**Emotion Detection Using Speech and Facial Recognition**

**Facial Recognition:**

**1. Introduction**

1. Facial emotion detection is a computer vision technique that identifies human emotions through facial expressions. It uses image processing and deep learning to classify emotions such as happiness, sadness, anger, and surprise based on facial features.
2. A Convolutional Neural Network (CNN) is a type of deep learning model particularly effective for analyzing visual data. It automatically detects patterns and features in images, making it ideal for tasks like facial recognition and emotion classification.
3. CNNs process facial images by extracting hierarchical features like edges, shapes, and expressions. These features are passed through fully connected layers to classify emotions based on learned patterns from labeled datasets.

**Current Trends in Facial Emotion Recognition**

* Use in customer satisfaction analysis
* Real-time facial analysis in virtual meetings
* Emotion-aware AI assistants and robots
* Mental health and stress detection tools

**Future of Facial Emotion Detection**

* Integration with multimodal AI systems (text, speech)
* More ethical and privacy-aware facial emotion analysis
* Advanced models like Vision Transformers (ViTs)
* Real-time emotion feedback in AR/VR and gaming

**2. Problem Statement**

**What the Project Does**

The project detects facial expressions from an image and classifies them into one of seven emotional states using a pre-trained Mini-XCEPTION CNN model.

**Objective**

* Identify facial emotions in static images.
* Use a lightweight and fast CNN architecture (Mini-XCEPTION).
* Build a simple, effective emotion recognition pipeline.

**3. Limitations of Existing Systems**

* Dependence on large datasets for training.
* Poor performance in low lighting or occlusion.
* Lack of context-aware emotion interpretation.
* May be biased due to imbalanced training data.

**4. Proposed System**

* Use pre-trained Mini-XCEPTION on FER2013 for robust emotion detection.
* Preprocess input images to match model input size and format.
* Use OpenCV for face detection and image handling.
* Display annotated emotion results visually using Streamlit or Colab.

**5. Pipeline of the System**

**System Flow Chart:**

[Image Input] --> [Face Detection] --> [Preprocessing] --> [CNN Model] --> [Emotion Classification] --> [Output Image with Label]

**Module Flow Chart:**

1. Data Collection
2. Preprocessing
3. EDA
4. Training
5. Testing
6. Implementation
7. Loss Evaluation
8. Coding

**Data Collection**

* Dataset used: **FER2013** (via pre-trained model)

**Preprocessing**

* Convert to grayscale
* Resize image to 64x64
* Normalize pixel values (0-1)
* Expand dimensions for CNN input

**EDA (Exploratory Data Analysis)**

* Understand class distribution in FER2013
* Sample visualization of emotions and expressions

**Training**

* Pre-trained model: Mini-XCEPTION
* Already trained on FER2013 (no re-training in this project)

**Testing**

* Run on test images manually uploaded
* Classify and annotate image output

**Implementation**

* OpenCV for face detection
* Keras to load and use CNN model

**Loss Evaluation**

* Pre-trained model achieved ~66% accuracy on FER2013 test set
* Cross-entropy loss during training (not recalculated here)

**Coding**

* Python (Google Colab or local)
* Core libraries: Keras, OpenCV, NumPy, Streamlit (optional)

**6. Dashboard / User Interface**

* Can be built using **Streamlit**
* Allows image upload, face detection, and emotion labeling
* Visual feedback with bounding boxes and predicted labels

**7. Libraries and Architecture**

**Libraries Used**

* **Keras** – Model loading and inference
* **OpenCV** – Image processing, face detection
* **NumPy** – Image array manipulation
* **Streamlit (optional)** – UI/dashboard

**CNN Architecture :**

**Mini-XCEPTION Architecture Breakdown**

Mini-XCEPTION is composed of **depthwise separable convolutions** and **residual connections**, which reduce computational cost while maintaining performance.

Here is a simplified version of the layers and their purposes:

**1. Input Layer**

* Accepts grayscale images of shape (64, 64, 1).

**2. Convolutional + BatchNorm + ReLU**

* First few layers use standard convolution to extract low-level features like edges and textures.
* Example: Conv2D(32, (3,3)) → BatchNormalization → ReLU

**3. Separable Convolutions**

* Replace regular convolutions with **Depthwise Separable Convolutions** to reduce parameters:
  + Depthwise Conv: Applies a single filter per input channel.
  + Pointwise Conv: Combines the outputs using 1×1 convolution.
* Efficient and retains performance.

