Smart Access Control For Resident With Transfer Learning and EasyOCR

A REPORT

submitted by

Sanjil K C - 20BCE1855

Karthic S - 20BCE1840

Devarinti Dhapatla Puneeth Reddy - 20BCE1852

in partial fulfilment for the award

of

B. Tech. Computer Science and Engineering

School of Computer Science and Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

November 2023

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

DECLARATION

We hereby declare that the project entitled **"Smart Access Control for Resident with Transfer Learning and EasyOCR"** submitted by us to the School of Computer Science and Engineering, Vellore Institute of Technology, Chennai Campus, Chennai 600127 in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology – Computer Science and Engineering** is a record of work carried out by us**.** We further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.

Signature

Sanjil K C - 20BCE1855
Karthic S - 20BCE1840
Devarinti Dhapatla Puneeth Reddy - 20BCE1852

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

CERTIFICATE

The project report entitled "**Smart Access Control for Resident with Transfer Learning and EasyOCR**" is prepared and submitted by **Sanjil K C (20BCE1855), Karthic S (20BCE1855), Devarinti Dhapatla Puneeth Reddy (20BCE1712)**. It has been foundsatisfactory in terms of scope, quality and presentation as partial fulfilment of the requirements forthe award of the degree of **Bachelor of Technology – Computer Science and Engineering (Undergraduate)** in Vellore Institute of Technology, Chennai, India.

Examined by:

Dr. Premanand V

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. Premanand V**, Assistant Professor (Sr), School of Computer Science and Engineering, for her consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course ofthe project work.

We are extremely grateful to **Dr. Ganesan R**, Dean, School of Computer Science and Engineering, VIT Chennai, for extending the facilities of the school towards our project and for her unstinting support.

We express our thanks to our Head of the Department **Dr. P. Nithyanandam** for their support throughout the course of this project.

We also take this opportunity to thank all the faculty of the school for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

TABLE OF CONTENTS

ABSTRACT

The enhanced security gate entry system now incorporates a face recognition module to further strengthen access control measures. In addition to the number plate detection system, the project integrates a camera capable of capturing facial images of individuals approaching the gate.

The face recognition module employs advanced computer vision algorithms to analyze and identify faces against a predefined database of authorized personnel. Upon successful face recognition, the system cross-verifies the number plate information with the authorized vehicles list stored in the Arduino-based controller. Only when both the number plate and facial recognition processes confirm the vehicle and driver's authorization, the gate is automatically opened.

This multi-layered approach significantly enhances the security of the premises by adding a biometric factor to the access control system. The integration of face recognition not only provides an additional layer of verification but also ensures that the access is granted only to the authorized driver associated with the registered vehicle.

Moreover, the system continues to log the time and date of entry and exit, now associating this information with both the vehicle's number plate and the recognized face. This comprehensive log can be valuable for security audits and monitoring visitor movements with an added layer of identification through facial recognition.

By combining number plate detection with face recognition, this project provides a robust and reliable security solution suitable for a variety of applications, including residential estates, commercial properties, and public institutions where stringent access control is essential. The modular design of the system allows for flexibility and scalability, ensuring adaptability to different security requirements.

OBJECTIVE

This project comprises a multifaceted set of objectives aimed at developing an advanced and real-time number plate detection system. Leveraging sophisticated computer vision algorithms and cutting-edge image processing techniques, the system is meticulously designed to accurately identify and extract number plates from images or video frames. The primary focus is on ensuring the system's robustness, particularly under challenging conditions such as varying lighting scenarios and diverse plate orientations. This technological foundation seeks to establish a reliable and responsive number plate recognition system capable of operating seamlessly in real-time.

In addition to the number plate detection system, the project introduces a strategic enhancement to the security infrastructure through the integration of a face recognition module. This supplementary module is poised to significantly elevate the system's identification capabilities, employing state-of-the-art facial recognition algorithms. The objective is to ensure precise and dependable authentication, even when faced with challenging scenarios like low-light conditions or partially obscured and tilted faces. This augmentation acknowledges the evolving landscape of security needs, introducing a biometric layer to the access control system that complements the vehicular identification capabilities of the number plate detection system.

Furthermore, the integration of the face recognition module extends beyond mere identification; it plays a pivotal role in linking recognized faces with the access control privileges associated with a pre-established database of authorized personnel. The seamless integration of facial recognition and number plate detection results enables a comprehensive approach to access control, where the system cross-references the identified individuals with the database to determine and grant access privileges.

The ultimate objective of this comprehensive project is the seamless amalgamation of the number plate detection system, facial recognition module, and access control mechanisms into a cohesive and automated security gate entry solution. Rigorous testing and validation protocols in diverse real-world settings are integral to ensuring the system's reliability, efficiency, and heightened security across a broad spectrum of scenarios. In achieving these goals, the project aims to deliver a versatile, adaptive, and robust solution that addresses the evolving needs of modern access control and security requirements.

INTRODUCTION

In recent years, the escalating frequency of security threats has underscored the imperative for robust and effective security systems. Among these systems, the automated security gate entry system assumes a pivotal role in upholding safety and security in restricted areas. These advanced systems leverage cutting-edge technology to swiftly and accurately identify vehicles, permitting only authorized entry and thwarting access to unauthorized ones. The impact of such systems extends beyond the reduction of theft and vandalism; they serve as a formidable deterrent against potential security threats attempting to breach restricted areas.

Automated security gate entry systems, renowned for their high accuracy, offer cost-effective solutions that minimize the risk of errors or false positives. Tailored to enhance the security posture of restricted zones, these systems ensure that only authorized vehicles gain entry, thereby mitigating the potential threat of unauthorized personnel accessing secured areas. To achieve precise identification of vehicle number plates, the system employs sophisticated technologies, including computer vision, character recognition (EasyOCR), and ensemble learning methods such as the cascade classifier. These integrated technologies collaborate seamlessly to accurately identify number plates, facilitating the system in its mission to authenticate entries effectively and efficiently.

In the context of this project, an additional layer of security is introduced through the incorporation of a face recognition module. This enhancement expands the capabilities of the system by employing ensemble learning methods not only for number plate identification but also for facial recognition. This multifaceted approach further elevates the accuracy and reliability of automated security gate entry systems. By leveraging these technologies, the project aims to ensure swift and precise identification of vehicles and individuals, granting access exclusively to authorized entities and minimizing the risk of potential security threats. In essence, automated security gate entry systems, fortified by this project's innovative integration of ensemble learning and face recognition, stand as indispensable guardians of safety and security in restricted areas, promising heightened accuracy and effectiveness in their operation.

LITERATURE SURVEY

Several papers propose using Automatic Number Plate Recognition (ANPR) and Number Plate Recognition (NPR) technology in vehicle identification systems. These systems capture images of vehicles' number plates and extract information using image processing techniques. The extracted information is compared with a database to identify the vehicle and grant or deny access.

Paper [1] presents an automatic number plate recognition (ANPR) system using Arduino for vehicle identification at security gates. The system captures images of the vehicle number plates and processes them to extract the number plate information using image processing techniques. The extracted information is compared with the database to identify the vehicle and grant or deny access. The experimental results show that the proposed system has high accuracy in identifying vehicles.

Paper [2] proposes a smart parking system that uses automatic number plate recognition (ANPR) to identify and authenticate vehicles. The system captures images of the vehicle number plates using a camera and processes them using image processing algorithms to extract the number plate information. The extracted information is matched with the database to identify the vehicle and grant or deny access. The system also provides real-time information about parking availability and guidance to the nearest available parking slot.

Paper [3] proposes an automated car parking system that uses number plate recognition (NPR) to identify and authenticate vehicles at the entry and exit points. The system captures images of the vehicle number plates and processes them using image processing algorithms to extract the number plate information. The extracted information is compared with the database to identify the vehicle and grant or deny access. The experimental results show that the proposed system has high accuracy in identifying vehicles and can handle high traffic volumes.

Paper [4] proposes a smart car parking system that uses number plate recognition (NPR) to identify and authenticate vehicles. The system captures images of the vehicle number plates using a camera and processes them using image processing algorithms to extract the number plate information. The extracted information is matched with the database to identify the vehicle and grant or deny access. The system also provides real-time information about parking availability and guidance to the nearest available parking slot.

Paper [5] proposes an automated car parking system that uses IoT and number plate recognition (NPR) to identify and authenticate vehicles. The system captures images of the vehicle number plates and processes them using image processing algorithms to extract the number plate information. The extracted information is sent to the cloud using IoT technology for verification and comparison with the database. The experimental results show that the proposed system has high accuracy in identifying vehicles and can handle high traffic volumes.

