Project Report

FACIAL EXPRESSION RECOGNIZATION

Submitted by

Ayesha Asghar

Muhammad Ali

Supervised by

Sir Mehmood Ali

Table of Contents

1.1 Introduction

Facial expressions are a fundamental aspect of human communication, conveying emotions and intentions often without the need for words. Recognizing and interpreting these expressions is crucial in various fields, from human-computer interaction to psychological studies and security systems. With the advent of artificial intelligence and advancements in computer vision, it has become possible to automate the process of facial expression recognition, leading to the development of systems that can analyze and understand human emotions in real-time.

The **Facial Impression Recognition System** is an AI-driven application designed to detect and classify human facial expressions using cutting-edge technologies in Python, OpenCV, and deep learning. This system leverages the power of deep neural networks (DNNs) to accurately identify emotions such as happiness, sadness, anger, surprise, fear, and neutrality from facial images or video streams.

By combining OpenCV, a robust library for computer vision tasks, with deep learning frameworks like TensorFlow or PyTorch, the system is capable of processing visual data in real-time, making it suitable for applications that require instant feedback. The system not only detects faces within a given frame but also analyzes the facial features to determine the underlying emotion. This functionality opens up possibilities for numerous applications, including enhancing human-computer interactions, enabling emotion-based AI systems, and contributing to fields like behavioral analysis and mental health monitoring.

As we move towards more immersive and personalized technology experiences, the ability to accurately recognize and respond to human emotions becomes increasingly important. The Facial Impression Recognition System represents a step forward in creating intelligent systems that can understand and interact with humans on an emotional level, bridging the gap between human and machine communication.

1.2 Objective

The objective of this code is to create a real-time emotion detection system that captures live video from a webcam, detects human faces within each video frame, and analyzes the detected faces to determine the dominant emotion. The system then displays the identified emotions on the video feed, along with visual indicators (rectangles) around the detected faces. This technology aims to provide insights into the emotional states of individuals in real-time, with potential applications in fields like security, customer service, and entertainment.

1.3 Requirements

Following are the requirements of the project:

HARDWARE

Tools	Description
Computer/ laptop	RAM min 4GB, Prcocessor minimum I 3, Hard drive for saving purpose minimun 128 GB
Webcam	Minimum 4 mega pixels

SOFTWARE

Tools	Description
Window/Mac/Linux	Graphical operating system and development
Python3.14	A high-level programing language.
Visual studio code	Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control.
Libraries	A collection of prewritten code that you can use to perform specific task.

2.0 Project Summary

The goal of this project is to develop a Facial Impression Recognition System that can detect and classify human facial expressions in real-time using Python, OpenCV, and deep learning techniques. The system aims to identify common facial expressions such as happiness, sadness, anger, surprise, fear, and neutral emotions, making it applicable in various fields such as human-computer interaction, behavioral analysis, and emotion-based AI systems.

Facial expression recognition involves detecting and analyzing human emotions through facial cues using computer vision and machine learning techniques. This process typically starts with face detection, followed by extracting and analyzing features from facial expressions. Advanced methods leverage deep learning models, such as Convolutional Neural Networks (CNNs), to accurately classify emotions like happiness, sadness, anger, and surprise. Real-time implementation can be achieved using tools like OpenCV and DeepFace, enabling dynamic emotion detection in live video feeds. The technology has applications in various fields, including security, healthcare, and human-computer interaction.

3.0 Methodologies

1. **Research and Data Collection:**

    a. **Literature Review**: Conduct a review of existing facial expression recognition systems to understand the state of the art.

    b. **Dataset Selection:** Choose a comprehensive dataset (e.g., FER2013) containing labeled images of various facial expressions. Research for the libraries which are execute in this from PYPI , search the pattern from multiple programing website to analyze the syntax of the libraries there implementation like w3 schools etc

2. **Environment Setup:**

    a. **Environment Variables**: Set up a Python virtual environment to manage dependencies and isolate the project. Python -m venv core

    b. **Library Installation:** Install necessary Python libraries such as OpenCV for image processing and DeepFace for emotion analysis using deep learning.

    c. **Example commands**: Using pip install requirement.txt to install libraries for our project.

    pip install opencv-python, pip install deepface.,numpy

3. **Model Training and Testing:**

    a. **Pre-trained Models**: Leverage pre-trained deep learning models for initial facial expression recognition tasks from kaggle plateform.

    b. **Fine-Tuning**: If needed, fine-tune the model on the selected dataset to improve accuracy.

