DE-41 (EE) Arslan, Arshmah, Junaid

Automated Attendance System Using Facial Recognition

COLLEGE OF
ELECTRICAL AND MECHANICAL ENGINEERING
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY
RAWALPINDI
2023

COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING

DE-41 EE
PROJECT REPORT

Automated Attendance System using Facial Recognition

Submitted to the Department of Electrical Engineering
in partial fulfillment of the requirements
for the degree of
Bachelor of Engineering
in
Electrical
2023

Submitted By:
Arslan Ali
Arshmah
Muhammad Junaid Iqbal

Certificate of Approval

I clarify that the project “Automatic Attendance System using Facial Recognition” was done by ASC Arslan Ali , PC Arshmah , NC Muhammad Junaid under the supervision of Dr Naeem Ul Islam. This project was submitted to Department of Electrical Engineering College of Electrical and Mechanical Engineering (Peshawar Road Rawalpindi), National University of Sciences and Technology, Pakistan in partial fulfilment of requirements for the degree of Bachelor of Engineering in Electrical engineering.

Students:

1- Arslan Ali
NUST ID _______________________ Signature:______________________

2- Arshmah
NUST ID:_______________________ Signature:______________________

3- Muhammad Junaid
NUST ID:_______________________ Signature:_____________________

APPROVED BY:

Project Supervisor: Dr Naeem Ul Islam
Signature _______________________ Date: ______________________
Head of Department: Dr Fahad Mumtaz Malik
Signature _______________________ Date: _____________________

ACKNOWLEDGMENTS

We would like to sincerely thank everyone who helped us finish this group project on "Automated Attendance System using Facial Recognition." We are really appreciative of their help; their support, direction, and encouragement have been helpful along the process.

Prior to anything else, we would want to express our sincere gratitude to our supervisor, Asst Prof Dr Naeem Ul Islam, for his constant leadership, knowledge, and priceless comments. His astute recommendations and unflinching support were crucial in helping to mould and improve the quality of this endeavour.

We also want to express our gratitude to our faculty members, whose knowledge and experience have served as a constant source of inspiration. Their dedication to excellence in research and instruction has had a significant influence on our academic development.

We would like to express our appreciation to the technical personnel and lab assistants who helped with the project's implementation. Their knowledge and desire to assist have tremendously aided in the efficient conduct of the tests and data gathering.

We are appreciative of our family and friends' unfailing support, inspiration, and tolerance during this effort. Our perseverance has been fuelled by their confidence in our ability and never-ending encouragement.

The participants who offered their time and gave their permission to be included in the dataset utilised for evaluation deserve our deepest gratitude. Their participation and openness to help have been essential in demonstrating the Automated Attendance System's efficacy.

Finally, we would want to express our gratitude to everyone who has helped with this group initiative, whether they were directly involved or not. We are grateful to have had their support as their contributions have been important in making it a success.

ABSTRACT

Automatic Attendance System has the potential to simplify attendance management procedures in many educational contexts, the development of an automatic attendance system utilizing facial recognition has attracted considerable attention in recent years. The system described in this paper uses a camera installed in a classroom to take pictures of students as they participate in class. The Multi-Task Cascaded Convolutional Networks (MTCNN) technique is used to analyze the collected images in order to identify and extract facial features.

A model built using transfer learning methods is used to enable facial recognition. With the aid of transfer learning, the model can improve its capacity for face recognition and classification by making use of prior knowledge from a sizable dataset. The model is learned using a deep learning framework [10], allowing it to pick up on complex facial patterns and features.

The system runs in a loop throughout the class period, utilizing the camera to continuously take pictures of students' faces. In the recorded photos, facial regions of interest are found and aligned using the MTCNN algorithm. The trained facial recognition algorithm uses these aligned faces as input and compares the derived attributes to an existing database of registered pupils to identify the subjects. The model gives identities to the observed faces, making it possible to track attendance accurately and effectively.

The suggested system has a number of benefits, such as real-time attendance tracking, less administrative workload, and improved attendance management accuracy. Automating the attendance process allows educators and institutions to devote more time to teaching activities, improving the learning process as a whole.

The suggested system's experimental assessments show encouraging results, with high accuracy rates in face detection and identification tasks. The system's effectiveness could be impacted by issues such varying lighting conditions, posture, and occlusion. To address these issues and raise the robustness and reliability of the system, more investigation and improvement are needed.

SUSTAINABLE DEVELOPMENT GOALS

Goal 4: Quality Education - By automating the attendance system, the project helps to improve the effectiveness and efficiency of tracking student attendance at educational institutions, encourage better student attendance monitoring, and support productive learning settings.

Goal 9: Industry, Innovation, and Infrastructure - The project develops an automatic attendance system using cutting-edge technology like face recognition and Raspberry Pi. This supports the creation of sustainable infrastructure in educational institutions and advances technical growth.

Goal 11: Sustainable Cities and Communities - The project uses an automatic attendance system to assist in the development of smart and sustainable campuses. This improves campus operations' overall effectiveness and lessens the paper waste caused by conventional attendance practices.

Goal 12: Responsible Consumption and Production - By eliminating the need for paper-based attendance registers, the automated attendance system helps educational institutions adopt more environmentally friendly consumption habits.

Goal 16: Peace, Justice, and Strong Institutions - The project encourages transparency and accountability in educational institutions by assuring accurate attendance tracking. It promotes a culture of integrity and responsibility while assisting in the establishment of a fair and just workplace for students, teachers, and staff.

TABLE OF CONTENTS

ACKNOWLEDGMENTS iv
ABSTRACT v
SUSTAINABLE DEVELOPMENT GOALS….......... vi
TABLE OF CONTENTS vii
LIST OF FIGURES viii
Chapter 1 – INTRODUCTION 1
1.2 Problem statement 1
1.3 Objectives 1
1.4 Significance Of The Study 2
1.5 Advantages 2
Chapter 2 – BACKGROUND AND LITERATURE REVIEW…........ 4
2.1 Background 4
2.2 Overview of Attendance Management Systems 4
2.3 Face Recognition Technology.......... 4
2.4 Previous Studies On Automated Attendance Systems 5
2.5 Existing Face Recognition Algorithms Technology 5
2.6 Evaluation Metrics For Face Recognition Systems 5
Chapter 3 – METHODOLOGY 6
3.1 System Architecture and Components 6
3.2 Software Setup and Configuration 6
3.3 Data Collection and Pre-processing 8
3.4 Dataset Training and Transfer Learning 11
3.5 Face Detection and Recognition Algorithms 15
3.6 Graphical User Interface 18
3.7 Attendance Recording and Management 30
3.8 System Evaluation and Optimization 30
Chapter 4 – RESULTS 31
4.1 Dataset Description 31
4.2 Face Detection Performance 31
4.3 Face Recognition Performance 32
4.4 System Performance and Efficiency 33
4.5 Discussion 33
Chapter 5 – CONCLUSIONS AND FUTURE WORK 35
5.1 Conclusion 35
5.2 Future Work 35
REFERENCES 37

LIST OF FIGURES

Figure 1.1. Flow Chart of Objective 2
Figure 3.1 Raspberry Pi Imager ………..…………………... 7
Figure 3.1 sudo raspi-config window………..…………….. 8
Figure 3.3 Data Collection view in Opencv window 10
Figure 3.4. Object Label in Qt Designer…..………………... 26
Figure 3.5. GUI of LoginWindow (left), error message after wrong input (right) ……........... 27
Figure 3.6. GUI of MainWindow…………………….. 27
Figure 3.7. GUI of Get_Data 28
Figure 3.8. Error message after invalid input………... 28
Figure 3.9. GUI of Start_Class(left), Message box after class Start and End(right)................. 29
Figure 3.10. GUI of Upload_Data 30
Figure 4.1.104 pictures of a student with different positions, lighting, and facial expressions 31
Figure 4.2. Training and validation accuracy plot against number of epochs of CNN model .. 31
Figure 4.3. Training and validation accuracy plot against number of epochs of VGG16 model(left), InceptionV3 model (right)…………………………………………………….. 32
Figure 4.4. Result after testing real time with VGG16 model(up), InceptionV3 model(down) 33
Figure 4.5. Output After the class ended on a excel sheet …………………………………. 33
Figure 4.6. Output on a raspberry pi OS (Debian) …………………………………………… 34

Chapter 1 - INTRODUCTION

In the modern world, where technology is developing quickly, it is essential for regulating attendance at educational institutions. Keeping track of attendance is essential for maintaining regular student evaluations. Old or traditional methods of taking attendance take a lot of time and frequently include flaws that allow students to mark absent or not attend regularly. Additionally, manually taking attendance wastes class time.

With the aid of technological innovation, numerous solutions have been put forth to address the issue of manually taking attendance. Biometric verification, RFID (Radio Frequency Identification) cards, and computer vision techniques, which include face-recognition algorithms whose accuracy depends on numerous parameters, are among these the finest solutions.

1.2 Problem Statement

Manually recording student attendance is difficult. It necessitates the instructor manually marking each student's attendance, which takes time and leaves room for system cheating by students by marking proxies for one another. Technology has made substantial improvements to the attendance system, such as the biometric validation attendance system that uses fingerprint, iris, and facial sensors to recognise individuals. However, these systems are expensive and require student cooperation.

Both the fingerprint-based attendance system and other attendance methods need students to manually go to the classroom and authenticate their fingerprints. The facial recognition system works similarly in that students must manually arrive one at a time so that the system can identify them and record their attendance. Additionally, this approach has a flaw in that students can skip classes after their attendance has been recorded.

1.3 Objectives

The major goal is to create an attendance system that is effective and prevents students from using substitutes or skipping class. so that educational institutions can offer kids a better environment in which to develop. The precise goals consist of:

- Creating an attendance system that does not require students to manually note their attendance at a location.
- Creating a face recognition algorithm that enables students to sit in the classroom while their faces are recognised throughout the lesson and their attendance is recorded at the end based on how much of the lesson they attended.
- Creating a user interface (GUI) that is simple to use and maintain for administrators and teachers.