Paper [6] presents a vehicle number plate detection and recognition system using Arduino for security gate entry. The system captures images of the vehicle number plates using a camera and processes them using image processing algorithms to detect and extract the number plate information. The extracted information is compared with the database to identify the vehicle and grant or deny access. The experimental results show that the proposed system has high accuracy in identifying vehicles.

Paper [7] presents an Arduino-based smart car parking system using number plate recognition. The system consists of a camera, an Arduino board, and an LCD display. The camera captures the image of the number plate of the vehicle entering the parking lot, and the number plate information is extracted using image processing techniques. The system also includes a database to store the number plate information of the parked vehicles. The authors report that the system has high accuracy in recognizing the number plates and can efficiently manage the parking lot.

Paper [8] proposes an automated car parking system that combines image processing and RFID technology to detect the entry and exit of vehicles. The system uses a camera to capture the license plate of the vehicle and a RFID reader to read the RFID tag attached to the vehicle. The captured license plate image is processed to extract the plate number, which is then compared with the plate number stored in the RFID tag to verify the identity of the vehicle. The proposed system achieved a recognition rate of 97.14% in the license plate recognition experiment and demonstrated good performance in the parking control experiment.

Paper [9] presents a vehicle access control system that is based on number plate recognition. The system uses a camera to capture the license plate of the vehicle and a Raspberry Pi to process the image and recognize the plate number. The recognized plate number is then compared with the plate numbers in the database to determine whether the vehicle is authorized to access the parking lot. The proposed system achieved a recognition rate of 96.6% in the license plate recognition experiment and demonstrated good performance in the access control experiment.

Paper [10] proposes a smart car parking system that is based on number plate recognition. The system uses a camera to capture the license plate of the vehicle and a Raspberry Pi to process the image and recognize the plate number. The recognized plate number is then compared with the plate numbers in the database to determine whether the vehicle is authorized to park. The system also includes an automatic payment system that calculates the parking fee based on the parking duration and accepts payment through a mobile application. The proposed system demonstrated good performance in the license plate recognition experiment and the payment experiment.

The experimental results show that the proposed systems have high accuracy in identifying vehicles, even in high traffic volumes, and efficiently manage parking lots. The recognition rates achieved range from 96.6% to 97.14%, demonstrating good performance in license plate recognition and access control experiments. ANPR and NPR technology have been shown to be effective in vehicle identification and access control systems, improving parking management and reducing traffic congestion. The use of these technologies can also provide valuable data on vehicle usage patterns and parking behavior, contributing to the development of smart cities

PROPOSED SYSTEM

- **Module 1 - Pre-processing**

In recent years, the use of computer vision and deep learning techniques has revolutionized the field of image processing. One such application of these techniques is automated license plate recognition (ALPR), which has various practical applications such as automated toll collection, parking lot management, and law enforcement.

To recognize license plates, high-resolution cameras are commonly used, which capture the image of the number plate. The captured image is then processed using pre-trained cascade classifiers, which help extract the exact cropped image of the number plate from the captured image. Cascade classifier technique is used to crop the number plate from the vehicle image. The extracted image is then passed through an Optical Character Recognition (OCR) model, such as EasyOCR, which recognizes the characters on the number plate.

However, OCR models are not always accurate and may sometimes produce errors in the recognized text. To address this issue, the obtained text is further refined using regular expression (regex) to ensure uniformity and increase accuracy. Finally, the recognized license plate numbers are stored in a resident CSV file, which can be used for various applications such as automated parking lot management.

- **Module 2 - Character Extraction**

The characters from the cropped image are extracted using EasyOCR module. And the obtained characters are refined using regex. Once the license plate number is recognized, it is searched for in the resident CSV file, which contains a list of permitted vehicles. If a match is found, the Arduino is simulated to permit entry for the visitor.

This system has various practical applications, such as in gated communities, corporate offices, and parking lots, where the security of the premises is of utmost importance. By automating the process of recognizing license plates, the system can provide a more efficient and secure way of managing visitors and their vehicles.

- **Module 3 - Arduino Hardware**

Connecting Python with Arduino using pyFirmata allows us to establish a communication interface between the two platforms. PyFirmata is a Python library that enables the control of Arduino boards from a Python environment.

With this, we can send and receive data between Python and the Arduino. After establishing the interface, we can connect a servo motor to the Arduino. A servo motor is a rotary actuator that allows for precise control of the position, velocity, and acceleration of the motor shaft. This servo motor can be controlled with the help of commands from Python.

Once the servo motor is connected, we can use it to open the gate if the vehicle belongs to a resident. This can be done by first checking the vehicle's number plate against the resident's CSV file using the automated license plate recognition system, as discussed earlier. If the number plate is found, we can use Python to send a signal to the servo motor connected to the Arduino to open the gate.

This system has various practical applications, such as in gated communities, where the security of the premises is of utmost importance. By automating the process of recognizing license plates and opening the gate, the system can provide a more efficient and secure way of managing the entry of residents into the community.

• **Module 4 - Face Recognition**

The integration of face recognition adds an additional layer of security and identity verification to the automated gate entry system. Utilizing computer vision techniques, the system captures facial images of individuals approaching the gate. These images are then processed through a pre-trained deep learning model for face recognition, enhancing the overall access control mechanism. The face recognition module cross-references the identified individuals with a database of authorized personnel. If a match is found, indicating that the individual is an authorized resident, the Arduino-controlled gate entry system receives a signal to allow entry. This module enhances the security measures by ensuring that access is granted only to recognized and authorized individuals associated with the resident database. The implementation of face recognition further extends the system's applicability to diverse environments, including corporate offices and gated communities, providing an additional biometric layer to the overall

security infrastructure and contributing to a more comprehensive and robust access control system.

IMPLEMENTATION

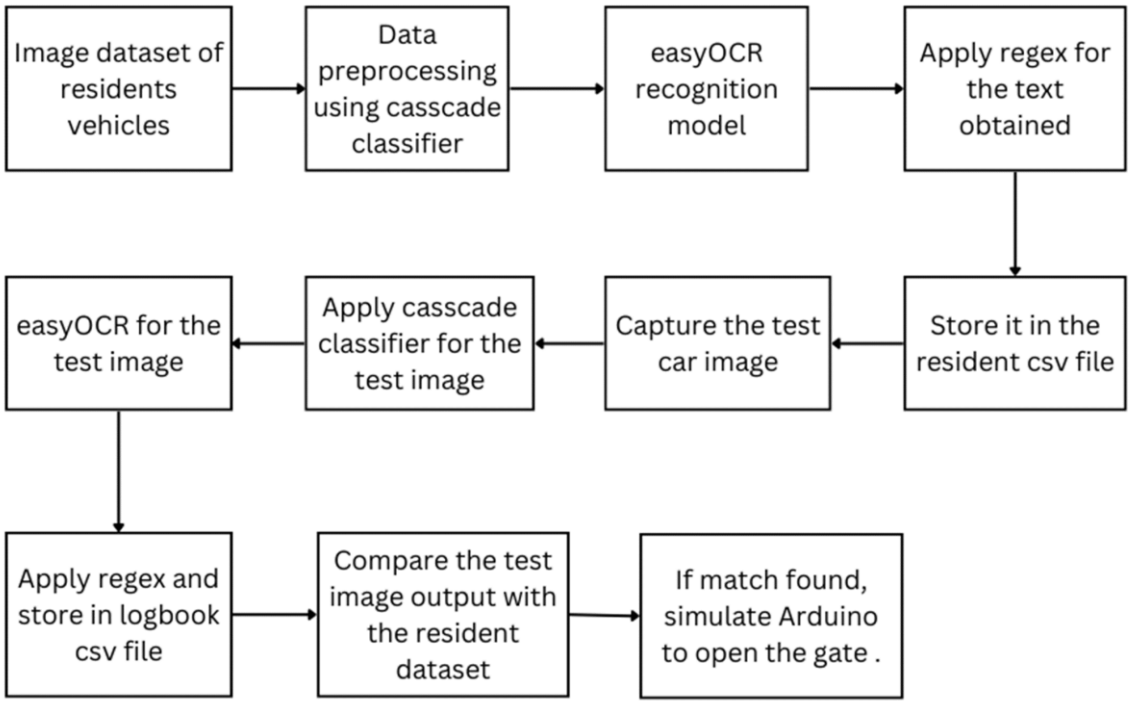


Fig.1. Block diagram for the model

1. Dataset creation: In this paper, the authors have collected their own dataset of car images from their apartment for the purpose of detecting the number plates of the cars. This dataset is important for the development and testing of their algorithm for number plate detection.
2. Detection of Number plate of the residents: The main objective of this study is to detect the number plates of the cars of the residents of an apartment. For this purpose, the authors

