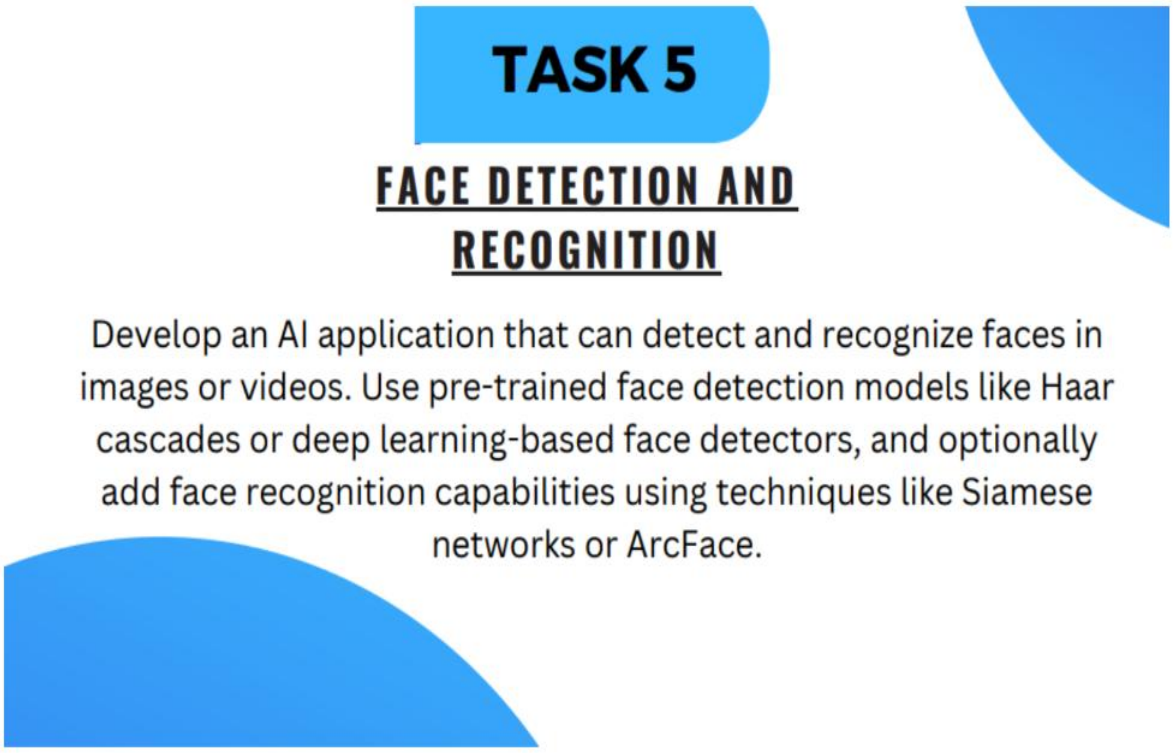
TASK-5

Face Detection and Recognition



TASK 5

FACE DETECTION AND RECOGNITION

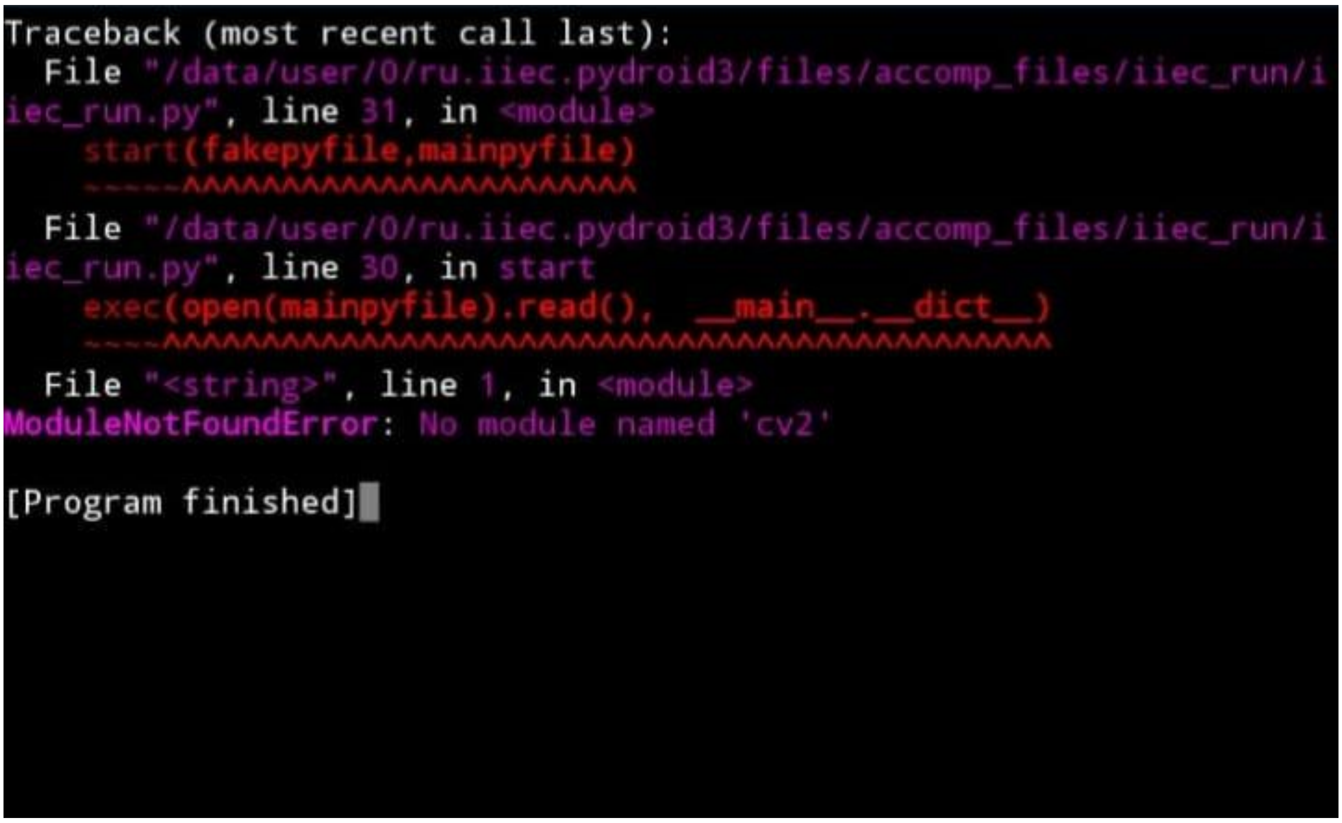
Develop an AI application that can detect and recognize faces in images or videos. Use pre-trained face detection models like Haar cascades or deep learning-based face detectors, and optionally add face recognition capabilities using techniques like Siamese networks or ArcFace.

CODE:

Input:

```
import cv2
import face recognition
import o s
import numpy as np

# Load known faces
Known faces dir = "known faces"
K nowne ncodings = []
```

```
Known names = []

print("Loading known faces...")
for name in os. Listdir (known faces dir):
    person dir = os. path. Join (known faces dir, name)
    for filename in os. Listdir (person dir):
        image path = os .path. join(person dir, filename)
        image = face recognition. load image file (image path)
        encodings = face recognition. Face encodings(image)
        if encodings:
            known encodings. Append (encodings [0])
            known names. Append (name)

# Start webcam
Video capture = cv2.VideoCapture(0)
Print ("Starting video stream...")

while True:
    ret, frame = video capture. Read ()
    if not ret:
        break

    small frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
    rgb small frame = small frame[:, :, ::-1]

    face locations = face recognition. face locations (rgb small frame)
    face encodings = face recognition. face encodings (rgb small frame, face
