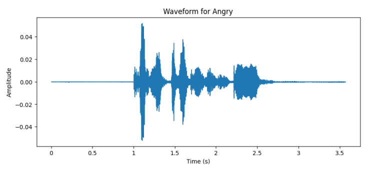
Comparative Analysis of Deep Learning Methods for Speech Emotion Recognition

Vindhya Jain [1] Shruti Chaudhary [1] Neermita Bhattacharya [1]

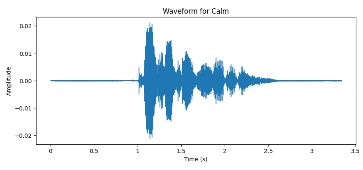
Indian Institute of Technology, Jodhpur, Jodhpur 342037, India
`{b22ai060, b22ai037, b22cs092}@iitj.ac.in`

Abstract

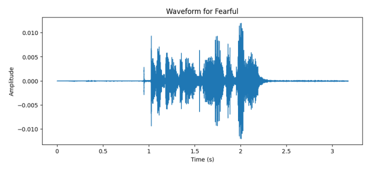
Speech Emotion Classification is a crucial task in affective computing, enabling machines to recognize human emotions from speech signals. For individuals who are deaf or hard of hearing, SER can enhance subtitles by adding emotional annotations (e.g., [angrily] or [happily]) to improve comprehension. Additionally, for blind individuals, SER can enable speech-to-Braille conversion with emotion indicators, helping convey the speaker's tone and intent. In this project, we explored multiple machine learning (ML) and deep learning (DL) models to classify emotions in speech using the RAVDESS dataset. Our approach included both traditional machine learning methods and deep learning architectures. We implemented a Random Forest classifier as a baseline ML model, followed by Convolutional Neural Networks (CNNs) to extract spatial features from spectrograms. Additionally, we experimented with CNN combined with a Multi-Layer Perceptron (CNN + MLP) for enhanced feature representation and an MLP-only model for direct classification. Furthermore, we leveraged transformer-based architectures, specifically Wav2Vec2 models, using both the pretrained facebook/wav2vec2-base model with a custom classifier and a fine-tuned ehcalabres/wav2vec2-lg-xlsr-en-speech-emotion-recognition model for direct classification. The Random Forest achieved an accuracy of 62%, while the MLP, CNN and CNN + MLP achieved accuracies of 69.91%, 77.88% and 46.3% respectively. The transformer models performed outstanding with accuracies reaching 86.81% and 97.22% on the test dataset. Our study provides insights into the effectiveness of different architectures for speech emotion recognition, highlighting the trade-offs between computational complexity, feature extraction capabilities, and classification performance. All code files and analyses are available at Speech-Emotion-Recognition-Methods.



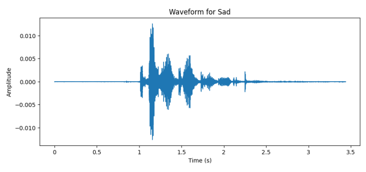
(a) Waveform for angry



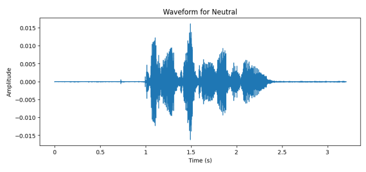
(b) Waveform for calm



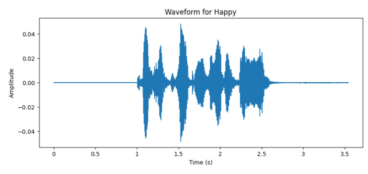
(c) Waveform for fearful



(d) Waveform for sad



(e) Waveform for neutral



(f) Waveform for happy

Figure 1: Waveforms of audio samples from RAVDESS.

Contents

1 Introduction

1.1 The Importance of Speech Emotion Recognition

Speech Emotion Recognition (SER) is a critical field in artificial intelligence that enables machines to understand human emotions through vocal cues.

Its significance spans multiple domains:

- Human-Computer Interaction (HCI)
  - Enhances user experience in voice assistants (e.g., Siri, Alexa) by enabling emotionally intelligent responses.
  - Improves accessibility for individuals with disabilities by adapting interfaces based on emotional state.
- Mental Health & Healthcare
  - Supports early detection of depression, anxiety, or stress through vocal biomarkers.
  - Enables telehealth platforms to monitor patient well-being remotely.
- Deaf Accessibility: Emotion-Enhanced Subtitles
  - Traditional subtitles only convey what is said, not how it's said—missing emotional context critical for full comprehension (e.g., sarcasm, urgency, or sadness).
  - AI-powered emotion-aware subtitles can label speech with emotional cues.

This project explores SER using RF, CNN, MLP and Transformer architectures to classify emotions from speech audios using traditional ml and deep learning approches.

1.1.1 Methods Overview

In our speech emotion recognition project, we implemented five machine learning approaches to comprehensively analyze and classify emotional states.

1. We began with ***Random Forest*** as a robust traditional ensemble method, chosen for its effectiveness with structured emotional feature data and inherent interpretability through feature importance rankings.

2. For neural network baselines, we employed ***Multilayer Perceptrons*** (MLPs) to establish non-linear decision boundaries capable of learning complex patterns in emotional feature spaces.

3. Recognizing the importance of spatial feature extraction, we implemented ***Convolutional Neural Networks*** (CNNs), which excel at processing spectrogram inputs and automatically learning hierarchical emotional patterns through their specialized architecture.

4. To leverage the complementary strengths of different architectures, we developed a ***hybrid CNN+MLP*** model that combines CNN's powerful feature extraction capabilities with MLP's classification prowess, enabling both local and global emotional pattern recognition.

5. Finally, we incorporated ***Transformer*** models to utilize state-of-the-art attention mechanisms, particularly effective for capturing long-range dependencies in speech signals and identifying emotionally salient features through self-attention.

This methodological progression from traditional machine learning to advanced deep learning architectures was deliberately designed to provide a systematic comparison of approaches, ranging from interpretable models to complex neural networks, while ensuring flexibility across different emotional feature representations and input modalities.

The selection of these five distinct approaches allows for thorough evaluation of their relative performance in emotion recognition tasks, with each method bringing unique advantages: Random Forest offers transparency and robustness, MLPs provide flexible non-linear modeling, CNNs specialize in spatial feature extraction, the hybrid model combines architectural strengths, and Transformers deliver cutting-edge sequence modeling capabilities. Together, they form a comprehensive framework for analyzing emotional patterns across the spectrum of contemporary machine learning techniques.

1.1.2 Classical Machine Learning Methods

Speech Emotion Classification is a well-studied problem in affective computing, where classical machine learning techniques have played a significant role in recognizing emotions from speech signals. Traditional approaches rely on carefully engineered features extracted from raw audio signals, which are then used to train machine learning classifiers. The most commonly used classifiers in this domain include Support Vector Machines (SVMs), Random Forests (RF), k-Nearest Neighbors (k-NN), Decision Trees (DT), and Gradient Boosting models. These methods have been effective in capturing emotion-specific patterns when trained on meaningful acoustic features.
Feature extraction is a crucial step in classical machine learning approaches, as raw waveforms alone do not provide sufficient discriminative information. Commonly extracted features include:

- Prosodic Features: Pitch, energy, intensity, speaking rate—representing the rhythm and tone of speech.
- Spectral Features: Spectral centroid, spectral bandwidth, spectral contrast—capturing frequency domain characteristics.
- Cepstral Features: Mel-Frequency Cepstral Coefficients (MFCCs), Linear Predictive Coding (LPC), and Chroma features—widely used in speech processing due to their ability to model the timbral aspects of speech.
- Temporal Features: Zero-crossing rate, formants, and voice onset time—describing dynamic changes in speech signals.

