CHAPTER 1
INTRODUCTION

1.1: GENERAL BACKGROUND

The success of an institution begins by engaging teaching staff's regular attendance. Having higher attendance result in students higher learning standards and marks and better educational experience. So not only students but also teacher's attendance also matters. So, educational institutions need to have higher quality attendance monitoring.

Even though keeping attendance data is an essential part of educational institute there has been a little advancement in attendance monitoring using finger print attendance system.

Still, many of institutions use traditional handwritten attendance. This makes it hard for teaching staffs to mark their attendance and track their progress. Chances of fraud in this traditional attendance system are higher than it is in automated attendance system. This project will help eliminate the traditional attendance system, minimize manipulation during attendance and record arrival and return of teaching staffs.

In our proposed facial attendance monitoring machine, there is an administrator where he or she must add user information according to data set. There is also a portal for teaching staff where they can monitor their progress. Also, a remark is generated which is an add on feature where the mood of the corresponding individual is predicted.

CHAPTER 2
LITERATURE SURVEY

2.1 : DEEP LEARNING

Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised. Deep-learning architectures such as deep neural networks, deep belief networks, deep reinforcement learning, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance. Artificial neural networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs have various differences from biological brains. Specifically, artificial neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic (plastic) and analogue.

2.2 MTCNN

MTCNN is a python (pip) library written by Github user ipacz, which implements the paper Zhang, Kaipeng et al. “Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks.” MTCNN performs quite fast on a CPU, even though S3FD is still quicker running on a GPU — but that is a topic for another post. MTCNN or Multi-Task Cascaded Convolutional Neural Networks is a neural network which detects faces and facial landmarks on images.

2.3 RELATED WORK

2.3.1 : Automated attendance system using ML approach

The conventional method of taking attendance is done manually by the teacher or the administrator which requires considerable amount of time and efforts also involving errors

and proxy attendance. As the number of students are increasing day by day, it is a challenging task for universities or colleges to monitor and maintain the record of the students. Automated systems involving use of biometrics like fingerprint and iris recognition are well developed in the recent years however, it is intrusive and cost required for deployment on large scale gets increased substantially. To overcome these issues, biometric feature like facial recognition can be used which involves the phases such as image acquisition, face detection, feature extraction, face classification, face recognition and eventually marking the attendance. The algorithms like Viola- Jones and HOG features along with SVM classifier are used to acquire the desired results. Various real time scenarios need to be considered such as scaling, illumination, occlusions and pose. The problem of redundancy in manual records and keeping attendance is solved by this system. Quantitative analysis is done on the basis of PSNR values.

2.3.2 : Automated attendance system using face recognition by k-means algorithms

Attendance plays a crucial role in educational institutions, several of the other industries and workplaces. Nowadays it's taken by standard methodology i.e., Attendance are taken manually. This method takes a great deal of your time and additionally there might be an error. Face recognition system could be a technology capable of distinguishing or confirming someone from a digital image or from a video supply. In this paper, we tend to build a system that marks the presence of students or employees by recognizing their faces and manufacturing the attending sheet automatically. Face Recognition's accuracy rate is definitely littered with the factors like changes in illumination, posture changes, expression changes, and occlusion. In this paper, a K-means clustering algorithmic rule is employed to research the facial expression. The biometric features of the face unit are extracted and also the K-mean clustering technique is used to cluster the face features. Then, SVM methodology is employed to classify the features of the image. It may accomplish high recognition performance with fewer feature numbers. Finally, a report (attendance sheet) is generated for interpretation.

2.3.3 : Attendance monitoring using CCTV

In Institutions/schools attendance maintaining is one of the major works for the faculties to check the strength of a class. The faculties are provided with paper-based attendance. They separately have it for various periods of subjects known log book. They mark the attendance every time when they go to class for their periods. To avoid manual paper-based attendance system nowadays smart attendance monitoring system like biometric facial recognition system is being suggested. It is enormously used in much application such as monitoring the class room using CCTV, Computer –human interaction, Accurate Attendance maintaining and in security issues. This system rectifies the problems in marking the student's entry as absent even they are inside the classroom. In the implementation process, detecting the face, identifying and marking the attendance automatically whether the student is present or not is done. Principle Component Analysis (PCA), Eigen face value detection, Convolutional Neural Network (CNN) are the methods being used in this paper to create an automatic attendance management system. This model is successfully done in comparison with database of student's face to control the movement of people with a predefined protocol.