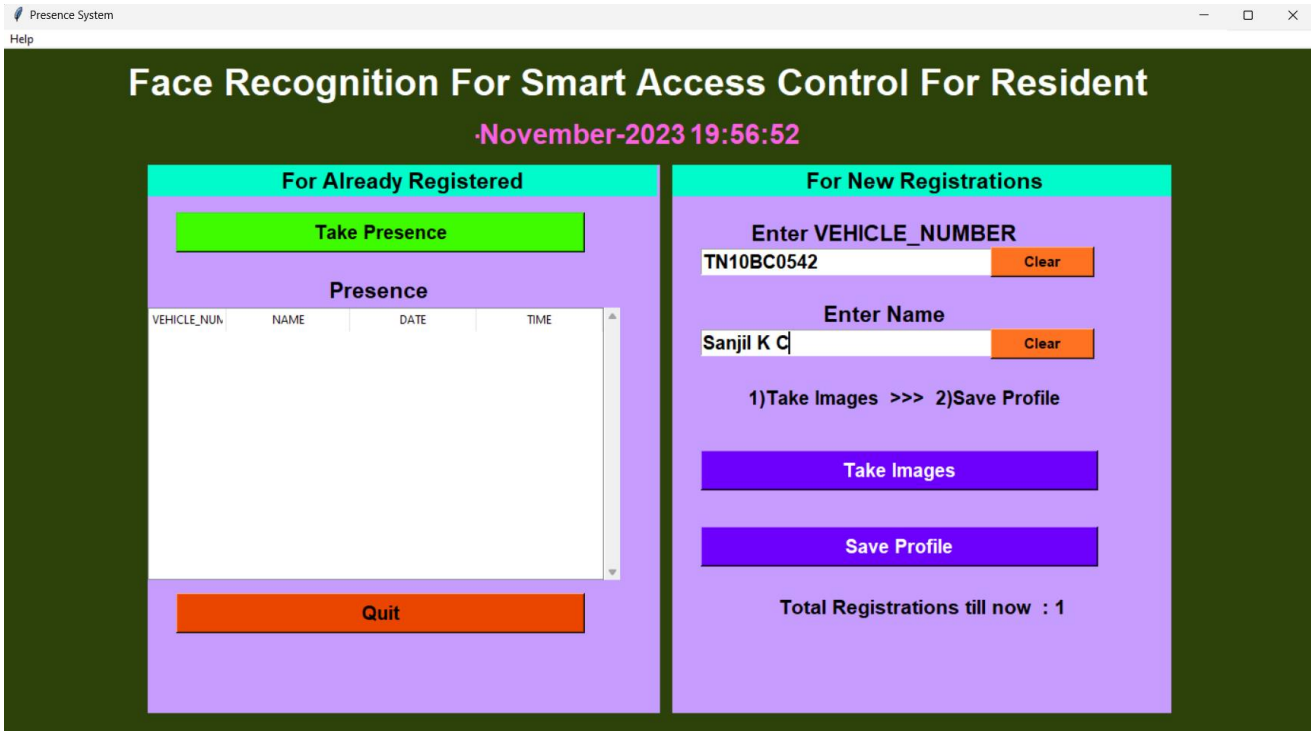
have developed an algorithm that uses image processing techniques to detect and localize the number plate in the car image.

3. Apply EasyOCR Recognition and regex: Once the number plate is detected, the authors have used the EasyOCR recognition model along with regular expressions to extract the alphanumeric characters from the number plate image.
4. Store in Resident CSV file: The extracted number plate information is then stored in a CSV file for each resident in the apartment.
5. Now, capture the test car image and apply the cascade classifier: In order to test the algorithm, a test car image is captured and the cascade classifier is applied to detect the number plate in the image.
6. Apply EasyOCR recognition model along with regex for the test car image: Once the number plate is detected, the EasyOCR recognition model along with regular expressions is applied to extract the alphanumeric characters from the number plate image.
7. Entry in Logbook: The information extracted from the test car image is then logged in a logbook for future reference.
8. Compare the test image with the resident CSV file one by one: The extracted number plate information from the test car image is then compared with the number plates stored in the resident CSV file one by one.
9. If the match is found, then the car's owner is a resident in the apartment: If a match is found between the number plate in the test car image and the number plates stored in the resident CSV file, it indicates that the car belongs to a resident in the apartment.

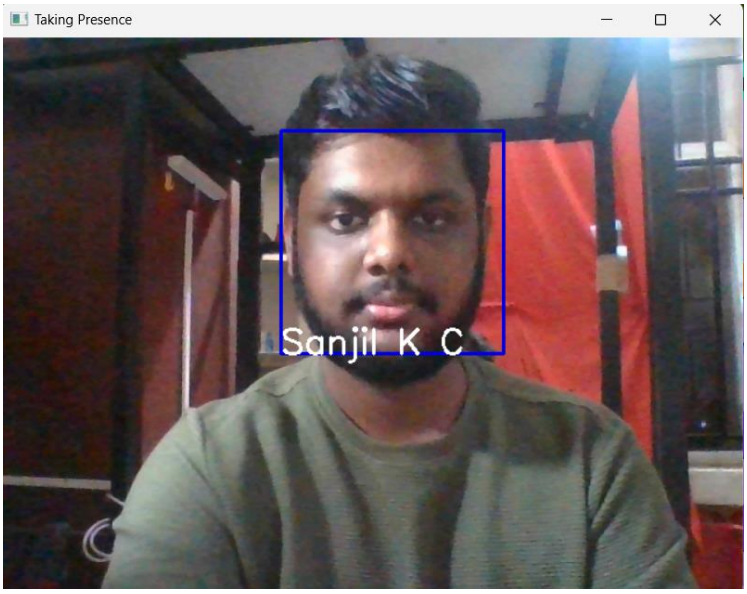
10. Dataset Creation and Algorithm Development : The authors have meticulously collected a unique dataset of car images from their apartment, crucial for the development and testing of their algorithm for number plate detection. This dataset serves as the foundation for training and refining the algorithm, ensuring its effectiveness in identifying number plates accurately.
11. Number Plate Detection and EasyOCR Recognition: The primary objective of the study is to detect the number plates of residents' cars within the apartment. The authors have crafted an algorithm that employs image processing techniques to detect and localize number plates in car images. Following detection, the EasyOCR recognition model, coupled with regular expressions, is applied to extract alphanumeric characters from the number plate image, ensuring precise identification.
12. Integration of Face Recognition for Enhanced Security: In a strategic move to bolster security, the authors have introduced face recognition to the system. Utilizing computer vision, facial images associated with registered vehicles are captured and processed through a pre-trained deep learning model. This face recognition module, seamlessly integrated into the existing system, adds an additional layer of identity verification. When a match is found between the recognized face and the resident database, it serves as an extra confirmation of the individual's authorized residency, contributing to a more robust and comprehensive access control system within the apartment premises.

