Live Facial Recognition App

Distributed Systems – Assignment 2

Vitor Wogel – 20101048

Table of Contents

1. Introduction, architecture and technologies

This application provides a Python web application that runs a live facial detection and recognition on the browser.

The program detects a face in front of the camera using some libraries and built-in functions that will be covered on the requirements and implementation sections. Then it compares the images stored on the environment with the faces of people who appear in front of the camera. If one of the pictures matches the face of someone, their name appears on screen.

2. Functional and non-functional requirements

The most important requirement for this project is the Python language, as the system is designed using libraries made specifically for this language.

If the project is run locally, it is required to have pip (the python dependency manager) and install all of the libraries below.

The libraries needed for this program are:

- Flask (https://pypi.org/project/Flask/)
- Opencv-python (https://pypi.org/project/opencv-python/)
- Certifi (https://pypi.org/project/certifi/)
- Chardet (https://pypi.org/project/chardet/)
- Click (https://pypi.org/project/click/)
- Cmake (https://pypi.org/project/cmake/)
- Decorator (https://pypi.org/project/decorator/)
- Dlib (https://pypi.org/project/dlib/)
- Face-recognition (https://pypi.org/project/face-recognition/)
- Face-recognition-models (https://pypi.org/project/face_recognition_models/)
- Idna (https://pypi.org/project/idna/)
- Imageio (https://pypi.org/project/imageio/)
- Imageio-ffmpeg (https://pypi.org/project/imageio-ffmpeg/)
- Moviepy (https://pypi.org/project/moviepy/)
- Numpy (https://pypi.org/project/numpy/)
- Pillow (https://pypi.org/project/Pillow/)
- Proglog (https://pypi.org/project/proglog/)
- Requests (https://pypi.org/project/requests/)
- Tqdm (https://pypi.org/project/tqdm/)
- Urllib3 (https://pypi.org/project/urllib3/)
- Wincertstore (https://pypi.org/project/wincertstore/)

Beside the language and its libraries, it is necessary to have pictures of the people you want to recognize on a folder inside the root of the project.

It is required to have a web browser in which it is possible to run the HTML template made for this assignment.

The following content is related to basic functionalities of the main libraries used on the code.

About Flask:

“Flask is used for developing web applications using python, implemented on Werkzeug and Jinja2. Advantages of using Flask framework are:

- There is a built-in development server and a fast debugger provided.
- Lightweight
- Secure cookies are supported.
- Templating using Jinja2.
- Request dispatching using REST.
- Support for unit testing is built-in.”

About OpenCV-Python:

“OpenCV-Python is a library of Python bindings designed to solve computer vision problems.”

“OpenCV has a function to read video, which is cv2. VideoCapture(). We can access our webcam using pass 0 in the function parameter.”

About Face Recognition:

“Recognize and manipulate faces from Python or from the command line with the world’s simplest face recognition library. Built using dlib’s state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark.”

About Numpy:

“Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.”

“NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.”

“NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.”

“The core of NumPy is well-optimized C code. Enjoy the flexibility of Python with the speed of compiled code.”

“NumPy’s high level syntax makes it accessible and productive for programmers from any background or experience level.”

“Distributed under a liberal BSD license, NumPy is developed and maintained publicly on GitHub by a vibrant, responsive, and diverse community.”

3. Implementation

Having all the requirements needed, the most important thing now is how to implement that in code and develop the project.

In the main script (app.py) it is needed to import all the libraries that have the functions required to perform the actions. For that, the following code was implemented as the first thing on this script:

```
from flask import Flask, render_template, Response

import cv2

import face_recognition

import numpy as np
```

Then we need to assign our libraries to variables, to then be able to create a web application based on python using Flask and use the cv2 to capture each frame of the video displayed on camera.

```
app=Flask(__name__)

camera = cv2.VideoCapture(0)
```

After that, we use the face recognition library to load a picture and have it learn how to recognize it, by encoding it.

```
# Load a sample picture and learn how to recognize it.

vitor_image = face_recognition.load_image_file("images/vitor.jpg")

vitor_face_encoding = face_recognition.face_encodings(vitor_image)[0]
```

