

PROJECT REPORT

(Project Term July –November 2021)

(FACE MASK DETECTION)

Submitted by:

Name: Mohammad Musheer Anwar

Registration Number: 11910270

Course Code:INT246

Under the Guidance of

(Dr. Sagar Pande)

School of Computer Science and Engineering



L LOVELY
P ROFESSIONAL
U NIVERSITY

DECLARATION

We hereby declare that the project work entitled (“FACE MASK DETECTION ”) is an authentic record of our own work carried out as requirements of Project for the award of B.Tech degree in COMPUTER SCIENCE AND ENGINEERING from Lovely Professional University, Phagwara, under the guidance of (Dr. Sagar Pandey), during August to November 2021. All the information furnished in this project report is based on our own intensive work and is genuine.

Name: Mohammad Musheer Anwar

Registration Number: 11910270



Signature:

CERTIFICATE

This is to certify that the declaration statement made by this group of students is correct to the best of my knowledge and belief. They have completed this Project under my guidance and supervision. The present work is the result of their original investigation, effort and study. No part of the work has ever been submitted for any other degree at any University. The Project is fit for the submission and partial fulfillment of the conditions for the award of B.Tech degree in COMPUTER SCIENCE AND ENGINEERING from Lovely Professional University, Phagwara.

Signature and Name of the Mentor

Designation

School of Computer Science and Engineering,
Lovely Professional University,
Phagwara, Punjab.

Date :19 November 2021

ACKNOWLEDGEMENT

In the present world of competition there is a race of existence in which those are having will to come forward succeed. Project is like a bridge between theoretical and practical working. With this willing I joined this particular project. First of all, I would like to thanks the supreme power of Almighty God who is obviously the one has always guided me to work on , the one has always guided me to work on the right path of life. Without his grace this project could not become a reality.

Next to him are my parents, whom I am greatly indebted for me brought up with love and encouragement to this stage. I am feeling oblige in taking an opportunity to sincerely thanks my all teachers department of computer Science and Engineering and mentors of Lovely Professional University who guided me towards doing this project of “FACE MASK DETECTION” under as my course helped me to know the different aspects of Machine Learning Application in different purposes.

I am highly obliged in taking the opportunity to thanks the Lovely Professional University which helped me and guided me to learn more about the Machine Learning and its application, and also to different platforms which provide me the path for learning.

Table of Contents:

S.No	Subject	Page No
1.	Title page	1
2.	Declaration	2
3.	Certificate	3
4.	Acknowledgement	4
5.	Table of contents	5
6.	Intoduction	6
7.	Problem statement	6
8.	Existing system	7
9.	Methodology	8
10.	Software used	9-10
11.	Design	10-13
12.	Implementation	14-19
13.	conclusion	19-20
14.	References	20

1. Introduction:

A face mask detection software reduces many efforts in the tasks which can be automated. A visual input is taken from a particular area using camera. Then using the predefined dataset, the visuals are compared to identify whether face mask is present or not. Once we have the visual form of the input, it can be given as an input to the deep learned model to predict the action or output that is supposed to be served to the user.

The model is needed to have patterns in which it can distinguish between masked and non-masked people. The model will also consist of the actions to be taken, once it determines the presence of face mask. Also, a basic thing that this software requires is visual. For that, any camera like CCTV, wireless webcams, etc. can be used to capture visuals.

Various python libraries like tensor flow, numpy, sklearn are used for implementation of machine learning. UI will consist of a window which will pop up on user's command and it will display the visuals given by the camera to identify the face mask and then it will display the output accordingly whether face mask is present or not. Face mask detection software has many wide ranges of applications in this pandemic era.

2. Problem Statement:

Covid-19 has created a whole new frequency, and people realize that it is entering a new world. Although our society is currently changing rapidly, we must respond quickly to the new needs surrounding all of us. A risk-free environment will be everyone's top priority to make life more meaningful than before. We need to find ways to protect those who return to work and keep ourselves and our loved ones out of trouble. New plans are made every day according to guidelines and rules. As masks become a new standard in daily life, it is necessary to be vigilant at all times to create a safe environment conducive to public safety. For security reasons, many areas of society seem to be using some covid tracking tools. One of the most important tools is the mask detector. Using this system, you can check who does not have the mask you need. These systems, in combination with existing surveillance systems, use innovative neural network algorithms to check whether a person is wearing a mask. In this chapter, we will briefly discuss artificial intelligence and its subgroups, namely machine learning and deep learning, and deep learning. Following the framework is a simple implementation of the mask detection system. This project "Face mask detection using machine learning" which is based on machine learning concepts, aims at providing service of identifying

whether a person is wearing mask or not. In this project we analyse the image by using CNN model.

3. Existing System:

So far, there are a lot of research papers related object detection and face recognition. For example, in A. Rosebrock, 2020. COVID-19: Face Mask Detector with OpenCV, Keras/TensorFlow, And Deep Learning – Pyimagesearch, it has been explained that how to implement face mask detection with the help of deep learning and open source machine learning libraries such as Keras and TensorFlow. Also, the article discusses about the dataset that has been used to train the face mask detector model. Python script has been used to train the the model with open-source libraries like Keras and Tensorflow. Also, in D. Dwivedi, “Data Science, Artificial Intelligence, Deep Learning, Computer Vision, Machine Learning, Data Visualization and Coffee”, it has been mentioned that face recognition would gain significant importance and promising applications in the upcoming years. Face detection is the fundamental part of face recognition process. In recent times, a lot of research and study work has done to make the face detection more advance enhance the accuracy of the prediction. It has wide applications across many fields such as law-enforcement, entertainment, safety, etc. The Object Detection Algorithm uses edge or line detection features proposed by Viola and Jones in their research paper “Rapid Object Detection using a Boosted Cascade of Simple Features” published in 2001. The algorithm is given a lot of positive images consisting of faces, and a lot of negative images not consisting of any face to train on them.

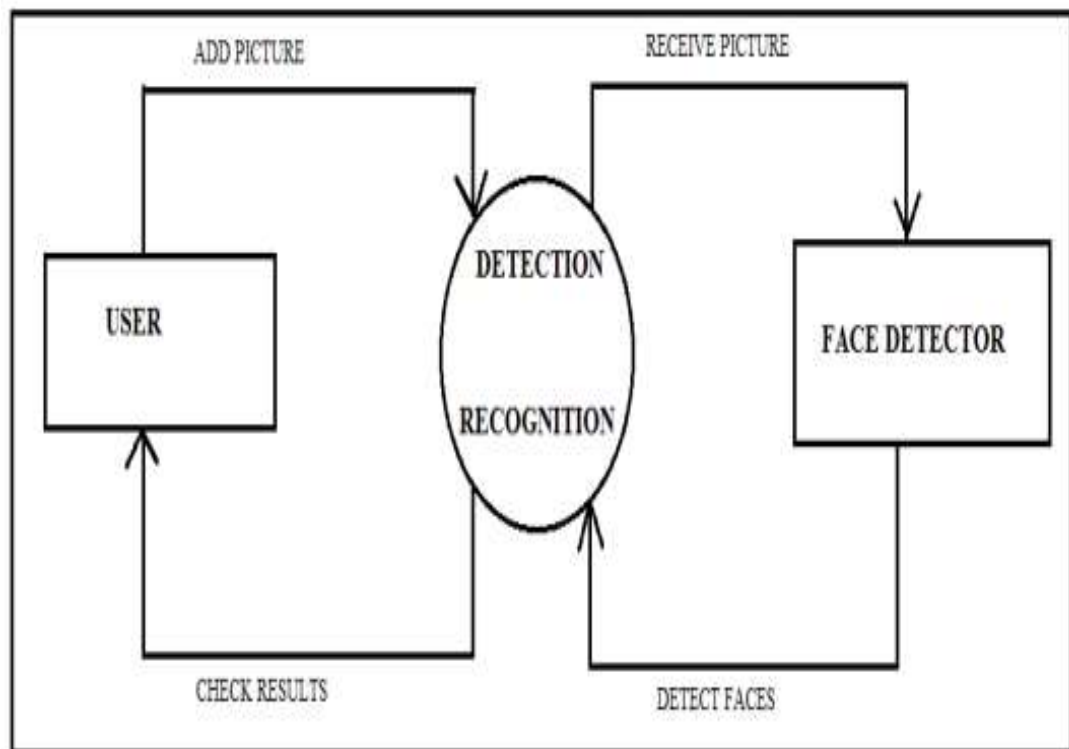
3.1 Methodology:

Dataset The data set used in the project consists of 3835 images, including 1916 faces with masks and 1919 faces without masks. All the images are actual images taken from Kaggle datasets and RMFD (Real-World Masked Face Dataset). The images in the dataset covers diverse races. The proportion of masked to unmasked faces determine that the dataset is balanced. In our approach, we have dedicated 80% of the dataset as the training data and the remaining 20% as the testing data, which makes the split ratio as 0.8:0.2 of train to test set.

Architecture The algorithm consists of two parts: first it detects the presence of multiple faces in a certain image or video sequence, and then in the second part it detects the presence or absence of a face mask on the face. To detect face we have used OpenCV library. The latest OpenCV includes a Deep Neural Network (DNN) module equipped with a pre-trained Convolutional Neural Network (CNN) for facial recognition. The new model improves face recognition performance compared to traditional models. The faces, we need to evaluate the bounding box around it and send it to the second part of the model to check whether the face has a mask or not. The second part of the model is trained with the labelled dataset. We used

Keras in conjunction with TensorFlow to train our model. The first part of the training includes storing all image labels in a NumPy array and the corresponding images are also reshaped for the model base (224, 244, 3) Before inputting, we perform the following random image augmentation, Rotations up to 20 degrees , zoom in and zoom out up to 15%, offset width or height up to 20%, cutting angle up to 15 degrees counter clockwise, flip inputs horizontally, and points outside the boundaries of the inputs are padded from the closest available pixel of the input. The basic model used here is MobileNetV2 with the specified 'ImageNet' weights. ImageNet is an image database that has been trained on hundreds of thousands of images and therefore helps a lot with image classification. For the base model, we cut off the head and used a number of our self-defined layers. We used an average grouping layer, a flat layer, a dense layer with exit shape (None, 128) and ReLU activation, a 50% demolition layer for optimization, finally another dense layer with exit shape (None, 2) and Softmax activation is used. The flow diagram of the algorithm process is shown below.

- **Data Flow Diagram:**



4. Software Used:

4.1 Introduction:

In this project i have used different softwares like jupyter notebook for writing the code, used python language, Machine Learning concepts, different libraries such as numpy, pandas, opencv, matplotlib and sklearn.

4.2 Description:

First i have install the jupyter notebook in our system after that I have import all the required library like pandas, numpy, matplotlib, sklearn, opencv.

```
In [1]: import cv2

In [2]: img = cv2.imread(r"C:\Users\Dell\Python ML\pic2.jpg")

In [3]: img.shape
Out[3]: (400, 600, 3)

In [4]: img[0]
Out[4]: array([[ 74, 127, 178],
               [ 75, 128, 179],
               [ 76, 129, 180],
               ...,
               [233, 231, 223],
               [232, 230, 222],
               [232, 230, 222]], dtype=uint8)

In [5]: import matplotlib.pyplot as plt

In [6]: plt.imshow(img)
Out[6]: <matplotlib.image.AxesImage at 0x1bcada74af0>
```




Figure: 1

```
In [23]: np.save("without_mask.npy",data)

In [25]: np.save("with_mask.npy",data)

In [27]: plt.imshow(data[5])

Out[27]: <matplotlib.image.AxesImage at 0x1bcb370f610>
```



Figure: 2

5. Design:

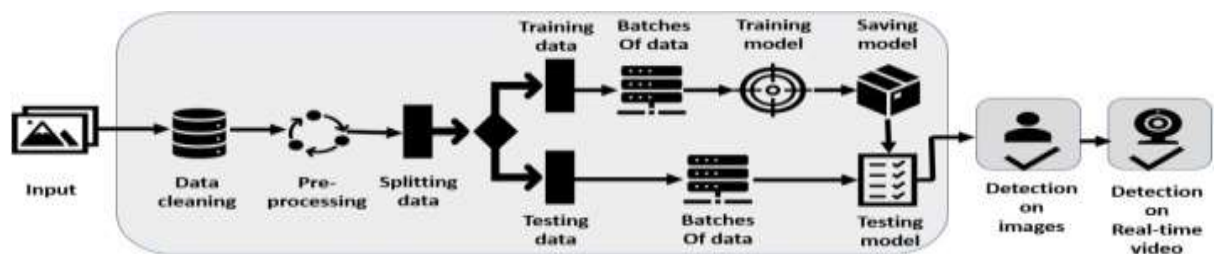


Figure: 3

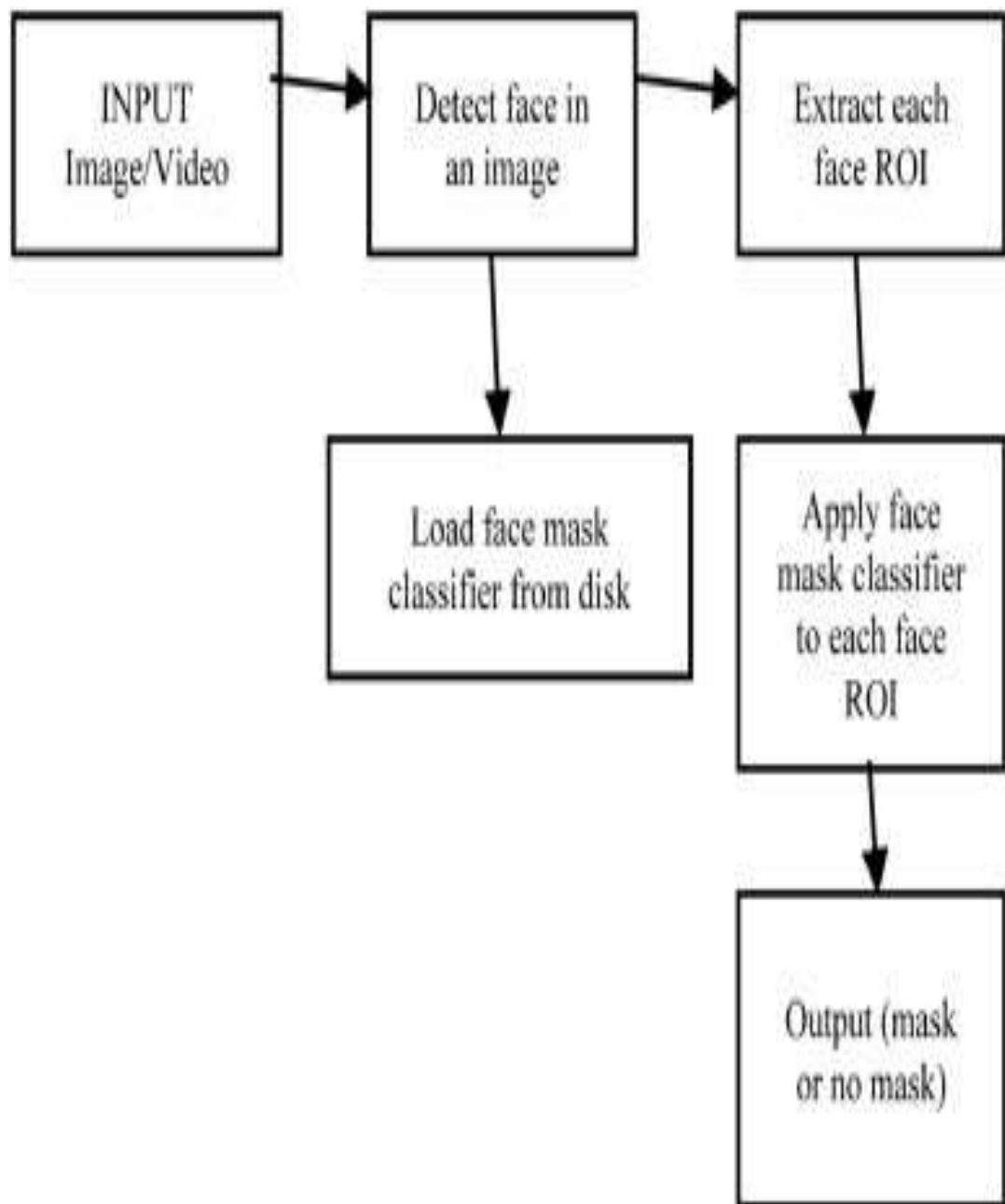


Figure: 4

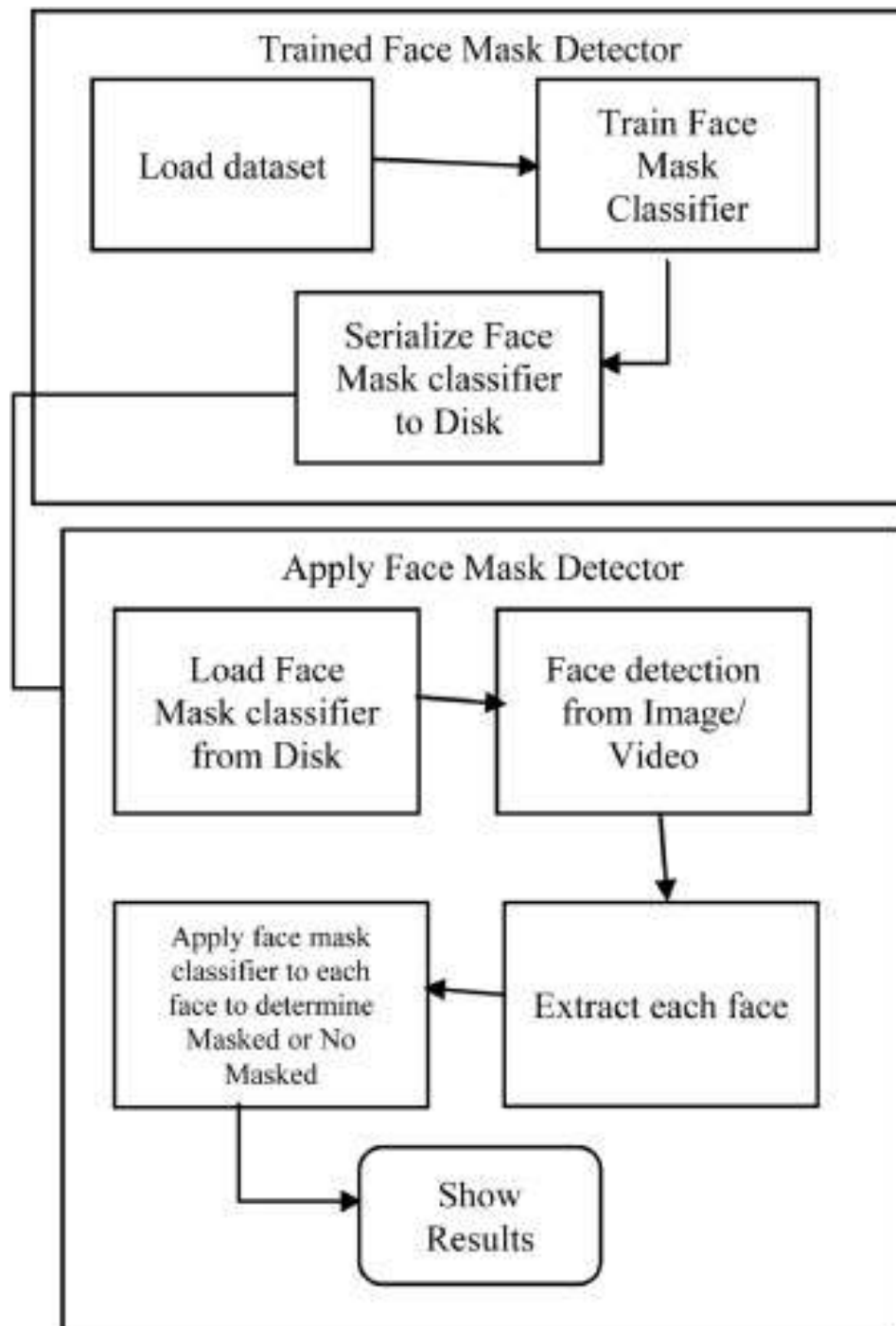


Figure: 5

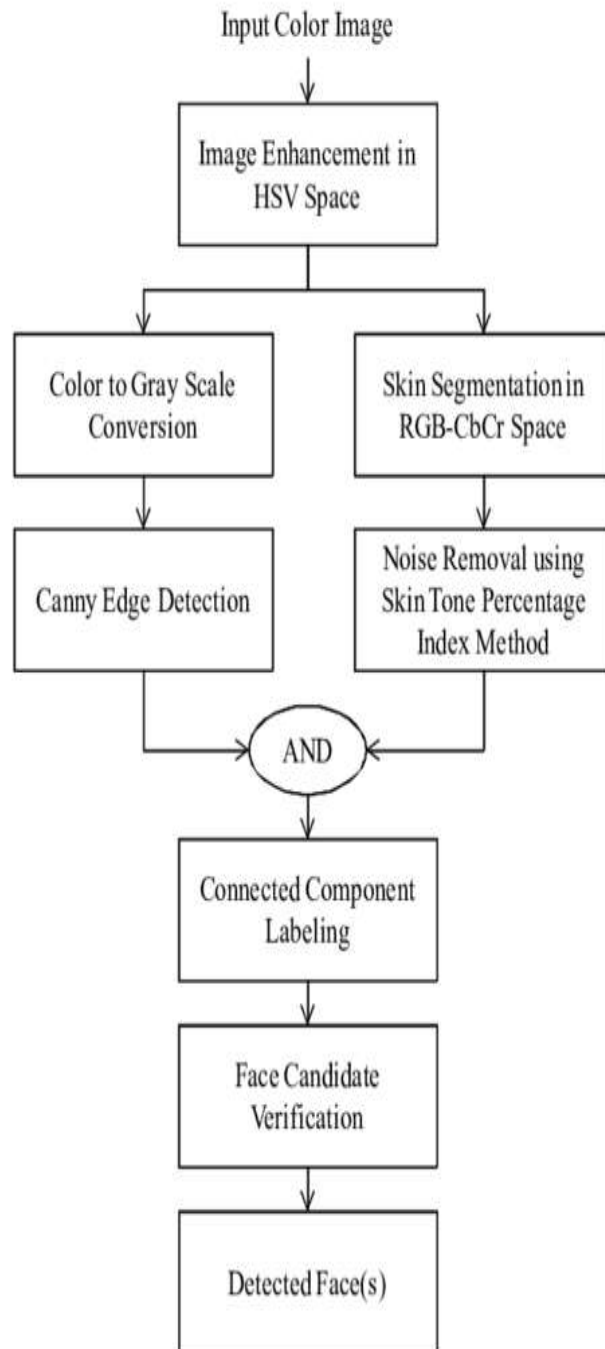


Figure: 6

6. Implementation:

Step:1

```
In [1]: import cv2

In [2]: img = cv2.imread(r"C:\Users\Dell\Python ML\pic2.jpg")

In [3]: img.shape
Out[3]: (400, 600, 3)

In [4]: img[0]
Out[4]: array([[ 74, 127, 178],
               [ 75, 128, 179],
               [ 76, 129, 180],
               ...,
               [233, 231, 223],
               [232, 230, 222],
               [232, 230, 222]], dtype=uint8)

In [5]: import matplotlib.pyplot as plt
```

Figure: 7

Step:2

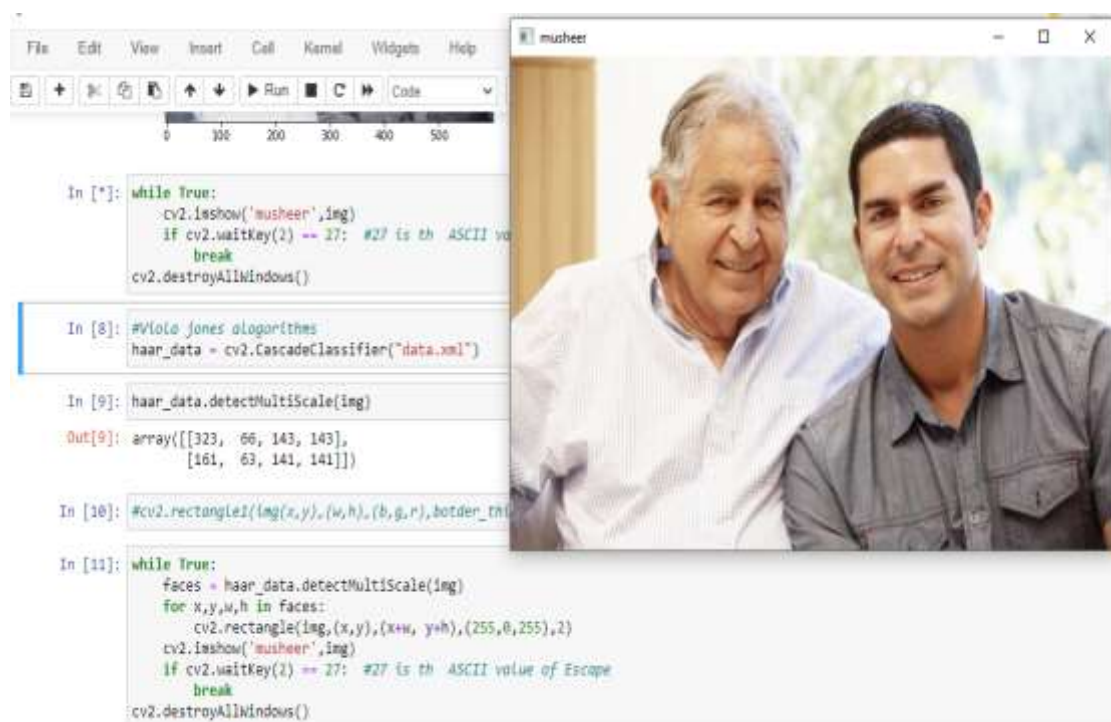


Figure: 8

Step:3



Figure: 9

Step:4

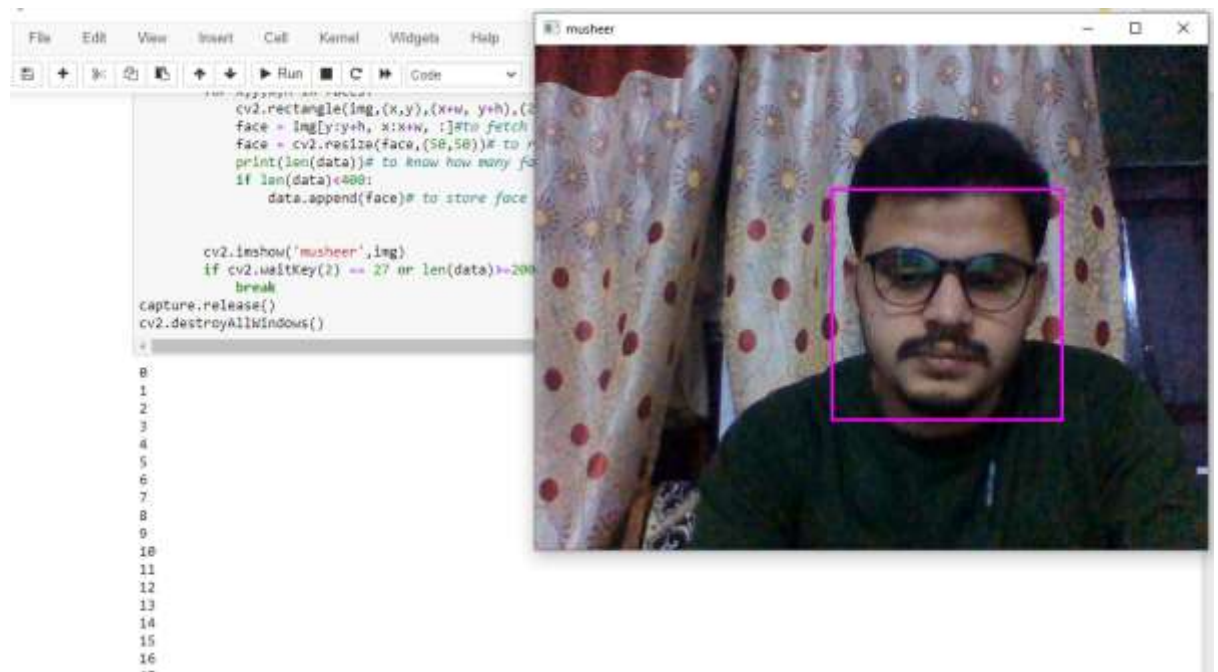


Figure: 10

Step:5

```
if cv2.waitKey(2) == 27: #27 is th ASCII value of escape
    break
cv2.destroyAllWindows()

capture = cv2.VideoCapture(0)# to open camera or if want to collect data from video then mention path of the video like(video.npy)
data = []

while True:
    flag, img = capture.read()
    if flag:
        faces = haar_data.detectMultiScale(img)
        for x,y,w,h in faces:
            cv2.rectangle(img,(x,y),(x+w, y+h),(255,0,255),2)
            face = img[y:y+h, x:x+w, : ]#to fetch the face from image
            face = cv2.resize(face,(50,50))# to resize the face image
            print(len(data))# to know how many face collected
            if len(data)<400:
                data.append(face)# to store face capture data in data variable

        cv2.imshow('musheer',img)
        if cv2.waitKey(2) == 27 or len(data)>=400: #27 is th ASCII value of Escape
            break
capture.release()
cv2.destroyAllWindows()
```

Figure: 11

Step:6

```
In [22]: x[:,1:4]
```

```
Out[22]: array([[4, 7, 5],
               [7, 8, 9],
               [6, 4, 3],
               [6, 7, 9]])
```

```
In [23]: np.save("without_mask.npy",data)
```

```
In [25]: np.save("with_mask.npy",data)
```

```
In [16]: plt.imshow(data[9])
```

```
Out[16]: <matplotlib.image.AxesImage at 0x139d7a2e100>
```

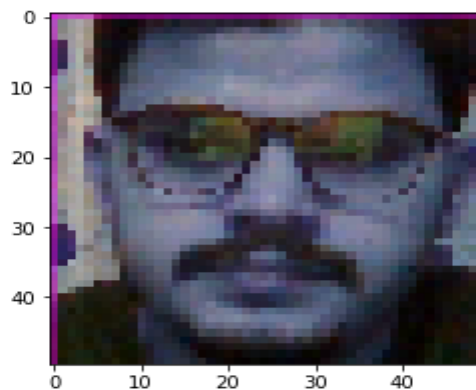


Figure: 12

Step:7

```
In [1]: import numpy as np
import cv2

In [2]: with_mask = np.load("with_mask.npy")
without_mask = np.load("without_mask.npy")

In [3]: with_mask.shape
Out[3]: (200, 50, 50, 3)

In [4]: without_mask.shape
Out[4]: (200, 50, 50, 3)

In [5]: with_mask = with_mask.reshape(200,50*50*3)#converting data into 2D
without_mask = without_mask.reshape(200,50*50*3)

In [6]: with_mask.shape
Out[6]: (200, 7500)

In [7]: without_mask.shape
Out[7]: (200, 7500)

In [8]: x=np.r_[with_mask, without_mask]#r_ is used to concatenate data
```

Figure: 13

Step:8

```
without_mask.shape
(200, 7500)

x=np.r_[with_mask, without_mask]#r_ is used to concatenate data

x.shape
(400, 7500)

labels = np.zeros(x.shape[0])

labels[200:] = 1.0 # next 200 data images are with mask data
# if the output is zero we are wearing a mask or if output is 1 then we are not wearing a mask

names = {0:"Mask", 1:"No Mask"}

# ML algo
#svm- support vector machine
#svc- support vector classification
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

Figure: 14

Step:9

```
Successfully installed sklearn-0.0

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, labels, test_size=0.25) # means 25% data reserve for testing and 75% data for

x_train.shape # as this data has 7500 cols, it will slow down the ML algo, for this we will use dimensionality reduction
               # technique in ml

(300, 7500)

#x_train, x_test, y_train, y_test = train_test_split(x, labels, test_size=0.25) # for shuffling of data we again doing testing and
#training of data