OUTPUT

Face Recognition for Smart Access Control for Resident



Taking Presence

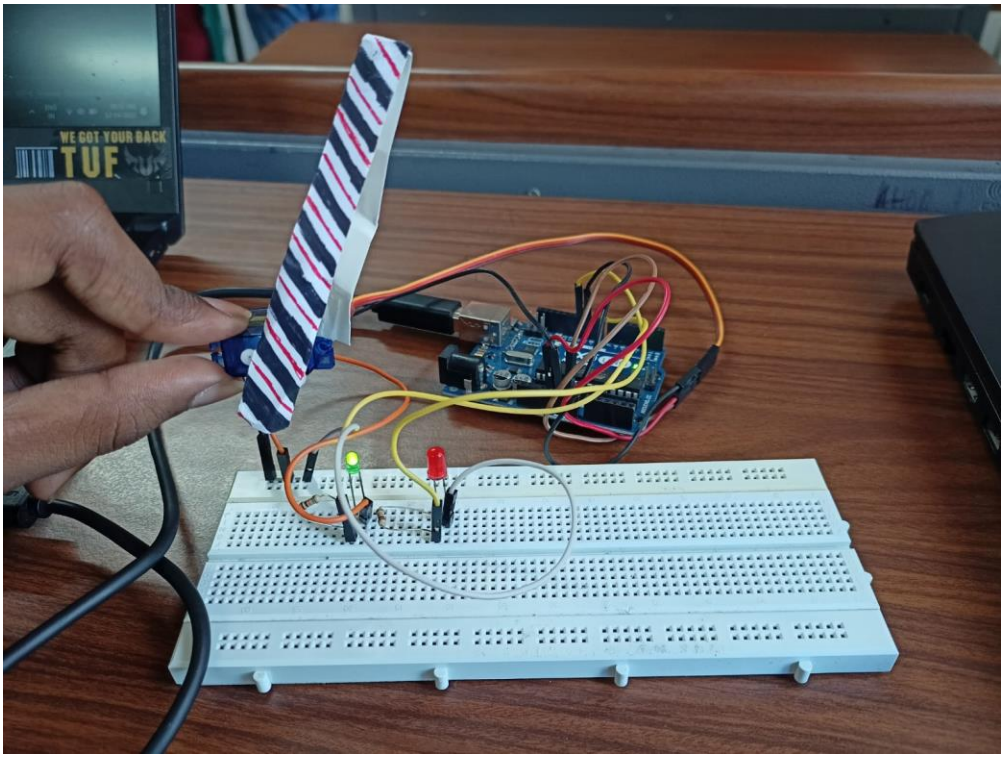


Log File for Take Presence

```
VEHICLE_NUMBER,,Name,,Date,,Time

'TN10BC0542',,Sanjil K C,,22-11-2023,,02:01:32

'TN10BC0542',,Sanjil K C,,22-11-2023,,02:01:43

'TN10BC0542',,Sanjil K C,,22-11-2023,,02:02:06

'TN10BC0542',,Sanjil K C,,22-11-2023,,12:10:45

'TN10BC0542',,Sanjil K C,,22-11-2023,,17:14:04

'TN10BC0542',,Sanjil K C,,22-11-2023,,17:29:00

'TN10BC0542',,Sanjil K C,,22-11-2023,,19:58:54
```

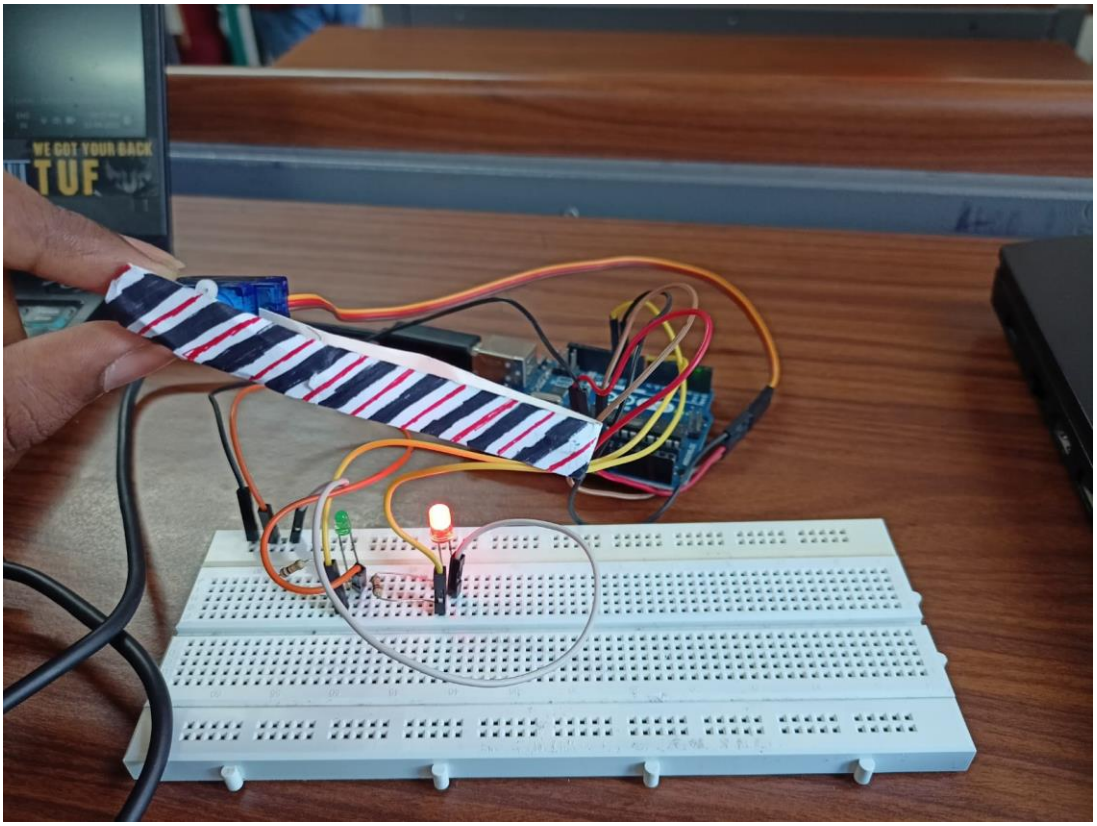
If the car is present, the green light is on and the gate opens.

```
CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.
TN 10 BC 0542]
TN10BC0542
WELCOME HOME! Open the gate

Process finished with exit code 0
```



If the car is not present in the resident dataset, the red light is on and the gate stays closed

```
CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.
IN63809216]
TN638D9216
Not a Resident

Process finished with exit code 0
```

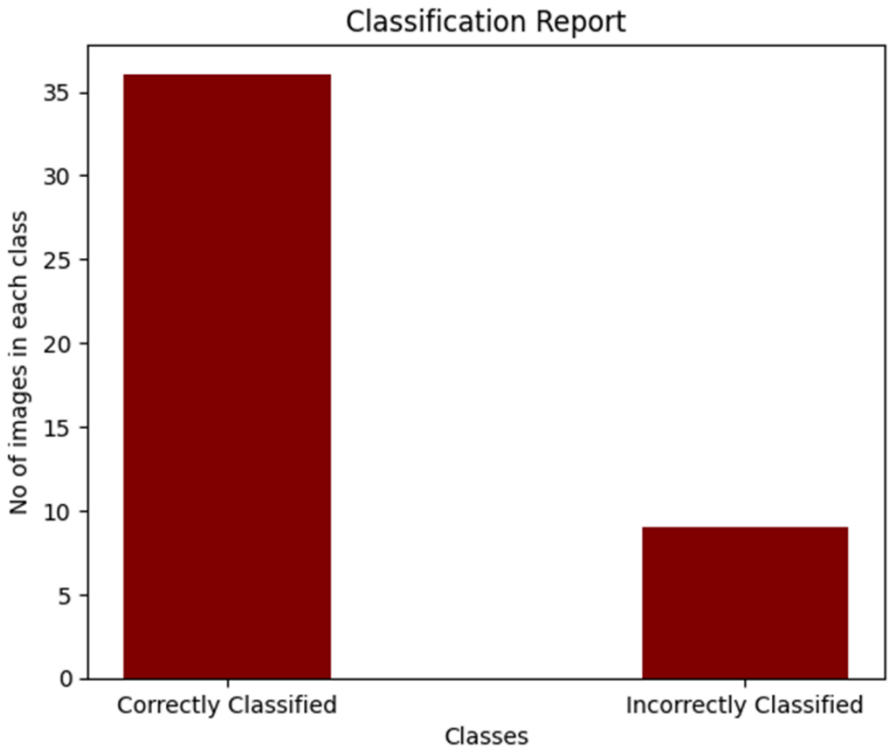


LogBookFile Smart Access Control for Resident with Transfer Learning and EasyOCR:

```
1    numberPlate,timeStamp
2    TN10BC0542,22-11-2023 12:09:50
```

RESULTS AND DISCUSSION

This bar graph shows the number of correctly classified and incorrectly classified images for the Resident car's dataset. The F1-score is found out to be 0.889 indicates that the model is good.