We are creating two arrays, one to store the face encodings and the other to save the names that should be displayed within each recognition.

```
# Create arrays of known face encodings and their names

known_face_encodings = [

    vitor_face_encoding,
```

```
    messi_face_encoding,
    neymar_face_encoding
]
known_face_names = [
    "Vitor",
    "Messi",
    "Neymar Jr."
]
# Initialize some variables
face_locations = []
face_encodings = []
face_names = []
process_this_frame = True
```

Finally we got to our main function, in which while the app is running we read each frame of the camera, use the method face_location of the face recognition to detect which frames correspond to a face, then we run a for loop to check if the face detected is compatible with one of the encoded pictures we have on our array of known faces using the compare_faces method.

We also try to see if the face is close enough to what we are seeking, because maybe the face is not exactly like the one in the picture, but if it is close enough that means that probably is the same person.

At last we display the results drawing a box around the detected face and displaying the name of the person who was recognized or “Unknown” for a person whose picture is not on the database.

```
def gen_frames():
    while True:
        success, frame = camera.read()  # read the camera frame
        if not success:
            break
        else:
```

```
            # Resize frame of video to 1/4 size for faster face recognition
processing
            small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
            # Convert the image from BGR color (which OpenCV uses) to RGB color
(which face_recognition uses)
            rgb_small_frame = small_frame[:, :, ::-1]



            # Only process every other frame of video to save time


            # Find all the faces and face encodings in the current frame of video
            face_locations = face_recognition.face_locations(rgb_small_frame)
            face_encodings = face_recognition.face_encodings(rgb_small_frame,
face_locations)
            face_names = []
            for face_encoding in face_encodings:
                # See if the face is a match for the known face(s)
                matches = face_recognition.compare_faces(known_face_encodings,
face_encoding)
                name = "Unknown"
                # Or instead, use the known face with the smallest distance to
the new face
                face_distances =
face_recognition.face_distance(known_face_encodings, face_encoding)
                best_match_index = np.argmin(face_distances)
                if matches[best_match_index]:
                    name = known_face_names[best_match_index]



                face_names.append(name)
```

```
            # Display the results
            for (top, right, bottom, left), name in zip(face_locations,
face_names):
                # Scale back up face locations since the frame we detected in was
scaled to 1/4 size
                top *= 4
                right *= 4
                bottom *= 4
                left *= 4


                # Draw a box around the face
                cv2.rectangle(frame, (left, top), (right, bottom), (255, 161,
81), 2)


                # Draw a label with a name below the face
                cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (255,
161, 81), cv2.FILLED)
                font = cv2.FONT_HERSHEY_DUPLEX
                cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255,
255, 255), 1)


            ret, buffer = cv2.imencode('.jpg', frame)
            frame = buffer.tobytes()
            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
```

The following piece of code is responsible for live streaming the app on the browser making use of the Flask library and its resources:

```
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/video_feed')
def video_feed():
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace;
boundary=frame')
if __name__=='__main__':
    app.run(debug=True)
```

4. Conclusion

We could see that the project implements a live streaming face recognition system based on libraries such as Flask, that allow web apps to be built and developed using Python, OpenCV to get the image frames from camera and Face Recognition to encode those faces into vectors and compare with the pre trained pictures from the local database.

To get a better performance while using the app some measures were taken, like resize the frames of video to 1/4 size for faster face recognition processing and only processing every other frame to save time.

5. References

SAINI, Ashrey, ‘An Easy introduction to Flask Framework for beginners’, https://www.analyticsvidhya.com/blog/2021/10/easy-introduction-to-flask-framework-for-beginners/

Unknown, Topcoder, ‘WHAT IS THE OPENCV LIBRARY AND WHY DO WE NEED TO KNOW ABOUT IT?’, https://www.topcoder.com/thrive/articles/what-is-the-opencv-library-and-why-do-we-need-to-know-about-it

Rajnis09, GeeksForGeeks, ‘Python OpenCV | cv2.imshow() method’, https://www.geeksforgeeks.org/python-opencv-cv2-imshow-method/

Documentation, Python, ‘face-recognition 1.3.0’, https://pypi.org/project/face-recognition/

Documentation, Numpy, ‘NumPy 1.23.0 released’, https://numpy.org/