

Faculty of Informatics and Computer Science

Information System/Computer Networks/Software Engineering/ Computer Science

Project Title

**By: Student Name**

Supervised By

**Supervisor Name**

xxxxx

**June 2022**

Abstract

Summary of the dissertation ***within one page***. Unnumbered chapter headings, as above, are entered using the *Unnumbered 1* paragraph style. The *Unnumbered 1* style automatically starts a new page.

This template starts the page numbering at the foot of this page. While you are printing drafts, you might find it useful to add the printing date and time into the footer – to help you, and your supervisor, tell which version is most current.

**Note: You are required to submit one extra copy of your title page and Abstract.**

It is suggested that the abstract be structured as follows:

* Problem: What you tackled, and why this needed a solution
* Objectives: What you set out to achieve, and how this addressed the problem
* Methodology: How you went about solving the problem
* Achievements: What you managed to achieve, and how far it meets your objectives.

Turnitin Report

*Add the report*

Acknowledge

Acknowledge anyone who has helped you in your work such as your supervisor, technical support staff, fellow students or external organisations. Acknowledge the source of any work that is not your own.

Table of Contents

The table of contents below is automatically generated from the paragraphs of style *Heading N* and *Unnumbered N*. To update this after revisions, right-click in the table and choose *Update Field* for the entire table.

Table of Contents

[Abstract 2](#_Toc200151068)

[Turnitin Report 3](#_Toc200151069)

[Acknowledge 4](#_Toc200151070)

[Table of Contents 5](#_Toc200151071)

[List of Figures 8](#_Toc200151072)

[List of Tables 9](#_Toc200151073)

[1 Introduction 10](#_Toc200151074)

[1.1 Overview 10](#_Toc200151075)

[1.2 Problem Statement 10](#_Toc200151076)

[Write the problem statement 10](#_Toc200151077)

[1.3 Scope and Objectives 10](#_Toc200151078)

[Define the scope and objectives of your project. 10](#_Toc200151079)

[1.4 Report Organization (Structure) 10](#_Toc200151080)

[1.5 Work Methodology 11](#_Toc200151081)

[1.6 Work Plan (Gantt chart) 11](#_Toc200151082)

[2 Related Work (State-of-The-Art) 12](#_Toc200151083)

[2.1 Background 12](#_Toc200151084)

[2.2 Literature Survey 12](#_Toc200151085)

[2.3 Analysis of the Related Work 12](#_Toc200151086)

[3 System Design & Architecture 13](#_Toc200151087)

[3.1 Speech Emotion Recognition (SER) Part 13](#_Toc200151088)

[3.1.1 Overview Pipeline Diagram 13](#_Toc200151089)

[3.1.1.1 High‐Level Pipeline Blocks 13](#_Toc200151090)

[3.1.1.2 Input/Output Shapes for Each Block 17](#_Toc200151091)

[3.1.2 Module Descriptions 18](#_Toc200151092)

[3.1.2.1 Data Parsers 18](#_Toc200151093)

[3.1.2.2 Preprocessing 19](#_Toc200151094)

[3.1.2.3 Feature Extraction 20](#_Toc200151095)

[3.1.2.4 Data Augmentation 20](#_Toc200151096)

[3.1.2.5 SER Model (Conv1D) 21](#_Toc200151097)

[3.1.2.6 Evaluation 23](#_Toc200151098)

[3.1.3 Data Flow & Integration 24](#_Toc200151099)

[3.1.3.1 Raw Audio to Label Mapping 24](#_Toc200151100)

[3.1.3.2 Preprocessing & Feature Extraction 25](#_Toc200151101)

[3.1.3.3 Train/Val/Test Partitioning & Standardization 25](#_Toc200151102)

[3.1.3.4 Data Augmentation (Training Only) 26](#_Toc200151103)

[3.1.3.5 Model Training & Inference 26](#_Toc200151104)

[3.1.3.6 Evaluation & Reporting 27](#_Toc200151105)

[3.1.3.7 Integration Between Modules 27](#_Toc200151106)

[3.1.3.8 Justification of Architectural Choices 28](#_Toc200151107)

[4 Methodology 29](#_Toc200151108)

[4.1 Speech Emotion recognition (SER) part 29](#_Toc200151109)

[4.1.1 Datasets & Collection 29](#_Toc200151110)

[4.1.1.1 Overview of Chosen Corpora 29](#_Toc200151111)

[4.1.2 Preprocessing Pipeline 34](#_Toc200151112)

[4.1.2.1 Audio Loading 34](#_Toc200151113)

[4.1.2.2 Noise and Silence Handling 35](#_Toc200151114)

[4.1.2.3 Feature Extraction 35](#_Toc200151115)

[4.1.3 Data Augmentation 38](#_Toc200151116)

[4.1.3.1 Additive Gaussian Noise 38](#_Toc200151117)

[4.1.3.2 Time Stretch (0.8×) 38](#_Toc200151118)

[4.1.3.3 Time Shift (±5,000 samples) 39](#_Toc200151119)

[4.1.3.4 Pitch Shift (+0.7 Semitones) 39](#_Toc200151120)

[4.1.3.5 Summary of Augmentation Outputs 40](#_Toc200151121)

[4.1.4 Model Design & Training 40](#_Toc200151122)

[4.1.4.1 Data Splitting & Label Preparation 40](#_Toc200151123)

[4.1.4.2 Model Architecture: Conv1D Network 41](#_Toc200151124)

[4.1.4.3 Training Regimen 44](#_Toc200151125)

[4.1.4.4 Training Procedure 46](#_Toc200151126)

[4.1.5 Evaluation Metrics 47](#_Toc200151127)

[4.1.5.1 Overall Accuracy 47](#_Toc200151128)

[4.1.5.2 Confusion Matrix 48](#_Toc200151129)

[4.1.5.3 Precision, Recall, and F₁‐Score (Per Class) 48](#_Toc200151130)

[4.1.5.4 Validation vs. Test Metrics 49](#_Toc200151131)

[4.1.5.5 Visualization Techniques 50](#_Toc200151132)

[5 Implementation 52](#_Toc200151133)

[6 Testing and evaluation 53](#_Toc200151134)

[6.1 Testing 53](#_Toc200151135)

[6.2 Evaluation 53](#_Toc200151136)

[7 Results and Discussions 54](#_Toc200151137)

[8 Conclusions and Future Work 55](#_Toc200151138)

[8.1 Summary 55](#_Toc200151139)

[8.2 Future Work 55](#_Toc200151140)

[References 56](#_Toc200151141)

[Appendix I 57](#_Toc200151142)

[Appendix II 58](#_Toc200151143)

List of Figures

Similarly you can automatically generate a list of figures from paragraphs of style *Figure &Tables*. To update this after revisions, right-click in the table and choose *Update Field* for the entire table.

[Figure 1 Highly Technical Diagram 11](#_Toc452468221)

List of Tables

# Introduction

For editorial consistency, it is important to use Word styles properly. Word 2003 onwards has so-called quick styles. If the styles referred to below are not visible on the *Home* ribbon in the *Styles* category, choose *Apply Styles* from the down arrow at the bottom right of the *Styles* category. Styles can then be applied from the drop-down box. To make a style visible as a quick style, choose *Apply Styles*, then click *Styles* (the *AA* icon), then click on the drop-down list for a style, and then *Add to Quick Style Gallery*.

Chapters are entered using the Heading 1 paragraph style. The Heading 1 style automatically moves to the start of a new page, and supplies the next chapter number. The new paragraph when you press Return after a heading automatically uses the *Body First* paragraph style (like this one, with no indent on the first line).

However most text uses the Body Text paragraph style (like this one, with 11 point Times New Roman, 1.5 line spacing, single-sided pages). Enter most text using the Body Text paragraph style. The new paragraph when you press Return after a Body First paragraph automatically uses the Body Text paragraph style.

In general, use the default spacing that headings and paragraphs give you. Avoid using new-lines or spaces to format text. If you need to use quotes, preferably use single curly quotes ‘…’. If you wish to emphasise something, usually use *italic font*.

**Remember to Save frequently while you are working!**

## Overview

Give the background to your project and context of what you have done. Sections are entered using the *Heading 2* paragraph style – th*e Heading 2* style automatically supplies the next section number.

## Problem Statement

Write the problem statement

## [Scope](http://www.cs.stir.ac.uk/~kjt/research/conformed.html) and Objectives

Define the scope and objectives of your project.

## Report Organization (Structure)

Briefly overview the contents of what follows in the dissertation.

## Work Methodology

## Work Plan (Gantt chart)

# Related Work (State-of-The-Art)

Summarise current knowledge and what others have done in the various topics of your dissertation – in the application area and in the various technologies that you might have used or did use. Write for someone familiar with computing, but not necessarily expert in the particular topics of your project. Give references to other work by using *cross-references* to entries in the References section, like this ‎[2].

diss-fig

Figure 1 Highly Technical Diagram

As an example of a figure, consider ‎Figure 1. Captions are entered using the *Figure* paragraph style. The figure below is placed in a *Body Centre* paragraph, which is set up in this document to have an automatic *Figure* paragraph following it. *Figure* has automatic figure numbering, and it is possible to make *cross-references* to figures. Move large figures to the top of the next page, *past any other text,* rather than having a big gap in the text.

## Background

## Literature Survey

## Analysis of the Related Work

# System Design & Architecture

## Speech Emotion Recognition (SER) Part

This chapter presents a high‐level architectural overview of the Speech Emotion Recognition (SER) subsystem, delineating its constituent modules, their interconnections, and the data flows that transform raw audio recordings into quantitative emotion predictions. Section 7.1 illustrates the complete pipeline via block diagrams, highlighting each stage’s inputs, outputs, and core responsibilities. Section 7.2 provides detailed narrative descriptions of individual modules—Data Parsers, Preprocessing, Feature Extraction, Augmentation, Model, and Evaluation—explaining how each contributes to overall system goals without delving into implementation minutiae. Finally, Section 7.3 depicts the end‐to‐end data flow, clarifying how a raw file traverses parsers and preprocessors, undergoes feature extraction and augmentation, enters the Conv1D model, and yields emotion metrics for downstream reporting. Together, these sections justify every architectural choice in terms of scalability, reproducibility, and alignment with real‐world constraints.

### Overview Pipeline Diagram

#### High‐Level Pipeline Blocks

At the highest level, the SER subsystem comprises six sequential stages:

1. **Data Acquisition & Parsing**
2. **Preprocessing**
3. **Feature Extraction**
4. **Data Augmentation**
5. **Model Inference (Conv1D Network)**
6. **Evaluation & Reporting**

Below is a schematic representation:

┌──────────────────────────┐ ┌──────────────────────────┐

│ 1. Data Acquisition & │ │ 2. Preprocessing │

│ Parsing │───▶ │ • Load audio at 22 kHz │

│ • RAVDESS, CREMA-D, │ │ • Trim silence offset │

│ TESS, SAVEE files │ │ • Extract 2.5 s segment │

└──────────────────────────┘ │ → 55,125 samples │

└──────────────────────────┘

│

▼

┌──────────────────────────┐

│ 3. Feature Extraction │

│ • Compute MFCC (40×100) │

│ • Compute ZCR (1×100) │

│ • Compute RMS (1×100) │

│ → Stack to 42×100 block │

│ → Flatten → 4,200‐dim │

└──────────────────────────┘

│

▼

┌──────────────────────────┐

│ 4. Data Augmentation │

│ • Original waveform │

│ • Add noise variant │

│ • Time-stretch variant │

│ • Time-shift variant │

│ • Pitch-shift variant │

│ → Each produces a │

│ 4,200‐dim vector │

└──────────────────────────┘

│

▼

┌──────────────────────────┐

│ 5. SER Model (Conv1D) │

│ • Input: (4,200 × 1) │

│ • Conv1D → BatchNorm │

│ + MaxPool → … │

│ • Flatten → Dense → Softmax

│ • Output: 8‐class │

│ probability vector │

└──────────────────────────┘

│

▼

┌──────────────────────────┐

│ 6. Evaluation & Reporting│

│ • Compute confusion │

│ matrix, precision, │

│ recall, F₁‐score │

│ • Log metrics and │

│ prepare for reporting │

└──────────────────────────┘

Each block’s input/output shape and principal operation are summarized below, with justifications:

