



Attendance Monitoring System using CNN

Chennu Vaishnavi Priya , Masrath Jahan and Sri Momitha Chiluveru

Department of Computer Science and Engineering, Stanley College of Engineering and Technology for Women,
Telangana, India

ABSTRACT

Uniqueness or individuality of an individual is his/her face. Attendance check plays an important role in classroom management. Checking attendance by calling names or passing around a sign- in sheet is time-consuming, inefficient and especially the latter is open to easy fraud. The main aim of this project is to improvise the traditional systems and introduce a new approach using Deep Learning techniques that presents the detailed implementation of a real-time attendance monitoring system. We first capture an image of students in a classroom and utilize the OpenCV module to detect and frame the faces in that image. In the next stage we enhance these frames using an image enhancing model. In the final stage of the project, we build a Convolution Neural Network (CNN) to train these facial images and compare them with the student records that are stored in the database and hence update the attendance status of the students. Our system promises easy to maintain hassle free attendance system and with other integrations it can be used for other necessities of industries and is not limited to educational institutions.

Keywords: CNN, Deep Learning, Open CV



TABLE OF CONTENTS

| | |
|----------------------------------------|--------------|
| 1. Introduction | 01-04 |
| 1.1 About Project | 01 |
| 1.2 Objectives of the Project | 01 |
| 1.3 Scope of the Project | 02 |
| 1.4 Advantages | 02 |
| 1.5 Disadvantages | 03 |
| 1.6 Applications | 03 |
| 1.7 Hardware and Software Requirements | 04 |
| 2. Literature Survey | 05-07 |
| 2.1 Existing System | 05 |
| 2.2 Proposed System | 05 |
| 3. Proposed Architecture | 08-17 |
| 4. Implementation | 18-42 |
| 4.1 Algorithm | 18 |
| 4.2 Code Implementation | 19 |

| | |
|------------------------|--------------|
| 5. Results | 43-46 |
| 6. Conclusion | 47 |
| 7. Future Scope | 48 |
| 8. References | 49 |

LIST OF FIGURES

1. Fig 3.3.a: Haar like Feature
2. Fig 3.3.b: Haar Cascade Flowchart
3. Fig 3.4.a: CNN Architecture
4. Fig 3.4.b: Xception Model Architecture
5. Fig 3.5.a: Data Flow Diagram
6. Fig 3.5.1.a: Data Flow Diagram Level 1
7. Fig 3.5.1.b: Data Flow Diagram Level 2
8. Fig 3.5.1.c: Data Flow Diagram Level 3
9. Fig 3.5.2.a: Usecase Diagram
10. Fig 3.5.3.a: Activity Diagram
11. Fig 3.5.4.a: Component Diagram
12. Fig 4.1.a: Implementation Steps
13. Fig 4.1.b: Algorithm Flow
14. Fig 5.a: Accuracy vs Epochs Graph
15. Fig 5.b: Student prediction using Image address
16. Fig 5.c: Student prediction using captured image from webcam
17. Fig 5.d: Attendance Marking using Face Recognition from live webcam
Fig 5.e: Output Excel Sheet



International Journal for Innovative Engineering and Management Research

A Peer Reviewed Open Access International Journal

www.ijiemr.org

CHAPTER 1 INTRODUCTION

1.1 ABOUT THE PROJECT

Convolutional Neural Networks has been playing a significant role in many applications including surveillance, object detection, object tracking, etc. Extensive research is recorded for face recognition using CNNs, which is a key aspect of surveillance applications. In most recent times, the Face Recognition technique is widely used in University automation systems, Smart Entry management systems, etc. In this project, a CNN architecture for face recognition system is proposed including the process of collecting face data of students. Experimentally it is shown that the proposed CNN architecture provides high accuracy. Further, the proposed CNN framework is used to develop a “Attendance Monitoring System Using CNN”, which is used to provide attendance of students using face recognition, in real time. The proposed application is easy to deploy and maintain.

Image recognition is playing an important role in modern living like driver assistance systems, medical imaging system, quality control system to name a few. An Artificial Neural Network along with image recognition used to enhance the reliability of the system. One such update used here is CNN. Deep learning is an emerging technology hence opted to implement the smart attendance system in it. The implementation basically consists of three components: 1) Face scanning and detection using HAAR cascade method 2) Training the CNN-ANN model 3) Recognize the face back and update the attendance. The main motivation of our work is to merge three of the emerging technologies: Machine learning, Image Processing and IOT. Key advantage of this implementation is that a deep learning model increases its accuracy with more epochs of training, and it optimizes the run time.

1.2 OBJECTIVES

The objective of face recognition is, from the incoming image, to find a series of data of the same face in a set of training images in a database. The great difficulty is ensuring that this process is carried out in real-time, something that is not available to all biometric facial recognition software providers.

The facial recognition process can perform two variants depending on when it is performed:

The one in which, for the first time, a facial recognition system addresses a face to register it and associate it with an identity, in such a way that it is recorded in the system. This process is also known as digital onboarding with facial recognition.

The variant in which the user is authenticated, prior to being registered. In this process, the incoming data from the camera is crossed with the existing data in the database. If the face matches an already registered identity, the user is granted access to the system with his credentials.

1.3 SCOPE

Problem Statement: Manually taking attendance and registering it in files and musters makes the daily attendance a mundane task for the faculty and unnecessarily consumes classroom time.

Problem Solution: SCOPE:

The scope of this project is wide as it can work on the system in which the software is installed, i.e. the project can be developed as a laptop or desktop application as well as we can create the project using Google Colab making it portable and platform-independent, and thus will work accordingly for a particular school or college. The objective of the attendance software is to reduce the time that is consumed when attendance is taken manually. Unlike the manual process, an online system easily helps management to analyze student's attendance details as per requirements and avoids errors inherent in the manual work and hence makes the output consistent and correct.

1.4 ADVANTAGES

- i. For saving the time in the classroom and accuracy in attendance will be maintained.
- ii. Second is availability of advanced technology
- iii. Less manpower required
- iv. It is more useful for the future generation
- v. Saves time and efforts.
- vi. Used for security purpose.
- vii. Multiple face detection.
- viii. Multiple face recognition.
- ix. Unknown faces are identified.
- x. As the system stores the faces that are detected during registration and automatically marks the attendance faster. Providing authorized access.

1.5 DISADVANTAGES

- i. Creates a CSV file every day, so this can take more storage however we can integrate all the files into one at the end of a month or so using MS excel tools.
- ii. Expensive as compared paper based/manual.
- iii. Difficulties with big data processing and storing without GPU and RAM below 4GB
- iv. Deluded by identical twins
- v. Weak camera angle, low lightning and image quality.

1.6 APPLICATIONS

Applications of face recognition using CNN are:

- i. Track school attendance.
- ii. Used to eliminate duplicates in a nationwide voter registration system.
- iii. Image database investigations:
- iv. Searching image databases of licensed drivers, benefit recipients, missing children, immigrants and police bookings.
- v. General identity verification: Electoral registration, banking, electronic commerce, identifying newborns, national IDs, passports, employee IDs.
- vi. Unlock mobile phones

1.7 HARDWARE AND SOFTWARE REQUIREMENTS

1.7.1 Software Requirements

Operating System : Windows 7/8/8.1/10 64 bit

Programming Language : Python

Back-end : Java Script

Text Editor : Google Collaboratory

1.7.2 Hardware Requirements

Processor : Intel i3 or more

Motherboard : Intel Chipset Motherboard

Ram : 8 GB

Cache : 512 KB

Hard Disk : 16 GB Hard Disk recommended

Disk Drive : 1.44MB Floppy Disk Drive

Monitor : 1024 x 720 Display

Speed : 2.7 GHZ

CHAPTER 2

LITERATURE SURVEY

2.1 STUDY OF EXISTING SYSTEM

Most of the attendance systems use paper based methods for taking and calculating attendance and this manual method requires paper sheets and a lot of stationery material. Previously a very few work has been done relating to the academic attendance monitoring problem. Some software's have been designed previously to keep track of attendance. But they require manual entry of data by the staff workers. So the problem remains unsolved. Furthermore idea of attendance tracking systems using facial recognition techniques have also been proposed but it requires expensive apparatus still not getting the required accuracy.

2.1.1 Disadvantages of present working system

1. Not User Friendly: The existing system is not user friendly because the retrieval of data is very slow and data is not maintained efficiently.
2. Difficulty in report generating: We require more calculations to generate the report so it is generated at the end of the session. And the student does not get a single chance to improve his/her attendance.
3. Manual control: All calculations to generate report is done manually so there is greater chance of errors.
4. Lots of paperwork: Existing system requires lot of paper work. Loss of even a single register/record can lead to difficult situations because all the papers are needed to generate the reports.
5. Time Consuming: Every work is done manually so we cannot generate report in the middle of the session or as per the requirement because it is very time consuming.

2.2 PROPOSED SYSTEM

To overcome the problems in existing attendance system we shall develop a Face recognition based attendance system over simple attendance system, Interactive system over static one and Digitized attendance system over file system. There are many solutions to automate the attendance management system like thumb-based system, simple computerized attendance system but all these systems have

limitations over work and security point of view. Our proposed system shall be "Attendance Monitoring System" which uses the basic idea of image processing which is used in many secure applications like banks, airports, universities, schools or colleges.

2.2.1 Proposed System Components

Following are the main components of the proposed system

1. Student Registration

2. Face Detection

3. Face Recognition

- Feature Extraction

- Feature Classification

4. Attendance management system

It will allow uploading, updating and deletion of the contents of the system. Attendance management will handle:

->Automated Attendance marking

->Manual Attendance marking

->Attendance details of users

2.2.2 Features of the Proposed System

1. Easy to use with interactive GUI support.

2. Provides high accuracy.

3. Creates/Updates CSV file for details of students on registration.

4. Creates a new CSV file everyday for attendance and marks attendance with proper date and time.

5. Displays live attendance updates for the day on the main screen in tabular format with Id, name, date and time.

2.2.3 Goals of the Proposed System

The following goals were kept in mind while designing the new system:

- a. To reduce the manual works required to be done in the existing system.

- b. To improve the management of permanent information of the company by keeping it is properly structured tables and to provide facilities to update this information as efficient as possible. Attendance Management System. Project Report On "Attendance Management System" International School of

Information and Management.

c. To make the system complete menu-driven and hence user-friendly. This is necessary so that even non-programmers could use the system effectively and system could act as catalyst in achieving objective.

d. To make the systems compatibility i.e. it should “fit in” in the total, integrated system.

e. To design the system in such a way that reduces feature maintenance and enhancement times and efforts. Student Attendance Management System Project Topics or another purpose for developing this software is to generate the desired reports automatically at the end

f. To make the system reliable, understandable and cost effective.

CHAPTER 3

PROPOSED ARCHITECTURE

Biometric Recognition is the statistical data analysis of people's unique behavioral and physical characteristics which is mainly used for security and identification which includes fingerprints, facial features, retina, iris, voice, gaits palm print etc. Among these methods face detection is considered to be most precise and safe. Facial_recognition is an activity of discovering a peep's face by estimating and evaluating motifs on the exclusive facial markers of the face. Biometric software is used for this purpose.

There are number of strategies for recognizing person's face. Some of them are adaptive regional blend matching method and generalized matching face detection method. The values of the nodal points on the person's face plays crucial task in face recognition system.

Many researches had been done on LBP and Haar cascading techniques. But either they are using only one algorithm or they are detecting a single face in the image. In the current work two algorithms are used to detect the faces in the image containing many faces to calculate the accuracy then the acquired accuracy will be compared by plotting the curve and bar graph to find the efficient algorithm.

There are two types of image positive image and negative image. Positive images are those images which contain the face in that and negative images are the images which contains non-face image. Classifier is a device which decides whether the taken image is negative or positive. It is trained on hundreds of thousands of face and non-face images to learn to classify a new image as face or non-face image correctly. OpenCV provides two pre-trained classifiers Haar Classifier and LBP Classifier. Both of these classifiers process images in gray-scales as it doesn't need color information to decide if image has a face or not.

