46      30
sadness       0.62      0.70      0.66      30
surprise      0.69      0.60      0.64      30

accuracy                           0.61      210
macro avg       0.67      0.61      0.63      210
weighted avg    0.67      0.61      0.63      210

[[19  0  0  6  1  3  1]
 [ 1 20  0  2  0  5  2]
 [ 0  0 20  4  1  1  4]
 [ 1  0  1 15 11  1  1]
 [ 2  1  1  7 16  3  0]
 [ 0  0  1  3  5 21  0]
 [ 2  0  0  4  6  0 18]]
```

1) WavLM (Large – + MLP)

Classification Report:

	precision	recall	f1-score	support
anger	0.83	0.72	0.77	60
disgust	0.95	0.98	0.97	60
fear	0.88	0.98	0.93	60
joy	0.65	0.58	0.61	60
neutral	0.64	0.57	0.60	60
sadness	0.89	0.92	0.90	60
surprise	0.74	0.87	0.80	60
accuracy			0.80	420
macro avg	0.80	0.80	0.80	420
weighted avg	0.80	0.80	0.80	420

2) WavLM (Base+ – MLP)

Classification Report:

	precision	recall	f1-score	support
anger	0.75	0.68	0.71	60
disgust	0.87	1.00	0.93	60
fear	0.89	0.97	0.93	60
joy	0.60	0.52	0.55	60
neutral	0.62	0.40	0.48	60
sadness	0.84	0.88	0.86	60
surprise	0.64	0.82	0.72	60
accuracy			0.75	420
macro avg	0.74	0.75	0.74	420
weighted avg	0.74	0.75	0.74	420

3) Wav2Vec2.0 (Large – + MLP)

Classification Report:

	precision	recall	f1-score	support
anger	0.54	0.55	0.55	60
disgust	0.71	0.88	0.79	60
fear	0.70	0.92	0.79	60
joy	0.55	0.30	0.39	60
neutral	0.53	0.50	0.51	60
sadness	0.70	0.65	0.67	60
surprise	0.61	0.60	0.61	60
accuracy			0.63	420
macro avg	0.62	0.63	0.61	420
weighted avg	0.62	0.63	0.61	420

4) Wav2Vec2.0 SER (Pretrained SER)

Classification Report:

	precision	recall	f1-score	support
anger	0.43	0.52	0.47	60
disgust	0.71	0.88	0.79	60
fear	0.75	0.83	0.79	60
joy	0.40	0.35	0.38	60
neutral	0.36	0.25	0.29	60
sadness	0.59	0.67	0.62	60
surprise	0.45	0.33	0.38	60
accuracy			0.55	420
macro avg	0.53	0.55	0.53	420
weighted avg	0.53	0.55	0.53	420

5) HuBERT (Large – MLP)

Classification Report:

	precision	recall	f1-score	support
anger	0.69	0.63	0.66	60
disgust	0.91	0.87	0.89	60
fear	0.76	0.90	0.82	60
joy	0.52	0.55	0.54	60
neutral	0.64	0.48	0.55	60
sadness	0.65	0.80	0.72	60
surprise	0.67	0.62	0.64	60
accuracy			0.69	420
macro avg	0.69	0.69	0.69	420
weighted avg	0.69	0.69	0.69	420

6) HuBERT (Base – + MLP)

Classification Report:

	precision	recall	f1-score	support
anger	0.78	0.70	0.74	60
disgust	0.87	0.97	0.91	60
fear	0.87	0.97	0.91	60
joy	0.59	0.60	0.60	60
neutral	0.58	0.53	0.56	60
sadness	0.87	0.90	0.89	60
surprise	0.76	0.68	0.72	60
accuracy			0.76	420
macro avg	0.76	0.76	0.76	420
weighted avg	0.76	0.76	0.76	420

7) HuBERT (Base – Fine-tuned)

⌚ Test Accuracy: **0.6643**

📊 Classification Report:

	precision	recall	f1-score	support
anger	0.72	0.57	0.64	63
disgust	1.00	0.92	0.96	59
fear	0.83	0.90	0.86	58
joy	0.30	0.12	0.17	66
neutral	0.52	0.66	0.58	61
sadness	0.90	0.78	0.83	58
surprise	0.44	0.80	0.57	55
accuracy			0.66	420
macro avg	0.67	0.68	0.66	420
weighted avg	0.67	0.66	0.65	420

📊 Confusion Matrix:

[[36 0 2 15 6 0 4]
[3 54 0 1 1 0 0]
[0 0 52 0 0 1 5]
[1 0 3 8 22 2 30]
[2 0 1 1 40 1 16]
[5 0 3 2 3 45 0]
[3 0 2 0 5 1 44]]

8) WavLM (Base+ – Fine-tuned)

⌚ Test Accuracy: **0.5452**

📊 Classification Report:

	precision	recall	f1-score	support
anger	0.44	0.71	0.55	68
disgust	0.86	0.98	0.92	61
fear	1.00	0.32	0.48	60
joy	0.29	0.21	0.24	57
neutral	0.52	0.51	0.51	65
sadness	0.84	0.74	0.79	50
surprise	0.27	0.34	0.30	59
accuracy			0.55	420
macro avg	0.60	0.54	0.54	420