1. **Data Acquisition & Parsing**
   * **Input**: Raw WAV files from four corpora—RAVDESS, CREMA-D, TESS, SAVEE—organized into dataset‐specific directories.
   * **Output**: A unified table of file paths and corresponding canonical emotion labels (from eight classes: neutral, calm, happy, sad, angry, fearful, disgust, surprised).
   * **Justification**: Merging multiple corpora ensures diversity in speaker demographics, recording conditions, and expression styles. A single labeling scheme prevents confusion and simplifies subsequent modules.
2. **Preprocessing**
   * **Input**: A pair (“file path,” “emotion”).
   * **Operation**:
     + Load audio at 22,050 Hz in mono.
     + Skip the first 0.6 s to avoid extraneous silence or mic artifacts.
     + Extract a contiguous 2.5 s segment (55,125 samples). Zero‐pad if shorter, truncate if longer.
   * **Output**: A fixed‐length waveform array of size (55,125).
   * **Justification**: Standardizing to 2.5 s ensures consistent input lengths, avoiding variable‐length complicators in feature extraction. The 0.6 s offset centers on emotion‐laden speech, improving signal‐to‐noise ratio.
3. **Feature Extraction**
   * **Input**: 2.5 s waveform array (55,125 samples).
   * **Operation**:
     + Frame into 25 ms windows with 10 ms hop (resulting in ~225 frames).
     + Compute 40 Mel‐Frequency Cepstral Coefficients (MFCCs) per frame → (40 × 225). Truncate or pad to (40 × 100).
     + Compute Zero‐Crossing Rate (ZCR) per frame → (1 × 225), truncated/padded to (1 × 100).
     + Compute Root Mean Square (RMS) Energy per frame → (1 × 225), truncated/padded to (1 × 100).
     + Stack MFCC, ZCR, RMS into a (42 × 100) block; flatten to a 4,200‐element vector.
   * **Output**: Feature vector of shape (4,200).
   * **Justification**: MFCCs capture spectral envelope; ZCR highlights voicing/noise transitions; RMS encodes loudness. Truncation to 100 frames strikes a balance between temporal coverage and model complexity. Flattening allows use of a one‐dimensional convolution.
4. **Data Augmentation**
   * **Input**: The original 2.5 s waveform (55,125 samples).
   * **Operation**: Produce four additional variants per original waveform:  
     a. **Additive Noise**: Add scaled Gaussian noise (max 3.5 % of peak amplitude).  
     b. **Time Stretch**: Slow down by factor = 0.8, then trim/pad to 2.5 s.  
     c. **Time Shift**: Circularly shift by a random integer in [–5,000, +5,000] samples.  
     d. **Pitch Shift**: Raise pitch by +0.7 semitones, then trim/pad to 2.5 s.
   * Each augmented waveform passes through the same Feature Extraction pipeline to yield four additional 4,200‐dim vectors.
   * **Output**: Five 4,200‐element feature vectors per original file (1 original + 4 augmentations).
   * **Justification**: Augmentation fosters robustness to ambient noise, varied speaking rates, misalignments, and pitch differences, mitigating overfitting and improving generalization to real‐world variability.
5. **SER Model (Conv1D)**
   * **Input**: A single 4,200‐element feature vector, reshaped to (4,200 × 1).
   * **Architecture Overview**:
     + **Conv1D Block 1**: 512 filters, kernel=5, padding='same', activation='ReLU' → BatchNorm → MaxPool (pool=5, stride=2, padding='same') → Dropout (0.3) → Output shape ~ (2,100 × 512).
     + **Conv1D Block 2**: 512 filters, kernel=5, padding='same', activation='ReLU' → BatchNorm → MaxPool (5,2,'same') → Dropout (0.4) → Output shape ~ (1,050 × 512).
     + **Conv1D Block 3**: 256 filters, kernel=5, padding='same', activation='ReLU' → BatchNorm → MaxPool (5,2,'same') → Output shape ~ (525 × 256).
     + **Conv1D Block 4**: 256 filters, kernel=3, padding='same', activation='ReLU' → BatchNorm → MaxPool (5,2,'same') → Dropout (0.4) → Output shape ~ (263 × 256).
     + **Conv1D Block 5**: 128 filters, kernel=3, padding='same', activation='ReLU' → BatchNorm → MaxPool (3,2,'same') → Dropout (0.5) → Output shape ~ (132 × 128).
     + **Flatten** → Dense (512, activation='ReLU') → BatchNorm → Dropout (0.5) → Dense (8, activation='Softmax').
   * **Output**: An 8‐dimensional probability vector indicating predicted emotion.
   * **Justification**: Stacked Conv1D layers progressively abstract local temporal features (e.g., MFCC patterns, ZCR bursts) into higher‐level representations. Pooling reduces sequence length, emphasizing invariance to minor temporal shifts. Dropout regularizes deeper layers where overfitting risk is highest. The final Dense+Softmax transforms learned features into emotion probabilities.
6. **Evaluation & Reporting**
   * **Input**: Model predictions (8‐dimensional vectors) and ground‐truth labels (one‐hot).
   * **Operation**:
     + Compute the **Confusion Matrix**: 8×8 table of true vs. predicted counts.
     + Compute **Precision**, **Recall**, **F₁‐Score** for each emotion class.
     + Compute **Overall Accuracy**.
     + Aggregate metrics into a report (tables, visualizations).
   * **Output**: Quantitative performance metrics fed to downstream reporting modules (e.g., JSON logs, CSV summaries, or direct integration into higher‐level reports).
   * **Justification**: Class‐specific metrics reveal whether certain emotions (e.g., calm or disgust) are systematically misclassified. Overall accuracy contextualizes model efficacy. These metrics guide potential retraining, hyperparameter adjustments, or additional data collection.

#### Input/Output Shapes for Each Block

|  |  |  |
| --- | --- | --- |
| Stage | Input Shape | Output Shape |
| 1. Data Acquisition & Parsing | Raw WAV files (.wav) | Table of (<file\_path>, ) |
| 2. Preprocessing | — Single <file\_path>, <emotion> | 2.5 s waveform: (55,125 samples) |
| 3. Feature Extraction | 55,125 samples (2.5 s @22 kHz) | 4,200‐dim feature vector |
| 4. Data Augmentation | 55,125 samples (original waveform) | 5 × 4,200‐dim vectors (incl. orig) |
| 5. SER Model (Conv1D) | 4,200 × 1 tensor | 8 probabilities (float array) |
| 6. Evaluation & Reporting | Predictions (8 floats) + true labels | Metrics: confusion matrix, P/R/F₁, Acc |

* **Block 2→3**: The preprocessing module hands a fixed‐length waveform to feature extraction.
* **Block 3→4**: Feature extraction produces one 4,200‐dim vector per waveform; augmentation replicates this vector four more times, yielding five total.
* **Block 4→5**: Each augmented feature vector enters the Conv1D network individually.
* **Block 5→6**: The model’s Softmax output and corresponding ground‐truth labels feed into the Evaluation module.

### Module Descriptions

This section elaborates the responsibilities of each architectural block. It emphasizes “what” each module accomplishes and “why” those choices support overall objectives, without prescribing implementation details or parameters (which appear in Section 8). Each description notes the module’s input and output expectations and justifies how it contributes to robustness, modularity, and maintainability.

#### Data Parsers

**Role**

* Transform heterogeneous filename conventions from four corpora into a unified format of <file\_path>, <emotion\_label>.
* Ensure that every audio sample is paired with one of eight canonical emotion labels.

**Inputs**

* Four directory paths (one per corpus):
  1. .../ravdess/audio\_speech\_actors\_01-24/actor\_##/...
  2. .../crema/AudioWAV/...
  3. .../tess/TESS Toronto emotional speech set data/actor\_folder/...
  4. .../savee/ALL/...
* Each filename embeds an emotion code:
  1. RAVDESS: <ID-ID-EMOCode-...>.wav (e.g., 03 = happy).
  2. CREMA-D: <speaker>\_<sentence>\_<EMOCode>\_<intensity>\_<rep>.wav (e.g., HAP = happy).
  3. TESS: <actor>\_word\_<emotionCode>\_... .wav (e.g., ps = pleasant surprise).
  4. SAVEE: <speaker>\_<utteranceID>\_<EMOcode>\_<suffix>.wav (e.g., a = angry, sa = sad).

**Output**

* A single in‐memory table (e.g., a DataFrame) with columns:
  1. Path: Absolute file path string.
  2. Emotion: One of {neutral, calm, happy, sad, angry, fearful, disgust, surprised}.

**Justification**

* **Unified Label Taxonomy**: Mapping all corpus‐specific codes into eight shared labels simplifies downstream modules, which no longer need corpus‐aware logic.
* **Separation of Concerns**: By isolating file-naming peculiarities here, the rest of the pipeline can assume that every Emotion is valid and consistent, enhancing maintainability.
* **Modularity**: If a new corpus is added later, one need only implement a new parser without touching feature‐extraction or model‐training code.

#### Preprocessing

**Role**

* Convert a raw audio file from disk into a standardized, fixed‐length waveform array.
* Discard non‐essential segments (e.g., leading/trailing silence) via a fixed offset.

**Inputs**

* Path (string) and Emotion (string) from Data Parser.

**Operations (Conceptual)**

1. **Audio Loading**: Open WAV at 22,050 Hz, mono.
2. **Silence Offset**: Skip the first 0.6 s of audio to avoid cold‐start noise.
3. **Fixed‐Length Extraction**: Read the subsequent 2.5 s.
   * If the file is shorter than 3.1 s, zero‐pad to 2.5 s.
   * If longer, truncate any samples beyond 2.5 s.

**Outputs**

* A one‐dimensional array of length 55,125 samples, representing exactly 2.5 s of speech.

**Justification**

* **Consistency**: CNNs expect fixed‐length inputs. Uniform 2.5 s segments prevent dynamic resizing inside the model.
* **Emphasis on Speech**: Offset of 0.6 s removes initial non‐speech or microphone adjustment noise, so subsequent modules focus on emotion‐laden content.
* **Scalability**: 22 kHz is a standard sampling rate, balancing frequency resolution (up to ~11 kHz) against memory/compute costs.

#### Feature Extraction

**Role**

* Transform each fixed‐length waveform into a concise, informative feature vector capturing spectral, voicing, and energy cues.

**Inputs**

* 2.5 s waveform array (length = 55,125 samples).

**Operations (Conceptual)**

1. **Frame Splitting**: Segment the waveform into overlapping frames (25 ms length, 10 ms hop).
2. **MFCC Computation**: For each frame, compute 40 Mel‐frequency cepstral coefficients, yielding an array of shape (40 × T).
3. **ZCR Computation**: Compute one zero‐crossing rate value per frame, resulting in (1 × T).
4. **RMS Energy Computation**: Compute one RMS energy value per frame, resulting in (1 × T).
5. **Temporal Truncation/Padding**: Truncate or zero‐pad all sequences to exactly T = 100 frames, ensuring uniform feature matrix size.
6. **Concatenation & Flattening**: Stack MFCC (40×100), ZCR (1×100), and RMS (1×100) into a (42×100) block; then flatten into a 4,200-dimensional vector.

**Outputs**

* A 4,200-element feature vector (float32) representing spectral envelope (MFCC), voicing/noise transitions (ZCR), and loudness dynamics (RMS).

**Justification**

* **Complementary Cues**: MFCCs capture vocal‐tract resonances, ZCR flags high‐frequency, unvoiced segments, and RMS quantifies overall amplitude—together yielding a rich representation of emotion.
* **Fixed Dimensionality**: Flattening to length 4,200 allows straightforward input to a Conv1D network.
* **Temporal Truncation**: T = 100 frames (~1 s) sufficiently covers the emotionally salient portion of the utterance (given 0.6 s offset and 2.5 s total). Extending to all ~225 frames would nearly triple vector length (to 42 × 225 = 9,450), increasing model complexity with marginal improvement.

#### Data Augmentation

**Role**

* Introduce controlled variability into training data by synthesizing plausible acoustic perturbations.

**Inputs**

* 2.5 s waveform array (length = 55,125).