APPENDIX -CODE

1. Face Recognition for Smart Access Control for Resident

```
################################################ IMPORTING
################################################
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox as mess
import tkinter.simpledialog as tsd
import cv2,os
import csv
import numpy as np
from PIL import Image
import pandas as pd
import datetime
import time

################################################ FUNCTIONS
################################################

def assure_path_exists(path):
    dir = os.path.dirname(path)
    if not os.path.exists(dir):
        os.makedirs(dir)

#####################################################################################
#

def tick():
    time_string = time.strftime('%H:%M:%S')
    clock.config(text=time_string)
    clock.after(200,tick)

#####################################################################################
##

def contact():
    mess._show(title='Contact us', message="Please contact us on :
'sanjilkc0@gmail.com' ")

#####################################################################################
##

def check_haarcascadefile():
    exists = os.path.isfile("haarcascade_frontalface_default.xml")
    if exists:
        pass
    else:
        mess._show(title='Some file missing', message='Please contact us for
help')
        window.destroy()

#####################################################################################
##

def save_pass():
    assure_path_exists("TrainingImageLabel/")
```

```
    exists1 = os.path.isfile("TrainingImageLabel\psd.txt")
    if exists1:
        tf = open("TrainingImageLabel\psd.txt", "r")
        key = tf.read()
    else:
        master.destroy()
        new_pas = tsd.askstring('Old Password not found', 'Please enter a new
password below', show='*')
        if new_pas == None:
            mess._show(title='No Password Entered', message='Password not set!!
Please try again')
        else:
            tf = open("TrainingImageLabel\psd.txt", "w")
            tf.write(new_pas)
            mess._show(title='Password Registered', message='New password was
registered successfully!!')
            return
    op = (old.get())
    newp= (new.get())
    nnewp = (nnew.get())
    if (op == key):
        if(newp == nnewp):
            txf = open("TrainingImageLabel\psd.txt", "w")
            txf.write(newp)
        else:
            mess._show(title='Error', message='Confirm new password again!!!')
            return
    else:
        mess._show(title='Wrong Password', message='Please enter correct old
password.')
        return
    mess._show(title='Password Changed', message='Password changed
successfully!!')
    master.destroy()

##################################################################################
##

def change_pass():
    global master
    master = tk.Tk()
    master.geometry("400x160")
    master.resizable(False,False)
    master.title("Change Password")
    master.configure(background="white")
    lbl4 = tk.Label(master,text='    Enter Old
Password',bg='white',font=('comic', 12, ' bold '))
    lbl4.place(x=10,y=10)
    global old
    old=tk.Entry(master,width=25 ,fg="black",relief='solid',font=('comic', 12, '
bold '),show='*')
    old.place(x=180,y=10)
    lbl5 = tk.Label(master, text='   Enter New Password', bg='white',
font=('comic', 12, ' bold '))
    lbl5.place(x=10, y=45)
    global new
    new = tk.Entry(master, width=25, fg="black",relief='solid', font=('comic',
12, ' bold '),show='*')
    new.place(x=180, y=45)
    lbl6 = tk.Label(master, text='Confirm New Password', bg='white',
```

```
font=('comic', 12, ' bold '))
    lbl6.place(x=10, y=80)
    global nnew
    nnew = tk.Entry(master, width=25, fg="black", relief='solid',font=('comic',
12, ' bold '),show='*')
    nnew.place(x=180, y=80)
    cancel=tk.Button(master,text="Cancel", command=master.destroy ,fg="black"
,bg="red" ,height=1,width=25 , activebackground = "white" ,font=('comic', 10, '
bold '))
    cancel.place(x=200, y=120)
    save1 = tk.Button(master, text="Save", command=save_pass, fg="black",
bg="#00fcca", height = 1,width=25, activebackground="white", font=('comic', 10, '
bold '))
    save1.place(x=10, y=120)
    master.mainloop()

####################################################################################
####

def psw():
    assure_path_exists("TrainingImageLabel/")
    exists1 = os.path.isfile("TrainingImageLabel\psd.txt")
    if exists1:
        tf = open("TrainingImageLabel\psd.txt", "r")
        key = tf.read()
    else:
        new_pas = tsd.askstring('Old Password not found', 'Please enter a new
password below', show='*')
        if new_pas == None:
            mess._show(title='No Password Entered', message='Password not set!!
Please try again')
        else:
            tf = open("TrainingImageLabel\psd.txt", "w")
            tf.write(new_pas)
            mess._show(title='Password Registered', message='New password was
registered successfully!!')
            return
    password = tsd.askstring('Password', 'Enter Password', show='*')
    if (password == key):
        TrainImages()
    elif (password == None):
        pass
    else:
        mess._show(title='Wrong Password', message='You have entered wrong
password')

####################################################################################
#####

def clear():
    txt.delete(0, 'end')
    res = "1)Take Images  >>>  2)Save Profile"
    message1.configure(text=res)


def clear2():
    txt2.delete(0, 'end')
    res = "1)Take Images  >>>  2)Save Profile"
    message1.configure(text=res)
```

```
##########################################################################################

def TakeImages():
    check_haarcascadefile()
    columns = ['SERIAL NO.', '', 'VEHICLE_NUMBER', '', 'NAME']
    assure_path_exists("VehicleDetails/")
    assure_path_exists("TrainingImage/")
    serial = 0
    exists = os.path.isfile("VehicleDetails\VehicleDetails.csv")
    if exists:
        with open("VehicleDetails\VehicleDetails.csv", 'r') as csvFile1:
            reader1 = csv.reader(csvFile1)
            for l in reader1:
                serial = serial + 1
        serial = (serial // 2)
        csvFile1.close()
    else:
        with open("VehicleDetails\VehicleDetails.csv", 'a+') as csvFile1:
            writer = csv.writer(csvFile1)
            writer.writerow(columns)
            serial = 1
        csvFile1.close()
    VEHICLE_NUMBER = (txt.get())
    name = (txt2.get())
    if ((name.isalpha()) or (' ' in name)):
        cam = cv2.VideoCapture(0)
        harcascadePath = "haarcascade_frontalface_default.xml"
        detector = cv2.CascadeClassifier(harcascadePath)
        sampleNum = 0
        while (True):
            ret, img = cam.read()
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            faces = detector.detectMultiScale(gray, 1.3, 5)
            for (x, y, w, h) in faces:
                cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
                # incrementing sample number
                sampleNum = sampleNum + 1
                # saving the captured face in the dataset folder TrainingImage
                cv2.imwrite("TrainingImage\ " + name + "." + str(serial) + "." +
VEHICLE_NUMBER + '.' + str(sampleNum) + ".jpg",
                            gray[y:y + h, x:x + w])
                # display the frame
                cv2.imshow('Taking Images', img)
            # wait for 100 miliseconds
            if cv2.waitKey(100) & 0xFF == ord('q'):
                break
            # break if the sample number is morethan 100
            elif sampleNum > 100:
                break
        cam.release()
        cv2.destroyAllWindows()
        res = "Images Taken for VEHICLE_NUMBER : " + VEHICLE_NUMBER
        row = [serial, '', VEHICLE_NUMBER, '', name]
        with open('VehicleDetails\VehicleDetails.csv', 'a+') as csvFile:
            writer = csv.writer(csvFile)
            writer.writerow(row)
        csvFile.close()
        message1.configure(text=res)
    else:
```

```
        if (name.isalpha() == False):
            res = "Enter Correct name"
            message.configure(text=res)

##################################################################################
#######

def TrainImages():
    check_haarcascadefile()
    assure_path_exists("TrainingImageLabel/")
    recognizer = cv2.face_LBPHFaceRecognizer.create()
    harcascadePath = "haarcascade_frontalface_default.xml"
    detector = cv2.CascadeClassifier(harcascadePath)
    faces, VEHICLE_NUMBER = getImagesAndLabels("TrainingImage")
    try:
        recognizer.train(faces, np.array(VEHICLE_NUMBER))
    except:
        mess._show(title='No Registrations', message='Please Register someone
first!!!')
        return
    recognizer.save("TrainingImageLabel\Trainner.yml")
    res = "Profile Saved Successfully"
    message1.configure(text=res)
    message.configure(text='Total Registrations till now  : ' +
str(VEHICLE_NUMBER[0]))

##################################################################################
############3

def getImagesAndLabels(path):
    # get the path of all the files in the folder
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    # create empth face list
    faces = []
    # create empty VEHICLE_NUMBER list
    VEHICLE_NUMBERs = []
    # now looping through all the image paths and loading the VEHICLE_NUMBERs and
the images
    for imagePath in imagePaths:
        # loading the image and converting it to gray scale
        pilImage = Image.open(imagePath).convert('L')
        # Now we are converting the PIL image into numpy array
        imageNp = np.array(pilImage, 'uint8')
        # getting the VEHICLE_NUMBER from the image