weighted avg	0.60	0.55	0.54	420

📊 Confusion Matrix:

[[48 4 0 4 5 1 6]
[0 60 0 0 0 1 0]
[5 0 19 3 4 1 28]
[19 2 0 12 7 3 14]
[19 0 0 6 33 1 6]
[2 4 0 6 1 37 0]
[15 0 0 10 14 0 20]]

9) Wav2Vec2.0 (Base – Fine-tuned)

⌚ Test Accuracy: **0.7786**

📊 Classification Report:

	precision	recall	f1-score	support
anger	0.74	0.84	0.79	50
disgust	0.94	0.95	0.94	63
fear	0.92	0.92	0.92	64
joy	0.54	0.51	0.53	51
neutral	0.62	0.67	0.65	64
sadness	0.75	0.97	0.85	60
surprise	0.95	0.57	0.72	68
accuracy			0.78	420
macro avg	0.78	0.78	0.77	420
weighted avg	0.79	0.78	0.77	420

📊 Confusion Matrix:

[[42 1 1 1 2 3 0]
[1 60 1 0 1 0 0]
[1 0 59 0 0 4 0]
[5 0 0 26 15 5 0]
[7 0 0 7 43 5 2]
[0 0 1 1 0 58 0]
[1 3 2 13 8 2 39]]

Model Training and Evaluation

1) ML MODELS

<i>Model</i>	<i>Best Feature Set</i>	<i>Accuracy</i>	<i>F1-Macro</i>	<i>Key Parameters (GridSearchCV)</i>
<i>SVM</i>	Boruta + Autoencoder	86.43%	86.51%	C=0.1, gamma=0.1, kernel=poly
<i>Random Forest</i>	Boruta	80.24%	79.48%	n_estimators=100, max_depth=10
<i>XGBoost</i>	Boruta + Autoencoder	82.86%	82.53%	learning_rate=0.05, max_depth=5
<i>KNN</i>	Boruta + Autoencoder	82.14%	81.77%	n_neighbors=3, weights='distance', metric='euclidean'
<i>Sklearn-MLP</i>	Boruta	80.24%	79.52%	hidden=(256,128), solver=adam, learning=adaptive
<i>Gradient Boosting</i>	Boruta	83.57%	83.46%	n_estimators=400, loss='log_loss'
<i>Bagging</i>	Boruta	83.33%	83.23%	base_estimator=DT(max_depth=5), n_estimators=150
<i>Hard Voting</i>	Boruta	84.29%	84.09%	majority vote from top 3 models
<i>Soft Voting</i>	Boruta	84.29%	84.13%	probability-based averaging

2) DL MODELS

<i>Model</i>	<i>Input Type</i>	<i>Accuracy</i>	<i>F1-Macro</i>	<i>Notes</i>
<i>CNN (Mel Spectrogram)</i>	Mel Image (128x128)	76.43%	76.70%	Early stopping, dropout
<i>CNN (MFCC Image)</i>	MFCC (128x128)	71.90%	70.85%	Delta + Delta ² not used here
<i>CNN (RGB Fusion)</i>	MFCC + Δ + Δ²	76.90%	75.82%	RGB Channel Mapping
<i>CNN (Multi-Input)</i>	MFCC + Mel Fusion	74.05%	72.92%	Two-branch CNN
<i>ResNet18 (Transfer)</i>	Mel (224x224)	70.24%	69.20%	Pretrained on ImageNet
<i>EfficientNetB0 (RGB Fusion)</i>	RGB Image (224x224)	78.33%	78.17%	Best DL model (Sadness, Fear ↑)
<i>BiLSTM (Keras)</i>	[T,120]	77.14%	77.01%	Bidirectional + Dropout
<i>BiLSTM (Torch)</i>	[T,120]	76.19%	75.42%	Torch implementation
<i>CNN + BiLSTM</i>	Mel image → CNN+LSTM	77.14%	75.92%	Mixed architecture
<i>Torch MLP</i>	Feature vector	83.10%	82.88%	ML-style pipeline
<i>Keras MLP</i>	Feature vector	58.10%	57.00%	Comparable to Torch MLP

3) SOTA MODELS

<i>Model Type</i>	<i>Version / Name</i>	<i>Approach</i>	<i>Accuracy</i>	<i>F1-Macro</i>
<i>WavLM</i>	microsoft/wavlm-large	SSL Feature + MLP	75.29%	74.66%
<i>WavLM</i>	microsoft/wavlm-base-plus	SSL Feature + MLP	80.54%	80.12%
<i>Wav2Vec2.0</i>	facebook/wav2vec2-large-960h-lv60	SSL Feature + MLP	63.10%	61.00%
<i>Wav2Vec2.0 SER</i>	ehcalabres/wav2vec2-lg-xlsr-en-speech	SSL Feature + MLP	55.11%	53.94%
<i>HuBERT</i>	facebook/hubert-large-ls960-ft	SSL Feature + MLP	69.05%	69.12%
<i>HuBERT</i>	facebook/hubert-base-ls960	SSL Feature + MLP	76.19%	76.03%
<i>HuBERT</i>	facebook/hubert-base-ls960	End-to-End Fine-tune	66.43%	66.20%
<i>WavLM</i>	microsoft/wavlm-base-plus	End-to-End Fine-tune	54.52%	54.01%
<i>Wav2Vec2.0</i>	facebook/wav2vec2-base	End-to-End Fine-tune	77.86%	77.60%

APPENDIX C

ML Best Predict Model → SVM