In this study, we employed a Random Forest classifier to evaluate the effectiveness of these handcrafted features for speech emotion classification. Our work highlights the strengths and limitations of traditional machine learning approaches, emphasizing the role of feature engineering and model selection in achieving robust speech emotion recognition. It also emphasizes the need for Deep Learning methods to learn robust features. [2]

1.1.3 Deep Learning Methods

- Multi-Layer Perceptron (MLP)
  - The Multi-Layer Perceptron (MLP) is one of the simplest forms of neural networks, consisting of multiple layers of neurons, including an input layer, one or more hidden layers, and an output layer. In the context of Speech emotion recognition, MLPs can be used to classify emotions based on extracted features from speech signals, such as Mel-frequency cepstral coefficients (MFCCs) or spectrograms.
- Convolutional Neural Network (CNN)
  - Convolutional Neural Networks (CNNs) are particularly effective for processing grid-like data, such as images and spectrograms. In SER, CNNs can be applied to spectrogram representations of audio signals, allowing the model to learn spatial hierarchies of features.
  - CNNs excel at capturing local patterns and features, making them suitable for identifying emotion-related characteristics in speech.
- CNN + MLP
  - Combining CNNs with MLPs leverages the strengths of both architectures. In this hybrid model, CNNs are used to extract features from spectrograms or other representations of audio data, while MLPs are employed to classify the extracted features into different emotional categories.
  - The model can capture both spatial and non-linear relationships, improving overall performance in SER tasks
- Transformer
  - Transformers have gained popularity in various domains, including natural language processing and, more recently, in audio processing tasks. They utilize self-attention mechanisms to capture long-range dependencies in sequential data, making them suitable for Speech Emotion Recognition.

- Transformers can model temporal relationships in speech data effectively, allowing them to capture context and nuances in emotional expression.
- Key Contributions of Transformers in SER:
  * Self-Attention Mechanism enables the model to focus on different parts of the speech signal, identifying which words or phonemes are most indicative of emotion.
  * Transformers excel at understanding the context over long sequences, which is crucial for interpreting emotions that depend on the entire sentence rather than isolated words.
  * Unlike recurrent neural networks (RNNs), which process data sequentially, transformers can analyze all parts of the input simultaneously.

2 About the Dataset: RAVDESS

The RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song) dataset is a well-established multimodal emotional database containing professional actors' vocal and facial expressions.

The audio-only portion available on Kaggle comprises 1,440 high-quality speech samples (16-bit, 48kHz .wav files) recorded by 24 trained actors (12 male, 12 female) producing two linguistically neutral North American English phrases. The dataset captures seven emotional states (calm, happy, sad, angry, fearful, disgust, surprised) plus neutral, with each emotion expressed at two intensity levels (normal and strong), except neutral which has no intensity variation.

Each audio file follows a systematic naming convention where a 7-part numerical code precisely identifies its characteristics. For instance, the filename "03-01-06-01-02-01-12.wav" decodes as: an audio-only recording (03) of speech (01) expressing fear (06) at normal intensity (01), using the "Dogs are sitting by the door" phrase (02), first repetition (01), performed by the 12th actor (female).
This granular encoding allows researchers to precisely control for variables like gender, emotional intensity, and lexical content during experimentation.

The dataset's particular strengths include its professional actor performances, balanced gender representation, controlled lexical content, and validated emotional expressions. The inclusion of two intensity levels for emotions (except neutral) makes it especially valuable for studying subtle gradations in emotional expression.

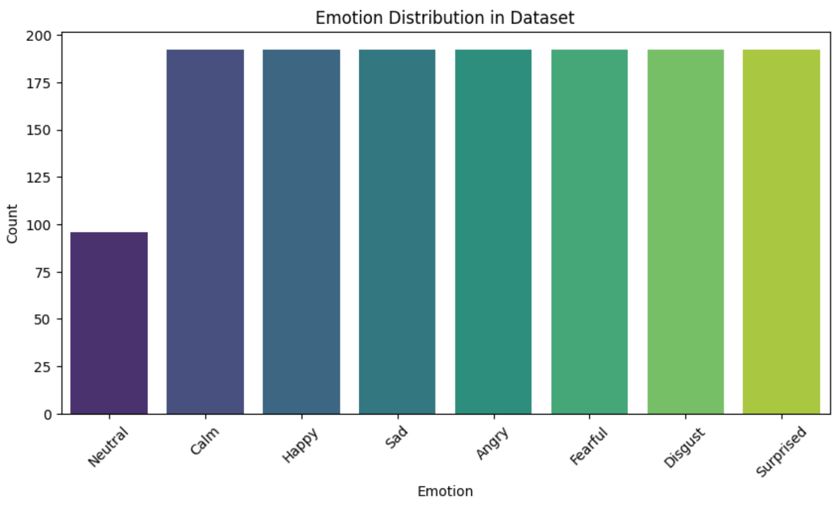


Figure 2: Emotion Distribution in the RAVDESS Dataset.

.

3 Methodologies

3.1 Random Forest

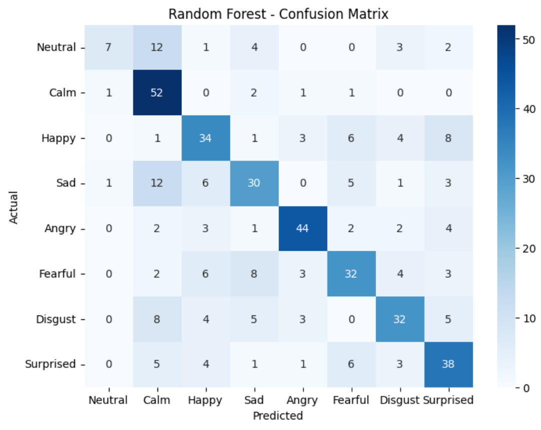
3.1.1 Overview of the Random Forest Classifier

- Model Name: Random Forest Classifier for Speech Emotion Recognition using RAVDESS dataset
- Purpose: This model is designed to classify emotional states from speech audio by analyzing acoustic features extracted from audio files.
- Components:
  - Feature Extraction Pipeline: Processes raw audio files to extract comprehensive acoustic features using librosa library.

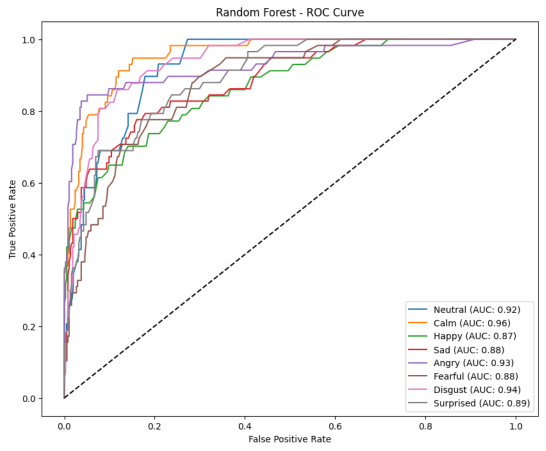
- Data Preprocessing: Handles filtering of speech files and organization of features into structured format.
- Random Forest Classifier: Ensemble learning method that operates by constructing multiple decision trees and outputting the class that is the mode of the classes of the individual trees.

- Training Strategy:
  - Grid Search Cross-Validation: Systematic hyperparameter optimization to find the best model configuration across multiple parameters.
  - Stratified Train-Test Split: Ensures balanced representation of emotion classes in both training and test sets.