        VEHICLE_NUMBER = int(os.path.split(imagePath)[-1].split(".")[1])
        # extract the face from the training image sample
        faces.append(imageNp)
        VEHICLE_NUMBERs.append(VEHICLE_NUMBER)
    return faces, VEHICLE_NUMBERs

##################################################################################
##########

def TrackImages():
    check_haarcascadefile()
    assure_path_exists("Presence/")
    assure_path_exists("VehicleDetails/")
    for k in tv.get_children():
        tv.delete(k)
    msg = ''
```

```
    i = 0
    j = 0
    recognizer = cv2.face.LBPHFaceRecognizer_create()  #
cv2.createLBPHFaceRecognizer()
    exists3 = os.path.isfile("TrainingImageLabel\Trainner.yml")
    if exists3:
        recognizer.read("TrainingImageLabel\Trainner.yml")
    else:
        mess._show(title='Data Missing', message='Please click on Save Profile to
reset data!!')
        return
    harcascadePath = "haarcascade_frontalface_default.xml"
    faceCascade = cv2.CascadeClassifier(harcascadePath);

    cam = cv2.VideoCapture(0)
    font = cv2.FONT_HERSHEY_SIMPLEX
    col_names = ['VEHICLE_NUMBER', '', 'Name', '', 'Date', '', 'Time']
    exists1 = os.path.isfile("VehicleDetails\VehicleDetails.csv")
    if exists1:
        df = pd.read_csv("VehicleDetails\VehicleDetails.csv")
    else:
        mess._show(title='Details Missing', message='Students details are
missing, please check!')
        cam.release()
        cv2.destroyAllWindows()
        window.destroy()
    while True:
        ret, im = cam.read()
        gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
        faces = faceCascade.detectMultiScale(gray, 1.2, 5)
        for (x, y, w, h) in faces:
            cv2.rectangle(im, (x, y), (x + w, y + h), (225, 0, 0), 2)
            serial, conf = recognizer.predict(gray[y:y + h, x:x + w])
            if (conf < 50):
                ts = time.time()
                date = datetime.datetime.fromtimestamp(ts).strftime('%d-%m-%Y')
                timeStamp =
datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
                aa = df.loc[df['SERIAL NO.'] == serial]['NAME'].values
                VEHICLE_NUMBER = df.loc[df['SERIAL NO.'] ==
serial]['VEHICLE_NUMBER'].values
                VEHICLE_NUMBER = str(VEHICLE_NUMBER)
                VEHICLE_NUMBER = VEHICLE_NUMBER[1:-1]
                bb = str(aa)
                bb = bb[2:-2]
                presence = [str(VEHICLE_NUMBER), '', bb, '', str(date), '',
str(timeStamp)]

            else:
                VEHICLE_NUMBER = 'Unknown'
                bb = str(VEHICLE_NUMBER)
            cv2.putText(im, str(bb), (x, y + h), font, 1, (255, 255, 255), 2)
        cv2.imshow('Taking Presence', im)
        if (cv2.waitKey(1) == ord('q')):
            break
    ts = time.time()
    date = datetime.datetime.fromtimestamp(ts).strftime('%d-%m-%Y')
    exists = os.path.isfile("Presence\Presence_" + date + ".csv")
    if exists:
        with open("Presence\Presence_" + date + ".csv", 'a+') as csvFile1:
```

```
            writer = csv.writer(csvFile1)
            writer.writerow(presence)
        csvFile1.close()
    else:
        with open("Presence\Presence_" + date + ".csv", 'a+') as csvFile1:
            writer = csv.writer(csvFile1)
            writer.writerow(col_names)
            writer.writerow(presence)
        csvFile1.close()
    with open("Presence\Presence_" + date + ".csv", 'r') as csvFile1:
        reader1 = csv.reader(csvFile1)
        for lines in reader1:
            i = i + 1
            if (i > 1):
                if (i % 2 != 0):
                    iidd = str(lines[0]) + '   '
                    tv.insert('', 0, text=iidd, values=(str(lines[2]),
str(lines[4]), str(lines[6])))
    csvFile1.close()
    cam.release()
    cv2.destroyAllWindows()

######################################## USED STUFFS
############################################

global key
key = ''

ts = time.time()
date = datetime.datetime.fromtimestamp(ts).strftime('%d-%m-%Y')
day,month,year=date.split("-")

mont={'01':'January',
      '02':'February',
      '03':'March',
      '04':'April',
      '05':'May',
      '06':'June',
      '07':'July',
      '08':'August',
      '09':'September',
      '10':'October',
      '11':'November',
      '12':'December'
      }

######################################## GUI FRONT-END
############################################

window = tk.Tk()
window.geometry("1280x720")
window.resizable(True,False)
window.title("Presence System")
window.configure(background='#2d420a')

frame1 = tk.Frame(window, bg="#c79cff")
frame1.place(relx=0.11, rely=0.17, relwidth=0.39, relheight=0.80)

frame2 = tk.Frame(window, bg="#c79cff")
frame2.place(relx=0.51, rely=0.17, relwidth=0.38, relheight=0.80)
```

```
message3 = tk.Label(window, text="Face Recognition For Smart Access Control For
Resident " ,fg="white",bg="#2d420a" ,width=55 ,height=1,font=('comic', 29, ' bold
'))
message3.place(x=10, y=10)

frame3 = tk.Frame(window, bg="#c4c6ce")
frame3.place(relx=0.52, rely=0.09, relwidth=0.09, relheight=0.07)

frame4 = tk.Frame(window, bg="#c4c6ce")
frame4.place(relx=0.36, rely=0.09, relwidth=0.16, relheight=0.07)

datef = tk.Label(frame4, text = day+"-"+mont[month]+"-"+year+"  |  ",
fg="#ff61e5",bg="#2d420a" ,width=55 ,height=1,font=('comic', 22, ' bold '))
datef.pack(fill='both',expand=1)

clock = tk.Label(frame3,fg="#ff61e5",bg="#2d420a" ,width=55
,height=1,font=('comic', 22, ' bold '))
clock.pack(fill='both',expand=1)
tick()

head2 = tk.Label(frame2, text="                       For New Registrations
", fg="black",bg="#00fcca" ,font=('comic', 17, ' bold ') )
head2.grid(row=0,column=0)

head1 = tk.Label(frame1, text="                       For Already Registered
", fg="black",bg="#00fcca" ,font=('comic', 17, ' bold ') )
head1.place(x=0,y=0)

lbl = tk.Label(frame2, text="Enter VEHICLE_NUMBER",width=20  ,height=1
,fg="black"  ,bg="#c79cff" ,font=('comic', 17, ' bold ') )
lbl.place(x=80, y=55)

txt = tk.Entry(frame2,width=32 ,fg="black",font=('comic', 15, ' bold '))
txt.place(x=30, y=88)

lbl2 = tk.Label(frame2, text="Enter Name",width=20  ,fg="black"  ,bg="#c79cff"
,font=('comic', 17, ' bold '))
lbl2.place(x=80, y=140)

txt2 = tk.Entry(frame2,width=32 ,fg="black",font=('comic', 15, ' bold ')  )
txt2.place(x=30, y=173)

message1 = tk.Label(frame2, text="1)Take Images  >>>  2)Save Profile"
,bg="#c79cff" ,fg="black"  ,width=39 ,height=1, activebackground = "#3ffc00"
,font=('comic', 15, ' bold '))
message1.place(x=7, y=230)

message = tk.Label(frame2, text="" ,bg="#c79cff" ,fg="black"  ,width=39,height=1,
activebackground = "#3ffc00" ,font=('comic', 16, ' bold '))
message.place(x=7, y=450)

lbl3 = tk.Label(frame1, text="Presence",width=20  ,fg="black"  ,bg="#c79cff"
,height=1 ,font=('comic', 17, ' bold '))
lbl3.place(x=100, y=115)

res=0
exists = os.path.isfile("VehicleDetails\VehicleDetails.csv")
if exists:
    with open("VehicleDetails\VehicleDetails.csv", 'r') as csvFile1:
```

```
        reader1 = csv.reader(csvFile1)
        for l in reader1:
            res = res + 1
    res = (res // 2) - 1
    csvFile1.close()
else:
    res = 0
message.configure(text='Total Registrations till now  : '+str(res))

##################### MENUBAR #################################

menubar = tk.Menu(window,relief='ridge')
filemenu = tk.Menu(menubar,tearoff=0)
filemenu.add_command(label='Change Password', command = change_pass)
filemenu.add_command(label='Contact Us', command = contact)
filemenu.add_command(label='Exit',command = window.destroy)
menubar.add_cascade(label='Help',font=('comic', 29, ' bold '),menu=filemenu)

################## TREEVIEW PRESENCE TABLE ####################

tv= ttk.Treeview(frame1,height =13,columns = ('name','date','time'))
tv.column('#0',width=82)
tv.column('name',width=130)
tv.column('date',width=133)
tv.column('time',width=133)
tv.grid(row=2,column=0,padx=(0,0),pady=(150,0),columnspan=4)
tv.heading('#0',text ='VEHICLE_NUMBER')
tv.heading('name',text ='NAME')
tv.heading('date',text ='DATE')
tv.heading('time',text ='TIME')

###################### SCROLLBAR ################################

scroll=ttk.Scrollbar(frame1,orient='vertical',command=tv.yview)
scroll.grid(row=2,column=4,padx=(0,100),pady=(150,0),sticky='ns')
tv.configure(yscrollcommand=scroll.set)

###################### BUTTONS ##################################

clearButton = tk.Button(frame2, text="Clear", command=clear  ,fg="black"
,bg="#ff7221"  ,width=11 ,activebackground = "white" ,font=('comic', 11, ' bold
'))
clearButton.place(x=335, y=86)
clearButton2 = tk.Button(frame2, text="Clear", command=clear2  ,fg="black"
,bg="#ff7221"  ,width=11 , activebackground = "white" ,font=('comic', 11, ' bold
'))
clearButton2.place(x=335, y=172)
takeImg = tk.Button(frame2, text="Take Images", command=TakeImages  ,fg="white"
,bg="#6d00fc"  ,width=34  ,height=1, activebackground = "white" ,font=('comic',
15, ' bold '))
takeImg.place(x=30, y=300)
trainImg = tk.Button(frame2, text="Save Profile", command=psw ,fg="white"
,bg="#6d00fc"  ,width=34  ,height=1, activebackground = "white" ,font=('comic',
15, ' bold '))
trainImg.place(x=30, y=380)
trackImg = tk.Button(frame1, text="Take Presence", command=TrackImages
,fg="black"  ,bg="#3ffc00"  ,width=35  ,height=1, activebackground = "white"
,font=('comic', 15, ' bold '))
trackImg.place(x=30,y=50)
quitWindow = tk.Button(frame1, text="Quit", command=window.destroy  ,fg="black"
```

```
,bg="#eb4600"  ,width=35 ,height=1, activebackground = "white" ,font=('comic',
15, ' bold '))
quitWindow.place(x=30, y=450)

###################### END ####################################

window.configure(menu=menubar)
window.mainloop()

##############################################################################
###################
```

