Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

# [Automated Classroom Attendance System Using Face Recognition]

By: Yonathan Cherkos  ATR/3871/08

Advisors' Name: Mr. Nebyu Yonas

**Date of submission**: December 14, 2020

# Abstract

An automatic classroom attendance system is a system that is used to automate the manual attendance marking method. The manual attendance marking method is a repetitive, time-consuming, tedious task in many schools and colleges. So by automating this traditional method, we can save more time for the actual lecture, increase the productivity of teachers, increase students punctuality, and also generate attendance reports and its analysis result easily. An automated attendance system can be implemented using various techniques of biometrics. Face recognition is one of them which does not involve human intervention and it has been widely used in many other applications. Face recognition is a method of identifying or verifying the identity of an individual using their face. The system or desktop app I built performs nearly 100 at recognizing faces. It can successfully recognize faces at different face orientation, distance and light conditions. but It sometimes fails to recognize faces in low light conditions. in spite of that the app can work great in classrooms which have sufficient lighting conditions or day-time.

# Acknowledgment

I would like to express my deep and sincere gratitude to my project advisor Nebyu Yonas, for his patient guidance, enthusiastic encouragement and useful critiques of this project. His willingness to give his time and support so generously has been very much appreciated. I would also like to thank Dr. Adrian Rosebrock founder and author of pyimagesearch, for his great blog posts on pyimagesearch and his very useful deep learning for computer vision books. My grateful thanks are also extended to Dr. Andrew Ng deep learning coursera specialization instructure, for his most popular and useful deep learning course.

Finally, I wish to thank my parents and my friend for their support and encouragement throughout my project.

# Contents

# Chapter 1

# Introduction and Background

The main object of this project is to build a system that will automate the way schools and colleges take and manage student attendance. There are many reasons why one should choose an automated attendance management system over the traditional one. speed and accuracy are one of the main reasons. The other critical reason is it will generate an easy to understand and use attendance report automatically.

Traditionally, student's attendance is taken manually by using the attendance sheet. This traditional method of attendance marking is a tedious task in many schools and colleges. It also adds an extra burden to the faculties who should mark the attendance by manually calling the names of students which might take about 5 minutes of the entire session. And this can add up to hours of lost productivity over the course. Moreover, it is very difficult to verify one by one student in a large classroom environment with distributed branches whether the authenticated students are actually responding or not. So if this can be automated we can save much more time and increase the productivity of teachers.

One of the big problems with manual taking attendance systems especially for high school and lower grades is a teacher can only take attendance in the first class or lecture. And many students are not attending the class after break or lunchtime. So one of the main goals of this project is to allow teachers to take attendance at every class(lecture) or at any time they desire and reduce drastically students who are not attending class after break or lunchtime. In this project, I built a system that detects the faces of students from a live video stream of the classroom, and attendance will be marked if the detected face is found in the database. the detected student will be marked after it has been found in the video for some repetition threshold to avoid false-positive recognition.

The system uses a face recognition based attendance system instead of other biometric methods. Face recognition-based attendance system is a process of recognizing the student's face for taking attendance by using face biometrics. In face recognition projects, a computer system will be able to find and recognize human faces fast and precisely in images or videos that are being captured through a surveillance camera for various tasks. Over the past few decades, a rapid improvement in computer vision gives the computer human-level capability to recognize faces. deep learning is one of the domains that contribute largely to give computer human-level face recognition capability. because of this face recognition is gaining more popularity and has been widely used in different computer vision applications

Face recognition has characteristics that other biometrics(example. fingerprint) do not have. Facial images can be captured from a distance and any special action is not required for authentication. Due to such characteristics, face recognition technology is applied widely, not only to security applications but also in many face applications. The input to the system is a video from an HD webcam which will be mounted in front of the class and the output is an excel sheet with attendance of the students in the video. In this project, attendance is registered from a video of students of a class by first performing face Detection which separates faces from non-faces, and then face encoding which encodes each face to 128-dimensional vectors, and finally face recognition or face matching is carried out which finds the match of the detected face from the face database (collection of student's name and encoded images). If it is a valid match then the attendance result is registered to an excel sheet.

The application is able to recognize successfully registered students at different face orientations, distances, and lighting conditions. It is able to recognize the face successfully in almost all cases. It sometimes fails to recognize faces in low light conditions. But since the app will be used in classrooms which have a sufficient lighting condition with sunlight or lamp, it will work fine for its purpose.

# Chapter 2

# Related works

One of the most powerful and compelling types of AI is computer vision [3] which you've almost surely experienced in any number of ways without even knowing it. Computer vision is the field of computer science that focuses on replicating parts of the complexity of the human visual system and enabling computers to identify and process objects in images and videos in the same way that humans do. Until recently, computer vision only worked in a limited capacity.

Over the past few decades, with rapid improvement in artificial intelligence and innovations in deep learning and neural networks, the field of computer vision has been able to take great leaps in recent years and have been able to surpass humans in some tasks related to detecting and labeling objects. Face detection is one of the most fundamental aspects of computer vision which has been showing promising results.

There are different approaches or algorithms which are used to implement face detection. The Viola-Jones algorithm (also known as Haar cascades) is the most common algorithm in the computer vision field used for face detection on the image. Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and based on the concept of features proposed by Paul Viola and Michael Jones in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" [14] in 2001. Haar-Features are good at detecting edges and lines. This makes it especially effective in face detection. A Haar-based classifier despite it being fast and easy to implement may result in more false-positives than deep learning-based face detection algorithms.

Face recognition is one of the most important applications in video surveillance and computer vision. However, the conventional algorithms of face detection are susceptible to multiple conditions, such as lighting, occlusion, viewing angle, or camera rotation. Therefore, face recognition based on deep learning can greatly improve the recognition performance and accuracy. There are various deep neural networks used in face recognition. one of the most popular deep learning-based face detection is based on the Single Shot Detector(SSD) framework with a ResNet base network [14].

There are different biometrics-based attendance tracking techniques. A student attendance system using fingerprint scanners [9] has been long in the industry. Fingerprint features are considered to be the best and fastest method for biometric identification. These features are more secure to use and unique for every person. and because of this, they have been used in different applications and mainly they have been used to mark the attendance of employees in offices and workplaces. Despite fingerprint-based attendance systems being fast and unique for everyone, it required students' interactions with the system to scan their fingers and mark attendance. And also it is hard to take attendance at every session(lecture) of a class or at any time the teachers want.

Authors in [1] proposed a model of an automated attendance system. The model focuses on how face recognition incorporated with Radio Frequency Identification (RFID) detects the authorized students and counts as they get in and get out of the classroom. The system keeps the authentic record of every registered student. The system also keeps the data of every student registered for a particular course in the attendance log and provides necessary information according to the need.

In this paper [7], the authors have designed and implemented an attendance system that uses iris biometrics. Initially, the attendees were asked to register their details along with their unique iris template. At the time of attendance, the system automatically took class attendance by capturing the eye image of each attendee, recognizing their iris, and searching for a match in the created database. The prototype was web-based.