CHAPTER 3
DESIGN

3.1 : BLOCK DIAGRAM

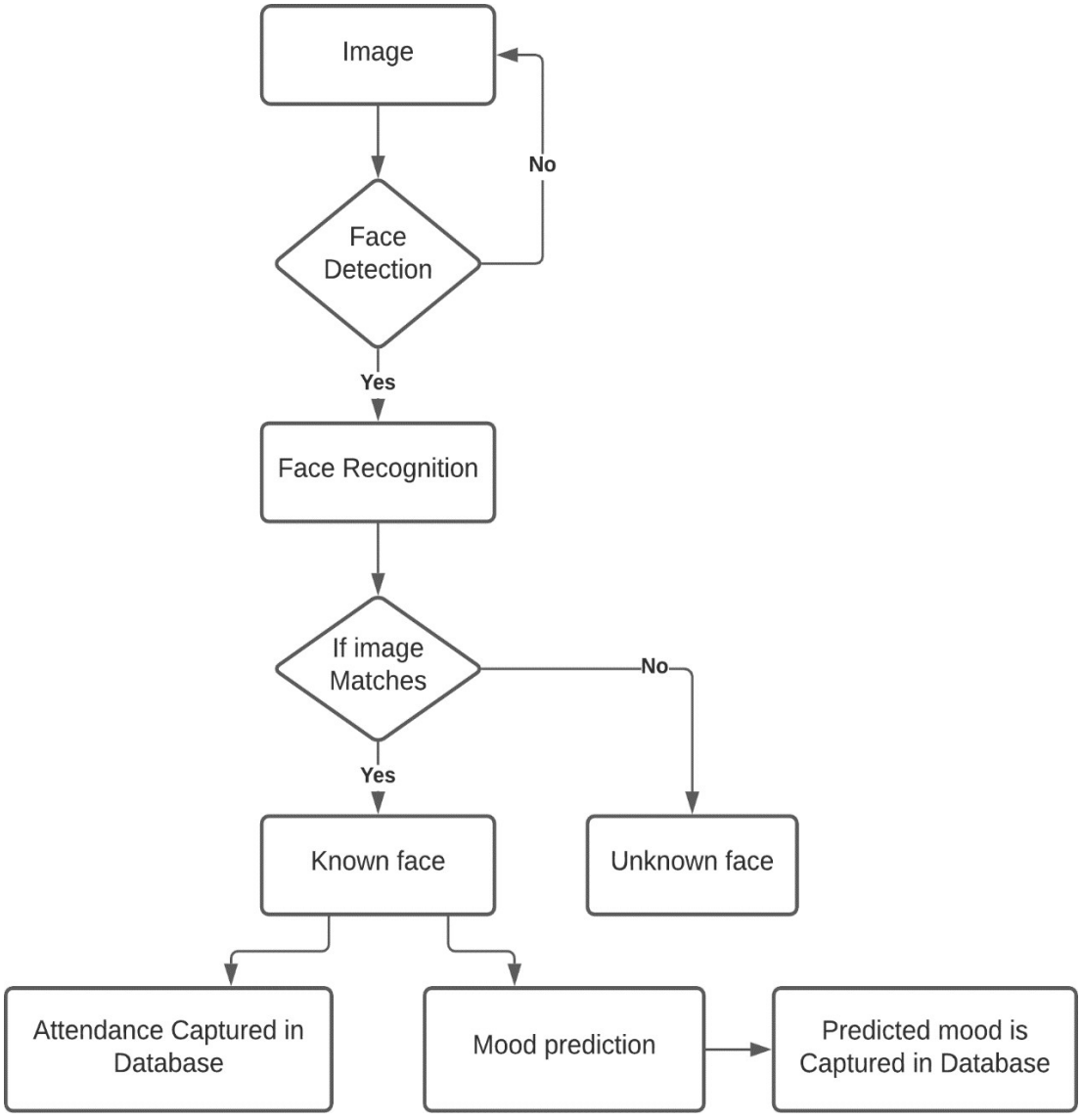


Fig 3.1 Block Diagram

The block diagram describes how the attendance is marked on the basics of face recognized for individual employees. So, there exist a camera module that is continuously recording the video and it is pre-processed to detect a face from the video. Once a face is detected the it is then feed to the facial recognition algorithm. And here the recognized face is matched with the database to identify the employee. If it's a match the attendance for the employee is capture in the database. As an add-on feature form the recognized facial data, we can predict their mood and add this as a remark for the employee.

3.2 : DFD

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.