The architectural design and the system modules of the proposed system are discussed here:

3.1 FACE DETECTION

In the field of technology Face detection is treated as the demanding and practically applied approach. The identification of each face present in an image is the major task of the face detection. Here the implementation is done using OpenCV.

1. Loading the input images.
2. Converting the input images into gray scale images.
3. Applying the Haar cascade.
 - a) Importing the required libraries.
 - b) Taking the images which are captured by the camera.
 - c) To process the image through the classifiers it is converted into gray scale image.
 - d) Image will be loaded using OpenCV.
 - e) By default, image will be loaded into BGR color space.

3.2 FACE IDENTIFICATION

For face identification we need to train the model using CNN. Training consists of three phases. They are:

1. Image Resizing: We first resize the image to a fixed resolution. As the images received from the enhancer might be of different resolution and aspect ratio, we use the PIL library in python. We loop through the images in a directory and then resize them into an aspect ratio of 1:1. The resolution can be for 64*64, 128*128, 512*512 pixels. This is dependent on the developer. As the resolution increases, so does the density of the neural network. As the neural network keeps getting denser, the more processing power and time it will need.

2. Image augmentation and flooding to create the dataset: The next step is to augment the images and flood them. We use the ImageDataGenerator preprocessing module from the keras library. This module augments the input image into any number of images we want. For the sake of this project, we augmented all the images of a single person to a total of 70 images. So, each class(person) in the dataset consists of 70 images.

3. Model training and testing: The third and final step is to train the model with Neural Networks. We use the Tensorflow Keras Library in python as they contain almost all the algorithms and required functions to build the model. We again use the ImageDataGenerator to preprocess the image. This is a very basic preprocessing used to store all the images into 2 variables as 2 different batches. We then loop through the batches and separate the training and the test dataset into a python list. These lists are further converted into numpy arrays for the Neural Network model which we build later. After the preprocessing of the images is done and finally converted into a numpy array, they are ready to be trained.

3.3 HAAR CASCADE CLASSIFIER

Haar Cascading is the Machine Learning method where a classifier is drilled from a great deal of positive and negative photos. The algorithm is put forwarded by Paul Viola and Michael Jones. Haar feature-based cascade classifiers are the classifiers implemented for object detection. This classifier chases machine learning procedure in which a cascade operation is inculcated from the photos to discover items in additional photos. Face detection and facial expressions in an image are also successfully detected. The exercise is finished by offering positive and negative pictures to the classifier. Then the characteristics are drawn out from the picture. Each characteristic is an individual value, which is acquired by subtracting sum of pixels in white rectangle from summation of pixels in black rectangle. In which it detects the faces of different individual in different environments. The Haar-like feature of any size can be calculated in constant time because of integral images.

- 1) Loading the input image using built in function `cv2.imread(img_path)`, here the passing the image path as an input parameter.
- 2) Converting it to gray scale mode and then displaying it.
- 3) Loading the haar cascade classifier.

Fig. 3.3.1 represents the Haar like feature. It consists of edge feature and line feature. In the gray-scale image the white bar represents the pixels that are closer to the light source.

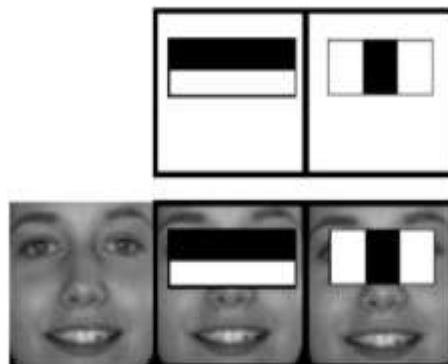


Fig. 3.3.a. Haar like feature.

$\text{Pixelvalue} = (\text{SumoftheDarkpixels} / \text{NumberofDarkpixels}) - (\text{SumoftheLightpixels} / \text{NumberofLightpixels})$

Haar Classifier is an object detection algorithm. In order to detect the object and to identify what it is; the features will be extracted from the image. Using Equation (1) haar pixel value can be calculated.

Fig. 3.3.2 represents the flowchart of the Haar cascade classifier. Once the camera acquires the image it converts the image into gray-scale. The cascade classifier detects the face, if the face is detected then the classifier once again checks for the both eyes in the detected face and if two eyes are detected it normalizes the face images size and orientation. Then the image is processed for face recognition where the image is compared with the face sample collection.

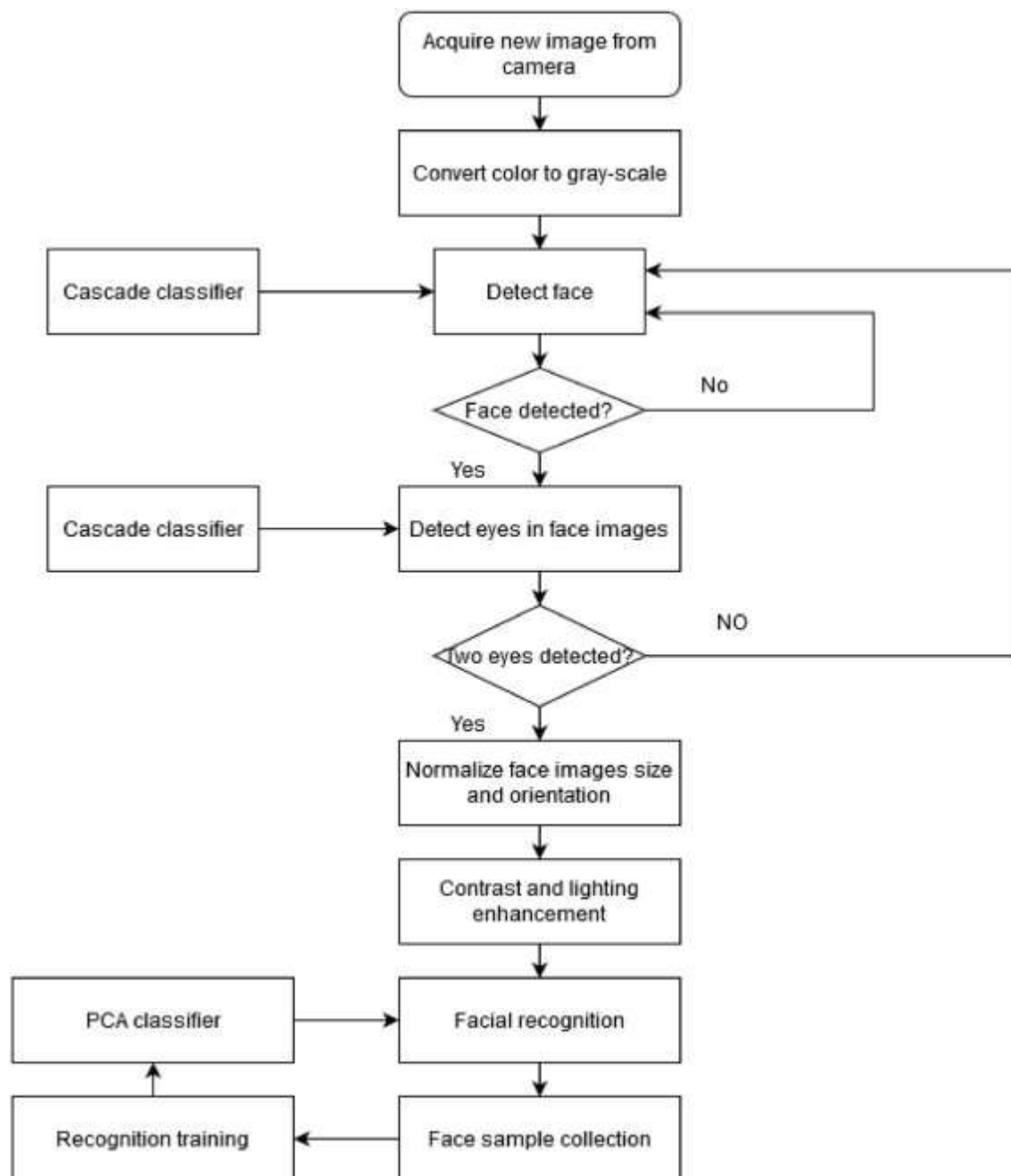


Fig. 3.3.b. Haar Cascade flowchart.

Importance of Haar Cascade classifier is that the Perception precision is more and the Positive rate is less.

3.4 CNN ALGORITHM NETWORK

In that MRI images is process and analyse and classify So using CNN algorithm.

Convolution Neural:

In machine learning, a convolutional neural network is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. CNN mimics the way humans see images, by focusing on one portion of the image at a time and scanning the whole image. CNN boils down every image as a vector of numbers, which can be learned by the fully connected Dense layers of ANN. Convolution neural network algorithm is a multilayer perceptron that is the special design for identification of two-dimensional image information and always has more layers such as input layer, convolution layer, sample layer and output layer. CNNs use comparatively small pre-processing is compared to other image classification algorithms. it means that the network learns the filters that in algorithms were hand-engineered.

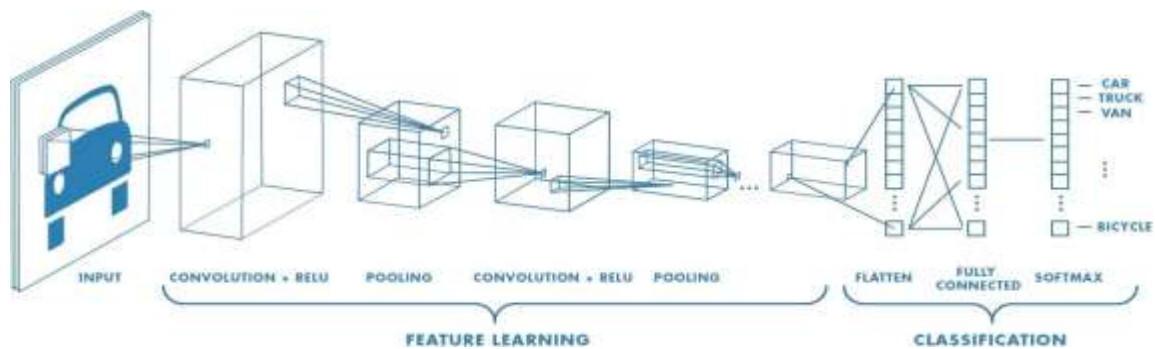


Fig. 3.4.a CNN Architecture

Matplotlib:

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications.

Xception Model:

Xception is a convolutional neural network that is 71 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database [1]. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 299-by-299.

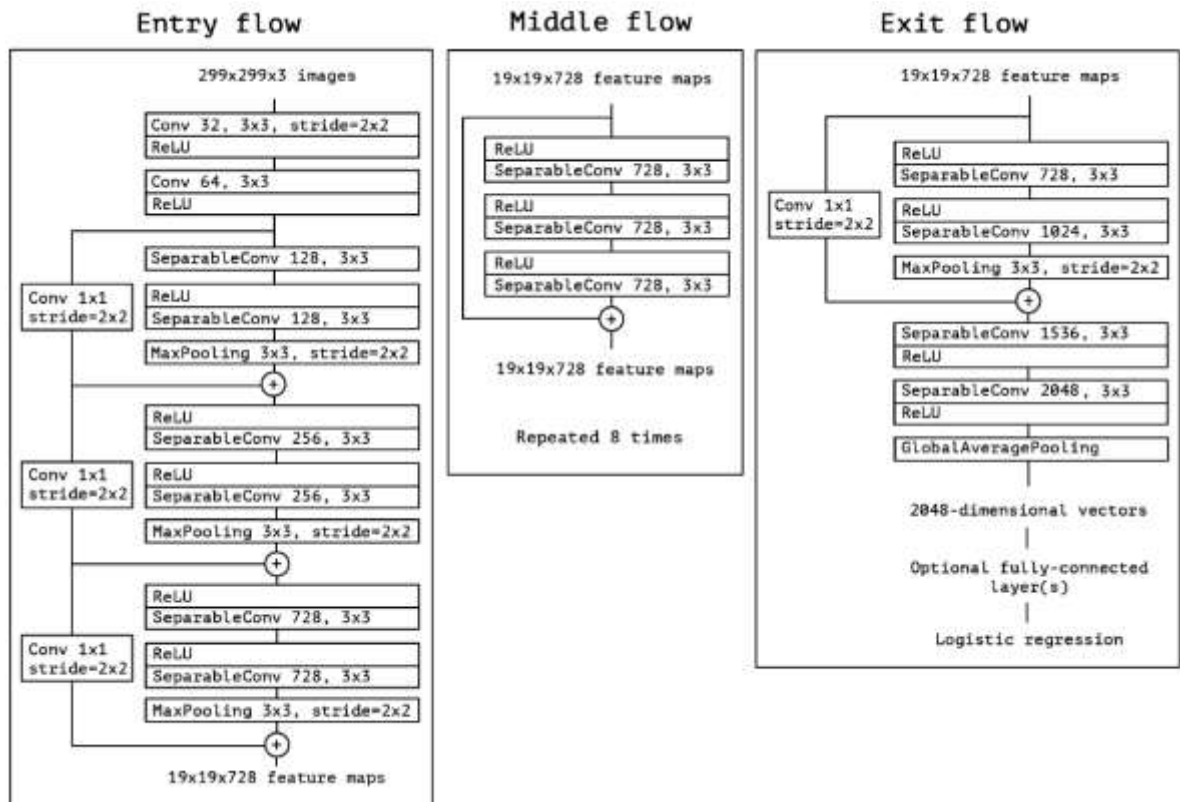


Fig. 3.4.b Xception Model Architecture

OpenCV:

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. In this system open cv library is used for image processing for the purpose of online attendance face recognition images.