In [10], the authors proposed an attendance system based on facial recognition. The algorithms like Viola-Jones and Histogram of Oriented Gradients (HOG) features along with Support Vector Machine (SVM) classifiers were used to implement the system. Various real-time scenarios such as scaling, illumination, occlusions, and pose were considered by the authors. Quantitative analysis was done on the basis of Peak Signal to Noise Ratio (PSNR) values and was implemented in MATLAB GUI.

# Chapter 3

# Specifications and Design

## 3.1 Objective

The major objective of this project is to build a system that will automate the manual attendance method and provide paperless work. It have many advantages over the traditional attendance marking method. such as:-

- increase speed and accuracy
- increase the productivity of teachers
- Auto-generate various types of reports of the student attendance
- Increase student and teachers punctuality
- Reduce paperwork and save time and money
- Eliminate duplicate data entry and errors in time and attendance entries
- Keep the parents informed about the student performance via Email and SMS alerts

## 3.2 System Overview

An HD webcam will be mounted in front of the classroom which will capture a live video stream and send the capture video stream data to the desktop application for processing using a USB cable. Then the app will detect faces present in the video stream. Each of the detected faces will be encoded to 128 dimensional vectors using pre-trained deep learning based face encoder model or to be specific using FaceNet. Then finally each of the encoded faces will pass through the face recognition or face matching subsystem, which will match encoded faces with previously registered student encoded faces. And if a match is found with sufficient similarity and repetition threshold, it will mark the student with the given face as a present. And it will do the same with all the detected faces.

An authenticated user or teacher can register students, take attendance, view attendance, and update student attendance if needed. A teacher can be registered by providing his/her necessary information in the sign up page. Then if all the required fields are filled, the registration information will be stored in the SQL database and he/she will be authonicate to use the system. After registering a teacher can login into the applications by entering his/her username and password.

A teacher can register students by entering a student's full name and capture or upload student's face images. There are two options to insert student face images. The first and recommended approach is capturing student faces using the mounted webcam. This is recommended approach because a face detection model will check for presence of face and draw bounding box on each frame and by doing so it will guide you not to save an frame on which face doesn't detected. The captured face images should be as different as possible. The face images should capture faces with different emotions, light conditions, orientation, and distance. It is highly recommended to capture or upload 10 to 20 images for each student to generate more diverse face encoding or face representation for each student. The second approach is by uploading already captured student face images.

A teacher can take student attendance automatically at the beginning and end of the classes or at any time he/she desires. When a command to start taking attendance is given to the app, it will start recording video and analyze each frame for the presence of a registered student in the video stream. A face detection model will detect the presence of faces and a face aligner algorithm will align the detected face. Then a face encoding model will encode the detected and aligned faces. And finally a face recognition or face matching algorithm will try to match each face with the encoded face database. and if a match found it will update the student attendance spreadsheet.

A student attendance result can be shown in the student attendance page where student attendance results are present in easy to use and understandable manner. Each student row will be marked with different colors based on their attendance result. if a student attends more than 90% of the classes, he/she will be marked with green color(good). and if a student attends greater than 80% and less than 90% of the classes, he/she will be marked with yellow(warning). and if a student only attended less than 80% of the classes, he/she will be marked red, and could not be able to sit for exams. There is a feature to filter student based on their attendance result. For example, it can filter student that are in danger zone or attending less than 80% of the class. There is also a feature to search for a specific student and display his/her attendance result. And if teacher need the attendance result in paper or pdf, there is a feature that will export the attendance result to a pdf file.

Besides displaying the attendance result for all the students in a single table, we can also see detailed attendance results for each student. If you click on one of the student attendance row, it will take users to a detailed page for that student, which shows all the classes the student attended or absent. And also if there is a convincing reason, the teachers can change the student attendance result for any classes and update the attendance CSV file.

## 3.3 Face recognition pipeline

Here are the steps I will be taking to implement face recognition Detect/identify faces in an image (using a face detection model) Apply face alignment on the detected faces Apply feature extraction to the aligned faces or calculate face encodings (numbers that describe the face) Compare the face encodings of known faces with those from test images to tell who is in the picture.

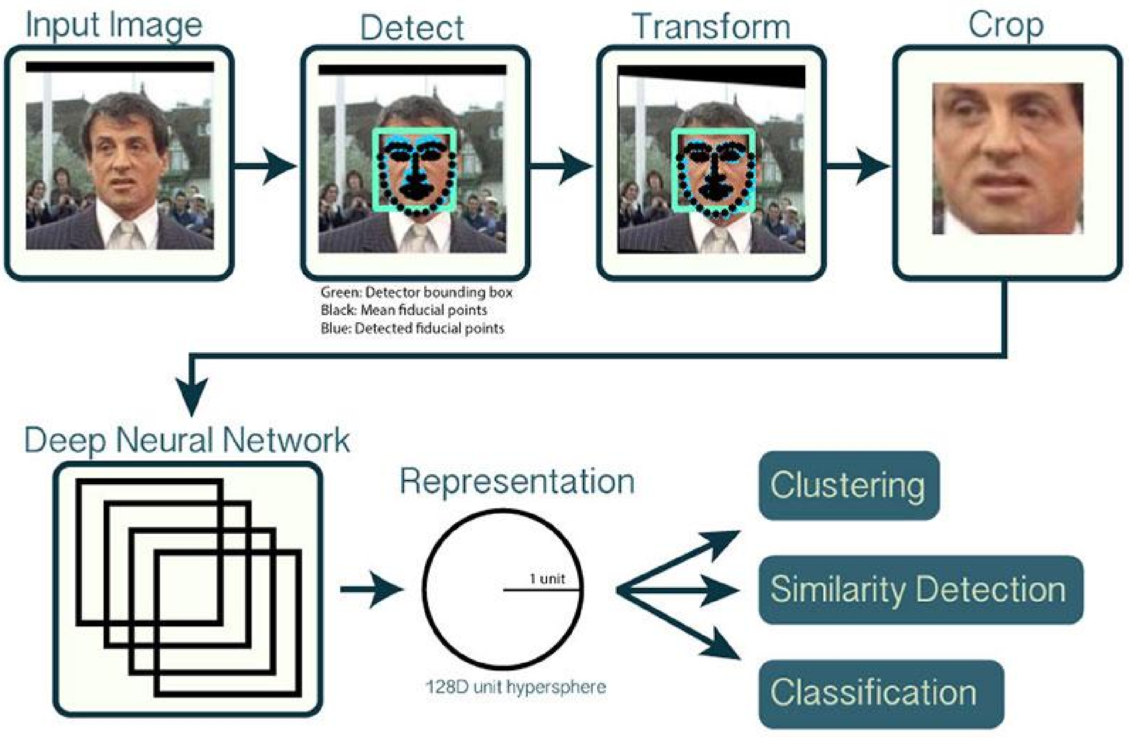


Figure 3.1: An overview of the OpenCV face recognition pipeline

## 3.4 UI Design

The user interface has been built using QtDesigner, PyQt5 ui designing tools. Qt Designer is the Qt tool for designing and building graphical user interfaces (GUIs) with Qt Widgets. You can compose and customize your windows or dialogs in a what-you-see-is-what-you-get (WYSIWYG) manner, and test them using different styles and resolutions. Widgets and forms created with Qt Designer integrate seamlessly with programmed code, using Qt's signals and slots mechanism, so that you can easily assign behavior to graphical elements. All properties set in Qt Designer can be changed dynamically within the code. Furthermore, features like widget promotion and custom plugins allow you to use your own components with Qt Designer.