- Implementation Details:
  - Dataset Processing:
    - Downloaded RAVDESS Emotional Speech Audio dataset using kagglehub.
    - Filtered to include only audio-only (03) and speech (01) files.
    - Created a structured output directory for the processed files.
  - Feature Extraction:
    - Extracted a rich set of acoustic features using librosa:
      - Signal-to-Noise Ratio (SNR)
      - Mel-Frequency Cepstral Coefficients (MFCCs) - 40 coefficients extracted, 20 used as individual features
      - Chroma STFT features
      - Mel spectrogram statistics
      - Spectral contrast
      - Tonnetz (tonal centroid features)
      - Spectral centroid, rolloff, and bandwidth
      - Tempo analysis
      - Root Mean Square (RMS) energy
      - Zero crossing rate
    - Features were computed for each audio file and organized into a pandas DataFrame.
    - Metadata was extracted from filenames including emotion, intensity, statement, repetition, actor ID, and gender.
  - Model Architecture:
    - RandomForestClassifier from scikit-learn with hyperparameter optimization.
    - Hyperparameters tuned via GridSearchCV:
      - n_estimators: [100, 300] - Number of trees in the forest
      - max_depth: [10, 20, None] - Maximum depth of the trees
      - min_samples_split: [2, 5] - Minimum samples required to split a node
      - min_samples_leaf: [1, 2] - Minimum samples required at a leaf node
      - criterion: ['gini', 'entropy'] - Function to measure the quality of a split
    - 5-fold cross-validation used during hyperparameter search.
  - Training Setup:
    - Categorical features encoded using LabelEncoder.
    - Data split into training (70%) and testing (30%) sets with stratification to maintain class distribution.
    - Random state fixed at 42 for reproducibility.
    - Grid search performed to find optimal hyperparameters.
  - Evaluation Metrics:
    - Classification report showing precision, recall, and F1-score for each emotion category.
    - Confusion matrix visualization to show per-class performance.
    - ROC curves with AUC scores for each emotion class when applicable.
    - The overall accuracy measured was 62%.



(a) Confusion Matrix of Random Forest



(b) ROC-AUC Curve of Random Forest

Figure 3: Test metrics for Random Forest

3.2 Multilayer Perceptron (MLP)

- Introduction :
  The Multi-Layer Perceptron (MLP) is a type of feedforward neural network that consists of multiple layers of neurons, enabling it to learn complex patterns in data. This report provides an overview of an MLP model implemented using PyTorch, highlighting its architecture, components, and functionality.

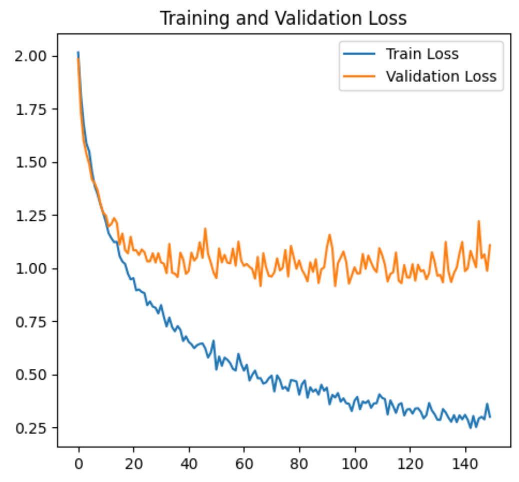
- Model Architecture :

Table 1: Hidden Layer Architecture of the MLP Model

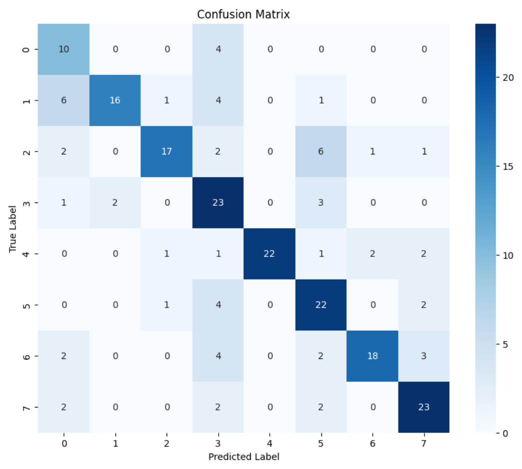
Layer	**Type**	**Input Size**	**Output Size**	**Activation Function**	**Dropout Rate**
1st Hidden Layer	Linear	*input_dim*	256	ReLU	0.4
	Batch Normalization	256	256	-	-
2nd Hidden Layer	Linear	256	128	ReLU	0.4
	Batch Normalization	128	128	-	-
3rd Hidden Layer	Linear	128	64	ReLU	0.3
	Batch Normalization	64	64	-	-

- Implementation Details:
  - Data Preprocessing
    - The `filter_speech_files` function traverses the directory structure, identifying .wav files that match a specific naming convention (e.g., files starting with "03-01"). These files are copied to a designated output directory for further processing.
  - Feature Extraction
    - The `compute_snr` function calculates the signal-to-noise ratio, which is a measure of signal quality.
    - The `extract_features` function extracts various audio features, including Mel-frequency cepstral coefficients (MFCCs), chroma features, spectral contrast, and more.
    - The `process_dataset` function iterates through the audio files, extracts features, and organizes the data into a Pandas DataFrame.
  - Dataset Splitting
    - The dataset is split into training, validation, and test sets.
    - Training Set: 70% of the data for model training.
    - Validation Set: 15% for tuning hyperparameters and preventing overfitting.

    - ∗ Test Set: 15% for evaluating the model's performance on unseen data.
  - Converting to PyTorch Tensors
    - ∗ The features and labels are converted into PyTorch tensors for compatibility with the MLP model.
  - SpeechDataset Class
    - ∗ A custom SpeechDataset class is defined which inherits from PyTorch's Dataset and manages the input features and labels.
    - ∗ It includes methods for initialization, length retrieval, and item access.
  - DataLoaders
    - ∗ Data loaders are created for training, validation, and test datasets with a batch size of 32.
  - Training Setup
    - ∗ The model uses the cross-entropy loss function
    - ∗ The Adam optimizer is employed to update the model's parameters with a learning rate of 0.001.
  - Visualization of Training and Validation Loss
    - ∗ After training the model, a plot is generated to compare the training and validation losses.
    - ∗ This visualization allows for a clear comparison between the training and validation losses, providing insights into the model's performance.
  - Model Evaluation
    - ∗ After evaluating the model, the test accuracy is reported at 69.91% .
    - ∗ A confusion matrix is also generated to further analyze the model's performance.



(a) Plot of loss during training and validation for MLP



(b) Confusion Matrix (MLP)

Figure 4: Loss and accuracy during training and validation, and the confusion matrix.

