## INTRODUCTION:

### Emotion Detection:

Emotion detection is the use of technology to figure out how a person is feeling. It looks at things like facial expressions, voice tone, body movements, or even the words someone uses to tell if they are happy, sad, angry, or feeling other emotions. It's used in things like improving customer service, helping with mental health, or making video games feel more real.

### LLM:

A **Large Language Model (LLM)** is a type of artificial intelligence that understands and uses human language. It has been trained on a lot of text from books, websites, and other sources so it can read, write, and answer questions like a person. LLMs are used in chatbots, translation apps, and voice assistants to help them talk and respond in a smart, natural way.

### How LLM used in Emotion Detection:

How LLMs are used in emotion detection:

1. **Text Analysis**: LLMs look at written words to figure out how a person might be feeling (happy, sad, angry, etc.).
2. **Sentiment Analysis**: They determine if the text is positive, negative, or neutral.
3. **Context Understanding**: LLMs understand the context to catch emotions, even if words have different meanings in different situations.
4. **Tone Detection**: They can detect tone, like sarcasm or excitement, to better understand emotions.

In short, LLMs read and understand text to detect emotions based on how words are used.

### Current Usage of Emotion Detection:

Where emotion detection is used:

1. **Customer Service:** To improve support and personalize responses.
2. **Mental Health:** Helps monitor emotions and identify signs of stress or depression.
3. **Marketing:** To understand how people feel about products or ads.
4. **Human-Computer Interaction:** Makes virtual assistants and games more responsive to emotions.
5. **Security:** Detects unusual emotional behavior for safety or threat alerts.
6. **Education:** Helps teachers understand student emotions and adjust teaching.
7. **Healthcare:** Assists doctors in assessing patients' emotional states for better care.

These are some ways emotion detection helps in daily life and various industries.

**Future Evolution:**

How emotion detection might evolve in the future:

1. **More Accurate Detection:** It will better understand subtle and mixed emotions.
2. **Multimodal Detection:** Combining facial expressions, voice, body language, and text for better results.
3. **Everyday Devices:** Emotion detection will be built into phones, wearables, cars, and smart homes.
4. **Personalized Experiences:** Apps and devices will adjust to your emotions in real-time (like changing music or lighting).
5. **Better Mental Health Support:** It will help monitor emotions for early mental health interventions.
6. **Ethical Concerns:** More focus on privacy and how emotional data is used.
7. **AI with Empathy:** Virtual assistants and robots could understand and respond with empathy.

In the future, emotion detection will become smarter, more integrated, and personalized**.**

**PROBLEM STATEMENT:**

**Main Objective:**

The **main goal** of this program is to:

- Listen to a video,
- Write down what is said (transcription),
- Find out what emotion the speaker is feeling (like happy, sad, angry, or neutral).

**How it works (in easy steps):**

1. Takes a video and turns it into audio.
2. Uses AI to understand and write down the speech.
3. Checks the sound features (like tone and pitch).
4. Uses a machine learning model to guess the emotion.
5. Shows the emotion, what was said, and the language used.

It's like giving a computer the ability to listen and understand both what is said and how the person is feeling.

**Existing System:**

Limitations of your emotion detection system:

1. **Not Accurate –** It uses random data to train the model, not real emotional voices.
2. **No Real Dataset –** It doesn't learn from real examples of emotional speech.

3. **Only Uses Audio –** It doesn't look at facial expressions or video, only sound.
4. **Few Emotions** – It only checks for basic emotions like happy, sad, angry, and neutral.
5. **Language Issues –** It may not work well for all languages or cultural ways of expressing emotion.
6. **Affected by Noise –** Background sounds can confuse the emotion and speech detection.
7. **Not Real-Time –** It works only on pre-recorded files, not live speech.
8. **Not Personalized –** It doesn't adjust to how different people show emotions.

**Proposed System:**

1. Take a video or audio file as input.
2. Convert the video to audio if needed.
3. Use Whisper AI to write down what was said and detect the language.
4. Extract sound features like pitch and tone using Librosa.
5. Use a real trained model (not random data) to detect emotions like happy, sad, angry, etc.
6. Show the results – what was said, the emotion, and the language.