locations)

    for face encoding, face location in zip (face encodings, face locations):
        matches = face recognition. compare faces (known encodings, face
encoding)
        name = "Unknown"

        face distances = face recognition. Face distance (known encodings, face
encoding)
        best match index = np. Arg min (face distances)
        if matches[best match index]:
            name = known names[best match index]

        top, right, bottom, left = [v * 4 for v in face location]
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)
        cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 255, 0),
cv2.FILLED)
        cv2.putText(frame, name, (left + 6, bottom - 6),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 0), 2)

    cv2.imshow("Face Detection & Recognition", frame)

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

video_capture.release()
cv2.destroyAllWindows()
```



```
Traceback (most recent call last):
  File "/data/user/0/ru.iiec.pydroid3/files/accomp_files/iiec_run/i
iec_run.py", line 31, in <module>
    start(fakepyfile,mainpyfile)
    ~~~~~^^^^^^^^^^^^^^^^^^^^^^^^
  File "/data/user/0/ru.iiec.pydroid3/files/accomp_files/iiec_run/i
iec_run.py", line 30, in start
    exec(open(mainpyfile).read(),  __main__.__dict__)
    ~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "<string>", line 1, in <module>
ModuleNotFoundError: No module named 'cv2'

[Program finished]
```

code explanation :