3.5 SOFTWARE DESIGN

3.5.1 Data flow diagram

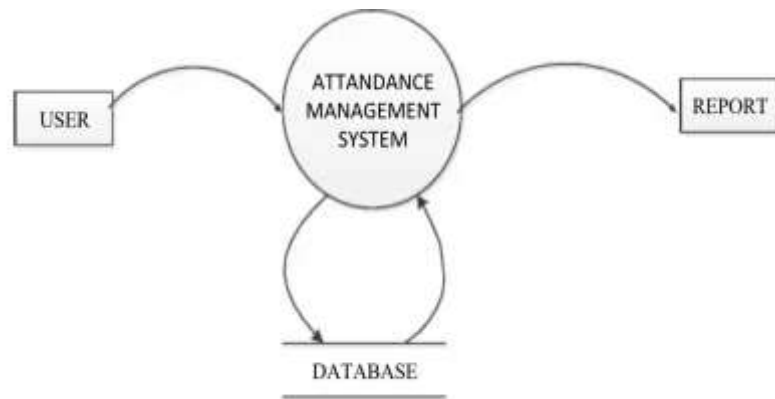


Fig 3.5.1.a Data flow diagram Level1

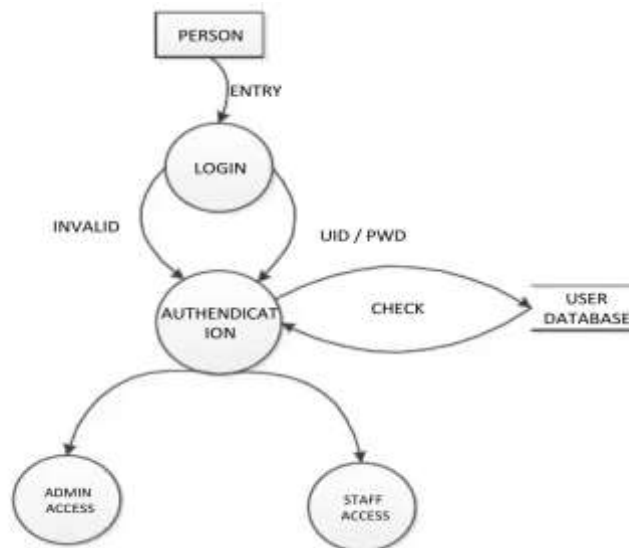


Fig 3.5.1.b Data flow diagram Level2

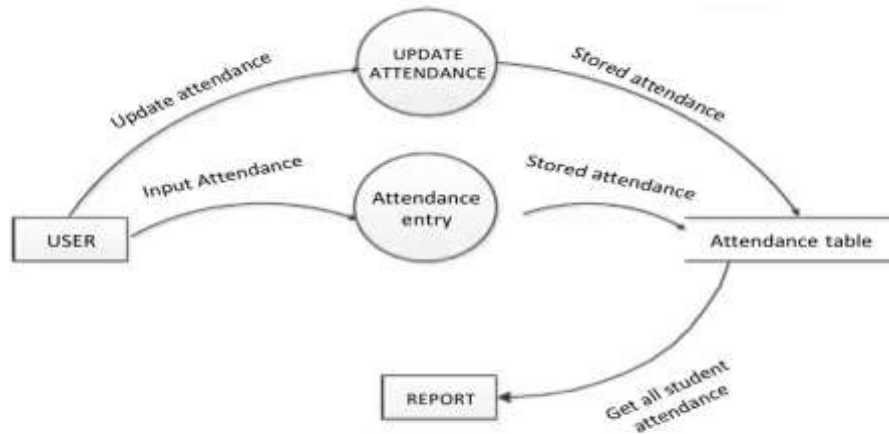


Fig 3.5.1.c Data flow diagram Level3

3.5.2 Usecase Diagram

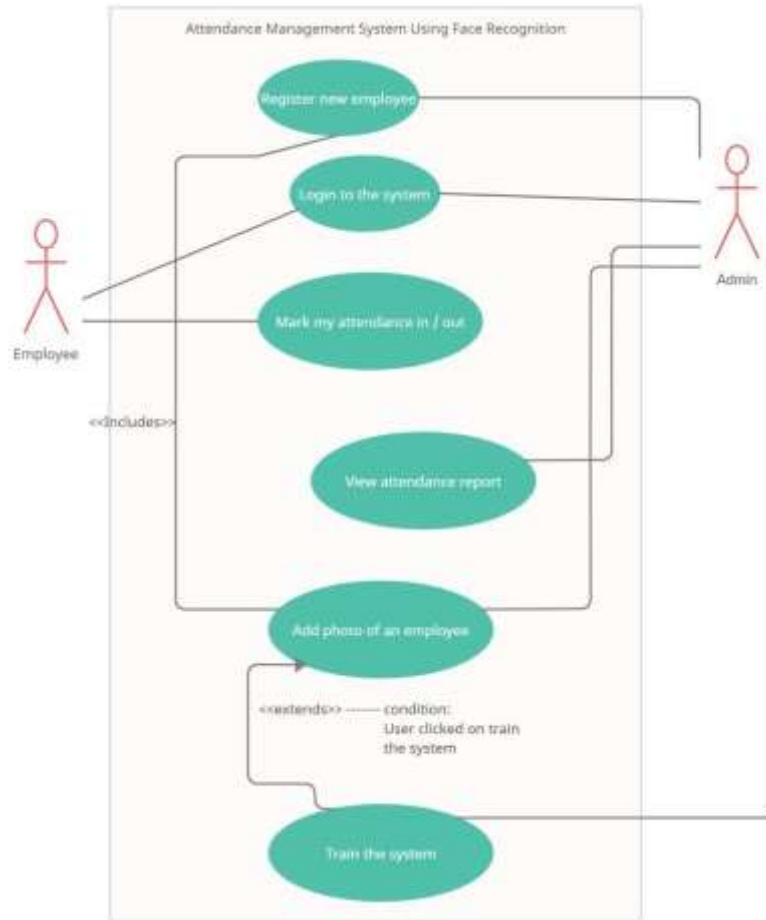


Fig 3.5.2.a Usecase Diagram

3.5.3 Activity Diagram

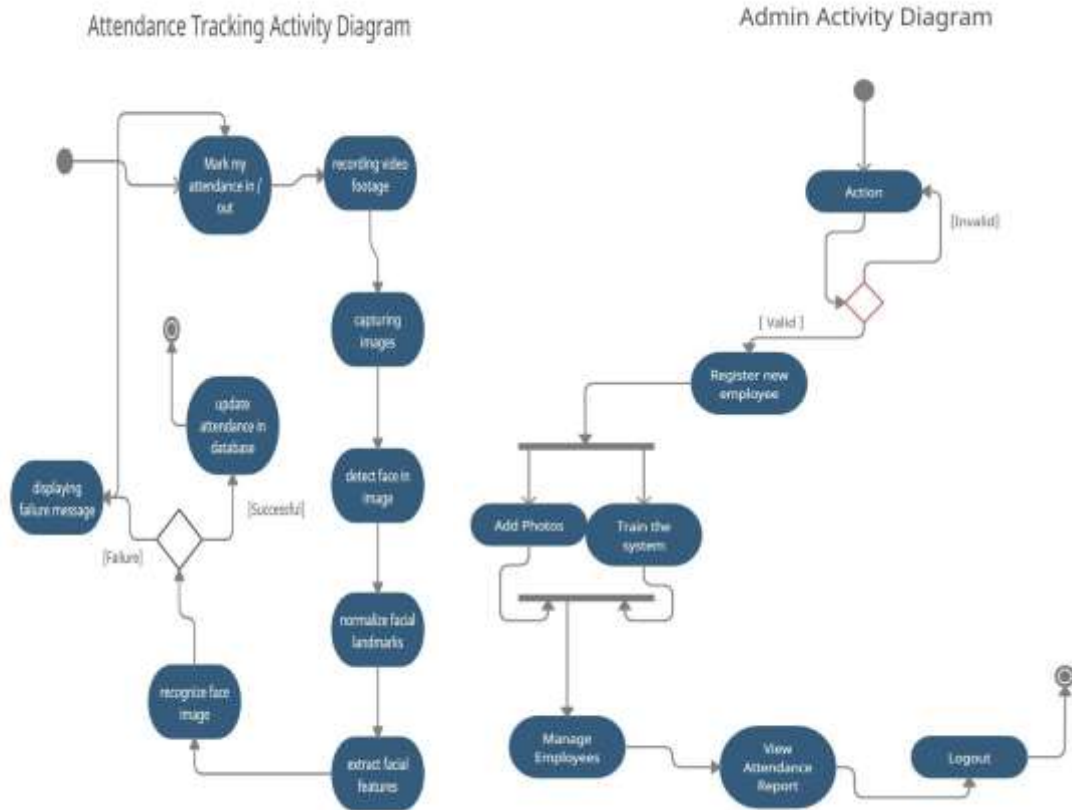


Fig 3.5.3.a Activity Diagram

3.5.4 Component Diagram

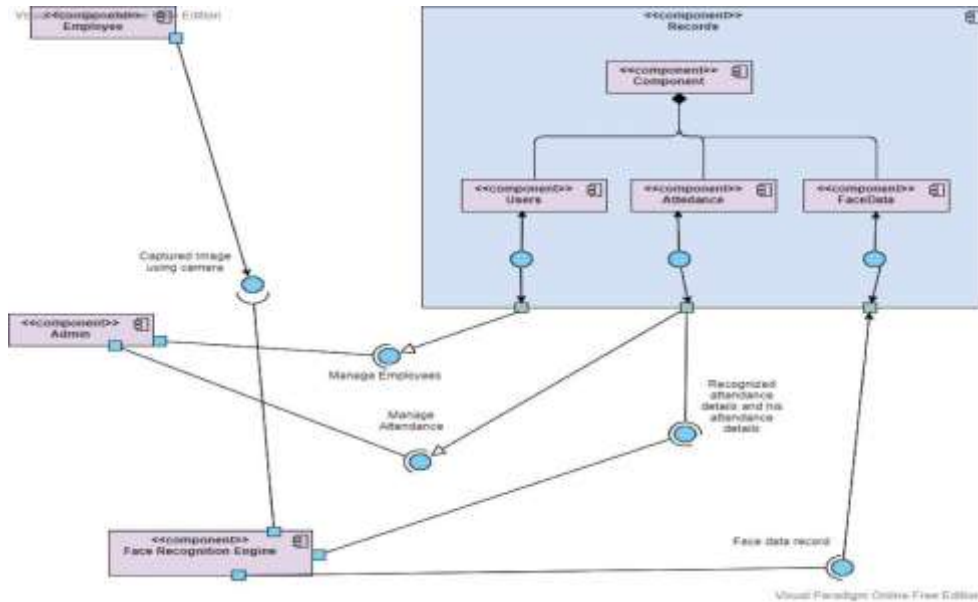


Fig 3.5.4.a Component Diagram

CHAPTER 4

IMPLEMENTATION

4.1 ALGORITHM

For implementing the attendance monitoring system using face recognition system, we have to follow the following steps in the same order. Those steps are as follows:

1. Enrollment of students (face detection)
2. Data pre-processing and building the model.
3. Train the CNN-ANN model, apply Test image
4. Face recognition
5. Face Encoding (displaying the label of the face)
6. Store attendance in database

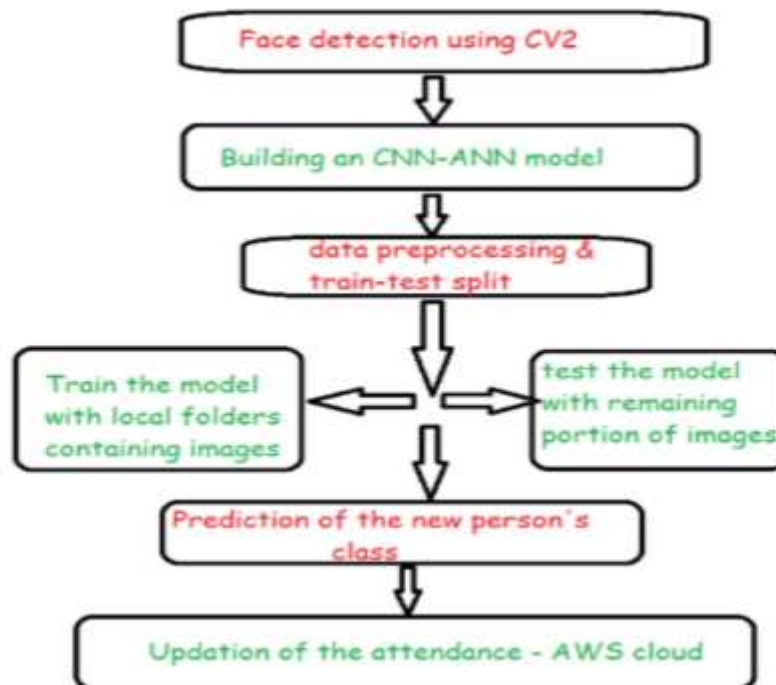


Fig 4.1.a Implementation Steps

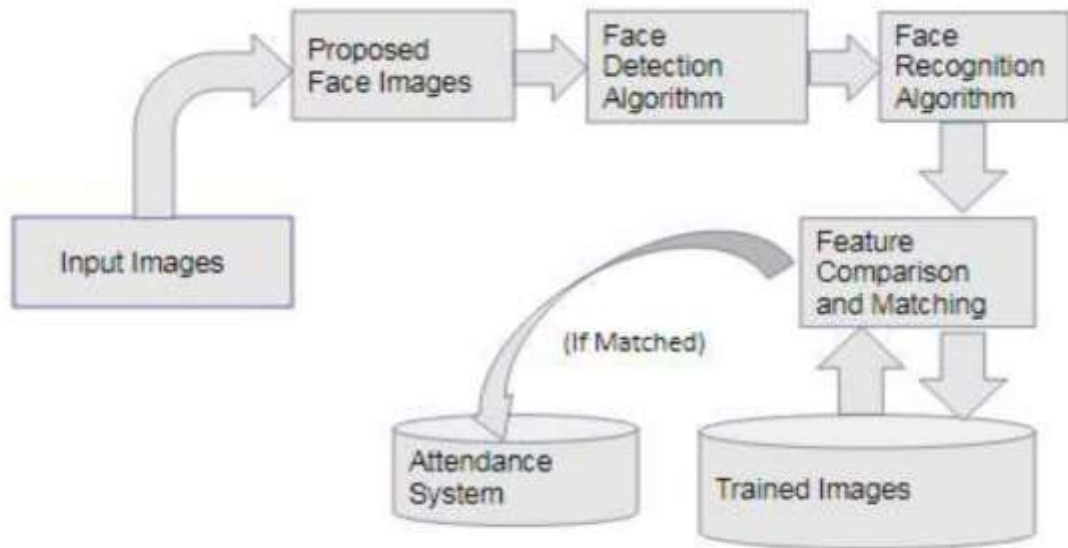


Fig 4.1.b Algorithm Flow

4.2 CODE IMPLEMENTATION

```
# -*- coding: utf-8 -*-
```

```
"""AttendaceMonitoringSystem_FaceRecognition_Using_CNN_CSEC009.ipynb
```

```
Automatically generated by Colaboratory.
```

```
Original file is located at
```

```
https://colab.research.google.com/drive/163V5D6cFbYSzXV-ukoPdnGj04rdAnf0V
```

```
"""
```

```
#Attendance_Monitoring_System_Using_CNN
```

```
*****NEW USER REGISTRATION*****
```

```
newfolder_name='siri-160618733161' # enter the folder name to create example :kohli-18(studentname-rollnum)
```

```
!mkdir attendance_monitoring/$newfolder_name
```

```
from IPython.display import display, Javascript
```

```
from google.colab.output import eval_js
```

```
from base64 import b64decode
```

```
def take_photo(filename='photo.jpg', quality=0.8):
```

```
    js = Javascript("""
```

```
async function takePhoto(quality) {
  const div = document.createElement('div');
  const capture = document.createElement('button');
  capture.textContent = 'Capture';
  div.appendChild(capture);

  const video = document.createElement('video');
  video.style.display = 'block';
  const stream = await navigator.mediaDevices.getUserMedia({ video: true });

  document.body.appendChild(div);
  div.appendChild(video);
  video.srcObject = stream;
  await video.play();

  // Resize the output to fit the video element.
  google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

  // Wait for Capture to be clicked.
  await new Promise((resolve) => capture.onclick = resolve);

  const canvas = document.createElement('canvas');
  canvas.width = video.videoWidth;
  canvas.height = video.videoHeight;
  canvas.getContext('2d').drawImage(video, 0, 0);
  stream.getVideoTracks()[0].stop();
  div.remove();
  return canvas.toDataURL('image/jpeg', quality);
}

display(js)
data = eval_js('takePhoto({})'.format(quality))
binary = b64decode(data.split(',')[1])
with open(filename, 'wb') as f:
  f.write(binary)
return filename

from IPython.display import Image
```

try:

```
filename = take_photo()
print('Saved to {}'.format(filename))
```

```
# Show the image which was just taken.
display(Image(filename))
```

except Exception as err:

```
# Errors will be thrown if the user does not have a webcam or if they do not
# grant the page permission to access it.
print(str(err))
```

```
! mv photo.jpg photo7.jpg
```

```
# ! mv photo1.jpg attendance_monitoring/$newfolder_name # move the recent photo to new folder
```

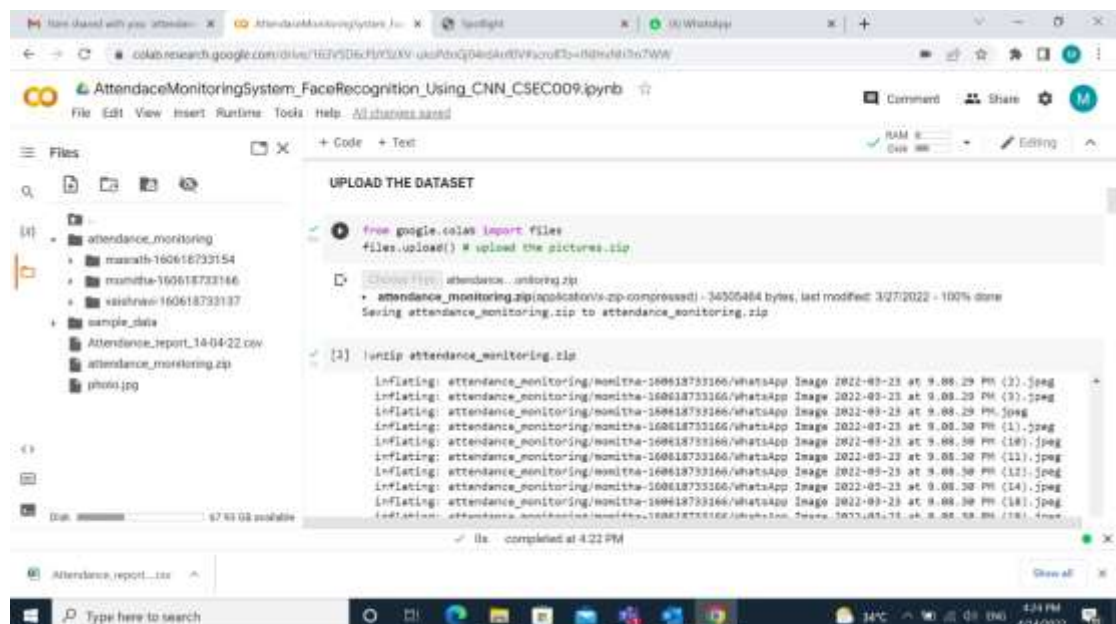
```
! mv photo*.jpg attendance_monitoring/$newfolder_name # move the recent photo to new folder
```

*******UPLOAD THE DATASET*******

from google.colab import files

files.upload() # upload the pictures.zip

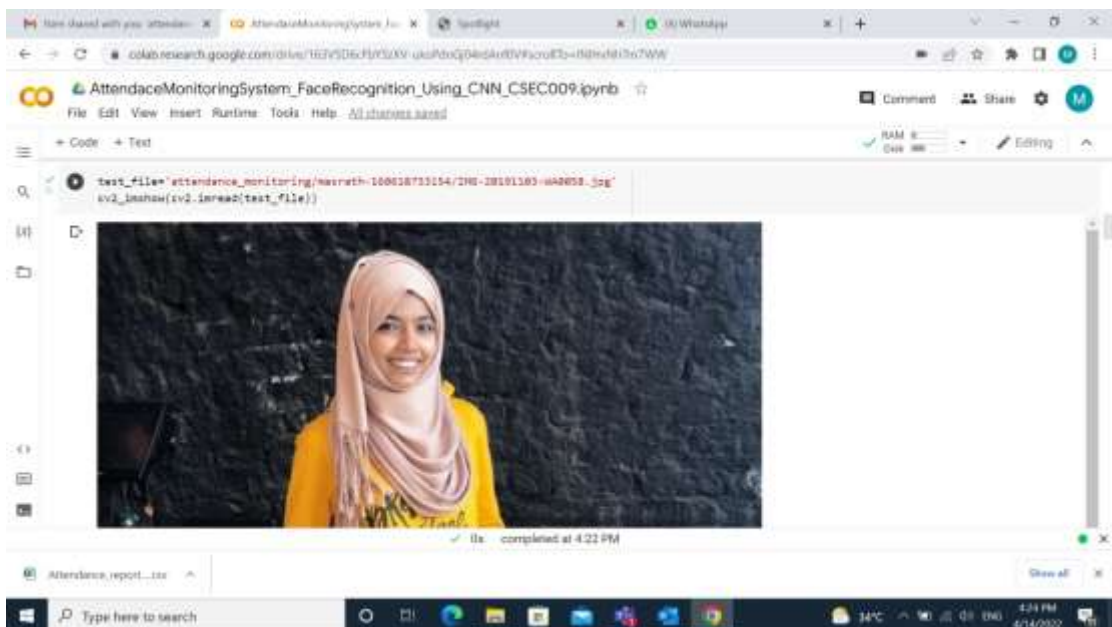
!unzip attendance_monitoring.zip



import matplotlib.pyplot as plt


```
import pandas as pd
import seaborn as sns
import numpy as np
import cv2
import os
import warnings
warnings.filterwarnings('ignore')
from google.colab.patches import cv2_imshow
from IPython.display import display, Javascript, Image
from google.colab.output import eval_js
from base64 import b64decode, b64encode
import PIL
import io
import html
import time
import pytz
from datetime import datetime

test_file='attendance_monitoring/masrath-160618733154/IMG-20191103-WA0058.jpg'
cv2_imshow(cv2.imread(test_file))
```