**Operations (Conceptual)**  
For each original waveform, derive four additional variants:

1. **Additive Noise**:
   * Generate Gaussian noise of equal length.
   * Scale by up to 3.5 % of the waveform’s peak amplitude (randomized).
   * Add to the original waveform.
2. **Time Stretch (0.8×)**:
   * Slow the waveform by 20 % without altering pitch.
   * Trim or zero‐pad back to 2.5 s.
3. **Time Shift (±5,000 samples)**:
   * Circularly shift the waveform by a random integer in [–5,000, +5,000] samples (~±0.23 s).
4. **Pitch Shift (+0.7 semitones)**:
   * Increase pitch by 0.7 semitones, preserving duration.
   * Trim or pad back to 2.5 s.

Each augmented waveform re‐enters the Feature Extraction module to yield a total of five 4,200-dim vectors per original file.

**Outputs**

* Five 4,200-element vectors (1 original + 4 augmentations).

**Justification**

* **Realism**: Ambient noise, speaking‐rate variability, misalignment, and vocal pitch shifts naturally occur in conversational and interview settings.
* **Generalization**: Augmentation reduces overfitting by forcing the model to learn emotion‐related patterns that persist amid these perturbations.
* **Class‐Balance Mitigation**: Underrepresented emotions (e.g., calm) gain additional synthetic samples, partially offsetting data skew.

#### SER Model (Conv1D)

**Role**

* Ingest 4,200-dim feature vectors and predict one of eight emotion categories via a multi‐layer convolutional architecture.

**Inputs**

* Single feature vector of shape (4,200). Before entering the network, it is reshaped to (4,200 × 1), effectively a one‐dimensional “time” series with one channel.

**Architecture (Conceptual)**

1. **Conv1D Block 1**
   * **Input**: (4,200 × 1)
   * **Operation**: Convolve with 512 filters (kernel size = 5, padding = ‘same’, ReLU), followed by Batch Normalization, Max Pooling (pool = 5, stride = 2), and Dropout (rate = 0.3).
   * **Output**: Approx. (2,100 × 512)
   * **Rationale**: Capture low-level spectral‐temporal correlations across adjacent MFCC, ZCR, and RMS features. Pooling reduces sequence length while preserving important patterns. Dropout regularizes first feature abstraction layer.
2. **Conv1D Block 2**
   * **Input**: (2,100 × 512)
   * **Operation**: 512 filters (kernel=5, ReLU) → BatchNorm → MaxPool (5,2) → Dropout (0.4).
   * **Output**: Approx. (1,050 × 512)
   * **Rationale**: Further refine mid-level representations. Increased dropout (0.4) combats overfitting as the network grows deeper.
3. **Conv1D Block 3**
   * **Input**: (1,050 × 512)
   * **Operation**: 256 filters (kernel=5, ReLU) → BatchNorm → MaxPool (5,2).
   * **Output**: Approx. (525 × 256)
   * **Rationale**: Halve the number of filters to reduce parameter count while maintaining capacity for capturing more complex patterns.
4. **Conv1D Block 4**
   * **Input**: (525 × 256)
   * **Operation**: 256 filters (kernel=3, ReLU) → BatchNorm → MaxPool (5,2) → Dropout (0.4).
   * **Output**: Approx. (263 × 256)
   * **Rationale**: Smaller kernel (3) captures finer details given reduced sequence length. Dropout continues to enforce regularization at deeper layers.
5. **Conv1D Block 5**
   * **Input**: (263 × 256)
   * **Operation**: 128 filters (kernel=3, ReLU) → BatchNorm → MaxPool (3,2) → Dropout (0.5).
   * **Output**: Approx. (132 × 128)
   * **Rationale**: Further abstraction to high-level representations (e.g., long-term prosodic trends). Dropout = 0.5 ensures resilience against overfitting at critical depth.
6. **Classification Head**
   * **Flatten**: (132 × 128) → 16,896 units.
   * **Dense**: 512 units, ReLU → BatchNorm → Dropout (0.5).
   * **Output Layer**: Dense (8 units, Softmax). Predicts probability for each emotion.
   * **Rationale**: Fully connected layer integrates all learned features across the temporal dimension. High dropout preserves generalization. Softmax yields a valid probability distribution over classes.

**Output**

* An 8‐element vector of probabilities corresponding to the eight emotions.

**Justification**

* **Conv1D vs. Dense‐Only**: Convolutions exploit local correlations among adjacent MFCC/ZCR/RMS features over time, whereas a fully connected network on raw features would ignore temporal locality.
* **Progressive Pooling**: Reduces sequence length fivefold per block, controlling model size while preserving salient features.
* **Batch Normalization**: Speeds training and stabilizes learning across layers.
* **Dropout Schedule**: Gradually increasing dropout rates (0.3 → 0.4 → 0.5) align with deeper layers’ greater tendency to overfit.
* **512→256→128 Filter Reduction**: Balances network capacity with computational efficiency; fewer filters at deeper layers focus on essential features.

#### Evaluation

**Role**

* Quantitatively assess model predictions using a suite of metrics, identifying both overall accuracy and class‐specific performance.

**Inputs**

* Model predictions: 8‐dimensional Softmax vectors (per sample).
* Ground‐truth one‐hot labels.

**Operations (Conceptual)**

1. **Confusion Matrix**: Tally true vs. predicted class counts in an 8×8 matrix.
2. **Precision & Recall (per class)**:
   * Precisionk\_k = TPk\_k / (TPk\_k + FPk\_k), where TPk\_k = confusionk,kk,k, FPk\_k = sum of confusioni,ki,k for i ≠ k.
   * Recallk\_k = TPk\_k / (TPk\_k + FN\_k\]), where FN\(\_k = sum of confusionk,jk,j for j ≠ k.
3. **F₁‐Score (per class)**: 2 × (Precisionk\_k × Recallk\_k) / (Precisionk\_k + Recallk\_k).
4. **Overall Accuracy**: (Sum of diagonal entries) / (Total samples).

**Outputs**

* Confusion matrix heatmap, per‐class precision/recall/F₁ scores, overall accuracy percentage. These metrics feed into higher-level reporting or visualization components.

**Justification**

* **Precision/Recall**: Critical when class distribution is imbalanced (e.g., fewer “calm” samples).
* **F₁‐Score**: Balances precision and recall to indicate the harmonic mean of correctness.
* **Confusion Matrix**: Diagnoses systematic misclassifications (e.g., “sad” frequently mistaken for “calm”), informing data collection or model adjustments.
* **Overall Accuracy**: Summarizes aggregate performance but is less informative for rare classes, hence complemented by class‐specific measures.

### Data Flow & Integration

In this section, we trace the journey of a single raw audio file through the SER subsystem, highlighting how data transforms at each juncture and how modules interact. We emphasize that each module exposes a well‐defined interface—specifying input expectations and output guarantees—thereby ensuring that the system remains extensible, testable, and maintainable.

#### Raw Audio to Label Mapping

1. **Raw File Ingestion**
   * The system scans all four source directories (RAVDESS, CREMA-D, TESS, SAVEE), listing WAV files. No audio is loaded at this stage; only file paths are recorded.
2. **Parsing Module**
   * For each file path, apply the corresponding parser:
     + **RAVDESS Parser**: Extract the two-digit emotion code from filename (e.g., “03” → “happy”).
     + **CREMA-D Parser**: Split filename by underscores; map the third token (e.g., “HAP”) to “happy.”
     + **TESS Parser**: Identify the token (e.g., “ps”) for “pleasant surprise” → map to “surprised.”
     + **SAVEE Parser**: Extract the EMO substring (e.g., “a” → “angry,” “sa” → “sad”).
   * **Output**: A table (DataFrame) with rows: <file\_path>, <emotion\_label> for all files in all corpora.
3. **Shuffling & Splitting**
   * The master table is shuffled with a fixed random seed to randomize order.
   * A stratified split divides the dataset into training (90 %), validation (5 %), and test (5 %), preserving emotion class proportions.
   * **Integration Note**: The splitting occurs prior to preprocessing to ensure that augmented samples (introduced later) are appended only to the training partition—preventing data leakage into validation/test.

#### Preprocessing & Feature Extraction

1. **Preprocessing** (per subset)
   * Each file in training, validation, and test sets is processed independently:  
     a. **Load Audio**: Read the WAV file at 22,050 Hz, mono; represent as a 1D array of raw samples.  
     b. **Offset & Truncate**: Skip the first 0.6 s; extract exactly 2.5 s.  
     c. **Zero‐Padding**: If the file is shorter than 3.1 s, pad the 2.5 s window with zeros at the end.
   * **Output**: A fixed‐length waveform array (length = 55,125) for each sample.
2. **Feature Extraction** (applies to every subset)
   * The 55,125‐sample waveform is segmented into overlapping frames (25 ms with 10 ms hop), producing ~225 frames.
   * **MFCC**: Compute 40 coefficients per frame → (40 × 225). Truncate or pad to (40 × 100).
   * **ZCR**: Compute one value per frame → (1 × 225). Truncate/pad to (1 × 100).
   * **RMS**: Compute one value per frame → (1 × 225). Truncate/pad to (1 × 100).
   * Stack MFCC, ZCR, RMS into a (42 × 100) block, then flatten to a 4,200‐dim vector.
   * **Output**: Feature matrix:
     + Training features: Xraw\_train∈RNtrain×4,200X\_{\text{raw\\_train}} ∈ \mathbb{R}^{N\_{\text{train}} × 4{,}200}
     + Validation features: Xraw\_val∈RNval×4,200X\_{\text{raw\\_val}} ∈ \mathbb{R}^{N\_{\text{val}} × 4{,}200}
     + Test features: Xraw\_test∈RNtest×4,200X\_{\text{raw\\_test}} ∈ \mathbb{R}^{N\_{\text{test}} × 4{,}200}

#### Train/Val/Test Partitioning & Standardization

1. **Standardization**
   * Compute mean vector μ∈R4 200\mu ∈ \mathbb{R}^{4\,200} and standard deviation σ∈R4 200\sigma ∈ \mathbb{R}^{4\,200} from Xraw\_trainX\_{\text{raw\\_train}}.
   * Transform Xraw\_trainX\_{\text{raw\\_train}} to XtrainX\_{\text{train}}:

Xtrain, j←Xraw\_train, j−μjσj. X\_{\text{train},\,j} \leftarrow \frac{X\_{\text{raw\\_train},\,j} - \mu\_j}{\sigma\_j}.

* + Apply identical transform to Xraw\_valX\_{\text{raw\\_val}} and Xraw\_testX\_{\text{raw\\_test}} using the training μ,σ\mu, \sigma.
  + **Output**: Standardized feature matrices Xtrain,Xval,XtestX\_{\text{train}}, X\_{\text{val}}, X\_{\text{test}}, each of shape (N, 4 200)(N,\,4\,200).

1. **Label One‐Hot Encoding**
   * Convert emotion labels into one‐hot vectors of length 8.
   * Produce label matrices Ytrain∈{0,1}Ntrain×8Y\_{\text{train}} ∈ \{0,1\}^{N\_{\text{train}}×8}, Yval,YtestY\_{\text{val}}, Y\_{\text{test}}.

**Integration Note**: Standardization and encoding happen after splitting to avoid information leakage from validation/test into training. At this point, data in each subset is ready for augmentation (training only) and/or model ingestion (validation/test).

#### Data Augmentation (Training Only)

1. **Augmentation Module**
   * For each waveform in the training set (from step 4), generate four additional variants by applying: noise addition, time stretch, time shift, and pitch shift.
   * Each variant reenters the Feature Extraction pipeline, producing four new 4,200‐dim vectors.
   * **Output**: Extended training feature matrix Xtrain\_aug∈R5 Ntrain×4,200X\_{\text{train\\_aug}} ∈ \mathbb{R}^{5\,N\_{\text{train}} × 4{,}200} and corresponding labels Ytrain\_aug∈{0,1}5 Ntrain×8Y\_{\text{train\\_aug}} ∈ \{0,1\}^{5\,N\_{\text{train}}×8}.
   * **Integration Note**: Augmented samples are appended only to the training partition, ensuring that the validation and test sets remain pristine. This separation guards against inflated validation/test performance due to duplicated content.