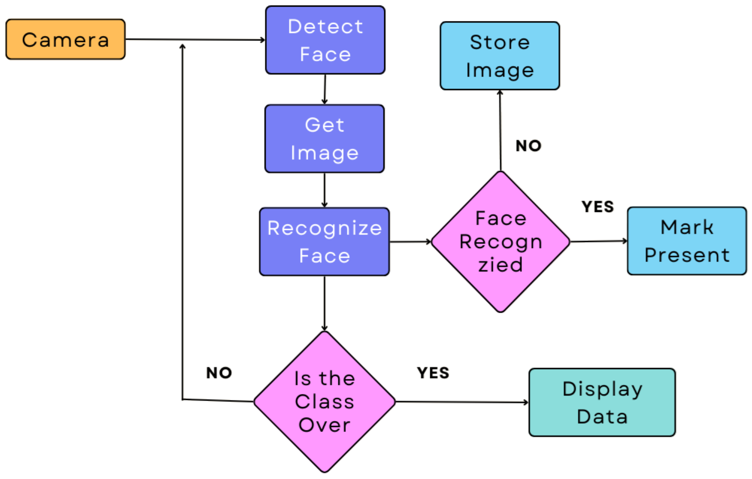


Figure 1.1. Flow chart of Objective

1.4 Significance of the Study

This study's importance arises from its ability to completely alter attendance tracking methods. The AASFR system has several benefits over manual procedures by utilising facial recognition technology. These benefits include greater resource allocation, real-time attendance tracking, improved accuracy, decreased administrative burden, and increased transparency.

Additionally, this study adds to the body of knowledge already available in the field of automatic attendance systems, particularly in the context of facial recognition technology. The results of this study can direct future improvements in attendance tracking procedures, guide policy choices, and open the door for the implementation of cutting-edge technologies in a variety of organisational and educational settings.

1.5 Advantages

Compared to conventional attendance techniques, an automated attendance system that uses face recognition, transfer learning, and MTCNN (Multi-task Cascaded Convolutional Networks) for face identification offers several benefits. Here are a few of the main advantages:

- Accuracy: Face recognition technology can identify people with great accuracy, especially when trained using transfer learning. It produces accurate attendance records since it can distinguish between students even in a variety of lighting and viewing angles.
- Efficiency: The automated method does away with the need for manual attendance taking, saving teachers and students a great deal of time and effort. The system can swiftly find and extract faces from images using MTCNN for face detection, significantly increasing efficiency.
- Real-time monitoring: The system offers real-time attendance tracking by executing the code in a loop until the class is over. It allows teachers to keep track of student presence throughout class by automatically updating the attendance records.
- Easy integration with existing administrative procedures thanks to the use of an automatic attendance system with facial recognition and the output of the results in an Excel sheet. The

attendance data is simple to export to other applications or to perform additional analysis, including computing attendance percentages.

- Elimination of proxy attendance: In manual systems, proxy attendance—in which one student notes the attendance of another—is a frequent problem. Face recognition makes it challenging for others to record a student's attendance on their behalf because each student's attendance is linked to their distinctive facial features.
- Enhanced security: The technology ups the level of security by confirming pupils' identities by looking them in the face. It aids in preventing unauthorised individuals from entering the classroom or inflating attendance.
- Data-driven insights: The system gives teachers the ability to examine attendance patterns and spot trends by organising and compiling attendance data. Decision-making, such as recognising students with a pattern of absences or improving class scheduling, might be influenced by these findings.
- Scalability: The automatic attendance system can accommodate huge class sizes without noticeably degrading in effectiveness. It offers a reliable and effective method for managing attendance and can readily scale to suit several classes or even entire schools.

Chapter 2 – BACKGROUND AND LITERATURE REVIEW

2.1 Background

The need for an effective attendance management system has grown due to technological advancements everywhere in the world, not just in educational institutions but also in offices and organizations. Older manual attendance-taking techniques, such as manually calling pupils by name or taking attendance on a sheet of paper, have flaws and take more time. It is not very dependable either. The adoption of developing technology, such as biometrics validation [1], can close these vulnerabilities.

Face recognition has emerged as a powerful technique for biometrics validation that has the ability to transform how we take attendance manually and assist us in automating it. It makes use of the most recent algorithms that have increased accuracy and efficiency by leveraging the advancements in the field of computer vision. To recognize faces, the most recent algorithms look at face patterns. These kinds of algorithms would make an attendance system more practical and effective.

Additionally, the adoption of an AASFR system is in line with the organizations and educational institutions' growing emphasis on security and access control measures. Unauthorized access can be stopped by precisely identifying people based on their face traits, resulting in a safer environment for students, staff, and visitors.

The creation and implementation of an AASFR system also progress biometric technology and their uses more generally. This project intends to improve the accuracy, scalability, and reliability of face recognition algorithms by thorough review and optimization, paving the path for their implementation in areas other than attendance management.

Overall, the ability to record, monitor, and manage attendance could significantly change with the incorporation of face recognition technology into an automated attendance system. This technology promises to simplify administrative procedures, improve security, and offer insightful data on attendance patterns and trends by combining the capabilities of computer vision with biometrics.

The approach, application, and evaluation of the AASFR system will be covered in detail in the following chapters, along with performance metrics and application-specific implications.

2.2 Overview of Attendance Management Systems

For many years, attendance management systems have been an essential part of educational institutions and businesses. The main techniques for keeping track of attendance in the past have been manual roll calls and sign-in sheets. However, these techniques are time-consuming, prone to mistakes, and frequently ineffective in large-scale situations. A rising number of people are now interested in creating automatic attendance systems to get around these constraints.

2.3 Face Recognition Technology

Due to its ability to accurately and painlessly identify people, face recognition technology has attracted a lot of interest. It makes use of computer vision algorithms to assess and contrast facial traits such the spacing between the eyes, the contour of the face, and the nose's shape. For identification and verification reasons, a number of face recognition systems have been suggested and put into practise, including Eigenfaces, Fisher faces, and Local Binary Patterns (LBP).

2.4 Previous Studies on Automated Attendance Systems

Face recognition technology has been used in automated attendance systems in a number of earlier research. A facial recognition-based attendance system for universities, for instance, was created by Zhang et al. (2015) [3] and obtained a high accuracy rate while lowering administrative load. Similar to this, Li et al. (2018) suggested an attendance management system that used deep learning techniques and showed increased accuracy in comparison to conventional approaches.

2.5 Existing Face Recognition Algorithms and Techniques

There are numerous facial recognition algorithms and methods out there, each with their own benefits and drawbacks. The most distinct features from face photos are extracted using Eigenfaces [2], a PCA-based method. By taking into account the class separability, Fisher faces, using Fisher Linear Discriminant Analysis (FLDA), improve classification accuracy. Local facial patterns are captured by LBP, a texture-based method, for recognition. Additionally, due to their capacity to acquire intricate feature representations, deep learning techniques, such as Convolutional Neural Networks (CNNs), have become more popular in face recognition applications [9].

2.6 Evaluation Metrics for Face Recognition Systems

The effectiveness of face recognition systems is evaluated using a variety of evaluation indicators. A popular metric for accuracy is the percentage of faces that were properly identified. Precision, recall, F1 score, and receiver operating characteristic (ROC) curve analysis are additional measurements. These measures support assessing the resilience and dependability of face recognition algorithms across a range of situations and datasets.

The literature review reveals the increased interest in face-recognition technology-based automatic attendance systems. Results from earlier trials in terms of accuracy, effectiveness, and decreased administrative load have been encouraging. A variety of facial recognition methods and algorithms, including as Eigenfaces, Fisher faces, LBP, and deep learning techniques, have also been investigated for use in attendance management applications.

Although face recognition technology has many advantages, there are still issues to be solved. These include changes in stance, facial expressions, lighting, and the potential for impersonation or spoofing. As a result, it's crucial to carefully consider and choose the right algorithms and methodologies that can address these issues and provide trustworthy and accurate attendance management solutions.

Chapter 3 – METHODOLOGY

3.1 System Architecture and Components

A system architecture that includes hardware elements and software modules is designed and implemented as part of the Automated Attendance System utilising Face Recognition (AASFR) approach. In this project, the following hardware parts were used:

- **Raspberry Pi 4B 2GB**: A potent single-board computer that acts as the Automatic Attendance System's central processing unit (CPU) [12]. It offers alternatives for networking, storage, and processing.

- **Mobile Camera**: Face photos were taken using a camera module attached to a mobile device. It offered flexibility and portability for taking pictures of people to track attendance.

- **Computer**: The system installation, configuration, and data administration were all done on a laptop computer. In addition to providing a platform for data processing and analysis, it permitted the installation of software components.

- **Raspberry Pi Power Cable**: The Raspberry Pi 4B was powered using the power cord to keep the machine running continuously.

- **Memory Card 32GB**: The operating system, software components, and facial image capturing data were stored on a memory card. It offered plenty of room for data retrieval and storage.

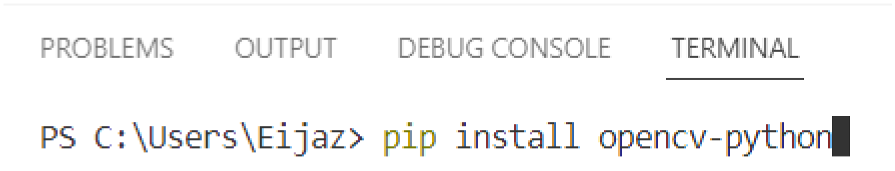
- **HDMI**: For system configuration, monitoring, and data visualisation, the Raspberry Pi was connected to a display monitor using an HDMI interface.

3.2 Software Setup and Configuration

The initial setup involved finishing the project on a laptop before transferring it to the Raspberry Pi. For the project to be coded on the laptop, an environment had to be installed. It was better to utilise lightweight IDEs because we were utilising low-end laptops, therefore Visual Studio Code was the better choice.

After installing the IDE, basic libraries were needed to be installed which include:
- opencv-python [11]
- os
- numpy
- mtcnn
- time
- pandas
- sys
- PyQt5
- google-api-client

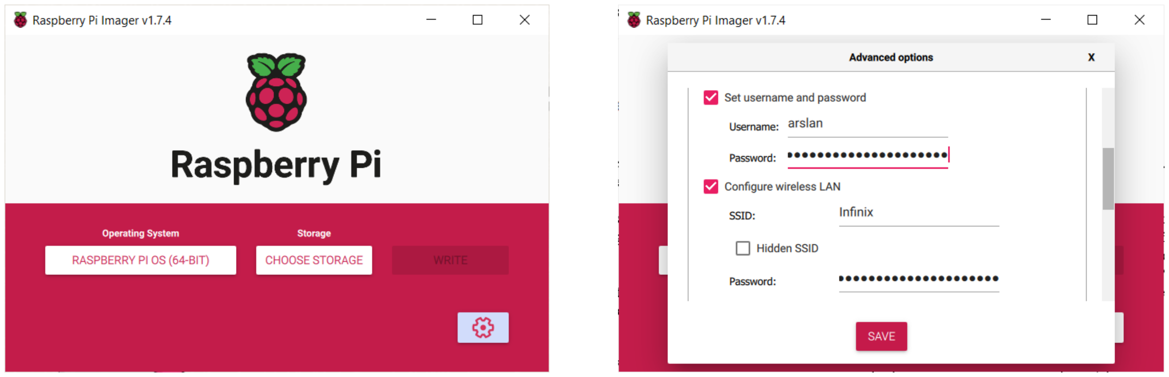
These libraries were installed using the pip command in terminal.



```
PS C:\Users\Eijaz> pip install opencv-python
```

It was also necessary to download the Qt Designer tool in order to effectively use the PyQt5 library.
The GUI of the application can be effectively built and enhanced in terms of beauty.