```
Model: svm_1 - svm_1_meld_features_selected_boruta_autoencoder_1_balanced
precision    recall   f1-score   support
angry        1.0000   1.0000   1.0000      31
disgust      1.0000   1.0000   1.0000       5
fear         1.0000   1.0000   1.0000       4
joy          0.9211   0.9722   0.9459      36
neutral      0.9860   0.9792   0.9826     144
sadness      1.0000   1.0000   1.0000      20
surprise     1.0000   0.9583   0.9787      24

accuracy           0.9811      264
macro avg       0.9867      0.9871      0.9867      264
weighted avg    0.9816      0.9811      0.9812      264

[[ 31  0  0  0  0  0  0]
 [ 0  5  0  0  0  0  0]
 [ 0  0  4  0  0  0  0]
 [ 0  0  0  35  1  0  0]
 [ 0  0  0  3 141  0  0]
 [ 0  0  0  0  0  20  0]
 [ 0  0  0  0  1  0  23]]
```

ML Models Prediction and Experimental Results

Model	Feature Set Used	Accuracy (%)	F1-macro (%)
SVM	Boruta + Autoencoder	98.11	98.67
Random Forest	Boruta	87.12	85.22
XGBoost	Boruta	93.94	92.78
K-Nearest Neighbors	Boruta \cap Autoencoder	83.71	78.31
MLP (sklearn)	Boruta	87.88	85.06
Gradient Boosting	Boruta	93.56	95.81
Bagging Classifier	Boruta + Autoencoder	91.29	87.45
Hard Voting Classifier	Boruta	87.88	85.96
Soft Voting Classifier	Autoencoder	94.32	94.84

DL Best Predict Model → BiLSTM