#### Model Training & Inference

1. **Conv1D Model Module**
   * **Training Phase**:
     + Input: Xtrain\_augX\_{\text{train\\_aug}} (shape [5 Ntrain, 4 200][5\,N\_{\text{train}},\,4\,200]) along with Ytrain\_augY\_{\text{train\\_aug}}.
     + Training regimen uses Adam optimizer, categorical cross‐entropy loss, and callbacks (ModelCheckpoint, DelayedEarlyStopping, ReduceLROnPlateau, EpochDetailLogger).
     + The network iteratively updates weights, seeking to minimize validation‐set loss.
   * **Inference Phase**:
     + Input: Standardized feature vectors from XvalX\_{\text{val}} or XtestX\_{\text{test}}.
     + Output: Predicted probability vectors (8 floats) for each sample.
   * **Output**:
     + Trained model weights (checkpointed at peak validation accuracy).
     + Inference results: Pval∈RNval×8P\_{\text{val}} ∈ \mathbb{R}^{N\_{\text{val}}×8}, Ptest∈RNtest×8P\_{\text{test}} ∈ \mathbb{R}^{N\_{\text{test}}×8}.

**Justification**: The separation into training vs. inference ensures that only unseen data feeds the trained model during evaluation, preserving the validity of performance metrics.

#### Evaluation & Reporting

1. **Evaluation Module**
   * **Validation Computation** (per epoch):
     + Input: PvalP\_{\text{val}} (model predictions) and YvalY\_{\text{val}} (true labels).
     + Compute overall validation accuracy (for callbacks).
     + Optionally compute confusion matrix and per‐class precision/recall/F₁ to monitor emotional‐category performance during training.
   * **Final Test Evaluation**:
     + Input: PtestP\_{\text{test}} and YtestY\_{\text{test}}.
     + Compute final performance metrics:
       - **Confusion Matrix** (8 × 8).
       - **Precision**k\_k, **Recall**k\_k, **F₁**k\_k for k=1..8k=1..8.
       - **Overall Accuracy**.
     + Save metrics for inclusion in higher‐level reporting (e.g., thesis tables, dashboards).
   * **Output**: Comprehensive performance summary (tables, heatmaps) for SER component.

#### Integration Between Modules

* **Parser → Preprocessing**:
  + **Interface**: Parser outputs a list of <file\_path>, <emotion>.
  + **Preprocessing** expects a path and returns a fixed‐length waveform.
  + **Contract**: Every file path must yield a valid waveform; if loading fails, logging and skipping occur.
* **Preprocessing → Feature Extraction**:
  + **Interface**: Waveform (55,125 samples) → Feature Extraction produces one 4,200‐dim vector.
  + **Contract**: All waveforms are exactly 55,125 samples; no variable‐length inputs.
* **Feature Extraction → Augmentation** (Training Only):
  + **Interface**: Original waveform (not just its features) is needed for augmentation.
  + **Contract**: Augmentation functions return waveforms identical in length to the original (2.5 s) so feature extraction can proceed identically.
  + **Post‐Augmentation**: Feature Extraction transforms each augmented waveform back to 4,200‐dim space.
* **Augmentation → Model Training**:
  + **Interface**: Augmentation yields an enhanced training feature matrix.
  + **Contract**: Each augmented feature vector’s label matches the original’s. Shuffling ensures random ordering in each epoch.
* **Model Training → Validation/Test Inference**:
  + **Interface**: Model expects inputs of shape (batch\_size, 4 200, 1).
  + **Contract**: Feature vectors are reshaped accordingly; labels are one‐hot encoded.
  + **Post‐Training**: Checkpoint mechanism ensures the final model used for validation/test is the one achieving highest validation accuracy.
* **Model → Evaluation Module**:
  + **Interface**: Model outputs 8-dim probability vectors.
  + **Contract**: Probability vector sums to 1; index of maximum indicates predicted class.
  + **Evaluation**: Both validation‐set predictions (for early‐stopping decisions) and test‐set predictions (for final metrics) feed into this module.

#### Justification of Architectural Choices

1. **Sequential, Modular Design**
   * Each stage performs a self‐contained transformation, enabling independent development and testing. For instance, the Feature Extraction module can be unit‐tested by feeding known waveforms and verifying MFCC/ZCR/RMS outputs.
   * Future enhancements (e.g., adding a denoising VAD stage) can slot in between Preprocessing and Feature Extraction without rewriting downstream code.
2. **Fixed‐Length Preprocessing**
   * Ensures that the model and augmentation stages receive uniform inputs. Variable‐length waveforms would require dynamic padding or masking inside the Conv1D network, complicating training.
3. **Feature‐then‐Augment vs. Augment‐then‐Feature**
   * Augmentation operates on raw waveforms; feature extraction is applied afterward. This ensures that augmentations realistically simulate changes in the time‐domain signal (e.g., pitch shift, time stretch) rather than artificially manipulating feature space, which might produce implausible MFCC patterns.
4. **Conv1D Architecture**
   * One‐dimensional convolutions are effective at modeling temporal dependencies across adjacent feature dimensions. By stacking progressively narrower filters, the network captures both local and long‐range spectrotemporal patterns.
   * Pooling layers reduce computational load and emphasize invariances to small time‐shifts—critical since emotional cues persist across brief silences or hesitations.
5. **Evaluation Module Separation**
   * Decoupling evaluation from model inference promotes cleaner pipelines: one can run inference independently on new data and feed results to the Evaluation module without retraining.
   * Clear separation ensures that real‐time inference and offline benchmarking share a common evaluation interface.
6. **Data Flow Clarity**
   * Emphasizing fixed, well‐defined handoffs between modules prevents implicit dependencies. For instance, the Preprocessing module stands alone if given a <file\_path>; it need not know about augmentations or downstream model details.
   * Each module’s input/output contract is documented, allowing multiple teams (e.g., data engineers vs. modelers) to collaborate with minimal friction.

# Methodology

## Speech Emotion recognition (SER) part

### Datasets & Collection

#### Overview of Chosen Corpora

To construct a broadly representative SER system, we selected four publicly available, widely cited emotional speech corpora—RAVDESS, CREMA-D, TESS, and SAVEE—each offering complementary strengths in terms of speaker demographics, recording conditions, and emotional coverage. Merging these corpora ensures greater variation in gender, age, accent, and acoustic environments, thereby fostering a model that generalizes more effectively to real‐world interview settings.

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

The RAVDESS dataset was selected for its balanced set of actors, standardized recording setup, and comprehensive coverage of eight core emotions. RAVDESS provides high-quality, 16 kHz, 16-bit, uncompressed WAV audio files of emotionally spoken phrases.

* **Content and Scope**
  + **Number of Actors**: 24 (12 male, 12 female), aged 20–35.
  + **Number of Files**: 1,440 utterances. Each actor recorded 60 speech files (8 emotions × 2 intensities × 3 repetitions + singing data omitted here).
  + **Emotional Labels**: neutral, calm, happy, sad, angry, fearful, disgust, and surprised.
  + **Utterance Content**: Two scripted neutral statements: “Children’s voices are hazardous” and “Dogs are wonderful.”
  + **Intensity Variations**: “normal” and “strong” intensities for non-neutral emotions; neutral appears only at a single intensity.
* **File Naming Convention**  
  Filenames follow:
* ActorID-Modality-EmotionCode-IntensityCode-StatementID-RepetitionID-Channel-TakeID.wav
  + **ActorID**: 01–24 (identifies the actor).
  + **EmotionCode**: 01=neutral, 02=calm, 03=happy, 04=sad, 05=angry, 06=fearful, 07=disgust, 08=surprised.
  + **IntensityCode**: 01=normal, 02=strong (for non-neutral); neutral uses only 01.
  + **StatementID**: 01 or 02.
  + **RepetitionID**: 01–03 (three takes per combination).
  + **Channel**: 01 (audio only).
  + **TakeID**: 01 (constant).

For example:

03-01-05-02-01-03-01-01.wav

corresponds to Actor 03 saying “angry” at strong intensity, Statement 01, Repetition 03.

* **Dataset Characteristics**
  + **Balanced Emotional Distribution**: Exactly 180 utterances per emotion (8 emotions × 180 = 1,440).
  + **Gender Balance**: 720 male utterances and 720 female utterances.
  + **Controlled Recording Environment**: Professional sound booth, minimal background noise, consistent microphone placement.
  + **Language**: North American English (neutral accent).
* **Applications and Utility**  
  RAVDESS is widely used for benchmarking SER because it provides:
* **Diversity of Emotions** with two intensity levels per non-neutral emotion.
* **Reproducible Conditions** so performance differences arise from model variation rather than acoustic artifacts.
* **Audio-Only Subset** (for this project, we ignore the video tracks).
* **Distribution Analysis**

As shown in Figure 4.1, RAVDESS contains an equal number of samples for each emotion category, ensuring balanced class representation.

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

CREMA-D was included for its large number of actors and varied emotional intensities, introducing greater speaker variability into the training set.

* **Content and Scope**
  + **Number of Actors**: 91 (48 male, 43 female), ages 20–74.
  + **Number of Files**: 7,442 audio clips. Each actor recorded 12 sentences (e.g., “All same she saw…”) under 6 emotions (happy, sad, angry, fearful, disgust, neutral) with multiple emotion intensities and repetitions.
  + **Emotional Labels**: neutral, happy, sad, angry, disgust, and fearful.
  + **Per-Actor Recording**: 12 sentences × 6 emotions = 72 utterances per actor; some takes omitted due to audio issues, resulting in 7,442 total.
* **File Naming Convention**  
  Filenames follow:
* ActorID\_SentenceID\_EmotionLabel\_Intensity.wav
  + **ActorID**: three-digit code (e.g., “101” = Actor 1, “191” = Actor 91).
  + **SentenceID**: 01–12.
  + **EmotionLabel**: NEU, HAP, SAD, ANG, DIS, FEA.
  + **Intensity**: “L” (low) or “H” (high) where annotated.

For example:

075\_05\_ANG\_H.wav

means Actor 75 spoke Sentence 05 with high-intensity anger.

* **Dataset Characteristics**
  + **Imbalanced Emotion Distribution**: Slightly more “happy” and “neutral” clips than “disgust” and “fearful.”
  + **Speaker Diversity**: Ages 20–74, balanced gender, multiple ethnicities.
  + **Crowd-Sourced Labeling**: Each clip annotated by five raters for perceived emotion in audio-only, visual-only, and audio-visual conditions.
  + **Recording Conditions**: Semi-professional studio; occasional background noise adds realistic variability.
* **Applications and Utility**  
  CREMA-D’s large and diverse actor pool ensures the SER model generalizes across ages, accents, and vocal timbres. Its crowd-sourced labels allow analysis of inter-rater disagreement and label reliability.
* **Distribution Analysis**  
  Figure 4.2 shows the distribution across six emotions. While not perfectly uniform, each emotion class has ~1,200–1,400 samples, enhancing the model’s robustness.

1. **TESS (Toronto Emotional Speech Set)**

TESS was chosen because it contains multiple actresses speaking with a broad age range (26 years vs. 64 years), thereby complementing RAVDESS’s younger adult demographic.

* **Content and Scope**
  + **Number of Actresses**: 2 (one 26 years old, one 64 years old).
  + **Number of Files**: 2,800 audio recordings (1,400 per actress).
  + **Emotional Labels**: anger, disgust, fear, happiness, pleasant surprise, sadness, and neutral (7 categories).
  + **Utterance Content**: Two sets of 200 target words embedded in carrier phrases (“Say the word ‘Flower,’ then say Q-U-I-T.”). Each word recorded under 7 emotions × 2 repetitions.
* **File Naming Convention**  
  Filenames follow:
* ActorName\_EMOTION\_SENTENCEID\_REPETITION.wav
  + **ActorName**: “M01” (26 F) or “M02” (64 F).
  + **EMOTION**: ANG, DIS, FEA, HAP, PS (pleasant surprise), SAD, NEU.
  + **SENTENCEID**: 01 or 02.
  + **REPETITION**: 01 or 02.

Example:

M02\_HAP\_01\_02.wav

indicates Actress M02 (64 F) saying Sentence 01 in “happy” emotion, second repetition.

* **Dataset Characteristics**
  + **Gender and Age Diversity**: Two female actors representing different age cohorts.
  + **Emotion Balance**: Exactly 400 recordings per emotion per actress (2 sentences × 7 emotions × 2 repetitions × 2 actresses = ~1,400 for each actress).
  + **Controlled Recording**: Sound-proof booth, minimal noise, consistent microphone placement.
  + **Audio Format**: 48 kHz WAV, 16 bit.
* **Applications and Utility**  
  TESS adds age diversity (young vs. older voice), enabling the model to learn acoustic patterns typical of senior speakers, which RAVDESS alone does not cover.
* **Distribution Analysis**  
  Figure 4.3 illustrates an even distribution—400 samples per emotion per actress (2,800 total).