An operating system had to be installed in the raspberry pi in order to set it up. Raspberry Pi Imager is a suggested programme for installing the operating system. Depending on the Raspberry Pi, a 32- or 64-bit OS can be put on the memory card of the imager. The Raspberry Pi must have its WIFI configured during installation so that it will connect to it immediately after booting. The IP address of the Raspberry Pi is required after the OS has been installed on the memory card it uses, and it can be discovered via the hotspot to which the Raspberry Pi is connected.

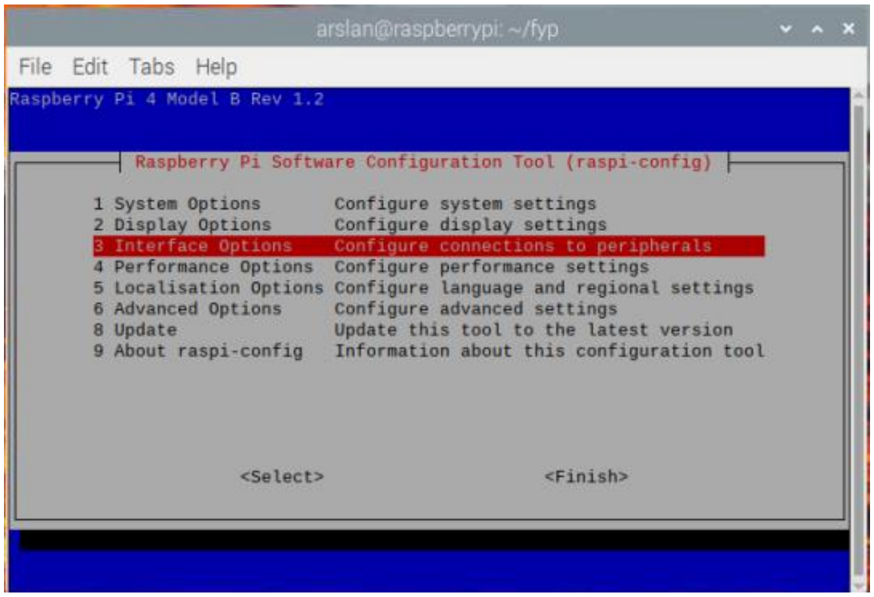


Set username and password: ✓
Username: arslan
Password: ••••••••••••••••••••
Configure wireless LAN: ✓
SSID: Infinix
Hidden SSID: ☐
Password: ••••••••••••••••••••

Figure 3.1. Raspberry Pi Imager

Since a laptop is more practical because it is portable and carrying an LCD with the raspberry pi is problematic, VNC viewer had to be installed on the laptop in order to utilise the raspberry pi headless. The connected Raspberry Pi can be utilised on the laptop using the VNC viewer. By entering the IP address of the Raspberry Pi that was previously discovered using a hotspot and to which it is linked on the VNC viewer, the Raspberry Pi can be connected.

VNC on the RPi needs to be enabled after obtaining the IP address. By starting the RPI first, it can be turned on. Once the RPi has booted, use your laptop's command prompt to establish an ssh connection. A window appears after using the command "sudo raspi-config" and entering the admin login and password. VNC is activated after the second option is chosen. The Raspberry Pi restarts after the setting has been saved.



```
Raspberry Pi 4 Model B Rev 1.2

Raspberry Pi Software Configuration Tool (raspi-config)

1 System Options       Configure system settings
2 Display Options      Configure display settings
3 Interface Options    Configure connections to peripherals
4 Performance Options  Configure performance settings
5 Localisation Options Configure language and regional settings
6 Advanced Options     Configure advanced settings
8 Update               Update this tool to the latest version
9 About raspi-config   Information about this configuration tool

              <Select>                      <Finish>
```

Figure 3.2. sudo raspi-config window

The initial setup for the Raspberry Pi is to update and upgrade it so that it may be used, as well as install new libraries. The standard setup commands consist of:

- sudo apt update
- sudo apt upgrade
- sudo rpi-update

These commands enable quicker installation of the libraries. Nearly all libraries can be installed and used in the same way as on a laptop, but there is a conflict between the opencv library and PyQt5 library. There are two ways to use both libraries:

- To delete the PyQt5 Plugins from the opencv directory.
- To install the opencv-python-headless library.

3.3 Data Collection and Pre-processing

```python
from mtcnn import MTCNN
import cv2
import os

idx = 0
clss = input("What is the degree and syn of the student ? ")
student = input("What is the name /number of the student ? ")
exist=os.path.exists(clss)
if exist == True:
    print('Directory already exists')
else:
    os.mkdir(clss)
    print('Class Directory successfully created')
isExist = os.path.exists(clss+'/'+student)
```

```
if isExist == True:
     print('Directory already exists')
     n1=(len(os.listdir(clss+'/'+student)))
     idx = n1
else:
    os.mkdir(clss+"/"+student)
    print('Directory successfully created')

n2=(len(os.listdir(clss)))
XLI_C=[]
i=0
data_dir_list = os.listdir(clss)
while i < n2:
   XLI_C.append(clss+str(i))
   XLI_C[i] = data_dir_list[i]
   i = i +1
cap = cv2.VideoCapture(0)
detector = MTCNN()
ret,frame = cap.read()
temp = 1
while ret:

    ret,frame = cap.read()
    output = detector.detect_faces(frame)

    for single_output in output:
      x,y,w,h = single_output['box']
      k=cv2.waitKey(1)
      if k == 13:
         temp = 1
      if idx<=150 and temp == 1:
        img = frame[y:y + h,x:x + w]
        resize = cv2.resize(img,(200,200))
        if k == 32:
            temp = 0
        cv2.imwrite(clss + "/" + student + '/' + student +'.'+ str(idx) + '.jpg', resize)
        cv2.putText(img, str(idx), (50, 50), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2)
        idx = idx + 1

      elif cv2.waitKey(1)==27:
          break
    cv2.imshow('win',frame)

    if cv2.waitKey(1) == 27 or int(idx) >= 200:
        break
```

The above-mentioned code is summarised as follows:

1. Requests the user's degree and the student's name and ID number.
2. Verifies whether the class directory already exists and creates it if not.
3. Establishes the student directory if it doesn't already exist in the class directory.
4. Determines how many files are present in the class directory.
5. Starts using the webcam to take pictures.
6. Employs MTCNN to find faces in the collected frames.
7. Analyses the output of the face detection algorithm and derives the bounding box coordinates.
8. Takes a maximum of 100 cropped facial photos and puts them in the student directory.
9. Pressing the spacebar will halt the image capture operation. Pressing the Esc key or shooting 100 images will stop it.

Data collection entails taking facial photos with a mobile device's attached camera. The faces of the subjects are captured in a variety of settings, including diverse lighting setups and positions. The laptop is then used to process the collected photographs further.

To improve the quality of the collected facial images and get them ready for face detection and recognition, pre-processing techniques are used. The photos may be resized, the brightness and contrast adjusted, and noise or artefacts that can impair the accuracy of the ensuing algorithms removed.

During the data collection a window is shown on the screen where the output from the camera can be seen. On the window the number of images that have been captured are shown in the form of a green number. If the spacebar button is clicked then the window will stop showing those numbers, which means that data collection has been paused. It can be resumed by pressing the Enter key. If the escape key is pressed, then the window will exit.