4. **Implementation:**

    a. **Face Detection:** Use OpenCV to detect faces in real-time from a webcam feed.

    b. **Emotion Recognition**: Apply the trained model (DeepFace) to classify detected facial expressions.

    c. **Result Display:** Visualize the detected faces and their corresponding emotions in real-time on the video feed.

5. **Evaluation:**

a. **Performance Assessment:** Evaluate the system's accuracy and response time using test data. Approximately 57.3% accuracy recorded by the train data.

b. **Optimization:** Adjust parameters and configurations to enhance performance if necessary.

This methodology ensures a structured approach to developing a facial expression recognition system, from initial research to real-time implementation and evaluation.

3.1 Development Structure :

1.**Create a Virtual Environment** :

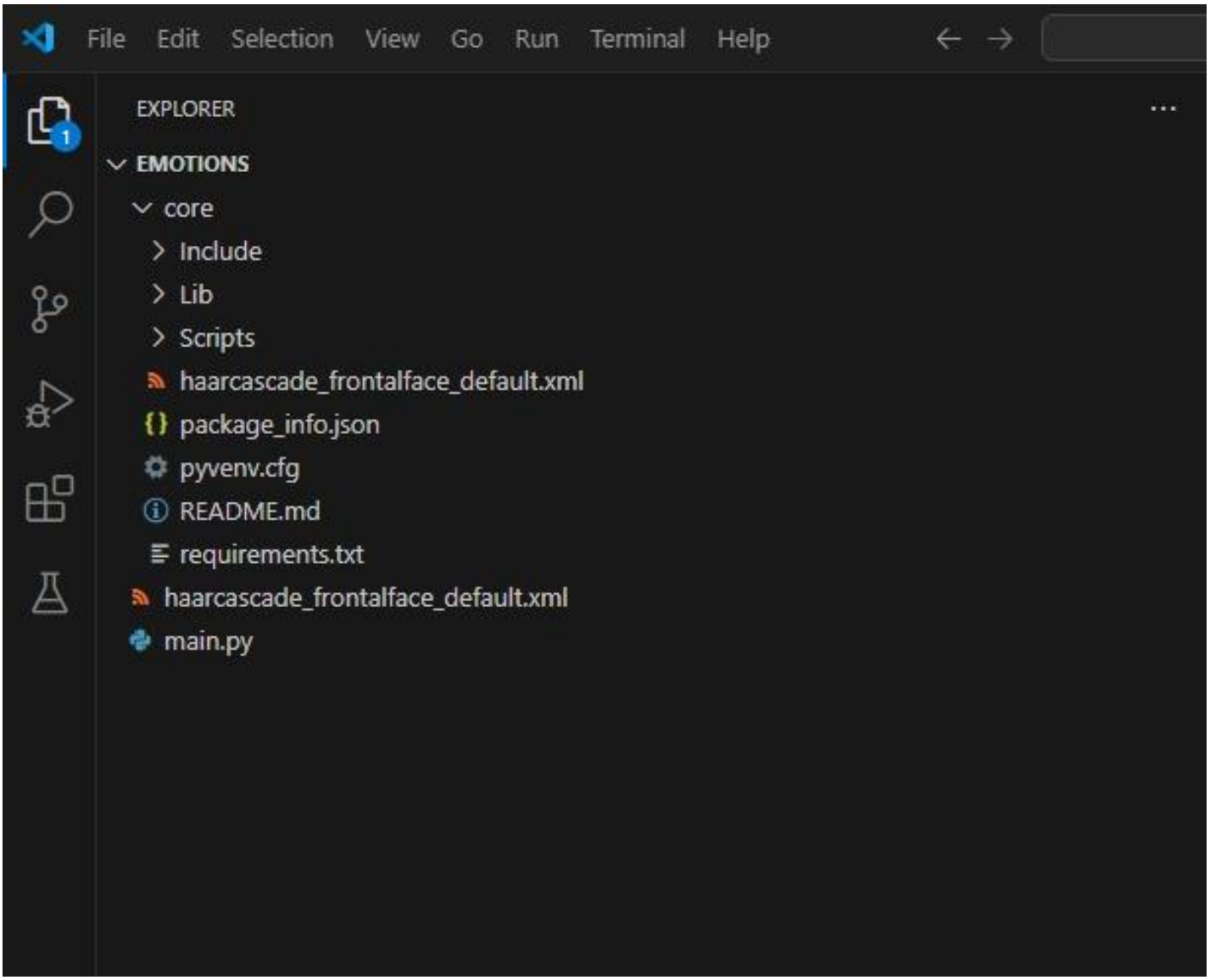

Figure 1final result after creating the virtual environment

We create the virtual environment from the Command Prompt, using the following commands :

a. On Cmd :

   python -m venv myenv

b. Activate the Virtual Environment:

   On Windows:

   bash

myenv\Scripts\activate

c. On macOS/Linux:

Bash

source myenv/bin/activate

2.Install Required Libraries :

Use pip to install necessary libraries. For facial expression recognition, we need the following line of code :

pip install numpy

pip install opencv-python

pip install deepface

3. Setup VS Code

Open VS Code and navigate to your project folder.

Create a new file named main.py.

4. Import Haar Cascade for Face Detection

Download haarcascade_frontalface_default.xml from the OpenCV GitHub repository or use it from the OpenCV package if installed.

In main.py, import the necessary libraries and load the Haar Cascade classifier:

import cv2

from deepface import DeepFace

Load the pre-trained Haar Cascade model for face detection

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

5.Run the Application :

Execute your code in VS Code's terminal :

Python main.py

3.2 Development Algorithm and Code :

main.py haarcascade_frontalface_default.xml

main.py > ...

```python

import cv2
from deepface import DeepFace

# Load the pre-trained face detector model
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml

cap = cv2.VideoCapture(0)  # Start video capture from the webcam

while True:
    ret, frame = cap.read()  # Read a frame from the webcam
    if not ret:
        break
    # Convert the frame to grayscale for face detection
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # Detect faces in the frame
    faces = faceCascade.detectMultiScale(gray, 1.3, 5)
    # Initialize result as None
    result = None

    if len(faces) > 0:
        # Analyze the frame for emotions using DeepFace with enforce_detection set to False
        try:
            result = DeepFace.analyze(frame, actions=['emotion'], enforce_detection=False)
        except ValueError:
            result = None

```

```
27
28          # Ensure result is a list of dictionaries
29          if isinstance(result, list) and len(result) > 0:
30              dominant_emotion = result[0]['dominant_emotion']
31              # Display the dominant emotion on the frame if a face was detected
32              font = cv2.FONT_HERSHEY_SIMPLEX
33              cv2.putText(frame, dominant_emotion, (50, 50), font, 1, (255, 0, 0), 2, cv2.LINE_AA)
34
35      for (x, y, w, h) in faces:
36          # Draw a rectangle around each detected face
37          cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
38
39      # Show the video feed with detected faces and emotions
40      cv2.imshow('Original_Video', frame)
41
42      if cv2.waitKey(2) & 0xFF == ord('a'):
43          break
44
45  cap.release()  # Release the webcam
46  cv2.destroyAllWindows()  # Close all OpenCV windows
47
```

