Final Project Report:
Face-Recognition Based Student Attendance System

Course: Intro to Artificial Intelligence (SOFE- 3720U)
Professor: Khalid Hafeez

Group # : 23
Group Members:
Esam Uddin - 100711116
Ashwin Shanmugam - 100700236
Mihir Patel - 100702168
Graem Sheppard - 100700978

Date: 3/30/2020

iAttend

A Face-Recognition Based Student Attendance System

Esam Uddin
Faculty of Engineering and Applied Science
Ontario Tech University
Oshawa, Canada
esam.uddin@ontariotechu.net

Ashwin Shanmugam
Faculty of Engineering and Applied Science
Ontario Tech University
Oshawa, Canada
ashwin.shanmugam@ontaritechu.net

Mihir Patel
Faculty of Engineering and Applied Science
Ontario Tech University
Oshawa, Canada
mihirkumar.patel@ontariotechu.net

Graem Sheppard
Faculty of Engineering and Applied Science
Ontario Tech University
Oshawa, Canada
graem.sheppard@ontariotechu.net

Abstract—**Face Recognition systems are crucial in almost every industry in this modern age. Face recognition is one of the most popularly used biometric systems and can be used for various purposes like authentication, security, identification, and many other applications. Furthermore, facial recognition technologies can be used to mark attendance in schools, universities, etc. Unlike the traditional attendance marking system which is a tedious process, using computer vision technology would make it effortless. To solve this real-world problem, we decided to make use of the OpenCV(Open Source Computer Vision) library including other face detection and recognition algorithms to create a face-recognition-based student attendance system. The Face recognition model could be set up by integrating a webcam that allows the student to take a picture and register for that particular course, and also provides them an option to mark the attendance if they are already registered. The student attendance system can also be tested under various conditions to eliminate any inconsistency in the program and improve its accuracy. It proves to be a robust application for marking attendance without performing tedious manual work. The application is cost-efficient and has several future applications to take into consideration.**

I. Introduction

A. *Task Definition*

With many institutions delivering instruction via online meeting platforms such as Zoom and Google Meets, there has been rapid advancement in video conferencing technology for teaching. Many schools require taking attendance of the students which uses either the instructor's or the students' valuable time. Moreover, the current attendance system would neither be user-friendly nor give professors insightful data on time. The purpose of this project is to overcome these limitations and alleviate the need for any logins or student ID cards by using facial recognition technology, taking only seconds for any student to register attendance. This will also reduce the class time spent taking attendance so that it may be used for instruction instead. If the project were to be successful, it would play a vital role in simplifying the current attendance taking process and incorporating facial recognition technology to manage attendance the intelligent way.

B. *Problem Solution*

To take the students' attendance automatically in video meetings, we will use facial recognition to compare against a database of pictures to identify the student. In doing this, it will not be necessary for the student to log in or fill out an attendance sheet manually. We will collect preliminary data to evaluate the performance of the face recognition system, for example, comparing the genuine acceptance rate, with the false acceptance rate. The picture captured by the student is stored in a folder which is then used to train the face recognition model. The attendance for the student is marked and stored on a CSV file with the student's name, number, date and time as soon as the face recognition model recognizes the face. Our proposed system takes only a couple seconds to recognize the students' face and create the attendance journal, saving precious teaching staff hours. The faces recognized by the system accurately enrolls students. It automatically scans multiple students, with 99.38% accuracy in face recognition.

C. Literature Review

There have been several researchers who have contributed to the field of attendance management and facial recognition. The main objective in most papers has been to improve the traditional attendance-based system and provide educators with a more feasible approach that is time-saving, accurate, and most importantly increases productivity for instruction.[4] The proposed system will integrate facial recognition technology to deliver precise face-detection algorithms using python's libraries like OpenCV, face_recogination and dlib. These libraries are constantly improved and developed to solve our problems more efficiently and precisely.[2]

P.Jonathan Phillip, a machine learning researcher, and his team introduced the method of using Support Vector Machines (SVM) and applied it to face recognition. This approach of recognition faces focuses on significantly improving the speed and efficiency of the process of comparing faces. They have also introduced the facial landmark estimation algorithm which is used to position faces in the frame.[5] It will help the program to train AI and improve the accuracy of recognition on the image.[9]

