**Emotion Detection**

**Using Speech Recognition and Large Language Models**

**1. Introduction**

**Speech Recognition**  
Speech recognition is the capability of a machine to convert spoken language into written text. It involves analyzing audio signals and mapping them to a sequence of words using acoustic and language models. This technology is foundational to voice assistants, transcription tools, and accessibility features.

**Large Language Model (LLM)**  
A Large Language Model (LLM) is an artificial intelligence system trained on extensive text data to understand and generate human-like language. Models such as GPT, BERT, and T5 can comprehend context, generate coherent sentences, and perform complex language-based tasks.

**How Are LLMs Used in Speech Recognition?**  
While traditional speech recognition focuses on converting audio to text, LLMs enhance this process by:

* Improving contextual understanding of transcribed speech
* Disambiguating homophones and sentence structures
* Extracting sentiments and emotions from text

**Multimodal Sentiment Analysis**  
Multimodal sentiment analysis involves using multiple forms of data—such as text, audio, and video—to better understand and classify human emotions. This technique is especially effective when voice tone and facial expressions are analyzed together to interpret emotional context more accurately.

**Current Trends in Emotion Detection and Speech Recognition:**

* Use in virtual meetings and smart assistants
* Customer sentiment analysis in call centers
* Applications in mental health monitoring and therapy

**Future Prospects:**

* Real-time multilingual emotion detection
* Combining facial and vocal cues for higher accuracy
* Adaptive AI for personalized emotion understanding

**2. Problem Statement**

**What the Project Does:**  
This project detects human emotions from speech and optionally from multimodal inputs by combining speech recognition with LLM-based text and audio analysis.

**Objective:**  
To build an advanced emotion detection system that integrates audio signal processing, speech-to-

text conversion, and natural language understanding to detect and classify user emotions in real time.

**3. Limitations of Existing Systems**

* Limited emotional range (only basic emotions detected)
* Ignorance of multimodal inputs like facial expressions
* Poor performance with noisy or low-quality audio
* Lack of contextual emotion understanding

**4. Proposed System**  
The enhanced system introduces a multimodal pipeline:

* Audio input is processed using Librosa to extract features like MFCCs and pitch.
* Whisper (by OpenAI) performs speech-to-text transcription.
* An LLM analyzes the transcribed text to understand semantic content and emotional tone.
* Emotion classification is done using a model such as SVM, integrating both audio and textual features.

**5. System Pipeline**

**Hierarchical Structure:**

1. **Input Layer**: MP4 file or live speech input (with audio and video)
2. **Preprocessing Layer**:
   * Extract audio using MoviePy
   * Noise reduction and silence trimming
3. **Feature Extraction**:
   * Audio: MFCCs (13), Pitch (13) → Combined into 26 features using Librosa
   * Text: Transcription using Whisper
4. **Classification Layer**:
   * Model (e.g., SVM) classifies emotions based on audio features
   * LLM enriches analysis using semantic context from text
5. **Output Layer**:
   * Transcribed speech
   * Detected language
   * Predicted emotion (happy, sad, angry, neutral, etc.)

**Data Collection**  
The following datasets were used:

* **RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song)**: Includes emotional speech and song recordings with 8 emotions.
* **CREMA-D (Crowd-Sourced Emotional Multimodal Actors Dataset)**: Contains audio and video recordings labeled with multiple emotions.
* **TESS (Toronto Emotional Speech Set)**: A collection of voice recordings from two actors, representing seven emotions.

These datasets were chosen due to their rich emotional diversity and multimodal features (audio and video).

**Training**

* Extracted MFCCs and pitch from audio clips
* Converted audio to text using Whisper
* Combined features passed to a classifier (e.g., SVM, MLP)
* LLM-enhanced text embeddings optionally included for richer context
* Training was conducted using cross-validation and grid search for parameter tuning

**Testing**

* Separate validation and test sets ensured unbiased evaluation
* Used unseen speakers and sentences for robust testing
* Performance compared across individual modalities and fused models

**Implementation**

* Implemented in Python using Jupyter Notebook and Google Colab
* Real-time dashboard developed using Streamlit
* Supports audio input via microphone or file upload
* Displays waveform, transcribed text, and predicted emotion

**Loss Evaluation**

* Classification models used **Cross-Entropy Loss**
* Evaluation Metrics:
  + Accuracy
  + Precision, Recall, and F1-Score
  + Confusion Matrix for visualizing predictions
  + ROC-AUC for multi-class evaluation

**Coding:**

**Emotion\_detection\_using voice**

!pip install librosa soundfile scikit-learn openai-whisper streamlit -q

!pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cpu

!pip install moviepy transformers

import librosa

import numpy as np

import soundfile as sf

import os

import whisper

from sklearn.svm import SVC

import joblib

from google.colab import files

# Step 2: Upload Audio Files

uploaded = files.upload()