3.3 Convolutional Neural Networks (CNN)

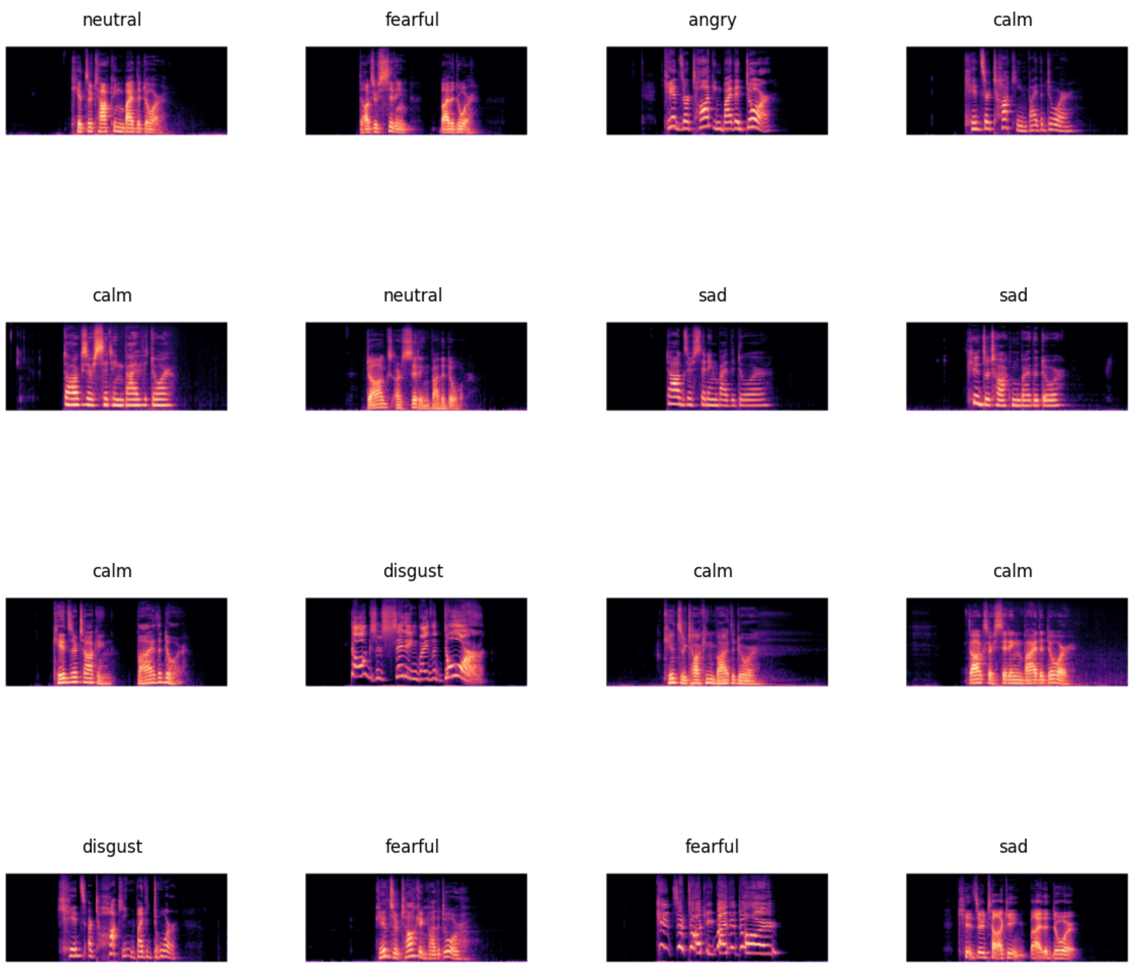
In this section, we use Convolutional Neural Networks (CNNs) to classify emotions from speech by converting audio signals into spectrograms. Spectrograms are visual representations of sound frequencies over time, capturing both temporal and frequency information in a format that CNNs can effectively process.

This approach is particularly powerful because CNNs excel at recognizing patterns in images, and spectrograms essentially transform audio into a visual format where emotion-related patterns can be detected. Unlike traditional methods that rely on handcrafted features such as pitch, energy, and MFCCs, CNNs can automatically learn relevant features from spectrograms, potentially leading to more robust and accurate emotion classification.

Moreover, spectrogram-based CNN models are well-suited for handling variations in speech, such as tone, intensity, and rhythm, which are crucial for distinguishing emotions. By utilizing deep learning, we reduce the need for extensive manual feature engineering and allow the model to learn rich, hierarchical representations of emotional cues from raw speech data.

3.3.1 Processing Spectrogram Images for Emotion Recognition

The RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song) dataset is downloaded and each audio file is converted into a mel spectrogram using librosa.



Dataset and Dataloader Creation:

- Images are resized to 128×128 pixels.
- Labels are extracted from filenames, representing emotion classes.
- Augmentations (if enabled) are applied before converting images to tensors to enhance model generalization:
  - Time Masking: Randomly masks a segment along the time axis, simulating missing speech.
  - Frequency Masking: Randomly masks a segment along the frequency axis, mimicking distortions in audio.
  - Gaussian Noise: Adds random noise to the spectrogram, helping the model generalize better.
- The images are normalized for efficient training.

- Splitting into Train, Validation, and Test sets (70:15:15)
- Dataloaders are created with batch size 64

3.3.2 Training and Evaluation Strategy

Training Process
The model is trained using *Cross-Entropy Loss*, a standard loss function for multi-class classification tasks. We employ the *Adam optimizer* with a learning rate of 0.001 and *weight decay* of 5e-5 to prevent overfitting. Additionally, *gradient clipping* (max_norm=1.0) is applied to stabilize training by preventing large gradient updates.

A *learning rate scheduler* (ReduceLROnPlateau) is used to adjust the learning rate dynamically based on validation loss, reducing it by a factor of 0.5 if the loss plateaus for 3 epochs. The model is trained for a maximum of 50 epochs (except pretrained CNN-3, finetuned for 25/30 epochs), but *early stopping* is implemented if the validation loss stabilizes, ensuring efficient training.

Evaluation Strategy
The trained model is evaluated on a separate test set, using metrics such as *accuracy*, *precision*, *recall*, and *F1-score* to measure classification performance. A *confusion matrix* is plotted to visualize model predictions across different emotion classes.

3.3.3 Implementation Details

Three CNN architectures with varying depths (3-layer, 4-layer, and 5-layer) were implemented and evaluated - with normal & augmented data. All models use:

- Convolutional Layers: 3×3 kernels with padding=1 to preserve spatial dimensions.
- Batch Normalization: Applied after each convolutional layer.
- Max Pooling: 2×2 pooling with stride=2 to downsample feature maps.
- Activation: ReLU non-linearity after each conv + BN.
- Fully Connected Layers: Final classification layers with softmax (implicit in cross-entropy loss).

Model Specifications

Model	Conv Layers	Feature Maps (Channels)	FC Layers	Output Dim (Before FC)	Parameters
CNN-3	3	$16 \to 32 \to 64$	2	$64 \times 16 \times 16$	~1.2M
CNN-4	4	$16 \to 32 \to 64 \to 128$	2	$128 \times 8 \times 8$	~1.8M
CNN-5	5	$32 \to 64 \to 128 \to 256 \to 512$	3	$512 \times 4 \times 4$	~9.5M

Table 2: Model Specifications

Notes:
- CNN-5 includes dropout (p=0.2) for regularization.
- Flattening is applied before FC layers.

Greedy Layer-wise Pretraining (CNN-3 Only)
To enhance feature learning, we implemented a greedy layer-wise unsupervised pretraining strategy using convolutional autoencoders. This method helps initialize the network with meaningful weights before fine-tuning on the classification task. The steps involved are:

- Autoencoder Design: Each convolutional layer was pretrained as an autoencoder (encoder: conv layer, decoder: transposed conv).
- Pretraining:
  - Epochs: 50 per layer.
  - Loss: MSE (reconstruction error).

– Features: Activations from previous layer (forward-propagated with frozen weights).

- Finetuning: Pretrained CNN-3 was finetuned for 25 and 30 epochs (separate runs).