Another algorithm used for face recognition is the Viola-Jones algorithm. This algorithm first detects the face on the grayscale image to find the Haar cascade. Haar Cascade is an object detection algorithm used to identify faces in an image or a video.[11] This algorithm used a lined detection feature to identify a facial feature to identify a person. In this algorithm, the AI is trained with multiple sets of data to improve its accuracy and performance.[8]

D. Project Goals

To summarize, the goal of our project is to create a student attendance system integrated with facial recognition technology that can be used by the teaching staff to mark student attendance in a fast and effective manner. We aim to allow students to mark their attendance within seconds simply by opening the login window in the app. We also aim to gain practical experience by creating a software system that is based on a real world problem.

II. Infrastructure

The application was developed using Python as per the project requirements, Python 3 was chosen as it has more support than Python 2 and is the version of choice for most users. The development took place on computers running Windows because most users of the application will be Windows users, though Linux may be a better choice to obtain consistent experiences for all users.

The list of Python packages that were required for iAttend is relatively small, simplifying the process of getting the required versions and updating to newer package versions in the future. The packages used are listed and explained below:

A. CMake

Is a cross-platform open-source application used for automation, testing, packaging and the installation of software that employs a compiler-independent approach. It is consistent with directory hierarchies and programs that depend on several libraries. Used for building the project, since it is cross platform this would allow developers to build and test their project on Linux or macOS as well as Windows.[2]

B. DLib

Is an open-source library for implementing many machine learning algorithms and supporting functionality like threading and networking that are used by face-recognition packages. The dlib library can be used for detecting facial landmarks for example when used on an image it will generate 68(x,y) coordinates of the facial landmark region.[1]

C. Face-Recognition

This library is the bulk of the programmer's interaction, it combine's dlibs generalized machine learning functions into optimized functions for facial recognition. Finds and recognizes unknown faces in a given image based on data collected from known images.[5]

D. NumPy

Is a python library that adds support for large multidimensional arrays and matrices. It also contains large sets of high-level mathematical functions to run on the arrays.Python library required for working with images as byte arrays and many other numerical applications.[2]

E. OpenCV

Is an open source computer vision and machine learning software library. The library has more than 2500 algorithms, which can be used to detect and recognize faces, identify objects and etc. OpenCV contains functions for getting images from a webcam, encoding images for facial detection, and uses numpy to represent images as arrays.[5]

F. PyQt5

A popular framework for developing user interfaces for Python programs, includes a graphical designer for creating the UI. PyQt5 module can be used for developing multi-platform applications and is compatible with Windows, Linux and macOS . User Interface can be created manually or by using PyQt5 Designer module.[7]

III. Approach

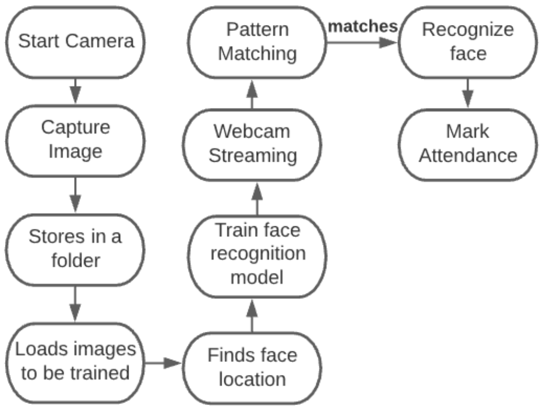


Figure 1.1 - System Architecture

The figure above represents the architecture of the face recognition model incorporated in the attendance system. The face recognition model involves the following three main stages:

A. Dataset Creation

When the student first opens the application, they can navigate to the register page where a webcam is used to capture a picture of the student. The student would need to enter their name and student number in the following format before clicking the ‘Capture’ button: “name_studentnumber”. The picture is then stored in a folder dedicated to storing all student images with their name and student number as the name of the file (name_studentnumber.jpg). The software is able to identify the name and id number so it only displays the name as it recognizes the face. The student number is added on to the attendance sheet when marking attendance. These pictures will be used to train the face recognition model by creating a deep convolutional network to generate 128 measurements for each face. The algorithm looks at each face added to the dataset folder and generates the measurements. It also makes sure that the measurement of multiple images of the same person taken in different angles is not varied extensively.