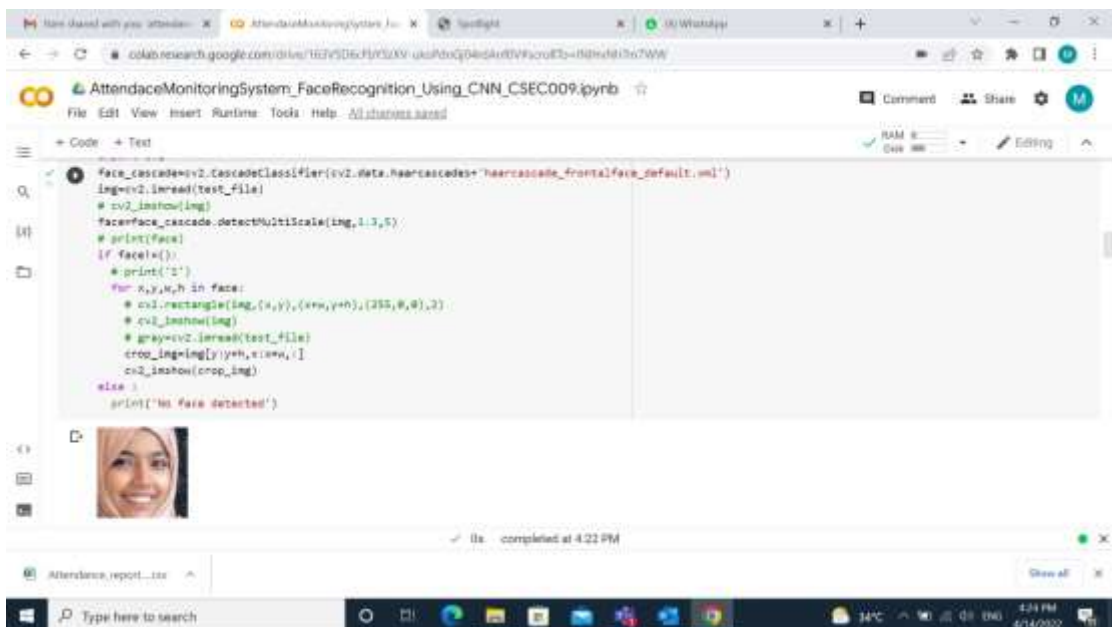
*****CROP THE IMAGE*****

```
import cv2
face_cascade=cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
```

```

img=cv2.imread(test_file)
# cv2_imshow(img)
face=face_cascade.detectMultiScale(img,1.3,5)
# print(face)
if face!=():
    # print('1')
    for x,y,w,h in face:
        # cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        # cv2_imshow(img)
        # gray=cv2.imread(test_file)
        crop_img=img[y:y+h,x:x+w,:]
        cv2_imshow(crop_img)
else :
    print('No face detected')

```



```

imgs_path='attendance_monitoring/'
class_names=os.listdir(imgs_path)
img_size=128

```

```
! ls -lrth attendance_monitoring/masrath*/ * | wc -l
```

*****DATA AUGMENTATION*****

The code will take around 5 minutes to execute

```

from tensorflow.keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array,
load_img

```

```
datagen = ImageDataGenerator(
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')

for folder in os.listdir(imgs_path):
    path=os.path.join(imgs_path,folder)
    for image in os.listdir(path):
        # print(image)
        img_loc=os.path.join(path,image)
        # print(img_loc)
        # cv2_imshow(cv2.imread(img_loc))

img = load_img(img_loc) # this is a PIL image
x = img_to_array(img) # this is a Numpy array with shape (3, 150, 150)
x = x.reshape((1,) + x.shape) # this is a Numpy array with shape (1, 3, 150, 150)

# the .flow() command below generates batches of randomly transformed images
# and saves the results to the `preview/` directory
i = 0
for batch in datagen.flow(x, batch_size=1,
                          save_to_dir=path, save_prefix='augmented', save_format='jpeg'):
    i += 1
    if i > 3:
        break # otherwise the generator would loop indefinitely

! ls -lrth attendance_monitoring/masrath* | wc -l

class_names

face_cascade=cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
def data_prep(data_dir):
    img_data = []
```

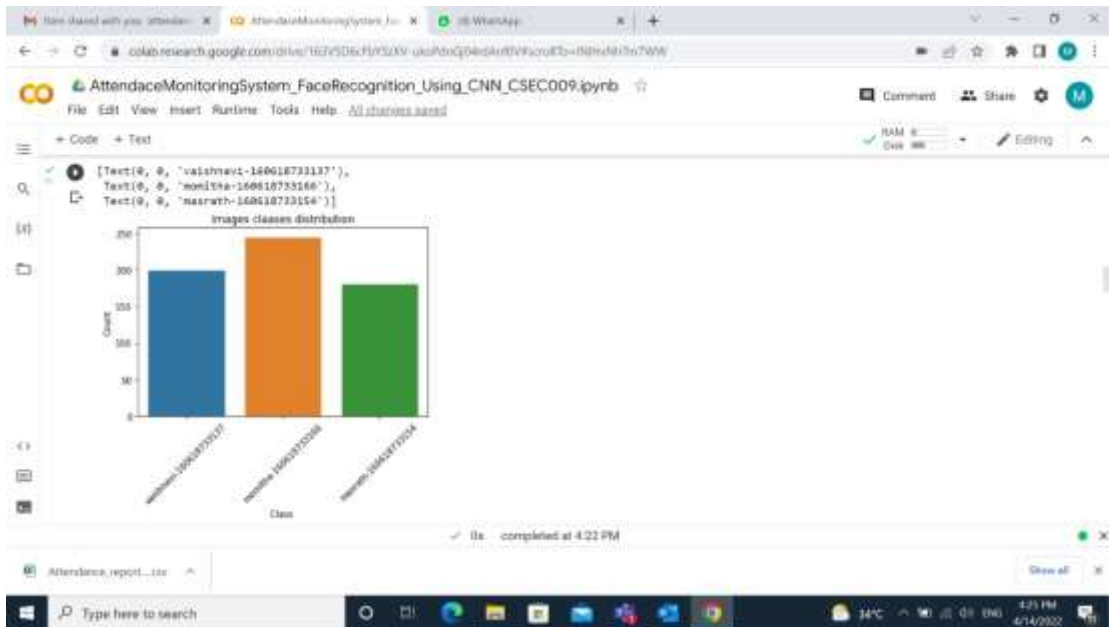
```
labels=[]
for label in class_names:
    path = os.path.join(data_dir, label)
    class_num = class_names.index(label)
    for img in os.listdir(path):
        try:
            img_arr = cv2.imread(os.path.join(path, img), cv2.IMREAD_COLOR)
            face=face_cascade.detectMultiScale(img_arr,1.3,5)
            if face!=():
                # print('face detectoed')
                for x,y,w,h in face:
                    # cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
                    # cv2_imshow(img)
                    # gray=cv2.imread(test_file)
                    crop_img=img_arr[y:y+h,x:x+w,:]
                    # cv2_imshow(crop_img)

                resized_arr = cv2.resize(crop_img, (img_size, img_size))
                img_data.append(resized_arr)
                labels.append(class_num)

        except Exception as e:
            print(e)
return np.array(img_data),np.array(labels)

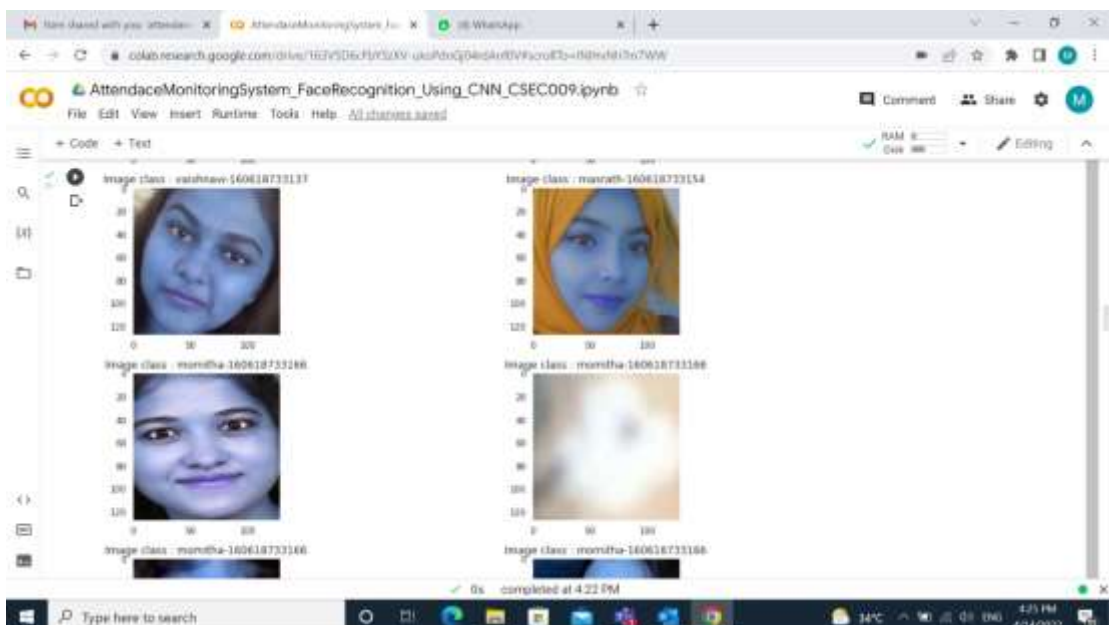
img_data,labels = data_prep(imgs_path)
print(f'the shape of input image data is {img_data.shape}, labels shape is {labels.shape}')

fig=plt.figure(figsize=(6,4))
ax=fig.add_subplot(111)
sns.set_style('dark')
sns.countplot(labels)
plt.title('Images claases distribution')
plt.xlabel('Class ')
plt.ylabel('Count')
ax.set_xticklabels(class_names,rotation=45)
```



```
import random
fig,ax=plt.subplots(5,2)
fig.set_size_inches(15,15)
for i in range(5):
    for j in range (2):
        l=random.randint(0,len(img_data))
        ax[i,j].imshow(img_data[l])
        ax[i,j].set_title('Image class : '+str(class_names[labels[l]]))

plt.tight_layout()
```



```
from tensorflow.keras.applications.xception import preprocess_input
img_data=preprocess_input(img_data)
img_data[0]

from tensorflow.keras.utils import to_categorical
labels=to_categorical(labels)
labels[0]

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(img_data,labels,test_size=0.2,random_state=0,stratify=la
bels)
print(f'X_train size is {X_train.shape}, X_test shape is {X_test.shape}')

from tensorflow.keras.applications.xception import Xception
from tensorflow.keras.layers import *
from tensorflow.keras.models import *
# x_train=preprocess_input(x_train)
xception=Xception(weights='imagenet',include_top=False,input_shape=(img_size,img_size,3))
for layer in xception.layers[:-1]:
    layer.trainable=False

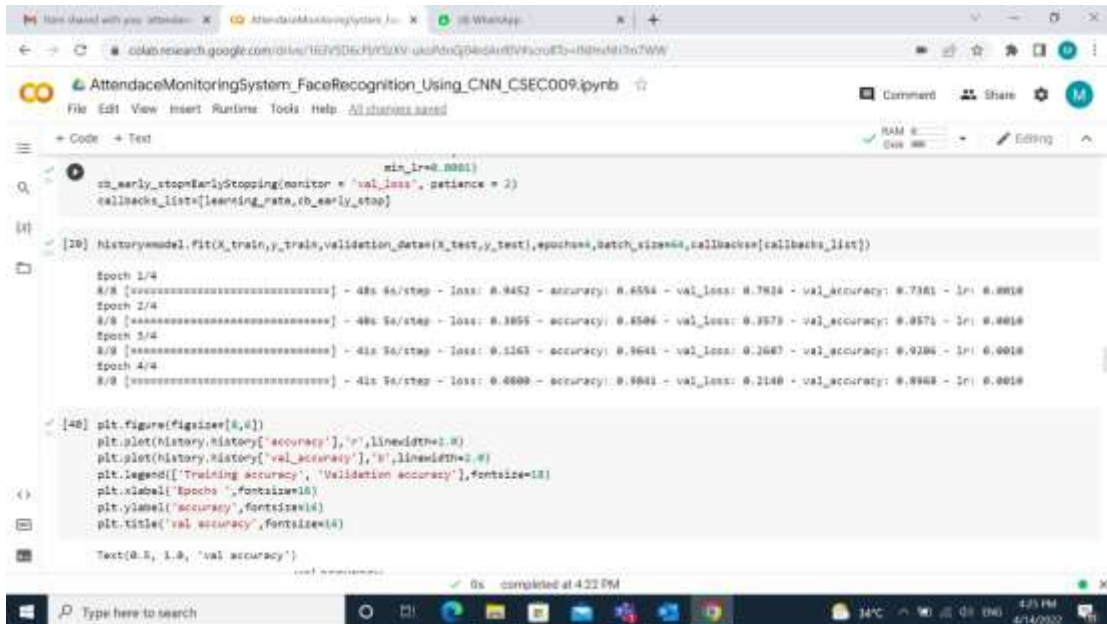
model=Sequential()
model.add(xception)
model.add(Flatten())
model.add(Dense(len(class_names),activation='softmax'))

model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])

model.summary()

from tensorflow.keras.callbacks import ReduceLROnPlateau,EarlyStopping
learning_rate=ReduceLROnPlateau(monitor='val_accuracy',
                                patience=3,
                                verbose=1,
                                factor=0.5,
                                min_lr=0.0001)
cb_early_stop=EarlyStopping(monitor = 'val_loss', patience = 2)
callbacks_list=[learning_rate,cb_early_stop]
```

```
history=model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=4,batch_size=64,callbacks=[callbacks_list])
```



```

min_lr=0.0001)
cb_early_stopping=EarlyStopping(monitor='val_loss',patience=2)
callbacks_list=[learning_rate_scheduler,cb_early_stopping]

[20]: history=model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=4,batch_size=64,callbacks=[callbacks_list])

Epoch 1/4
8/8 [#####] - 48s 8s/step - loss: 0.9452 - accuracy: 0.6554 - val_loss: 0.7824 - val_accuracy: 0.7381 - lr: 0.0010
Epoch 2/4
8/8 [#####] - 48s 8s/step - loss: 0.3855 - accuracy: 0.8506 - val_loss: 0.3573 - val_accuracy: 0.8571 - lr: 0.0010
Epoch 3/4
8/8 [#####] - 41s 5s/step - loss: 0.1263 - accuracy: 0.9641 - val_loss: 0.2687 - val_accuracy: 0.9286 - lr: 0.0010
Epoch 4/4
8/8 [#####] - 41s 5s/step - loss: 0.0800 - accuracy: 0.9841 - val_loss: 0.2140 - val_accuracy: 0.8948 - lr: 0.0010

[40]: plt.figure(figsize=[8,6])
plt.plot(history.history['accuracy'],'r',linewidth=2.0)
plt.plot(history.history['val_accuracy'],'b',linewidth=2.0)
plt.legend(['Training accuracy','Validation accuracy'],fontsize=18)
plt.xlabel('Epochs ',fontsize=16)
plt.ylabel('accuracy',fontsize=16)
plt.title('val accuracy',fontsize=16)

Text(0.5, 1.0, 'val accuracy')
    
```

```
plt.figure(figsize=[8,6])
```

```
plt.plot(history.history['accuracy'],'r',linewidth=2.0)
```

```
plt.plot(history.history['val_accuracy'],'b',linewidth=2.0)
```

```
plt.legend(['Training accuracy','Validation accuracy'],fontsize=18)
```

```
plt.xlabel('Epochs ',fontsize=16)
```

```
plt.ylabel('accuracy',fontsize=16)
```

```
plt.title('val accuracy',fontsize=16)
```



```
test1='attendance_monitoring/masrath-160618733154/augmented_0_2525.jpeg'
#enter file name
x1=[]
img_1 = cv2.imread(test1, cv2.IMREAD_COLOR)
face=face_cascade.detectMultiScale(img_1,1.3,5)
    # print(face)
if face!=():
    print('face detected')
    for x,y,w,h in face:
        cv2.rectangle(img_1,(x,y),(x+w,y+h),(255,0,0),2)
        # cv2_imshow(img)
        # gray=cv2.imread(test_file)
        crop_img=img_1[y:y+h,x:x+w,:]
        # cv2_imshow(crop_img)
        # cv2.putText(frame, label+" "+str(score*100)+'%', (x, y), font, 1, (0, 255, 0), 2)
        # print(crop_img.shape)
        resized_img = cv2.resize(crop_img, (img_size, img_size))
        # print(crop_img.shape)
        img_preproc=preprocess_input(resized_img)
        img_1 = img_preproc.reshape(-1, img_size, img_size, 3)
        # print(img_1.shape)
        #Calling the predict method on model to predict 'me' on the image
        pred = model.predict(img_1)
        # label=np.argmax(prediction)
        score = np.max(pred)
        # label=get_labels(label)
        pred1=np.argmax(pred,axis=1) # for predicting class
        # print(class_names[pred1[0]])
        label=class_names[pred1[0]]
        print('Predicted person is :',label)
else:
    print('No face detected')

pred_results=pd.DataFrame(data=pred,columns=class_names)
# pred_results.head()
import seaborn as sns
# sns.set_theme(style="darkgrid")
ax=sns.barplot(data=pred_results)
```



```
ax.set_xticklabels(class_names,rotation=45)
plt.show()
```

