**Emotion Recognition from Audio**

A deep learning project for classifying emotions from audio files using MFCC features, spectral features, and neural networks.

**Overview**

This project implements an emotion recognition system that can classify audio samples into 7 different emotional categories. The system uses advanced audio feature extraction techniques and deep learning models to achieve high accuracy in emotion classification.

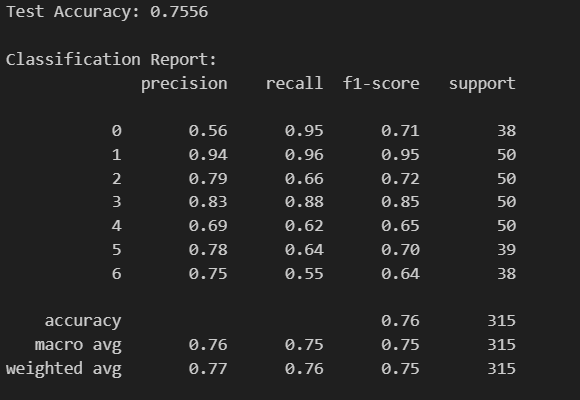
**Features**

* **Audio Preprocessing**: Noise reduction, trimming, and normalization
* **Feature Extraction**: MFCC, Mel-spectrogram, spectral features, and chroma features
* **Data Augmentation**: Time stretching, pitch shifting, and noise addition
* **Deep Learning Models**: Dense neural networks and 1D CNN architectures
* **Comprehensive Evaluation**: Accuracy metrics, confusion matrix, and classification reports

**Emotion Classes**

The system classifies audio into 7 emotion categories:

1. **Neutral** (Class 0)
2. **Happy** (Class 1)
3. **Sad** (Class 2)
4. **Angry** (Class 3)
5. **Fearful** (Class 4)
6. **Disgust** (Class 5)
7. **Surprised** (Class 6)

**Important Note**: The original dataset contained a "Calm" emotion class (originally class 2), but it was removed from the classification task due to overfitting issues. This decision improved the overall model performance and generalization.

**Installation**

**Required Libraries**

pip install pandas numpy librosa matplotlib scikit-learn tensorflow seaborn joblib

**Configuration**

The main configuration parameters are defined at the top of the script:

SAMPLING\_RATE = 16000 # Audio sampling rate

N\_MFCC = 13 # Number of MFCC coefficients

N\_MELS = 48 # Number of Mel frequency bands

HOP\_LENGTH = 512 # Hop length for feature extraction

N\_FFT = 1024 # FFT window size

DURATION = 3.5 # Audio duration in seconds

**Usage**

**1. Data Processing**

The script automatically handles:

* Loading audio files from the specified directory
* Splitting data into training and test sets (80/20 split)
* Applying data augmentation to training data only
* Feature extraction and normalization

**2. Model Training**

Two model architectures are available:

**Dense Neural Network (Default)**:

* 4 dense layers with batch normalization and dropout
* Input: 512 → 256 → 128 → 64 → 7 (output classes)

**1D Convolutional Neural Network**:

* 3 Conv1D layers with pooling and batch normalization
* Global average pooling followed by dense layers

**3. Running the Code**

1. Update the data path in the script:
2. path = r" ……"
3. Run the complete pipeline:
4. python emotion\_recognition.py

The script will:

* Process all audio files
* Train the model with early stopping and learning rate reduction
* Save the best model and preprocessing objects
* Generate evaluation metrics and visualizations

**Data Augmentation**

The training data is augmented using:

* **Time Stretching**: Changing playback speed (0.9x rate)
* **Pitch Shifting**: Shifting pitch by 2 semitones
* **Noise Addition**: Adding Gaussian noise (0.5% of signal amplitude)

**Note**: Augmentation is applied only to the first 5 emotion classes during training to balance the dataset.

**Feature Extraction**

The system extracts multiple audio features:

* **MFCC Features**: 13 coefficients + statistics (mean, std)
* **MFCC Delta**: First-order derivatives of MFCC
* **Mel-Spectrogram**: 48 mel-frequency bands
* **Spectral Features**: Centroid, rolloff, zero-crossing rate
* **Chroma Features**: 12 pitch class profiles

Total feature vector size: ~122 features per audio sample

**Model Architecture**

**Dense Model (Default)**

Input (122 features)

↓

Dense(512) + BatchNorm + Dropout(0.3)

↓

Dense(256) + BatchNorm + Dropout(0.3)

↓

Dense(128) + BatchNorm + Dropout(0.2)

↓

Dense(64) + Dropout(0.2)

↓

Dense(7) + Softmax

**Training Configuration**

* **Optimizer**: Adam (lr=0.001)
* **Loss Function**: Sparse Categorical Crossentropy
* **Batch Size**: 64
* **Epochs**: 100 (with early stopping)
* **Validation Split**: 20% of training data

**Callbacks**

The training uses several callbacks for optimization:

* **ModelCheckpoint**: Saves best model based on validation accuracy
* **EarlyStopping**: Stops training if no improvement for 10 epochs
* **ReduceLROnPlateau**: Reduces learning rate when validation loss plateaus

**Output Files**

The script saves several files to assist with deployment and analysis:

saved/

└── best\_model\_corrected.h5 # Best trained model

**Evaluation Metrics**

The system provides comprehensive evaluation:

* **Accuracy Score**: 0.77
* A screenshot of a computer screen

  AI-generated content may be incorrect.**Classification Report**:
* **Confusion Matrix**: Visual representation of classification resultsA screenshot of a graph

  AI-generated content may be incorrect.
* **Training History**: Loss and accuracy curves

A graph of loss and loss

AI-generated content may be incorrect.

**Performance Considerations**

* **Memory Usage**: Large datasets may require batch processing
* **Training Time**: GPU acceleration recommended for faster training
* **Overfitting Prevention**: Dropout, batch normalization, and early stopping implemented

**Troubleshooting**

**Common Issues**

1. **Memory Errors**: Reduce max\_files\_per\_emotion parameter
2. **CUDA Errors**: Ensure compatible TensorFlow and CUDA versions
3. **Audio Loading Issues**: Verify file paths and audio format compatibility
4. **Feature Extraction Errors**: Check librosa installation and audio file integrity

**Audio File Requirements**

* **Format**: WAV files (other formats may work but not guaranteed)
* **Duration**: Files are automatically padded/truncated to 3.5 seconds
* **Quality**: Higher quality audio generally produces better results