B. Facial Recognition

Once all students have registered successfully, their images should be saved in the specified folder. Our software then finds all the known images from the folder and stores them in a list which is used later to compare with the faces shown in the webcam. Using the face_locations function in the face_recognition package, we first find the faces shown on the webcam. The function returns an 2d array of bounding boxes of human faces in an image using the cnn face detector as shown in the figure below. “cnn” is a deep-learning model which is GPU/CUDA accelerated.

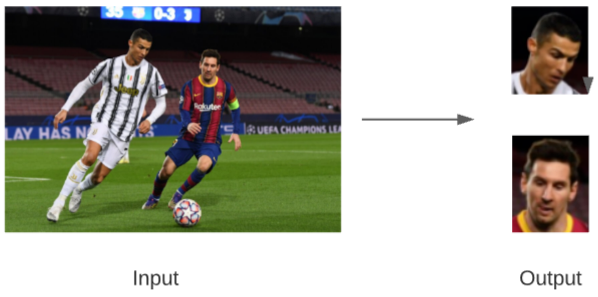


Figure 1.2 - Finding Faces

After finding the face location, we used the face_ecoding function which takes each image and returns the 128-dimension face encoding for each face in the image. We use the face encodings to compare the faces in the frame of video to the database of images that we stored previously. To actually compare the faces, we use the compare_faces function in the module which compares the list of face encodings against another encoding from the frame of video to check if they match. We also used the face_distance function from the module to get a euclidean distance for each comparison face and the distance would tell us how similar the faces are. This doesn't only give us a better indication of the match, but also allows us to simply return the smallest distance from the list of distances, the face with the match.

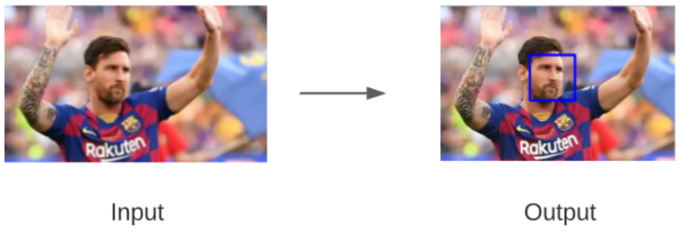
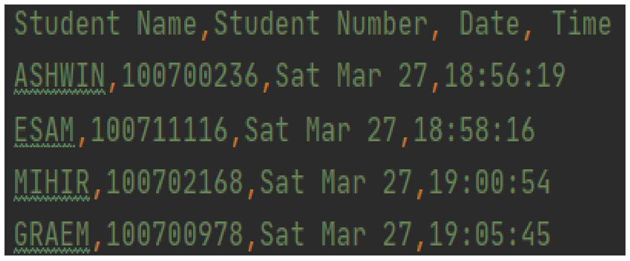


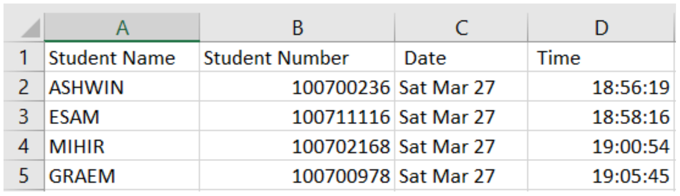
Figure 1.3 - Face Recognized

C. *Marking Attendance*

When the facial recognition process is completed, the software recognizes the face from the webcam through the list of known images and marks the attendance. With the help of the takeAttendance function in main.py file, we will be able to create an attendance sheet in the CSV file format and append the list of student's names and numbers along with the date and time to mark attendance. The student number and the student name will be passed as parameters and the function will only be called once the software finds a match. Due to the well-defined facial recognition model, the process of recognizing faces and marking attendance takes place rapidly with a high 99.38% accuracy. The following figure shows a sample attendance sheet generated upon running the application (attendance_list.csv).



```
Student Name,Student Number, Date, Time
ASHWIN,100700236,Sat Mar 27,18:56:19
ESAM,100711116,Sat Mar 27,18:58:16
MIHIR,100702168,Sat Mar 27,19:00:54
GRAEM,100700978,Sat Mar 27,19:05:45
```