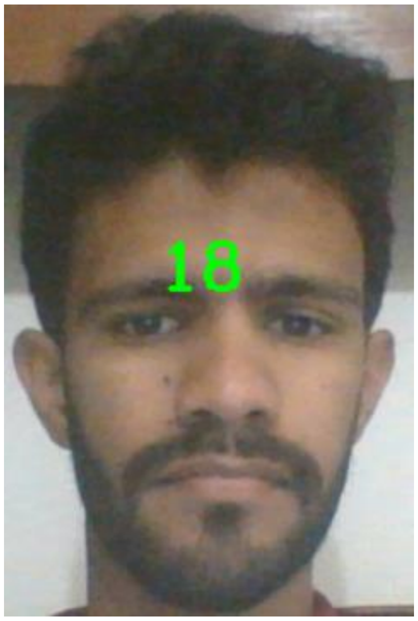


Figure 3.3. Data Collection view in opencv window

3.4 Dataset Training using Transfer learning:

The code for the training of dataset through transfer learning is shown below. It is done using a google colab file since they provide free GPU for a certain amount of time in a month.

```
from google.colab import drive
drive.mount('/content/drive')
import warnings
import os
warnings.filterwarnings('ignore')
# Get all the paths
data_dir_list = os.listdir('/content/drive/MyDrive/XLI_C')
print(data_dir_list)
path, dirs, files = next(os.walk("/content/drive/MyDrive/XLI_C"))
file_count = len(data_dir_list)
print(file_count)
# Make new base directory
original_dataset_dir = '/content/drive/MyDrive/XLI_C'
base_dir = '/content/drive/MyDrive/student_Data/'
Exist=os.path.exists(base_dir)
if Exist== True:
 print('Directory Exists')
else:
 os.mkdir(base_dir)
#create two folders (train and validation)
train_dir = os.path.join(base_dir, 'train')
Exist1=os.path.exists(train_dir)
if Exist1== True:
 print('Directory Exists')
else:
 os.mkdir(train_dir)

validation_dir = os.path.join(base_dir, 'validation')
Exist2=os.path.exists(validation_dir)
if Exist2== True:
 print('Directory Exists')
else:
 os.mkdir(validation_dir)
i=0
train=[]
validation=[]
while i < file_count:
   train.append('train'+str(i))
   train[i] = os.path.join(train_dir,data_dir_list[i])
   Exist3=os.path.exists(train[i])
```

```
  if Exist3== True:
     print('Directory Exists')
  else:
     os.mkdir(train[i])
  validation.append('validation'+str(i))
  validation[i]= os.path.join(validation_dir, data_dir_list[i])
  if Exist3== True:
    print('Directory Exists')
  else:
   os.mkdir(validation[i])
  i = i +1

def split_data(SOURCE, TRAINING, VALIDATION, SPLIT_SIZE):
    files = []
    for filename in os.listdir(SOURCE):
        file = SOURCE + filename
        if os.path.getsize(file) > 0:
            files.append(filename)
        else:
            print(filename + " is zero length, so ignoring.")

    training_length = int(len(files) * SPLIT_SIZE)
    valid_length = int(len(files) - training_length)
    shuffled_set = random.sample(files, len(files))
    training_set = shuffled_set[0:training_length]
    valid_set = shuffled_set[training_length:]

    for filename in training_set:
        this_file = SOURCE + filename
        destination = TRAINING + filename
        copyfile(this_file, destination)

    for filename in valid_set:
        this_file = SOURCE + filename
        destination = VALIDATION + filename
        copyfile(this_file, destination)
i=0
Source=[]
Training=[]
Validation=[]
while i < file_count:
 Source.append('Source'+str(i))
 Training.append('Training'+str(i))
 Validation.append('Validation'+str(i))
 Source[i] = '/content/drive/MyDrive/XLI_C/'+data_dir_list[i]+'/'
 Training[i] = '/content/drive/MyDrive/student_Data/train/'+data_dir_list[i]+'/'
 Validation[i] = '/content/drive/MyDrive/student_Data/validation/'+data_dir_list[i]+'/'
 i=i+1
import os
import random
from shutil import copyfile

split_size = .85

i=0
while i < file_count:
 split_data(Source[i],Training[i],Validation[i],split_size)
 i=i+1
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.image import imread
import pathlib

image_folder = data_dir_list
nimgs = {}
for i in image_folder:
    nimages = len(os.listdir('/content/drive/MyDrive/student_Data/train/'+i+'/'))
    nimgs[i]=nimages
plt.figure(figsize=(9, 6))
plt.bar(range(len(nimgs)), list(nimgs.values()), align='center')
plt.xticks(range(len(nimgs)), list(nimgs.keys()))
plt.title('Distribution of different classes in Training Dataset')
plt.show()
for i in data_dir_list:
    print('Training {} images are:
'.format(i)+str(len(os.listdir('/content/drive/MyDrive/student_Data/train/'+i+'/'))))
image_folder = data_dir_list
nimgs = {}
for i in image_folder:
    nimages = len(os.listdir('/content/drive/MyDrive/student_Data/validation/'+i+'/'))
    nimgs[i]=nimages
plt.figure(figsize=(9, 6))
plt.bar(range(len(nimgs)), list(nimgs.values()), align='center')
plt.xticks(range(len(nimgs)), list(nimgs.keys()))
plt.title('Distribution of different classes in Validation Dataset')
plt.show()
for i in data_dir_list:
    print('Valid {} images are:
'.format(i)+str(len(os.listdir('/content/drive/MyDrive/student_Data/validation/'+i+'/'))))
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.inception_v3 import InceptionV3
from keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.inception_v3 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
#import matplotlib.pyplot as plt
IMAGE_SIZE = [200, 200]
img_width=200; img_height=200
batch_size=16
inception = InceptionV3(input_shape=IMAGE_SIZE + [3], weights='imagenet',
include_top=False)
from keras import models
from keras import layers
from keras import optimizers
# don't train existing weights
for layer in inception.layers:
    layer.trainable = False
# our layers - you can add more if you want
x = Flatten()(inception.output)
prediction = Dense(4, activation='softmax')(x)

# create a model object
model = Model(inputs=inception.input, outputs=prediction)
model.summary()
# tell the model what cost and optimization method to use
model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
```

```
test_datagen = ImageDataGenerator(rescale = 1./255)
training_set = train_datagen.flow_from_directory('/content/drive/MyDrive/student_Data/train',
                                                 target_size = (200, 200),
                                                 batch_size = 16,
                                                 class_mode = 'categorical')
test_set = test_datagen.flow_from_directory('/content/drive/MyDrive/student_Data/validation',
                                            target_size = (200, 200),
                                            batch_size = 16,
                                            class_mode = 'categorical')
# fit the model
r = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=10,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)
model.save('/content/drive/MyDrive/10IV3.h5')
```

The code above does the following, in brief:
1. Mounts Google Drive and generates required directories.
2. Separates the data into sets for validation and training.
3. Shows how the classes are distributed across the datasets.
4. Creates and trains a transfer learning model based on InceptionV3 [7].
5. Stores the built-in model in a file.

3.5 Face Detection and Recognition Algorithms

The code shown below is for detecting and predicting the faces of students present in the classroom according to the model trained using transfer learning. When the code is run it will check the time and when the time of starting the class comes, the camera will open and start recognizing students faces. It will
be done when the class time is over.

```
import cv2
from mtcnn import MTCNN
from keras.models import load_model
from numpy import asarray
import numpy as np
import datetime
import time
import pandas as pd
import os
import requests
import cv2
import numpy as np
import imutils



n2=(len(os.listdir('41C')))
total=0
XLI_C=[]
counter=[]
status=[]
i=0
data_dir_list = os.listdir('41C')
while i < n2:
   XLI_C.append('41C'+str(i))
   XLI_C[i] = data_dir_list[i]
   counter.append('counter'+str(i))
   counter[i]=0
   status.append('status' + str(i))
   status[i] = "A"
   i = i +1

temp=1
j=0
while temp==1:
 now = datetime.datetime.now().time()
 if now.hour == 13 and now.minute == 17:
    temp =0
 else:
    temp = 1

while temp==0:
  print('Class started')
  cap = cv2.VideoCapture("video/test7.mp4")
  detector = MTCNN()
  ret,frame = cap.read()
  idx=0
  new_img=[]
  while temp==0:

   ret,frame = cap.read()
   total = total + 1
   output = detector.detect_faces(frame)
   for i in output:
     x, y, w, h = i['box']
     new_img.append('new_img' + str(idx))
     new_img[idx] = frame[y:y + h, x:x + w]
     new_img[idx] = cv2.resize(new_img[idx], (200, 200))
     cv2.imwrite('Test/' + 'test' + '.' + str(j) + '.jpg', new_img[idx])
     idx=idx+1
     j=j+1
     print('Photo Clicked')
   model = load_model('model/10IV3.h5', compile=False)
   j=0
   id = 0
   while id < len(output):
     x = asarray(new_img[id])
     y = np.expand_dims(x, axis=0)
     y = y/255

     classes = model.predict(y)
     print(classes)
     print(classes.max())
     answer = np.argmax(classes, axis=1)

     i=0
     while i < n2:
      if classes.max() > 0:
        if answer == i:
           counter[i]=counter[i]+1
           cv2.imwrite('Test/' + XLI_C[i] + '.' + str(id) + '.jpg', new_img[id])
           print(XLI_C[i])
        else:
           a=1
           print(answer)
      else:
        print('Not Recognized')
      i = i + 1
     id = id + 1
     now = datetime.datetime.now().time()
     if now.hour == 1 and now.minute == 18:
      i=0
      while i<n2:
          counter[i]=counter[i]/total
          if counter[i] > 0.75:
           status[i]='P'
          else:
            status[i]='A'
          df = pd.DataFrame([[status[0], counter[0]], [status[1], counter[1]], [status[2], counter[2]],
[status[3], counter[3]],],
                    index=[XLI_C[0],XLI_C[1],XLI_C[2],XLI_C[3]], columns=['Status', 'Avg'])
         i=i+1
      df.to_excel(time.strftime('E:/VScodeFYP/%Y-%m-%d', time.localtime())+'.xlsx',
sheet_name='CV')
      print('Class ended')
      temp=1
     else:
      temp=0
```

The above-mentioned code:
1. Imports the relevant modules and libraries.
2. Sets up lists and variables for recording pupils' attendance.
3. Uses a webcam to take pictures from a video stream.
4. Employs the MTCNN face detection model to find faces in the collected images.
5. Resizes and saves the photos that were taken.
Loads a facial recognition model that has already been trained.
7. Employs the loaded model to identify each observed face.
8. Based on the outcome of the prediction, updates the status and attendance counter.
9. Excel file is used to save the attendance information.
10. Continue the process until a predetermined period has passed.

3.6 Graphical User Interface:

```
from PyQt5.QtWidgets import QApplication, QMessageBox, QMainWindow
import sys
from PyQt5 import uic
import os,cv2,datetime,time,sys
from mtcnn import MTCNN
from keras.models import load_model
from numpy import asarray
import numpy as np
import pandas as pd
import PyQt5.QtWidgets as qtw
from google.oauth2.credentials import Credentials
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError
from google.auth.transport.requests import Request
from google.auth.exceptions import RefreshError
from google.oauth2.credentials import Credentials
from googleapiclient.http import MediaFileUpload


os.chdir("c:/Users/Eijaz/Desktop/")
```

```
class Get_Data(QMainWindow):
    def __init__(self):
        super(Get_Data,self).__init__()
        uic.loadUi("GetData.ui",self)
        self.show()

        self.pushButton.clicked.connect(self.test)
        self.actionClose.triggered.connect(exit)
    def test(self):
        self.close()
        clss = self.class_input.text()
        student = self.reg_input.text()
        idx=0
        if len(clss) == 3 and clss.isalnum() and clss.isalpha() != True and clss.isdigit() != True:
            digit_count = 0
            letter_count = 0
            for char in clss:
             if char.isdigit():
              digit_count += 1
             elif char.isalpha():
              letter_count += 1
            if digit_count == 2 and letter_count == 1:
              a=1
            else:
              QMessageBox.warning(self, 'Wrong Input', 'Invalid Class and Syndicate')
              self.close()
              return

        else:
              QMessageBox.warning(self, 'Wrong Input', 'Invalid Class and Syndicate')
              self.close()
              return

        if len(student) == 6 and student.isdigit():
           a=1
        else:
           QMessageBox.warning(self, 'Wrong Input', 'Invalid Registration Number')
           self.close()
           return
        exist=os.path.exists("E:/Test/"+clss)
        if exist == True:
             a=1
        else:
            os.mkdir("E:/Test/"+clss)
        isExist = os.path.exists("E:/Test/"+clss+'/'+student)
        if isExist == True:
             n1=(len(os.listdir("E:/Test/"+clss+'/'+student)))
             idx = n1
        else:
            os.mkdir("E:/Test/"+clss+"/"+student)
        n2=(len(os.listdir("E:/Test/"+clss)))
        XLI_C=[]
        i=0
        data_dir_list = os.listdir("E:/Test/"+clss)
        while i < n2:
           XLI_C.append("E:/Test/"+clss+str(i))
           XLI_C[i] = data_dir_list[i]
           i = i +1
        cap = cv2.VideoCapture(0)
        detector = MTCNN()
        ret,frame = cap.read()
        temp = 1
        while ret:

            ret,frame = cap.read()
            output = detector.detect_faces(frame)

            for single_output in output:
              x,y,w,h = single_output['box']
              k=cv2.waitKey(1)
              if k == 13:
                temp = 1
              if idx<=150 and temp == 1:
                img = frame[y:y + h,x:x + w]
                resize = cv2.resize(img,(200,200))
                if k == 32:
                    temp = 0
                cv2.imwrite("E:/Test/"+clss + "/" + student + '/' + student +'.'+ str(idx) + '.jpg', resize)
                cv2.putText(img, str(idx), (50, 50), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255,
0), 2)
                idx = idx + 1

              elif cv2.waitKey(1)==27:
                  break
            cv2.imshow('win',frame)
            if int(idx) >= 20:
                break
```

