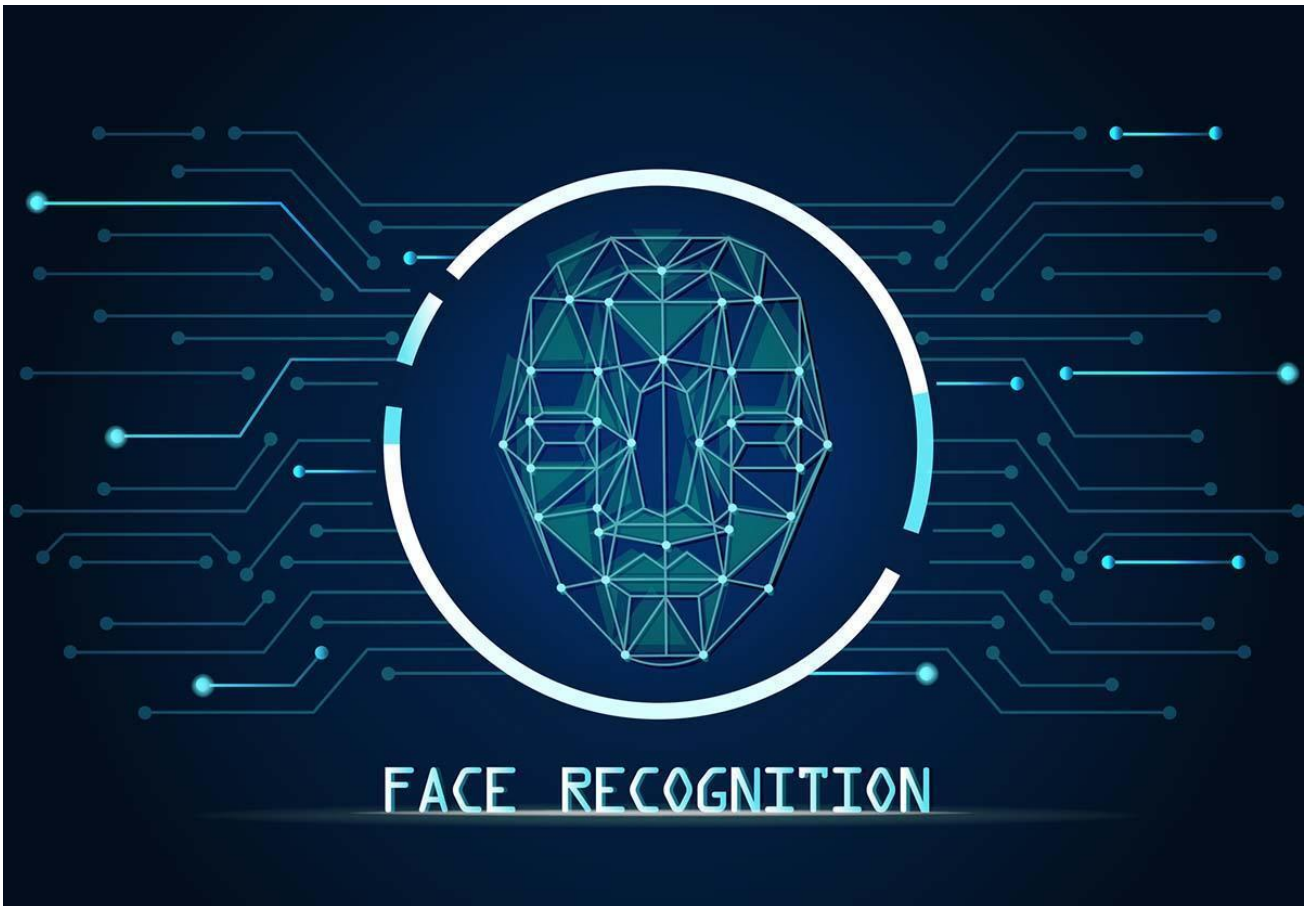


Face Recognition Based Attendance Manager

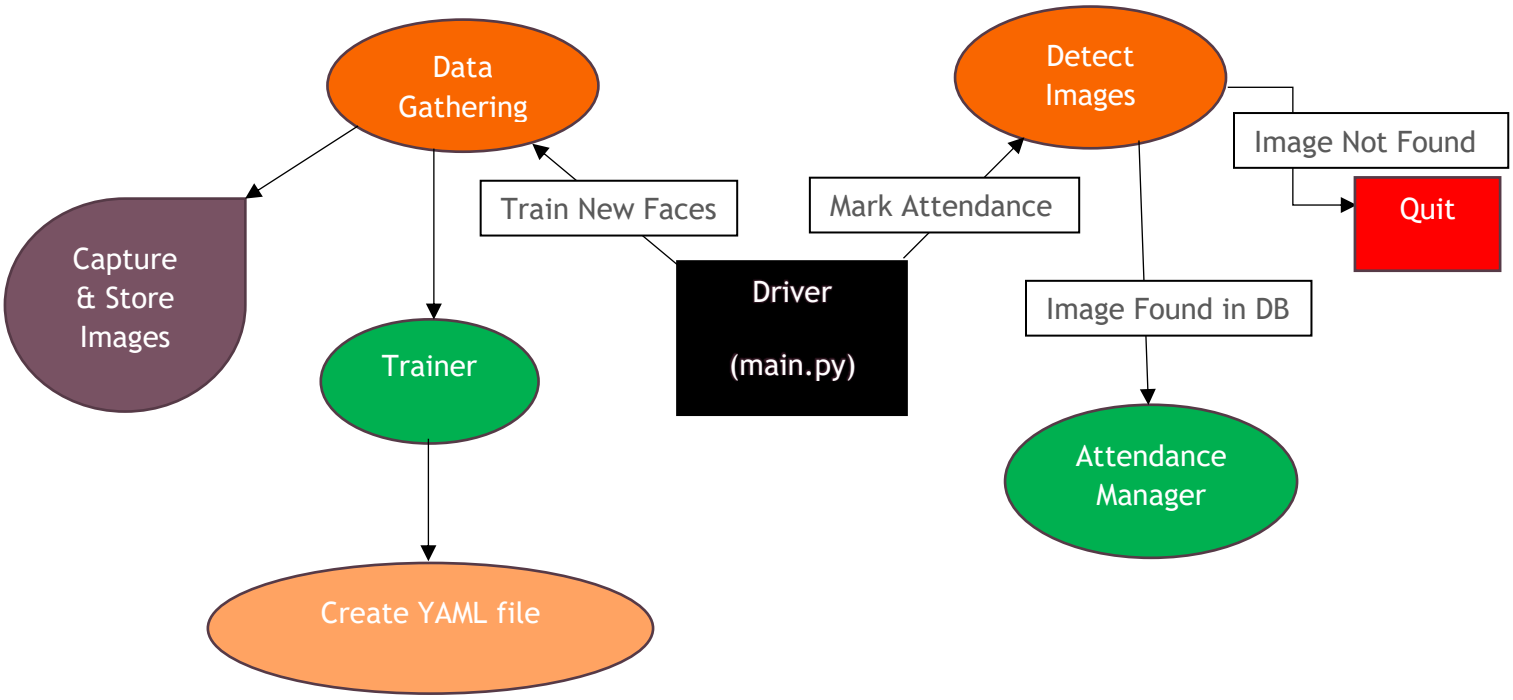
Kartikey V Hebbar

15th August 2021

Introduction

This project is intended towards building a software capable of recognizing the faces of the registered users and marking the attendance correspondingly. To accomplish this, the software is built upon Machine Learning and uses the OpenCV library of Python. This project primarily uses the Haar-Cascade Classifier along with Haar-Cascade Frontal Face Dataset to detect and train facial images. It also uses Local Binary Pattern Histogram (LBPH) Algorithm to recognize images in real-time. The whole project has been built using Python programming language with an extensive use of Object-Oriented Programming techniques and modularity. A separate module has been built to manage the attendance report of the recognized images.

Project Flow



- Driver: main.py contains the driver code. It is used to interact with the user, take necessary inputs and direct the whole application in a definite path.
- Data Gathering: This stage uses dataGathering.py file which handles the image detection and capturing process. It uses CascadeClassifier() function of the OpenCV library to read the Haar-Cascade Frontal Face Dataset and detect facial images. Thereafter, the images are captured and stored in the dataset directory. This program captures 50 images of every user in a small amount of time. In addition to the above functionality, this step also involves

two other files, datasetconfig.py – to manage the image counters and json_operator.py – to handle the updates to the JSON file (userdata.json). This JSON file is used to maintain a central record for the existing/added users. Every new user's information (name and ID) is stored in this JSON file.

- Trainer: This step involves training the captured data to the model. In simpler terms, trainer.py makes the application learn every user's image with their name (just the way humans learn). This learning helps the application to recognize the users when they appear in front of the camera. To recognize the images in order to learn them, LBPHFaceRecognizer_create() function is used. The trained model is created in the form of a YAML file which is basically a Serialized file containing the data.