# Step 3: Feature Extraction

def extract\_features(audio\_path, sr=22050):

y, sr = librosa.load(audio\_path, sr=sr)

    mfccs = librosa.feature.mfcc(y=y, sr=sr, n\_mfcc=13)

    pitch = librosa.piptrack(y=y, sr=sr)[0].mean(axis=0)

    pitch = pitch[:mfccs.shape[1]]

    pitch = np.pad(pitch, (0, mfccs.shape[1] - pitch.shape[0]), 'constant')

    features = np.vstack((mfccs, pitch))

    return features.mean(axis=1)

# Step 4: Model Training and Saving

def train\_model(X\_train, y\_train):

    model = SVC(kernel='linear', probability=True)

    model.fit(X\_train, y\_train)

    joblib.dump(model, 'emotion\_model.pkl')

    return model

def load\_model(path='emotion\_model.pkl'):

    return joblib.load(path)

# Step 5: Prepare Dataset (Replace with real files)

files\_list = ['happy.mp3', 'sad.mp3']  # Replace with uploaded filenames

targets = ['happy', 'sad']

X, y = [], []

for f, label in zip(files\_list, targets):

    if os.path.exists(f):

        features = extract\_features(f)

        X.append(features)

        y.append(label)

X, y = np.array(X), np.array(y)

model = train\_model(X, y)

print("Model trained and saved successfully.")

# Step 6: Load Whisper and Test on Another Audio File

test\_file = 'test.mp3'  # Replace with a test file name from upload

if os.path.exists(test\_file):

    print("\nRunning prediction on:", test\_file)

    model = load\_model()

    asr = whisper.load\_model("base")

    transcription = asr.transcribe(test\_file)

    print("\nTranscribed Text:", transcription['text'])

    test\_features = extract\_features(test\_file).reshape(1, -1)

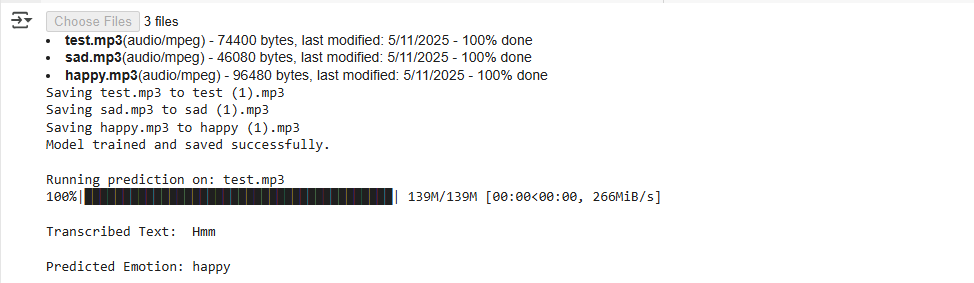
    prediction = model.predict(test\_features)[0]

    print("\nPredicted Emotion:", prediction)

else:

    print("Test file not found. Upload and re-run this block.")

**Output:**

****

**Emotion\_detection\_using image**

#Install dependencies

!pip install -q keras opencv-python

# Download pre-trained emotion model (Mini-XCEPTION, expects 64x64 grayscale)

!wget -q [https://github.com/oarriaga/face\_classification/raw/master/trained\_models/emotion\_models/fer2013\_mini\_XCEPTION.102-0.66.hdf5 -O emotion\_model.h5](https://github.com/oarriaga/face_classification/raw/master/trained_models/emotion_models/fer2013_mini_XCEPTION.102-0.66.hdf5%20-O%20emotion_model.h5)

#Import everything

import cv2

import numpy as np

from keras.models import load\_model

from google.colab import files

from google.colab.patches import cv2\_imshow

#Load model

model = load\_model("emotion\_model.h5", compile=False)

emotion\_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral']

#Upload image

uploaded = files.upload()

#Load face detector

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade\_frontalface\_default.xml")

#Process uploaded image

for filename in uploaded.keys():

    img = cv2.imread(filename)

    gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)

for (x, y, w, h) in faces:

        roi = gray[y:y+h, x:x+w]

        roi = cv2.resize(roi, (64, 64))  # 🔥 fixed shape

        roi = roi.astype("float32") / 255.0

        roi = np.expand\_dims(roi, axis=-1)  # (64, 64, 1)

        roi = np.expand\_dims(roi, axis=0)   # (1, 64, 64, 1)

preds = model.predict(roi, verbose=0)

        label = emotion\_labels[np.argmax(preds)]