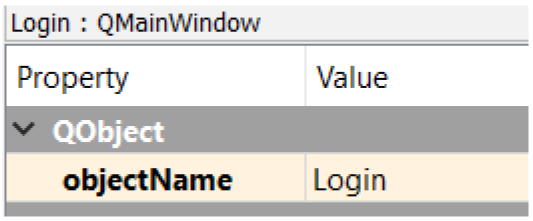
```
class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow,self).__init__()
        uic.loadUi("window_2.ui",self)
        self.show()

        self.GetData.clicked.connect(self.open_GetData)
        self.UploadData.clicked.connect(self.open_UploadData)
        self.StartClass.clicked.connect(self.open_StartClass)
        self.actionClose.triggered.connect(exit)

    def open_GetData(self):
        # Create and show window 1
        self.window1 = Get_Data()
        self.window1.show()

    def open_UploadData(self):
        # Create and show window 2
        self.window2 = Upload_Data()
        self.window2.show()

    def open_StartClass(self):
        # Create and show window 2
        self.window3 = Start_Class()
        self.window3.show()

class Start_Class(QMainWindow):
    def __init__(self):
        super(Start_Class,self).__init__()
        uic.loadUi("start_Class.ui",self)
        self.show()
        self.StartClass.clicked.connect(self.start)
        self.actionClose.triggered.connect(exit)


    def start(self):
        start_min = int(self.start_min.text())
        start_hour = int(self.start_hour.text())
        end_min = int(self.end_min.text())
        end_hour = int(self.end_hour.text())
        qtw.QMessageBox.information(self, "Class Started", "The Class has started.")
        n2=(len(os.listdir('E:/VScodeFYP/41C')))
        total=0
        XLI_C=[]
```

```
counter=[]
status=[]
i=0
data_dir_list = os.listdir('E:/VScodeFYP/41C')
while i < n2:
   XLI_C.append('E:/VScodeFYP/41C'+str(i))
   XLI_C[i] = data_dir_list[i]
   counter.append('counter'+str(i))
   counter[i]=0
   status.append('status' + str(i))
   status[i] = "A"
   i = i +1
temp=1
j=0
while temp==1:
 now = datetime.datetime.now().time()
 if now.hour == start_hour and now.minute == start_min:
    temp =0
 else:
   temp = 1

while temp==0:
  cap = cv2.VideoCapture(0)
  detector = MTCNN()
  ret,frame = cap.read()
  idx=0
  new_img=[]
  while temp==0:

   ret,frame = cap.read()
   total = total + 1
   output = detector.detect_faces(frame)
   for i in output:
     x, y, w, h = i['box']
     new_img.append('new_img' + str(idx))
     new_img[idx] = frame[y:y + h, x:x + w]
     new_img[idx] = cv2.resize(new_img[idx], (200, 200))
     cv2.imwrite('E:/VScodeFYP/Test/' + 'test' + '.' + str(j) + '.jpg', new_img[idx])
     idx=idx+1
     j=j+1
  # load the model we saved
   model = load_model('E:/VScodeFYP/model/10IV3.h5', compile=False)
   j=0
   id = 0
   while id < len(output):
```

```
            x = asarray(new_img[id])
            y = np.expand_dims(x, axis=0)
            y = y/255

            classes = model.predict(y)
            answer = np.argmax(classes, axis=1)

            i=0
            while i < n2:
             if classes.max() > 0:
              if answer == i:
                  counter[i]=counter[i]+1
                  cv2.imwrite('E:/VScodeFYP/Test/' + XLI_C[i] + '.' + str(id) + '.jpg', new_img[id])
              else:
                  a=1
             else:
               print('Not Recognized')
             i = i + 1
            id = id + 1
            now = datetime.datetime.now().time()
            if now.hour == end_hour and now.minute == end_min:
             i=0
             while i<n2:
                 counter[i]=counter[i]/total
                 if counter[i] > 0.75:
                  status[i]='P'
                 else:
                   status[i]='A'
                 df = pd.DataFrame([[status[0], counter[0]], [status[1], counter[1]], [status[2],
counter[2]], [status[3], counter[3]],],
                       index=[XLI_C[0],XLI_C[1],XLI_C[2],XLI_C[3]], columns=['Status', 'Avg'])
                 i=i+1
             df.to_excel(time.strftime('E:/VScodeFYP/%Y-%m-%d', time.localtime())+'.xlsx',
sheet_name='CV')
             qtw.QMessageBox.information(self, "Class Ended", "The Class has ended.")
             temp=1
            else:
              temp=0

class Upload_Data(qtw.QDialog):
    def __init__(self):
       uic.loadUi("Upload_Data.ui",self)
        super().__init__()

        # Create the file dialog
        self.folder_dialog = qtw.QFileDialog()
        self.folder_dialog.setFileMode(qtw.QFileDialog.Directory)
        self.folder_dialog.setOption(qtw.QFileDialog.ShowDirsOnly)

        # Connect signals to slots
        self.browse_button.clicked.connect(self.show_folder_dialog)
        self.upload_button.clicked.connect(self.upload_folder)
        self.cancel_button.clicked.connect(self.close)

    def show_folder_dialog(self):
        if self.folder_dialog.exec_() == qtw.QDialog.Accepted:
            folder_path = self.folder_dialog.selectedFiles()[0]
            self.textbox.setText(folder_path)

    def upload_folder(self):
        # Get the folder path from the textbox
        folder_path = self.textbox.text()

        # Get the Google Drive credentials
        try:
            creds = Credentials.from_authorized_user_file('token.json')
        except RefreshError:
            # If the credentials have expired, prompt the user to re-authorize
            flow = InstalledAppFlow.from_client_secrets_file('credentials.json',
['https://www.googleapis.com/auth/drive'])
            creds = flow.run_local_server(port=0)
            with open('token.json', 'w') as token:
                token.write(creds.to_json())

        # Create the Google Drive API client
        service = build('drive', 'v3', credentials=creds)

        # Create the folder on Google Drive
        folder_metadata = {
            'name': os.path.basename(folder_path),
            'mimeType': 'application/vnd.google-apps.folder'
        }
        folder = service.files().create(body=folder_metadata, fields='id').execute()

        # Upload each file in the folder
        for dirpath, dirnames, filenames in os.walk(folder_path):
            for filename in filenames:
                file_path = os.path.join(dirpath, filename)
                file_metadata = {
                    'name': filename,
                'parents': [folder['id']]
            }
            media = MediaFileUpload(file_path)
            file = service.files().create(body=file_metadata, media_body=media,
fields='id').execute()

        # Show a message box to confirm the upload is complete
        qtw.QMessageBox.information(self, "Upload Complete", "Folder uploaded to Google
Drive.")

        # Close the dialog
        self.accept()




class LoginWindow(QMainWindow):
    def __init__(self):
        super(LoginWindow,self).__init__()
        uic.loadUi("login.ui",self)
        self.show()


        self.login_button.clicked.connect(self.login)
        self.actionClose.triggered.connect(exit)


    def login(self):
        username = self.username_input.text()
        password = self.password_input.text()

        if username == 'admin' and password == 'password':
            self.window1 = MainWindow()
            self.window1.show()
            self.close()

        else:
            QMessageBox.warning(self, 'Login Failed', 'Invalid username or password')


if __name__ == '__main__':
    app = QApplication(sys.argv)

    window = LoginWindow()
    window.show()
    sys.exit(app.exec_())
```

1. The code begins with importing the required libraries/modules.
2. It defines a class **Get_Data** that inherits from **QMainWindow**. This class represents a window for getting data and contains a method **test()** that handles user input and performs various operations based on the input.
3. The **MainWindow** class represents the main window of the application. It inherits from **QMainWindow** and defines methods for opening different windows.
4. The **Start_Class** class represents a window for starting a class. It is inherited from **QMainWindow** and contains a method **start()** that performs certain operations based on user input.
5. The **Upload_Data** class represents a dialog window for uploading a folder to Google Drive. It inherits from **QDialog** and contains methods for selecting a folder and uploading it to Google Drive.
6. The **LoginWindow** class represents a login window. It inherits from **QMainWindow** and handles the login functionality.
7. In the **__main__** block, an instance of the **LoginWindow** class is created, and the application is executed.

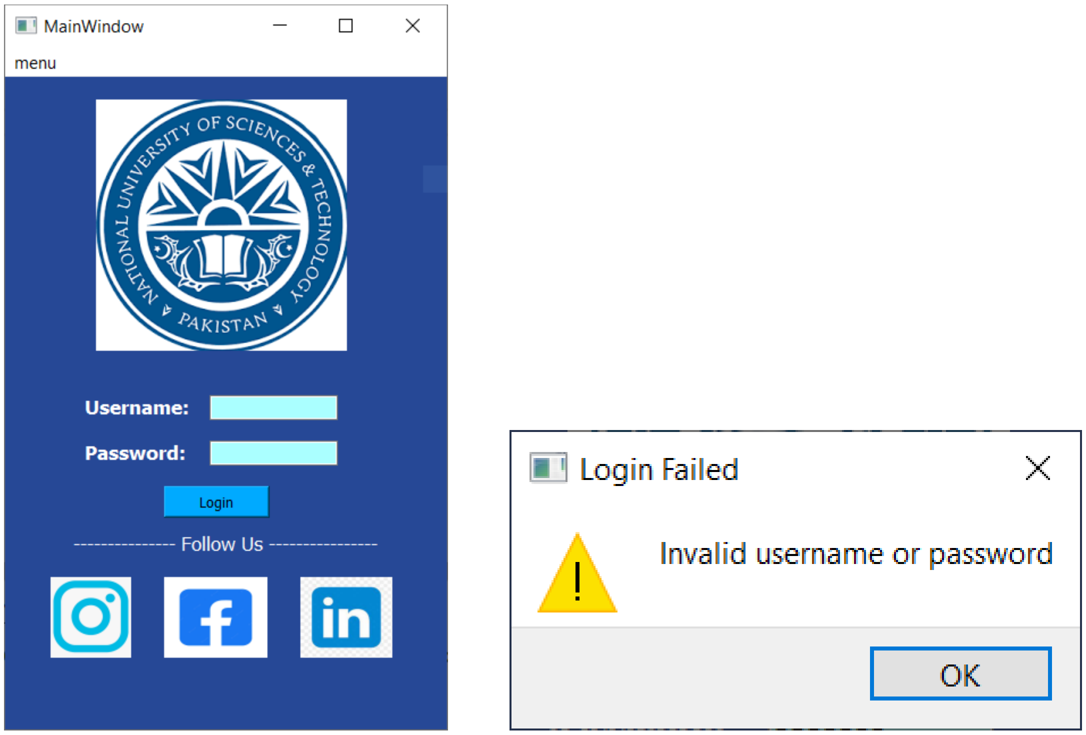
The .ui files loaded in the code above were made using the Qt Designer tool, which makes it simple to develop Graphical User Interface and contributes to its speed and aesthetic appeal. The required buttons on the GUIs were thoughtfully identified during creation, making it simple for the above code to call them.
The labels for the buttons and line edits are depicted in the diagram below.