""code below is the alternative to use webcam in colab as we have to access google api to do that .so we have to capture image and then apply model to get predictions""

```
from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode
```

```
def take_photo(filename='photo.jpg', quality=0.8):
    js = Javascript("""
    async function takePhoto(quality) {
      const div = document.createElement('div');
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);
      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia({ video: true });

      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();

      // Resize the output to fit the video element.
      google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

      // Wait for Capture to be clicked.
      await new Promise((resolve) => capture.onclick = resolve);

      const canvas = document.createElement('canvas');
      canvas.width = video.videoWidth;
      canvas.height = video.videoHeight;
      canvas.getContext('2d').drawImage(video, 0, 0);
      stream.getVideoTracks()[0].stop();
    }
    """)
```

```
div.remove();
return canvas.toDataURL('image/jpeg', quality);
}
")
display(js)
data = eval_js('takePhoto({})'.format(quality))
binary = b64decode(data.split(',')[1])
with open(filename, 'wb') as f:
    f.write(binary)
return filename

from IPython.display import Image
try:
    filename = take_photo()
    print('Saved to {}'.format(filename))

    # Show the image which was just taken.
    display(Image(filename))
except Exception as err:
    # Errors will be thrown if the user does not have a webcam or if they do not
    # grant the page permission to access it.
    print(str(err))

"""enter the photo name below to check the prediction"""

test1='photo.jpg' # enter file name    ex: photo.jpg
x1=[]
img_1 = cv2.imread(test1, cv2.IMREAD_COLOR)
face=face_cascade.detectMultiScale(img_1,1.3,5)
    # print(face)
if face!=():
    print('face detected')
    for x,y,w,h in face:
        cv2.rectangle(img_1,(x,y),(x+w,y+h),(255,0,0),2)
        # cv2_imshow(img)
        # gray=cv2.imread(test_file)
        crop_img=img_1[y:y+h,x:x+w,:]
        # cv2_imshow(crop_img)
```

```
# cv2.putText(frame, label+" "+str(score*100)+'%', (x, y), font, 1, (0, 255, 0), 2)
# print(crop_img.shape)
resized_img = cv2.resize(crop_img, (img_size, img_size))
# print(crop_img.shape)
img_preproc=preprocess_input(resized_img)
img_1 = img_preproc.reshape(-1, img_size, img_size, 3)
# print(img_1.shape)
#Calling the predict method on model to predict 'me' on the image
pred = model.predict(img_1)
# label=np.argmax(prediction)
score = np.max(pred)
# label=get_labels(label)
pred1=np.argmax(pred,axis=1) # for predicting class
# print(class_names[pred1[0]])
label=class_names[pred1[0]]
print('Predicted person is :',label)

else:
    print('No face detected')

pred_results=pd.DataFrame(data=pred,columns=class_names)
# pred_results.head()
import seaborn as sns
# sns.set_theme(style="darkgrid")
ax=sns.barplot(data=pred_results)
ax.set_xticklabels(class_names,rotation=45)
plt.show()

*****REAL TIME FACE RECOGNITION*****
# JavaScript to properly create our live video stream using our webcam as input
def video_stream():
    js = Javascript("""
        var video;
        var div = null;
        var stream;
        var captureCanvas;
        var imgElement;
        var labelElement;
```

```
var pendingResolve = null;
var shutdown = false;

function removeDom() {
    stream.getVideoTracks()[0].stop();
    video.remove();
    div.remove();
    video = null;
    div = null;
    stream = null;
    imgElement = null;
    captureCanvas = null;
    labelElement = null;
}

function onAnimationFrame() {
    if (!shutdown) {
        window.requestAnimationFrame(onAnimationFrame);
    }
    if (pendingResolve) {
        var result = "";
        if (!shutdown) {
            captureCanvas.getContext('2d').drawImage(video, 0, 0, 640, 480);
            result = captureCanvas.toDataURL('image/jpeg', 0.8)
        }
        var lp = pendingResolve;
        pendingResolve = null;
        lp(result);
    }
}

async function createDom() {
    if (div !== null) {
        return stream;
    }

    div = document.createElement('div');
    div.style.border = '2px solid black';
```

```
div.style.padding = '3px';
div.style.width = '100%';
div.style.maxWidth = '600px';
document.body.appendChild(div);

const modelOut = document.createElement('div');
modelOut.innerHTML = "<span>Status:</span>";
labelElement = document.createElement('span');
labelElement.innerText = 'No data';
labelElement.style.fontWeight = 'bold';
modelOut.appendChild(labelElement);
div.appendChild(modelOut);

video = document.createElement('video');
video.style.display = 'block';
video.width = div.clientWidth - 6;
video.setAttribute('playsinline', '');
video.onclick = () => { shutdown = true; };
stream = await navigator.mediaDevices.getUserMedia(
  { video: { facingMode: "environment" } });
div.appendChild(video);

imgElement = document.createElement('img');
imgElement.style.position = 'absolute';
imgElement.style.zIndex = 1;
imgElement.onclick = () => { shutdown = true; };
div.appendChild(imgElement);

const instruction = document.createElement('div');
instruction.innerHTML =
  '<span style="color: red; font-weight: bold;">' +
  'When finished, click here or on the video to stop this demo</span>';
div.appendChild(instruction);
instruction.onclick = () => { shutdown = true; };

video.srcObject = stream;
await video.play();
```

```
captureCanvas = document.createElement('canvas');
captureCanvas.width = 640; //video.videoWidth;
captureCanvas.height = 480; //video.videoHeight;
window.requestAnimationFrame(onAnimationFrame);

return stream;
}
async function stream_frame(label, imgData) {
  if (shutdown) {
    removeDom();
    shutdown = false;
    return "";
  }

  var preCreate = Date.now();
  stream = await createDom();

  var preShow = Date.now();
  if (label != "") {
    labelElement.innerHTML = label;
  }

  if (imgData != "") {
    var videoRect = video.getClientRects()[0];
    imgElement.style.top = videoRect.top + "px";
    imgElement.style.left = videoRect.left + "px";
    imgElement.style.width = videoRect.width + "px";
    imgElement.style.height = videoRect.height + "px";
    imgElement.src = imgData;
  }

  var preCapture = Date.now();
  var result = await new Promise(function(resolve, reject) {
    pendingResolve = resolve;
  });
  shutdown = false;

  return {'create': preShow - preCreate,
```

```
'show': preCapture - preShow,  
'capture': Date.now() - preCapture,  
'img': result};  
}  
")  
  
display(js)  
  
def video_frame(label, bbox):  
    data = eval_js('stream_frame("{} ", "{}")'.format(label, bbox))  
    return data  
  
# function to convert the JavaScript object into an OpenCV image  
def js_to_image(js_reply):  
    """  
    Params:  
        js_reply: JavaScript object containing image from webcam  
    Returns:  
        img: OpenCV BGR image  
    """  
    # decode base64 image  
    image_bytes = b64decode(js_reply.split(',')[1])  
    # convert bytes to numpy array  
    jpg_as_np = np.frombuffer(image_bytes, dtype=np.uint8)  
    # decode numpy array into OpenCV BGR image  
    img = cv2.imdecode(jpg_as_np, flags=1)  
  
    return img  
  
# function to convert OpenCV Rectangle bounding box image into base64 byte string to be overlaid  
on video stream  
def bbox_to_bytes(bbox_array):  
    """  
    Params:  
        bbox_array: Numpy array (pixels) containing rectangle to overlay on video stream.  
    Returns:  
        bytes: Base64 image byte string
```

```
""
# convert array into PIL image
bbox_PIL = PIL.Image.fromarray(bbox_array, 'RGBA')
iobuf = io.BytesIO()
# format bbox into png for return
bbox_PIL.save(iobuf, format='png')
# format return string
bbox_bytes = 'data:image/png;base64,{}'.format((str(b64encode(iobuf.getvalue()), 'utf-8'))

return bbox_bytes

from tensorflow.keras.applications.xception import preprocess_input
font = cv2.FONT_HERSHEY_SIMPLEX
# start streaming video from webcam
video_stream()
# label for video
label_html = 'Capturing...'
# initialize bounding box to empty
face_cascade=cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
bbox = ""
count = 0
face_data=[]
dt_data=[]
roll_num=[]
while True:
    js_reply = video_frame(label_html, bbox)
    if not js_reply:
        break

# convert JS response to OpenCV Image
img = js_to_image(js_reply["img"])

# img=cv2.imread(test_file)
# cv2_imshow(img)
# grayscale image for face detection
# gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
# create transparent overlay for bounding box
# bbox_array = np.zeros([480,640,4], dtype=np.uint8)
```




```
face=face_cascade.detectMultiScale(img,1.3,5)
# print(face)
if face!=():
    print('face detected')
    for x,y,w,h in face:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        # cv2.imshow(img)
        # gray=cv2.imread(test_file)
        crop_img=img[y:y+h,x:x+w,:]
        # cv2.imshow(crop_img)
        # cv2.putText(frame, label+" "+str(score*100)+'%', (x, y), font, 1, (0, 255, 0), 2)
        # print(crop_img.shape)
        resized_img = cv2.resize(crop_img, (img_size, img_size))
        # print(crop_img.shape)
        img_preproc=preprocess_input(resized_img)
        img_1 = img_preproc.reshape(-1, img_size, img_size, 3)
        # print(img_1.shape)
        #Calling the predict method on model to predict 'me' on the image
        pred = model.predict(img_1)
        # label=np.argmax(prediction)
        score = np.max(pred)
        # label=get_labels(label)
        pred1=np.argmax(pred,axis=1) # for predicting class
        # print(class_names[pred1[0]])
        label=class_names[pred1[0]]
        if label not in face_data:
            face_data.append(label.split('-')[0])
            roll_num.append(label.split('-')[1])
            # curr = datetime.now()
            IST = pytz.timezone('Asia/Kolkata')
            curr=datetime.now(IST)
            dt = curr.strftime("%d/%m/%Y, %H:%M:%S")
            dt_data.append(dt)

        # label='hi'
        # cv2.putText(img,pred1[0],(x,y),font, 1, (0, 255, 0), 2)
        # cv2.putText(img, label, (x, y), 1, (0, 255, 0), 2)
```



```
cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
cv2.putText(img, label+" "+str(score*100)+'%', (x, y), font, 1, (0, 255, 0), 2)
cv2.imshow(img)
# cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 2)

# bbox_array[:,3] = (bbox_array.max(axis = 2) > 0 ).astype(int) * 255
# convert overlay of bbox into bytes
# bbox_bytes = bbox_to_bytes(bbox_array)
# update bbox so next frame gets new overlay
# bbox = bbox_bytes
else :
    print('No face detected')

results=pd.DataFrame(data={'roll number':roll_num,'Student Name':face_data,'Date':dt_data})
current_date=(datetime.now()).strftime('%d-%m-%y')
results.to_csv('Attendance_report_'+current_date+'.csv',index=False)
print(results.head())

!ls -lrth # list of files

files.download('Attendance_report_28-03-22.csv') # enter file name to download
```