**Better Than the Old System**

- Trained with real emotional voices
- Can detect more types of emotions
- Can work better with different people and voices
- Can be upgraded to also read facial emotions

**PIPELINE OF THE PROJECT:**

**Dataflow for Emotion Detection:**

**1. Data Collection**

└——> Randomly create features (MFCC + pitch)

└——> Assign random emotion labels (happy, sad, angry, neutral)

**2. Data Preprocessing**

└——> Convert audio to same format (WAV, 16kHz)

└——> Extract features (MFCC, pitch, etc.)

**3. Model Training**

└——> Train a machine learning model (e.g., SVM)

└——> Fit model using simulated features and labels (like happy, sad, etc)

**4. Model Testing/Validation**

└——> Test on simulated or sample test data

└——> Check basic working of classifier

**5. Implementation**

└——> Input: User audio/video file

└——> Convert to audio (if video)

└——> Transcribe using Whisper

└——> Extract features using Librosa (MFCC + pitch)

└——> Use trained model to predict emotion

└——> Show: Transcription, Detected Emotion, Language

**CODING:**

```
!pip install -q openai-whisper moviepy librosa scikit-learn keras opencv-python
```

```
!pip install whisper
```

```
# Import Libraries
import whisper
import librosa
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
from moviepy.editor import AudioFileClip
from google.colab import files
import os
import cv2
from keras.models import load_model
from google.colab.patches import cv2_imshow

# Step 1: Load Whisper Model
asr_model = whisper.load_model("base")
```

```
# Step 2: Transcribe Speech
def transcribe_audio(audio_path):
    result = asr_model.transcribe(audio_path)
    print("Transcription:", result["text"])
    return result["text"], result["language"]
```

```
# Step 3: Extract Audio Features (MFCC + Pitch)
def extract_audio_features(audio_path):
    y, sr = librosa.load(audio_path, sr=16000)
    mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
    pitch, _ = librosa.core.piptrack(y=y, sr=sr)
    pitch_mean = np.mean(pitch, axis=1)
    mfcc_features = np.mean(mfcc, axis=1)
    pitch_features = pitch_mean[:13]
```

```
    return np.concatenate([mfcc_features, pitch_features])

# Step 4: Train Dummy Emotion Classifier
def train_emotion_classifier():
    X = np.random.rand(100, 26)
    y = np.random.choice(['happy', 'sad', 'angry', 'neutral'], size=100)
    le = LabelEncoder()
    y_encoded = le.fit_transform(y)
    classifier = SVC(kernel='linear')
    classifier.fit(X, y_encoded)
    return classifier, le

# Step 5: Classify Audio Emotion
def classify_emotion(features, classifier, le):
    emotion_idx = classifier.predict([features])[0]
    emotion = le.inverse_transform([emotion_idx])[0]
    return emotion

# Step 6: Convert MP4 to WAV
def convert_mp4_to_wav(mp4_path, wav_path):
    audio_clip = AudioFileClip(mp4_path)
    audio_clip.write_audiofile(wav_path, codec='pcm_s16le')
    audio_clip.close()

# Step 7: Combined Audio Processing Function
def emotion_aware_speech_recognition(mp4_path):
    wav_path = "/content/temp_audio.wav"
    convert_mp4_to_wav(mp4_path, wav_path)
    transcription, language = transcribe_audio(wav_path)
    audio_features = extract_audio_features(wav_path)
    emotion = classify_emotion(audio_features, emotion_classifier,
label_encoder)
    print(f"Detected Emotion (Audio): {emotion}")
    print(f"Detected Language: {language}")

# Step 8: Load Pretrained Face Emotion Model
!wget -q
https://github.com/oarriaga/face_classification/raw/master/trained_models/emot
ion_models/fer2013_mini_XCEPTION.102-0.66.hdf5 -O emotion_model.h5
face_model = load_model("emotion_model.h5", compile=False)
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise',
'Neutral']
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
"haarcascade_frontalface_default.xml")

# Step 9: Process Uploaded Face Image
def detect_face_emotion(image_path):
    img = cv2.imread(image_path)
    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))
```

```
        roi = roi.astype("float32") / 255.0
        roi = np.expand_dims(roi, axis=-1)
        roi = np.expand_dims(roi, axis=0)
        preds = face_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)
        print(f"Detected Emotion (Face): {label}")

    cv2_imshow(img)

# Step 10: Train Audio Emotion Classifier
emotion_classifier, label_encoder = train_emotion_classifier()

#Step 11: Upload and Process Media
print("Upload an MP4 file for speech transcription and audio emotion
detection:")
uploaded_audio = files.upload()
for filename in uploaded_audio.keys():
    emotion_aware_speech_recognition(f"/content/{filename}")

print(" Upload a face image (JPG/PNG) for visual emotion detection:")
uploaded_images = files.upload()
for filename in uploaded_images.keys():
    detect_face_emotion(filename)
```

