" FACE MASK DETECTION SYSTEM "

PROJECT REPORT

DELHI TECHNOLOGICAL

UNIVERSITY

SUBJECT : MODELLING AND SIMULATION

3rd SEM PROJECT

DONE BY :   **SUBMIT TO:**

UTKARSH(2K19/IT/136)   MR. ANKIT YADAV

 VIKAS BHANDARI (2K19/IT/142)

<u>CERTIFICATE</u>

This is to certify to that Mr. *VIKAS BHANDARI (2K19/IT/142)* and Mr. *UTKARSH(2K19/IT/36)* of Batch IT-2 of Delhi Technological University have done their project on   FACE-MASK DETECTION SYSTEM under the guidance of teacher Mr. Ankit yadav of course " Modelling and Simulation " during Academic year 2020-2021 ,3rd semester.


  (Signature of Teacher)

INTRODUCTION

AIM OF PROJECT : The aim of the project is to find out whether a person is wearing a mask or not using live image or live video of the person using deep learning concept i.e CNN architecture.

This project uses the concept of Convulational Neural Network (CNN) of Deep Learning for training the model by using image dataset to determine whether a person is wearing mask or not .

HERE ARE SOME SCREENSHOTS OF THE PROJECT IMPLEMENTATION:
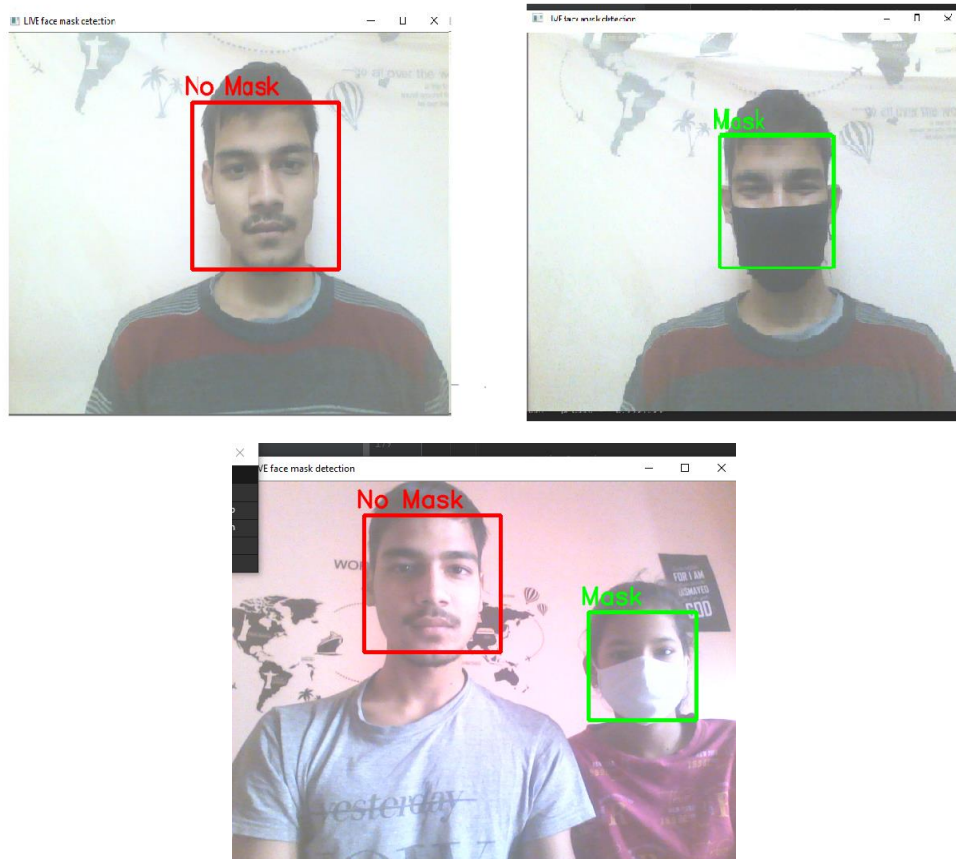
- **PROJECT IMPLEMENTATION IS BASICALLY IN 4 PARTS :**