3.2.1 : 0-LEVEL DFD

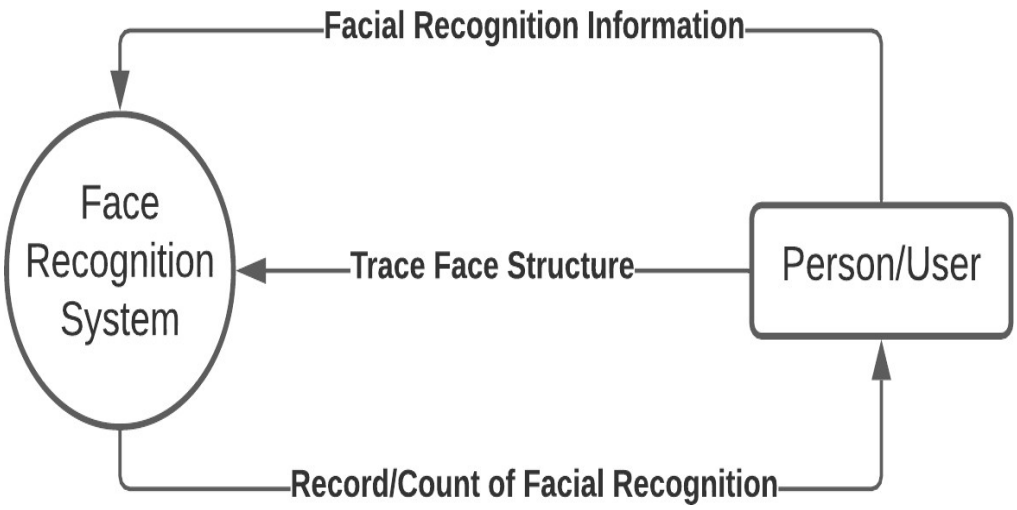


Fig 3.2 0-Level DFD

This is the simple view of our entire system processes. In figure 3.2 we can see that, the user's facial information is recorded and the facial structure is processed with face recognition system and a record or count of facial recognition is given back to the user.

3.2.2: 1-LEVEL DFD

Level 1 DFD provide a broad overview but go into greater depth than a context diagram. The following are essential data to accommodate:

- Face Recognition Records
- Visitors Records
- Employees/Staff Records

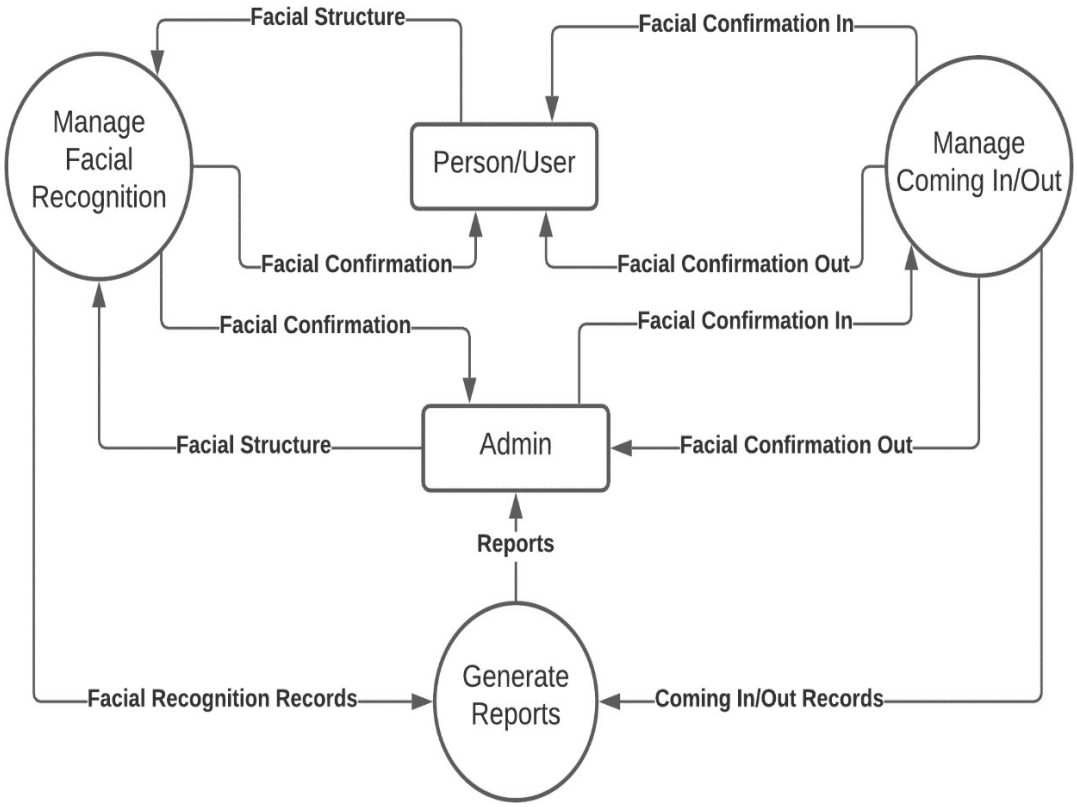


Fig 3.3 1-Level DFD

3.2.3: 2-LEVEL DFD

The Processes that the system should prioritize are as follows:

• Managing Facial Records

• Records of Unfamiliar Faces in Real-Time

• Records of Familiar Faces in Real-Time

• Secure Real-Time Records

• Count of Individuals that comes in and out of the Establishment

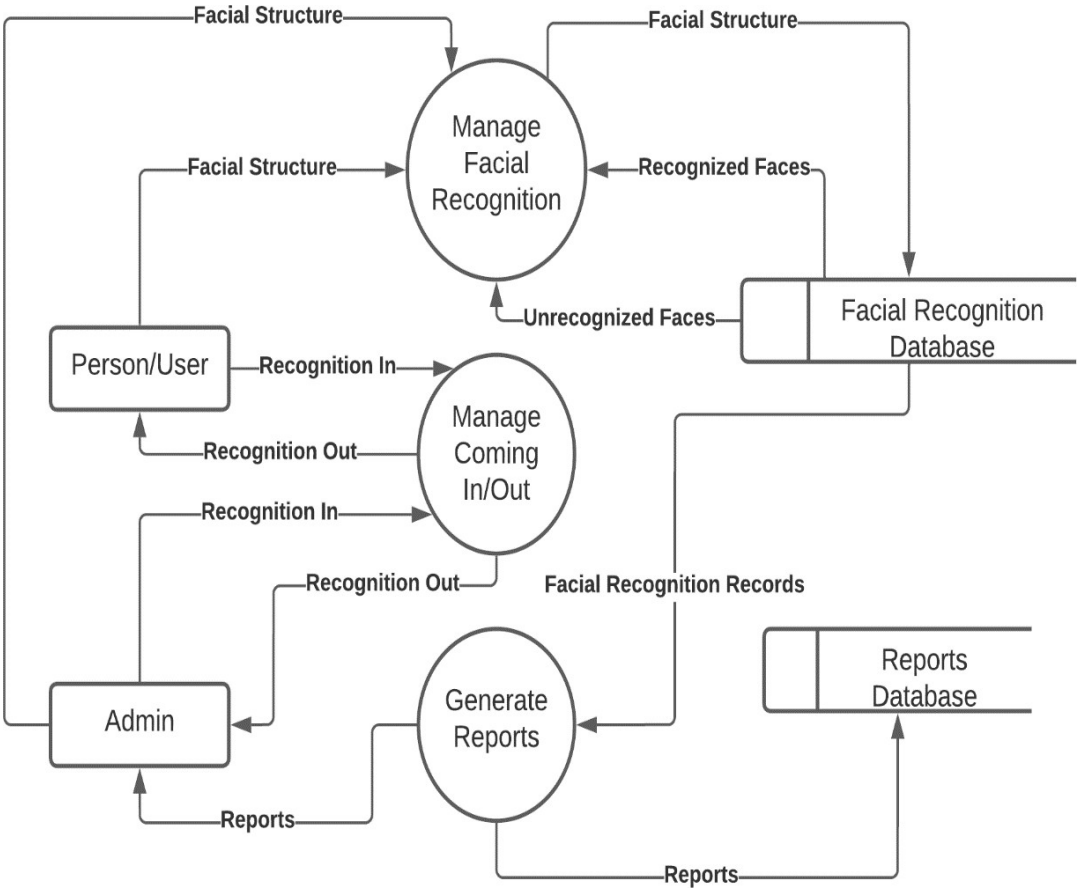


Fig 3.4 2-Level DFD

CHAPTER 4
IMPLIMENTATION DETAILS

5.1 : Platform – Anaconda

Anaconda is an open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. Anaconda is popular because it brings many of the tools used in data science and machine learning with just one install, so it's great for having short and simple setup. Like Virtualenv, Anaconda also uses the concept of creating environments so as to isolate different libraries and versions.

5.2 : IDE- JUPYTER

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more. Jupyter Notebook provides you with an easy-to-use, interactive data science environment across many programming languages that doesn't only work as an IDE, but also as a presentation or education tool. It's perfect for those who are just starting out with data science.

5.3 : Python

Python is an elevated level programming language for broadly useful programming. It was created by Guido Van Rossum and released in 1991. It enables clear programming on both small and large scales. Python bolsters various programming standards including object arranged, useful and procedural. Python is an easily readable language. It uses English keywords whereas other programming languages use punctuations. Python utilizes whitespace space as opposed to wavy sections to delimit squares. Python was mainly

developed to read codes easily. Python supports various libraries such as Pandas, NumPy, SciPy, Matplotlib etc. It supports various packages such as Xlsx Writer and Xl Rd. Python is an exceptionally helpful language for web improvement and programming advancement. It tends to be utilized to make web applications. It very well may be utilized to peruse and alter documents. It very well may be used to perform complex science. Python has gotten a very well-known language since it can chip away at various stages. Python code can be executed when it is composed. Python is a very significant language since the program is updated without investing additional exertion and energy. Python bolsters many working frameworks.