Login : QMainWindow

Property	Value
QObject	
objectName	Login

Figure 3.4. Object Label in Qt Designer

Because the main function in the code above only refers to the class "LoginWindow" and displays its window, when the code is executed, that window will prompt. The window is depicted below. It has a Login button that is connected to the "MainWindow" function and calls out that class, as well as two line-edits for user input. It will also display an error or warning popup informing the user that the username and password they submitted are invalid. This window will close once the proper username and password have been entered.


MainWindow
menu
NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY
PAKISTAN
Username:
Password:
Login
--------------- Follow Us ----------------
Login Failed
OK

Invalid username or password

Figure 3.5. GUI of LoginWindow (left), error message after wrong input (right)

The "MainWindow" class is called and the prompts in its UI file are displayed once you enter the right username and password. There are three pushbuttons in this window, and each one has a certain purpose.

• The "Get_Data" class is invoked by the Get Data button.
• The "Upload_Data" class is invoked by the Upload Data button.
• The "Start_Class" class is called by the Start Class button.

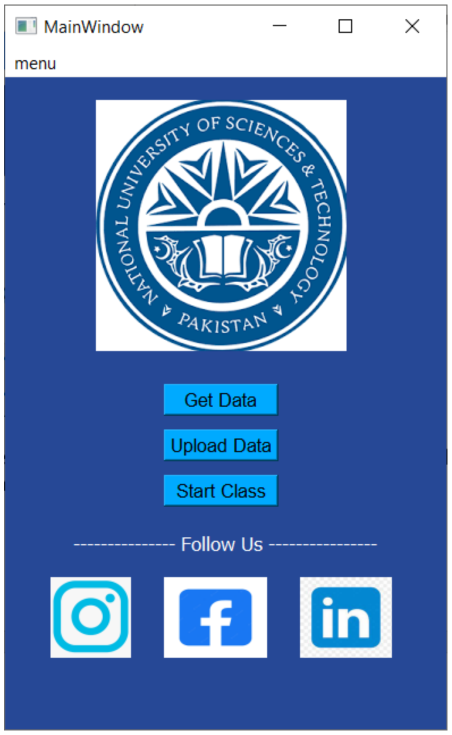

MainWindow
menu
NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY
PAKISTAN
Get Data
Upload Data
Start Class
-------------- Follow Us ---------------

Figure 3.6. GUI of MainWindow

When the Get Data button is clicked, a window like the one below will appear. Two line edits in the window are used to solicit feedback from the user. If the "Class & Synd" entry is accurate, a window will open with the error message "Invalid Class and Syndicate". Like this, a window will pop up with the error "Invalid Registration number" if the registration number is incorrect. The previously written code from section 3.3 will be called as soon as the proper inputs have been given.

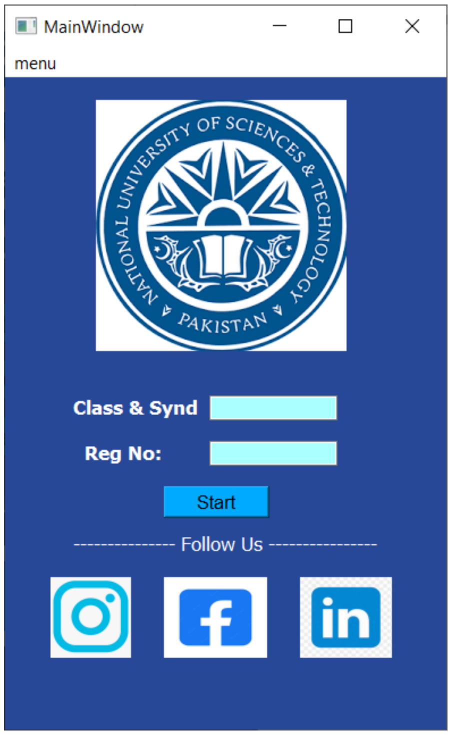

MainWindow
menu
NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY
PAKISTAN
Class & Synd
Reg No:
Start
--------------- Follow Us ----------------

Figure 3.7. GUI of Get_Data

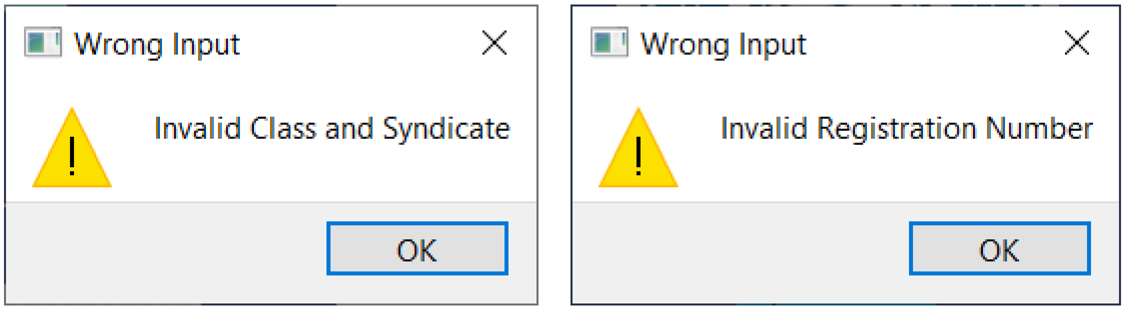
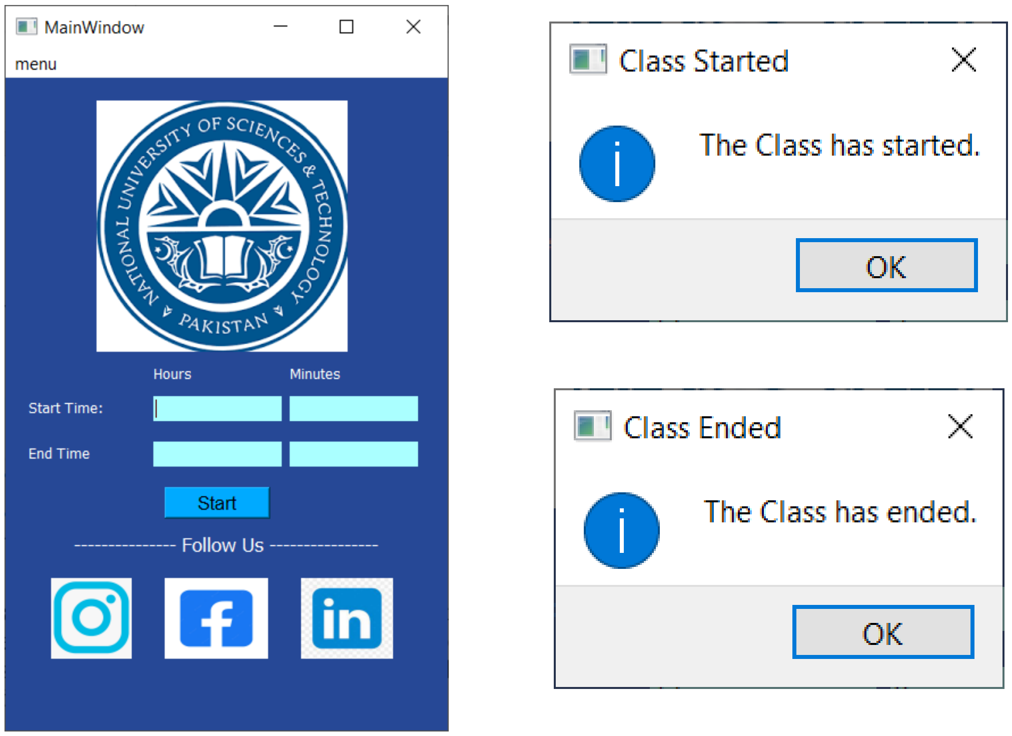


Figure 3.8. Error message after invalid class input (left), Error message after invalid registration no input (right)

When the Start Class button is clicked a window will appear as shown below. In this window there are 4 line edits that are used to take input from the user and one push button that is used to start the class. The four inputs from the user is used get the time when the class will start and the time when the class will end. When the start button is clicked a message box will appear saying that the class has started. When Ok is clicked on that message box the class will start. When the time is over another message box will appear stating that the class has ended. When OK is clicked on that message box the class will end. The output will be stored on an excel sheet in the directory the code is running. It will be named according to the date on that day. The output in excel sheet will be registrations number of the students along with the percentage of time they were present in the class and if or not they were marked present.



The Class has started.

The Class has ended.

Figure 3.9. GUI of Start_Class (left), message box after started and ended (right)

In the MainWindow if the Upload Data button is clicked, a window will appear as shown below. In this window there are three buttons with a line edit that is used to take input from the user. The input is the directory with the filename and type so when the upload button is clicked, that file or folder is uploaded to the google drive. It can also be done manually by clicking the Browse button. A window will appear, and it will ask to select a file or folder. Once selected, it can be uploaded to google drive by clicking the upload button. To make this functional, an desktop application needs to be created using google drive api on google cloud [13]. Once the application is created and published, the "credentials.json" file can be downloaded. That file needs to be put in the same directory as the python file. After that, if the upload button is clicked it will as only one time for authentication from Google. After the access is given, it will automatically download a "token.json" file which is basically the authentication and allows user to not give authentication every time they upload a file or a folder.