	A	B	C	D
1	Student Name	Student Number	Date	Time
2	ASHWIN	100700236	Sat Mar 27	18:56:19
3	ESAM	100711116	Sat Mar 27	18:58:16
4	MIHIR	100702168	Sat Mar 27	19:00:54
5	GRAEM	100700978	Sat Mar 27	19:05:45

Figure 2.1- Attendance Sheet

IV. Data and Experiments

To familiarize ourselves with the technologies that we would be using for the program, we searched for similar projects that implemented facial recognition. There were multiple articles and videos explaining how to write a Python program using the Face-Recognition library as well as OpenCV, and many resources on creating a user interface using PyQt5.

A. *Facial Recognition*

The first project we created to learn the functions of OpenCV and Face-Recognition. It was a simple program with no user interface and no webcam input that could compare one image to another and return whether the faces were likely of the same person or not. The source file for this is also included in the project folder (test.py).

The next part of this project was to get an image from the user's webcam and compare it to images already saved into the training set. We learned how to get the face locations from an image and using OpenCV, we could draw a box around the face and, if recognized, display the name of the person that matches the image.

It was critical to understand how to use these library functions before we began work on the application to ensure that we could structure the program properly.

B. *User Interface*

We discovered that one common library for creating a user interface for Python programs is PyQt5. We started by creating a simple UI in the designer consisting of just a push button and some text. From the designer's UI, it is possible to convert it into Python code that can be executed. This allowed us to program the logic that occurs when the button is pressed, in this case the logic was to change the text shown to the user, however in the application it would be to capture the user's image and save it to the training set. Once we had completed this program, we had an understanding of both UI and facial recognition so we could begin programming the iAttend application.

C. Data

Many machine learning algorithms require large training sets to be able to function properly, when it comes to facial recognition this would mean we would require multiple pictures for each user and this would take up significant space. This problem is avoided by the face recognition library, because the images are encoded to be facing directly forwards, even if they are looking away from the webcam. As mentioned previously, the face_encodings function returns the 128-dimension face encoding for each face in the image. Therefore, our dataset only needs to consist of one picture per user.

V. Testing and Error Analysis

A. Challenges

One major challenge that was encountered during development was making OpenCV work with PyQt. The issue was with passing an OpenCV image to the UI. This had to be done by passing the image as a two dimensional byte array that could be converted into an image to display. This matter was further complicated by PyQt's skewing of the image. After some research and troubleshooting we discovered that PyQt has a parameter for the width of the image in bytes and that without this parameter it would guess the width and reassemble the bytes incorrectly.

Another challenge that we faced was the updating of the UI, in the original design, we looped over the function to recognize the user's face. This did not work with PyQt however because it blocked updates to the UI. To workaround this we implemented threading provided by the PyQt library which enabled the calculations to be done in the background and update the UI accordingly.

Another challenge that we faced was liveness detection. As of now, our application doesn't have the capability to ensure that no student uses a photograph or video to mark their attendance. We have worked on looking at alternative solutions to the problem, but due to the complexity of the issue, we are yet to integrate liveness detection to the iAttend application. A short term solution for this could be to screen capture the login window when the student opens their webcam and store all these screenshots in another folder. A faculty member can then access this folder to check for fraud attendance entries. For the long term, we are looking to enhance the fairness and reliability of the system.

B. *Performance Evaluation*

When performing error analysis on the face recognition model proposed in the system, it is realized that there are several factors that affect performance evaluation. The results of the performance evaluation depend on varying factors such as lighting, facial expressions, shooting time, angle, and distance. There are also environmental influences that make a difference including background features like color and shadow. All factors mentioned must be taken into serious consideration during the performance evaluation of the system once the images are saved in the specified folder.