CHAPTER 5
DATABASE

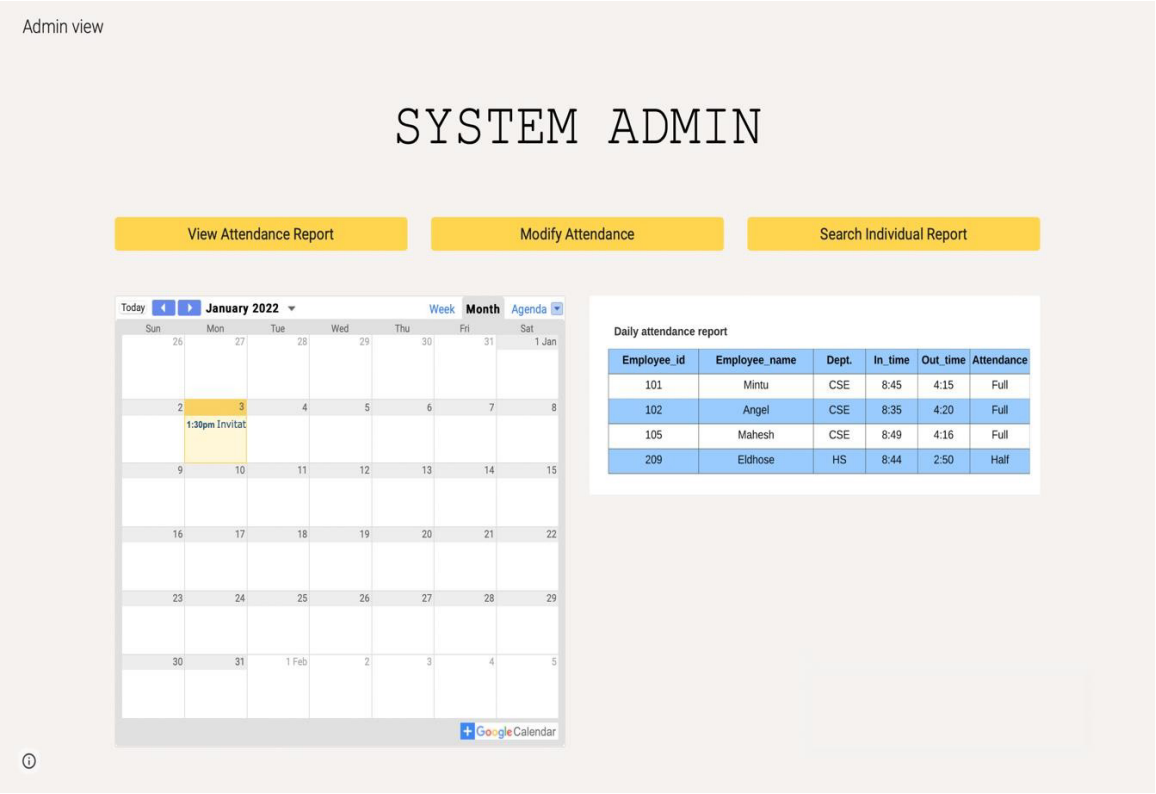
Daily attendance report

Employee_id	Employee_name	Dept.	In_time	Out_time	Attendance
101	Mintu	CSE	8:45	4:15	Full
102	Angel	CSE	8:35	4:20	Full
105	Mahesh	CSE	8:49	4:16	Full
209	Eldhose	HS	8:44	2:50	Half

Individual attendance report of employee_id 101

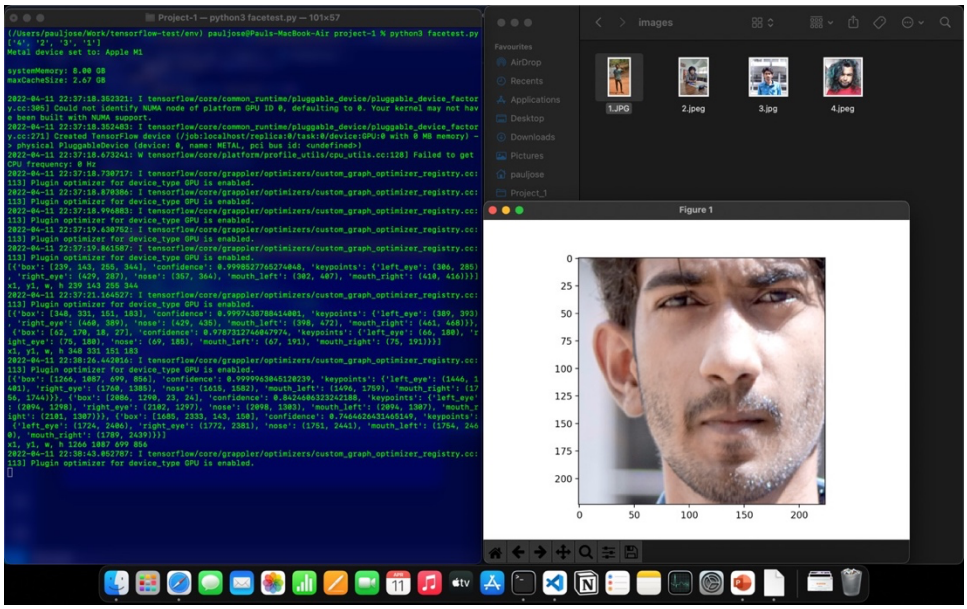
Employee_id	Employee_name	Dept	Date	In_time	Out_time	Attendance	Remark
101	Mintu	CSE	03/01/2022	8:45	4:15	Full	Nill
101	Mintu	CSE	04/01/2022	8:44	4:16	Full	Nill
101	Mintu	CSE	05/01/2022	8:46	4:17	Full	Nill
101	Mintu	CSE	06/01/2022	8:46	4:15	Full	Nill
101	Mintu	CSE	07/01/2022	8:45	12:45	Half	Nill