2. Resident dataset preprocessing

```
import cv2
import easyocr
import os
import csv
import re
import time
from datetime import datetime

""" Writing Number plates recognised into List.csv (Resident dataset file) """

def markCars(name):
    with open('D:\\SEM 7\\CSE4015 HCI PROJECT\\SMART ACCESS CONTROL FOR RESIDENT
WITH TRANSFER LEARNING AND EASYOCR\\List.csv', 'r+') as f:
        myDataList = f.readlines()
        nameList = []
        for line in myDataList:
            entry = line.split(',')
            nameList.append(entry[0])
        if name not in nameList:
            now = datetime.now()
            dtString = now.strftime('%H:%M:%S')
            f.writelines(f'\n{name},{dtString}')

""" REGEX and processing """

def numberplate(string):
    # Convert the string into uppercase and remove special characters and
whitespaces
    string = re.sub(r"[^A-Z0-9]", "", string.upper())

    # Check if the string is of length 10
    if len(string) != 10:
        return 0

    def repl(match):
        if match.group() == "I":
            return "T"
    string = re.sub(r"[I]", repl, string[0:2]) + string[2:]

    # Replace the 3rd and 4th characters using mapping
    def replace(match):
        if match.group() == "I":
            return "1"
```

```
        elif match.group() == "T":
            return "1"
        elif match.group() == "O":
            return "0"
        elif match.group() == "L":
            return "4"
        elif match.group() == "Z":
            return "2"
    string = string[:2] + re.sub(r"[ITOLZ]", replace, string[2:4]) + string[4:]

    # Replace the 5th and 6th characters using mapping
    def repl56(match):
        if match.group() == "0":
            return "D"
        elif match.group() == "4":
            return "L"
        elif match.group() == "2":
            return "Z"
        elif match.group() == "1":
            return "I"
    string = string[:4] + re.sub(r"[0421]", repl56, string[4:6]) + string[6:]

    # Replace the last 4 characters using mapping
    string = string[:6] + re.sub(r"[ITOLZ]", replace, string[6:])
    return string


""" Residents dataset pre-processing using Cascade classifier """

plateCascade = cv2.CascadeClassifier("D:\\SEM 7\\CSE4015 HCI PROJECT\\SMART
ACCESS CONTROL FOR RESIDENT WITH TRANSFER LEARNING AND EASYOCR\\List.csv")
minArea = 500
count = 0
path = 'D:\\SEM 7\\CSE4015 HCI PROJECT\\SMART ACCESS CONTROL FOR RESIDENT WITH
TRANSFER LEARNING AND EASYOCR\\CarsDataset'

for cl in os.listdir(path):
    img = cv2.imread(os.path.join(path, cl))
    imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    numberPlates = plateCascade.detectMultiScale(imgGray, 1.1, 4)

    for (x, y, w, h) in numberPlates:
        area = w * h
        if area > minArea:
            imgRoi = img[y:y + h, x:x + w]

    cv2.imwrite("D:\\SEM 7\\CSE4015 HCI PROJECT\\SMART ACCESS CONTROL FOR
RESIDENT WITH TRANSFER LEARNING AND EASYOCR\\NumplateCropped_Sample\\" +
str(count) + ".jpg", imgRoi)
    cv2.waitKey(500)
    count += 1

""" EasyOCR recognition model and CSV file creation """

path = 'D:\\SEM 7\\CSE4015 HCI PROJECT\\SMART ACCESS CONTROL FOR RESIDENT WITH
TRANSFER LEARNING AND EASYOCR\\NumberPlateCropped'

for cl in os.listdir(path):
    img = cv2.imread(os.path.join(path, cl))
```

```
    reader = easyocr.Reader(['en'])
    result = reader.readtext(img)
    if len(result) > 0:                         # Printing the result
        text = result[0][-2]
        res = numberplate(text)                 # Apply regex to remove whitespaces
and special characters
        print(res)
        if res != 0:
            markCars(res)                       # Add the list of cars dataset in the
CSV file List
    else:
        print('No text found.')
    time.sleep(1)
```