One way to test this is to collect a database of all images that will be used in face recognition, and perhaps include images consisting of performance evaluation items such as lighting, posture, facial expressions, and etc. This database can be used to conduct performance evaluation measurements like false acceptance rate (FAR), false rejection rate (FRR), and equal error rate (ERR) which will give a strong indication of the performance and foster enhancement of reliability within the system.

VI. Results

A. Key Features

- Responsive User Interface: Our easy to use interface was created using the PyQt5 Graphical User Interface(GUI) python framework.

- Time Saving: Our facial recognition application takes only seconds to recognize faces and mark attendance.

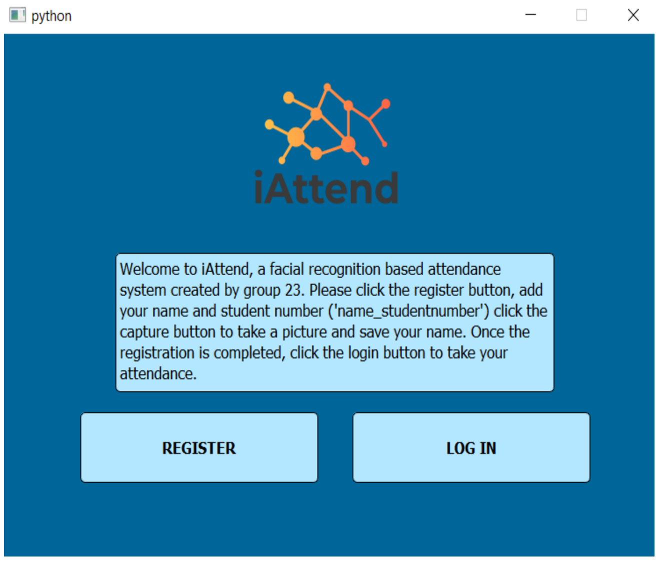
- Accuracy: Our well defined facial recognition algorithm with the use of OpenCV python module recognizes faces with a high accuracy of 99.38%.

- Offline Capability : Our application could be easily integrated with hardware devices like Arduino and many other devices for biometrics and security purposes.

B. *System Modules*

Home

The home window provides a short description about the purpose of our application and the students/users can start by either registering or logging in.

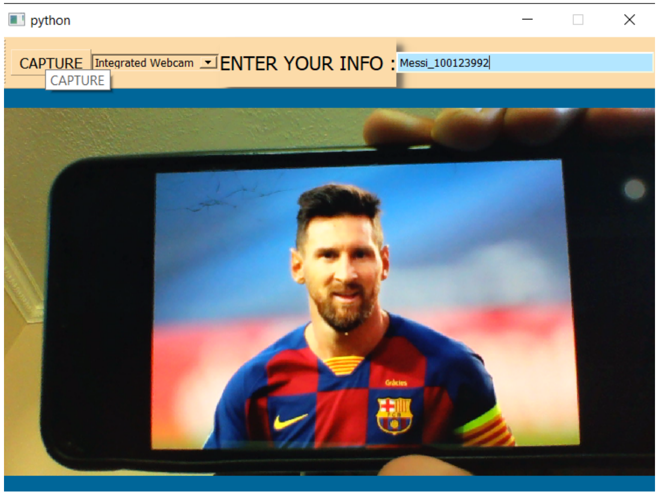


Welcome to iAttend, a facial recognition based attendance system created by group 23. Please click the register button, add your name and student number ('name_studentnumber') click the capture button to take a picture and save your name. Once the registration is completed, click the login button to take your attendance.

Figure 2.2- Home Window

Register

The register window allows the students to enter their information and capture their picture.



ENTER YOUR INFO : Messi_100123992

Figure 2.3- Register Window

Login

The recognition algorithm takes place in the login window, where it recognizes the student's face and draws a rectangle around the face labeled with the student's name.