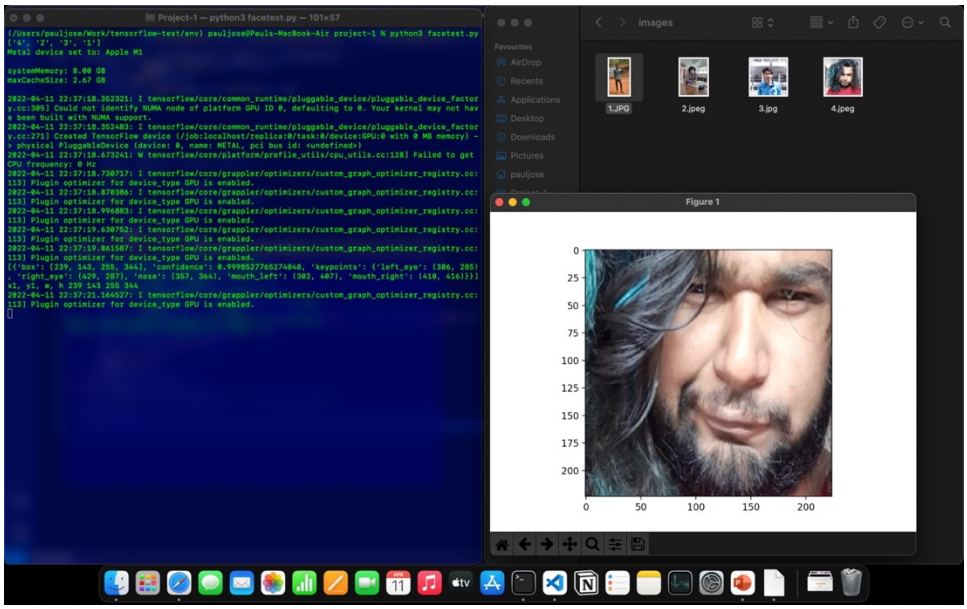
CHAPTER 6
PAGE DESIGN



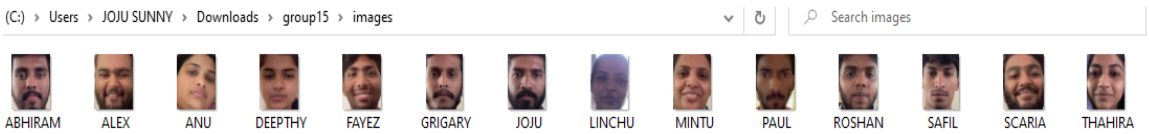
CHAPTER 7
SCREENSHOTS

Face Detection





Dataset



Face Recognition

CHAPTER 8

CONCLUSION

Face recognition systems are part of facial image processing applications and their significance as a research area are increasing recently. Implementations of system are crime prevention, video surveillance, person verification, and similar security activities. The goal is reached by face detection and recognition methods. Knowledge-Based face detection methods are used to find, locate and extract faces in acquired images. Implemented methods are skin colour and facial features. Neural network is used for face recognition. An automatic attendance management system aims at solving the issues of manual methods of existing systems. We have used the concept of face recognition to implement a system that marks the attendance of a particular person by detecting and recognizing the face. These systems perform satisfactorily with different facial expressions, lighting and pose of the person. There are future scopes to make a more compact ergonomics to make it a more user-friendly product to make an impact in building a healthier academic environment. Face detection is the key of biometric recognition. Face detection technology is used in the field of public security video monitoring. The application of face recognition technology in intelligent video surveillance system has broad application prospects and great practical significance. So, we combine these facial recognition methods to improve our design and the results can be of a higher accuracy.

CHAPTER 9

REFERENCE

[1] P. Glauner, J. A. Meira, P. Valtchev, R. State, and F. Bettinger, ‘‘The challenge of non-technical loss detection using artificial intelligence: A survey,’’ Int. J. Comput. Intell. Syst., vol. 10, no. 1, pp. 760–775, 2016.

[2] M. A. Tebbi and B. Haddad, ‘‘Artificial intelligence systems for rainy areas detection and convective cells’ delineation for the south sh ore of Mediterranean Sea during day and nighttime using MSG satellite images,’’ Atmos. Res., vols. 178–179, pp. 380–392, Sep. 2016.

[3] P. Dai, X. Wang, W. Zhang, P. Zhang, and W. You, ‘‘Implicit relative attribute enabled cross-modality hashing for face image-video retrieval,’’ Multimedia Tools Appl., vol. 77, no. 18, pp. 23547–23577, Sep. 2018.

[4] Q. W. Wang and Z. L. Ying, ‘‘A face detection algorithm based on Haarlike t features,’’ Pattern Recognit. Artif. Intell., vol. 28, no. 1, pp. 35– 41, 2015.

[5] M. Sepandi, M. Taghdir and A. Rezaianzadeh, "Assessing breast cancer risk with an artificial neural network", *Asian Pacific J. Cancer Prevention*, vol. 19, no. 4, pp. 1017-1019, 2018.

[6] Y.-H. Lai and C.-K. Yang, "Video object retrieval by trajectory and appearance", *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 6, pp. 1026-1037, Jun. 2015.

[7] A. K. G. Worner, "Realtime quality monitoring of compressed video signals", *SMPTE J.*, vol. 111, no. 9, pp. 373-377, Sep. 2002.

