**Facial Expression Recognition for Mental Health using YOLO and Flask**

## 

## **Project Overview**

This project implements a **Real-Time Facial Expression Recognition System** using **YOLOv8** to detect facial expressions and emotions such as **anger**, **happiness**, **sadness**, etc. The system is deployed using **Flask** and streams real-time webcam video for emotion recognition. This system can be used for **mental health monitoring** by analyzing emotional states.

## **Technologies Used:**

* **YOLOv8** for face detection and expression recognition.
* **Flask** for the web application backend.
* **OpenCV** for capturing webcam video feed.
* **AffectNet Dataset (YOLO format)** for training the YOLO model.
* **HTML/CSS** for designing the UI.
* **VScode**

## **Setup and Installation:**

## **Installation and Setup (No Repositories Used)**

Follow these steps to set up the **Facial Expression Recognition** project **locally** on your system:

## **1. Project Folder Structure:**

Create a folder named Facial-Expression-Recognition and organize it as follows:

Facial-Expression-Recognition/

│

├── app.py # Flask main file for video streaming

├── trail\_yolo.py # YOLO expression detection logic

├── Detect\_emotions.py # Emotion detection and logging

├── report\_emotions.py # Emotion report generation

├── camera\_test.py # Webcam testing script

├── weights/

│ └── best.pt # Your trained YOLO model weights

│

├── templates/

│ └── index.html # Web UI for video display

│

└── requirements.txt # All Python dependencies

## **2. Install Python**

Ensure **Python 3.10+** is installed on your machine.

You can verify with: python --version

If not installed, download it from  [https://www.python.org](https://www.python.org/downloads/).

## 

## **3. Create a Virtual Environment:**

python -m venv venv

venv\Scripts\activate

## **4. Install Required Libraries:**

Use the requirements.txt file (you will create this) to install dependencies:

requirements.txt (create this file)

Flask

opencv-python

ultralytics

numpy

## **5. Add YOLOv8 Model:**

Put your best.pt YOLO model file into the weights/ folder.

**NOTE:** This file should be the one you trained using the AffectNet dataset in YOLO format.

## **6. Run the Project:**

In your terminal or command prompt, navigate to the project folder and run:

python app.py:

## **7. Open in Browser:**

Once Flask starts, open a browser and go to:

http://127.0.0.1:5000/

You'll see the **Facial Expression Recognition UI** and the **live webcam feed** with emotion predictions displayed on it.

### 

### **Code Implementation**

#### **Backend: app.py**

This is the main application file, which sets up a Flask server and handles the streaming of video feed from the webcam, processes it through YOLO, and serves it to the web interface.

from flask import Flask, render\_template, Response

import cv2

from ultralytics import YOLO

app = Flask(\_\_name\_\_)

# Load the YOLO model (ensure the model path is correct)

model = YOLO('weights/best.pt')

# Update this with your model path print ("Model loaded successfully!")

# Open the camera (webcam)

camera = cv2.VideoCapture(0)

camera.set(cv2.CAP\_PROP\_FRAME\_WIDTH, 640)

camera.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, 480)

if not camera.isOpened():

print("Error: Could not open camera.")

exit()

def gen\_frames():

while True:

success, frame = camera.read()

if not success:

print("Failed to grab frame")

continue

results = model(frame)

annotated\_frame = results[0].plot()

if annotated\_frame is None or annotated\_frame.size == 0:

print("Invalid annotated frame")

continue

ret, buffer = cv2.imencode('.jpg', annotated\_frame)

if not ret:

print(" Failed to encode frame")

continue

frame\_bytes = buffer.tobytes()

yield (b'--frame\r\n'

b'Content-Type: image/jpeg\r\n\r\n' + frame\_bytes + b'\r\n')

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/video')

def video():

return Response(gen\_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

#### **Frontend: index.html**

This is the **UI for displaying the webcam video feed** and the real-time detected expressions.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Real-Time Facial Expression Recognition</title>

<style>

/\* Global Styles \*/

body {

font-family: 'Arial', sans-serif;

background-color: #121212;

color: white;

margin: 0;

padding: 0;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

height: 100vh;

}

h1 {

font-size: 3rem;

text-align: center;

margin-bottom: 20px;

font-weight: 700;

letter-spacing: 1px;

text-transform: uppercase;

background: linear-gradient(45deg, #ff6b6b, #f7b42c, #febc63);

background-clip: text;

color: transparent;

animation: glow 1.5s ease-in-out infinite alternate;

}

@keyframes glow {

0% {

text-shadow: 0 0 5px #ff6b6b, 0 0 10px #ff6b6b, 0 0 15px #f7b42c, 0 0 20px #febc63;

}

100% {

text-shadow: 0 0 10px #ff6b6b, 0 0 20px #f7b42c, 0 0 30px #febc63;