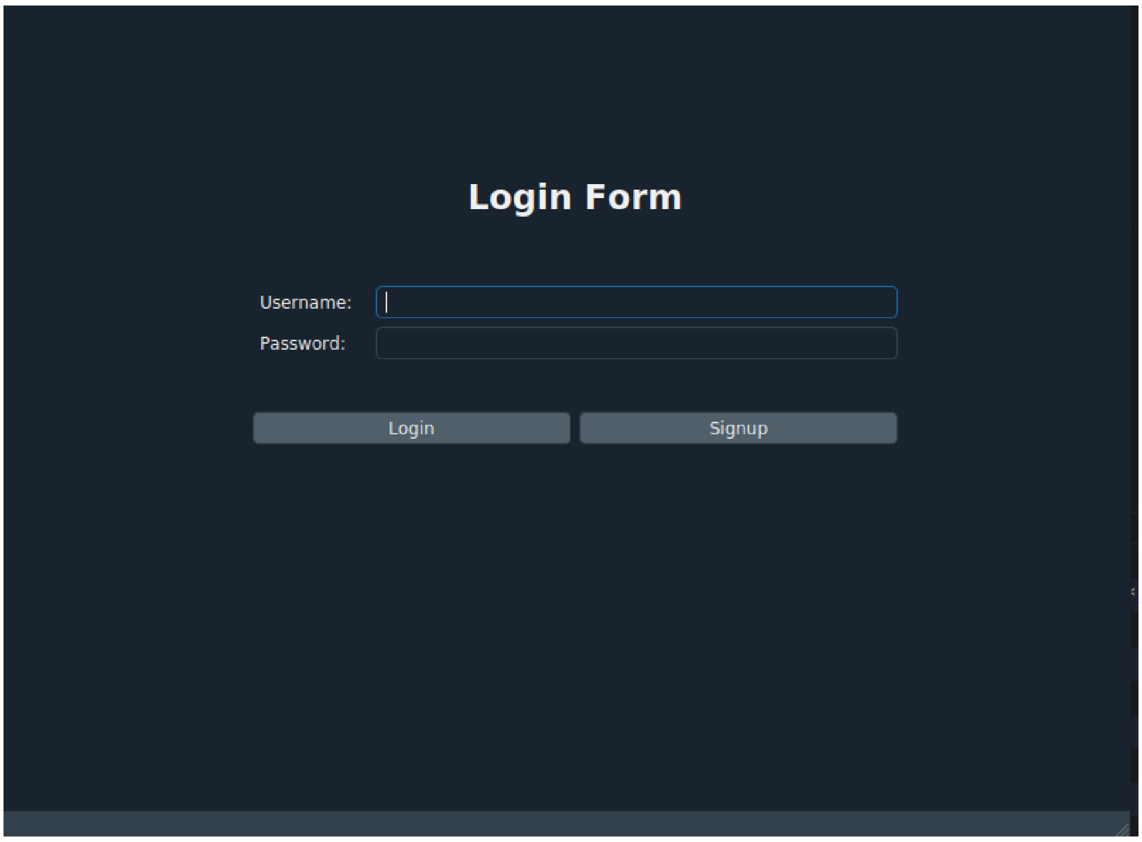


Figure 3.2: login form

# Chapter 4

# Implementation

The primary programming language used to implement the entire application the face recognition modules, user interfaces, and others, has been written and built with python program. The video pre-processing, face detection, face recognition, and other computer vision tasks have been implemented using the popular OpenCV library. I use QtDesigner to design the user interface. The app is built and integrated with the face recognition modules using PyQt5 python gui library. Since the primary requirement of this system is higher accuracy I use a deep learning based face detector. I also use different python packages like numpy, pandas, imutils, pdfkit, qdarkstyle and others packages.

## 4.1 Python and OpenCV

The primary programming language used in this project is python. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level build-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development. So for this project, I entirely use python both the face recognition part and when designing and building the desktop app.

The image and video pre-processing, face detection, face recognition, and other computer vision tasks have been implemented using the popular OpenCV [2] library. OpenCV is the huge open-source library for computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. OpenCV the most powerful and popular coputer vision library. It supports a wide variety of programming languages such as C++, Python, Java etc. OpenCV Python is nothing but a wrapper class for the original C++ library to be used with Python. Using this, all of the OpenCV array structures gets converted to/from NumPy arrays.

I use git and GitHub to track my progress. GitHub is a Git repository hosting service, but it adds many of its own features. While Git is a command line tool, GitHub provides a Web-based graphical interface. It also provides access control and several collaboration features, such as a wikis and basic task management tools for every project.

## 4.2 GUI With Python

Python has been extensively used for data analysis, computer vision, and machine learning, and others. But building a GUI with python is a little bit complex. Fortunately, there are some options available for programmers looking to create an easy way for users to interact with their programs. some of the common python GUI libraries are TKinter, kivy, PyQt5, and others. Let me now give an overview of the two libraries I tested and used to implement apps which are Tkinter and PyQt5.

Tkinter is one of the most popular Python GUI libraries for developing desktop applications. It's a combination of the TK and python standard GUI framework. Tkinter provides diverse widgets such as labels, buttons, text boxes, checkboxes that are used in a graphical user interface application. The button control widgets are used to display and develop applications while the canvas widget is used to draw shapes like lines, polygons, rectangles, etc. in the application. Furthermore, Tkinter is a built-in library for Python, so you don't need to install it like other GUI frameworks. Despite Tkinter is easy and simple to use, it is very basic. It cannot work appropriately with video processing and other advanced features. and aesthetics wise it is kind of hard to build a beautiful UI with this library.

The other popular python GUI framework is called PyQT5 [6]. PyQT5 is a graphical user interface (GUI) framework for Python. It is very popular among developers and the GUI can be created by coding or a QT designer. A QT designer a visual framework that allows drag and drop of widgets to build user interfaces. It is a free, open-source binding software and is implemented for a cross-platform application development framework. It is used on Windows, Mac, Android, Linux, and Raspberry PI. Absolutely, PyQt is not easy for newbie programmers!, but it's the best framework for building more complex and beautiful GUI using python. QtDesigner is a very good ui design tool to build beautiful and interactive UI interfaces. using QtDesigner we can easily design the GUI and then add the logic parts of the program. For this system, since it needs video processing and other advanced and complex features I choose PyQt5 and implement the app with it. I use QtDesigner the graphic ui designing tool to design the user interface.

## 4.3 Face detection and Face recognition

Face detection [11] also called facial detection is an artificial intelligence (AI) based computer technology used to find and identify human faces in digital images. Face detection technology can be applied to various fields – including security, biometrics, law enforcement, entertainment, and personal safety – to provide surveillance and tracking of people in real-time.

Face detection has progressed from rudimentary computer vision techniques to advances in machine learning (ML) to increasingly sophisticated artificial neural networks (ANN) and related technologies; the result has been continuous performance improvements. It now plays an important role as the first step in many key applications including face tracking, face analysis, and facial recognition. Face detection has a significant effect on how sequential operations will perform in the application.