1. Preparing image dataset to train images
2. Preparing a trained model using CNN
3. To get live image or video using opencv library
4. To detect face in a video or image

- Platform Used for Model Training : Google Colab
- For normal operation IDE used is : Pycharm

So let's understand how this project was implemented step-wise :
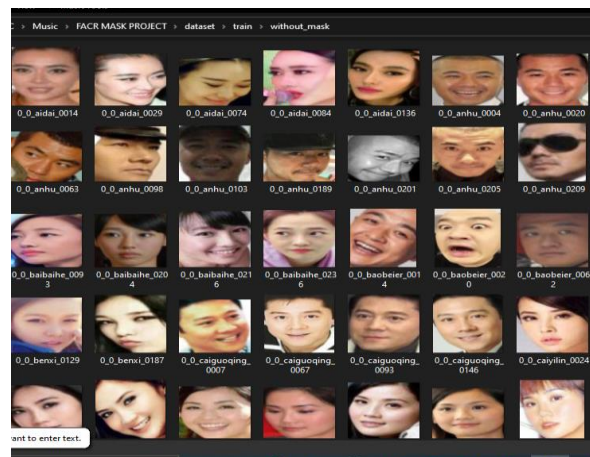
STEP – 1 DATASET – PREPARATION

The image dataset is divided into three sub-data :

1. Training image dataset
- With mask
- Without mask
2. Validation Image dataset
- With mask
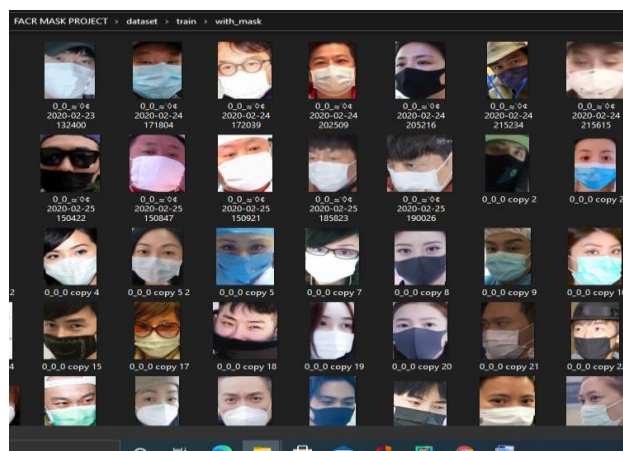- Without mask
3. Test image dataset

FOR EXAMPLE :

- <u>Let's take train dataset , it has two sub-directories containing image data as given below</u>
<u>.</u>

### "Without Mask "directory

"With Mask" directory:

- Similarly valid dataset also has two directories with mask and without mask .
- Test dataset has random image dataset to test our model .

STEP –2 MODEL TRAINING

The most important aspect of the project is application of CNN network of deep learning. So , here is a little info about convolutional neural network.

Convolutional Neural Network Basic Idea :

Convolutional Neural Network (CNN) is a deep learning network used for classifying images. The basic premise behind CNN is using predefined convolving filters to identify patterns in image edges, parts of objects and the build on to this knowledge to detect complete objects like animals, human being, automobiles etc.
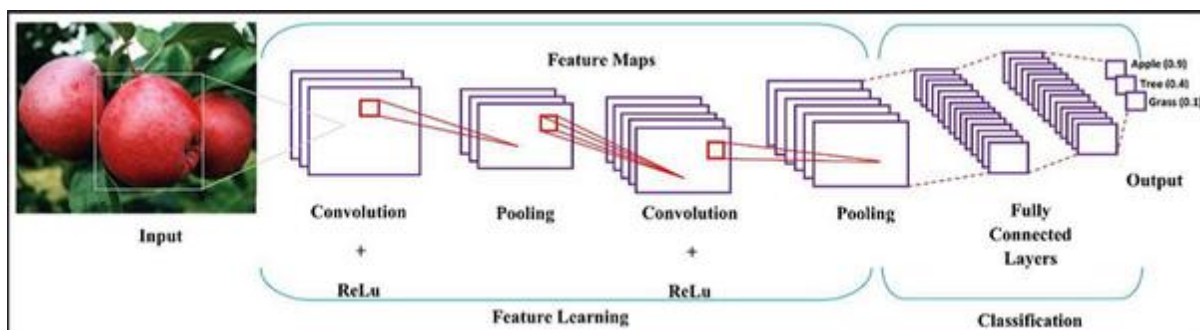
Input Layer : This layer is responsible for resizing input image to a fixed size and normalize pixel intensity values.