3. Numberplate detection

```
import time
import cv2
import easyocr
import re
import csv
from datetime import datetime
import pyfirmata

def markCarsLogbook(name):
    with open('D:\\SEM 7\\CSE4015 HCI PROJECT\\SMART ACCESS CONTROL FOR RESIDENT
WITH TRANSFER LEARNING AND EASYOCR\\LogBookFile.csv', 'r+') as f:
        myDataList = f.readlines()
        nameList = []
        for line in myDataList:
            entry = line.split(',')
            nameList.append(entry[0])
        if name not in nameList:
            now = datetime.now()
            dtString = now.strftime('%d-%m-%Y %H:%M:%S')
            f.writelines(f'\n{name},{dtString}')

def numberplate(string):
    # Convert the string into uppercase and remove special characters and
whitespaces
    string = re.sub(r"[^A-Z0-9]", "", string.upper())

    # Check if the string is of length 10
    if len(string) != 10:
        return 0

    def repl(match):
        if match.group() == "I":
            return "T"

    string = re.sub(r"[I]", repl, string[0:2]) + string[2:]

    # Replace the 3rd and 4th characters using mapping
    def replace(match):
        if match.group() == "I":
            return "1"
        elif match.group() == "T":
            return "1"
```

```
        elif match.group() == "O":
            return "0"
        elif match.group() == "L":
            return "4"
        elif match.group() == "Z":
            return "2"

    string = string[:2] + re.sub(r"[IOLZ]", replace, string[2:4]) + string[4:]

    # Replace the 5th and 6th characters using mapping
    def repl56(match):
        if match.group() == "0":
            return "D"
        elif match.group() == "4":
            return "L"
        elif match.group() == "2":
            return "Z"
        elif match.group() == "1":
            return "I"

    string = string[:4] + re.sub(r"[0421]", repl56, string[4:6]) + string[6:]

    # Replace the last 4 characters using mapping
    string = string[:6] + re.sub(r"[IOLZ]", replace, string[6:])
    return string

""" Open camera to capture the test car image """

plateCascade = cv2.CascadeClassifier("D:\\SEM 7\\CSE4015 HCI PROJECT\\SMART
ACCESS CONTROL FOR RESIDENT WITH TRANSFER LEARNING AND
EASYOCR\\haarcascade_russian_plate_number.xml")
videoCaptureObject = cv2.VideoCapture(0)
minArea = 500
flag1 = True

while flag1:
    ret, frame = videoCaptureObject.read()
    cv2.putText(frame, 'Press q to capture', (10, 30), cv2.FONT_HERSHEY_SIMPLEX,
0.5, (0, 255, 0), 1, cv2.LINE_4)


    imgGray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    numberPlates = plateCascade.detectMultiScale(imgGray, 1.1, 4)
    for (x, y, w, h) in numberPlates:
        area = w * h
        if area > minArea:
            cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)
            imgRoi = frame[y:y + h, x:x + w]
    cv2.imshow("Number plate detection", frame)


    if cv2.waitKey(1) & 0xFF == ord('q'):
        cv2.imwrite("D:\\SEM 7\\CSE4015 HCI PROJECT\\SMART ACCESS CONTROL FOR
RESIDENT WITH TRANSFER LEARNING AND
EASYOCR\\SingleSampleTestBeforeCropping\\NewPicture.jpg", frame)
        flag1 = False
        videoCaptureObject.release()
        cv2.destroyAllWindows()

""" Apply Cascade Classifier for the test image """
```

```
frameWidth = 640  # Frame Width
franeHeight = 480  # Frame Height
minArea = 500

img = cv2.imread("D:\\SEM 7\\CSE4015 HCI PROJECT\\SMART ACCESS CONTROL FOR
RESIDENT WITH TRANSFER LEARNING AND
EASYOCR\\SingleSampleTestBeforeCropping\\NewPicture.jpg")
imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
numberPlates = plateCascade.detectMultiScale(imgGray, 1.1, 4)

for (x, y, w, h) in numberPlates:
    area = w * h
    if area > minArea:
        cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
        imgRoi = img[y:y + h, x:x + w]

cv2.imwrite("D:\\SEM 7\\CSE4015 HCI PROJECT\\SMART ACCESS CONTROL FOR RESIDENT
WITH TRANSFER LEARNING AND EASYOCR\\SingleSampleTest\\Testimg" + ".jpg", imgRoi)

""" EasyOCR recognition for test image """

reader = easyocr.Reader(['en'])
result = reader.readtext(imgRoi)
s = ""
if len(result) > 0:                        # Store the result
    text = result[0][-2]
    print(text)
    s = numberplate(text)                  # Call the regex function to correct the
output
    print(s)
    if s!=0:
        markCarsLogbook(s)                     # Entry of cars in the Logbook CSV
file List
else:
    print('No text found.')

""" Integrating ARDUINO """

from pyfirmata import Arduino, SERVO
from time import sleep

port = 'COM12'
servo_pin = 10
board = Arduino(port)
board.digital[servo_pin].mode = SERVO

def rotateservo(pin, angle):
    board.digital[pin].write(angle)
    sleep(0.015)

flag_gate = 0

""" Compare with the resident dataset """

reader1 = csv.reader(open('D:\\SEM 7\\CSE4015 HCI PROJECT\\SMART ACCESS CONTROL
FOR RESIDENT WITH TRANSFER LEARNING AND EASYOCR\\List.csv', 'r'))

while True:
    try:
```

```
        row1 = next(reader1)
        if row1[0] == s:
            print("WELCOME HOME! Open the gate")
            flag_gate = 1
    except StopIteration:
        break

led_pin1 = 8      # Green light
led_pin2 = 11     # Red light

# Set up the pins as outputs
board.digital[led_pin1].mode = pyfirmata.OUTPUT
board.digital[led_pin2].mode = pyfirmata.OUTPUT

# Initially red light
board.digital[led_pin1].write(0)          # Turn LED 1 off
board.digital[led_pin2].write(1)          # Turn LED 2 on (Red)

if flag_gate == 1:
    board.digital[led_pin1].write(1)      # Turn LED 1 on (Green)
    board.digital[led_pin2].write(0)      # Turn LED 2 off
    for i in range(0, 90):
        rotateservo(servo_pin, i)               # Rotate the servo motor if match
found (GATE OPEN)

    time.sleep(4)

    board.digital[led_pin1].write(0)      # Turn LED 1 off
    board.digital[led_pin2].write(1)      # Turn LED 2 on (Red)

    for i in range(90, 0, -1):            # GATE CLOSE
        rotateservo(servo_pin, i)
else:
    board.digital[led_pin1].write(0)      # Turn LED 1 off
    board.digital[led_pin2].write(1)      # Turn LED 2 on (Red)
    print("Not a Resident")
```

CONCLUSION

In the contemporary landscape where security concerns have taken centre stage, the imperative implementation of automated security gate entry systems in restricted areas becomes paramount. This project represents a significant stride towards addressing these concerns by providing an efficient and effective solution for access management. Through the integration of cutting-edge technologies, the system demonstrates its capacity to swiftly and accurately recognize and authenticate vehicles, permitting entry exclusively to authorized entities and thwarting access to unauthorized ones.

A noteworthy feature of this system lies in its potential for real-time monitoring, robust data management, and remote access capabilities. By reducing the risk of theft, vandalism, and various security threats, the system offers an unparalleled level of protection to restricted areas. Moreover, as technology continues to advance, this project stands as a pivotal contribution to the development of smarter and more secure access control systems. The integration of computer vision, character recognition, and ensemble learning methods, such as the cascade classifier, results in an innovative system with exceptional accuracy in identifying and authenticating number plates.

Crucially, the project goes beyond conventional measures by incorporating a face recognition module, adding an extra layer of security and identity verification. This augmentation enhances

the system's capability to recognize and grant access to authorized individuals associated with the resident database. The comprehensive approach of integrating both vehicular identification through number plates and individual identification through facial recognition solidifies the system's position as a reliable and convenient solution for managing access to restricted areas.

In conclusion, this project not only offers a robust solution for current security needs but also lays the foundation for the development of more sophisticated and secure access control systems. The incorporation of advanced technologies, including face recognition, underscores the commitment to enhancing security and user experience in diverse settings.

Code GitHub Link:

https://github.com/sanjil2/SMART-ACCESS-CONTROL-FOR-RESIDENT-WITH-TRANSFER-LEARNING-AND-EASYOCR

REFERENCES

[1] "Automatic Car Parking System using Number Plate Recognition" by R. Gupta: https://ieeexplore.ieee.org/document/8821924

[2] "Smart Parking System Using Image Processing and Arduino" by R. K. Bagga and M. K. Ghose: https://ieeexplore.ieee.org/document/7893386

[3] "Automated Car Parking System using Number Plate Recognition" by A. G. Deshpande, S. A. Lokhande, and S. N. Nalawade: https://ieeexplore.ieee.org/document/8449859

[4] "Number Plate Recognition System Using Arduino" by M. A. Al-Qutti, M. A. Jassim, and A. A. K. Al-Khalaf: https://ieeexplore.ieee.org/document/8689146

[5] "Smart Parking System Based on Arduino and IoT" by M. I. Pratama and H. W. Nugroho: https://ieeexplore.ieee.org/document/8443691

[6] "Automated Car Parking System with Number Plate Recognition" by S. M. Tamjidul Hoque and M. S. Siam: https://ieeexplore.ieee.org/document/8646984

[7] "Smart Car Parking System using Number Plate Recognition" by R. S. Bhatia and P. S. Pande: https://ieeexplore.ieee.org/document/8448822

[8] "Smart Parking System with Number Plate Recognition Using Arduino" by N. M. M. Nasir, M. S. J. Jaffar, and S. S. S. Singh: https://ieeexplore.ieee.org/document/8217244

[9] "Automated Parking System Based on Number Plate Recognition Using Arduino" by A. D. Bagade and N. A. Patil: https://ieeexplore.ieee.org/document/8658988

[10] "Intelligent Parking System Based on Number Plate Recognition and Arduino" by M. A. Shaikh and M. A. Ansari: https://ieeexplore.ieee.org/document/8294648