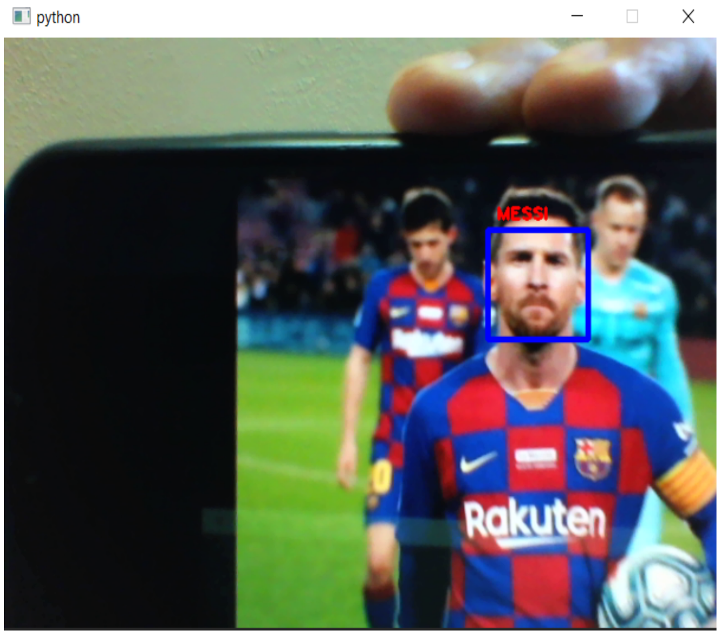
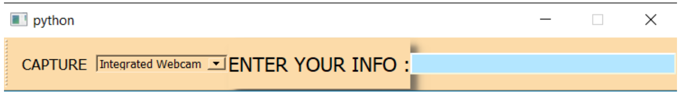


Figure 3.1- Login Window

Toolbar

The toolbar in the registration window contains a button to capture the student's picture, a dropdown menu that allows the user to select the type of webcam they would like to use, and lastly contains a text box providing a place for the student to enter their information.

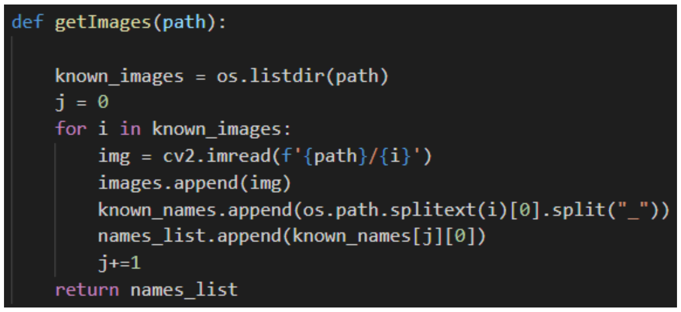


ENTER YOUR INFO :

Figure 3.2- Toolbar

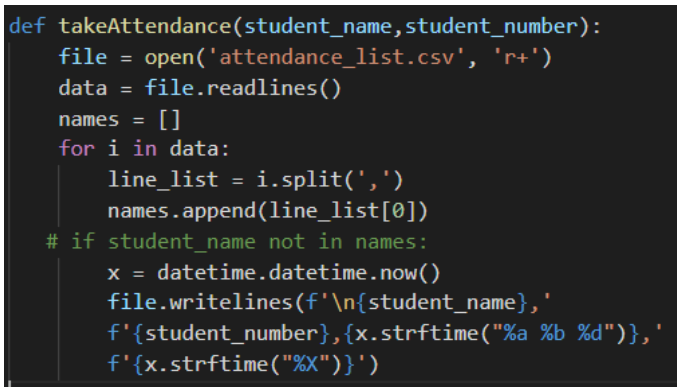
VII. Code Implementation

Following are the most important functions in the iAttend application. The getImages function finds all the images from the image database folder and returns a list of the names of the students which will be used to display it on the screen once recognized. The takeAttendance function is responsible for adding an entry in the attendance_list.csv file when the face is recognized. The getEncodings Function finds the encodings of all images and stores it in a list which is used by the compare_faces function to find a match.