CHAPTER 5

RESULTS



Fig 5.a Accuracy vs Epochs Graph

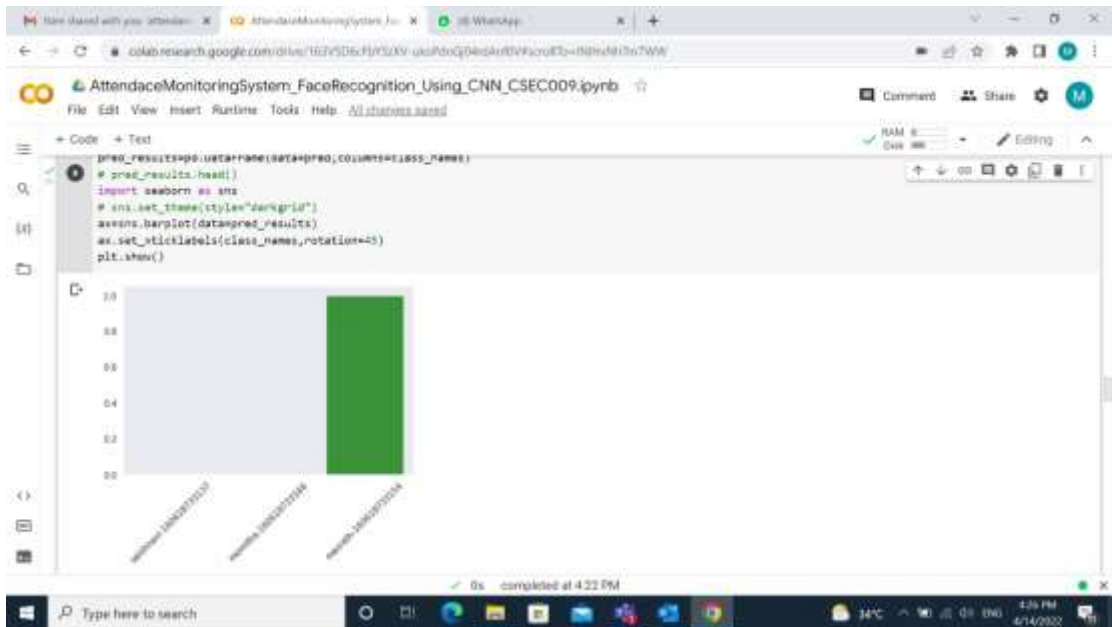
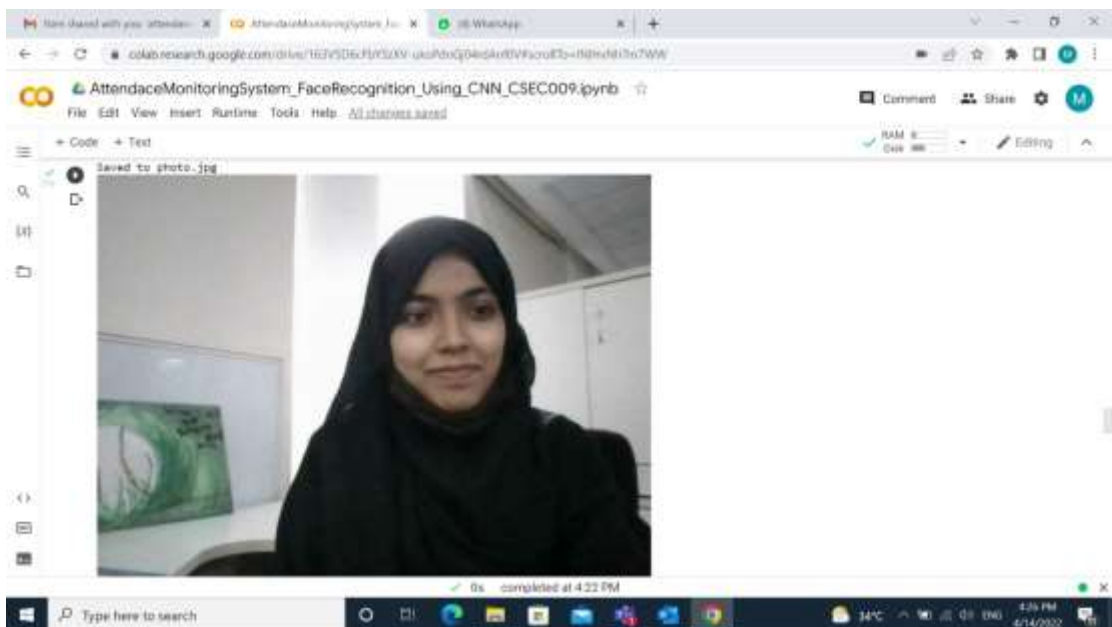
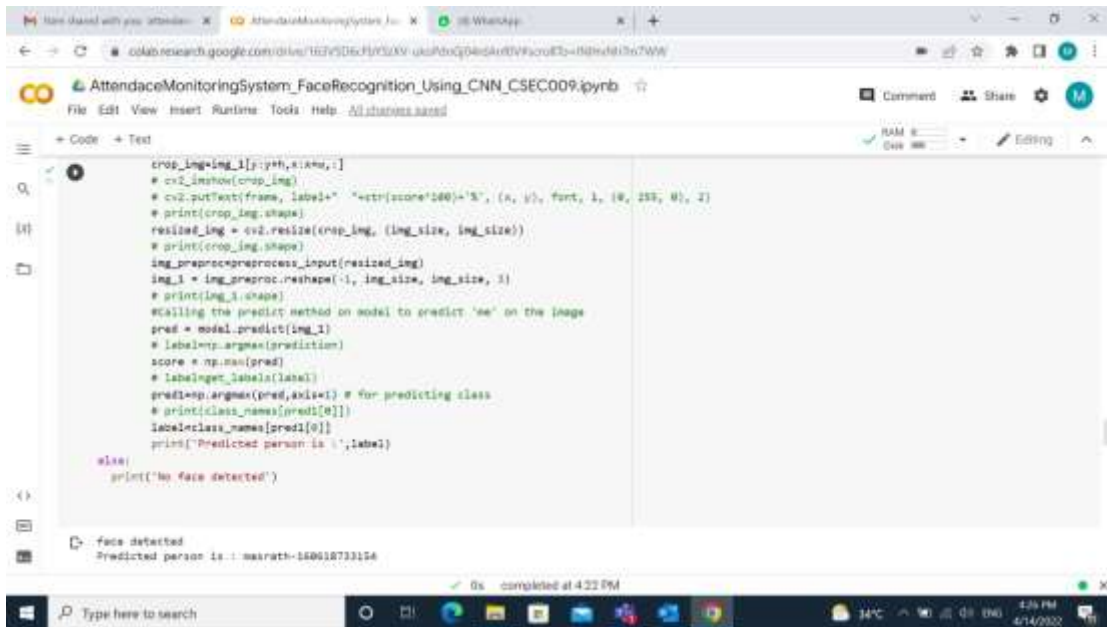


Fig 5.b Student prediction using Image address

The above screen shows the percentage prediction of the student in the form of bar graph.