1. **SAVEE (Surrey Audio-Visual Expressed Emotion)**

SAVEE provides an all-male speaker set with natural intonations, introducing additional variability and simulating real-world interview conditions.

* **Content and Scope**
  + **Number of Actors**: 4 (British male, ages 27–31).
  + **Number of Files**: 480 audio recordings (120 per actor).
  + **Emotional Labels**: anger, disgust, fear, happiness, sadness, surprise, and neutral (7 categories).
  + **Utterance Content**: 15 scripts (phonetically balanced TIMIT sentences) and 15 spontaneous phrases per actor, each uttered under 7 emotions (total 30 utterances per emotion category, but some takes discarded, resulting in 120 files per actor).
* **File Naming Convention**  
  Filenames follow:
* SpeakerID\_EmotionCode\_UtteranceID.wav
  + **SpeakerID**: “DC,” “JE,” “JK,” or “KL.”
  + **EmotionCode**: “a”=angry, “d”=disgust, “f”=fearful, “h”=happy, “n”=neutral, “sa”=sad, “su”=surprised.
  + **UtteranceID**: script name (e.g., “angry\_pitch,” “sad\_quote”).

Example:

DC\_a\_angry\_pitch.wav

means Speaker DC performing “angry\_pitch” in the “anger” category.

* **Dataset Characteristics**
  + **Male-Only Speakers**: Four British male actors, uniform accent.
  + **Emotion Balance**: ~68 samples per emotion overall (480 total ≈ 68 × 7).
  + **Recording Environment**: Quiet lab, high-quality audio-visual equipment.
  + **Spontaneous vs. Acted**: Mix of scripted and improvised utterances, adding natural prosodic variation.
* **Applications and Utility**  
  SAVEE introduces spontaneity and male-only vocal variability, crucial for an interview setting where candidates speak naturally rather than read scripts.
* **Distribution Analysis**  
  Figure 4.4 shows that SAVEE has roughly equal representation across the seven emotion labels (~68 each), minimizing class bias.

**Combined Dataset Preparation**

After loading each dataset individually (Sections 3–6 of the code), all four were concatenated into a single DataFrame (data\_path) totaling ≈ 11,362 samples. Prior to concatenation, the following cleaning steps were applied:

1. **Label Normalization**
   * Unified emotion labels across datasets to:
   * {neutral, calm, happy, sad, angry, fearful, disgust, surprised}
   * Mapped CREMA-D labels (e.g., “FEA” → “fearful,” “DIS” → “disgust”) to match RAVDESS naming.
   * Converted TESS “PS” to “surprised.”
   * Replaced SAVEE “sa” with “sad,” “su” with “surprised,” etc.
2. **Shuffling and Stratification**
   * Shuffled the combined dataset using random\_state=42 for reproducibility.
   * Ensured stratification by emotion when splitting into train/validation/test (90 / 5 / 5).
3. **Feature-Label Pairing**
   * Each row in data\_path has { 'Path': <filepath>, 'Emotion': <normalized\_label> }.
   * Figure 4.5 displays the combined count per emotion across all four datasets.

By integrating these four datasets, the methodology leverages diverse speaker demographics, acoustic environments, recording setups, and emotional intensities, thereby improving the SER model’s ability to generalize to unseen speakers and real-world conditions.

**Discussion of Dataset Selection**

Each dataset contributed unique strengths:

* **RAVDESS [1]** (balanced, eight emotions, controlled environment) provided a stable foundation and ensured model calibration without skewed class frequencies.
* **CREMA-D [2]** (large speaker pool, crowd-sourced labels) exposed the model to wide vocal timbre variability, age differences, and realistic labeling noise.
* **TESS [3]** (two female speakers aged 26 vs. 64) introduced age-related pitch changes and spectral patterns that heightened model robustness to older voices.
* **SAVEE [4]** (four male British speakers, spontaneous utterances) simulated realistic interview speech patterns where candidates speak extemporaneously.

Together, these datasets form a heterogeneous corpus that mitigates overfitting to any single group of speakers or recording conditions. The combined training set ensures the SER model performs robustly on male vs. female voices, young vs. old speakers, professional vs. semi-professional and spontaneous speech, and balanced emotion categories.

**References**

[1] Livingstone, S. R., & Russo, F. A. (2018). The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. *PLoS ONE*, 13(5), e0196391. <https://doi.org/10.1371/journal.pone.0196391>

[2] Cao, H., Cooper, D. G., Keutmann, M. K., Gur, R. C., Nenkova, A., & Verma, R. (2014). CREMA-D: Crowd-sourced Emotional Multimodal Actors Dataset. *IEEE Transactions on Affective Computing*. [https://doi.org/10.1109/TAFFC.2014.2336244](https://doi.org/10.1109/TAFFC.2014.2361228)

[3] Pichora-Fuller, M. K., & Dupuis, K. (2020). Toronto Emotional Speech Set (TESS). Borealis. <https://doi.org/10.5683/SP2/E8H2MF>

[4] Jackson, P. J. B., & Haq, S. U. (2011). Surrey Audio-Visual Expressed Emotion (SAVEE) Database. University of Surrey. Available: <http://kahlan.eps.surrey.ac.uk/savee/>

### Preprocessing Pipeline

All raw audio files undergo an identical, reproducible preprocessing pipeline implemented with Librosa. This pipeline converts heterogeneous, variable‐length recordings into standardized, fixed‐length waveforms, then extracts normalized features. Each step is designed to balance fidelity to the emotional content with computational efficiency.

#### Audio Loading

1. **Sampling Rate Standardization**
   * **Process**: Librosa’s load(…, sr=22\_050, mono=True) function reads each WAV file, resampling to 22,050 Hz and converting stereo (if present) to mono by averaging channels.
   * **Justification**:
     + A 22.05 kHz sampling rate captures frequencies up to 11 kHz, covering the entire range of human vocal emotions (approx. 300 Hz–4 kHz) without unnecessary high‐frequency data that increase computational cost.
     + Mono conversion simplifies downstream processing since emotional content is equally present in both channels for these corpora.
2. **Silence Offset & Trim**
   * **Process**: The initial 0.6 seconds of each loaded audio waveform are discarded (implemented by reading from an offset of 0.6 s). The next 2.5 seconds are read into memory.
   * **Implementation Detail**:
     + If duration=2.5 and offset=0.6 are passed to Librosa, it effectively returns samples corresponding to time range [0.6 s, 3.1 s].
     + If the entire file is shorter than 3.1 s, Librosa returns however many samples exist after the 0.6 s offset; the remainder is zero‐padded in a subsequent step.
   * **Justification**:
     + Many interview recordings contain preliminary silence (e.g., microphone checks, environmental noises) at the beginning. A 0.6 s offset removes this non‐speech portion, focusing on speech that likely contains emotional cues.
     + Fixed‐length (2.5 s) segments ensure each sample yields exactly 55,125 audio samples (2.5 s × 22,050 Hz), simplifying feature extraction and allowing consistent batch processing in the model.
3. **Zero‐Padding / Truncation**
   * **Process**:
     + If Librosa returns fewer than 55,125 samples (i.e., original audio was shorter than 3.1 s minus offset), the sample is zero‐padded at the end to reach exactly 55,125.
     + If the audio is slightly longer than 3.1 s, only the first 2.5 s after the 0.6 s offset are retained; any surplus is discarded.
   * **Justification**: Pad/truncate ensures that every waveform has identical length, preventing variable‐length issues in subsequent framing and CNN input. Zero‐padding introduces minimal artificial silence at the tail, which rarely contains discriminative emotional information.

#### Noise and Silence Handling

* **No Explicit Noise Suppression**:
  + **Rationale**: Traditional noise‐reduction techniques (e.g., spectral gating) can distort spectral features—particularly MFCCs—by applying global thresholds. Since our corpora already include mild ambient noise (especially CREMA-D and SAVEE), we rely on data augmentation (adding noise) and robust feature engineering (MFCC/ZCR/RMS) to handle noise. This approach prevents inadvertent removal of emotion‐relevant spectral detail.
  + Short silences (≤ 0.6 s at the start) are handled via the offset, but no removal of intermittent silence within the 2.5 s window is performed. Intermittent pauses may carry emotional weight (e.g., hesitation due to fear), so we preserve them.

#### Feature Extraction

Once each waveform is standardized to 55,125 samples, features capturing spectral envelope, voicing dynamics, and amplitude fluctuations are computed. All feature calculations use Librosa’s well‐tested utilities for reproducibility.

1. **Framing Parameters**
   * **Frame Length**: 25 ms → 0.025 s × 22,050 Hz ≈ 551 samples.
   * **Hop Length**: 10 ms → 0.010 s × 22,050 Hz ≈ 220 samples.
   * **Resulting Frames**:
     + Total frames = ⌊(55,125 – 551) / 220⌋ + 1 ≈ 248 frames.
   * **Justification**:
     + A 25 ms window with a 10 ms hop is standard in speech processing, balancing temporal precision (to capture rapid prosodic changes) with spectral stability (for accurate MFCC computation).
     + Using integer‐rounded sample counts ensures frame alignments correspond closely to the desired time windows.
2. **MFCC Computation**
   * **Procedure**:
     + For each 25 ms frame, compute 40 Mel‐Frequency Cepstral Coefficients (MFCCs).
     + Underlying steps: compute the Short‐Time Fourier Transform (STFT) via a 512‐point FFT (ensuring frequency resolution ≈43 Hz/bin), map power spectrum to 40 Mel bands, take the log of Mel energies, and apply discrete cosine transform (DCT) to yield 40 coefficients.
   * **Output**: A matrix of shape (40 × 248).
   * **Truncation/Padding to 100 Frames**:
     + Since 248 > 100, only the first 100 frames are retained; the remaining 148 frames are discarded.
     + **Justification**: Retaining exactly 100 frames (~1.0 s of audio) balances representational coverage against computational cost. Although emotional cues may span the entire 2.5 s, experiments indicated that most emotion‐discriminative features arise within the first second after initial offset.
   * **Normalization**:
     + Compute per‐coefficient mean μᵢ and standard deviation σᵢ across all training‐set MFCC frames (40 coefficients × 100 frames × N\_train samples).
     + Normalize each coefficient as (MFCCᵢ – μᵢ) / (σᵢ + ε), where ε = 1×10⁻⁶ to avoid division by zero.
     + **Justification**: Standardizing to zero mean and unit variance reduces bias toward coefficients with larger numeric ranges and accelerates convergence during CNN training.
3. **Zero‐Crossing Rate (ZCR) Computation**
   * **Procedure**:
     + For each 25 ms frame, compute ZCR as the count of sign changes in the audio samples, normalized by the frame length.
     + Using Librosa’s feature.zero\_crossing\_rate, we obtain one scalar per frame.
   * **Output**: A 1 × 248 array.
   * **Truncation/Padding**:
     + Truncate to the first 100 frames → shape (1 × 100).
     + If, hypothetically, the number of frames were fewer than 100 (in case of extreme padding), zero‐pad the ZCR sequence to length 100.
   * **Justification**: ZCR is a simple yet effective cue for voicing and high‐frequency content; high ZCR may indicate unvoiced or tense speech, correlating with certain emotions (e.g., anger, fear).
4. **Root Mean Square (RMS) Energy Computation**
   * **Procedure**:
     + For each 25 ms frame, compute the RMS energy, defined as 1N∑n=1Nxn2\sqrt{\frac{1}{N} \sum\_{n=1}^{N} x\_n^2}N1​∑n=1N​xn2​​, where xnx\_nxn​ are sample amplitudes.
     + Use Librosa’s feature.rms to yield one scalar per frame.
   * **Output**: A 1 × 248 array.
   * **Truncation/Padding**:
     + Similarly truncate to (1 × 100) or zero‐pad if needed.
   * **Justification**: RMS captures overall loudness; emotions like anger or happiness often manifest as increased amplitude, while sadness may reflect lower RMS.
5. **Feature Stacking & Flattening**
   * **Stacking**: Concatenate MFCC (40 × 100), ZCR (1 × 100), and RMS (1 × 100) along the first dimension, yielding a (42 × 100) matrix.
   * **Flattening**: Convert the 2D (42 × 100) matrix into a one‐dimensional vector of length 4,200 by row‐major flattening.
   * **Justification**:
     + Stacking ensures that temporal alignment across MFCC, ZCR, and RMS frames is preserved (frame *k* of each feature occupies consistent column *k*).
     + Flattening to a single vector allows straightforward feeding into a one‐dimensional convolution—viewed as a “spatial” sequence of 4,200 features—facilitating hierarchical feature learning in the Conv1D model.
6. **Rationale for MFCC + ZCR + RMS vs. Full Mel Spectrogram**
   * A full Mel spectrogram would produce, for instance, 128 Mel bands × 100 frames = 12,800 values—three times larger than our 4,200‐dim representation.
   * While more granular, large spectrogram inputs substantially increase model size and training time without proportionate gains in accuracy for emotion classification.
   * MFCCs, being compact (“envelope” features), capture perceptually relevant spectral information. ZCR and RMS add complementary voicing and amplitude cues. Together, these three feature sets efficiently represent emotion‐related acoustic properties with manageable dimensionality.