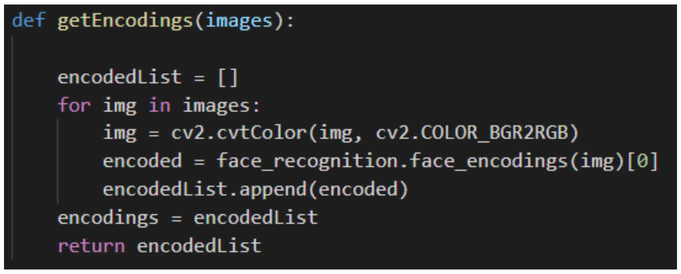
```python
def getImages(path):

    known_images = os.listdir(path)
    j = 0
    for i in known_images:
        img = cv2.imread(f'{path}/{i}')
        images.append(img)
        known_names.append(os.path.splitext(i)[0].split("_"))
        names_list.append(known_names[j][0])
        j+=1
    return names_list
```

Figure 3.3- getImages function



```python
def takeAttendance(student_name,student_number):
    file = open('attendance_list.csv', 'r+')
    data = file.readlines()
    names = []
    for i in data:
        line_list = i.split(',')
        names.append(line_list[0])
   # if student_name not in names:
        x = datetime.datetime.now()
        file.writelines(f'\n{student_name},'
        f'{student_number},{x.strftime("%a %b %d")},'
        f'{x.strftime("%X")}')
```

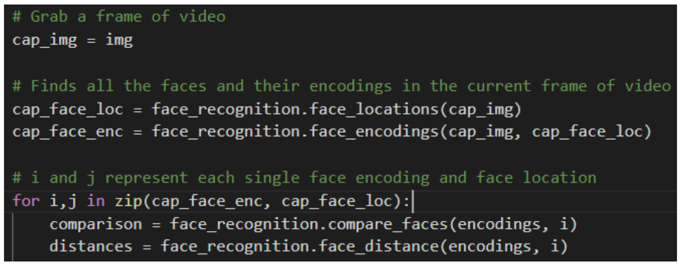
Figure 4.1- takeAttendance function



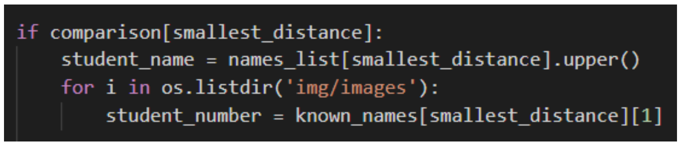
```python
def getEncodings(images):

    encodedList = []
    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encoded = face_recognition.face_encodings(img)[0]
        encodedList.append(encoded)
    encodings = encodedList
    return encodedList
```

Figure 4.2- getEncoding function

The following code snippet is taken from the face_recog function which opens up the webcam using the open cv2 module. As shown below, we first find the faces and then the encodings of the faces shown in the video stream. Then, we loop through each single face encoding and location to compare the faces and find the face distances. When the match is found, we display the name on the screen, set the appropriate student number, and call the takeAttendance function.



```python
# Grab a frame of video
cap_img = img

# Finds all the faces and their encodings in the current frame of video
cap_face_loc = face_recognition.face_locations(cap_img)
cap_face_enc = face_recognition.face_encodings(cap_img, cap_face_loc)

# i and j represent each single face encoding and face location
for i,j in zip(cap_face_enc, cap_face_loc):
    comparison = face_recognition.compare_faces(encodings, i)
    distances = face_recognition.face_distance(encodings, i)
```



```python
if comparison[smallest_distance]:
    student_name = names_list[smallest_distance].upper()
    for i in os.listdir('img/images'):
        student_number = known_names[smallest_distance][1]
```

VIII. Concluding Remarks

A. Conclusion

The iAttend application applied multiple concepts of artificial intelligence, including the more specific domain known as machine learning, in the form of a CNN. The interface to the CNN was provided by the Facial-Recognition library, allowing rapid development of an application to take attendance via webcam. The application demonstrated the ability to use artificial intelligence to increase productivity in classroom environments by reducing the amount of time spent taking attendance. Furthermore, experimentation on the application showed that it is able to successfully output the correct student information with a high success rate, even at different camera angles or under different lighting conditions.

It will be possible to run the attendance software on the server-side by sending the webcam image via HTTP for processing or alternatively, integrating the application into the institution's meeting software and processing the image on the client side. For further details, a readme document will be included on installation instructions as mentioned previously.

B. Future Application

In future works, we intend to improve the face recognition based attendance system to achieve the real-time detection of specific students and teachers in the surveillance premises. Instead of taking images and storing images in the database, we can use the recorded videos captured through surveillance to track people's movement around the campus.[2] However, video surveillance will only be maintained for a specified amount to record the images, because continuous recording puts constant load on the database and increases computational time which affects the system performance. We can also ensure no one uses a photograph or a video to record their attendance with the webcam and prevent fraud attendance entries. Furthermore, we aim to improve our face recognition algorithm to resolve unintentional changes in a person, for example, losing hair on your head and/or growing a beard. [9]

Automated attendance systems can also be implemented in larger areas of a university, for example, university libraries and gymnasium.[10] However, working with a large scale environment has external factors that will influence the performance such as poor light condition, which indirectly degrades the quality of the images and increases the error rate for correctly identifying the person. We can overcome this problem by improving the quality of video cameras or by using advanced algorithms to train the AI for detecting such errors.[3]

Acknowledgment

The authors would like to thank the professor and the teaching assistants for teaching related concepts during the lectures and tutorials, and guiding us throughout the course of the semester which helped to considerably improve the quality of both the iAttend application as well as the project report.

References

[1] "face-recognition," PyPI, 20-Feb-2020. [Online]. Available: https://pypi.org/project/face-recognition/. [Accessed: 12-Mar-2021].

[2] A. Geitgey, "Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning," Medium, 24-Sep-2020. [Online]. Available: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78. [Accessed: 28-Feb-2021]

[3] A. Ponnusamy, "CNN based face detector from dlib," Medium, 12-Sep-2018. [Online]. Available: https://towardsdatascience.com/cnn-based-face-detector-from-dlib-c3696195e01c.[Accessed: 06-Mar-2021].

[4] S. M. Bah and F. Ming, "An improved face recognition algorithm and its application in attendance management system," Array, 26-Dec-2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590005619300141#bib10. [Accessed: 10-Mar-2021]

[5] G. Guo, S. Z. Li, and K. L. Chan, "Support vector machines for face recognition," Image and Vision Computing, 31-Jul-2001. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0262885601000464. [Accessed: 10-Mar-2021].

[6] S. Dev and T. Patnaik, "Student Attendance System using Face Recognition," 2020 International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2020, pp. 90-96, doi: 10.1109/ICOSEC49089.2020.9215441.

[7] S. Bhattacharya, G. S. Nainala, P. Das and A. Routray, "Smart Attendance Monitoring System (SAMS): A Face Recognition Based Attendance System for Classroom Environment," *2018 IEEE 18th International Conference on Advanced Learning Technologies (ICALT)*, Mumbai, India, 2018, pp. 358-360, doi: 10.1109/ICALT.2018.00090.

[8] Davis King, "dlib," PyPI, 04-Dec-2020. [Online]. Available: https://pypi.org/project/dlib/. [Accessed: 12-Mar-2021].

[9] Riverbank Computing, "PyQt5," *PyPI*, 24-Nov-2020. [Online]. Available: https://pypi.org/project/PyQt5/. [Accessed: 12-Mar-2021].

[10] K. Okokpujie, E. Noma-Osaghae, S. John, K. Grace and I. Okokpujie, "A face recognition attendance system with GSM notification," 2017 IEEE 3rd International Conference on Electro-Technology for National Development (NIGERCON), Owerri, Nigeria, 2017, pp. 239-244, doi: 10.1109/NIGERCON.2017.8281895

[11] S. Lukas, A. R. Mitra, R. I. Desanti and D. Krisnadi, "Student attendance system in classroom using face recognition technique," *2016 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju, Korea (South), 2016, pp. 1032-1035, doi: 10.1109/ICTC.2016.7763360

[12] P. Wagh, R. Thakare, J. Chaudhari and S. Patil, "Attendance system based on face recognition using eigenfaces and PCA algorithms," 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), Greater Noida, India, 2015, pp. 303-308, doi: 10.1109/ICGCIoT.2015.