VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANA SANGAMA, BELAGAVI 590018

An Internship Report

“FACE MASK DETECTION MODEL”

Submitted
by

PUSHKAR JHA-1CR21ISI22

Internship carried out
at
COE-MIBD

ABSTRACT:
In current times after the rapid expansion and spread of the COVID-19 outbreak globally, people have experienced severe disruption to their daily lives. One idea to manage the outbreak is to enforce people wear a face mask in public places. Therefore, automated and efficient face mask detection methods are essential for such enforcement. In this paper, a face mask detection model for static and real time videos has been presented which classifies the image as “yes mask” and “no mask” using deep learning algorithms. The model is trained and evaluated using a dataset from GitHub. The gathered data-set comprises approximately about 1,500 picture (750 with mask and 750 without mask) and attained a performance accuracy rate of 90%. Various Python libraries such as OpenCV, Scikit-learn, Keras and TensorFlow. The VGG16 architecture based on convolutional neural network is used as detection and classification algorithm. This system will help public service providers ensure a safe environment.

CONTENTS:

CHAPTER 1: Introduction

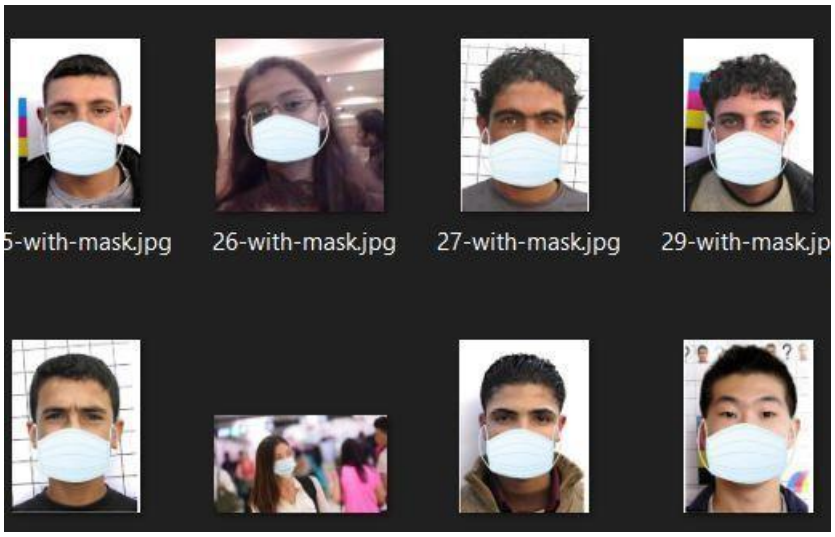
- Problem Statement – To develop a face mask detection machine learning/ deep learning model for face mask detection.

- Objective –
  - To prevent the spread of Coronavirus by using effective face mask detection technology.
  - To ensure that the public places have a safe environment.
  - To make sure nobody is breaking mask mandatory rules.

- Scope of project – To prevent the spread of Coronavirus or any other medical and social situation such that wearing a mask is in welfare of society this project has made to develop a real-time face mask detection through webcam. A Coronavirus (COVID-19) report by the World Health Organization (WHO) reveals that, as of 22 November 2021, there were 258 million confirmed cases of COVID-19 cases and 5,148,221 deaths worldwide. Therefore, people should wear face masks and keep a social distance to avoid viral spread of disease. An effective and efficient computer vision strategy intends to develop a real-time application that monitors individuals publicly, whether they are wearing face masks or not. This system uses Machine Learning and Artificial Intelligence with image processing. This project can detect face mask into two classes. First, people wearing a face mask and another one is people without a mask. If people are wearing a face mask, the system will show a text message “yes mask” with green signal. If people are not wearing a face-mask then the system will show “no mask” with red signal.