**4. Residual Blocks**

* Each block has a **shortcut connection**, inspired by ResNet, that helps gradients flow better.
* Structure:
  + SeparableConv2D → BatchNorm → ReLU → SeparableConv2D → BatchNorm
  + Added with the shortcut input (residual connection).

**5. Global Average Pooling**

* Replaces fully connected layers to reduce overfitting and model size.
* Computes the average of each feature map.

**6. Output Layer**

* A Dense (fully connected) layer with **softmax** activation.
* Size = 7 (for 7 emotion classes: Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral).

**Code:**

!wget https://github.com/oarriaga/face\_classification/raw/master/trained\_models/emotion\_models/fer2013\_mini\_XCEPTION.102-0.66.hdf5 -O emotion\_model.h5

from google.colab import files

uploaded = files.upload()

# 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

# 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)]

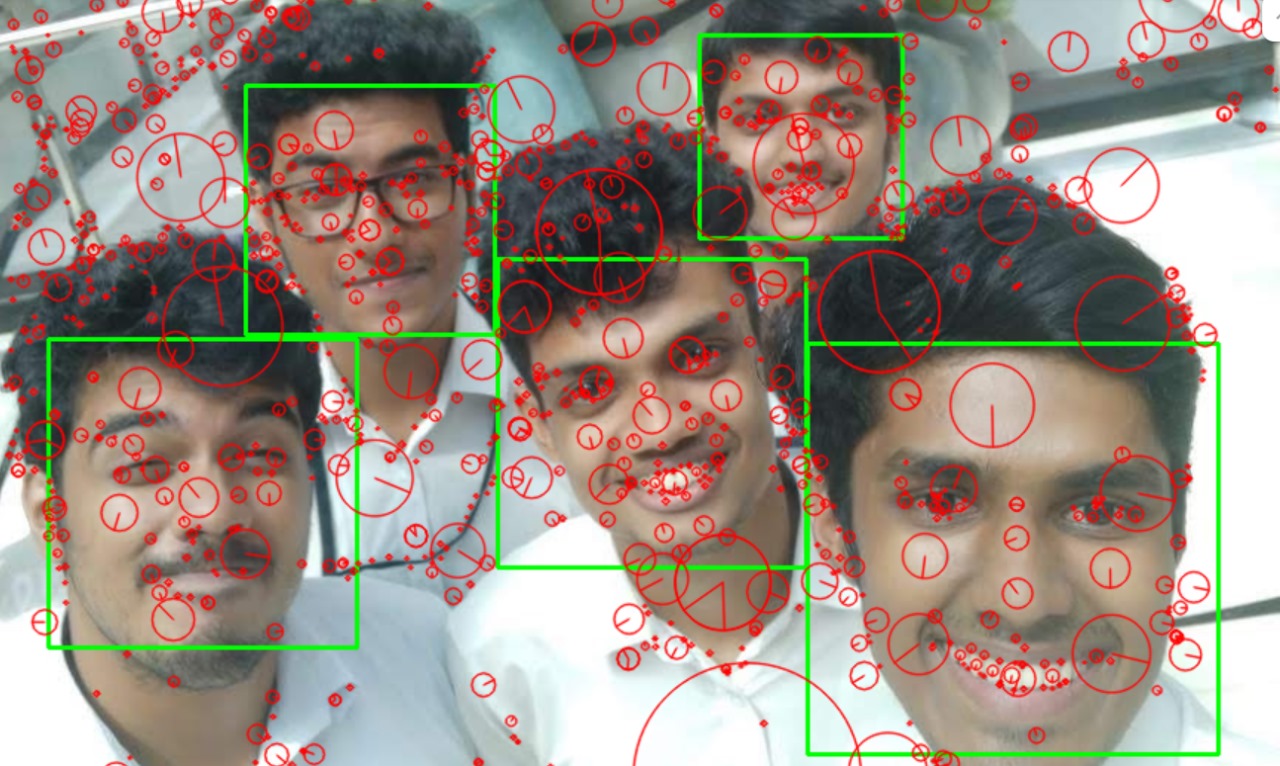
cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

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

# Show result

    cv2\_imshow(img)

**Output**

****

**8. Project Limitations**

* Relies on frontal face detection
* May misclassify subtle or mixed emotions
* Sensitive to lighting, occlusions, and pose variations
* Cannot interpret emotional context (e.g., sarcasm)

**9. Future Enhancements**

* Integrate with audio/text-based emotion recognition (multimodal)
* Replace CNN with Vision Transformers (ViT)
* Use real-time video stream for continuous analysis
* Improve dataset diversity to reduce bias
* Add temporal emotion tracking over sequences

**10. Conclusion**

This project demonstrates effective facial emotion recognition using a lightweight CNN model. While current limitations exist, the system provides a strong foundation for emotion-aware applications, and future improvements can transform it into a multimodal, context-aware emotional AI system.