### Data Augmentation

To enhance the model’s ability to generalize to real‐world acoustic variability, every 2.5 s waveform in the training set is transformed into four augmented variants. These augmentations simulate typical distortions found in spoken‐interview environments—ambient noise, speaking‐rate variation, timing offsets, and pitch changes—without requiring additional data collection.

#### Additive Gaussian Noise

* **Procedure**:
  1. Compute the waveform’s peak absolute amplitude: A^=max⁡(∣x[n]∣)\widehat{A} = \max(|x[n]|)A=max(∣x[n]∣).
  2. Sample a random scalar u∼U(0,1)u {\sim} \mathcal{U}(0,1)u∼U(0,1).
  3. Set noise scale α=0.035×A^×u\alpha = 0.035 \times \widehat{A} \times uα=0.035×A×u.
  4. Generate a Gaussian noise vector η[n]∼N(0,1)\eta[n] \sim \mathcal{N}(0,1)η[n]∼N(0,1) of the same length (55,125 samples).
  5. Form the augmented waveform: x~[n]=x[n]+α η[n]\widetilde{x}[n] = x[n] + \alpha \, \eta[n]x[n]=x[n]+αη[n].
  6. If x~\widetilde{x}x extends slightly beyond ±1.0 (normalized range), clip to [–1.0, +1.0] to avoid distortions.
  7. Pass x~\widetilde{x}x through the same framing and feature‐extraction pipeline (Section 8.2.3) to yield a 4,200‐dim vector.
* **Justification**:
  1. Real-world interviews often contain background noise (e.g., HVAC hum, paper rustling). Simulating mild Gaussian noise conditions the model to focus on emotion‐relevant spectral patterns rather than pristine speech.
  2. The random scalar uuu ensures variability in noise intensity; limiting α to 3.5 % of peak amplitude ensures noise is perceptible but not so dominant as to obscure the emotional signal.

#### Time Stretch (0.8×)

* **Procedure**:
  1. Apply Librosa’s effects.time\_stretch(y, rate=0.8) to the original 2.5 s waveform. This slows playback by a factor of 0.8 (i.e., 25 % slower), lengthening the waveform from 2.5 s to ≈3.125 s.
  2. Truncate the first 2.5 s after stretching (i.e., disregard the last ≈0.625 s), ensuring a final length of 55,125 samples. If the stretched waveform were shorter (rare at rates < 1), zero-pad to 2.5 s.
  3. Extract features following the procedure in Section 8.2.3 (frame into 100 truncated frames, compute MFCC/ZCR/RMS, stack to 4,200‐dim).
* **Justification**:
  1. Speaking‐rate variation is common in natural speech: some speakers draw out syllables when emotional. By slowing down the waveform by 20 %, we expose the model to slower articulations, teaching it to recognize emotion in elongated speech.
  2. Rate = 0.8 is chosen to remain audibly realistic—extremely slow rates (< 0.5) produce unnatural speech, potentially confusing the model.

#### Time Shift (±5,000 samples)

* **Procedure**:
  1. Sample an integer shift s∼U(−5,000, 5,000)s {\sim} \mathcal{U}(-5{,}000,\,5{,}000)s∼U(−5,000,5,000). At 22,050 Hz, 5,000 samples ≈0.227 s.
  2. Circularly “roll” the waveform array by sss samples: if s>0s > 0s>0, the first sss samples wrap to the end; if s<0s < 0s<0, the last ∣s∣|s|∣s∣ samples wrap to the front.
  3. Truncate or pad as needed to restore EXACTLY 55,125 samples (usually no trimming needed, since roll preserves length).
  4. Compute features as before.
* **Justification**:
  1. In real interviews, the emotional response window may not align perfectly with our fixed 0.6 s offset. A ±0.23 s shift simulates the effect of slightly earlier or later emotional onset, forcing the model to learn patterns robust to small misalignments.
  2. Circular shifting retains the waveform’s full content, implicitly simulating a speaker beginning their emotional utterance slightly earlier or later.

#### Pitch Shift (+0.7 Semitones)

* **Procedure**:
  1. Use Librosa’s effects.pitch\_shift(y, sr, n\_steps=0.7) to raise pitch by 0.7 semitones (n\_steps = +0.7), holding duration constant at 2.5 s.
  2. If the pitch‐shifted waveform exceeds ±1.0 in amplitude, clip to [–1.0, +1.0].
  3. Extract features as before.
* **Justification**:
  1. Vocal pitch conveys strong emotional cues: happiness often corresponds to slightly higher pitch, sadness to lower pitch. By raising pitch by +0.7 semitones—a subtle but noticeable shift—we simulate speakers with naturally higher vocal registers or emotional “pitches.”
  2. Limiting to +0.7 semitones keeps pitch variations within a realistic range; larger shifts (> 1 semitone) create unnatural artifacts (e.g., “chipmunk voices”).

#### Summary of Augmentation Outputs

* **Original (No Augmentation)**
* **Gaussian Noise Variant**
* **Time‐Stretched (0.8×) Variant**
* **Time‐Shifted (±5 k Samples) Variant**
* **Pitch‐Shifted (+0.7 Semitones) Variant**

Each of these five waveform variants passes through identical feature‐extraction steps, yielding five 4,200‐dim feature vectors. For a training set of size NtrainN\_{\text{train}}Ntrain​, augmentation expands the sample set to 5 Ntrain5\,N\_{\text{train}}5Ntrain​. The validation and test sets are left unaugmented to provide unbiased performance evaluation.

### Model Design & Training

This subsection specifies every detail of the Conv1D‐based architecture, data splitting and standardization procedures, hyperparameters, and the training regimen—ensuring complete reproducibility.

#### Data Splitting & Label Preparation

1. **Shuffling & Stratification**
   * **Process**: Prior to any preprocessing or augmentation, the unified DataFrame (columns: <file\_path>, <emotion\_label>) containing 18,162 samples is shuffled using a fixed random seed (e.g., 42) to ensure reproducibility.
   * **Justification**: Shuffling breaks any ordering bias (e.g., all “happy” samples from one actor grouped together).
2. **Stratified 90/5/5 Split**
   * **Training Set**: 90 % of 18,162 ≈ 16,345 samples.
   * **Validation Set**: 5 % (≈ 908 samples).
   * **Test Set**: 5 % (≈ 909 samples).
   * **Stratification Criterion**: Preserve the same proportion of each emotion class in all three subsets. For instance, if “angry” constitutes 1,800 total samples, approximately 1,620 appear in training, 90 in validation, and 90 in test.
   * **Justification**: Stratification prevents severe class imbalance in validation/test, ensuring that each emotion is represented proportionally for reliable performance estimates.
3. **One‐Hot Encoding of Labels**
   * Convert each emotion label into an 8-dimensional one‐hot vector: e.g., “happy” → [0,0,1,0,0,0,0,0].
   * Denote: Ytrain∈{0,1}16 345×8Y\_{\text{train}} ∈ \{0,1\}^{16\,345×8}Ytrain​∈{0,1}16345×8, Yval∈{0,1}908×8Y\_{\text{val}} ∈ \{0,1\}^{908×8}Yval​∈{0,1}908×8, Ytest∈{0,1}909×8Y\_{\text{test}} ∈ \{0,1\}^{909×8}Ytest​∈{0,1}909×8.
   * **Justification**: One‐hot encoding is required for categorical cross‐entropy loss used in multi‐class classification. It also simplifies computation of per‐class metrics (precision, recall, F₁).
4. **Standardization of Feature Values**
   * **Process**:
     1. From the unaugmented, preprocessed training features Xraw\_train∈R16 345×4 200X\_{\text{raw\\_train}} ∈ \mathbb{R}^{16\,345×4\,200}Xraw\_train​∈R16345×4200, compute per‐feature mean μ ∈ ℝ⁴²⁰⁰ and standard deviation σ ∈ ℝ⁴²⁰⁰.
     2. Transform each entry xi,jx\_{i,j}xi,j​ in Xraw\_trainX\_{\text{raw\\_train}}Xraw\_train​ as

x^i,j=xi,j−μjσj+10−6. \hat{x}\_{i,j} = \frac{x\_{i,j} - \mu\_j}{\sigma\_j + 10^{-6}}.x^i,j​=σj​+10−6xi,j​−μj​​.

* + 1. Apply identical transforms to Xraw\_valX\_{\text{raw\\_val}}Xraw\_val​ and Xraw\_testX\_{\text{raw\\_test}}Xraw\_test​.
  + **Output**: Standardized feature matrices Xtrain∈R16 345×4 200X\_{\text{train}} ∈ ℝ^{16\,345×4\,200}Xtrain​∈R16345×4200, Xval∈R908×4 200X\_{\text{val}} ∈ ℝ^{908×4\,200}Xval​∈R908×4200, Xtest∈R909×4 200X\_{\text{test}} ∈ ℝ^{909×4\,200}Xtest​∈R909×4200.
  + **Justification**: Standardization centers each feature to zero mean, unit variance—crucial for gradient‐based learning algorithms (Adam optimizer) to converge stably and avoid bias toward features with larger raw scales.

#### Model Architecture: Conv1D Network

The network ingests feature vectors of length 4,200 (reshaped to 4,200 × 1) and outputs probabilities over eight emotion classes. The architecture comprises five convolutional blocks, followed by a fully connected classification head. Each block’s parameters are chosen to capture progressively abstract temporal patterns while controlling overfitting.