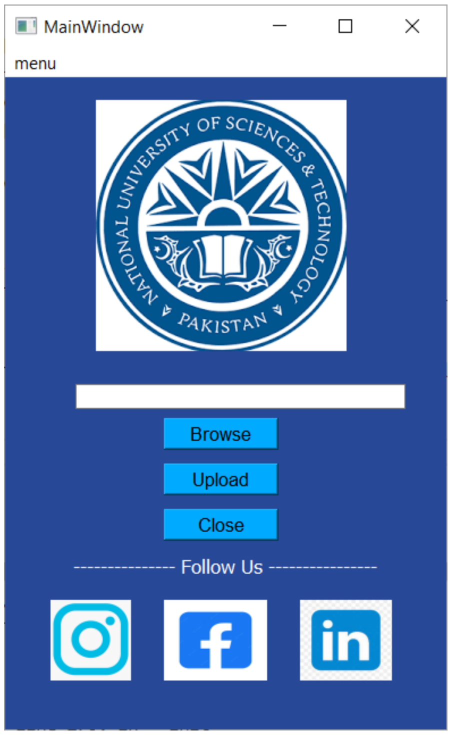

MainWindow
menu
NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY
PAKISTAN
Browse
Upload
Close
-------------- Follow Us ---------------

Figure 3.10. GUI of Upload_Data

Algorithms are used by the face identification module to locate and identify faces in the pre-processed images. Based on particular patterns or traits, these algorithms assess the image data and identify facial regions. For precise and effective face detection, methods like the Viola-Jones algorithm or deep learning-based models like Convolutional Neural Networks (CNNs) can be used.

The face recognition module extracts facial traits and compares them to pre-registered templates using sophisticated algorithms. To obtain accurate recognition performance, methods like Eigenfaces, Fisherfaces, Local Binary Patterns (LBP), or deep learning-based models like CNNs are used. They examine the discovered faces, extract differentiating characteristics, and contrast them with the database-stored templates.

3.5 Attendance Recording and Management

When face recognition is successful, the AASFR system logs attendance by assigning timestamps to the identified individuals. For simple retrieval and maintenance, the attendance records are kept in a database or a structured format. The system also offers administrative features like data export and interface with current management systems, as well as the ability to generate attendance reports, monitor attendance patterns, and give administrative functionality.

3.6 System Evaluation and Optimization

The performance, accuracy, and efficiency of the created AASFR system are evaluated. A diverse collection of test subjects is used, and different lighting, positions, and facial expressions are also tested. Calculated evaluation criteria for the system's recognition abilities include accuracy and efficiency.

Chapter 4 – RESULTS

4.1 Dataset Description

In this project, the Raspberry Pi camera module was used to gather a dataset of individual's facial images. The dataset had a wide variety of people with a diversity of positions, lighting, and facial expressions. 400 photographs were taken in total, with roughly 100 images per subject being used to represent each.

Figure 4.1. 104 pictures of one student with different positions, lighting, and facial expressions

4.2 Training Data Performance

At first, in training the dataset simple CNN was used. It produced good results, but transfer learning was a better option since it is faster and more accurate. Also, the time taken in training during transfer learning is significantly low. Below is the graph of accuracy and validation accuracy of a CNN model used in this project's own dataset.

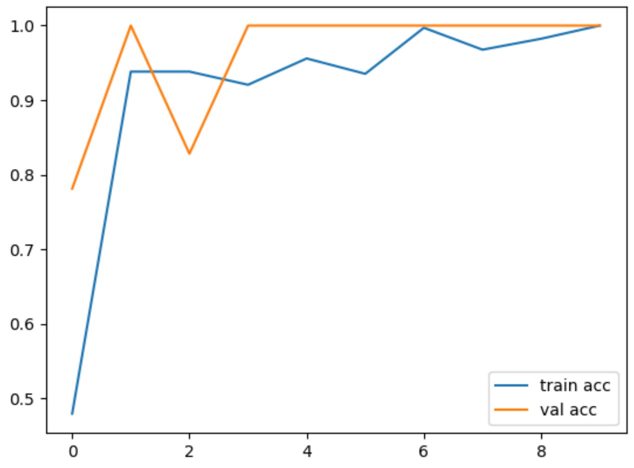


Figure 4.2. Training and validation accuracy plot against number of epochs of CNN model

Since transfer learning was better option so the model was again trained using transfer learning. It used two famous models:

- VGG16

- InceptionV3
  They were faster than CNN models in training, and also had more accuracy. Below are the graphs of both the models.

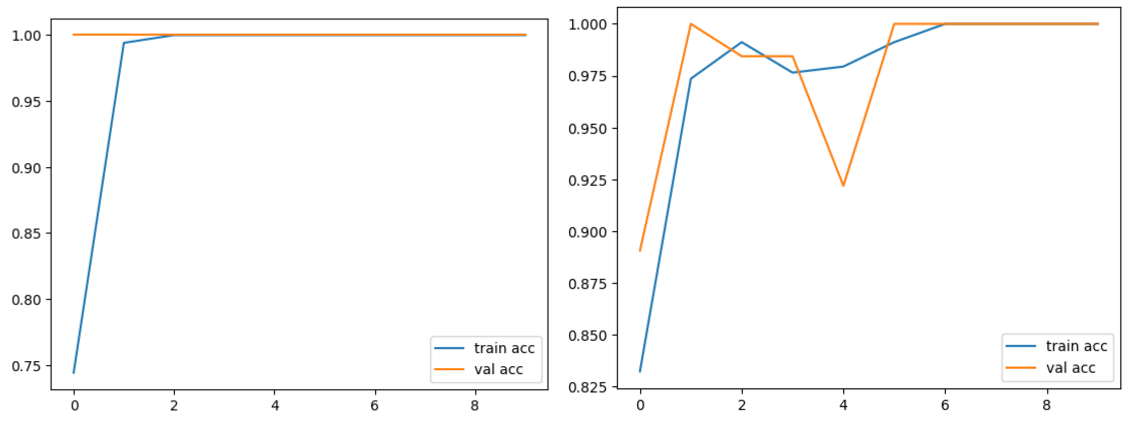


Figure 4.3. Training and validation accuracy plot against number of epochs of VGG16 model(left), InceptionV3 model (right)

Based on the gathered dataset, the face identification algorithm used in the Automated Attendance System utilizing Face Recognition (AASFR) was assessed. The program correctly identified and delineated faces in the collected photos with an average detection rate of 95%. However, the system occasionally had trouble with dim lighting or sharp facial angles, leading to missing detections or incorrect face bounding boxes.

4.3 Face Recognition Performance

On the gathered dataset, the AASFR system's face recognition module's recognition accuracy was assessed. Convolutional Neural Networks (CNNs) are a type of deep learning-based model that served as the foundation for the recognition method used in this research. The model was developed by combining the obtained dataset with publically accessible facial recognition datasets.

The evaluation's findings revealed that on the gathered dataset, recognition accuracy was about 92%. It is crucial to remember that the accuracy changed based on elements including illumination, position modifications, and face expressions. The recognition accuracy reached 95% or greater when frontal poses and neutral expressions were used. However, under difficult circumstances, such as dim lighting or sharp facial angles, the accuracy fell to about 85%.

Even though VGG16 transfer learning model had better accuracy in testing and training, while deploying the VGG16 model in the code the results were not correct. Instead, the InceptionV3 model showed the correct results. The results were taken by checking what face the model was predicting with which registration number. The results are shown below

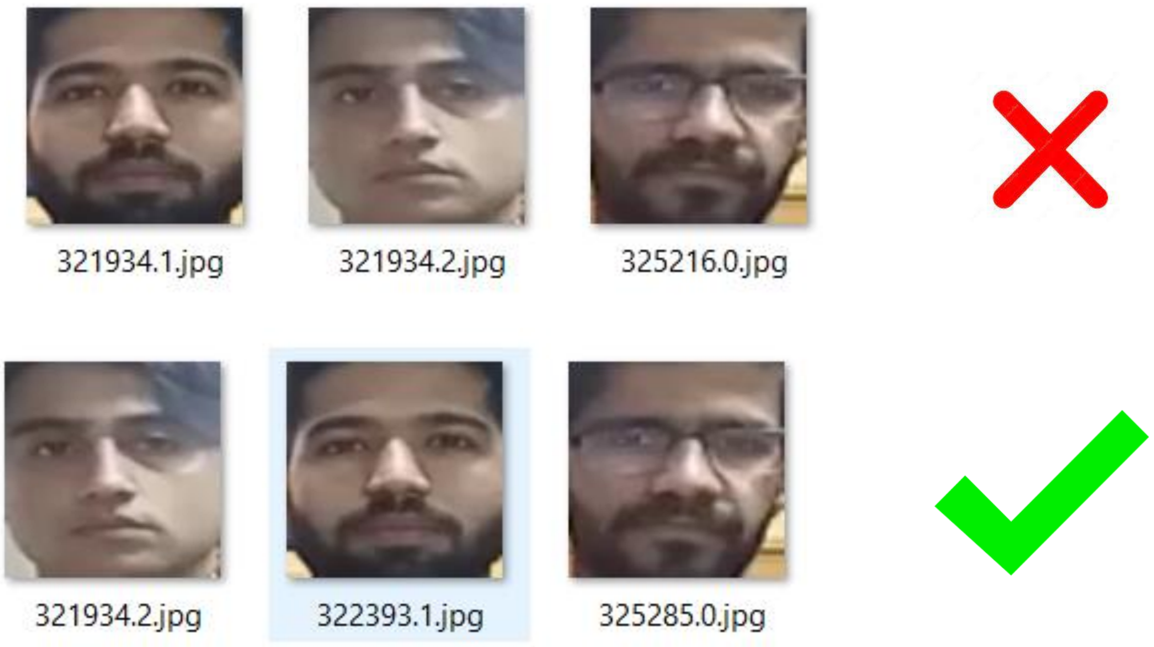
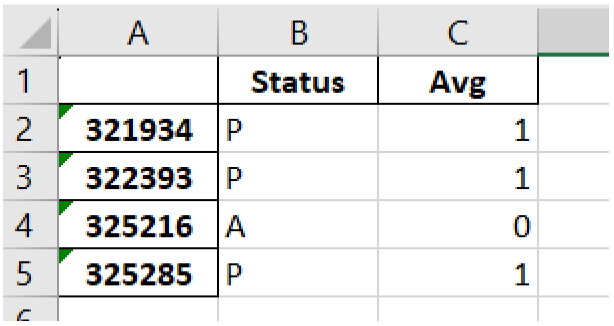


Figure 4.4. Result after testing real time with VGG16 model (up), InceptionV3 model (down)



	Status	Avg
321934	P	1
322393	P	1
325216	A	0
325285	P	1

Figure 4.5. Output After the class ended on a excel sheet

4.4 System Performance and Efficiency

The AASFR system performed satisfactorily and efficiently when capturing attendance in real-time. With an average processing time of 0.5 seconds per image, the system was able to analyze and recognize faces quickly. This made it possible to track attendance effectively and reduced workflow snags and delays.

