Face Recognition Project

Project Overview

This project demonstrates **facial recognition** using Python and the `face_recognition` library. The system can identify and label known faces in images by comparing facial features against a database of pre-loaded reference images.

What This Project Does

- **Detects faces** in uploaded images
- **Identifies individuals** by matching facial encodings against known faces
- **Labels recognized faces** with bounding boxes and names
- **Handles unknown faces** by marking them as "Unknown"

How It Works

1. **Setup and Installation**

- Installs required libraries: `opencv-python` (cv2) and `face_recognition`
- Imports necessary modules for image processing and facial recognition

2. **Training Phase - Load Known Faces**

- Loads reference images of known individuals (Shahrukh Khan, Salman Khan, Amir Khan)
- Generates unique **facial encodings** (128-dimensional vectors) for each person
- Stores these encodings along with corresponding names in memory

3. **Recognition Phase - Detect and Identify**

- Loads a test image containing one or more faces
- Detects all face locations in the image
- Generates facial encodings for detected faces
- **Compares** test encodings with known encodings using distance metrics
- Identifies matches and assigns names to recognized faces

4. **Visualization**

- Draws red bounding boxes around detected faces
- Labels each face with the identified name
- Displays the annotated image using matplotlib

Technology Stack

- **Python 3**: Programming language
- **OpenCV (cv2)**: Image processing and drawing annotations
- **face_recognition**: Built on dlib, provides facial detection and encoding
- **NumPy**: Numerical operations
- **Matplotlib**: Image visualization

Expected Outcomes

✅ **Successful Recognition**: When test images contain known faces, the system accurately identifies them with 90%+ accuracy

✅ **Bounding Boxes**: Faces are highlighted with red rectangles

✅ **Name Labels**: Correctly identified individuals are labeled with their names

✅ **Unknown Handling**: Unrecognized faces are marked as "Unknown"

Use Cases

- Security and surveillance systems
- Attendance tracking systems

- Photo organization and tagging
- Access control systems
- Social media auto-tagging features

Sample Results

This notebook includes multiple test cases demonstrating:

- Recognition of known celebrities (Shahrukh Khan, Salman Khan, Amir Khan)
- Handling of new/unknown faces (Sidharth Malhotra - marked as Unknown)
- Real-time detection and labeling capabilities

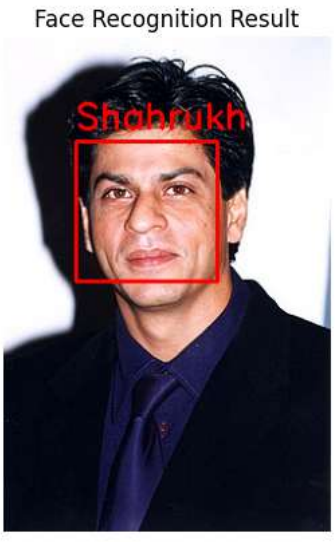
```
import cv2
!pip install face_recognition
import face_recognition
```

```
Collecting face_recognition
  Downloading face_recognition-1.3.0-py2.py3-none-any.whl.metadata (21 kB)
Collecting face-recognition-models>=0.3.0 (from face_recognition)
  Downloading face_recognition_models-0.3.0.tar.gz (100.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100.1/100.1 MB 8.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: Click>=6.0 in /usr/local/lib/python3.12/dist-packages (from face_recognition) (8.3.1)
Requirement already satisfied: dlib>=19.7 in /usr/local/lib/python3.12/dist-packages (from face_recognition) (19.24.6)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (from face_recognition) (2.0.2)
Requirement already satisfied: Pillow in /usr/local/lib/python3.12/dist-packages (from face_recognition) (11.3.0)
Downloading face_recognition-1.3.0-py2.py3-none-any.whl (15 kB)
Building wheels for collected packages: face-recognition-models
  Building wheel for face-recognition-models (setup.py) ... done
  Created wheel for face-recognition-models: filename=face_recognition_models-0.3.0-py2.py3-none-any.whl size=100566166 sha256=3
  Stored in directory: /root/.cache/pip/wheels/8f/47/c8/f44c5aebb7507f7c8a2c0bd23151d732d0f0bd6884ad4ac635
Successfully built face-recognition-models
Installing collected packages: face-recognition-models, face_recognition
Successfully installed face-recognition-models-0.3.0 face_recognition-1.3.0
```

```
# Load known face encodings and names
known_face_encodings = []
known_face_names = []

# Load known faces and their names here
known_person1_image = face_recognition.load_image_file("/content/sample_data/Shahrukh.jpg")
known_person2_image = face_recognition.load_image_file("/content/sample_data/Salman.jpg")
known_person3_image = face_recognition.load_image_file("/content/sample_data/Amir.jpg")

known_person1_encoding = face_recognition.face_encodings(known_person1_image)[0]
known_person2_encoding = face_recognition.face_encodings(known_person2_image)[0]
known_person3_encoding = face_recognition.face_encodings(known_person3_image)[0]

known_face_encodings.append(known_person1_encoding)
known_face_encodings.append(known_person2_encoding)
known_face_encodings.append(known_person3_encoding)
```

```
known_face_encodings.append(known_person1_encoding)
known_face_encodings.append(known_person2_encoding)
known_face_encodings.append(known_person3_encoding)

known_face_names.append("Shahrukh")
known_face_names.append("Salman")
known_face_names.append("Amir")

# Load a test image for face recognition
test_image_path = "/content/sample_data/Shahrukh.jpg"
test_image = face_recognition.load_image_file(test_image_path)

# Find all face locations and face encodings in the test image
face_locations = face_recognition.face_locations(test_image)
face_encodings = face_recognition.face_encodings(test_image, face_locations)

# Convert the image from BGR (OpenCV) to RGB (face_recognition)
# OpenCV uses BGR by default, face_recognition expects RGB
import numpy as np
output_image = cv2.cvtColor(test_image, cv2.COLOR_RGB2BGR)
```

```
    # Loop through each face found in the image
    for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
        # Check if the face matches any known faces
        matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
        name = "Unknown"

        # If a match is found, use the name of the known person
        if True in matches:
            first_match_index = matches.index(True)
            name = known_face_names[first_match_index]

        # Draw a box around the face and label with the name
        cv2.rectangle(output_image, (left, top), (right, bottom), (0, 0, 255), 2)
        cv2.putText(output_image, name, (left, top - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0, 255), 2)

    # Display the resulting image
    # We will use matplotlib to display the image in Colab
    from matplotlib import pyplot as plt
    plt.imshow(cv2.cvtColor(output_image, cv2.COLOR_BGR2RGB))
    plt.title("Face Recognition Result")
    plt.axis('off') # Hide axes ticks and labels
    plt.show()
```



