- **age**: Rounds the predicted age.
- **cv2.putText**: Displays the age on the frame.

Display and Exit

```
cv2.imshow('Gender and Age Detector', frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break

cap.release()
cv2.destroyAllWindows()
```

- **cv2.imshow**: Displays the frame with annotations.
- **cv2.waitKey(1)**: Waits for 1 millisecond for a key press. If ‘q’ is pressed, the loop breaks.
- **cap.release()**: Releases the video capture.
- **cv2.destroyAllWindows()**: Closes all OpenCV windows.

Feel free to ask if you have any questions or need further clarification on any part of the code!

You

explain in more detail

Copilot

Absolutely, let’s dive deeper into each part of your code to understand it thoroughly.

Importing Libraries

```
from keras.models import load_model
from keras.losses import mean_squared_error
from time import sleep
from keras.preprocessing.image import img_to_array
from keras.preprocessing import image
import cv2
import numpy as np
```

- **Keras**: A high-level neural networks API, written in Python and capable of running on top of TensorFlow. Here, it’s used to load pre-trained models and handle image preprocessing.
    - `load_model`: Loads a pre-trained Keras model.
    - `mean_squared_error`: A loss function used in the models.
- **OpenCV (cv2)**: An open-source computer vision and machine learning software library. It’s used here for real-time image processing.
    - `CascadeClassifier`: Used for face detection.
    - `VideoCapture`: Captures video from the camera.
    - `cvtColor`: Converts images between color spaces.
    - `rectangle`, `putText`, `imshow`, `waitKey`, `destroyAllWindows`: Various functions for image processing and display.

- **NumPy**: A library for numerical operations on arrays.
    - `np.array`: Creates an array.
    - `reshape`: Reshapes an array.

Loading Models and Classifiers

```
face_classifier = cv2.CascadeClassifier(r'c:\Python312\Lib\site-packages\cv2\data\haarcascade_frontalface_alt.xml')
age_model = load_model(r'c:\Users\prithviraj\Downloads\age_model_3epochs.h5', custom_objects={'mse': mean_squared_error})
gender_model = load_model(r'c:\Users\prithviraj\Downloads\gender_model_3epochs.h5', custom_objects={'mse': mean_squared_error})
```

- **face_classifier**: Loads a Haar Cascade classifier for face detection. Haar Cascades are machine learning object detection methods used to identify objects in images or video.
- **age_model** and **gender_model**: Load pre-trained models for predicting age and gender. These models are trained on datasets to recognize patterns in images and make predictions.

Gender Labels

```
gender_labels = ['Male', 'Female', 'Transgender']
```

- Defines the possible labels for gender classification. The model's output will be mapped to one of these labels.

Capturing Video

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

- **cap**: Initializes video capture from the default camera (index 0). This object will be used to read frames from the camera.

Main Loop

```
while True:
    ret, frame = cap.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray, 1.3, 5)
```

- **while True**: An infinite loop to continuously capture and process frames from the camera.
- **ret, frame**: Reads a frame from the video capture. `ret` is a boolean indicating if the frame was read correctly, and `frame` is the captured image.
- **gray**: Converts the captured frame to grayscale. Face detection often works better on grayscale images.
- **faces**: Detects faces in the grayscale image using the Haar Cascade classifier. The parameters `1.3` and `5` are the scale factor and minimum neighbors, respectively, which control the accuracy and speed of detection.

Processing Each Detected Face

```
for (x, y, w, h) in faces:
    cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)
    roi_gray = gray[y:y + h, x:x + w]
    roi_gray = cv2.resize(roi_gray, (48, 48), interpolation=cv2.INTER_AREA)
```

- **for (x, y, w, h) in faces**: Iterates over each detected face. `x` and `y` are the coordinates of the top-left corner, and `w` and `h` are the width and height of the bounding box.
- **cv2.rectangle**: Draws a rectangle around the detected face in the original frame.
- **roi_gray**: Extracts the region of interest (ROI) in grayscale, which is the detected face.
- **cv2.resize**: Resizes the ROI to 48x48 pixels. This size is often used for face recognition models.

Gender Prediction

```
roi_color = frame[y:y + h, x:x + w]
roi_color = cv2.resize(roi_color, (200, 200), interpolation=cv2.INTER_AREA)
gender_predict = gender_model.predict(np.array(roi_color).reshape(-1, 200, 200, 3))
gender_predict = (gender_predict >= 0.5).astype(int)[:, 0]
gender_label = gender_labels[gender_predict[0]]
gender_label_position = (x, y + h + 50)
cv2.putText(frame, gender_label, gender_label_position, cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
```

- **roi_color**: Extracts the ROI in color and resizes it to 200x200 pixels. This size is used by the gender prediction model.
- **gender_predict**: Predicts the gender using the gender model. The model outputs a probability, which is thresholded at 0.5 to get a binary prediction.
- **gender_label**: Maps the binary prediction to a label (‘Male’, ‘Female’, ‘Transgender’).
- **cv2.putText**: Displays the gender label on the frame at the specified position.

Age Prediction

```
age_predict = age_model.predict(np.array(roi_color).reshape(-1, 200, 200, 3))
age = round(age_predict[0, 0])
age_label_position = (x + h, y + h)
cv2.putText(frame, "Age=" + str(age), age_label_position, cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
```

- **age_predict**: Predicts the age using the age model. The model outputs a continuous value, which is rounded to get an integer age.
- **age**: Rounds the predicted age to the nearest integer.
- **cv2.putText**: Displays the age on the frame at the specified position.

Display and Exit

```
cv2.imshow('Gender and Age Detector', frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break

cap.release()
cv2.destroyAllWindows()
```

- **cv2.imshow**: Displays the frame with annotations in a window titled ‘Gender and Age Detector’.
- **cv2.waitKey(1)**: Waits for 1 millisecond for a key press. If the ‘q’ key is pressed, the loop breaks.

- **cap.release()**: Releases the video capture object.
- **cv2.destroyAllWindows()**: Closes all OpenCV windows.

This detailed explanation should help you understand each part of your code better. If you have any more questions or need further clarification, feel free to ask!