[8] D. Saravanan and S. Srinivasan, "Video data mining information retrieval using BIRCH clustering technique" in Advances in Intelligent Systems and Computing, New Delhi, India:Springer, vol. 325, pp. 583-594, 2015

APPENDIX

Face Registration

```
import cv2
import os
import datetime
from mtcnn.mtcnn import MTCNN
import numpy as np
from keras_vggface.utils import preprocess_input
from keras_vggface.vggface import VGGFace
from PIL import Image
from numpy import asarray
from numpy import expand_dims
from matplotlib import pyplot
print("Starting...")
def encodeFaceData():
    count = 0
    print("Encoding . . .")
    image = []
    classnames =[]
    encodeClassNames = []
    path = 'images'
    pathToEncode = 'Encodings'
    myList = os.listdir(path)
    myEncodeList = os.listdir(pathToEncode)
    print(myEncodeList)
    # loading images from the folder and appending into a list
    # also appending the image names into a list
    for entry in myEncodeList:
        encodeClassNames.append(os.path.splitext(entry)[0])
    for images in myList:
        if images.lower().endswith(('.png', '.jpg', '.jpeg')):
            if (os.path.splitext(images)[0]) not in encodeClassNames:
                pixels = pyplot.imread(f'{path}/{images}')
                image.append(pixels)
                classnames.append(os.path.splitext(images)[0])
        else :
            print("File Unknown " + os.path.splitext(images)[0] + "\n")
    detector = MTCNN()
    for i, cl in zip(image, classnames):
        # detecting faces from the image set
        faces = detector.detect_faces(i)
        if faces:
            x1, y1, w, h = faces[0]['box']
            # only one face is choosen from multiples faces
            x2, y2 = x1 + w, y1 + h
```

```
            faces = i[y1:y2, x1:x2]
            faces = Image.fromarray(faces)
            faces = faces.resize((224,224))
            # resizing the face to feed into the algorithm
            face_array = asarray(faces)
            pixels = face_array.astype('float32')
            samples = expand_dims(pixels, axis=0)
            samples = preprocess_input(samples, version=2)
            model = VGGFace(model='resnet50', include_top=False, input_shape=(224, 224,
3), pooling='avg')
            # predicting the encoding or feature extracting
            yhat = model.predict(samples)
            print(classnames[count])
            path = 'Encodings'
            with open(os.path.join(path, f'{classnames[count]}.txt'), 'xb') as file:
                # saving the encodings to a folder Encodings with image name as file name
                np.save(file, yhat)
            count = count + 1
        else:
            print(f'Face not detected for {cl} \n')
            continue
detector = MTCNN()
cap = cv2.VideoCapture(0)
i = 1
path = 'images'
# Starting a loop to captue the frames continuesly
while True:
    ret, frame = cap.read()
    if ret:
        cv2.imshow("Web cam", frame)
        if cv2.waitKey(1) & 0xFF == ord(' '):
            detector = MTCNN()
            # detecting faces from the currnet frame
            faces = detector.detect_faces(frame)
            if faces:
                x1, y1, w, h = faces[0]['box']
                # only one face is choosen from multiple faces detected
                x2, y2 = x1 + w, y1 + h
                croped = frame[y1:y2, x1:x2]
                cv2.imshow("Captured face", croped)
                if cv2.waitKey(0) & 0xFF == ord(' '):
                    key = input("Enter Y to continue press R to retake else press any other key: ")
                    if key == 'y' or key == 'Y':
                        # saving the image file into the folder images
                        name = input("Enter the name(in UPPERCASE): ")
                        cv2.imwrite(os.path.join(path, f'{name}.jpeg'),croped)
                        cv2.destroyAllWindows()
                        continue
```

```
                elif key == 'R' or key == 'r':
                    cv2.destroyAllWindows()
                    continue
                else:
                    encodeFaceData()
                    exit(0)
            else:
                print("Face missing")
                continue
    else:
        print("Cam missing . . .")
```

Face Encoding

```
import os
import numpy as np
from keras_vggface.utils import preprocess_input
from keras_vggface.vggface import VGGFace
from mtcnn.mtcnn import MTCNN
from PIL import Image
from numpy import asarray
from numpy import expand_dims
from matplotlib import pyplot
def encodeFaceData():
    count = 0
    print("Encoding . . .")
    image = []
    classnames =[]
    encodeClassNames = []
    path = 'images'
    pathToEncode = 'Encodings'
    myList = os.listdir(path)
    myEncodeList = os.listdir(pathToEncode)
    print(myEncodeList)
    # loading images from the folder and appending into a list
    # also appending the image names into a list
    for entry in myEncodeList:
        encodeClassNames.append(os.path.splitext(entry)[0])
    for images in myList:
        if images.lower().endswith(('.png', '.jpg', '.jpeg')):
            if (os.path.splitext(images)[0]) not in encodeClassNames:
                pixels = pyplot.imread(f'{path}/{images}')
                image.append(pixels)
                classnames.append(os.path.splitext(images)[0])
        else :
            print("File Unknown " + os.path.splitext(images)[0] + "\n")

    detector = MTCNN()
    for i, cl in zip(image, classnames):
```

```
        # detecting faces from the image set
        faces = detector.detect_faces(i)
        if faces:
            x1, y1, w, h = faces[0]['box']
            # only one face is choosen from multiples faces
            x2, y2 = x1 + w, y1 + h
            faces = i[y1:y2, x1:x2]
            faces = Image.fromarray(faces)
            faces = faces.resize((224,224))
            # resizing the face to feed into the algorithm
            face_array = asarray(faces)
            pixels = face_array.astype('float32')
            samples = expand_dims(pixels, axis=0)
            samples = preprocess_input(samples, version=2)
            model = VGGFace(model='resnet50', include_top=False, input_shape=(224, 224,
 3), pooling='avg')
            # predicting the encoding or feature extracting
            yhat = model.predict(samples)
            print(classnames[count])
            path = 'Encodings'
            with open(os.path.join(path, f'{classnames[count]}.txt'), 'xb') as file:
                # saving the encodings to a folder Encodings with image name as file name
                np.save(file, yhat)
            count = count + 1
        else:
            print(f'Face not detected for {cl} \n')
            continue
 encodeFaceData()
```