Face detection applications use algorithms and ML to find human faces within larger images, which often incorporate other non-face objects such as landscapes, buildings, and other human body parts like feet or hands. Face detection algorithms typically start by searching for human eyes one of the easiest features to detect. The algorithm might then attempt to detect eyebrows, the mouth, nose, nostrils, and the iris. The methods used in face detection can be knowledge-based, feature-based, template matching, or appearance-based, and deep learning-based.

Since for this system, the required face detector algorithm needs to be with higher accuracy but tolerable speed, I choose deep learning-based face detection. OpenCV ships out-of-the-box with pre-trained Haar cascades that can be used for face detection. But beginning from OpenCV 3.3 deep learning-based face detector[11] that has been part of OpenCV since. Now can perform fast, accurate face detection with OpenCV using a pre-trained deep learning face detector model shipped with the library. The deep learning-based face detector is found in the dnn module of OpenCV.

When using OpenCV's deep neural network module with Caffe models, you'll need two sets of files. The first is the .prototxt file(s) which defines the model architecture (i.e., the layers themselves). and the second is the .caffemodel file which contains the weights for the actual layers Both files are required when using models trained using Caffe for deep learning.

Face recognition [12] is a method of identifying or verifying the identity of an individual using their face. It is a technology capable of matching a human face from a digital image or a video frame against a database of faces. Face recognition systems can be used to identify people in photos, video, or in real-time.

Face recognition systems use computer algorithms to pick out specific, distinctive details about a person's face. These details, such as distance between the eyes or shape of the chin, are then converted into a mathematical representation and compared to data on other faces collected in a face recognition database. The data about a particular face is often called a face template and is distinct from a photograph because it's designed to only include certain details that can be used to distinguish one face from another.

It is a task that is trivially performed by humans, even under varying light and when faces are changed by age or obstructed with accessories and facial hair. Nevertheless, it remained a challenging computer vision problem for decades until recently.

Deep learning methods are able to leverage very large datasets of faces and learn rich and compact representations of faces, allowing modern models to first perform as-well and later to outperform the face recognition capabilities of humans.

When researching a face recognition system, it is important to look closely at the "false positive" rate and the "false negative" rate, since there is almost always a trade-off. For example, if you are using face recognition to unlock your phone, it is better if the system fails to identify you a few times (false negative) than it is for the system to misidentify other people like you and let those people unlock your phone (false positive). If the result of a misidentification is that an innocent person goes to jail (like a misidentification in a mugshot database), then the system should be designed to have as few false positives as possible.

The approach I am going to use for face recognition is fairly straightforward. The key here is to get a deep neural network to produce a bunch of numbers that describe a face (known as face encodings). When you pass in two different images of the same person, the network should return similar outputs (i.e. closer numbers) for both images, whereas when you pass in images of two different people, the network should return very different outputs for the two images. This means that the neural network needs to be trained to automatically identify different features of faces and calculate numbers based on that. The output of the neural network can be thought of as an identifier for a particular person's face — if you pass in different images of the same person, the output of the neural network will be very similar/close, whereas if you pass in images of a different person, the output will be very different.

If we are trying to train a face recognition model using the typical deep learning algorithm, we need more than 10000 face image for each of person's we want to recognize. And having more than 1000 image for each of a student is really hard, time consuming and highly impractical. The secret is a technique called deep metric learning. A typical network training accepts a single input image and outputs a classification/label for that image. However, deep metric learning is different. Instead of trying to output a single label (or even the coordinates/bounding box of objects in an image), it instead outputting a real-valued feature vector or encoding features.

For the dlib facial recognition network, the output feature vector is 128-d (i.e., a list of 128 real-valued numbers) that is used to quantify the face. Training the network is done using triplets. The network quantifies the faces, constructing the 128-d embedding (quantification) for each. From there, the general idea is that we'll tweak the weights of our neural network so that the 128-d measurements of the two Will Ferrel will be closer to each other and farther from the measurements for Chad Smith. Our network architecture for face recognition is based on ResNet-34 [5], but with fewer layers and the number of filters reduced by half.

Thankfully, we don't have to go through the hassle of training or building our own neural network. We have access to a trained model through dlib and OpenCV that we can use. It does exactly what we need to do: outputs a bunch of numbers (face encodings) when we pass in the image of someone's face; comparing face encodings of faces from different images will tell us if someone's face matches with anyone we have images of.

## 4.4 Create a custom face recognition dataset

To build a custom face recognition system first we need to gather examples of faces we want to recognize and then quantify them in some manner. This process is typically referred to as facial recognition enrollment. We call it “enrollment” because we are “enrolling” and “registering” the user as an example person in our dataset and application. The method I use to enroll students is to use OpenCV and a webcam to detect faces in a video stream and save the example face images/frames to disk. This method to create your own custom face recognition dataset is appropriate when You are building an “on-site” face recognition system and you need to have physical access to a particular person to gather example images of their face

Such a system would be typical for companies, schools, or other organizations where people need to physically show up and attend every day.

To gather example face images of these people, we may escort them to a special room where a video camera is set up to detect the (x, y)-coordinates of their face in a video stream and write the frames containing their face to disk. We may even perform this process over multiple days to gather examples of their face in Different lighting conditions, Times of day, and Moods, and emotional states to create a more diverse set of images representative of that particular person’s face. I built a simple Python script to facilitate building our custom face recognition dataset.

This Python script will do the following:

1. Access our webcam
2. Detect faces
3. Write the frame containing the face to disk

## 4.5 Encoding the faces using OpenCV

Before we can recognize faces in images and videos, we first need to quantify the faces in our training set. Keep in mind that we are not actually training a network here — the network has already been trained to create 128-d embeddings on a dataset of 3 million images. Instead, it's easier to use the pre-trained network called FaceNet [13] and then use it to construct 128-d embeddings for each of the faces in our dataset. Then, during classification, we can use a simple k-NN model to make the final face classification.

I built a FaceEncoder class which will have access to face image path to be encoded and ResNet + SSD face detector caffee prototxt file and model weight path. To encode face first it will loop of the faces image path, then it will detect the face locations or bounding box of the image using caffee face detector model. After finding a face for a given image it will encode the face into a 128-d vector using face recognition module. Finally, it will serialize the face encoding and corresponding student name to disk, then it appends the student name to the student attendance spreadsheet.

## 4.6 Recognizing faces in video

After creating our 128-d face embeddings for each image in our dataset, we are now ready to recognize faces in an image using OpenCV, Python, and deep learning. The next step is to recognize faces in video streams or match faces. To match faces in the video frame to the register faces, first, we load the pre-computed encoding and face names. We then proceed to detect all faces in the input video and compute their 128-d encodings.

To implement the above steps I built a FaceRecognizer class which has two properties containing a path to the caffee face detector prototxt and model path and three methods called encodeFace, matchFace, and drawFaceBB.

The encodeFace method accepts the OpenCV formatted image as an argument, and then converts the image format from BGR to RGB to match the face recogniton module. Then it will detect faces in the input image or frame using caffee deep learning face detection. Finally, it will encode the faces using the encode face function from the face recognition module.