Results Summary

Model	Augmented Data	Pretraining	(Fine)tuning Epochs	Test Accuracy	Precision	Recall	F1-Score	Remarks
CNN-3	No	No	50	0.7512	0.7719	0.7512	0.7497	Baseline
CNN-3	Yes	No	50	0.5253	0.5326	0.5253	0.5220	
CNN-4	No	No	50	0.7788	0.7989	0.7788	0.7738	Deeper than CNN-3
CNN-4	Yes	No	50	0.5945	0.6102	0.5945	0.5968	
CNN-5	No	No	50	0.6728	0.6834	0.6728	0.6702	+Dropout
CNN-5	Yes	No	50	0.5991	0.6177	0.5991	0.6022	
CNN-3	No	Yes	25	0.7005	0.7174	0.7005	0.7018	Pretrained
CNN-3	No	Yes	30	0.7281	0.7277	0.7281	0.7243	Pretrained

Table 3: Results Summary with Augmented Data, Precision, Recall, and F1-Score

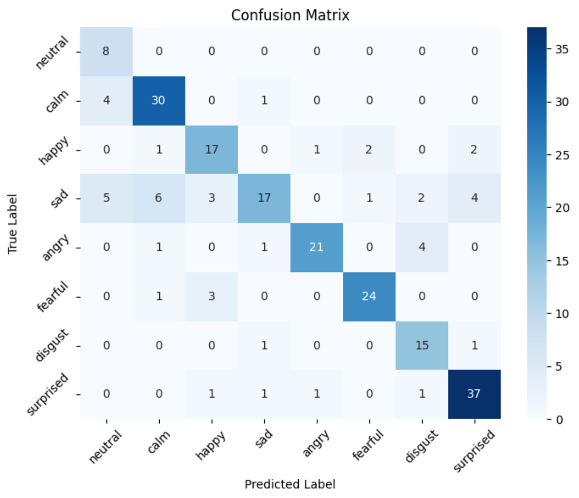


Figure 5: CNN-4

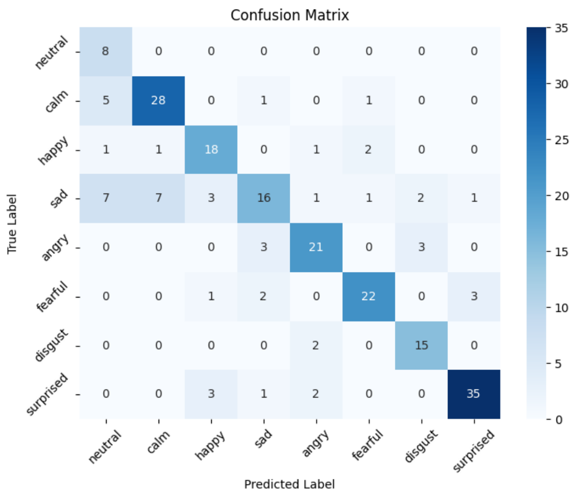


Figure 6: CNN-3

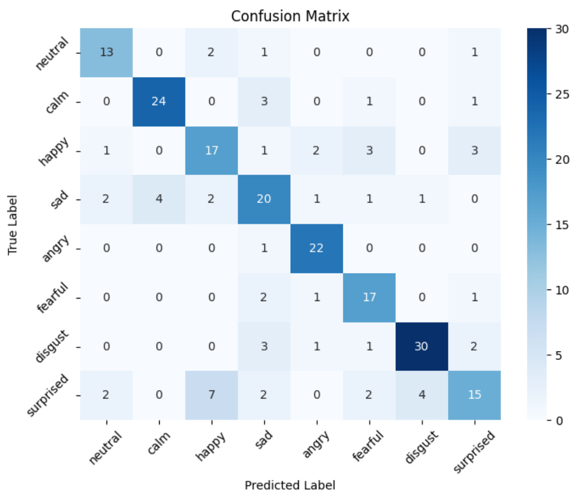


Figure 7: CNN-3 with pretraining

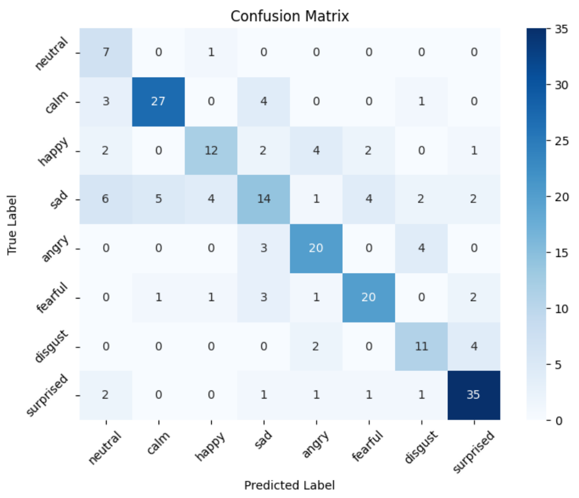


Figure 8: CNN-5

Figure 9: Confusion Matrices for top performing CNNs

3.3.4 Key Observations from Results

- Impact of Model Depth

  - CNN-4 (4 layers) achieved the highest test accuracy (77.88%) without augmentation or pretraining, outperforming CNN-3 (75.12%) and CNN-5 (67.28%).
  - CNN-5 (5 layers + dropout) underperformed despite its capacity, likely due to the small dataset size (insufficient data to leverage deeper architecture).

- Data Augmentation Harmed Performance
  - All models trained with augmentation showed significant drops in accuracy (e.g., CNN-3: 75.12% → 52.53%, CNN-4: 77.88% → 59.45%).
  - Possible Explanation: Augmentation may have introduced unrealistic variations or excessive noise for the small dataset, degrading feature learning.

- Pretraining Benefits (CNN-3)
  - Pretrained CNN-3 achieved 72.81% accuracy with 30 finetuning epochs, narrowing the gap with non-pretrained CNN-3 (75.12%).
  - Pretraining improved parameter initialization, as evidenced by the pretrained CNN-3's competitive accuracy with fewer finetuning epochs

- Precision-Recall Tradeoffs
  - Non-augmented models consistently showed higher precision/recall (e.g., CNN-4: Precision=0.7989, Recall=0.7788).
  - Augmented models had lower metrics, indicating poorer class-wise reliability (e.g., CNN-4 with augmentation: Precision=0.6102, Recall=0.5945).

- Why CNN-5 Underperformed
  - Despite dropout, CNN-5's lower accuracy (67.28%) suggests insufficient data to regularize its large capacity (9.5M parameters).
  - The small dataset likely caused high variance, though dropout mitigated extreme overfitting (implied by less severe drops than CNN-3/4 with augmentation).

- Confusion Matrix Analysis
  - It is noticed that models get confused with emotions with similar audio features. Eg: sad/calm/neutral, happy/surprised
  - The model struggles most with 'calm' labels, frequently misclassifying them as 'neutral' or 'sad,' likely due to overlapping low-intensity features. In contrast, 'angry' achieves high accuracy, benefiting from its distinctive high-intensity characteristics. Future work could refine 'calm' labeling or augment training data to emphasize its unique signatures.

Summary Table (Simplified for Clarity)

Model	Augmentation	Pretraining	Test Accuracy	Key Insight
CNN-4	No	No	77.88%	Optimal depth
CNN-3	No	No	75.12%	Baseline
CNN-3	No	Yes (30 epochs)	72.81%	Pretraining helped
CNN-5	No	No	67.28%	Overcapacity
All	Yes	No	53–60%	Augmentation harmed

Table 4: Summary Table with Key Insights

3.3.5 Conclusion

The experiments highlight:

1. Depth Tradeoff: CNN-4's balance of depth and parameters worked best for the small dataset.
2. Augmentation Caution: Traditional augmentation strategies may degrade performance when data is scarce.
3. Pretraining Utility: Unsupervised pretraining is promising but requires sufficient finetuning.

3.4 Convolutional Neural Networks (CNN) + Multilayer Perceptron (MLP)

- Model Overview
  The CNN + MLP (Convolutional Neural Network + Multi-Layer Perceptron) model is a hybrid architecture designed to process both spatial and scalar data effectively. The CNN processes the spectrograms to extract spatial features. Simultaneously, scalar features is fed into an MLP, which classifies emotions based on these numerical inputs.