**Speech Recognition:**

**1.Introduction**

1. Speech recognition is a field of artificial intelligence that enables machines to interpret and convert spoken language into text. It plays a crucial role in natural language processing (NLP) applications such as voice assistants, automated transcription, and interactive voice response systems.
2. A Large Language Model (LLM) is an AI model trained on massive amounts of text data to understand and generate human-like language. Examples include OpenAI's GPT models, Google's PaLM, and Meta's LLaMA. These models are capable of answering questions, summarizing text, translating languages, and understanding context-rich language inputs.
3. While traditional speech recognition systems used acoustic models and HMMs, modern systems utilize LLMs like OpenAI's Whisper. These models not only transcribe speech but also understand context, handle multilingual inputs, and recognize nuances in speech, significantly improving accuracy and adaptability.

**Speech Recognition Uses in Current Trends**

* Virtual assistants (Siri, Alexa, Google Assistant)
* Real-time transcription (Zoom, YouTube captions)
* Call center automation
* Accessibility tools for the hearing impaired
* Emotion-aware chatbots and health monitoring

**Speech Recognition Future**

* Deep integration with multimodal AI systems
* Real-time emotion-aware communication
* Context-aware transcription
* Enhanced performance in noisy environments
* AI tutors and coaches with emotional intelligence

**2. Problem Statement**

**What the Project Does**

This project detects human emotions using two modalities: facial expressions from images and tone/emotion from voice recordings. It combines computer vision, speech recognition (LLM), and audio feature extraction to classify emotions like Happy, Sad, Angry, etc.

**Objective**

* Accurately detect emotions from facial images and speech audio.
* Integrate Whisper LLM for transcription and language detection.
* Create a simple pipeline for emotion classification.
* Develop a dashboard for user interaction.

**3. Limitations of Existing Systems**

* Most systems handle only one modality (either image or audio).
* Traditional audio systems rely on handcrafted features and shallow classifiers (SVM, RF).
* Limited language support and poor noise robustness in basic ASR.
* No integration with modern LLMs for contextual understanding.

**4. Proposed System (Solutions)**

* Use **Whisper (LLM)** for robust and multilingual speech-to-text transcription.
* Use **Mini-XCEPTION CNN** for facial expression recognition.
* Combine voice emotion analysis (MFCC + SVM) with facial emotion analysis.
* Provide an interactive **Streamlit dashboard** for end-users.

**5. Pipeline of the System**

**System Flow Chart:**

[Input: Image or Audio] --> [Preprocessing] --> [Model Prediction] --> [Emotion Output Display]

**Module Flow Chart:**

1. Data Collection
2. Preprocessing
3. EDA (Exploratory Data Analysis)
4. Training
5. Testing
6. Implementation
7. Loss Evaluation
8. Coding (Model integration)

**Data Collection**

* **Facial Emotion**: FER2013 dataset via pre-trained model.
* **Audio Emotion**: Random synthetic labels for simulation; MFCC and pitch extracted.

**Preprocessing**

* Images: Grayscale, resize to 64x64.
* Audio: Convert to mono WAV, extract MFCC and pitch features.

**EDA**

* Visual inspection of MFCC features
* Distribution of facial emotion classes (if labeled)

**Training**

* CNN (Mini-XCEPTION) pre-trained for facial emotions.
* SVM classifier trained on synthetic audio features.

**Testing**

* Run separate image and audio inputs through their pipelines.
* Evaluate accuracy manually.

**Implementation**

* Face detection using OpenCV
* Transcription using Whisper
* Streamlit interface for uploads and outputs

**Loss Evaluation**

* CNN loss evaluated during pretraining
* SVM accuracy based on labeled dataset (mocked in this demo)

**Coding**

* Implemented in Python (Colab environment)
* Libraries: Keras, OpenCV, Whisper, Librosa, scikit-learn, Streamlit

# ----------------------- Fix Dependencies -----------------------

!pip install -q openai-whisper librosa==0.10.0.post2 scikit-learn numpy==1.23.5 pydub ffmpeg-python

# ----------------------- Imports -----------------------

import whisper

import librosa

import numpy as np

import os

from sklearn.preprocessing import LabelEncoder

from sklearn.svm import SVC

from pydub import AudioSegment

from google.colab import files

# ----------------------- Load Whisper Model -----------------------

model = whisper.load\_model("base")