The matchFace method accepts the face encodings for the input image or frame and the encoding data for the register face with their corresponding name as an argument. It begins to loop over the face encodings computed from our input frame. And attempt to match each face in the input image encoding to our known encodings dataset held in data encodings using face recognition.compare faces. This function returns a list of True /False values, one for each image in our dataset.

Internally, the compare_faces function is computing the Euclidean distance between the candidate embedding and all faces in our dataset If the distance is below some tolerance (the smaller the tolerance, the more strict our facial recognition system will be) then we return True, indicating the faces match. Otherwise, if the distance is above the tolerance threshold we return False as the faces do not match. Essentially, we are utilizing a more fancy KNN model [4] for classification. Given our matches list, we can compute the number of "votes" for each name (number of True values associated with each name), tally up the votes, and select the person's name with the most corresponding votes.

The drawFaceBB method accepts the input image or frame, the face bounding box, and a list of enrolled persons' names as arguments. The first loop over the bounding boxes and labeled names for each person and draw them on our output image for visualization purposes. First, it uses the box coordinates to draw a green rectangle, and also uses the coordinates to calculate where we should draw the text for the person's name followed by actually placing the name text on the image.

## 4.7 User's Registration and Login

A user or teacher can be registered by providing his/her necessary information into the input fields. when the user clicks the signup button and if there is no empty required field, and the password and confirmation password matches, it will register the user to SQL database using SQLite python module.

SQLite3 [8] can be integrated with Python using the sqlite3 module, which was written by Gerhard Haring. It provides an SQL interface compliant with the DB-API 2.0 specification described by PEP 249. You do not need to install this module separately because it is shipped by default along with Python version 2.5.x onwards. To use the sqlite3 module, you must first create a connection object that represents the database, and then optionally you can create a cursor object, which will help you in executing all the SQL statements.

After a user successfully registered he/she can log in to the system by providing his/her username and password. and after his/her username and password verified using the SQLite database connection, he/she can proceed to the home page. From the home page, a user can choose either to register a new student by pressing the register student button or take student attendance by pressing the take attendance button or view student attendance results by pressing the view attendance button.

## 4.8 Register Students

A new student can be registered by entering his/her full name and capturing or uploading 10-20 face images of the student. There are two options to enter student face images. The first approach is by capturing a student's face using a webcam. As the webcam captures a video stream and sends it to the desktop app, face detection utility widgets will detect faces and draw their bounding box with their prediction confidence in the detected face image. After that, a user can save the image which contains the face image with higher or sufficient prediction confidence.

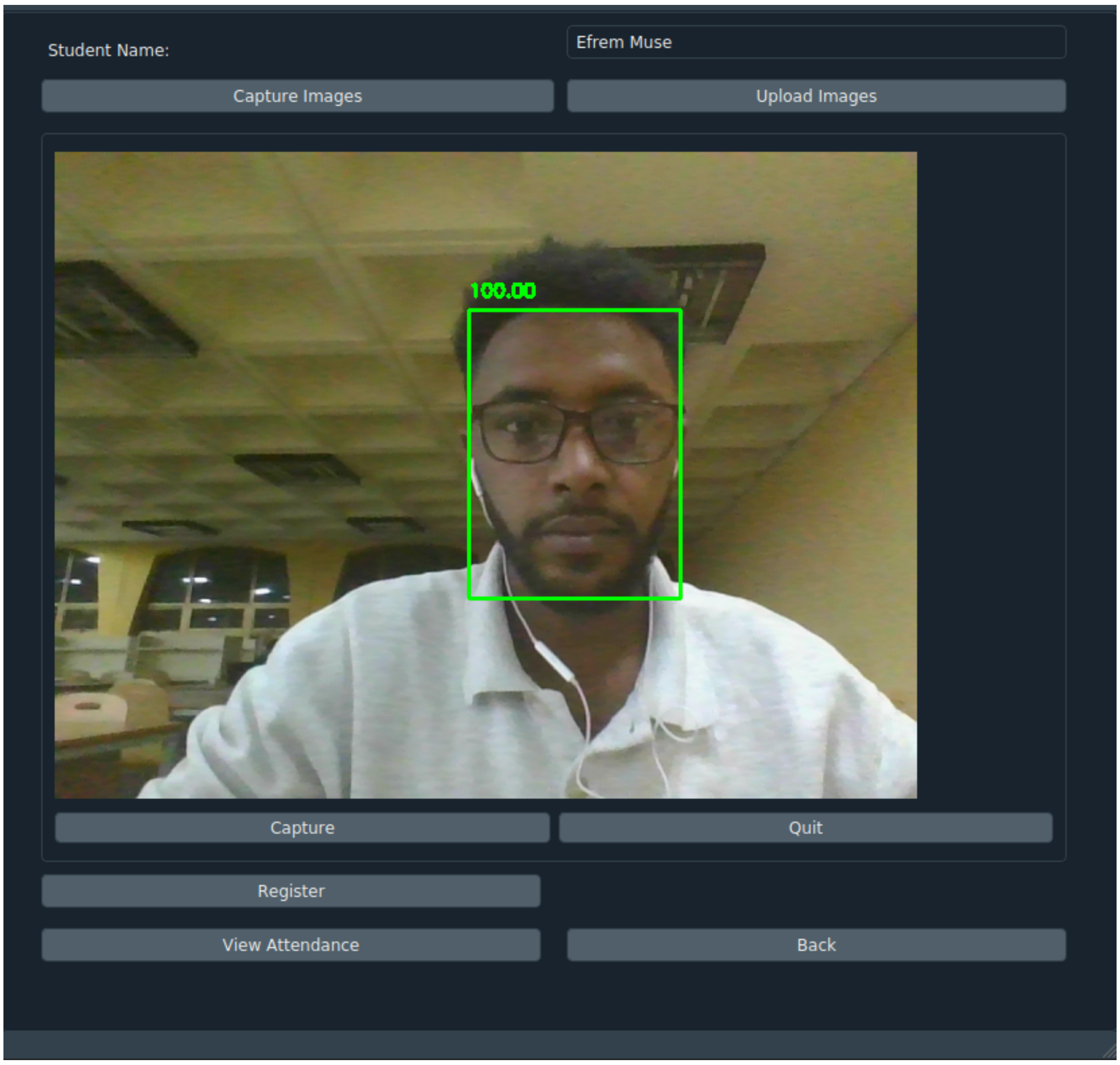

Student Name: Efrem Muse

Figure 4.1: register student

## 4.9 Take Attendance

To take student attendance we will apply face recognition using a face recognition module and face recognition widgets which i implement. The face recognition widget is connected to the take attendance page with signal and slot. So a face recognition system will be applied to each frame from the video stream or uploaded video. The webcam and face recognition system will run for about 5 minutes, and after that, it will stop recording and display the student name which is not detected in the video stream for cross-checking purposes.