3.3 Output :

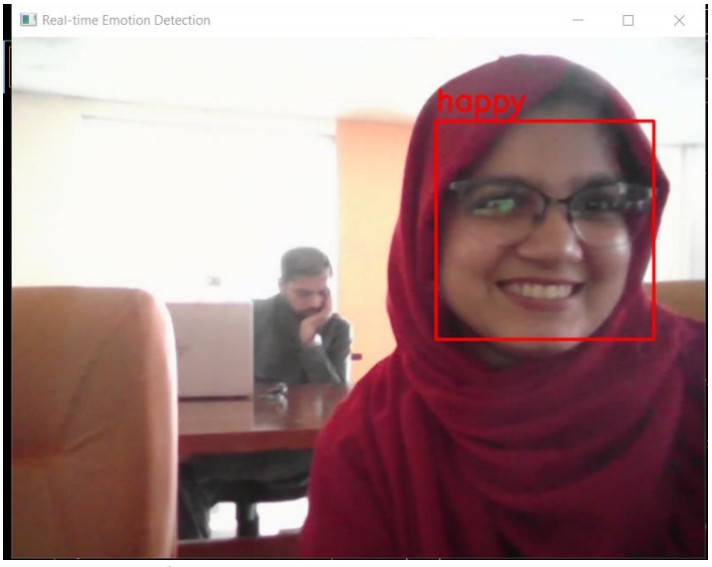

Figure1 _Happy face

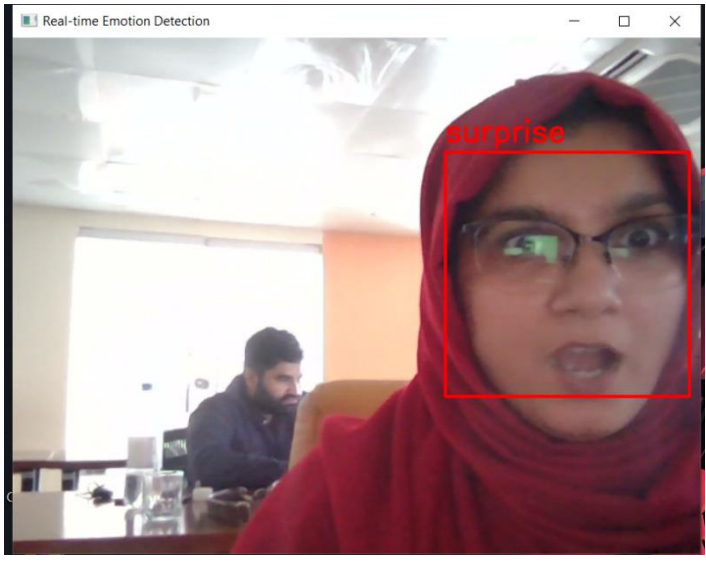

Figure 2Surprise_face

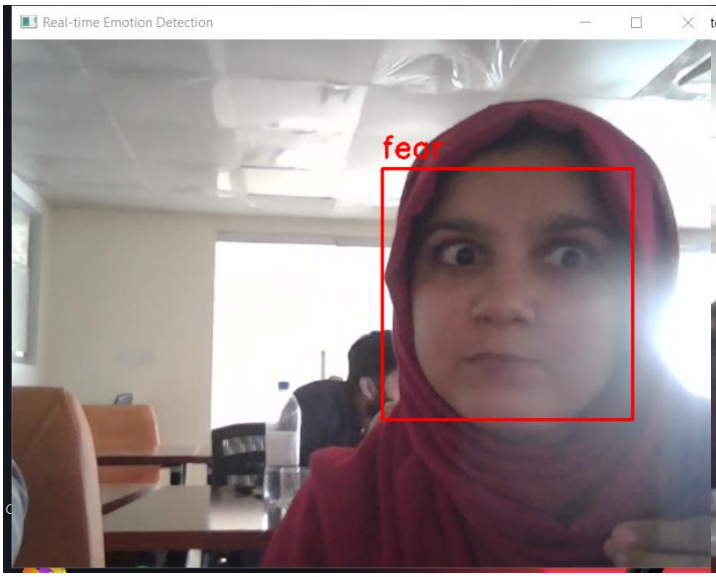

Figure 3Fear_face

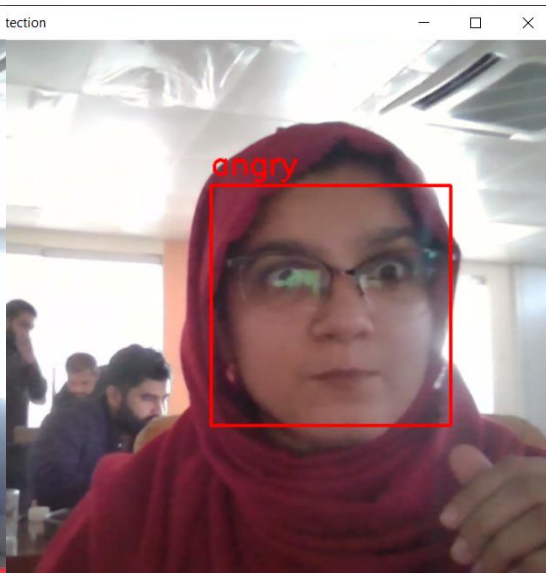

Figure 4angry_face

3.4 Conclusion:

The facial expression recognition detector implemented in Python provides a robust solution for real-time emotion analysis. By leveraging OpenCV for face detection and DeepFace for emotion recognition, this code can effectively identify and display emotions such as anger, sadness, happiness, disgust, surprise, fear, and neutral. This technology has vast applications, from enhancing user experience in applications to aiding in psychological studies. As we continue to explore the intersection of technology and human emotion, tools like this will play a crucial role in bridging the gap between machines and human understanding.

3.5 Recommendations:

Use Deep Learning: Prefer deep learning models like CNNs for higher accuracy in emotion detection.

Data Augmentation: Enhance training data with varied lighting, angles, and expressions for robustness.