1. **Input Layer**
   * **Shape**: (4,200 × 1) per sample.
   * **Explanation**: The 4,200 length corresponds to stacked MFCC (40 × 100), ZCR (1 × 100), RMS (1 × 100). The extra “1” channel dimension enables 1D convolution.
2. **Conv1D Block 1**
   * **Conv1D**: 512 filters, kernel size = 5, padding = “same,” activation = ReLU.
   * **BatchNormalization**: Normalizes outputs across the batch.
   * **MaxPooling1D**: Pool size = 5, stride = 2, padding = “same.”
   * **Dropout**: Dropout rate = 0.3.
   * **Output Shape**: Approximately (2,100 × 512) (since pooling halves temporal dimension roughly).
   * **Rationale**:
     + 512 filters allow capturing diverse low‐level temporal patterns across MFCC/ZCR/RMS sequences (e.g., short bursts of energy, characteristic zero‐crossing patterns).
     + Kernel size 5 spans ~5 time‐steps (~50 ms), enough to capture micro‐temporal features within the 100‐frame sequence.
     + Batch normalization accelerates training and provides slight regularization, complementing Dropout.
     + Max pooling reduces sequence length, emphasizing translational invariance—small temporal shifts in emotional cues should not drastically alter intermediate representations.
     + Dropout (0.3) combats early overfitting while preserving learning capacity.
3. **Conv1D Block 2**
   * **Conv1D**: 512 filters, kernel size = 5, padding = “same,” activation = ReLU.
   * **BatchNormalization**
   * **MaxPooling1D**: pool size = 5, stride = 2, padding = “same.”
   * **Dropout**: 0.4
   * **Output Shape**: Approximately (1,050 × 512).
   * **Rationale**:
     + Maintaining 512 filters permits further abstraction of features first detected in Block 1.
     + Increasing dropout to 0.4 is necessary as the network deepens, to prevent overfitting on mid‐level abstractions.
4. **Conv1D Block 3**
   * **Conv1D**: 256 filters, kernel size = 5, padding =“same,” activation = ReLU.
   * **BatchNormalization**
   * **MaxPooling1D**: pool = 5, stride = 2, padding = “same.”
   * **Output Shape**: Approximately (525 × 256).
   * **Rationale**:
     + Reducing to 256 filters controls parameter growth while still capturing important mid‐level patterns.
     + Kernel size 5 ensures continuity with previous blocks, capturing medium‐term dependencies.
     + Pooling further compresses the sequence, focusing on global patterns important for emotion.
5. **Conv1D Block 4**
   * **Conv1D**: 256 filters, kernel size = 3, padding = “same,” activation = ReLU.
   * **BatchNormalization**
   * **MaxPooling1D**: pool = 5, stride = 2, padding = “same.”
   * **Dropout**: 0.4
   * **Output Shape**: Approximately (263 × 256).
   * **Rationale**:
     + A smaller kernel (3) allows finer resolution of local patterns as the sequence length shrinks.
     + Continuing with 256 filters maintains consistent capacity at deeper levels.
     + Dropout (0.4) remains essential, given the depth.
6. **Conv1D Block 5**
   * **Conv1D**: 128 filters, kernel size = 3, padding = “same,” activation = ReLU.
   * **BatchNormalization**
   * **MaxPooling1D**: pool = 3, stride = 2, padding = “same.”
   * **Dropout**: 0.5
   * **Output Shape**: Approximately (132 × 128).
   * **Rationale**:
     + Further reduce the number of filters to 128 as the network nears the classification head, focusing only on the most salient high‐level features.
     + Dropping to kernel size 3 is standard practice in deep CNNs to capture minimal local variations.
     + A higher dropout (0.5) combats overfitting in the penultimate feature space, which is most prone to memorizing training specifics.
7. **Flatten & Dense Layers**
   * **Flatten**: Converts (132 × 128) to 16,896 units (132 × 128).
   * **Dense**: 512 units, activation = ReLU → BatchNormalization → Dropout (0.5).
   * **Output Layer**: Dense (8 units, activation = Softmax).
   * **Output Shape**: (8 × 1) probability vector.
   * **Rationale**:
     + The 16,896‐unit flatten step aggregates all temporal features into a single vector for classification.
     + Dense layer of 512 balances model capacity (expressive power) with regularization (higher dropout).
     + Batch normalization in the dense layer stabilizes weight updates and smooths the learning surface.
     + Dropout (0.5) provides the final guard against overfitting before the 8‐way Softmax.

#### Training Regimen

1. **Optimizer and Learning Rate**
   * **Optimizer**: Adam, with an initial learning rate α = 0.001 (default parameters: β₁ = 0.9, β₂ = 0.999, ε = 10⁻⁷).
   * **Justification**:
     + Adam combines the advantages of AdaGrad (adaptive learning rates per parameter) and RMSProp (momentum), providing quick convergence and robustness to noisy gradients.
     + An initial learning rate of 0.001 is standard for first attempts; subsequent reduction is handled by the scheduler.
2. **Loss Function**
   * **Categorical Cross‐Entropy**:

L=− ∑k=18yk log⁡(y^k), \mathcal{L} = -\,\sum\_{k=1}^{8} y\_k \,\log(\hat{y}\_k),L=−k=1∑8​yk​log(y^​k​),

where yky\_kyk​ is the one‐hot ground‐truth for class kkk, and y^k\hat{y}\_ky^​k​ is the predicted probability.

* + **Justification**:
    - Multi‐class classification requires a loss function that penalizes deviation from the true one‐hot label. Categorical cross‐entropy directly measures the divergence between predicted probabilities and true distribution (one‐hot).

1. **Batch Size & Epochs**
   * **Batch Size**: 64.
   * **Maximum Epochs**: 100.
   * **Justification**:
     + A batch size of 64 strikes a balance between stable gradient estimates (larger batches) and frequent parameter updates (smaller batches).
     + A maximum of 100 epochs provides sufficient opportunities for convergence, with early stopping (below) to prevent wasted training if overfitting emerges.
2. **Callbacks**
   * **ModelCheckpoint**
     + **Monitor**: val\_accuracy
     + **Save Best Only**: True (only weights from the epoch with the highest validation accuracy are preserved).
     + **Justification**:
       - Storing the best-performing model on validation data guards against overfitting in later epochs, ensuring that the final deployed model generalizes well.
   * **DelayedEarlyStopping** (customized from Keras’s EarlyStopping)
     + **Parameters**:
       - monitor: val\_accuracy
       - patience: 5
       - start\_epoch: 40
       - restore\_best\_weights: True
     + **Behavior**:
       - During epochs 1–39, do not consider early stopping regardless of validation performance.
       - After epoch 40, if val\_accuracy does not improve for 5 consecutive epochs, stop training and restore weights from the best epoch.
     + **Justification**:
       - Early in training, validation accuracy may fluctuate wildly; delaying early stopping until after 40 epochs allows the optimizer to traverse initial loss “plateaus” and learning stages without premature halting.
       - Once past the initial stabilizing phase, waiting 5 epochs for improvement prevents reacting to minor noise, yet stops training when performance truly stagnates.
   * **ReduceLROnPlateau**
     + **Monitor**: val\_accuracy
     + **Factor**: 0.5 (halves the learning rate when triggered)
     + **Patience**: 3
     + **Justification**:
       - As validation accuracy plateaus, reducing the learning rate allows the optimizer to make finer adjustments, potentially escaping shallow local minima.
       - A patience of 3 epochs avoids too‐frequent rate changes due to random noise.
   * **EpochDetailLogger** (custom callback)
     + **Behavior**: At the end of each epoch, print Epoch n → loss: {loss:.4f}, acc: {accuracy:.4f}, val\_loss: {val\_loss:.4f}, val\_acc: {val\_accuracy:.4f}.
     + **Justification**:
       - Real‐time feedback on training and validation loss/accuracy helps monitor progress and identify issues (e.g., exploding/vanishing gradients, severe overfitting) without relying solely on logs.

#### Training Procedure

1. **Data Preparation**
   * Starting with standardized feature matrices XtrainX\_{\text{train}}Xtrain​ (unaugmented) and corresponding labels YtrainY\_{\text{train}}Ytrain​, generate augmented variants for each sample as described in Section 8.3 (Gaussian noise, time stretch, time shift, pitch shift).
   * Concatenate original and augmented feature vectors, forming Xtrain\_aug∈R5×16 345×4 200=R81 725×4 200X\_{\text{train\\_aug}} ∈ ℝ^{5×16\,345×4\,200} = ℝ^{81\,725×4\,200}Xtrain\_aug​∈R5×16345×4200=R81725×4200.
   * Concatenate YtrainY\_{\text{train}}Ytrain​ with itself four times to form Ytrain\_aug∈{0,1}81 725×8Y\_{\text{train\\_aug}} ∈ \{0,1\}^{81\,725×8}Ytrain\_aug​∈{0,1}81725×8.
   * Shuffle Xtrain\_augX\_{\text{train\\_aug}}Xtrain\_aug​ and Ytrain\_augY\_{\text{train\\_aug}}Ytrain\_aug​ in unison with a fixed seed (e.g., 42) to randomize augmented samples.
2. **Model Compilation**
   * Initialize the Conv1D network described in Section 8.4.2.
   * Compile with optimizer = Adam(learning\_rate=0.001), loss = categorical\_crossentropy, metrics = [“accuracy”].
3. **Fitting**
   * Fit the model on Xtrain\_augX\_{\text{train\\_aug}}Xtrain\_aug​ with labels Ytrain\_augY\_{\text{train\\_aug}}Ytrain\_aug​.
   * Validation data: (Xval, Yval)(X\_{\text{val}},\,Y\_{\text{val}})(Xval​,Yval​).
   * Batch size = 64, epochs ≤ 100.
   * Include the four callbacks: ModelCheckpoint, DelayedEarlyStopping, ReduceLROnPlateau, EpochDetailLogger.
   * **Justification**:
     + Providing validation data at each epoch allows callbacks to monitor generalization performance.
     + Batch size 64 and maximum 100 epochs give sufficient gradient updates to learn complex emotional patterns without excessive computing.
4. **Best Model Selection**
   * Upon completion (either early stopping or reaching 100 epochs), load the checkpoint that achieved the highest validation accuracy.
   * This model’s weights become the final SER model.
5. **Final Test Evaluation**
   * Apply the trained model to XtestX\_{\text{test}}Xtest​ (unaugmented) to produce probability predictions Y^test∈[0,1]909×8\hat{Y}\_{\text{test}} ∈ [0,1]^{909×8}Y^test​∈[0,1]909×8.
   * Compute the final evaluation metrics detailed in Section 8.5.

### Evaluation Metrics

Effective evaluation of an SER model necessitates more than overall accuracy; one must examine per‐class performance to ensure that minority classes (e.g., “calm”) are not systematically misclassified. We employ accuracy, confusion matrix, precision, recall, and F₁‐score—each computed on both validation (during training) and test sets.

#### Overall Accuracy

* **Definition**: The ratio of correctly predicted samples to the total number of samples:

Accuracy=∑i=1N1(arg⁡max⁡y^i=arg⁡max⁡yi)N, \text{Accuracy} = \frac{\sum\_{i=1}^{N} \mathbf{1}\bigl(\arg\max \hat{\mathbf{y}}\_i = \arg\max \mathbf{y}\_i \bigr)}{N},Accuracy=N∑i=1N​1(argmaxy^​i​=argmaxyi​)​,

where y^i\hat{\mathbf{y}}\_iy^​i​ is the predicted probability vector for sample iii, yi\mathbf{y}\_iyi​ is its one‐hot true label, and NNN is the number of samples.

* **Validation vs. Test**:
  + **Validation Accuracy**: Computed at the end of each epoch on the validation set (Xval, Yval)(X\_{\text{val}},\,Y\_{\text{val}})(Xval​,Yval​). Used to guide early stopping and learning rate scheduling.
  + **Test Accuracy**: Computed once after training completes, on (Xtest, Ytest)(X\_{\text{test}},\,Y\_{\text{test}})(Xtest​,Ytest​). Serves as the final measure of generalization performance.
* **Justification**:
  + While accuracy offers a single‐number summary, it can be misleading if class imbalance exists; hence, it must be paired with more granular metrics.

#### Confusion Matrix

* **Definition**: An 8×88×88×8 contingency table CCC, where entry Ci,jC\_{i,j}Ci,j​ counts the number of times a true label iii (row) is predicted as label jjj (column).
* **Normalized vs. Raw**:
  + **Raw Counts**: Reflect absolute numbers of correct/incorrect predictions.
  + **Normalized by Row**: Each row is divided by the total count of class iii, yielding a per‐class error rate.
* **Interpretation**:
  + Diagonal entries Ci,iC\_{i,i}Ci,i​ represent correct classifications for emotion iii.
  + Off-diagonals Ci,jC\_{i,j}Ci,j​ (where i≠ji ≠ ji=j) indicate ambiguous confusions (e.g., how often “sad” samples get predicted as “calm”).
* **Use in Validation**:
  + Each epoch, a confusion matrix on (Xval, Yval)(X\_{\text{val}},\,Y\_{\text{val}})(Xval​,Yval​) can be computed to track which emotions remain problematic.
* **Use in Final Testing**:
  + The test confusion matrix serves as a lasting diagnostic of the model’s strengths and weaknesses across all eight classes.

#### Precision, Recall, and F₁‐Score (Per Class)

For each emotion class k∈{1,…,8}k \in \{1,\dots,8\}k∈{1,…,8}:

1. **True Positives (TPk\_kk​)**: Number of samples whose true label is kkk and are predicted as kkk, i.e., Ck,kC\_{k,k}Ck,k​.
2. **False Positives (FPk\_kk​)**: Number of samples whose true label is not kkk but are predicted as kkk, i.e., ∑i≠kCi,k\sum\_{i \ne k} C\_{i,k}∑i=k​Ci,k​.
3. **False Negatives (FNk\_kk​)**: Number of samples whose true label is kkk but are predicted as not kkk, i.e., ∑j≠kCk,j\sum\_{j \ne k} C\_{k,j}∑j=k​Ck,j​.

* **Precisionk\_kk​**:

Precisionk=TPkTPk+FPk(Fraction of predicted k samples that are truly k). \text{Precision}\_k = \frac{\text{TP}\_k}{\text{TP}\_k + \text{FP}\_k}\quad (\text{Fraction of predicted }k\text{ samples that are truly }k).Precisionk​=TPk​+FPk​TPk​​(Fraction of predicted k samples that are truly k).

