Emotion Recognition using OpenCV

Code Explanation

import os

import cv2

import numpy as np

from tensorflow.keras.preprocessing import image

import warnings

warnings.filterwarnings("ignore")

from tensorflow.keras.utils import img_to_array,load_img

from keras.models import load_model

import matplotlib.pyplot as plt

import numpy as np

Explanation:

- The import statements at the beginning of the script import necessary Python libraries and modules.
- os module provides a portable way of using operating system dependent functionality such as reading or writing to the file system.
- cv2 is a Python module for computer vision and image processing.
- numpy is a library for numerical computing in Python.
- tensorflow.keras.preprocessing provides a set of functions for data preprocessing in Keras.
- warnings module provides a mechanism to control warnings that are emitted by Python interpreter.
- tensorflow.keras.utils provides utility functions for Keras.
- load_img() and img_to_array() functions are part of tensorflow.keras.utils and are used to load an image file and convert it into a NumPy array.
- load_model() function is used to load a pre-trained Keras model from a file
- matplotlib.pyplot is a plotting library for Python.

model = load_model("best_model.h5")

Explanation: This line loads a pre-trained deep learning model from a file called best_model.h5.

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

Explanation: This line initializes a Haar Cascade classifier for detecting faces in an image using OpenCV.

cap = cv2.VideoCapture(0)

Explanation: This line captures a video stream from the default camera (with device ID 0) using the OpenCV cv2.VideoCapture() function.

while True:

_, test_img = cap.read()

gray_img = cv2.cvtColor(test_img, cv2.COLOR_BGR2RGB)

Explanation:

- This starts an infinite loop that reads frames from the video stream until the loop is interrupted.
- The _ variable is used to capture the return value of the cap.read() function, which is a boolean value indicating whether a frame was successfully read from the video stream.
- The cv2.cvtColor() function is used to convert the captured image from the BGR color space to the RGB color space.

faces_detected = face_haar_cascade.detectMultiScale(gray_img, 1.3, 5)

Explanation:

- This line detects faces in the current frame using the Haar Cascade classifier initialized earlier.
- The detectMultiScale() function takes as input a grayscale image, a scaling factor, and a minimum number of neighbor rectangles that need to be accepted to retain a detected region.

for (x, y, w, h) in faces_detected:

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

roi_gray = gray_img[y:y + w, x:x + h]

roi_gray = cv2.resize(roi_gray, (224, 224))

img_pixels = image.img_to_array(roi_gray)

img_pixels = np.expand_dims(img_pixels, axis=0)

img_pixels /= 255

predictions = model.predict(img_pixels)

max_index = np.argmax(predictions[0])

emotions = ('angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'suprise')

predicted_emotion = emotions[max_index]

cv2.putText(test_img, predicted_emotion, (int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

Explanation:

- for (x, y, w, h) in faces_detected:: This line loops over all the faces detected in the input image. The faces_detected variable contains the coordinates of the bounding boxes around the detected faces.
- cv2.rectangle(test_img, (x, y), (x + w, y + h), (255, 0, 0), 2): This line draws a rectangle around the detected face. The test_img variable contains the original input image.
- roi_gray = gray_img[y:y + w, x:x + h]: This line extracts the region of interest (ROI) from the grayscale input image. The ROI corresponds to the face region detected by the face detector.
- roi_gray = cv2.resize(roi_gray, (224, 224)): This line resizes the ROI to a fixed size of 224x224 pixels. This is the input size required by the model.
- img_pixels = image.img_to_array(roi_gray): This line converts the resized ROI to a numpy array.
- img_pixels = np.expand_dims(img_pixels, axis=0): This line adds an extra dimension to the numpy array. This is required by the model as it expects a batch of images as input
- img_pixels /= 255: This line normalizes the input image by dividing each pixel value by 255.
- predictions = model.predict(img_pixels): This line runs the input image through the model to generate predictions for each of the possible emotions.
- max_index = np.argmax(predictions[0]): This line finds the index of the emotion with the highest probability.
- emotions = ('angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'suprise'): This line defines a tuple of all the possible emotions.
- predicted_emotion = emotions[max_index]: This line finds the predicted emotion based on the index of the highest probability.
- cv2.putText(test_img, predicted_emotion, (int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2): This line writes the predicted emotion on the input image. The text is written at the top-left corner of the bounding box around the detected face.

resized_img = cv2.resize(test_img, (1000, 700))

cv2.imshow('Facial emotion analysis ', resized_img)

if cv2.waitKey(10) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

Explanation:

- resized_img = cv2.resize(test_img, (1000, 700)): This line resizes the output image to 1000x700 pixels. This is done to ensure that the image fits in the display window and is not too small to see. You can change the size of the output image based on your requirements.
- cv2.imshow('Facial emotion analysis ', resized_img): This line displays the output image in a window titled "Facial emotion analysis". The resized image is passed as an argument to the function.
- if cv2.waitKey(10) & 0xFF == ord('q'):: This line waits for a key press event for 10 milliseconds. If the key pressed is 'q', the program exits. This is done to ensure that the program exits gracefully when the user is done using it.
- cap.release(): This line releases the camera resource when the program is done running. This is done to ensure that the camera is available for other programs to use.
- cv2.destroyAllWindows(): This line closes all windows that were created by the program. This is done to ensure that there are no lingering windows after the program is done running.