Convolution Layer: Image convolution is process of convolving a small 3x5, 5x5 matrix called kernel with image and subsample the image. This is used for extracting a specific features like edges in the image. There are several such filters used for specific purpose.

Pooling Layer: Pooling layers perform max or average operation on output of convolution layers to retain most significant information about patterns and forget insignificant ones

Fully Connected Layer: These are final logistic regression layers which maps visual features to desired output functions

Output Layer: This layer contains probabilities of classes of for each image imput. For instance if the image is cat. , Outout would be [0.85, 0.15] where 0.85 represents probability of image being cat and 0.15 represents probability of being a dog.

THESE WERE THE STEPS WE FOLLOWED IN TRAINING OUR MODEL :

Step 1. Import All necessary modules

2. Create object of imagedata generator

3. Data augmentation of train & valid dataset

4. Define path where model is to be saved

5. Create object of Model checkpoint

6. Define CNN architecture

7. Compile the Model

8. Train the Model

9. Save the best valid-accuracy using checkpoint

10. Save the final model.

Few Screenshots of model training :


```
Successfully uninstalled keras-2.4.3
Successfully installed keras-2.3.1
```

```
[ ]  """# Training model"""

     #import libraries
     import keras
     from keras.preprocessing.image import ImageDataGenerator
     from keras.optimizers import Adam
     from keras.callbacks import ModelCheckpoint
```

```
Using TensorFlow backend.
```

```
[ ]  import matplotlib.pyplot as plt

     import tensorflow

     print(keras.__version__)
     print(tensorflow.__version__)
```

```
2.3.1
2.1.0
```

```
[ ]  train_data_path = "/content/drive/My Drive/FACE DETECTOR PROJECT/dataset/train"
     validation_data_path = "/content/drive/My Drive/FACE DETECTOR PROJECT/dataset/valid"
```

```
[ ]  # this is the augmentation configuration we will use for training
     # It generate more images using below parameters
     training_datagen = ImageDataGenerator(rescale=1./255,
                                           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')
```

```
]
    # save best model using vall accuracy
    model_path = '/content/drive/My Drive/FACE DETECTOR PROJECT/Model/trainedmodel.h5'
    checkpoint = ModelCheckpoint(model_path, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')
    callbacks_list = [checkpoint]
```

```
]   #Building cnn model
    cnn_model = keras.models.Sequential([
                                keras.layers.Conv2D(filters=32, kernel_size=5, input_shape=[200, 200, 3]),
                                keras.layers.MaxPooling2D(pool_size=(4,4)),
                                keras.layers.Conv2D(filters=64, kernel_size=4),
                                keras.layers.MaxPooling2D(pool_size=(3,3)),
                                keras.layers.Conv2D(filters=128, kernel_size=3),
                                keras.layers.MaxPooling2D(pool_size=(2,2)),
                                keras.layers.Conv2D(filters=256, kernel_size=2),
                                keras.layers.MaxPooling2D(pool_size=(2,2)),

                                keras.layers.Dropout(0.5),
                                keras.layers.Flatten(), # neural network beulding
                                keras.layers.Dense(units=128, activation='relu'), # input layers
                                keras.layers.Dropout(0.1),
                                keras.layers.Dense(units=256, activation='relu'),
                                keras.layers.Dropout(0.25),
                                keras.layers.Dense(units=2, activation='softmax') # output layer

    ])
```

```
]   # compile cnn model
    cnn_model.compile(optimizer = Adam(lr=0.0001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    #cnn_model.compile(optimizer = Adam(lr=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
```

```
]   #Continuation :
    history = cnn_model.fit(training_data,
                            steps_per_epoch=20,
                            epochs=5,
                            verbose=1,
                            validation_data= valid_data,
                            callbacks=callbacks_list) # time start 14.25

    cnn_model.save('/content/drive/My Drive/FACE DETECTOR PROJECT/Model/modelmaskface.h5')

    Epoch 1/5
     4/20 [=====>......................] - ETA: 31s - loss: 0.3454 - accuracy: 0.8574/usr/local/lib/python3.6/dist-packages/PIL/Image.p
       "Palette images with Transparency expressed in bytes should be "
    20/20 [==============================] - 32s 2s/step - loss: 0.3159 - accuracy: 0.8746 - val_loss: 0.1975 - val_accuracy: 0.9169

    Epoch 00001: val_accuracy improved from 0.90943 to 0.91687, saving model to /content/drive/My Drive/FACE DETECTOR PROJECT/Model/train
    Epoch 2/5
    20/20 [==============================] - 30s 2s/step - loss: 0.3163 - accuracy: 0.8791 - val_loss: 0.1332 - val_accuracy: 0.9293

    Epoch 00002: val_accuracy improved from 0.91687 to 0.92928, saving model to /content/drive/My Drive/FACE DETECTOR PROJECT/Model/train
    Epoch 3/5
    20/20 [==============================] - 30s 2s/step - loss: 0.3233 - accuracy: 0.8654 - val_loss: 0.2137 - val_accuracy: 0.9293
```

