Natural Language Processing (NLP)

Experimental Learning

18AMC306T

Department of Artificial Intelligence and Machine Learning

Year : III Semester : VI

Done by

R KANNAN

927622BAL019

AIML

Source Code:

```
import cv2
import numpy as np
from deepface import DeepFace
import time
def main():
    cap = cv2.VideoCapture(0)
    if not cap.isOpened():
        return
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT,  480)
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
    emotion_colors = {
        "happy": (0, 255, 0),
        "sad": (0, 0, 255),
        "angry": (0, 0, 255),
        "fear": (255, 0, 255),
        "disgust": (0, 128, 128),
        "surprise": (255, 165, 0),
        "neutral": (255, 255, 0)
    }
    last_detection_time = time.time()
    detection_interval = 0.5
    last_emotion = "happy”
    emotion_confidence = 0.0
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        display_frame = frame.copy()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.1, 4)
```

```
        current_time = time.time()
        if len(faces) > 0 and (current_time - last_detection_time) >= detection_interval:
            try:
                result = DeepFace.analyze(
                    img_path=frame,
                    actions=['emotion'],
                    enforce_detection=False,
                    silent=True
                )
                emotions = result[0]['emotion']
                highest_score = -1
                dominant_emotion = None
                for emotion, score in emotions.items():
                    if score > highest_score:
                        highest_score = score
                        dominant_emotion = emotion
                last_emotion = dominant_emotion
                emotion_confidence = highest_score
                last_detection_time = current_time
            except:
                pass
        for (x, y, w, h) in faces:
            cv2.rectangle(display_frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
            emotion_color = emotion_colors.get(last_emotion, (255, 255, 255))
            cv2.putText(display_frame, f"{last_emotion} ({emotion_confidence:.2f})", (x, y-10),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, emotion_color, 2)
        cv2.putText(display_frame, f"Detected emotion: {last_emotion}", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        cv2.putText(display_frame, f"Confidence: {emotion_confidence:.2f}", (10, 60),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        legend_y = 90
        for i, (emotion, color) in enumerate(emotion_colors.items()):
            cv2.putText(display_frame, emotion, (10, legend_y + i*25), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
color, 1)
```

```
        cv2.putText(display_frame, "Press 'q' to exit", (10, legend_y + len(emotion_colors)*25 + 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

        cv2.imshow('Facial Expression Detection', display_frame)

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

            break

    cap.release()

    cv2.destroyAllWindows()

if __name___== "__main__":

    main()
```

Output: