King Saud University

College of Computer and Information Sciences

Computer Engineering Department

Design and Implementation of an Attendance System Using Machine Learning

تصميم وتنفيذ نظام تحضير باستخدام تعلم الآلة

Abdullah Mana Theeb	Mohammed Nasser Alanazi	Abdullah Nasser Alhumaid
Student ID: 437106714	Student ID: 437100230	Student ID: 437102089
437106714@student.ksu.edu.sa	437100230@student.ksu.edu.sa	437102089@student.ksu.edu.sa

Advisor
Dr. Ayed AlQahtani

Submitted in partial fulfilment of the requirements

For the Degree of Bachelor of Science in the Department of Computer Engineering at the College of Computer and Information Sciences.

Riyadh, Kingdom of Saudi Arabia

April 2021

Abstract

In this project, we will design and implement an attendance system using Machine Learning (ML) on PC and mobile. The lecturer should use a mobile phone or PC with an external camera to take a picture of the attendees, then the system detects faces in the taken image and recognizes those faces. The solution exports the attendance sheet in a formatted way saved on the lecturer's PC or mobile. The results after testing both implementations show that the implementation on PC is significantly faster than mobile when increasing the number of students to take their attendance. The mobile implementation costs less to deploy compared to PC, while both implementations are easier to use and faster than manual attendance taking.

الخلاصة

في هذا المشروع سنقوم بتصميم وتنفيذ نظام تحضير طلاب باستخدام تعلم الآلة على الكمبيوتر والهاتف المحمول. على المحاضر أن يستخدم هاتفه المحمول أو جهاز الكمبيوتر مزوداً بكاميرا خارجية لأخذ صورة للحضور، ومن ثمَ النظام سيحدد الوجوه الموجودة في الصورة التي تم التقاطها و من ثمَ يتم التعرف على الوجوه التي تم تحديدها. النظام يصدر ملف الحضور بطريقة منسقة محفوظة على جهاز الكمبيوتر الخاص بالمحاضر أو هاتفه. تُظهر النتائج بعد اختبار كلا التطبيقين أن التنفيذ على جهاز الكمبيوتر أسرع بشكل ملحوظ من الهاتف المحمول عند زيادة عدد الطلاب الذين سيتم تحضيرهم. تكلفة تنفيذ التطبيق للهاتف المحمول أقل مقارنة بالكمبيوتر الشخصي نظراً للحاجة إلى كاميرا خارجية وكمبيوتر مزود بوحدة معالجة رسومية. في حين أن كلا التطبيقين أسهل في الاستخدام و أسرع من عملية التحضير اليدوي.

Key words: Attendance System, Machine Learning, Face Detection, Face Identification, Excel Sheet Manipulation.

Acknowledgements

In the name of Allah, Most Gracious, Most Merciful

We would like to express our sincere gratitude to all those who have given us the chance to complete our project. Firstly **Dr. Ayed AlQahtani** who gave us the chance to do this amazing project while following-up on our work step by step monitoring, suggesting, and commenting in every weekly meeting.

We will not forget to thank **Dr. Yakoub Bazi** for the outstanding course on Artificial Intelligence, **Dr. Abdulmohsen Almutairi** for the seminar and design courses since it was a grateful help to our project. **Dr. Abdulwadood Abdulwaheed** for applying the knowledge that we have learned from ''Digital Image Processing'' that helped us greatly in our project, **Computer Engineering Department** headed by **Dr. Saleh Almowuena** for all support. Lastly **King Saud University (KSU)** for facilitating our access to IEEE and Mendeley.

April 2021

Table of contents

List of Figures

List of Tables

List of Abbreviations

A.M.	Advisor Meeting
ANN	Artificial Neural Network
CV	Computer Vision
CNN	Convolution Neural Network
DB	Data Base
GUI	Graphical User Interface
KSU	King Saud University
KNN	K-Nearest Neighbours
ML	Machine Learning
NMS	Non-Maximum Suppression
O-net	Output Network
PID	Process ID
P-net	Proposal Network
R-net	Refine Network
SDAIA	Saudi Data and Artificial Intelligence Authority
MTCNN	Multi-Task Cascade Convolutional Neural Network

Chapter 1: Introduction

1.1 Introduction

In this project we aim to build an attendance system that employs Machine Learning (ML) to take the attendance from a picture of the students in the class. We will take the bottom-up strategy to design this system by dividing the top-level system into low-level modules designing each low-level module separately, then integrate them to build our complete system.

In this chapter the problem statement of our capstone project is described in Section 1.2 containing detailed information of the project requirement and its objectives, the Engineering Requirements of this project are mentioned in Section 1.3. Followed by the tasks undertaken in Section 1.4 titled Phase I and Phase II in section 1.5.

The motivation behind such project is mentioned in Section 1.6, lastly in Section 1.7 the report organization where we will describe the skeleton of this report and how it is organized.

1.2 Problem Statement

Faculty members in King Saud University (KSU) take students' attendance either manually or by using the “Hader” system, which is obsolete starting in semester 421.

Manual attendance usually consumes time, also needs careful observation, and can take too much effort. Even though the Hader system is faster compared to the manual choice, it is complex, and it depends largely on the students’ behavior. Some students can exploit the attendance system by falsely reporting their attendance state as “present” when they are “absent”, leaving attendance integrity to be questioned.

1.2.1 Objectives

Design and implement an attendance system that is easy to use. The system will reside on the lecturer's PC and or the lecturer's Android phone to automatically detect and recognize faces in a picture given to the system using ML, then create or update an Excel sheet in PC and mobile with the attendance and export it to the user.

1.2.2 Marketing Requirements

A vital part at the start of this project is to identify the project requirements. These requirements ensure project deliverability and act as the reference for the project completion. They also have a straightforward vision of what must be completed regardless of how it can be done, as the latter would be expressed more fully in the concept and execution chapters [1].

1- The user takes a picture, to take the attendance.

2- The system detects the faces in the taken picture.

3- The system recognizes faces in the taken picture.

4- The system updates and export an attendance Excel sheet for the PC.

5- The system updates an attendance table for the mobile.

6- The system maintains the attendance integrity.

7- The system should be easy to use.

8- The system should be fast.

1.2.3 Objective Tree

To guide the project design and identify key project goals In Figure 1.1 an objective tree outlining the three main objectives of this project while the branches are the means achieve the objective.

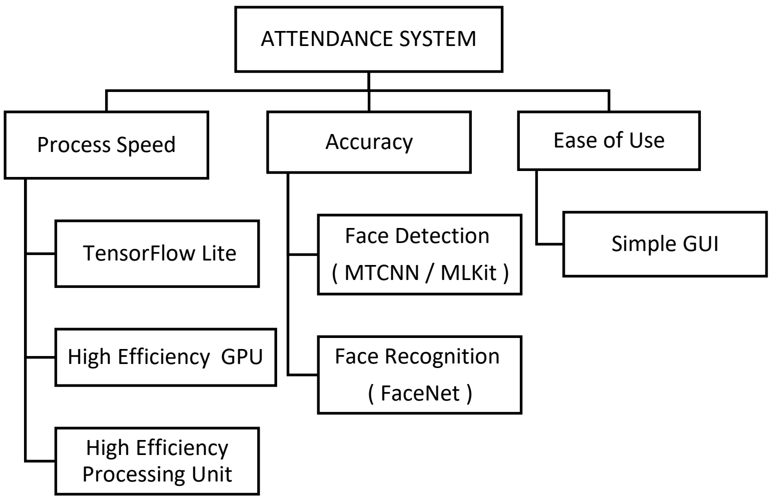


Figure 1.1 Project Objective Tree

1.3 Engineering Requirements

Engineering requirements are the technical needs for a successful project application. In Table 1-1 we have an engineering requirement table for this project each row shows the marketing requirements on the left column the middle column shows the relative engineering requirement that needs to be satisfied to ensure achieving marketing requirements the last column shows the justification behind each engineering requirement. These engineering requirements should be met in the design and implementation phases [1].

Table 1-1 Engineering Requirements Table

Marketing Requirement	Engineering Requirement	Justification
1,2,3,8	The system face detection and recognition time should be less than 50 seconds.	To eliminate time waste from the lecture.
1,2,3,4,8	The system should operate on a Windows PC with a minimum RAM of 8 GB and a minimum GPU of 4GB memory.	To process graphical pictures, we need to employ minimum system requirements.
1,2,3,4,8	The system should operate on an Android phone with a minimum RAM of 4 GB.	To process graphical pictures, we need to employ minimum system requirements.
4,5,6	The system should export attendance in Excel sheet (xlsx) or span an attendance table view.	For keeping records and Edugate integration.
6,7	The system should have only the Lecturer as a user.	To eliminate time waste and to enhance security by eliminating the students as a user that controls attending.

1.4 Phase I

In the first phase of our project we will focus on the analysis and the design of the system, research the different methods of face detection and face recognition, and build each module proposed in design chapter 4 separately.

1.5 Phase II

In the second phase of our project we will focus on the implementation of the system on both environments, then setup a prober benchmarking test for the PC and mobile implementations in testing weeks, lastly give an objective conclusion based on testing for the feasibility and promise of the whole project.

1.6 Motivation

The attendance system project using ML presents an advanced challenge to learn and work in the artificial intelligence field while having a motivational purpose in our academic life where current attendance systems fall short on many sides such as time delay, complexity, and security by eliminating the currently opposed issue of students sharing “Hader” system barcode with other students outside the class leading to false attendance. We will also get to work on ML where it is being used in wide areas and focused on by our own country creating Saudi Data and Artificial Intelligence Authority (SDAIA) for a better automated artificial future. Engaging in this project will help us to be creative in this domain before graduating raising our collective knowledge. Achieving this final step in our bachelor’s academic life is very important to us. Thus, choosing a project that is productive, challenging, being used in modern research, and personally appealing.

1.7 Report Organization

In this chapter, we have explained our project description and purpose, the motivation behind the project, and the overall objectives. Literature review where we will setup a knowledge base on ML is included in chapter 2. In chapter 3 we will explore face detection using MTCNN. chapter 4 will discuss face recognition using FaceNet.

The project proposed design and implementation are included in chapter 5 and chapter 6 respectively. In chapter 7 we will illustrate the testing of our implementation. Lastly in chapter 8 conclusion of our project.

Chapter 2: Literature Review

2.1 Introduction

This chapter will contain a brief explanation of various methodologies of neural networks techniques. It will also contain mathematical background for artificial neural networks.

2.2 Artificial Neural Networks

Artificial Neural Networks (ANN) are computational techniques designed to simulate the way the human brain performs a specific task, by manipulating huge parallel distributions and made of simple processing units which are nothing but mathematical elements they are called neurons or nodes, which have a neuronal property in that they store practical knowledge and empirical information to make it available to the user by adjusting weights. So, ANN is like the human brain in that it acquires knowledge by training and stores this knowledge using connecting forces within neurons called synaptic weights [2].

2.3 Components of ANN

Just as a person has input units that connect him to the outside world, also neural networks need input units (input layer) and processing units (processing layer). Interconnections in between these layers contains weights that leads an input to a certain output and by adjusting these weights we obtain the appropriate reaction for each of the inputs to the network. [2].

2.3.1 Description of Neural Networks and Neurons

A neural network can consist of several layers, each layer has a weighted W array, neurons in the first layer (Input layer), neurons in the second layer, and so on. It is also noticeable that the output of each layer is an input to the next layer. The layer that gives the output is called the output layer, while the input is not considered to be a layer, and the rest of the layers are called the hidden layers. Feedforward neural network is one of the most important types of multi-layered neural networks, and it is so-called because it adopts the principle of forward propagation where the output of all the neurons of the m layer is the input of each neuron in the layer in front of it m + 1. In Figure 2.1 we have an example of a neural network architecture.

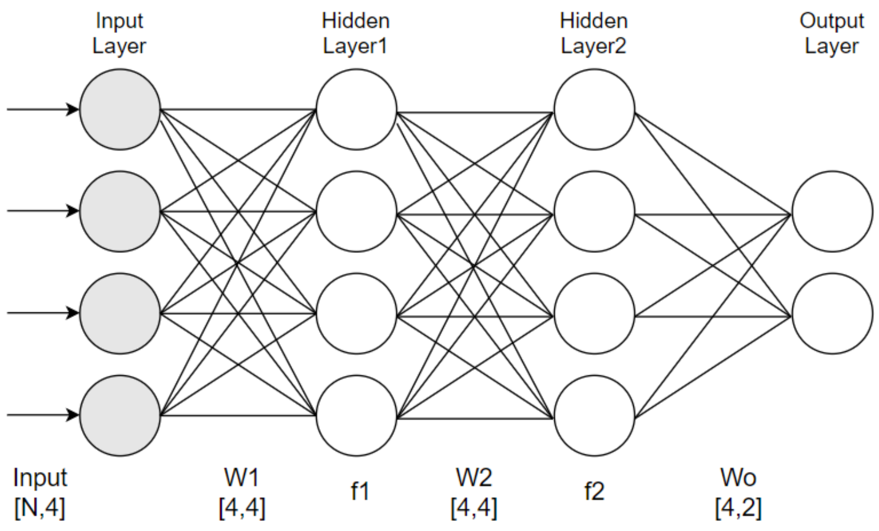


Figure 2.1 Neural Network Architecture

ANN consist of nodes or what we have previously mentioned as neurons or elements processing units, connected to form a network of nodes, and each connection between these nodes has a set of values called weights that contribute to determining the values resulting from each processing element based on the values entered for this element [2], In Figure 2.2 a diagram showing single neuron processing element.

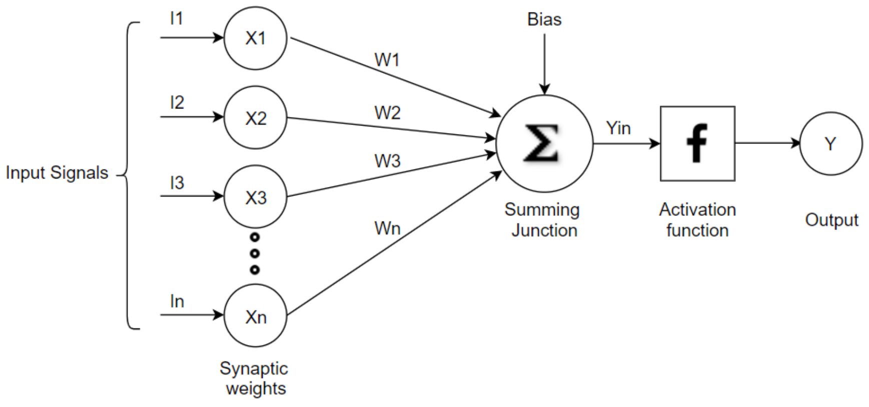


Figure 2.2 Schematic Diagram of a Single Processing Element Containing a Neuron

2.4 Training of ANNs

Learning or training the ANNs is the process of obtaining an output approximately equal to the required output, and this is done by adjusting network parameters such as weights. There are several types of networks whose characteristics differ depending on the number of layers, such as single-layer networks and multi-layer networks as well as on the type of methods used in training them such as Supervised and Unsupervised learning [3].

2.4.1 Supervised Learning of ANNs

This method of training is used to teach single-layer linear networks that are used to solve linear mapping problems between input and output, where the network calculates the error signal through the difference between the neuron's output and the required output, and the values of the weights are adjusted by the error function to reduce the difference to the synaptic weights. The training phase needs special data that the network learns by providing it with input data alongside the desired output data, and then the network makes a feed-forward of the input data to obtain the value of the network output, then it makes a comparison between the calculated output and the desired output. If the results do not match the network calculates the value of the difference between them for each neuron from the output layer, which represents the error value. Finally, the weight update stage comes, where the network recalculates all weights and replaces them with the new calculated values [3].

2.4.2 Unsupervised learning of ANNs

This training category uses an input vector only without presenting the target on the network, and this method is called self-learning where artificial neural networks build learning methods based on their ability to discover the characteristics of what is presented to them in terms of shapes and formats and their ability to develop an internal representation of these shapes without prior knowledge (showing examples of what it should produce as an output), in contrast to the principle followed in the learning method by a Supervised learning method [3].

2.5 Computer Vision (CV)

Computer Vision is one of the very important areas in deep learning [4], whose importance is increasing day by day, whether in Mobile or Desktop applications, or security implementations in airports or surveillance cameras. There are several CV use cases, such as:

- The classification of objects in images: the algorithm identifies a specific object in the input image.
- Face detection: the algorithm determines whether there is a face in the input image and specifies its location by placing a square around it.

Among the most important difficulties with CV is the size of data, If we deal with a small RGB image of (50 pixels in length and width), the number of overall pixel data will be (50 * 50 * 3) = 7500 feature, which is an acceptable number of inputs, where the image is very small. But if we deal with a clear image of 1000 pixels in length and width, it will lead to having 3 million features. The idea of CNN is based on a scanning process of images and applying certain filters, where each of them designed to have a specific purpose. One of the most important processes is called 2d convolution.

2.5.1 2D Convolution

2D Convolution applies the concept of filtering to manipulate an image. Since images are composed of a matrix, 2D Convolution is used by the given formula:

$$\mathrm{y}[\mathrm{i},\mathrm{j}] = \sum_{\mathrm{m}=-\infty}^{\infty}\sum_{\mathrm{n}=-\infty}^{\infty} \mathrm{h}[\mathrm{m},\mathrm{n}]\cdot \mathrm{x}[\mathrm{i}-\mathrm{m},\mathrm{j}-\mathrm{n}]$$

Where x being the input image, h is the kernel [5].

Chapter 3: Face Detection

3.1 Introduction

In this chapter, we will explore the method of face detection in ML, explain the framework built on top of it. Then present a conclusion.

3.2 Method of Face Detection

There exists a wide verity of frameworks to detect human facial properties. While researching for a candidate methodology for our project we discovered the following:

3.2.1 Multi-Task Cascade Convolutional Neural Network (MTCNN)

MTCNN uses Convolutional Neural Networks (CNN) in the framework which has multiple benefits like the ability to train and supervise the framework to enhance the accuracy. In figure 3.1 an example of MTCNN showcasing input to output behavior.

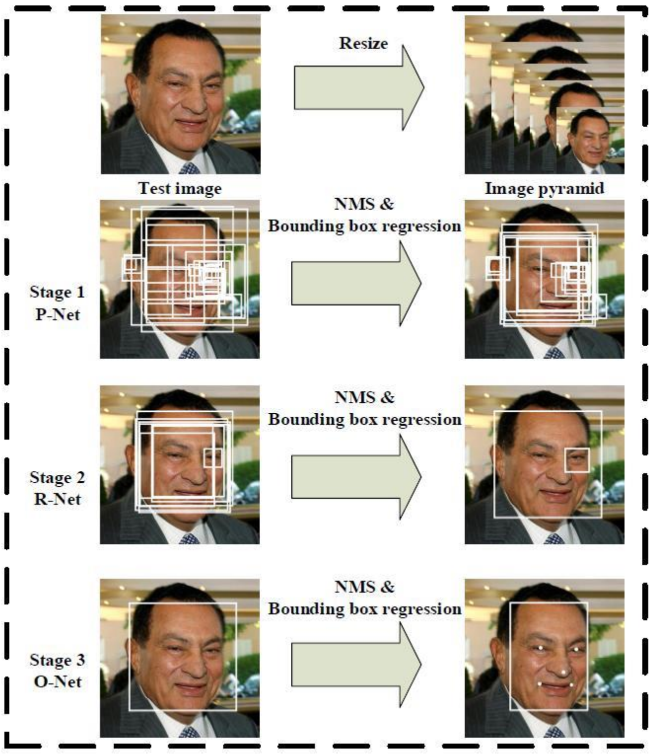


Figure 3.1 MTCNN Input to Output Example

MTCNN can be used for both face detection and alignment, firstly it resizes the input image into a pyramid then it puts it in the following three stages:[6]

- Stage 1 - Proposal Network (P-net):

The first stage is a fully connected Neural Network it takes the input picture and gives a face candidate and their bounding box regression vectors, then it uses non-maximum suppression (NMS) to reduce the number of bounding boxes.

The module for P-net is demonstrated in Figure 3.2.

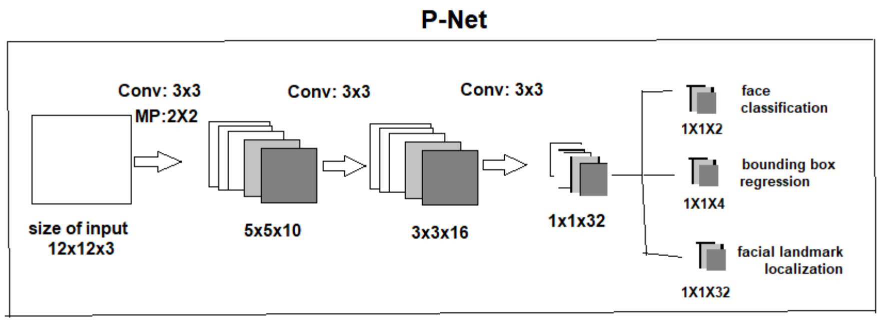


Figure 3.2 Proposal Network

- Stage 2 - Refine Network (R-net):

All face candidates from stage one P-net CNN is taken as an input through the same process of calibration using the bounding box regression and NMS to further reject false facial candidates. R-net is demonstrated in Figure 3.3.

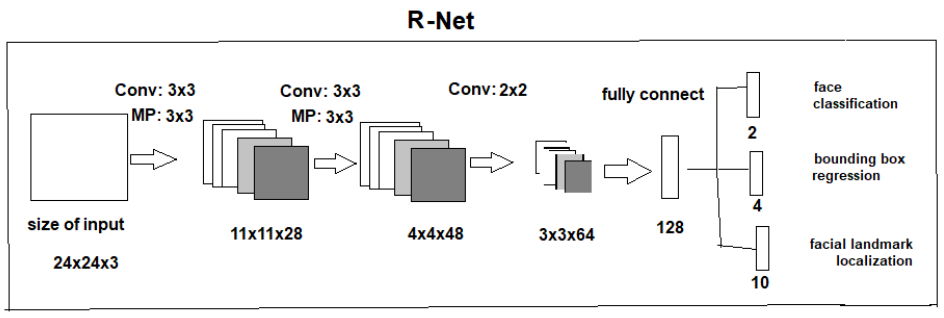


Figure 3.3 Refine Network

- Stage 3 - (O-net):

Similarly, the final stage only here the output is much refined and outputs five facial landmarks. It is showed in Figure 3.4.

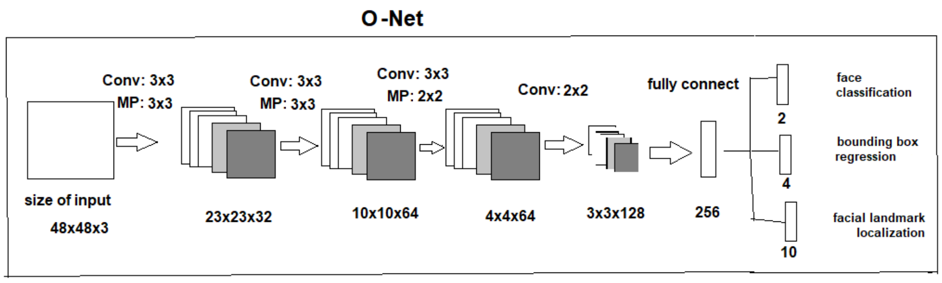


Figure 3.4 Output Network

- Classification and regression:

In Table 3-1 there are the calculations of face classification where cross entropy is used as Loss of function, bounding box regression uses Euler distance''L2 loss'' , and facial landmark localization also L2 loss is used as a loss function : [7]

Table 3-1 Classification and Regression

	Mathematical Definition	Description
face classification	$L_i^{det} = -(L_i^{det}\log(pi)) + (1 - L_i^{det})(1 - L_i^{det})$	P_i stands for the probability. L_i^{det} ∈ 0, 1 is the ground-truth label of the faces.
bounding box regression	$L_i^{box} = \|\|y^{\wedge}{}_i^{box} - y_i^{box}\|\|_2^2$	L_i^{box} : the regression goal produced by the network. y_i^{box} : the ground-truth location.
facial landmark localization	$L_i^{landmark} = \|\|y^{\wedge}{}_i^{landmark} - y_i^{landmark}\|\|_2^2$	$y^{\wedge}{}_i^{landmark}$: location and place of facial landmark. $y_i^{landmark}$: the ground-truth location.

This weighted sum of these three loss values is used as the loss for back propagation. The weights in the P-Net and R-net are fixed, boxes regression and landmark localization are 1.0 in P-Net, 0.5 in R-net, and 0.5 in R-net. The last stage weights are 1.0 in P-Net, 0.5 in R-net, and 1.0 in R-net.

3.3 Conclusion

At the end of this chapter, we learned about face detection. We learned about face detection and how it works. We have chosen MTCNN. It is suitable for our project needs. MTCNN gives us high accuracy and a positive rate performance adhering to the requirements mentioned in chapter 1.

Chapter 4: Face Recognition

4.1 Introduction

Our main objective of this project is to identify faces in the class, thus needing a face recognition system. While there exist a substantial number of frameworks that achieve this objective most depend on deep learning. In this chapter we will go through a suitable framework for our project.

4.2 FaceNet

FaceNet is a deep neural network to recognize human faces by working with Euclidean Space and extract an output embedding (128 vector output) in Figure 4.1 an example showing input to output in FaceNet. Output embeddings represent how two faces are alike. FaceNet employs multiple layers such as convolutions, normalizations, max-pooling layers, and non-linear activations [8].

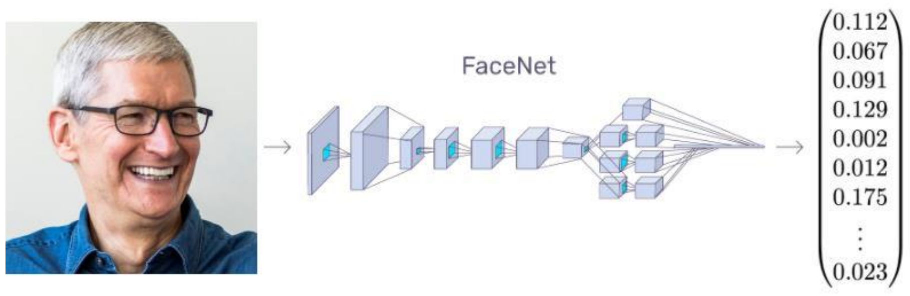


Figure 4.1 FaceNet Input to Output Example

4.2.1 Architecture

FaceNet determines person similarity in a picture based on finding for each image an Euclidean embedding through a CNN. It is trained such that the squared value of L2 (the output normalization) layer corresponds directly to persons similarity. Ideally the same person has low distance between embeddings, while different people have a larger distance between their embeddings.

The input representing an image is processed through a CNN. The output is an Euclidean embedding as we described earlier. When we place a different picture with the same person in the CNN, the squared distance between the two embeddings will be small. A threshold value of error checks whether this error value is small.

In Figure 4.2 an arbitrary example of face recognition by calculating the deference between output embeddings an operation called Euclidian distance (instead of the output embeddings the input image is used in the figure).

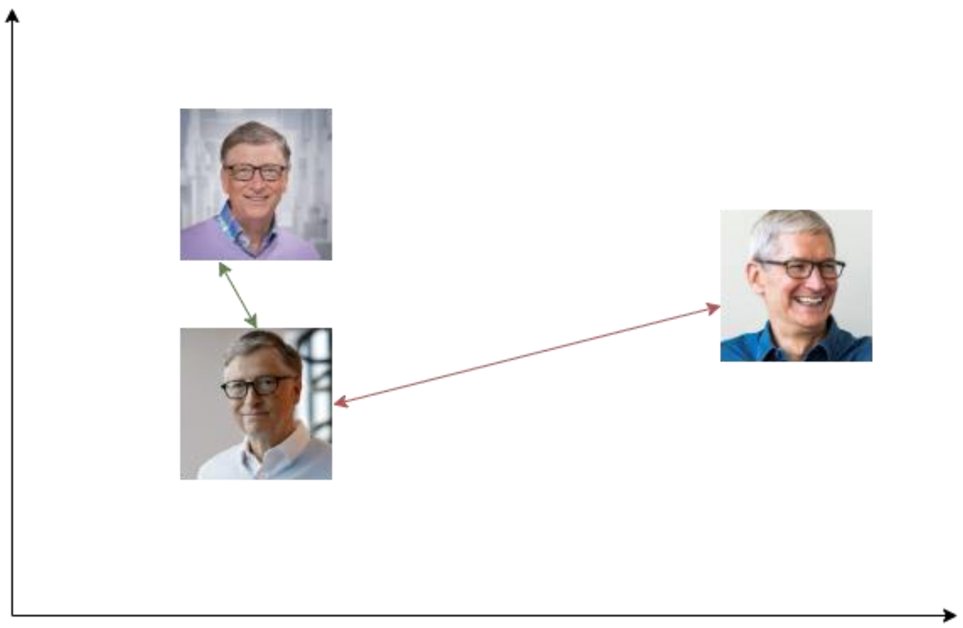


Figure 4.2 Recognition by Calculating Embedding Distances

This is done by FaceNet through training the CNN where a batch of the same person in different pictures is the input FaceNet will output a k-nearest neighbors (KNN) classifier (which is a 128-D embedding). KNN rule is one of the most established and easiest strategies for designing classifiers. All things considered, it regularly yields serious outcomes in specific domains (especially in Euclidean space), when joined with earlier learned information [9].

4.3 Conclusion

In this chapter we had an overview on the current method of face recognition, to conclude we have decided to work with FaceNet framework being the latest in this field and has an unmatched accuracy, as face recognition part of our project is critical and needs to be highly accurate.

Chapter 5: System Design

5.1 Introduction

In this chapter we will design the overall system, starting with the proposed system design, after that we will discuss the user use case diagram, then we will have the Desktop Graphical User Interface (GUI) design, and lastly the mobile (GUI).

5.2 Proposed System Design

In Figure 5.1 the key modules while approaching this system in modular design are shown.

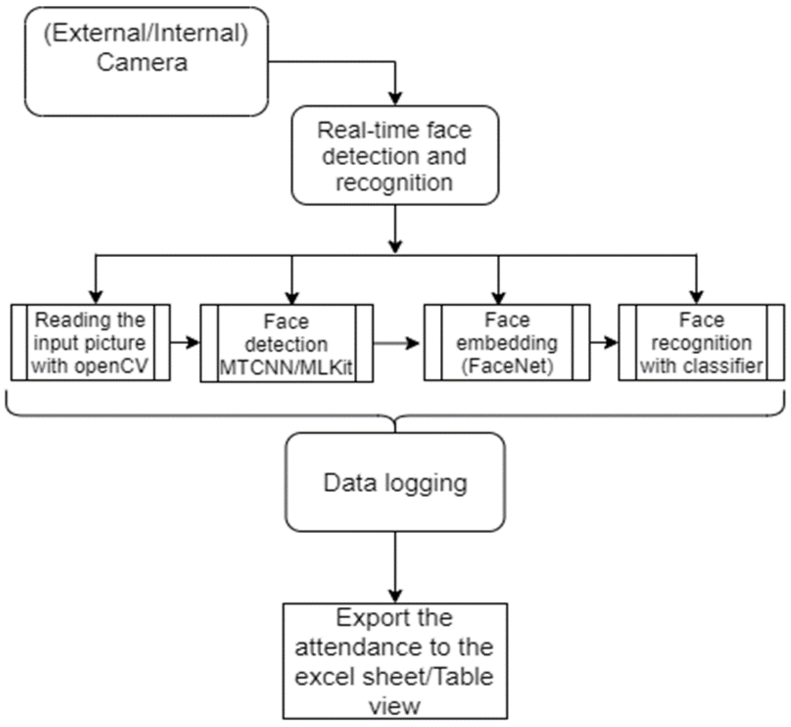
1- External camera module.

2- Image reading and manipulation module.

3- Face detection module.

4- Face recognition modules.

5- Data logging module.



(External/Internal) Camera
Real-time face detection and recognition
Reading the input picture with openCV
Face detection MTCNN/MLKit
Face embedding (FaceNet)
Face recognition with classifier
Data logging
Export the attendance to the excel sheet/Table view

Figure 5.1 System Modular Design

5.2.1 External Camera Module

The objective of the external camera module is to take the attendance pictures. After deliberations to find the suitable camera, that will ensure a wide field of view to take an image of all students in a class the chosen candidate was a GoPro Hero3+ camera.

Our way to communicate with the chosen camera through python and Wi-Fi, will be through an open source python library called “goprocam”.

5.2.2 Image Reading and Manipulation Module

To read and manipulate images we will use the long standing and will documented OpenCV python library, since it is the most widely used when it comes to operations on images, OpenCV contains a large set of algorithms to achieve computer vision, such as Viola-Jones face detection algorithm [10].

Our main need for OpenCV is to read the image taken from the external camera module.

5.2.3 Face Detection Module

After going through the method mentioned in chapter 3, the chosen algorithm for Face detection was MTCNN, for the accuracy advantage [11].

MTCNN will be implemented through TensorFlow python library, to detect faces in the picture after reading it using the previous module and output a set of pictures of the faces detected.

5.2.4 Face Recognition Module

The chosen framework for face recognition and verification is FaceNet, due to the high performance and accuracy of FaceNet in this field [12].

To verify the images outputted from the previous module we need a reference database of student pictures, in the following structure:

SectionID/StudentID-1.jpg

SectionID/StudentID-2.jpg

…

We also need the output from the face detection module:

```
Ready/FACE-1.jpg

Ready/FACE-2.jpg

...
```

This module will iterate over above-mentioned inputs and apply FaceNet through TensorFlow. The output of this operation will be Face Embedding for each picture in two lists, Reference Embeddings List (REL), and Attended Embeddings List (AEL).

Then we iterate over the lists in the following manner:

```
for i in REL do:
        for j in AEL do:
                if ((AEL[j]- REL[i])$^2$<= Threshold)) do:
                        output.add(REL[i].STUDENTID)
```

The output will be a list of student IDs recognized.

5.2.5 Data Logging Module

Attendance data logging will be done through “openpyxl” [13] an open source python library to read, edit, and write on an Excel files.

The input to this module is the output from Face recognition module, which is a list of student IDs that the previous module recognized. This module will iterate over the attendance Excel sheet and take the attendance of the given students, while taking into consideration the current date.

5.3 Graphical User Interface

We aim to build a simple and easy to use system, while most of the system functionality happens in the backend, some depends on the user input, thus we have to build a way to interact with the user. We have chosen to build a Desktop GUI which consists of the following screens:

5.3.1 Login Screen

The user will input his Email and Password that exists in our DB, there are two text input objects, and one button to log in as shown in Figure 5.2.

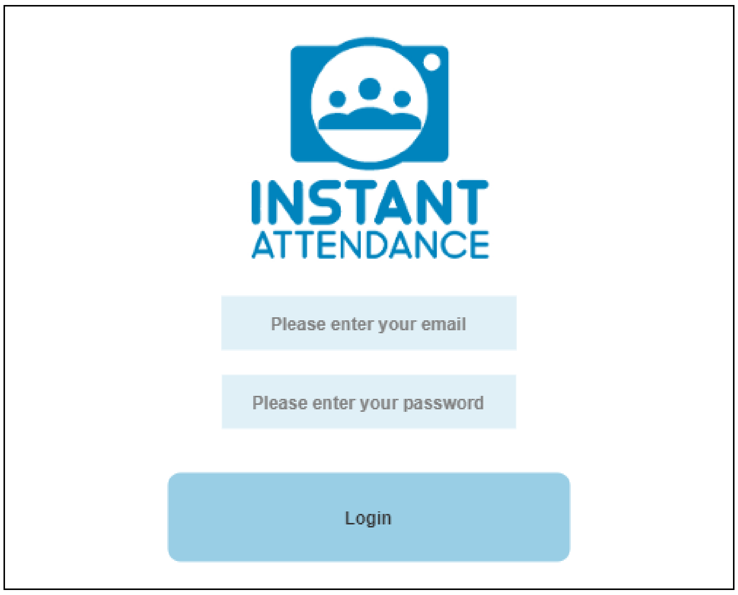


Figure 5.2 Log In Screen

5.3.2 Home Screen

After the user successfully logged in, the system will get the user information, and greet the user with the following screen which consists of a toolbar at the top, today's date and time, and a list view of the user's sections as shown in Figure 5.3.

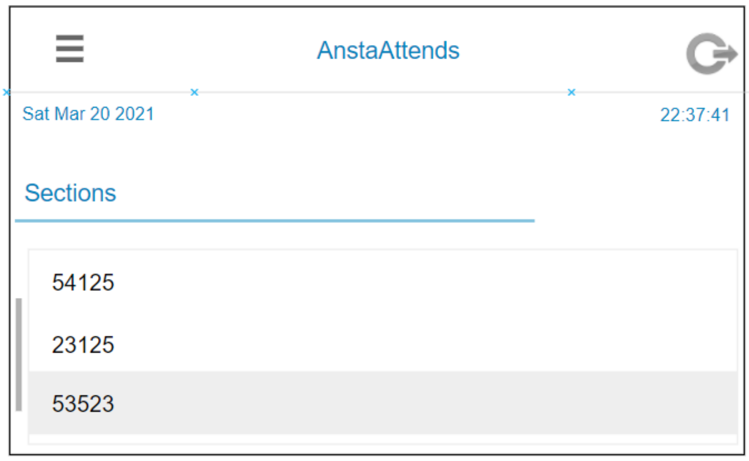


Figure 5.3 Home Screen

5.3.3 Camera View Screen

If the user has clicked on a section from the list view portion, the system will change the screen to the following screen is shown in Figure 5.4, which consists of three buttons. Run the camera will start the camera and display the view box of the camera. Take the picture will take the picture currently on the view box of the camera view box. Done will proceed to take the attendance from the picture.

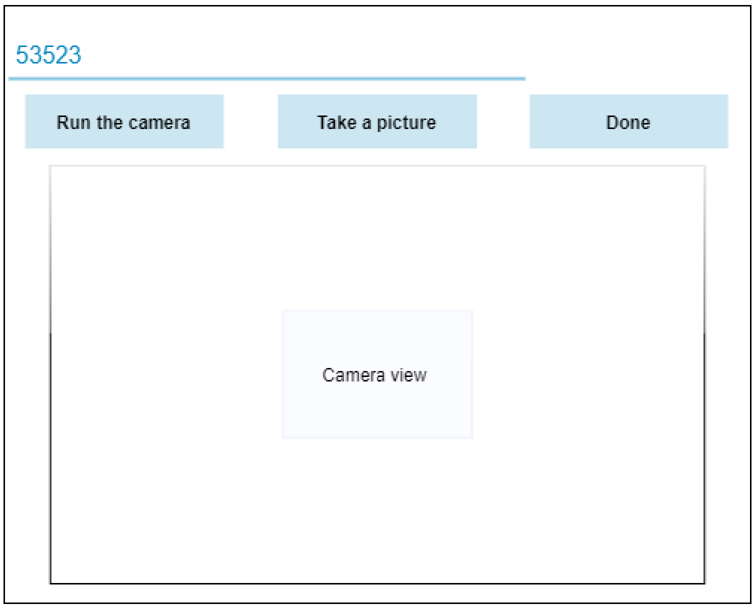


Figure 5.4 Camera View Screen

5.3.4 Export Screen

After the face detection, recognition, and excel updating modules have ended, the user gets an Export screen is shown in Figure 5.5, where the user has the choice of returning to the home screen to take the attendance again, or view the updated Excel sheet in Microsoft Excel.

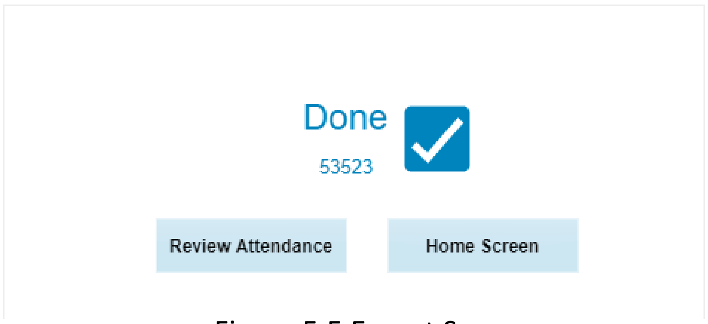


Figure 5.5 Export Screen

5.4 Mobile Application

The design on mobile phone is slightly different from what we have seen on the desktop environment, the difference is that we will change the last module to be displayed as a table rather than an Excel sheet. Figure 5.6 shows the main screens of the mobile application:

- Sign-Up Screen: The faculty member should be able to sign up to use the system.
- Sign-In Screen: To synchronize information about the given user.
- Main Screen: shows the list of sections for the user with the ability for each to take a new attendance or view the attendance history, the main screen also includes an add course functionality as a (+) button which will move the user to the add course screen.

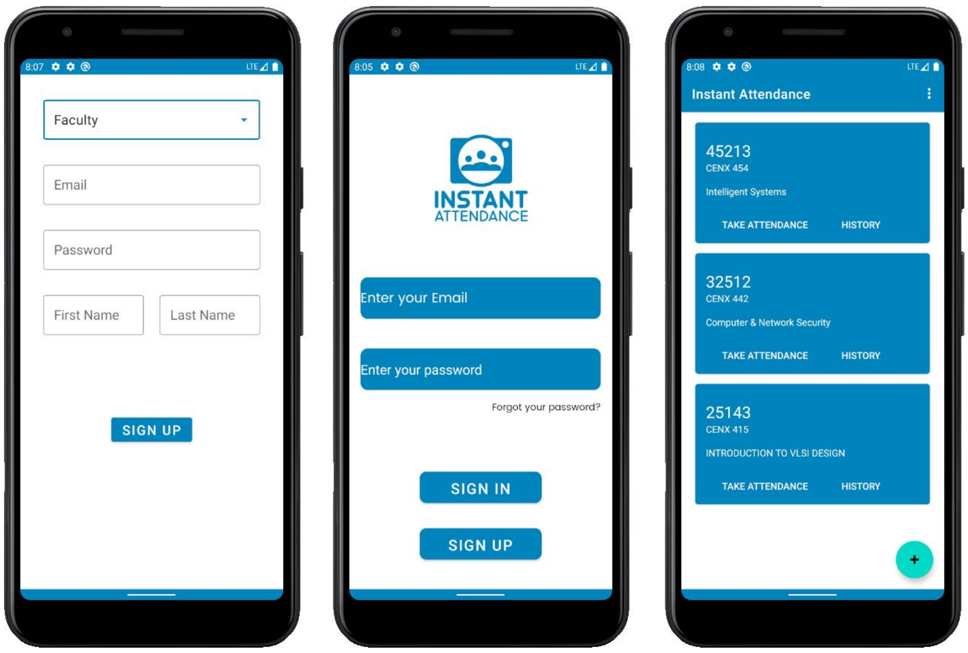
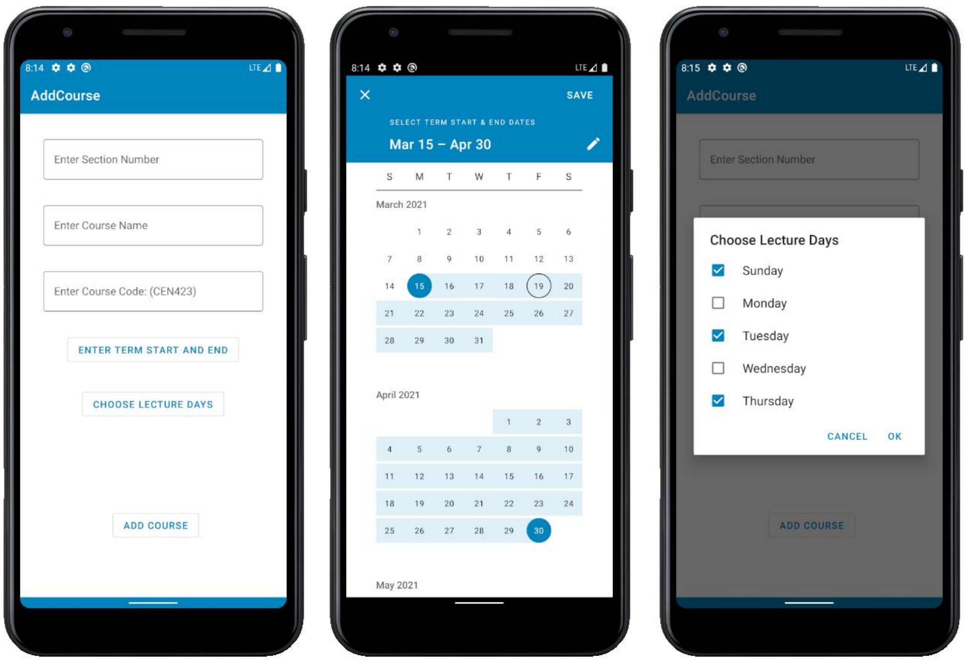


Figure 5.6 Sign Up, Sign In, and Main screens.

The add course screen is shown in figure 5.7, where the user must fill out specific information about the section to add.

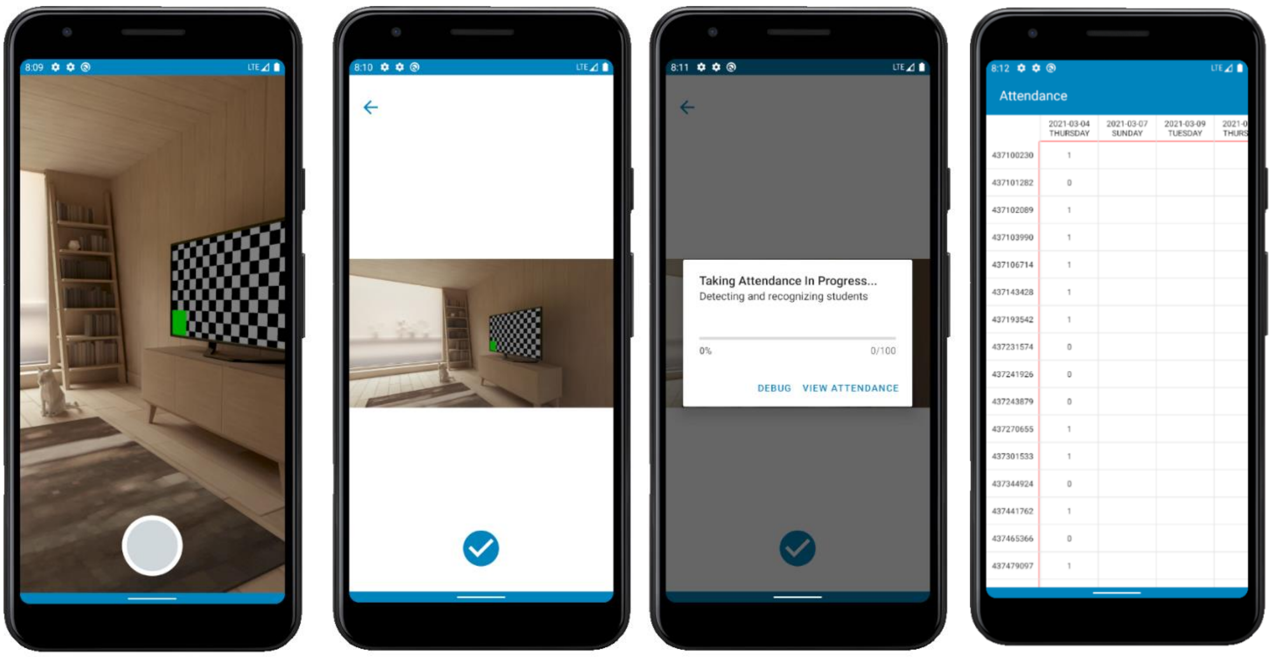


Choose Lecture Days

- ✓ Sunday
- ☐ Monday
- ✓ Tuesday
- ☐ Wednesday
- ✓ Thursday

Figure 5.7 Add Course Screen.

The user chooses to take a new attendance for a selected section from the Main Screen. In Figure 5.8, the Camera View Screen will show a live preview of the screen with a shutter button to take an image. Once pressed the Taken Image Screen will give the user the ability to review the image if the user is not satisfied with it a back button will show the Camera View Screen to take another one, otherwise if the user is satisfied with the taken image a checkmark button will start the attendance taking proccess in the background with a pop-up dialog showing the current percintile of that proccess.Once the proccess finishes the pop-up dialog will offer the user to view the attendance table.


Taking Attendance In Progress...
Detecting and recognizing students
0%
0/100
DEBUG
VIEW ATTENDANCE
Attendance

	2021-03-04 THURSDAY	2021-03-07 SUNDAY	2021-03-09 TUESDAY	2021-0 THURS
437100230	1			
437101282	0			
437102089	1			
437103990	1			
437106714	1			
437143428	1			
437193542	1			
437231574	0			
437241926	0			
437243879	0			
437270655	1			
437301533	1			
437344924	0			
437441762	1			
437465366	0			
437479097	1			

Figure 5.8 Camera View, Taken Image, and Attendance Table View Screens.

Chapter 6: Implementation

6.1 Introduction

In this chapter we will discuss the implementation phase of the system on PC and mobile. While trying to design and implement the project we have carried out the implementation in four basic phases to be observed in Change Control.

6.2 Change Control

The implementation phase acquired most of our project time and effort, we have had to carefully plan the objectives and minimize change in critical modules mentioned in the previous chapter. The implementation phase was carried out in four main steps at each advancement the implementation difficulty increases while the change should be minimized as possible because of increased cascading effect. In Table 6-1 the four implementation phases that were carried out.

Table 6-1 Implementation Phases

Phase	Objective	Change Control
System Planning	Exploration Phase	Low
System Analysis	Selection and Installation	Medium
System Design	Initial Implementation	High
System Operation	Full Implementation	Highest

6.3 System Implementation

The overall solution was built using Python on PC and using java on Android Studio for Android mobile phones. The project modules needed to be reimplemented with the same objective in mind on these two platforms.

We chose a modular approach to design and implement the system where each module design was mentioned separately in chapter 5, in this chapter the implementation will also be discussed separately on each platform.

6.3.1 External Camera Module

On PC a goprohero3+ was acquired to begin the implementation during the System Analysis phase. To interface with the camera we have used “goprocam”[14] open source library and slightly modified the source code because of runtime connection errors, once those issues were eliminated we had a successful way to interface with the camera to achieve its objective in a series of steps as follows:

- Connect to the camera using WiFi.
- Take a picture of the attendance.
- Download the taken picture using the WiFi connection.
- Delete the picture from camera storage (if the storage is full, we can’t take a new picture).

The outcome of this module is a 1080p wide angle picture fitting the students attending the class.

On mobile instead of having to use an external camera mobile phones have cameras that are way better than webcams on PCs. Also, an external camera would defeat the mobility when we aim for handheld platforms. Android introduces a new problem when trying to communicate with cameras as an open platform Android phones are equipped with hundreds of different sensors. We used a new library called CameraX and while not being easy to implement we have implemented a camera module based on CameraX capable of running on Android phones.

6.3.2 Image Reading and Manipulation Module

An important part of the system is to manipulate images. On PC during implementation we have chosen to use a popular well documented python library named “OpenCV”[15]. This module is deeply integrated into the system because of its importance, this integration can be summed up in the following instances:

- Reading the downloaded image from the External Camera Module, to be used in Face Detection Module.

- Cropping and resizing the original image each time a face is detected using the bounding box output from Face Detection module, to be used by Data Logging Module.
- Reading the face images of attendees and reference students to be used by Face Recognition module.

On Android we don't have to use OpenCV instead we have used Bitmap library to perform the objective of this module.

6.3.3 Face Detection Module

This module was implemented on PC using TensorFlow and MTCNN[16] libraries for the high accuracy compared to other methods such as Haar cascade.

While the implementation of MTCNN is possible on mobile. Through prototyping and testing. It showed infeasibility to large number of students. We have implemented this module using Android MLKit [17] as an alternative. It operates similarly giving a list of bounding boxes for each face detected. Achieving the same objective while being designed for mobile devices.

6.3.4 Face Recognition Module

On PC we have worked on the FaceNet implementation by David Sandberg[18] designing the method mentioned in Design chapter for this module. Where we have a list of registered known students in each section and an unknown list of students that we must identify. We input the two list into the FaceNet model the output would be an array of Euclidean Embeddings then for each embedding from the known list we take the Euclidean distance to the unknown list if the distance is less than a threshold the student is identified.

Since we worked CPU only on mobile, we have faced force closes (application shutdowns) from the android OS trying to implement FaceNet because of low hardware resources. We solved these issues by using multi-thread programming where we have separated the load of FaceNet computation on multiple threads to work in parallel consuming less time and lowering load on the main thread to avoid force closure by android task schedular.

6.3.5 Data Logging Module

On PC we have used several ways to log data. The easiest being the included python library “OS” to open, edit, and save log files (text files) to act as input and output between modules. To log student’s attendance sheets, we have used “openpyxl”[13] an open source library to edit Excel sheets, a draft Excel sheet was used including most importantly Student IDs, and Dates of lectures. To take attendance when we have identified present students, we look for the current section open the sheet corresponding to the section number (create one if the sheet doesn’t exist). After opening the sheet, we will iterate over columns to look for the current date of the lecture, then iterate over rows marking attendant student with a “1” and absent students with a “0”.

On Android we have used “TableView” [19] library to create an extra screen within the system to act as a replacement for the Excel sheet while visually looking and objectively the same. Figure 5.9 shows a screenshot of the attendance sheet on Android.

6.4 Backend Implementation

The project needed a database so that the two platforms can communicate the same information, we have created a none-SQL database on Google Cloud platform using Firebase. The main functionalities:

- **Log In.**
- **Read data.**
- **Write data.**

Chapter 7: Testing

7.1 Introduction

In this chapter we have created a test for our system to measure its performance on both PC and mobile.

7.2 Test

We have setup three test Sections and measured execution time and memory usage on each Section. Table 7-1 shows two device specifications the test was conducted on.

We will be testing the attendance taking process in three test sections:

- Section 1: contains 10 registered students and 10 attending students (worst case).
- Section 2: contains 20 registered students and 20 attending students.
- Section 3: contains 30 registered students and 30 attending students.

Each test is performed three different times noting the average of the three runs to eliminate extreme values.

Table 7-1 Test Devices Specification

	PC	**Mobile**
CPU	Intel® Core™ i7-8750H	Qualcomm® Snapdragon™ 845
Number of Cores	6 cores	8 cores
Base Frequency	2.20 GHz	4 cores x2.5 GHz & 4x1.6 GHz
Number of Threads	12 Threads	8 Threads
GPU	GeForce GTX 1050	Not Used
RAM	16GB	4GB
Face Detection	MTCNN	Android MLKit
Face Recognition	FaceNet (TensorFlow)	FaceNet (TensorFlowLight)

7.3 Result

The outcome in Figure 7.1 shows the three tests performed on the PC device, the left Y-Axis shows time in seconds and applicable for Total Time, MTCNN, and FaceNet legends. The right Y-Axis measures memory usage in GB. The X-Axis is the total number of student images.

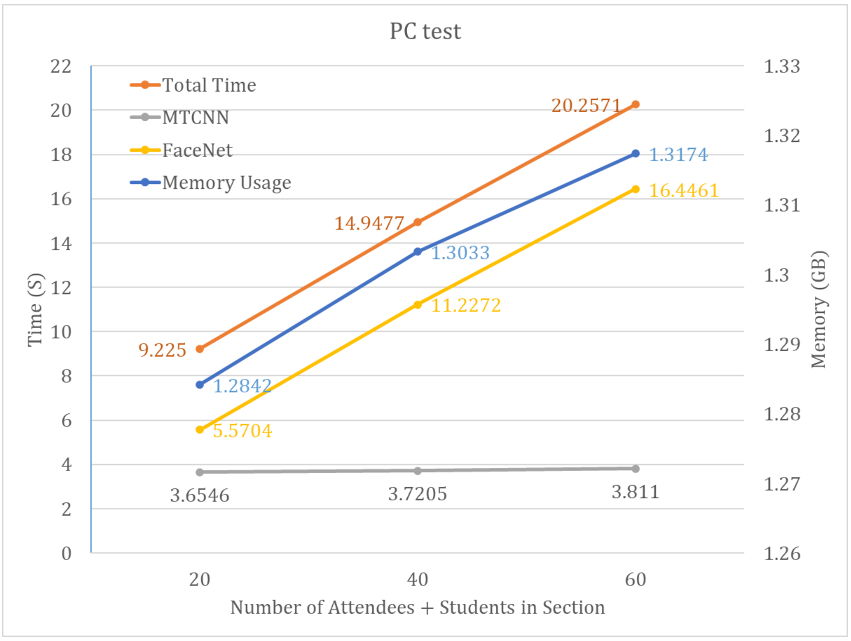


Figure 7.1 PC Test Result

MTCNN showed a steady growth in execution time rising about 0.1S for each 10 students increment. FaceNet showed a steady growth nearly 5.3S for each 20 students increment. Lastly allocating Tensors took averagely one full second on both MTCNN and FaceNet collectively raising the total time of attendance recognition by two seconds.

Figure 7.2 shows the outcome of testing the attendance system on the mobile device, the left Y-Axis accounts for time in Seconds and applicable for Total Time, MLKit, and FaceNet legends. The right Y-Axis measures memory usage in MB. The X-Axis is the total number of student images.

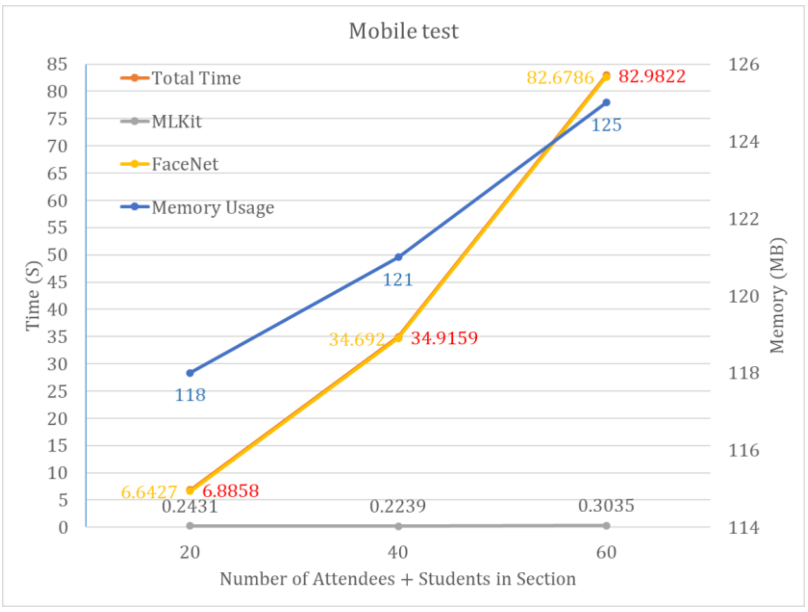


Figure 7.2 Mobile Test Result

While MLKit advances compared to MTCNN in face detection by a large margin and smaller growth. Mobile suffer in FaceNet as students number increase the longer the time and higher the growth gets. FaceNet counts for 99% of the total time. When comparing the total time of the two implementations in Figure 7.3 the PC solution is much stable and reliable when increasing the number of attendees.

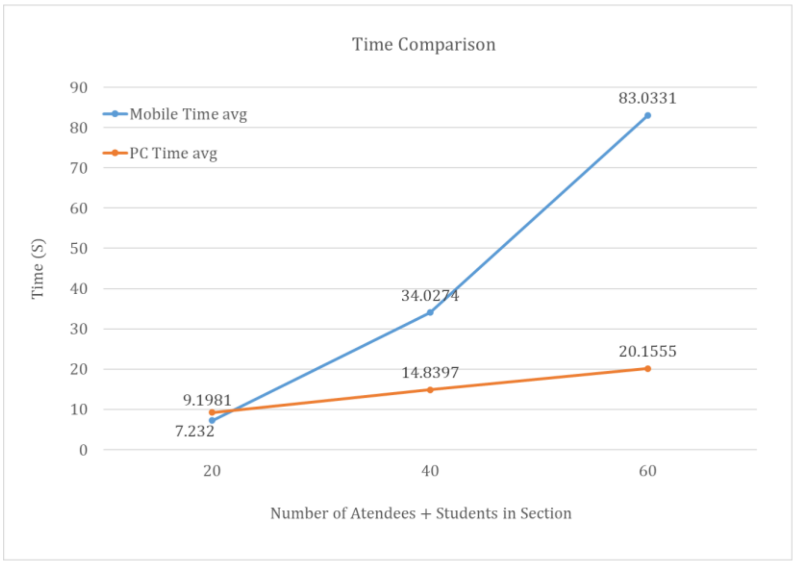


Figure 7.3 Time Comparison

unlike the PC solution memory usage of the system on mobile is much lower as shown in Figure 7.4.

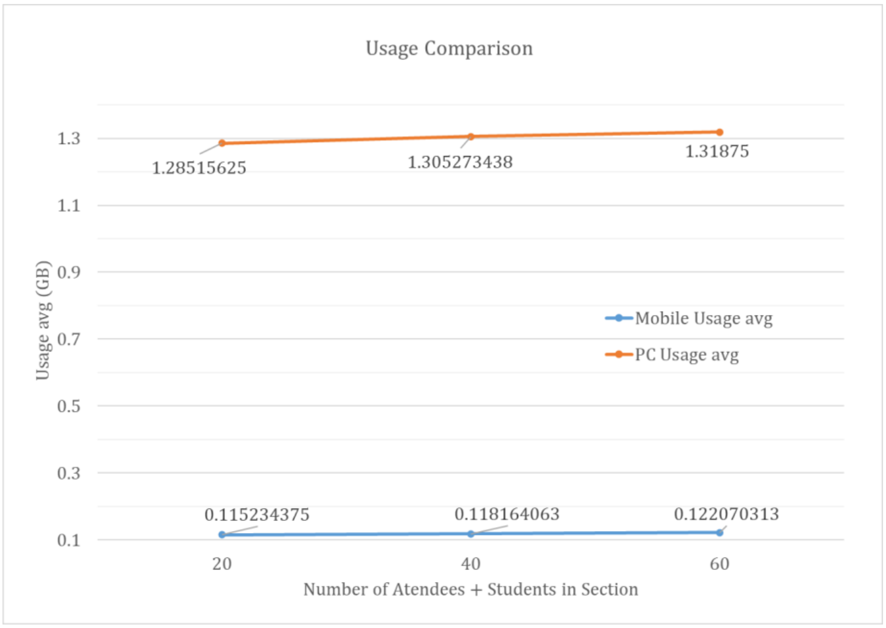


Figure 7.4 Memory Usage Comparison

Chapter 8: Conclusion

8.1 Project Timeline

In Figure 6.1 the project phase I timeline is shown. Every week on Sunday an Advisor Meeting (A.M.) with Dr. Ayed AlQahtani was conducted where the discussion of project development, problems, and comments are established.

Group member's meetings were at least three days a week but not strictly as more often we have met more than mentioned while always abiding by social distancing rules. Most phases were carried out concurrently as a problem encountered in a phase may lead to a cascade change in other phases which is a robust timelining strategy for efficient time and effort management.

- Requirement Gathering Phase: a preliminary phase at the start of the project to understand hardware and software requirements.
- Prototype Phase: hands-on development to validate Conceptual Design ideas.
- Design Phase: devising the system design.
- Literature Review: researching the old works in this field.
- Phase I Report: working to update project report.

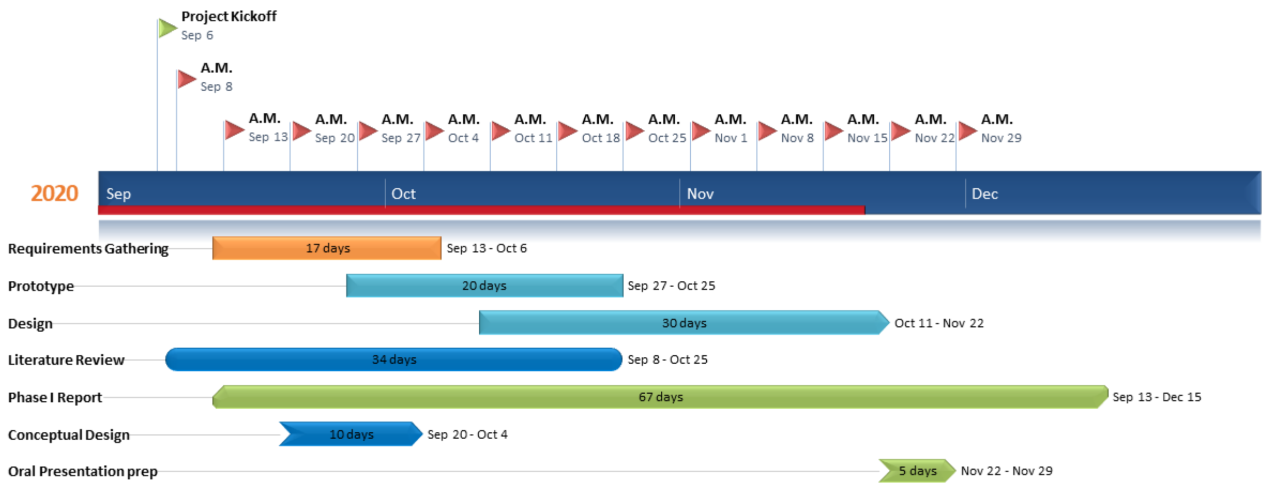


Figure 8.1 Phase One Project Timeline

In Figure 8.2 the phase two project timeline is shown.

- Prototype Phase: create a runnable version to validate implementation ideas.

- Implementation Phase: hands-on implementation phase of the system on PC and mobile.

- Testing Phase: setup a test for our system to measure its performance on both PC and mobile.

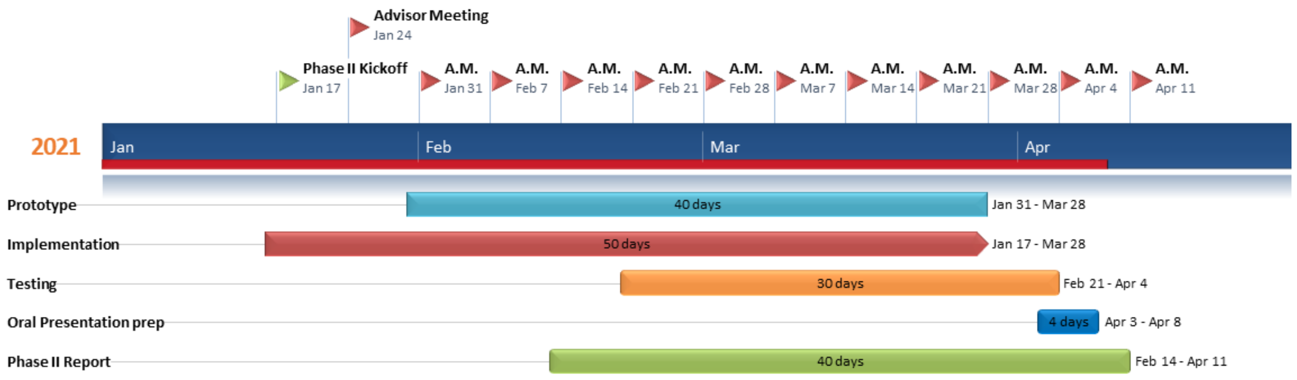


Figure 8.2 Phase Two Project Timeline

During the two project phases the team members tasks were mostly shared, while all team members are involved in all phases mentioned in the timelines only some tasks were individually carried out:

- Abdullah Alhumaid handled Weekly Reports of our weekly meetings.
- Mohammed Alanazi searched and acquired necessary hardware equipment.
- Abdullah Theeb executed mobile testing.

8.2 Problems Encountered

- Managing device permissions on Android:

On mobile we need to request camera, storage read, and storage write permissions at the first run of the system or when those permissions are changed later by the user from device settings.

- Mobile performance:

Due to performance issues on mobile we have used multi-threading to have better performance results rather than always running on the systems' main thread. Running a resource intensive neural network like MTCNN is possible but it yields lesser performance and can lead to force closure of the system, which made it unfeasible for the project requirements thus changing to a light alternative Android MLKit.

- TensorFlow installation and hardware requirements:

We went through TensorFlow documentation from google, installed it along with CUDA library for our GPU from INVIDIA.

- Face detection coordinates:

We had a hard time understanding it and the problem was solved by trial and error.

- Finding a graphical library for python:

Since python had a few options for graphical library, we have chosen to work with KIVY, which is still under development (Limited specifications).

- Permissions to edit on a file (PC):

The operating system refuses changes on an existing file, because of the operating system creating a new PID (process ID) at each time the system is initiated. The solution was to copy the file from the operating system to our program shared memory and delete the original file, while editing on the copy file then we will save it in the file system.

8.3 Conclusion

We have designed and implemented the attendance system on PC and mobile. using a modular design proposed in chapter 5 which eliminated time waste during the implementation phase. These modules are heavily dependent on CNN to detect human faces using MTCNN and recognize them in FaceNet framework using Tensorflow, while record keeping in a widely used Excel format on PC.

On mobile the modules changed slightly to accommodate the mobile environment preserving the same objective we have used Android MLKit to detect faces rather than MTCNN because of the performance drawbacks on mobile and the same FaceNet framework to recognize faces but converted to a TensorFlowLight model.

After testing both implementations in chapter 7, the result shows that the implementation on PC is significantly faster than mobile especially when increasing the number of students to take their attendance. Nevertheless, the mobile implementation is still effective compared with conventional manual attendance taking.

Usually faculty manual attendance of a section with 20 students can take from 2 to 5 minutes regardless of interventions, where our system on PC with the same number of students takes up-to 15 seconds and on mobile up-to 35 seconds saving lecture time while taking less effort.

Lastly, faculty members can use their personal mobile units costing less than building the attendance system in every class because of the requirement to deploy an external camera and equip the class podiums with a GPU enabled PC.

References

[1] R. Ford and C. Coulston, “The Requirements Specification,” in *Design for Electrical and Computer Engineers*, 1st ed., USA: McGraw-Hill, Inc., 2007, pp. 35–62.

[2] E. Grossi and M. Buscema, “Introduction to artificial neural networks,” *European journal of gastroenterology & hepatology*, vol. 19, pp. 1046–1054, Jan. 2008, doi: 10.1097/MEG.0b013e3282f198a0.

[3] K. L. Du and M. N. S. Swamy, “Fundamentals of Machine Learning,” in *Neural Networks and Statistical Learning*, 2nd ed., London: Springer, 2013, pp. 16–56.

[4] C. Rasche, “Recognition Processes,” in *Computer vision*, 1st ed., Polytechnic University, 2019, pp. 8–17.

[5] R. C. Gonzalez and R. E. Woods, “The 2-D Convolution Theorem,” in *Digital image processing*, 3rd ed., Upper Saddle River, N.J.: Prentice Hall, 2008, pp. 271–274.

[6] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks,” *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016, doi: 10.1109/LSP.2016.2603342.

[7] H. Ku and W. Dong, “Face Recognition Based on MTCNN and Convolutional Neural Network,” *Frontiers in Signal Processing*, vol. 4, 2020, doi: 10.22606/fsp.2020.41006.

[8] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face recognition and clustering,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815–823, doi: 10.1109/CVPR.2015.7298682.

[9] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967, doi: 10.1109/TIT.1967.1053964.

[10] F. K. Noble, “Comparison of OpenCV’s feature detectors and feature matchers,” in *2016 23rd International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, 2016, pp. 1–6, doi: 10.1109/M2VIP.2016.7827292.

[11] R. Padilla, C. F. F. C. Filho, and M. Costa, “Evaluation of Haar Cascade Classifiers Designed for Face Detection,” *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 6, pp. 466–469, 2012.

[12] I. William, D. R. I. M. Setiadi, E. H. Rachmawanto, H. A. Santoso, and C. A. Sari, “Face Recognition using FaceNet (Survey, Performance Test, and Comparison),” in *2019 Fourth International Conference on Informatics and Computing (ICIC)*, 2019, pp. 1–6, doi: 10.1109/ICIC47613.2019.8985786.

[13] charlie clark, “openpyxl,” *PyPI*. 2021, [Online]. Available: https://pypi.org/project/openpyxl/.

[14] K. Iturbe, “GoPro API for Python,” *GitHub*. 2020, [Online]. Available: https://github.com/KonradIT/gopro-py-api.

[15] O. Heinisuo, “opencv,” *PyPI*. 2021, [Online]. Available: https://pypi.org/project/opencv-python/.

[16] I. Centeno, “MTCNN,” *GitHub*. 2021, [Online]. Available: https://github.com/ipazc/mtcnn.

[17] G. Developers, “MLKit,” *Google Developers*. 2021, [Online]. Available: https://developers.google.com/ml-kit.

[18] D. Sandberg, “Face Recognition using Tensorflow,” *GitHub*. 2018, [Online]. Available: https://github.com/davidsandberg/facenet.

[19] E. Coskun, “TableView,” *GitHub*. 2021, [Online]. Available: https://github.com/evrencoskun/TableView.

Appendix A: Project Source Code

The mobile implementation source code is available [Online]:

Mobile Source Code

The PC implementation source code is available [Online]:

PC Source Code