Imports

import cv2

import face_recognition

import os

import numpy as np

- cv2: OpenCV library used for video capture and drawing on frames.
- face_recognition: Library built on top of dlib for face detection and recognition.
- os: For navigating directories to load known faces.
- numpy: For array operations, especially calculating distances between face encodings.

```
known_faces_dir = "known_faces"

known_encodings = []

known_names = []
```

Loading Known Faces

- known_faces_dir: The directory that stores images of people (each person in a subfolder).
- known_encodings: List of face encodings (128-dimensional vectors) for each known person.
- known_names: Stores the names corresponding to each encoding.

Looping Through Images

```
for name in os.listdir(known_faces_dir):
    person_dir = os.path.join(known_faces_dir, name)
    for filename in os.listdir(person_dir):
        image_path = os.path.join(person_dir, filename)
        image = face recognition. Load image file(image path)
        encodings = face recognition. face encodings(image)
        if encodings:
            known encodings. Append (encodings[0])
            known names. append(name)
```

- Reads each image in every person’s folder.
- Converts the image into a face encoding.
- Stores the encoding and the person’s name for later recognition.

Starting Webcam

```
Video capture = cv2.VideoCapture(0)
```

Starts the webcam feed (0 = default camera).

Main Loop: Face Detection & Recognition

```
while True:
    ret, frame = video capture. read()
    if not ret:
        break
```

- Reads each frame from the webcam.
- If the frame isn’t received correctly (ret == False), break the loop.

Resize and Convert to RGB

```
 Smal _frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
 rgb small frame = small frame[:, :, ::-1]
```

- Downscales the frame for faster processing (¼ of the original size).
- Converts BGR to RGB (OpenCV uses BGR, face recognition expects RGB).

Face Detection and Encoding

```
  Face locations = face recognition. Face locations (rgb small frame)
 face encodings = face recognition. Face encodings (rgb small frame, face
locations)
```

- Face locations: Returns coordinates of detected faces.
- Face encodings: Generates face encodings for detected faces.

```
 for face encoding, face location in zip (face encodings, face locations):
     matches = face recognition. compare faces (known encodings, face
encoding)
     name = "Unknown"
```

- Compares the new face encoding with the known ones.
- Initially assumes the person is "Unknown".

Find the Closest Match

```
Face distances = face recognition. Face distance(known encodings, face encoding)

best match index = np. Argmin(face distances)

if matches[best match index]:

    name = known names [best match index]
```

- Calculates distances between the current face and known faces.
- Picks the best match using the smallest distance.
- If the best match is True in matches, it assigns the correct name.

Draw Bounding Box and Label

```
top, right, bottom, left = [v * 4 for v in face location]

cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)

cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 255, 0), cv2.FILLED)

cv2.putText(frame, name, (left + 6, bottom - 6), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 0), 2)
```

- Multiplies coordinates by 4 (since we scaled down earlier).
- Draws a green rectangle around the face.
- Displays the person’s name below their face.

```
cv2.imshow("Face Detection & Recognition", frame)

if cv2.waitKey(1) & 0xFF == or d("q"):

    break


cv2.imshow("Face Detection & Recognition", frame)
```

```
if cv2.waitKey(1) & 0xFF == or d("q"):
    break
```

Show the Frame

```
cv2.imshow("Face Detection & Recognition", frame)
if cv2.waitKey(1) & 0xFF == or d("q"):
    break
```

- How is the annotated video in a window.
- Exits when the user presses the "q" key.

Clean Up

```
Video capture release ()
cv2.destroyAllWindows()
```

- Stops the camera.
- Closes all OpenCV windows.