Here is the link for model training full code
:https://colab.research.google.com/drive/1rNupk-
EuaY00_bR94azLpp0JnBit0Htl?usp=sharing

STEP- 3 USING OPENCV LIBRARY TO GET LIVE VIDEO

IDE USED: PYCHARM

OpenCV is a vast library that helps in providing various functions for image and video operations. With OpenCV, we can capture a video from the camera. It lets you create a video capture object which is helpful to capture videos through webcam and then you may perform desired operations on that video.

Steps to capture a video:

- Use `cv2.VideoCapture()` to get a video capture object for the camera.
- Set up an infinite while loop and use the `read()` method to read the frames using the above created object.
- Use `cv2.imshow()` method to show the frames in the video.
- Breaks the loop when the user clicks a specific key.

Here is an example of the code :

```python
# import the opencv library
import cv2


# define a video capture object
vid = cv2.VideoCapture(0)

while(True):

    # Capture the video frame
    # by frame
    ret, frame = vid.read()

    # Display the resulting frame
    cv2.imshow('frame', frame)

    # the 'q' button is set as the
    # quitting button you may use any
    # desired button of your choice
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# After the loop release the cap object
vid.release()
# Destroy all the windows
cv2.destroyAllWindows()
```

- We have used this concept to take live video as input.

KEY POINT:

- We know that our model works in an image dataset so we have to understand how we are getting this live video . So let's understand:

The live video we are getting is nothing but images which are continuosly changing at high rate using while loop as you can see in the above code also . So we will have to apply our training model in image at every iteration and show result in some form for example showing red circle or rectangle in case of not wearing mask etc. in each iteration.

STEP – 4 DETECTING FACE IN A VIDEO OR IMAGE

To detect face in an image we used CascadeClassifier method of opencv module.

What are Haar Cascades?
Haar Cascade classifiers are an effective way for object detection. This method was proposed by Paul Viola and Michael Jones in their paper . Rapid Object Detection using a Boosted Cascade of Simple Features .Haar Cascade is a machine learning-based approach where a lot of positive and negative images are used to train the classifier.

As you can see in the above image ,the rectangle shows the face of the person.So int this way we can detect face .Also we can change the color of the rectangle and add certain text also to the image which can be seen in our project also.

Haarcascade is a different topic so we will not go deep into this topic :

Final Step:

Now since we have detected face using Haar Cascade method we will save that image of the face in a directory and then use it to implement our trained model and according to the predicted result we will show result as shown below.

- Since we already know that video is nothing but images changing at high rate so we can apply this haarcascade method in each iteration considering in iteration we only get image and then repeat the same steps.

!!! Hence ,we did this project in this way. !!!

- **HERE IS A SIMPLE ALGORITHM WE APPLIED IN OUR PROJECT OF THE IMPLEMENTATION OF OPENCV AND HAARCASCADE IN PYCHARM :**

```
Import all necessary libraries

Import trained model

a = 1
    while a == 1:
            b = input ("Enter Mode 1= live image, 2 = live video")
            If b == 1:
                    Use opencv method to get live image
                    Detect no. of faces in image.
                    for each face:
                            apply trained model
                            show result.
                            save result/output

            else:
                    while loop:          Here we used while loop to get live
                            Use opencv method to get live image   image continuously so it becomes a video
                            Detect no. of faces
                            for each face:
                                    apple trained model
                                    show result.
                                    Don't save output.
                            exit condition

            a = input ("Enter 1 = Run Program Again / 2 = Exit")

    print ("Thank You for using face Mask Detection System")
```
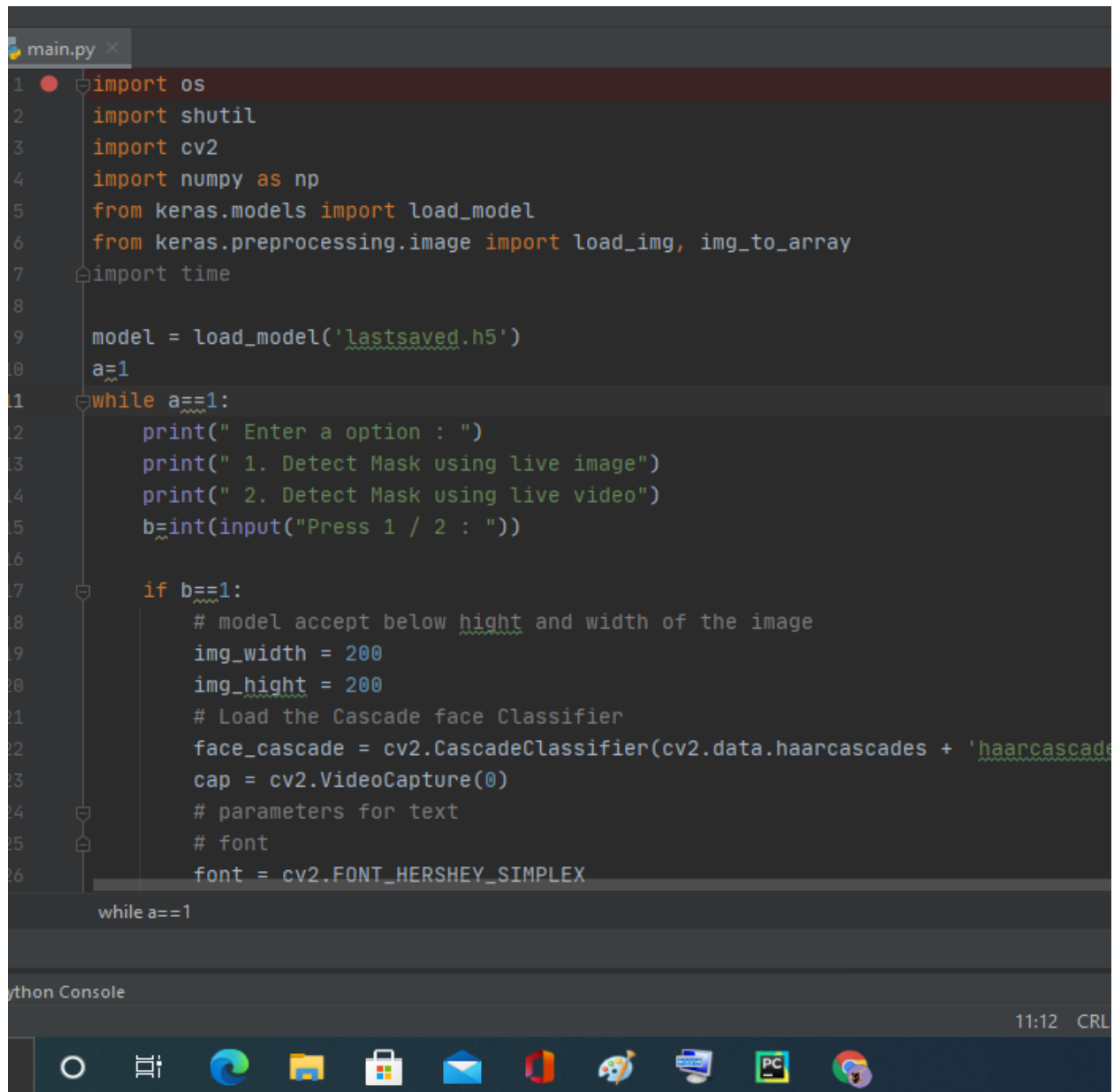
Here is the link for full code of this method :