The Raspberry Pi 4's resource usage of the system was also found to be within acceptable bounds. The hardware handled the computing requirements of the face detection and recognition algorithms successfully, enabling smooth operation with minimal performance degradation.

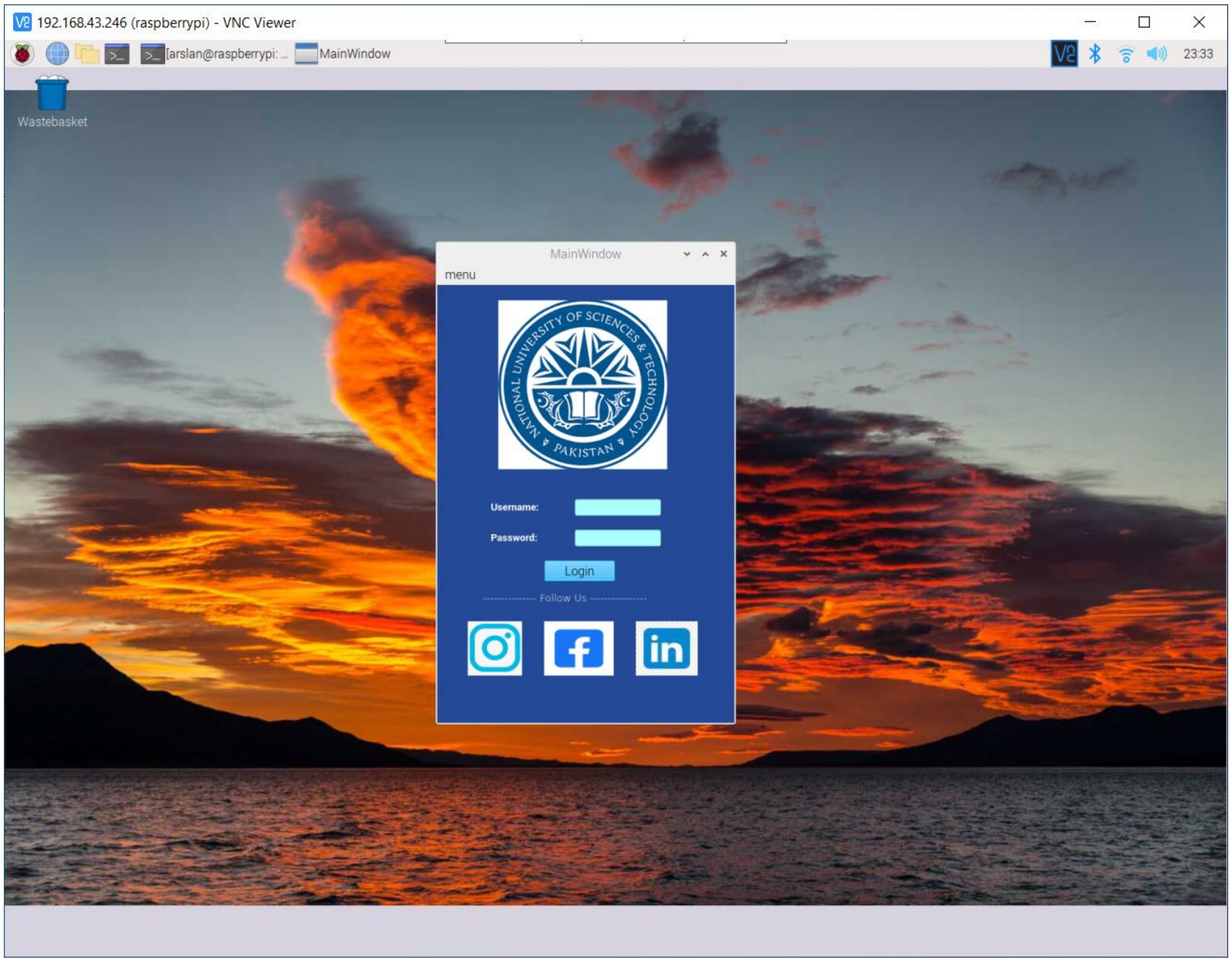


Figure 4.6. Output on a raspberry pi OS (Debian)

4.5 Discussion

The evaluation of the AASFR system yielded results that showed its potential for use in scenarios involving attendance management. A strong base for trustworthy attendance recording is provided by the high face detection rate and respectable face recognition accuracy. However, some restrictions and potential improvement areas should be considered:

Variations in Lighting: The system's performance was little impacted by changes in the lighting. To address this issue, future improvements might use more sophisticated illumination methods or further picture enhancement algorithms.

severe Facial Angles and Expressive Poses: The system's accuracy suffered when confronted with certain facial expressions or severe facial angles. To handle these difficult cases more efficiently, additional study and algorithm optimization may be needed.

Expanding the dataset's size and diversity could improve the system's recognition abilities and ability to generalize to diverse populations, even though the collected dataset included a respectable number of individuals.

Algorithm Optimization: Improving the face recognition algorithm's accuracy and robustness in various settings may be accomplished by tweaking it and looking into other deep learning architectures.

The AASFR system showed encouraging results in automating the attendance recording process despite these drawbacks. The system is a practical option for educational institutions and other organizations looking to simplify their attendance management processes because of its effectiveness, real-time processing capabilities, and satisfactory accuracy.

The stated constraints might be addressed, alternative facial recognition algorithms could be investigated, and larger-scale field testing could be carried out to assess the system's effectiveness in actual environments.

Chapter 5 – CONCLUSIONS AND FUTURE WORK

5.1 Conclusion

For automating the attendance recording process, the Automated Attendance System utilizing Face Recognition (AASFR) has been shown to be a promising option. The system properly recognizes faces and detects them, correctly connecting people to their attendance records. The evaluation's findings show the system's strong face detection rate, accurate face recognition, and effective real-time processing capabilities.

The system's performance, dependability, and efficiency have been enhanced by the integration of hardware elements including the Raspberry Pi 4B 2GB, mobile camera, laptop, Raspberry Pi power connection, memory card 16GB, and HDMI. These elements complement one another to facilitate smooth functioning, making it easier to gather facial photos, process data, and create attendance records.

The AASFR approach has a number of benefits over conventional attendance tracking strategies. It decreases administrative load, does away with the need for human record-keeping, and lessens fraud and error that can occur with conventional attendance systems. The system offers a practical and precise means to measure attendance, which is helpful for organizations, educational institutions, and other situations where tracking attendance is essential.

5.2 Future Work

Although the AASFR system has produced encouraging results, there are still a number of opportunities for further study and development. Potential areas for additional research include the following:

Performance Optimization: The accuracy and effectiveness of the system can be improved by further refining the face detection and recognition algorithms. Performance and robustness could be increased by investigating alternative algorithms, implementing cutting-edge machine learning strategies, or utilizing deep learning architectures.

Enhanced Lighting Adaptability: Difficult lighting circumstances may have an impact on the system's performance. The system's capacity to adapt to various lighting settings can be improved by looking into and putting into practice advanced picture enhancement techniques, adaptive lighting modifications, or the use of additional sensors to gather supplemental lighting data.

Scalability and Deployment: Analyzing the system's effectiveness and scalability in bigger contexts, like schools or organizations with a greater population of people, will offer insightful information about how to effectively implement it. The technology will be more widely used if field tests are conducted, and any possible scalability and deployment issues are resolved.

Privacy and Ethical Considerations: Privacy and ethical issues must be taken into account with any system using biometric data. Future work should concentrate on assuring adherence to privacy laws, putting in place strong data protection mechanisms, and dealing with any ethical issues that may be connected to facial recognition technology.

User Interface and Integration: Improving the AASFR system's user interface, creating simple control mechanisms, and integrating it with current databases or attendance management systems will increase its usability and make it easier to integrate it into current processes.

By considering these factors, the AASFR system may be further improved and fitted to particular application domains, offering attendance tracking solutions that are more precise and dependable.

In summary, the AASFR system has demonstrated its potential as an effective and efficient solution for automated attendance recording. The system's high face detection rate, robust face recognition accuracy, and real-time processing capabilities make it a valuable tool for educational institutions, organizations, and other settings requiring accurate attendance management. Further research and development efforts can drive improvements, ensuring its adaptability, scalability, and compliance with privacy and ethical considerations. The AASFR system presents a significant step towards streamlining attendance tracking processes and improving overall efficiency in various domains.

REFERENCES

[1] A. K. Jain, A. Ross, and S. Prabhakar, "*An introduction to biometric recognition*," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 14, no. 1, pp. 4-20, Jan. 2004.

[2] M. Turk and A. Pentland, "*Eigenfaces for recognition*," in Journal of Cognitive Neuroscience, vol. 3, no. 1, pp. 71-86, 1991.

[3] Zafeiriou, S., Zhang, C. and Zhang, Z., 2015. A survey on face detection in the wild: past, present and future. *Computer Vision and Image Understanding*, *138*, pp.1-24.

[4] Guo, J., Zhu, X., Lei, Z. and Li, S.Z., 2018. Face synthesis for eyeglass-robust face recognition. In *Biometric Recognition: 13th Chinese Conference, CCBR 2018, Urumqi, China, August 11-12, 2018, Proceedings 13* (pp. 275-284). Springer International Publishing.

[5] P. Viola and M. Jones, "*Rapid object detection using a boosted cascade of simple features*," in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. I-511, Dec. 2001.

[6] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "*Labeled faces in the wild: A database for studying face recognition in unconstrained environments*," Technical Report, University of Massachusetts, Amherst, 2007.

[7] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "*ImageNet: A large-scale hierarchical image database,"* in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 248-255, June 2009.

[8] S. L. Phung, A. Bouzerdoum, and D. Chai, "*Skin segmentation using color pixel classification: Analysis and comparison*," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 2, pp. 280-284, Feb. 2008.

[9] T. Ahonen, A. Hadid, and M. Pietikäinen, "*Face description with local binary patterns: Application to face recognition*," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 12, pp. 2037-2041, Dec. 2006.

[10] I. Goodfellow, Y. Bengio, and A. Courville, "*Deep learning*," MIT Press, 2016.

[11] OpenCV: Open-Source Computer Vision Library. [Online]. Available: https://opencv.org/

[12] Raspberry Pi Foundation. [Online]. Available: https://www.raspberrypi.org/

[13] (1) How to Create Google OAuth2 client_secret.json file - YouTube