- Model Architecture

Table 5: Model Architecture of the CNN + MLP Model

Layer	**Type**	**Input Size**	**Output Size**	**Activation Function**	**Dropout Rate**
CNN Component					
1st Convolutional Layer	Conv2d	(1, H, W)	(64, H, W)	-	-
	Batch Normalization	(64, H, W)	(64, H, W)	-	-
	ReLU	(64, H, W)	(64, H, W)	ReLU	-
	MaxPool2d	(64, H, W)	(64, H/2, W)	-	-
2nd Convolutional Layer	Conv2d	(64, H/2, W)	(128, H/2, W)	-	-
	Batch Normalization	(128, H/2, W)	(128, H/2, W)	-	-
	ReLU	(128, H/2, W)	(128, H/2, W)	ReLU	-
	Flatten	(128, H/2, W)	(128 * H/2 * W)	-	-
MLP Component					
1st Hidden Layer	Linear	*scalar_input_dim*	128	ReLU	0.3
	Batch Normalization	128	128	-	-
2nd Hidden Layer	Linear	128	64	ReLU	0.3
	Batch Normalization	64	64	-	-
Final Classifier					
Combined Layer	Linear	(128 * H/2 * W + 64)	128	ReLU	0.3
	Dropout	128	128	-	0.3
2nd Final Layer	Linear	128	64	ReLU	0.3
	Dropout	64	64	-	0.3
Output Layer	Linear	64	num_classes	-	-

- Implementation Details
  - Feature Extraction
    - The `extract_features` function processes individual audio files to extract various features, including MFCCs, Chroma Features, Spectral Features, Tonnetz, Tempo etc .
  - Dataset Processing
    - The `process_dataset` function iterates through .wav files in a specified directory, extracting features and metadata (emotion, intensity, actor, gender) from the filenames.
    - It compiles the extracted data into a Pandas DataFrame.
    - Non-numeric and irrelevant columns are removed from the DataFrame df to focus on features relevant for model training.
    - Features are scaled using Min-Max scaling to ensure they contribute equally to the model training.
  - Data Splitting
    - The dataset is split into training, validation, and test sets.

    - Initial Split: The data is first split into training (70%) and temporary sets (30%).
    - Final Split: The temporary set is then divided equally into validation and test sets (15% each),
  - Custom PyTorch Dataset Class
    - A custom dataset class, EmotionDataset, is defined to handle the loading of the MFCC and scalar features along with their corresponding labels.
  - Data Loader
    - DataLoaders are created for the training, validation, and test datasets to facilitate batch processing during model training, with a batch size set to 32
  - Loss Function and Optimizer
    - CrossEntropyLoss is used as the loss function.
    - The Adam optimizer is chosen as optimizer, with a learning rate set to 0.001.
  - Visualizing Training and Validation Metrics
    - After training the model the training and validation losses and accuracies are visualized.
  - Model Evaluation
    - Achieved an accuracy of 46.3%
    - A confusion matrix is also generated to further analyze the model's performance.

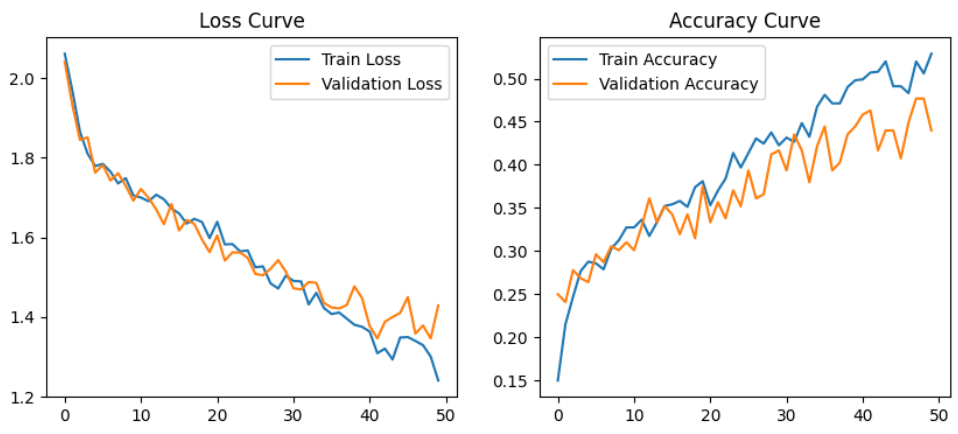


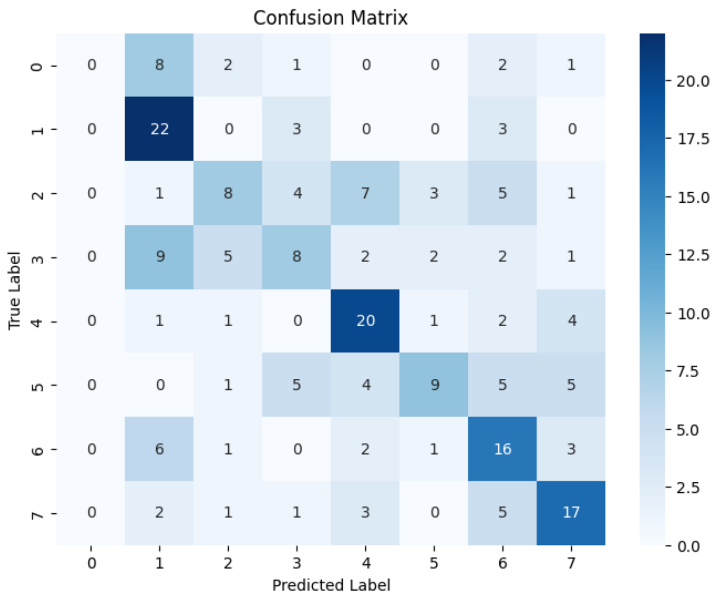
Figure 10: Loss and accuracy plots during training and validation for CNN + MLP

3.5 Transformer

3.5.1 Pretrained and Fine-tuned Wav2Vec2 Model for Speech Emotion Recognition

Model Overview

- Model Name: ehcalabres/wav2vec2-lg-xlsr-en-speech-emotion-recognition available here.
- Purpose: This model is specifically designed for speech emotion recognition tasks. It has been pretrained on a large dataset and fine-tuned on the RAVDESS dataset, which contains emotional speech recordings.
- Components:
  - Wav2Vec2 Feature Extractor: The feature extractor processes raw audio input and converts it into a format suitable for the model. It extracts relevant features from the audio waveform, such as Mel-frequency cepstral coefficients (MFCCs) or spectrograms, which are essential for understanding the emotional content of speech.
  - AutoModelForAudioClassification: This component is a specialized model architecture that takes the extracted features and classifies them into different emotion categories. It is built on top of the Wav2Vec2 architecture, which is a transformer-based model that learns to represent audio data effectively.