# ----------------------- Transcribe Audio -----------------------

def transcribe\_audio(audio\_path):

    result = model.transcribe(audio\_path)

    print("Transcription:", result["text"])

    return result["text"], result["language"]

# ----------------------- Extract Features -----------------------

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.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])

# ----------------------- 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)

    clf = SVC(kernel='linear')

    clf.fit(X, y\_encoded)

    return clf, le

# ----------------------- Classify Emotion -----------------------

def classify\_emotion(features, clf, le):

    idx = clf.predict([features])[0]

    return le.inverse\_transform([idx])[0]

# ----------------------- Convert to WAV Mono -----------------------

def convert\_to\_wav(input\_path, output\_path):

    sound = AudioSegment.from\_file(input\_path)

    sound = sound.set\_channels(1)

    sound = sound.set\_frame\_rate(16000)

    sound.export(output\_path, format="wav")

# ----------------------- Full Pipeline -----------------------

def emotion\_aware\_speech\_recognition(audio\_path):

    wav\_path = "/content/temp.wav"

    convert\_to\_wav(audio\_path, wav\_path)

    transcription, language = transcribe\_audio(wav\_path)

    features = extract\_audio\_features(wav\_path)

    emotion = classify\_emotion(features, emotion\_classifier, label\_encoder)

    print(f"Detected Emotion: {emotion}")

    print(f"Language Detected: {language}")

    os.remove(wav\_path)

# ----------------------- Train Classifier Once -----------------------

emotion\_classifier, label\_encoder = train\_emotion\_classifier()

# ----------------------- Upload and Process -----------------------

uploaded = files.upload()

for filename in uploaded.keys():

    path = f"/content/{filename}"

    print(f"\nProcessing file: {path}")

    emotion\_aware\_speech\_recognition(path)

**6. Dashboard / User Interface**

* Implemented using **Streamlit**
* Allows user to upload either an image or an audio file
* Displays transcription, detected language, and emotion result
* Visual feedback with image annotations and text output

**7. Libraries and Architecture**

**Libraries Used**

* **Keras** – Load and run CNN for facial recognition
* **OpenCV** – Image loading, face detection, preprocessing
* **Whisper (OpenAI)** – LLM-based speech-to-text
* **Librosa** – Audio processing, MFCC, pitch extraction
* **scikit-learn** – SVM classifier for audio-based emotion
* **Streamlit** – UI/dashboard for interaction

**LLM Architecture (Whisper)**

**1. Encoder: Processes Audio**

**Input:**

* A 30-second audio is converted into a **log-Mel spectrogram** of shape (80, 3000) (frequency × time steps).

**Encoder Layers:**

* **Convolutional Layers**: 2 layers for initial downsampling and local feature extraction.
* **Positional Encoding**: Adds time-position context.
* **Transformer Blocks**: 6–12 stacked layers (depending on model size) with:
  + **Multi-Head Self-Attention**
  + **LayerNorm**
  + **Feedforward Neural Network (FFN)**
  + **Residual Connections**

This produces a sequence of **audio embeddings** that summarize the full context of the input.

**2. Decoder: Language Modeling**

**Purpose:**

* Takes the encoder’s audio embeddings and generates text **token-by-token**.

**Decoder Layers:**

* **Token Embeddings**: For previously generated tokens (auto-regressive input).
* **Positional Embeddings**: To track order of text tokens.
* **Transformer Decoder Blocks**:
  + **Masked Self-Attention**: For autoregressive generation.
  + **Cross-Attention**: Allows the decoder to attend to encoder outputs (audio context).
  + **Feedforward Networks**: Standard dense layers for transformation.
  + **Softmax Output**: Predicts the next token from a vocabulary (~50K tokens).

This part is very similar to GPT-style LLMs.

**8. Project Limitations**

* SVM is not as accurate as deep models like wav2vec for audio emotion.
* Emotion classifier is mocked with synthetic data; real dataset would improve results.
* No integration between facial and audio outputs.
* Not yet real-time; works in batch mode.

**9. Future Enhancements**

* Replace SVM with wav2vec2.0 or SER models
* Collect or integrate real labeled audio emotion datasets (e.g., RAVDESS, IEMOCAP)
* Create unified multimodal fusion of image + audio inputs
* Add context-aware emotion recognition using LLM's text understanding
* Enable real-time emotion feedback system

**10. Conclusion**

This project demonstrates a functional and extendable emotion recognition system using CNNs for facial detection and LLMs (Whisper) for speech processing. It highlights the current capabilities and outlines a clear path toward building future-ready, emotionally intelligent AI systems through multimodal data fusion and LLM integration.