}

}

/\* Style for the video feed \*/

.video-container {

background-color: rgba(0, 0, 0, 0.8);

border-radius: 10px;

padding: 10px;

border: 3px solid #f7b42c;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.7);

transition: transform 0.3s ease-in-out;

}

.video-container:hover {

transform: scale(1.05);

}

/\* Style for the video stream \*/

img {

border-radius: 8px;

box-shadow: 0 0 15px rgba(255, 255, 255, 0.3);

width: 640px;

height: 480px;

transition: filter 0.3s ease-in-out;

}

img:hover {

filter: brightness(1.2);

}

/\* Footer \*/

footer {

margin-top: 30px;

font-size: 1rem;

color: #f7b42c;

}

footer a {

color: #f7b42c;

text-decoration: none;

font-weight: bold;

}

</style>

</head>

<body>

<h1>Facial Expression Recognition</h1>

<div class="video-container">

<img src="{{ url\_for('video') }}" alt="Video Stream">

</div>

<footer>

<p>Powered by <a href="https://ultralytics.com" target="\_blank">Ultralytics YOLO</a></p>

</footer>

</body>

</html>

### **Additional Python Scripts**

These additional scripts are part of your project and contribute to the system's overall functionality.

### **train\_yolo.py:**

This script is responsible for setting up the YOLO model, running the object detection, and annotating the detected objects (i.e., the face and facial expression).

### **code:**

from ultralytics import YOLO

model = YOLO('yolov8n.pt')

model.train(

data=(r'C:\Users\lohith\OneDrive\Desktop\major\YOLO\_format\data.yaml',)

# Provide the correct path to your data.yaml

imgsz=640, # Image size for training (640x640)

epochs=8, # Number of epochs for training

batch=16, # Batch size for training

name='affectnet\_emotion' # Name for the training run

### **detect\_emotions.py:**

This script processes the webcam frame, detects the facial expression using YOLO, and then logs the detected emotion to a file or database for tracking.

### **code:**

from ultralytics import YOLO

import cv2

from datetime import datetime

model = YOLO(r"C:\Users\lohith\runs\detect\affectnet\_emotion3\weights\best.pt")

class\_names = ['Anger', 'Contempt', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

cap = cv2.VideoCapture(0)

log\_file = open('emotion\_log.csv', 'a')

log\_file.write('timestamp,emotion\n')

while True:

ret, frame = cap.read()

if not ret:

break

results = model(frame)[0]

annotated\_frame = results.plot()

cv2.imshow("Emotion Detection", annotated\_frame)

if results.boxes:

top\_box = results.boxes[0]

class\_id = int(top\_box.cls[0])

emotion = class\_names[class\_id]

timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

log\_file.write(f'{timestamp},{emotion}\n')

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

if cv2.waitKey(1) & 0xFF == ord('q'):

break

log\_file.close()

cap.release()

cv2.destroyAllWindows()

### **report\_emotions.py:**

This script generates a report based on the emotions detected by the system over time.

### **code:**

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv('emotion\_log.csv')

summary = df['emotion'].value\_counts()

print("Emotion Summary:")

print(summary)

summary.plot(kind='bar', color='skyblue', title='Detected Emotions Frequency')

plt.xlabel('Emotion')

plt.ylabel('Frequency')

plt.tight\_layout()

plt.savefig('emotion\_summary.png')

plt.show()

### **camera\_test.py:**

This script is used for testing the webcam feed to ensure it is working correctly before integrating with the rest of the system.

code:

import cv2

import os

camera = cv2.VideoCapture(0, cv2.CAP\_DSHOW)

if not camera.isOpened():

print("Error: Could not open video stream.")

exit()

camera.set(cv2.CAP\_PROP\_FRAME\_WIDTH, 640)

camera.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, 480)

frame\_counter = 0

while True:

success, frame = camera.read()

if not success:

print("Failed to grab frame")

break

else:

if frame is None or frame.size == 0:

print("Error: Empty frame received")

break

cv2.imshow('Camera Feed', frame)

if frame\_counter == 0:

if not os.path.exists('captured\_frames'):

os.mkdir('captured\_frames')

cv2.imwrite('captured\_frames/first\_frame.jpg', frame)

print("First frame saved as 'captured\_frames/first\_frame.jpg'")

frame\_counter += 1

if cv2.waitKey(1) & 0xFF == ord('q'):

break

camera.release()

cv2.destroyAllWindows()

## **Conclusion**

This system provides a **real-time facial expression recognition system** that uses **YOLO** for face detection and expression classification, helping in **mental health monitoring** by analyzing emotional states. The use of **Flask** enables efficient video streaming, and the **HTML/CSS interface** offers an easy-to-use interactive front-end.

## **References**

* [AffectNet Dataset](https://www.affectnet.com/)
* [YOLOv8](https://ultralytics.com/)