True \ Predicted	0	1	2	3	4	5	6	7
0	0	8	2	1	0	0	2	1
1	0	22	0	3	0	0	3	0
2	0	1	8	4	7	3	5	1
3	0	9	5	8	2	2	2	1
4	0	1	1	0	20	1	2	4
5	0	0	1	5	4	9	5	5
6	0	6	1	0	2	1	16	3
7	0	2	1	1	3	0	5	17

Figure 11: Confusion Matrix for CNN + MLP

- Training Strategy:
  - The model has been fine-tuned on the RAVDESS dataset, meaning it has learned to recognize specific patterns associated with different emotions (happiness, sadness, anger, etc) based on the training data. The classifier layers were trained only.
  - Optimization: Uses Adam optimizer with a learning rate of 1e-4 and CrossEntropyLoss with label smoothing of 0.1 to prevent overfitting and improve generalization.
- Implementation Details:
  - Mapping RAVDESS Emotions to Model Labels:
    * A dictionary `ravdess_to_model` is created to map emotion IDs from the RAVDESS dataset to their corresponding emotion labels.
    * A list `model_labels` contains the labels used for classification in the model.
    * A new dictionary `ravdess_to_model_index` is created to map the RAVDESS emotion IDs to the indices of the `model_labels`.
  - Loading the Wav2Vec2 Feature Extractor:
    * The Wav2Vec2 feature extractor is loaded using the pretrained model.
    * This feature extractor will convert raw audio input into a format suitable for the model.
  - Function to Extract Labels from Filenames:
    * The function `get_label_from_filename` takes a filename as input and extracts the emotion label based on the filename structure.
    * The emotion ID is extracted from the filename, and the corresponding label index is retrieved from `ravdess_to_model_index`.
  - RAVDESS Dataset Class:
    * A custom dataset class `RAVDESSDataset` is defined.
    * The constructor initializes the dataset by listing all WAV files in the specified directory.
  - Collate Function:
    * The `collate_fn` function is defined to handle batching of data.
    * It pads the input sequences to ensure they are of the same length and returns a batch of input values and labels.
  - Dataset Splitting:

Table 6: Architecture of Wav2Vec2ForSequenceClassification loaded via AutoModelForAudioClassification

Component	**Description**
wav2vec2	Main model for sequence classification.
feature_extractor	Extracts features from audio input.
conv_layers	7 convolutional layers:
- 1 layer with 1 input channel, 512 output channels (kernel size 10, stride 5)	
- 4 layers with 512 input/output channels (kernel size 3, stride 2)	
- 2 layers with 512 input/output channels (kernel size 2, stride 2)	
feature_projection	Projects features to 1024 dimensions.
encoder	24 encoder layers for processing features.
attention	Self-attention mechanism in each encoder layer.
projector	Projects features to 256 dimensions.
classifier	Final layer for classifying into 8 emotion categories.

  * The dataset is split into training, validation, and test sets with proportions of 80%, 10%, and 10%, respectively.
  * The `random_split` function is used to create the splits based on the defined sizes.
  * The number of train samples: 1152, number of validation samples: 144, and number of test samples: 144.

- Model Initialization:
  * The model is initialized using `AutoModelForAudioClassification` with the pretrained Wav2Vec2 model.
  * The number of labels is set to the length of `model_labels`, and mappings for label indices are provided.
  * `ehcalabres/wav2vec2-lg-xlsr-en-speech-emotion-recognition` was originally fine-tuned as an audio classification model using the Wav2Vec2 architecture. The AutoModelForAudioClassification class detected that this was a Wav2Vec2-based model intended for classification, so it created a Wav2Vec2ForSequenceClassification instance, but the weights in the checkpoint might have been saved with a slightly different architecture than what the current version of the Wav2Vec2ForSequenceClassification class expects. Hence, the weights related to the classifier layers (classifier.dense.bias, classifier.dense.weight, etc.) were not used while loading. Therefore, the classifier layers were trained further.
  * The parameters of the Wav2Vec2 model are frozen to only train the classifier layers.

- Training Setup:
  * An Adam optimizer is defined with a learning rate of 1×10^{-4}.
  * The loss function used is CrossEntropyLoss with label smoothing set to 0.1.
  * The number of epochs the model was trained for were 5.

- Plotting Loss and Accuracy Curves:
  * After training, loss and accuracy curves for both training and validation sets are plotted.
  * The plots provide a visual representation of the model's performance over epochs. As the model was pre-trained on this dataset, the training accuracies start from a very high value of 80% itself, and the validation accuracies reach upto 94%.

- Testing metrics:
  * An impressive accuracy of 97.22% was achieved on the test dataset.
  * An AUC-ROC Score (Multi-class) of 0.9998 was achieved. This means that the model has an exceptional ability to distinguish between the 8 different classes.

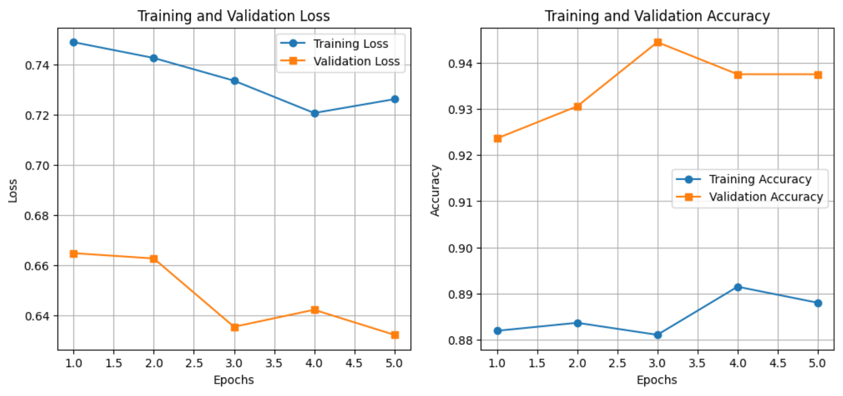
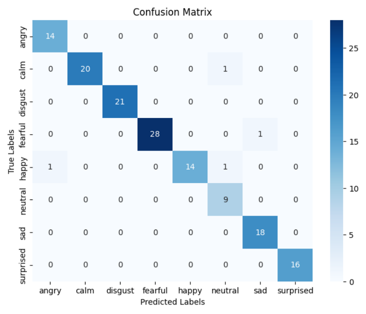
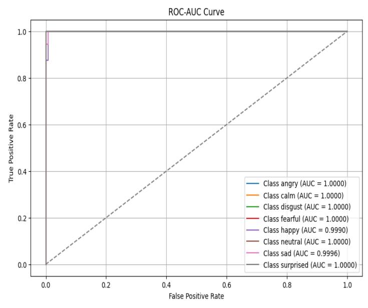


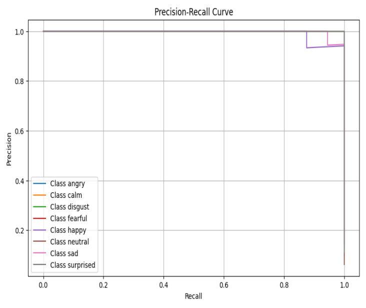
Figure 12: Loss and Accuracy plots of Training and Validation of Pretrained and Fine-tuned Wav2Vec2