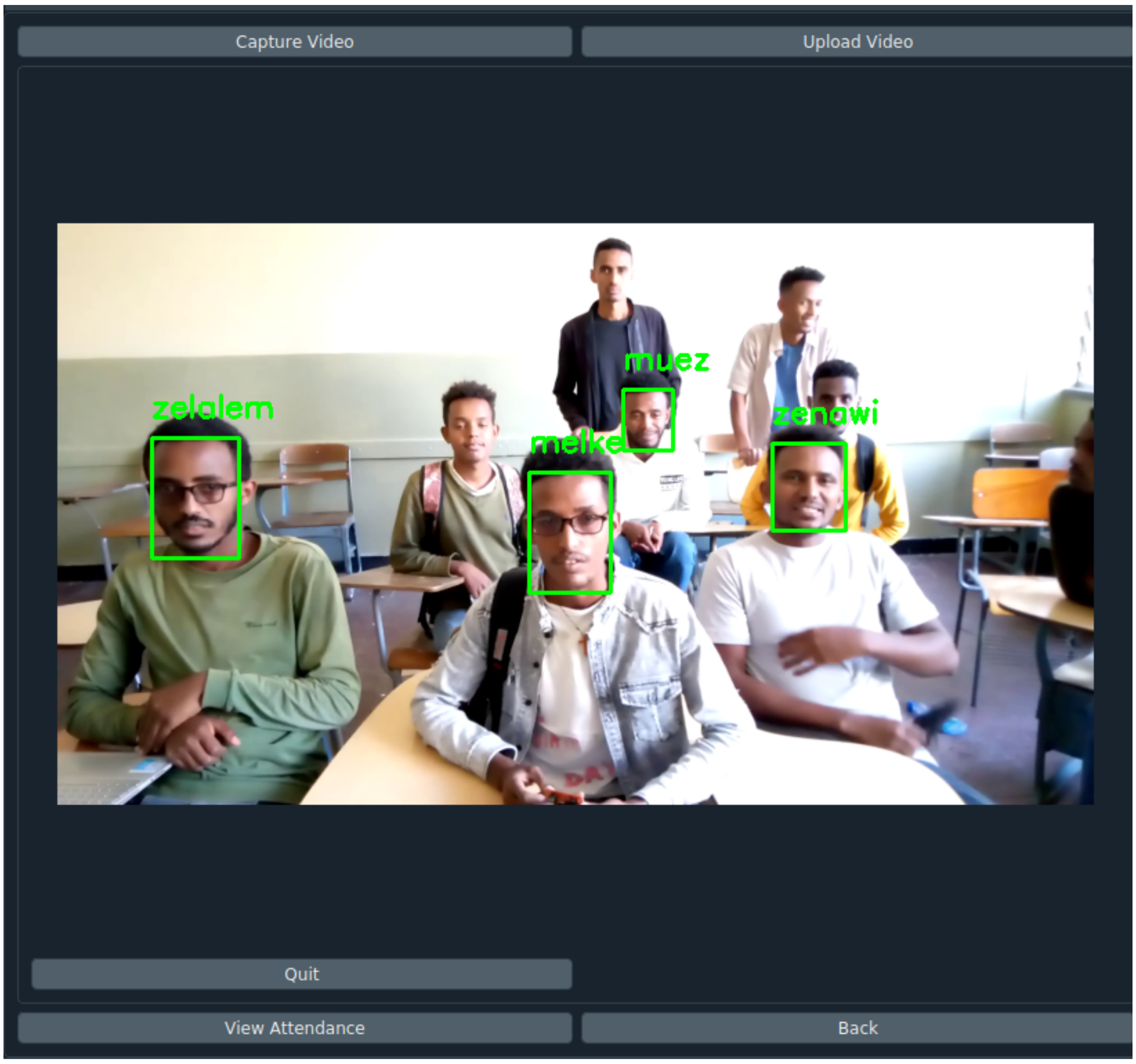
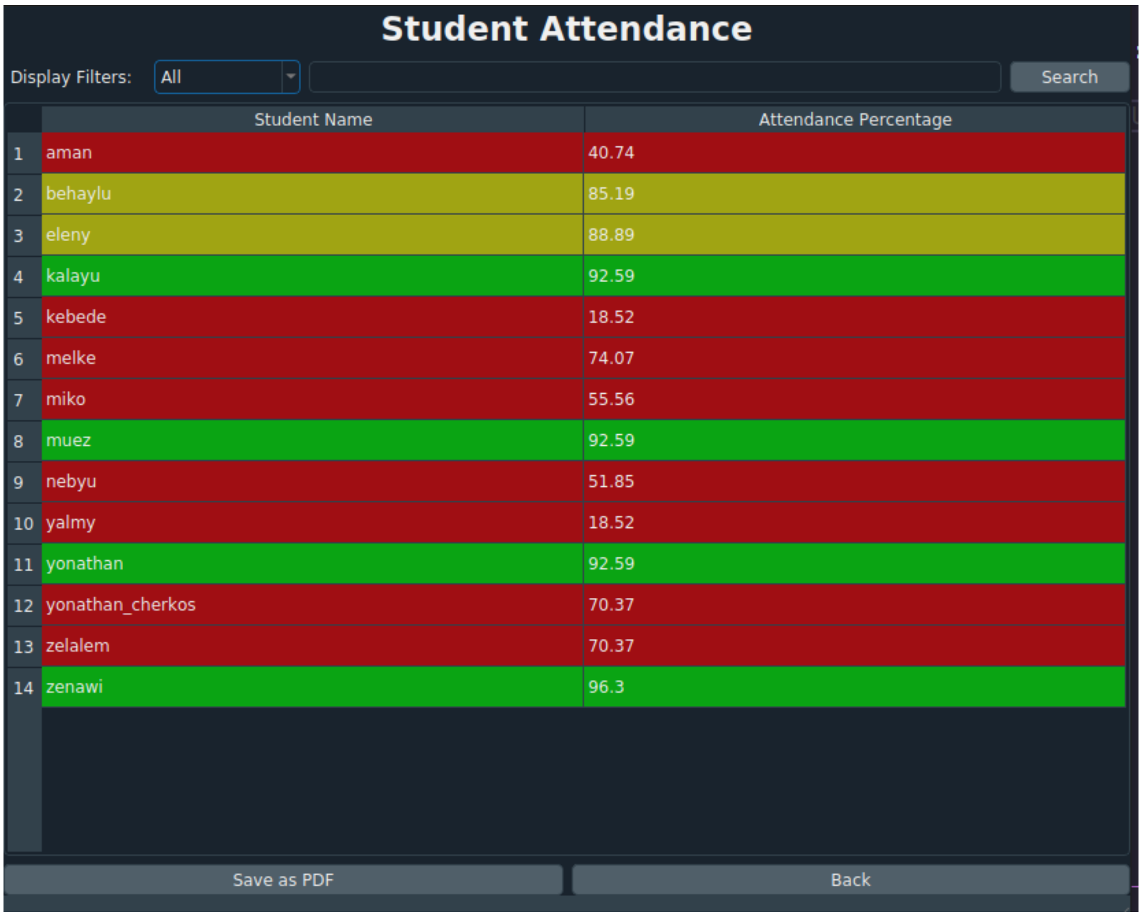


Figure 4.2: take attendance

## 4.10 View Attendance

A teacher can also view students' attendance results in table format. The table contains lists of the student names and corresponding student attendance percentages. The table rows are colored with different colors based on how much the student attends class. for example, if a student attends class less than 80% the row with the student's name will be colored red(danger).