Facial Attendance

```
import os
import os.path
import numpy as np
from PIL import Image
import PIL
from mtcnn.mtcnn import MTCNN
from matplotlib import pyplot
from numpy import asarray
from keras_vggface.utils import preprocess_input
from keras_vggface.vggface import VGGFace
from scipy.spatial.distance import cosine
import cv2
from numpy import expand_dims
from datetime import date, datetime

def readCam():
    print("Reading")
    camencoding = []
```

```
    detector = MTCNN()
    cap = cv2.VideoCapture(0)
    i = 1
    # starting the camera inside a loop to continuesly capture the frames and it will the
working inrealtime
    while True:
        ret, frame = cap.read()
        faces = detector.detect_faces(frame)
        # detect a face when someone passes throught the camera's view
        if faces:
            x1, y1, w, h = faces[0]['box']
            x2, y2 = x1+w, y1+h
            faces = cv2.rectangle(frame, (x1, y1),(x2, y2), (255,255,255), 2)
            croped = frame[y1:y2,x1:x2]
            cv2.imshow("faces", faces)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
            faceEncoding = Image.fromarray(croped)
            faceEncoding = faceEncoding.resize((224,224))
            face_array = asarray(faceEncoding)
            pixels = face_array.astype('float32')
            samples = expand_dims(pixels, axis=0)
            samples = preprocess_input(samples, version=2)
            model = VGGFace(model='resnet50', include_top=False, input_shape=(224, 224, 3),
pooling='avg')
            yhat = model.predict(samples)
            camencoding = yhat
            compare(camencoding)
        else:
            # print("Empty")
            continue

def compare(x):
    # print("Comparing")
    for i, name in zip(loaded_arr, classnames):
        score = cosine(x, i)
        # from the encoding values campare the values
        #  set a threshold value of .3
        if score < .3:
            markAttendance(name)
def markAttendance(name):
    path = f'{date.today()}.csv'
    if not (os.path.isfile(path)):
        f = open(path,'w')
        print("File Created")
        f.close()
    nameList = []
    timeInList = []
```

```
    timeOutList = []
    presenceList = []
    with open(path,'r') as filer:
        myDataList = filer.readlines()
        for line in myDataList:
            entry = line.split(',')
            nameList.append(entry[0])
            timeInList.append(entry[1])
            timeOutList.append(entry[2])
            presenceList.append(entry[3])
    if name not in nameList:
        with open(path,'a') as filea:
            now = datetime.now()
            dtString = now.strftime('%H:%M:%S')
            filea.writelines(f'{name},{dtString},0,NA,0\n')
    else:
        index = nameList.index(name)
        timeIn = timeInList[index]
        timeIn = timeIn.strip()
        timeIn = datetime.strptime(timeIn,'%H:%M:%S')
        now = datetime.now()
        timeOut = now.strftime('%H:%M:%S')
        now = datetime.strptime(timeOut,'%H:%M:%S')
        inTimeDifference = str(now - timeIn)
        inTimeDifference = inTimeDifference.split(':')
        minutes = int(inTimeDifference[1])
        hours = int(inTimeDifference[0])
        if minutes >= 1:#minutes >= 5
            with open(path,'w') as filew:
                for line in myDataList:
                    entry = line.split(',')
                    if entry[0] == name:
                        if minutes >= 3:#hours >=7
                            print("Reached")
                            filew.writelines(f'{name},{timeInList[index]},{timeOut},FULL,{hours}\n')
                        elif minutes >= 2:#hours >= 4

filew.writelines(f'{name},{timeInList[index]},{timeOut},HALF,{hours}\n')
                        else:
                            filew.writelines(f'{name},{timeInList[index]},{timeOut},NA,{hours}\n')
                    else:
                            filew.writelines(line)
                            print("Writting", line)
def retriveEncode():
    path = 'Encodings'
    global loaded_arr
    loaded_arr = []
    global classnames
```

```
    classnames = []
    # loading the encodings and class names from the folder
    myList = os.listdir(path)
    for item in myList:
        if item.lower().endswith('txt'):
            loaded_arr.append(np.load(f'{path}/{item}'))
            classnames.append(os.path.splitext(item)[0])
retriveEncode()
readCam()
```