(a) Confusion Matrix of Pretrained and Fine-tuned Wav2Vec2



(b) ROC-AUC Curve of Pretrained and Fine-tuned Wav2Vec2



(c) Precision-Recall Curve of Pretrained and Fine-tuned Wav2Vec2

Figure 13: Test metrics of Pretrained and Fine-tuned Wav2Vec2

3.5.2 Base Wav2Vec2 Model for Speech Emotion Recognition

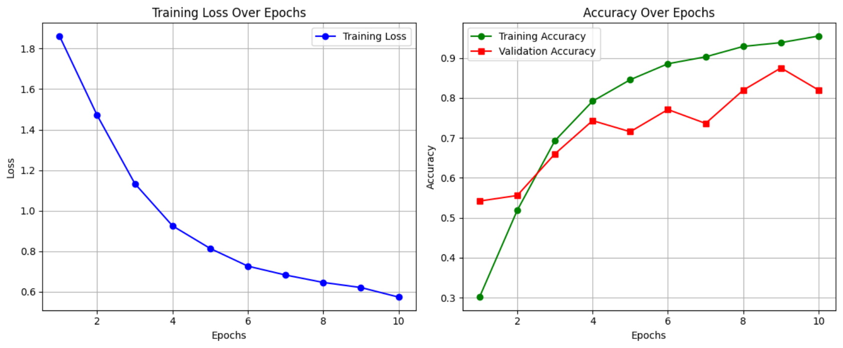
Model Overview

- Model Name: Custom Speech Emotion Classifier based on facebook/wav2vec2-base available here.[1]
- Purpose: This model is custom-designed for speech emotion recognition tasks. It leverages the pre-trained Wav2Vec2 base model as a feature extractor and adds custom classification layers to detect emotions in speech.
- Components:
  - Wav2Vec2 Feature Extractor: Processes raw audio input and converts it into a format suitable for the model. It handles the audio preprocessing steps necessary for the Wav2Vec2 architecture to effectively process speech data.
  - Wav2Vec2Model: The base pretrained model from Facebook that captures rich contextual speech representations. This component acts as the encoder that transforms audio features into high-dimensional embeddings.
  - Custom Classifier Head: A sequence of neural network layers (Linear $\rightarrow$ ReLU $\rightarrow$ Dropout $\rightarrow$ Linear) that takes the 768-dimensional embeddings from Wav2Vec2 and maps them to emotion categories. The architecture progressively reduces dimensionality from 768 to 256 to the number of emotion classes.
- Training Strategy:
  - Transfer Learning: The model utilizes a pretrained Wav2Vec2 backbone and only fine-tunes the last two transformer layers (10 and 11) while keeping earlier layers frozen, which helps preserve the general speech understanding capabilities while adapting to emotion recognition.
  - Optimization: Uses AdamW optimizer with a learning rate of 1e-4 and CrossEntropyLoss with label smoothing of 0.1 to prevent overfitting and improve generalization.

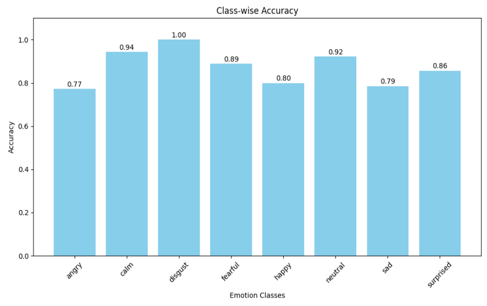
- Implementation Details:
  - Custom Speech Emotion Recognition Model:
    - A custom SpeechEmotionClassifier class is defined, which inherits from PyTorch's nn.Module.
    - The model uses the pretrained "facebook/wav2vec2-base" model as its foundation.
    - The Wav2Vec2FeatureExtractor is loaded from the pretrained model to process raw audio inputs.
  - Model Architecture:
    - The model consists of the Wav2Vec2 base model followed by a custom classifier head.
    - The Wav2Vec2 model extracts 768-dimensional hidden representations from audio inputs.
    - A classifier head is added, consisting of:
      - Linear layer reducing dimensions from 768 to 256
      - ReLU activation function
      - Dropout layer with 0.2 probability for regularization
      - Final linear layer mapping to the number of emotion classes
    - A dropout layer with 0.1 probability is applied after the Wav2Vec2 feature extraction.
    - Mean pooling is used to convert variable-length sequence representations to fixed-size vectors.
  - Training Setup:
    - Transfer Learning: Only the last 2 transformer layers (layers 10 and 11) of the Wav2Vec2 model are fine-tuned, while the rest are frozen.
    - The AdamW optimizer is used with a learning rate of 1×10^{-4}.
    - Cross-Entropy Loss with label smoothing of 0.1 is used as the loss function.
    - The model is trained for 10 epochs.
    - The best model is saved based on validation performance.
  - Training Monitoring:
    - Training loss, training accuracy, and validation accuracy are tracked for each epoch.
    - Visualization of training metrics is performed using matplotlib, showing loss and accuracy curves.
    - The best training and validation accuracies achieved were 95.49% and 87.5% respectively.
  - Evaluation Metrics:
    - Final test accuracy measurement using the best saved model. It was an impressive 86.81% overall.
    - Confusion matrix visualization to show per-class performance.
    - Detailed classification report showing precision, recall, and F1-score for each emotion is displayed below.
    - An AUC-ROC Score of 0.9872 was achieved which proves the fact that transfer learning greatly helps in generalization across tasks.

Table 7: Speech Emotion Classifier Architecture (Wav2Vec2 base)

Component	**Description**
Feature Extractor	Wav2Vec2FeatureEncoder with 7 convolutional layers: • Initial Conv1d layer: 1 → 512 channels, kernel=10, stride=5 • 4 intermediate layers: 512 → 512 channels, kernel=3, stride=2 • 2 final layers: 512 → 512 channels, kernel=2, stride=2
Feature Projection	Converts extracted features to the encoder input dimension: • LayerNorm on 512-dimensional features • Linear projection: 512 → 768 dimensions • Dropout with probability 0.1
Encoder	Transformer-based with 12 encoder layers: • Positional embeddings via convolutional layer • Each encoder layer contains: – Self-attention mechanism – Feed-forward network (768 → 3072 → 768) – Layer normalization and dropout • Only layers 10-11 fine-tuned, others frozen
Classifier Head	Custom classification layers: • Mean pooling across time dimension • Dropout with probability 0.1 • Linear layer: 768 → 256 • ReLU activation • Dropout with probability 0.2 • Output layer: 256 → number of emotion classes

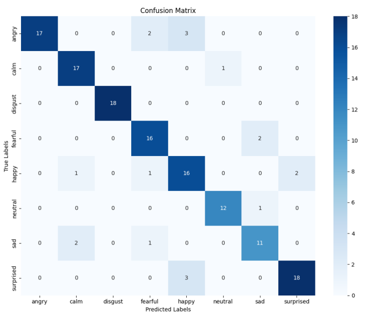


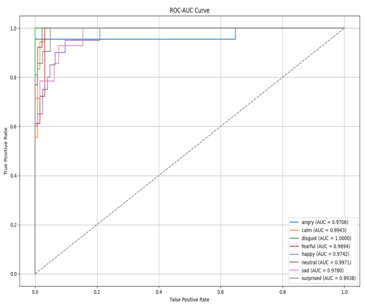
(a) Loss and Accuracy plots for Training and Validation of Custom Speech Emotion Classifier (Wav2Vec2 Base)

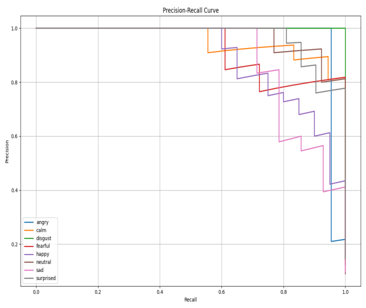


(b) Class-wise Accuracies of Custom Speech Emotion Classifier (Wav2Vec2 Base)

Figure 14: Accuracies over epochs and Class-wise accuracies of Custom Speech Emotion Classifier (Wav2Vec2 Base)







(a) Confusion Matrix of Custom Speech Emotion Classifier (Wav2Vec2 Base)

(b) ROC-AUC Curve of Custom Speech Emotion Classifier (Wav2Vec2 Base)

(c) Precision-Recall Curve of Custom Speech Emotion Classifier (Wav2Vec2 Base)

Figure 15: Test metrics on Custom Speech Emotion Classifier (Wav2Vec2 Base)

References

[1] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations, 2020. URL https://arxiv.org/abs/2006.11477.

[2] MLG FCU. Using python to classify sounds: A deep learning approach. *Medium*, 2021. URL https://medium.com/@mlg.fcu/using-python-to-classify-sounds-a-deep-learning-approach-ef00278bb6ad. Accessed: 2025-03-31.

[3] Diego Rios. Speech emotion recognition with convolutional neural network, 2025. URL https://medium.com/@diego-rios/speech-emotion-recognition-with-convolutional-neural-network-ae5406a1c0f7. Accessed: 2025-03-31.

[3]