```
Classification Report BiLSTM
precision    recall   f1-score   support
anger      1.0000   1.0000   1.0000     31
disgust    1.0000   1.0000   1.0000      5
fear       0.8000   1.0000   0.8889      4
joy        0.7200   1.0000   0.8372     36
neutral    1.0000   0.8403   0.9132    144
sadness    0.8333   1.0000   0.9091     20
surprise   0.8571   1.0000   0.9231     24

accuracy          0.9129     264
macro avg       0.8872   0.9772   0.9245     264
weighted avg    0.9332   0.9129   0.9149     264

[[ 31  0  0  0  0  0]
 [ 0  5  0  0  0  0]
 [ 0  0  4  0  0  0]
 [ 0  0  0  36  0  0]
 [ 0  0  1  14 121  4]
 [ 0  0  0  0  0  20]
 [ 0  0  0  0  0  24]]
```

DL Prediction and Experimental and Results

No	Model	Input Type	Accuracy	F1-Macro
1	CNN (Mel)	Mel Spectrogram (128×128)	89.02%	89.20%
2	CNN (MFCC)	MFCC Spectrogram (128×128)	87.12%	84.06%
3	CNN (RGB Fusion)	RGB Fusion (128×128)	65.91%	61.13%
4	CNN (Multi-Input)	MFCC + Mel Dual Input	62.12%	55.79%
5	ResNet18 (Mel)	Mel Spectrogram (224×224)	75.00%	74.50%
6	EfficientNetB0 (RGB)	RGB Fusion (224×224)	79.17%	77.72%
7	BiLSTM	Acoustic Features (1D)	91.29%	92.45%
8	Keras-BiLSTM	Acoustic Features (1D)	71.97%	68.92%
9	Torch-MLP	Handcrafted Features	78.41%	76.21%
10	Keras-MLP	Handcrafted Features	85.98%	85.48%
11	CNN + BiLSTM	Mel Spectrogram (128×128)	77.65%	75.22%

SOTA Best Predict Model → Wav2Vec2.0 (Fine-Tuned)

```
Classification Report Wav2Vec2.0 Fine-Tuned
precision    recall   f1-score   support
anger        0.7561   1.0000   0.8611      31
disgust       0.5000   1.0000   0.6667       5
fear         0.7500   0.7500   0.7500       4
joy          0.6970   0.6389   0.6667      36
neutral      0.9237   0.8403   0.8800     144
sadness      0.8696   1.0000   0.9302      20
surprise      0.8636   0.7917   0.8261      24

accuracy           0.8409      264
macro avg       0.7657   0.8601   0.7973      264
weighted avg    0.8529   0.8409   0.8416      264

[[ 31  0  0  0  0  0]
 [ 0  5  0  0  0  0]
 [ 0  0  3  1  0  0]
 [ 4  1  0  23  7  0]
 [ 6  3  1  8  121  3]
 [ 0  0  0  0  20  0]
 [ 0  1  0  1  3  0]
 [ 0  0  0  0  0  19]]
```

SOTA Prediction and Experimental Results

No	Model	Accuracy (%)	F1-macro (%)
1	WavLM (Large) + MLP	82.95	79.11
2	WavLM (Base+) + MLP	76.89	72.58
3	Wav2Vec2.0 (facebook/960h) + MLP	70.45	66.64
4	Wav2Vec2.0 (ehcalabres) + MLP	65.15	63.36
5	HuBERT (Large) + MLP	77.27	74.18
6	HuBERT (Base) + MLP	76.14	73.49
7	HuBERT (Fine-tuned, PyTorch)	77.65	79.45
8	WavLM (Fine-tuned, PyTorch)	63.64	49.53
9	Wav2Vec2.0 (Fine-tuned, PyTorch)	84.09	79.73