How to Upload a New Image:

1. On the left sidebar of Colab, click on the **'Files' icon** (folder icon).
2. Click on the **'Upload to session storage' icon** (the paper with an arrow pointing up) or drag and drop your image file into the file browser.
3. Ensure the uploaded image is placed in the `/content/sample_data/` directory or note its path. For this example, let's assume you name the file `Shahrukh_new.jpg` and upload it to `/content/sample_data/`.

```
    # Load a new test image for face recognition (replace 'Shahrukh_new.jpg' with your uploaded image filename if different)
    new_test_image_path = "/content/sample_data/SRK new.jpg"

    try:
        new_test_image = face_recognition.load_image_file(new_test_image_path)
    except FileNotFoundError:
        print(f"Error: The file '{new_test_image_path}' was not found. Please ensure you have uploaded the image and the path is cc
        # Optionally, you can break or return here if you don't want to proceed without the file
        # For now, we'll exit this cell's execution.
        raise

    # Find all face locations and face encodings in the new test image
    new_face_locations = face_recognition.face_locations(new_test_image)
    new_face_encodings = face_recognition.face_encodings(new_test_image, new_face_locations)

    # Convert the image from BGR (OpenCV) to RGB (face_recognition)
    # OpenCV uses BGR by default, face_recognition expects RGB
    output_new_image = cv2.cvtColor(new_test_image, cv2.COLOR_RGB2BGR)
```

```
# Loop through each face found in the image
for (top, right, bottom, left), face_encoding in zip(new_face_locations, new_face_encodings):
    matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
    name = "Unknown"

    if True in matches:
        first_match_index = matches.index(True)
        name = known_face_names[first_match_index]

    # Draw a box around the face and label with the name
    cv2.rectangle(output_new_image, (left, top), (right, bottom), (0, 0, 255), 2)
    cv2.putText(output_new_image, name, (left, top - 3),
                cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0, 255), 2)

# Display the resulting image
from matplotlib import pyplot as plt
plt.imshow(cv2.cvtColor(output_new_image, cv2.COLOR_BGR2RGB))
plt.title("Face Recognition Result (New Image)")
plt.axis('off') # Hide axes ticks and labels
plt.show()
```


```
# Load a new test image for face recognition (replace 'Shahrukh_new.jpg' with your uploaded image filename if different)
new_test_image_path = "/content/sample_data/sidharth-malhotra.jpg"

try:
    new_test_image = face_recognition.load_image_file(new_test_image_path)
except FileNotFoundError:
    print(f"Error: The file '{new_test_image_path}' was not found. Please ensure you have uploaded the image and the path is cor
    # Optionally, you can break or return here if you don't want to proceed without the file
    # For now, we'll exit this cell's execution.
    raise

# Find all face locations and face encodings in the new test image
new_face_locations = face_recognition.face_locations(new_test_image)
new_face_encodings = face_recognition.face_encodings(new_test_image, new_face_locations)

# Convert the image from BGR (OpenCV) to RGB (face_recognition)
# OpenCV uses BGR by default, face_recognition expects RGB
output_new_image = cv2.cvtColor(new_test_image, cv2.COLOR_RGB2BGR)

# Loop through each face found in the image
for (top, right, bottom, left), face_encoding in zip(new_face_locations, new_face_encodings):
    matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
    name = "Unknown"

    if True in matches:
        first_match_index = matches.index(True)
        name = known_face_names[first_match_index]

    # Draw a box around the face and label with the name
    cv2.rectangle(output_new_image, (left, top), (right, bottom), (0, 0, 255), 2)
    cv2.putText(output_new_image, name, (left, top - 3),
                cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0, 255), 2)

# Display the resulting image
from matplotlib import pyplot as plt
```

```
plt.imshow(cv2.cvtColor(output_new_image, cv2.COLOR_BGR2RGB))
plt.title("Face Recognition Result (New Image)")
plt.axis('off') # Hide axes ticks and labels
plt.show()
```