ITP 499
Final Project:
Face Identification
Design Doc

An Pham

Files/Folders:

capture_Data.py: client side code

- This file has a class named WebCam, which uses the OpenCV library to capture the client's video's frames & send it to the server as bytes.
    - A unique features is that it returns a box & message to the client so they'll capture a frontal face image. This would hopefully return a more accurate identification result.
- The constructor initializes the camera
- save_photo() captures the current frame of the video feed from OpenCv and saves it into the /unknown photo for facial identification

receive_Data.py: server side code

- Initially loads the database of known faces before starting the server
- Handles both upload_photo and capture_image
    - Both of these eventually call access_control

face_Identification.py:

- This module provides two functions:
    1. load_known()
        a. Loads our files of images within the "known" directory to provide data to our model
        b. Checks if mysql database already holds data for certain people
    2. detect_unknown()
        a. At the end of the function removes files within the "unknown" directory to prevent multiple matches from the previous run
- Uses face_recognition library to perform facial recognition
- Holds functions used to initialize the database

access_Control.py:

- Imports the detect_unknown() function from the face_Identification module
- Calls detect_unknown() within access_control() function
    - If we receive an empty list, we return "No Matches" to the server
    - Else we return the first match to the server (Yes there can be a case where there is more than 1 match)

/known

- This folder contains the images we will use to load into our training model from face_recognition

- Uploading new images to /known folder must be named as client's name and can be saved as any image extension (png, jpg, jpeg)

/unknown

- A directory to hold the image file of the client after he or she either uploads to capture a new photo. Also used to organize or project.

/templates

- Contains the .html files for or webpages:
    1. Capture_image.html: provides instructions and ability for client to use our program over their web browser
    2. Home_page.html: The client's entry point when running or application. Provides two button to either redirect to capture_image or upload_image
    3. Upload_image.html: button to allow client to upload photo from his/her machine, then send the file to the server

/dlib

- Holds the dependencies to run face_recognition

/cascades

- Contains haarcascade_frontalface_default.xml, which is used by openCV to recognize face, primarily from the frontal view

Motivation and need:

- The project is a basic client/server web based application that provides facial recognition as a method of authorization and access control. Although many of the core functionalities were implemented with open-source libraries, the application works fairly well and shows how machine learning can be integrated with future applications. As a double major studying both electrical engineering & computer science, this project captured my interest and allowed me to understand at a high level the concepts of machine learning. Hopefully in the future, I could dive into a more detailed understanding behind the algorithms used to create these libraries.

Libraries:

- Face_recognition:
    - Algorithms for facial recognition
- Dlib:
    - Dependency for face_recognition
- OpenCV:
    - Capture frames via webcam and facial detection
- Numpy:
    - Supports the algorithms used for face_recognition
- Flask:
    - Light-weight web framework for quick web application development
- Os:
    - Allows us to move around directories when saving pictures
- Pymysql:
    - Saves new photos that are inputted into the /known folder

Classes:

- Created a WebCam class within capture_Data.py to help organize the OpenCv functionalities
- Other python files were broken into independent methods instead of belonging to a class that encapsulated them.

Program flow:

- Majority of the code was organized by independent methods to represent stages of the application with the exception of WebCam class.
    - **Stages:**
        1. Load our DataBase preloaded with photos in the known directory then launch the server in recieve_Data.py
        2. Handle Client Requests:

2.1. If Client asks to upload photo:
    - Redirect them to the upload_photo() method: provides an html form for client o POST request
2.2. If Client asks to take new photo:
    - Redirect them to the capture_photo() method: this method uses a while true loop to provide a stream of bytes which later gets converted back into .jpeg format for the user to view
    - **Note:**

        I. Camera will take awhile to load, please be patient!

        II. Internet explorer browser does not support the application, please try another browser (Chrome)

3. Access Control:

3.1. Client has successfully uploaded a photo:

3.1.1. Redirect client's url to upload_photo() method, this function will call access_control

3.2. Client has successfully captured a new photo:

3.2.1. Redirect client's url to captured_photo() method, this function will save_photo() method from WebCam() class before calling access_control

Future Work:

- Make this application more scalable for many people to use
- Increase the appeal of the user interface as well as adding more functionalities
- Implement this into an IoT project such as a raspberry pi, ordroid, or other cheap single board computers.