| | Student Name | Attendance Percentage |
|---|---|---|
| 1 | aman | 40.74 |
| 2 | behaylu | 85.19 |
| 3 | eleny | 88.89 |
| 4 | kalayu | 92.59 |
| 5 | kebede | 18.52 |
| 6 | melke | 74.07 |
| 7 | miko | 55.56 |
| 8 | muez | 92.59 |
| 9 | nebyu | 51.85 |
| 10 | yalmy | 18.52 |
| 11 | yonathan | 92.59 |
| 12 | yonathan_cherkos | 70.37 |
| 13 | zelalem | 70.37 |
| 14 | zenawi | 96.3 |

Figure 4.3: view attendance

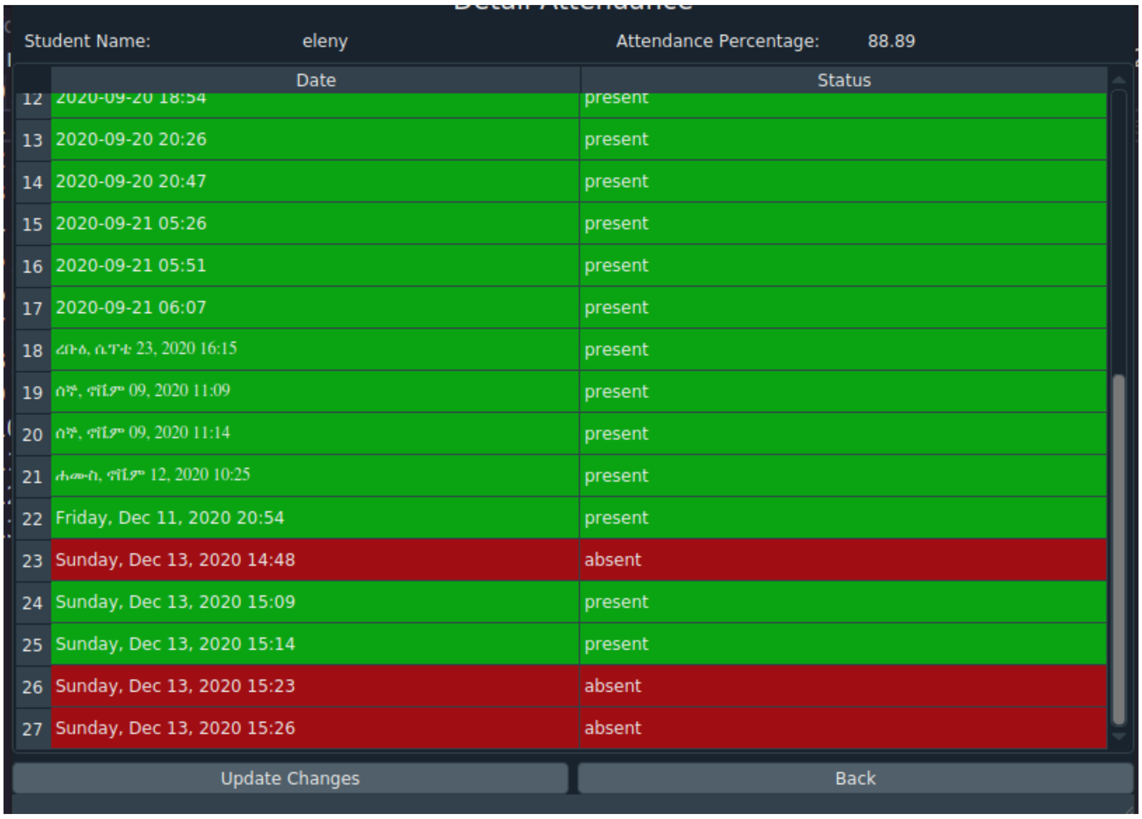
The view attendance page has different additional functionality. such as filtering, searching, and saving attendance results to pdf. The filtering option used to display students in different categories separately like good, warning, or danger. I implement this using pandas boolean indexing. There is also a searching option that allows to search for a specific student and display his/her attendance result. The save to pdf options are used to save the table to pdf format. When a user press saveToPdf a prompt screen will be displayed asking where to save the pdf file. After the user chooses the location to save the pdf file, the CSV file will be converted to pdf format and stored to disk by using a pdfkit module.

I implement this functionality by creating a ViewAttendance class which extends QMainWindow. The ViewAttendance class has properties that are used to connect the button to their corresponding methods. And it also has a property containing a path to the attendance CSV file.

The ViewAttendance also has about 7 methods. These are search, filter, displayTable, detailAttendance, saveAsPdf, back, and showDialog methods. The search methods search for the entered student name using pandas boolean index and return a data frame only to enter student attendance results. The filter methods also filter for students by indexing the data frame with different breakpoints. The detailAttendance method lets the user navigate to a student detail page when one of the rows a student double pressed.

## 4.11 Detail Attendance

The detailed attendance page contains detailed information about a given student. It shows the attendance result for the student for each class. And it also allows teachers to change student attendance results for a specific class if necessary. This has been implemented when a user(teacher) changes the attendance status for one or some of the class and press update change button, the data frame value will be replaced with the table widget data and the update the store data.



Student Name: eleny    Attendance Percentage: 88.89

| | Date | Status |
|---|---|---|
| 12 | 2020-09-20 18:54 | present |
| 13 | 2020-09-20 20:26 | present |
| 14 | 2020-09-20 20:47 | present |
| 15 | 2020-09-21 05:26 | present |
| 16 | 2020-09-21 05:51 | present |
| 17 | 2020-09-21 06:07 | present |
| 18 | ረቡዕ, ሴፕቴ 23, 2020 16:15 | present |
| 19 | ሰኞ, ኖቬም 09, 2020 11:09 | present |
| 20 | ሰኞ, ኖቬም 09, 2020 11:14 | present |
| 21 | ሐሙስ, ኖቬም 12, 2020 10:25 | present |
| 22 | Friday, Dec 11, 2020 20:54 | present |
| 23 | Sunday, Dec 13, 2020 14:48 | absent |
| 24 | Sunday, Dec 13, 2020 15:09 | present |
| 25 | Sunday, Dec 13, 2020 15:14 | present |
| 26 | Sunday, Dec 13, 2020 15:23 | absent |
| 27 | Sunday, Dec 13, 2020 15:26 | absent |

Figure 4.4: detail attendance

# Chapter 5

# Experiment and Result

## 5.1 Face Detection

I have experimented with the two most common face detection algorithms. The first is the most popular viola jones haar cascade face detector. It is very fast to run on a CPU and IoT device but it is less accurate. It fails to recognize faces at a little low lighting condition and different face orientations.

The second face detector algorithm is a deep learning base that has been constructed using ResNet + SSD. Since it uses a large deep learning network architecture and computational expansion object detector, it is a little smaller than the haar cascade face detector. But its ability to detect faces at different face orientations and lighting conditions is much better.

Face detection is the first step in any face recognition/verification pipeline. A face detection algorithm outputs the locations of all faces in a given input image, usually in the form of bounding boxes. A face detector needs to be robust to variations in pose, illumination, view-point, expression, scale, skin-color, some occlusions, disguises, make-up, etc. The app works with good accuracy even with the following variation. For example, it can recognize students with a face mask with good accuracy, even if there is no single example of their face with face mask on. It only uses the encoded face without face mask to recognize faces with face masks.