# for dimensionality reduction
#PCA- Principal component Analysis
from sklearn.decomposition import PCA

pca = PCA(n_components=3) # convert into 3 D
x_train = pca.fit_transform(x_train)

x_train.shape

(300, 3)
```

Figure: 15

Step:10

```
svm = SVC()
svm.fit(x_train, y_train)

SVC()

x_test = pca.fit_transform(x_test)
y_pred = svm.predict(x_test)

accuracy_score(y_test, y_pred)

0.76

haar_data = cv2.CascadeClassifier("data.xml")
capture = cv2.VideoCapture(0) # to open camera or if want to collect data from video then mention path of the video like {video.mp4}
data = []
font = cv2.FONT_HERSHEY_COMPLEX

while True:
    flag, img = capture.read()
    if flag:
        faces = haar_data.detectMultiScale(img)
        for x, y, w, h in faces:
            cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 255), 4)
            face = img[y:y+h, x:x+w, :]
            face = cv2.resize(face, (50, 50))
            #face = pca.transform(face)
            face = face.reshape(-1, 3)
            pred = svm.predict(face)
            n = names[int(pred)][0]
            cv2.putText(img, n, (x, y), font, 1, (255, 255, 255), 3)
```

Figure: 16

Step:11

```
accuracy_score(y_test, y_pred)

0.76

haar_data = cv2.CascadeClassifier("data.xml")
capture = cv2.VideoCapture(0) # to open camera or if want to collect data from video then mention path of the video like (video.mp4)
data = []
font = cv2.FONT_HERSHEY_COMPLEX

while True:
    flag, img = capture.read()
    if flag:
        faces = haar_data.detectMultiScale(img)
        for x,y,w,h in faces:
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,255),4)
            face = img[y:y+h, x:x+w, :]
            face = cv2.resize(face, (50,50))
            #face = pca.transform(face)
            face = face.reshape(-1, 3)
            pred = svm.predict(face)
            n = names[int(pred)][0]
            cv2.putText(img, n, (x,y), font, 1, (244,233,250), 2)
            print(n)

        cv2.imshow('musheer',img)
        if cv2.waitKey(2) == 27:
            break
    capture.release()
cv2.destroyAllWindows()
```

Figure:17

Conclusion:

The computational models, which were implemented in this project, were chosen after extensive research, and the successful testing results confirm that the choices made by the researcher were reliable. The system with manual face detection and automatic face recognition did not have recognition accuracy over 90%, due to the limited number of eigen faces that were used for the PCA transform. This system was tested under very robust conditions in this experimental study and it is envisaged that real-world performance will be far more accurate.

The fully automated frontal view face detection system displayed virtually perfect accuracy and in the researcher's opinion further work need not be conducted in this area. The fully automated face detection and recognition system was not robust enough to achieve a high recognition accuracy. The only reason for this was the face recognition subsystem did not display even a slight degree of invariance to scale, rotation or shift errors of the segmented face image. This was one of the system requirements identified in section However, if some sort of further processing, such as an eye detection technique, was implemented to further normalize the segmented face image, performance will increase to levels comparable to the manual face detection and recognition system.

There are better techniques such as iris or retina recognition and face recognition using the thermal spectrum for user access and user verification applications since this need a very high degree of accuracy. The real-time automated pose invariant face detection and recognition system would be ideal for crowd surveillance applications. The implemented fully automated face detection and recognition system (with an eye detection system) could be used for simple surveillance applications such as ATM user security, while the implemented manual face detection and automated recognition system is ideal of mug shot matching., were we obtained in this study, which was conducted under adverse conditions. Implementing an eye detection technique would be a minor extension to the implemented system and would not require a great deal of additional research. All other implemented systems displayed commendable results and reflect well on the deformable template and Principal Component Analysis strategies.

References:

- https://www.google.com/search?q=software+used+for+face+mask+detection&rlz=1C1CHBF_enIN957IN957&oq=&aqs=chrome.3.69i59i450l8.3634112j0j7&sourceid=chrome&ie=UTF-8
- https://www.google.com/search?q=haar+cascade&rlz=1C1CHBF_enIN957IN957&oq=&aqs=chrome.7.69i59i450l8.3672408j0j7&sourceid=chrome&ie=UTF-8
- https://www.google.com/search?q=flow+chart+for+face+mask+detection&rlz=1C1CHBF_enIN957IN957&oq=&aqs=chrome.6.69i59i450l8.3691592j0j7&sourceid=chrome&ie=UTF-8
- https://www.google.com/search?q=face+mask+detection+system+project&rlz=1C1CHBF_enIN957IN957&oq=&aqs=chrome.6.69i59i450l8.3722304j0j7&sourceid=chrome&ie=UTF-8