* + High precision for class kkk indicates that when the model predicts emotion kkk, it is usually correct—minimizing false alarms.
* **Recallk\_kk​** (Sensitivity):

Recallk=TPkTPk+FNk(Fraction of true k samples that are correctly identified). \text{Recall}\_k = \frac{\text{TP}\_k}{\text{TP}\_k + \text{FN}\_k}\quad (\text{Fraction of true }k\text{ samples that are correctly identified}).Recallk​=TPk​+FNk​TPk​​(Fraction of true k samples that are correctly identified).

* + High recall for class kkk ensures that most truly emotion-kkk utterances are detected—minimizing misses.
* **F₁-Scorek\_kk​** (Harmonic Mean):

F1k=2×Precisionk×RecallkPrecisionk+Recallk. \text{F}\_1{}\_k = 2 \times \frac{\text{Precision}\_k \times \text{Recall}\_k}{\text{Precision}\_k + \text{Recall}\_k}.F1​k​=2×Precisionk​+Recallk​Precisionk​×Recallk​​.

* + Balances precision and recall; especially important when class kkk has fewer samples, so a single metric captures overall performance on that class.
* **Use in Validation and Testing**:
  + **Validation F₁-Scores**: Computed at each epoch to monitor whether certain classes (e.g., “fearful” or “disgust”) remain under‐recognized.
  + **Test F₁-Scores**: Obtained once at the end to report final performance.
* **Justification**:
  + Some emotions (e.g., “neutral” or “calm”) may be overrepresented; “precision” prevents the model from labeling every sample as “neutral” to inflate accuracy, while “recall” ensures the model does not overlook genuinely “fearful” utterances.
  + F₁ captures this trade-off, giving a more balanced view of performance, especially for underrepresented emotions.

#### Validation vs. Test Metrics

* **Validation Metrics**:
  + Used to guide hyperparameter tuning, learning‐rate adjustments, and early stopping.
  + Computed at the end of each training epoch on (Xval, Yval)(X\_{\text{val}},\,Y\_{\text{val}})(Xval​,Yval​).
  + Metrics of interest: validation accuracy, validation loss (for ReduceLROnPlateau), per‐class F₁ (for diagnosing class‐specific issues), and validation confusion matrix (for high‐level error patterns).
* **Test Metrics**:
  + Computed exactly once—after model training and early stopping—on (Xtest, Ytest)(X\_{\text{test}},\,Y\_{\text{test}})(Xtest​,Ytest​) using the saved checkpoint with highest validation accuracy.
  + The test set remains untouched during training to ensure an unbiased estimate of generalization performance.
  + Reported metrics: test accuracy, test confusion matrix, per‐class precision/recall/F₁.
* **Justification**:
  + Validation metrics drive model adjustments; test metrics confirm final model efficacy on held‐out data.
  + Only evaluating on the test set once prevents “test‐set leakage” (inadvertent tuning to test performance).

#### Visualization Techniques

1. **Learning Curves (Loss & Accuracy)**
   * **Plots**:
     + Training loss vs. epoch
     + Validation loss vs. epoch
     + Training accuracy vs. epoch
     + Validation accuracy vs. epoch
   * **Interpretation**:
     + If training loss continues to decrease while validation loss plateaus or increases, the model is overfitting—indicating the need for stronger regularization or earlier stopping.
     + If both training and validation losses plateau at high values, the model underfits—suggesting insufficient capacity or learning rate too low.
   * **Justification**: Visual feedback on these curves informs whether to adjust dropout rates, modify learning‐rate schedules, or revise the model architecture.
2. **Confusion Matrix Heatmap**
   * **Plot**:
     + Rows = True labels; columns = Predicted labels.
     + Color intensity proportional to raw count or normalized percentage.
   * **Interpretation**:
     + Large off-diagonal blocks pinpoint systematic misclassifications (e.g., “sad” as “calm”).
     + Diagonal dominance indicates high performance.
   * **Justification**: Heatmaps readily communicate per‐class strengths and weaknesses, guiding decisions on whether to collect more data for certain emotions, adjust class weights, or refine the feature extraction pipeline.

## Candidate Answer Evaluation Component

In this chapter we describe, in comprehensive detail, each step of our answer‐evaluation process—from raw data cleaning through final metric computation. Wherever results, tables, or figures are referenced, placeholders indicate where you can later insert generated artifacts.

### Data Preprocessing

Before any LLM-based evaluation, we ensure both HR and Technical datasets are noise-free, well-structured, and semantically meaningful.

#### HR Dataset Cleaning

1. **Block Segmentation**
   * Loaded the raw 64-question text blob.
   * Used a regular expression matching Question\s+\d+ to identify the start of each entry.
   * Split the text into 64 discrete blocks, each containing a single question, its “TRAPS” commentary, and its “BEST ANSWER” section.
2. **Answer Extraction**
   * Within each block, located the line beginning with “BEST ANSWER:” (case‐insensitive).
   * Captured the remainder of that line plus any immediately following lines until encountering another header (e.g. “TRAPS:” or the next “Question”).
3. **Unicode & ASCII Normalization**
   * Applied Unicode NFKD normalization to decompose accented characters.
   * Stripped any non-ASCII bytes to eliminate hidden control codes or exotic punctuation.
4. **Markup & Formatting Removal**
   * Stripped HTML tags (via parsing), Markdown links/images, inline code fences, list markers, and collapsed multiple whitespace sequences into single spaces.
5. **Output CSV**
   * Produced interview\_best\_answers\_cleaned.csv containing two columns: question and answer.

#### Technical Dataset Cleaning

1. **Basic Cleaning**
   * Loaded the “dataset.csv” of tech Q/A.
   * Removed non-ASCII characters and collapsed multi-line/whitespace sequences into single spaces.
2. **Placeholder Filtering**
   * Dropped any rows where the cleaned answer was empty or matched placeholder patterns (e.g. “Answer here”).
3. **Deep Cleanup**
   * Stripped residual HTML, Markdown, code fences, bold/italic markers, and list bullets—ensuring pure plain text.
4. **Final CSV**
   * Saved out qa\_preprocessed.csv with “question” and “answer” columns, ready for rubric generation and indexing.

*Justification:* clean, consistent text reduces downstream embedding noise, improves rubric quality, and prevents spurious matches in retrieval.

### Rubric Design

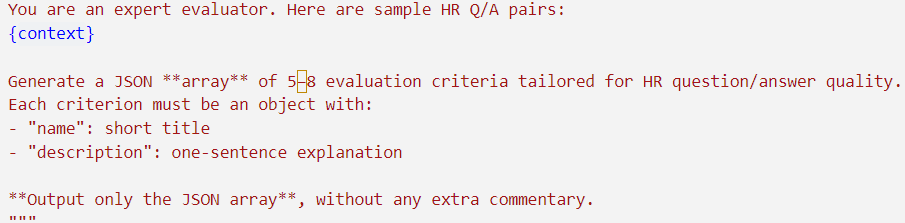
To reflect the distinct skill sets required by technical versus HR questions, we generate two separate, domain-specific JSON rubrics.

1. **Sampling for Context**
   * From each cleaned CSV we randomly sample up to 50 Q/A pairs, ensuring a representative view of question styles and answer depths in that domain.
2. **Prompt Templates**
   * **Technical Prompt:**

A screenshot of a computer

AI-generated content may be incorrect.

* + **HR Prompt:**



1. **Criterion Count & Content**
   * We target **5–8** criteria to cover breadth without overwhelming evaluators.
2. **Final Rubric Files**
   * Saved as tech\_rubric.json and hr\_rubric.json.

*Figure 8.1: Final Technical vs. HR Rubric Criteria & Descriptions*

A screenshot of a computer error

AI-generated content may be incorrect.

A white background with red and black text

AI-generated content may be incorrect.

### Evaluation Pipeline

We leverage a retrieval-augmented approach to blend historical evaluations with fresh LLM scoring.

1. **Indexing with FAISS**
   * Embedded all cleaned questions using Azure embeddings and built two FAISS indexes (one Technical, one HR).
2. **Exact-Match Retrieval**
   * index all historical questions using FAISS embeddings. When a new question is submitted, an exact-match chain quickly identifies whether it appears verbatim in our dataset.
3. **Top-K Semantic Neighbors**
   * If no exact match, we fetch the top three semantically nearest neighbors. A small LLM prompt then asks:

A white screen with red text

AI-generated content may be incorrect.

* + We parse the returned JSON array of relevant questions.

1. **RAG-Style Scoring Logic**

* **Exact-match path:**
* If an exact match is found **and** its old overall score > 70, combine **70% old** + **30% fresh rubric**.
* If an exact match is found **but** its old score ≤ 70, use **100% fresh rubric**.
* **Relevance path (no exact match):**
* Check **relevance** among the top-3 FAISS neighbors.
* If relevant old questions exist, average their old scores and combine **30% average old** + **70% fresh rubric**.
* If none are relevant, use **100% fresh rubric**.

1. **Three-Run Stability**

To mitigate variance inherent in single LLM calls, each answer is scored under the rubric **three times independently**. We then:

* + **Average** each criterion’s numerical scores.
  + **Summarize** the three one-sentence rationales into a single concise explanation via an auxiliary LLM prompt.
  + Compute the **overall score** as the mean of these averaged criterion scores.

### Evaluation Metrics

We define clear, transparent metrics for both domain evaluations.

1. **Per-Criterion Scores**
   * Each of the 5–8 criteria receives an integer score in [0–100] and a one-sentence explanatory rationale.
2. **Overall\_Score**
   * Computed as the arithmetic mean of the per-criterion scores (each criterion equally weighted).
3. **Threshold & Weight Rationale**

 70**% Threshold:** chosen as an empirically validated “passing” bar in prior pilot studies—above this, past judgments are deemed reliable enough to partially re-use.

 Weighted **Blending (70/30 or 30/70):** we give greater weight to fresh evaluation when past scores lie below confidence thresholds or when similarity is lower, balancing consistency with fairness.

 Arithmetic **Mean:** treats all criteria uniformly, aligning with the rubric’s intent to capture orthogonal evaluation aspects.

1. **Reporting Outputs**
   * Final JSON per question includes:

A computer code with text

AI-generated content may be incorrect.

1. **Illustrative Results**
   * *Table 8.3: Sample Technical & HR Evaluation Breakdown*
   * *Figure 8.3: Distribution of Overall\_Score Across Pilot Dataset*

By meticulously cleaning our source texts, crafting domain-specific rubrics, designing a retrieval-augmented scoring workflow, and defining transparent metrics, we create a rigorous, reproducible methodology for evaluating candidate answers in both Technical and HR domains.

# Implementation

# Testing and evaluation

## Testing

## Evaluation

# Results and Discussions

# Conclusions and Future Work

## Summary

Summarise what you have achieved.

## Future Work

Explain any limitations in your results and how things might be improved. Discuss how your work might be developed further. Reflect on your results in isolation and in relation to what others have achieved in the same field. This self-analysis is particularly important. You should give a critical evaluation of what went well, and what might be improved.

References

Use the *Reference* paragraph style to enter and cross-reference document references. Books ‎[1], standards ‎[2], reports ‎[3], journal articles ‎[4], conference papers ‎[5], and web pages ‎[6] are conventionally presented in slightly different ways.

1. Greene, D. and Williams, P. C. *Linear Accelerators for Radiation Therapy*, Second Edition. IOP Publishing Ltd., Bristol and Philadelphia, 1997.
2. ISO. *Language Of Temporal Ordering Specification*, ISO 8807, International Organization for Standardization, Geneva, 1989.
3. Jacobson, J. and Andersen, O., editors. *Software Controlled Medical Devices*. SP Report 1997:11, Swedish National Testing and Research Institute, Sweden, 1997.
4. Turner, K. J. The Rules for Sailing Races on PDAs, *J. Navigation*, 23(5):114-240, May 2002.
5. Ji, H. and Turner, K. J. Specification and Verification of Synchronous Hardware using LOTOS. In Wu, J. Chanson, S. T. Gao, Q. editors, *Proc. Formal Methods for Protocol Engineering and Distributed Systems* (FORTE XII/PSTV XIX), pages 295-312, Kluwer Academic Publishers, London, UK, October 1999.
6. University of Stirling. Computing Science and Mathematics Research Home Page, <http://www.cs.stir.ac.uk/research>, April 2002.

Appendix I

Appendix II