🧠 Real-Time Emotion Detection with OpenCV + DeepFace

📌 Project Overview

This Python script uses your webcam to:

- Detect faces in real time using OpenCV’s Haar cascades
- Analyze emotions using DeepFace
- Display the dominant emotion on the video feed

🧩 Dependencies

Make sure you’ve installed:

bash

```
pip install deepface opencv-python
```

1.**IMPORT LIBRARIES**

```
import cv2

from deepface import DeepFace
```

2.INITIALIZE WEBCAM

```
cap = cv2.VideoCapture(0)
```

3. . Load Haar Cascade for Face Detection

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

4.MAIN LOOP

```
while True:

    ret, frame = cap.read()

    if not ret or frame is None:

        print("Failed to capture frame. Check your webcam.")

        continue
```

5. Convert to Grayscale & Detect Faces

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
```

6.ANALYZE EACH FACE

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

    face_roi = frame[y:y+h, x:x+w]

    try:

        result = DeepFace.analyze(face_roi, actions=['emotion'], enforce_detection=False)

        emotion = result[0]['dominant_emotion']
```

7.DRAW RECTANGLE AND LABLE

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

        cv2.putText(frame, emotion, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 2)

    except Exception as e:

        print("Error:", e)
```

8.DISPLAY FRAME

```
cv2.imshow("Real-Time Emotion Detection", frame)

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

    break
```

9.CLEANUP

```
cap.release()

cv2.destroyAllWindows()
```

🧠 How DeepFace Works

- Deepface.analyze() returns a dictionary with keys like emotion ,age ,gender ,race .
- We use action= [‘emotion’]to focus only on emotion detection.
- enforce_detection=False prevents crashes if no face is found.

🧪 Suggested Exercises

- Add support for age and gender detection.
- Log emotions over time to a CSV file.
- Replace Haar cascade with a DNN-based face detector.
- Optimize performance by analyzing every N frames.

📊 Flowchart of Project Execution

```
graph TD
    A[Start Program] --> B[Initialize Webcam]
    B --> C[Load Haar Cascade Face Detector]
    C --> D[Capture Frame from Webcam]
    D --> E[Convert Frame to Grayscale]
    E --> F[Detect Faces]
    F --> G[For Each Face: Extract ROI]
    G --> H[Resize ROI to 224x224]
    H --> I[Analyze Emotion with DeepFace]
    I --> J[Draw Rectangle and Label on Frame]
    J --> K[Display Frame in Window]
    K --> L{Press 'q'?}
    L -- No --> D
    L -- Yes --> M[Release Webcam & Close Window]
```

CODE EXPLANATION

```
# 📦 Import necessary libraries
import cv2                    # OpenCV for image processing and webcam access
from deepface import DeepFace     # DeepFace for emotion analysis
```

```
# 🎥 Initialize webcam (device 0 is usually the default camera)
cap = cv2.VideoCapture(0)

# 🧠 Load Haar Cascade face detector from OpenCV's built-in models
face_cascade = cv2.CascadeClassifier(
    cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
)

# 🛡️ Wrap main loop in try-except for graceful exit
try:
    while True:
        # 📸 Capture a frame from the webcam
        ret, frame = cap.read()
        if not ret or frame is None:
            print("Failed to capture frame. Check your webcam.")
            continue

        # ⚫ Convert the frame to grayscale for face detection
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # 🔍 Detect faces in the grayscale image
        faces = face_cascade.detectMultiScale(
            gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30)
        )

        # 🧠 Loop through each detected face
        for (x, y, w, h) in faces:
```

```
face_roi = frame[y:y+h, x:x+w]  # Extract region of interest (ROI)

try:
    # ⚡ Resize ROI to 224x224 for DeepFace model input
    if face_roi.size == 0:
        continue
    small_roi = cv2.resize(face_roi, (224, 224))

    # 🧠 Analyze emotion using DeepFace
    result = DeepFace.analyze(
        small_roi, actions=['emotion'], enforce_detection=False
    )

    # 🧾 Handle different return formats (dict or list of dicts)
    if isinstance(result, list) and len(result) > 0:
        data = result[0]
    elif isinstance(result, dict):
        data = result
    else:
        data = {}

    # 😊 Extract dominant emotion
    emotion = data.get('dominant_emotion', 'Unknown')

    # 🖼️ Draw rectangle around face and label with emotion
    cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
    cv2.putText(
        frame, emotion, (x, y-10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 2
                )

            except Exception as e:
                # 🛠️ Log error and continue processing other faces
                print("DeepFace error:", e)

        # 🖥️ Show the annotated frame in a window
        cv2.imshow("Real-Time Emotion Detection", frame)

        # ⌨️ Exit loop if 'q' is pressed
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

# 🛑 Handle Ctrl+C interruption gracefully
except KeyboardInterrupt:
    print('\nInterrupted by user')

# 🧹 Release resources and close window
finally:
    cap.release()
    cv2.destroyAllWindows()
```