HERE IS A SCREENSHOT OF THE IMPLEMENTATION :

```python
import os
import shutil
import cv2
import numpy as np
from keras.models import load_model
from keras.preprocessing.image import load_img, img_to_array
import time

model = load_model('lastsaved.h5')
a=1
while a==1:
    print(" Enter a option : ")
    print(" 1. Detect Mask using live image")
    print(" 2. Detect Mask using live video")
    b=int(input("Press 1 / 2 : "))

    if b==1:
        # model accept below hight and width of the image
        img_width = 200
        img_hight = 200
        # Load the Cascade face Classifier
        face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade
        cap = cv2.VideoCapture(0)
        # parameters for text
        # font
        font = cv2.FONT_HERSHEY_SIMPLEX
```

while a==1

ython Console

11:12   CRL

FURTHER USES OF THIS PROJECT :

- Can be used in mall ,cinemas or any public places to check for mask during entry.

- Since this project saves images of the person taking test so it can be used as a entry point to high security places which will allow a persons's entry only if he's following the rules and the data will be saved to check later also for further use.

- During pandemics like corona , it is a really good application to check whether people are wearing mask or not using computer .

CONCLUSION :

The project works with good accuracy and shows the required output :
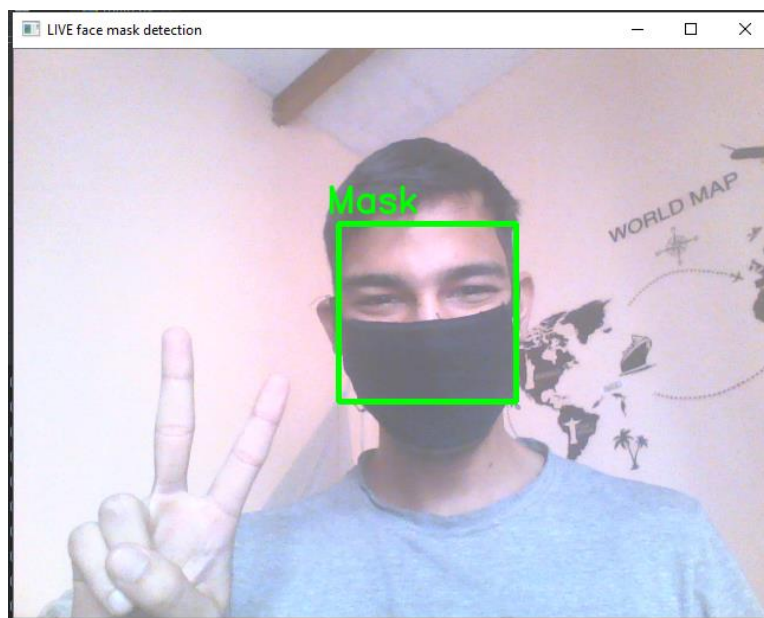
- **Here is the link for full project file :**

https://drive.google.com/drive/folders/1EHTBYHI7C9FS4ahCOTqGXT4vJMtn_85X?usp=sharing

BIBLIOGRAPHY :

- For open cv :

https://www.geeksforgeeks.org/python-opencv-capture-video-from-camera/

- For face detection

https://www.geeksforgeeks.org/python-haar-cascades-for-object-detection/

- For removing and dealing with errors

    Stackoverflow,gfg and many other sources !