- Detect Images: This step uses detector.py file which again use the CascadeClassifier() and LBPHFaceRecognizer_create() functions from the OpenCV library to detect images. The aim of this step is to detect the faces in front of the camera and compare with the application's learnt model (YAML file). This comparison dispays the name of the user from the userdata.json file in case the person in front of the camera is already a user, else it shows "Unknown" and the program terminates. In case the user is recognized, then update_attendance.py is used to append the username in the attendance_report.csv along with the time and date, and a status "P" denoting the user is present.

Technical Description

1. Haar-Cascade Classifier: It is an Object Detection Algorithm used for identifying faces in an image or a real-time video. In this project, Haar-Cascade Classifier is used primarily for detecting the users' faces with the help of the Haar-Cascade Frontal Face Dataset. This is facilitated by OpenCV library of Python which helps in using the functions of Haar-Cascade Classifier.

2. Local Binary Pattern Histogram (LBPH) Algorithm: The second and the most important part of this project is Face Recognition. LBPH algorithm is a Face Recognition algorithm which is used to recognize the faces of the users from the database. OpenCV library of Python provides the functions to use LBPH Algorithm.

Further Development

- The best use cases of this project would be in places where attendance of many people need to be captured. This system can be readily used in such scenarios once a proper training of the images is done. Proper training of the images is a necessary step to avoid errors in case of many user data is stored. Although, both Haar-Cascade and LBPH algorithms are capable of detecting and recognizing low-light and similar images quite well, still they require a good amount of training to yield best results. The program shows the confidence % below the image which tells how well the real-time video image matches the dataset. The higher this number, the better will be the performance of the application.
- Currently, the application can append the attendance status of any one person which it recognizes first. Also, it would append the same person's attendance again and again if it is recognized which would not be suitable for real-time usage. There can be a further development in this to make the attendance capturing process more robust than it is now. The system can detect and recognize multiple people together now but must be able to mark attendance for multiple entries also.
- This project once fully developed can be an ideal choice for managing attendance in classrooms.