**Libraries Used:**

1. **openai-whisper**
    - Used for speech recognition (transcribes spoken words from audio).
2. **librosa**
    - Extracts audio features like MFCC and pitch.
    - Useful for sound analysis in emotion detection.
3. **numpy**
    - Handles arrays and numerical operations (e.g., combining features).
4. **scikit-learn**
    - Provides machine learning tools:
        - LabelEncoder: Converts emotion labels to numbers.
        - SVC: Support Vector Classifier, used to classify emotions.
5. **moviepy**
    - Used to extract audio from video files (MP4 → WAV).

**Architecture Used:**

LLMs read text and understand the meaning, tone, and context.
They can be used to detect emotions expressed in written or spoken language (after transcription).

**Whisper by OpenAI**

- Whisper is **not a traditional LLM** like ChatGPT or GPT-4.
- It is a **pre-trained deep learning model for speech recognition**.
- Built by OpenAI, Whisper is designed to:
    - **Transcribe audio** into text.
    - **Detect language** spoken in the audio.

**Dataset Used:**

- **Simulated Dataset -** It includes
- Randomly generated audio features:
    - 13 MFCC features
    - 13 pitch features
- Randomly assigned emotion labels:
    - Example emotions:

**LIMITATIONS:**

1. **Fake Data Used**
    - The model is trained on random (simulated) data, not real voices.
2. **Not Accurate in Real Life**
    - Since the data is fake, it won't work well with real emotions.
3. **Few Emotions Only**
    - It can detect only 4 emotions: happy, sad, angry, and neutral.
4. **Ignores Text Meaning**
    - It doesn’t use the actual words (transcription) to find emotion.
5. **Basic Audio Features Only**
    - It uses simple features like MFCC and pitch — not enough for deep emotion understanding.
6. **No Performance Check**
    - It doesn’t check how accurate or correct the model really is.
7. **No Live Audio**
    - It can’t work with real-time audio (like a live microphone).

**FUTURE ENHANCEMENT:**

1. **Use Real Datasets**
    - Replace fake data with **real emotional audio** from actual people to make it more accurate.
2. **Detect More Emotions**
    - Add more emotions like **surprise**, **fear**, or **boredom** for a wider range of feelings.
3. **Real-Time Emotion Detection**
    - Allow the system to listen and detect emotions **live** (from a microphone), not just pre-recorded files.
4. **Better Audio Features**
    - Use more **advanced features** from speech (like speed or tone) to improve emotion detection.
5. **Text Emotion Detection**
    - Analyze the **words themselves** (from transcription) for emotion, using models like **GPT** to understand the **context**.

6. **Multimodal Emotion Detection**
    - Combine **audio, text, and visual data** (like face expressions) to detect emotions even more accurately.
7. **Better User Interface**
    - Create a **simple website** or **app** so users can easily upload files and see results instantly.
8. **Personalized Detection**
    - Make the system **learn from individual users**, improving accuracy for each person.
9. **Support More Languages**
    - Add **more languages** for transcription and emotion detection, making it usable worldwide.
10. **Cloud Service**
    - Make it available as an **online service** that anyone can use via a website or API.

**CONCLUSION:**

This emotion detection project showcases how speech can be used to identify emotions such as happy, sad, angry, and neutral. By using Whisper for speech-to-text conversion and a Support Vector Machine (SVM) for emotion classification, the system processes audio features like pitch and MFCC. However, it relies on simulated data, which limits its real-world accuracy, and it only detects a few basic emotions. To improve the system, we can use real datasets, detect more emotions, and create a real-time user interface. With future upgrades like text analysis using LLMs and multimodal input (audio and visual), the system can become more accurate and practical for real-world applications.