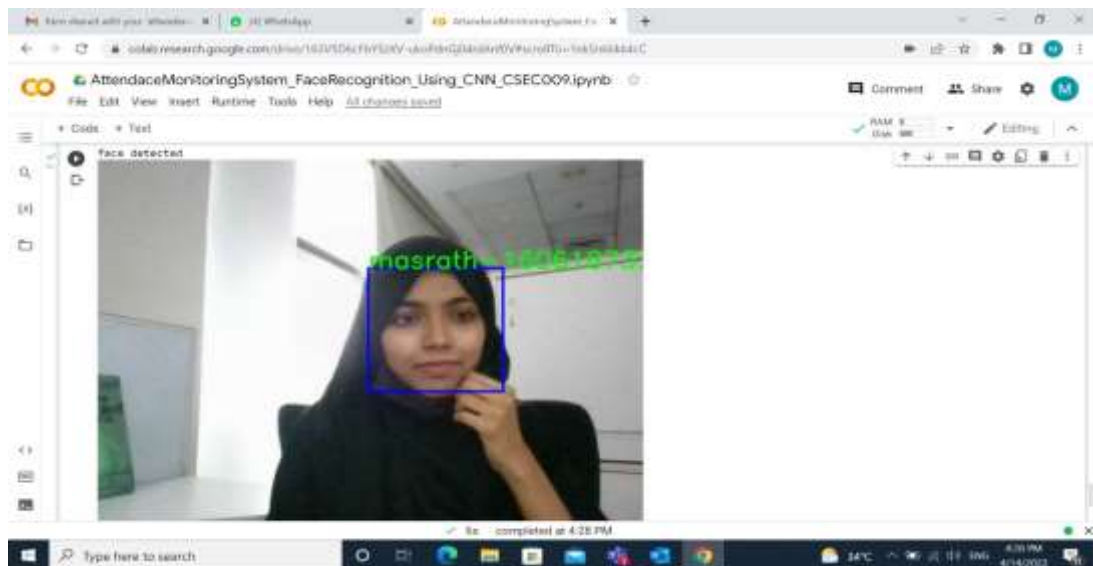
```
crop_img=crop_img[y:y+h,x:x+w,:]
# cv2.imshow('crop_img')
# cv2.putText(frame, label+" "+str(score*100)+"%", (x, y), font, 1, (0, 255, 0), 2)
# print(crop_img.shape)
resized_img = cv2.resize(crop_img, (img_size, img_size))
# print(crop_img.shape)
img_preprocess=process_input(resized_img)
img_1 = img_preproc.reshape(-1, img_size, img_size, 3)
# print(img_1.shape)
# calling the predict method on model to predict 'ee' on the image
pred = model.predict(img_1)
# label=np.argmax(prediction)
score = np.max(pred)
# label=get_label(label)
pred=np.argmax(pred,axis=1) # for predicting class
# print(class_names[pred[0]])
label=class_names[pred[0]]
print("Predicted person is :",label)

else:
    print("No face detected")
```

face detected
Predicted person is : masrath-160618733154

Fig 5.c Student Prediction using captured image from webcam

In the above output screen, webcam clicks the picture of the student and takes it as input and predicts the name of that student along with roll number.



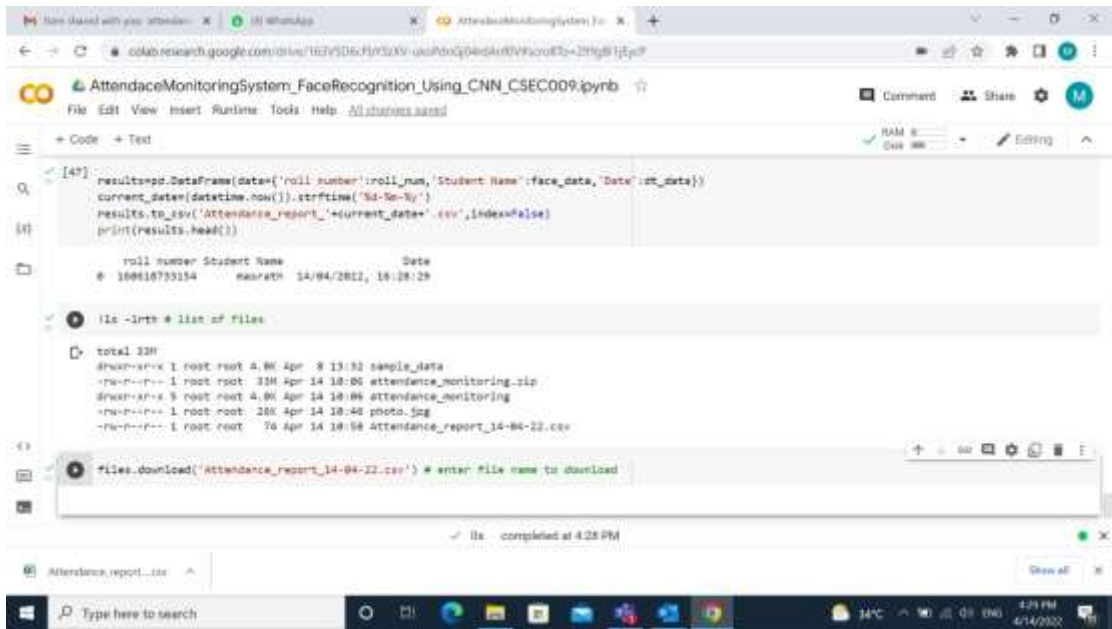


Fig 5.d Attendance Marking using Face Recognition from live webcam

In the above output screen, input is taken through live video stream using webcam and then the predicted students name is appeared on the screen around his/her face along with green rectangular box indicating correct prediction.

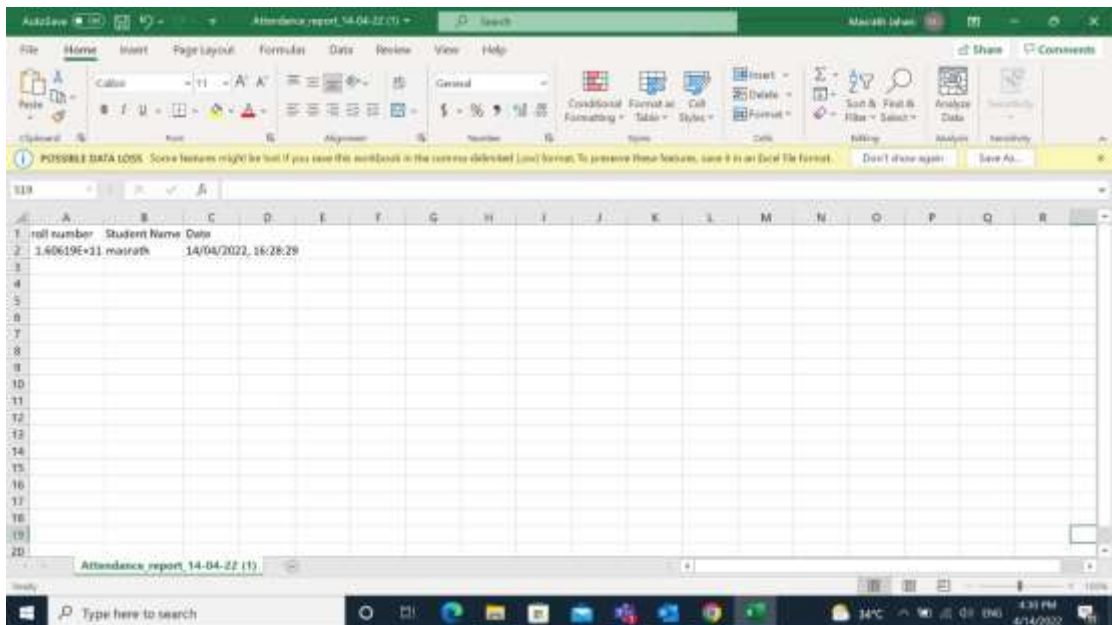


Fig 5.e Output Excel Sheet



In the above output screen, attendance is recorded in an excel sheet each day along with roll number, student name, date and time. The recorded Attendance report is saved each day with a particular date and can be downloaded.

CHAPTER 6

CONCLUSION

The CNN algorithm has helped us create many great applications around us! Facebook is the perfect example! It has trained its DeepFace CNN model on millions of images and has an accuracy of 97% to recognize anyone on Facebook. This may surpass even humans! as you can remember only a few faces.

Face recognition technologies have been associated generally with very costly top secure applications. Certain applications of face recognition technology are now cost effective, reliable and highly accurate. As a result there are no technological or financial barriers for stepping from the pilot project to widespread deployment.

The results of face recognition are greatly influenced by:

1. Training data
2. The preprocessing function
3. The type of network selected
4. Activation functions

CNN is being used in the medical industry as well to help doctors get an early prediction about benign or malignant cancer using the tumor images. Similarly, get an idea about typhoid by looking at the X-ray images, etc. The usage of CNN are many, and developing fast around us!

CHAPTER 7

FUTURE SCOPE

1. A feature which can give intruder alert can be included in the system. Furthermore, the images of unknown people can be saved in an efficient manner and displayed in the system for better security.
2. The number of training images can be reduced so that less storage is required. This can be done by removing duplicate images of the same person, or images with similar embeddings.
3. The training time can be reduced by retraining the classifier only for the newly added images.

4. A feature can be added where an employee is automatically sent a warning if his attendance or working hours are below the threshold.
5. Wrongly classified images can be added to the training dataset with the correct label so as to increase the accuracy of the recognition model.
6. For access control verification and identification of authentic users it can also be installed in bank lockers and vaults. For identification of criminals the system can be used by police force also.
7. It can also be used during examinations such as Civil Services Exam, SSC, IIT, MBBS, and others to identify the candidates. This system can be deployed for verification and attendance tracking at various government offices and corporates.

CHAPTER 8

REFERENCES

1. HAO YANG and XIAOFENG HAN in 2020 have developed a model which catches the live images from camera. Then it applies different algorithms for face recognition and face detection.
2. RONG QI1 , RUI-SHENG JIA 1,2, QI-CHAO MAO1 , HONG-MEI SUN 1,2, AND LING-QUN ZUO1 in 2019 has proposed a system for face detection method based on cascaded convolution networks.
3. MUHAMMAD ZEESHAN KHAN1, SAAD HAROUS 2, SALEET UL HASSAN1, MUHAMMAD USMAN GHANI KHAN1, RAZI IQBAL 3, (Senior Member, IEEE) AND SHAHID MUMTAZ in 2019 have proposed Deep Unified Model For Face Recognition Based on Convolution Neural Network and Edge Computing.



1. 4.FOTEINI P. FILIPPIDOU, GEORGE A. PAPAKOSTAS, in 2020 came up with the Single Sample Face Recognition Using Convolutional Neural Networks for Automated Attendance Systems.
2. 5.ANATASIYA YU, STRUEVA, ELEVA V.IVANOVA, in 2021 proposed Student Attendance Control System with Face Recognition Based on Neural Network.
3. Kishor Kumar Reddy C and Vijaya Babu B, “ISPM: Improved Snow Prediction Model to Nowcast the Presence of Snow/No-Snow”, International Review on Computers and Software, 2015.
4. (<http://www.praiseworthyprize.org/jsm/index.php?journal=irecos&page=article&op=view&path%5B%5D=17055>)
5. Kishor Kumar Reddy C, Rupa C H and Vijaya Babu B, “SLGAS: Supervised Learning using Gain Ratio as Attribute Selection Measure to Nowcast Snow/No-Snow”, International Review on Computers and Software, 2015.
6. (<http://www.praiseworthyprize.org/jsm/index.php?journal=irecos&page=article&op=view&path%5B%5D=16706>)
7. Kishor Kumar Reddy C, Vijaya Babu B, Rupa C H, “SLEAS: Supervised Learning using Entropy as Attribute Selection Measure”, International Journal of Engineering and Technology, 2014.
8. (<http://www.enggjournals.com/ijet/docs/IJET14-06-05-210.pdf>)
9. Kishor Kumar Reddy C, Rupa C H and Vijaya Babu B, “A Pragmatic Methodology to Predict the Presence of Snow/No-Snow using Supervised Learning Methodologies”, International Journal of Applied Engineering Research, 2014.
10. (<http://www.ripublication.com/Volume/ijaerv9n21.htm>)
11. Kishor Kumar Reddy C, Rupa C H and Vijaya Babu, “SPM: A Fast and Scalable Model for Predicting Snow/No-Snow”, World Applied Sciences Journal, 2014.