Since my requirement for the system is a face detector model with higher accuracy or less false positive face detection result and tolerable speed. After making some comparison experiments, I chose the deep learning-based face detector. And I use it when building the face dataset and recognize faces in live video streams.

## 5.2 Face Recognition

After registering some students I performed a different experiment with the face recognition feature of the app. I try to recognize registered students at different face orientations, distances, and lighting conditions. It is able to recognize the face successfully in almost all cases. It sometimes fails to recognize faces in low light conditions. But since the app will be used in classrooms which have a sufficient lighting condition with sunlight or lamp, it will work fine for its purpose.

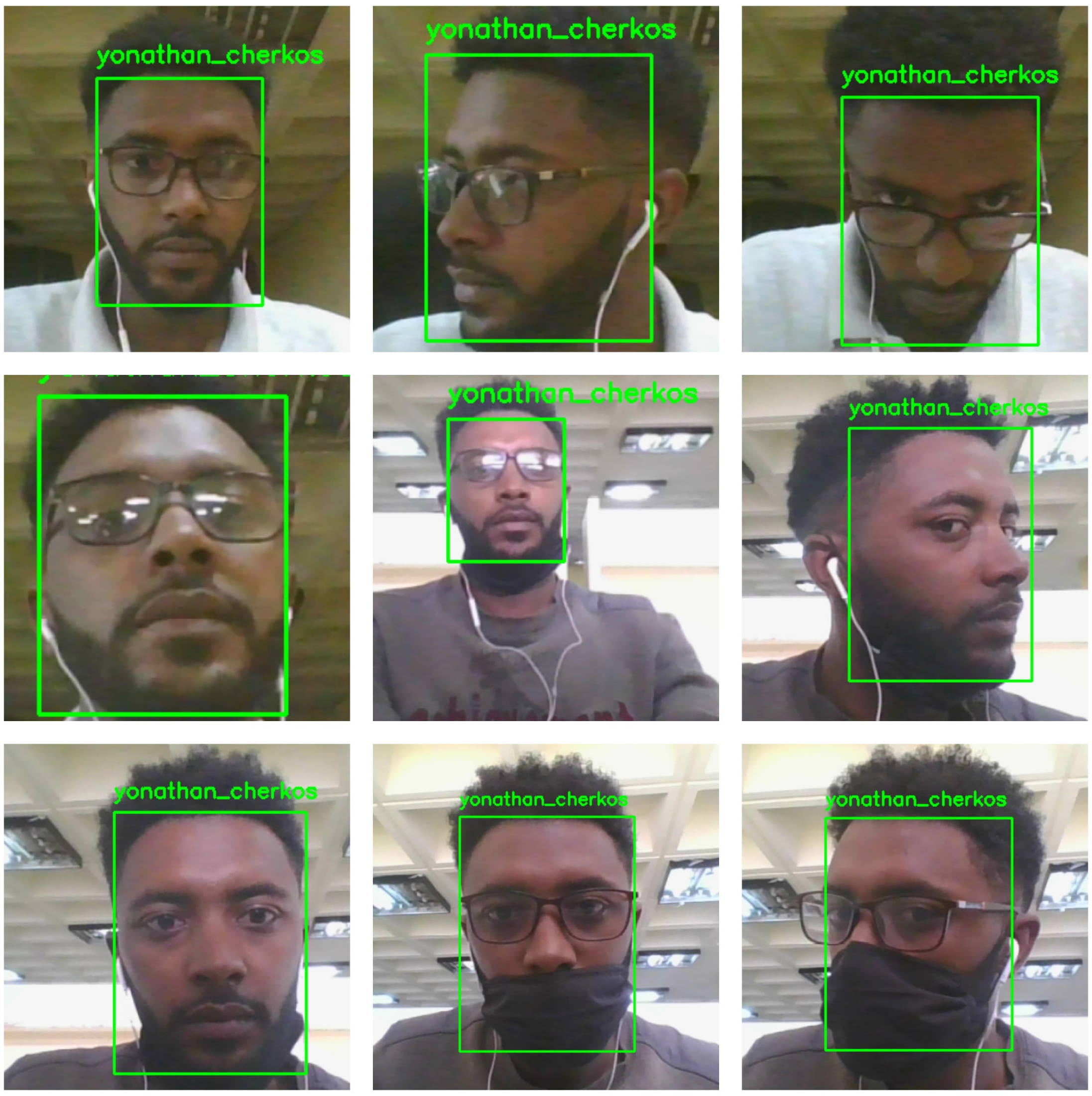


Figure 5.1: face recognition

# Chapter 6

# Conclusion

To conclude, A face recognition based attendance system doesn't require direct human interaction and also it is faster, so it is better than the other biometric systems such as fingerprint based attendance systems. The other thing is deep learning based face detectors have higher accuracy but they're a little bit slower than the haar cascade face detector. And for this project since recognizing student faces accurately is more useful than recognizing faces fast, I used a deep learning based face detector.

In further work, I intend to improve the system by integrating the entire face recognition modules into separate hardware modules which will do the whole face recognition functionally automatically at predefined time. And by adding a cloud database the attendance result will be sent directly from the hardware to the database. And beside having having only a desktop application, i intend to add a web app which can be accessed from any device at any time and anywhere.

# References

[1] Md Sajid Akbar, Pronob Sarker, Ahmad Tamim Mansoor, Abu Musa Al Ashray, and Jia Uddin. Face recognition and rfid verified attendance system. In *2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, pages 168–172. IEEE, 2018.

[2] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

[3] Jason Brownlee. A gentle introduction to computer vision. *Machine Learning Mastery*, 5, 2019.

[4] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. Knn model-based approach in classification. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 986–996. Springer, 2003.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] Michael Herrmann. Pyqt5 tutorial 2020: Create a gui with python and qt.

[7] Kennedy O Okokpujie, Etinosa Noma-Osaghae, Olatunji J Okesola, Samuel N John, and Okonigene Robert. Design and implementation of a student attendance system using iris biometric recognition. In *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 563–567. IEEE, 2017.

[8] Sayak Paul. sqlite in python.

[9] Sifatnur Rahman, Mahabur Rahman, Md Mijanur Rahman, et al. Automated student attendance system using fingerprint recognition. *Edelweiss applied science and technology*, 1(2):90–94, 2018.

[10] Hemantkumar Rathod, Yudhisthir Ware, Snehal Sane, Suresh Raulo, Vishal Pakhare, and Imdad A Rizvi. Automated attendance system using machine learning approach. In *2017 International Conference on Nascent Technologies in Engineering (ICNTE)*, pages 1–5. IEEE, 2017.

[11] A Rosebrock. Face detection with opencv and deep learning. *Web: https://www. pyimagesearch. com/2018/02/26/face-detectionwith-opencv-and-deep-learning*, 2018.

[12] A Rosebrock. Face recognition with opencv, python, and deep learning. In *Recuperado de https://www. pyimagesearch. com/2018/06/18/-face-recognition-with-opencvpython-and-deep-learning/Triantafillou, D., y Tefas, A.(2016). Face detection based on deep convolutional neural networks exploiting incremental facial part learning. 23rd International Conference on Pattern Recognition (ICPR)*, 2018.

[13] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[14] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001.