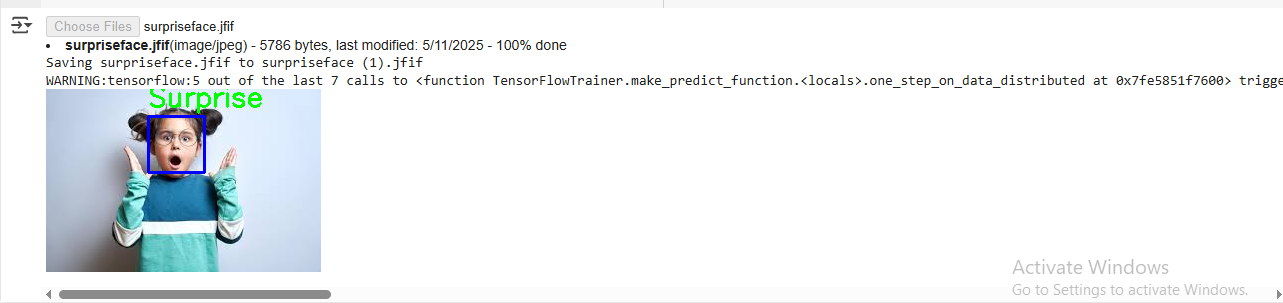
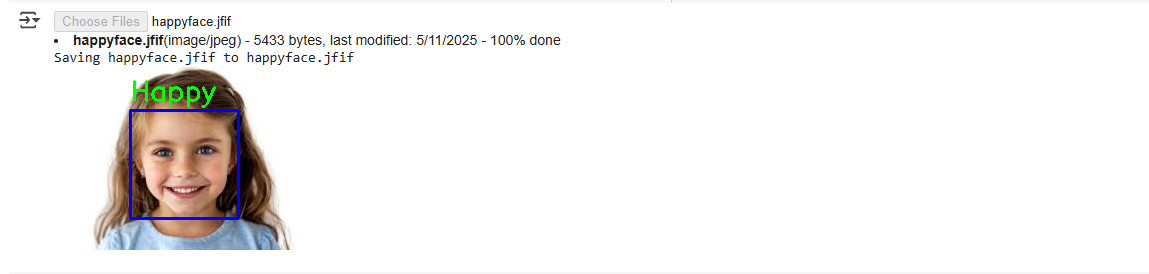
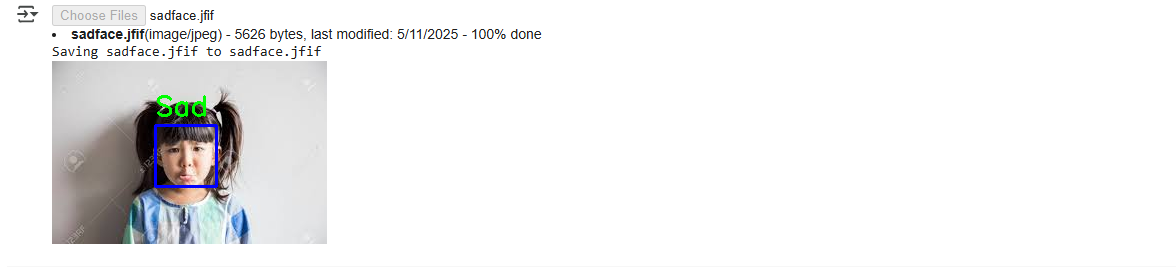
cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

cv2.putText(img, label, (x, y - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, (0, 255, 0), 2)

#Show result

    cv2\_imshow(img)

**Output:**

****

**11. User Interface**

* Upload or record audio/video
* System shows waveform and spectrogram
* Displays transcribed text and detected emotion
* Real-time feedback and interaction

**12. Libraries and Architecture (LLM)**

**Libraries Used:**

* librosa: Audio processing and feature extraction
* transformers: Accessing pretrained LLMs
* streamlit: Web-based dashboard interface
* moviepy: Video processing (extract audio)
* scikit-learn: Model training (e.g., SVM)
* matplotlib, seaborn: Data visualization

**Whisper Architecture:**

* **Input Layer**: Audio preprocessing
* **Encoder**: Feature extraction
* **Decoder**: Text generation

**Librosa Architecture:**

* Load → Preprocess → Feature → Visualize → Utility

**Activation Functions Used:**

* **ReLU**: For hidden layers
* **Softmax**: Final classification layer

**13. Project Limitations**

* May struggle with overlapping or subtle emotions
* High resource demand for real-time inference
* Multimodal analysis requires aligned datasets
* LLMs can misinterpret sarcasm or ambiguous speech

**14. Future Enhancements**

* Integrate facial emotion detection using OpenCV or MediaPipe
* Support extended emotion categories (boredom, excitement, etc.)
* Add support for multilingual and dialectal speech inputs
* Optimize for mobile and edge devices

**15. Conclusion**  
This project demonstrates how modern AI techniques—specifically speech recognition, multimodal analysis, and large language models—can be combined to build an advanced, real-time emotion detection system. Its applications are far-reaching, from virtual therapy and education to customer support and human-AI interaction.