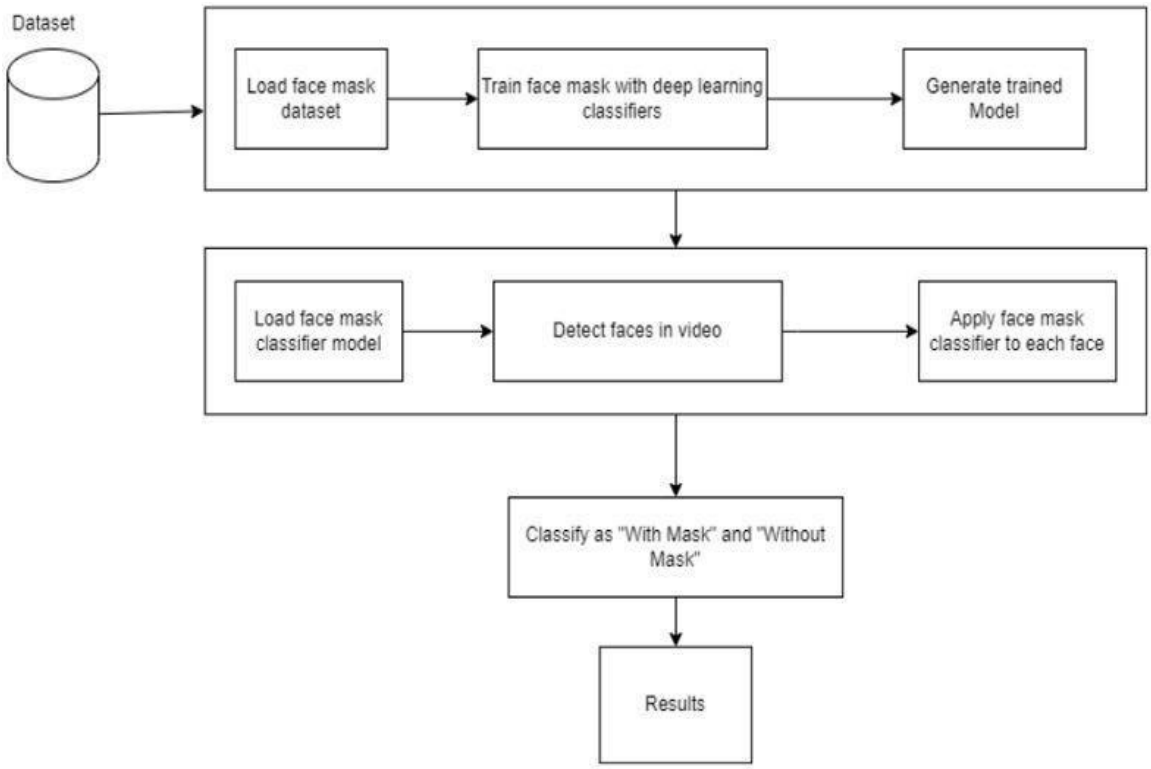
CHAPTER 2: System requirement and design.

- Dataset –
  - With mask



  - Without mask

- Flow (Block Diagram) –



- Related technology for designing the system –

  - Python - Python is the most common language for machine learning and deep learning. For a long time, python has been preferred programming language for machine learning and deep learning researchers. Python provides developers with some of the most flexible and feature-rich tools that improve not only their productivity but also the quality of their code. It is one of the most developer-friendly programming languages, with a diverse set of libraries to serve every use case or projects.
  - OpenCV – It is a large open-source library for image processing, machine learning, and computer vision. Python, C++, java, and other programming languages are among the languages supported by OpenCV.
    It can recognize objects, faces, and even human writing by analyzing efficient library for numericaloperations. One of OpenCV's goals is to provide a simple-to-use computer vision infrastructure that allows individuals to quickly create rather complex vision applications. Over 500 functions in the OpenCV library cover a wide range of vision topics, including factory product inspection, medical imaging,

security, user interface, camera calibration, stereo vision, and robotics. Because computer vision and machine learning are frequently used together, OpenCV also includes a comprehensive machine learning library.

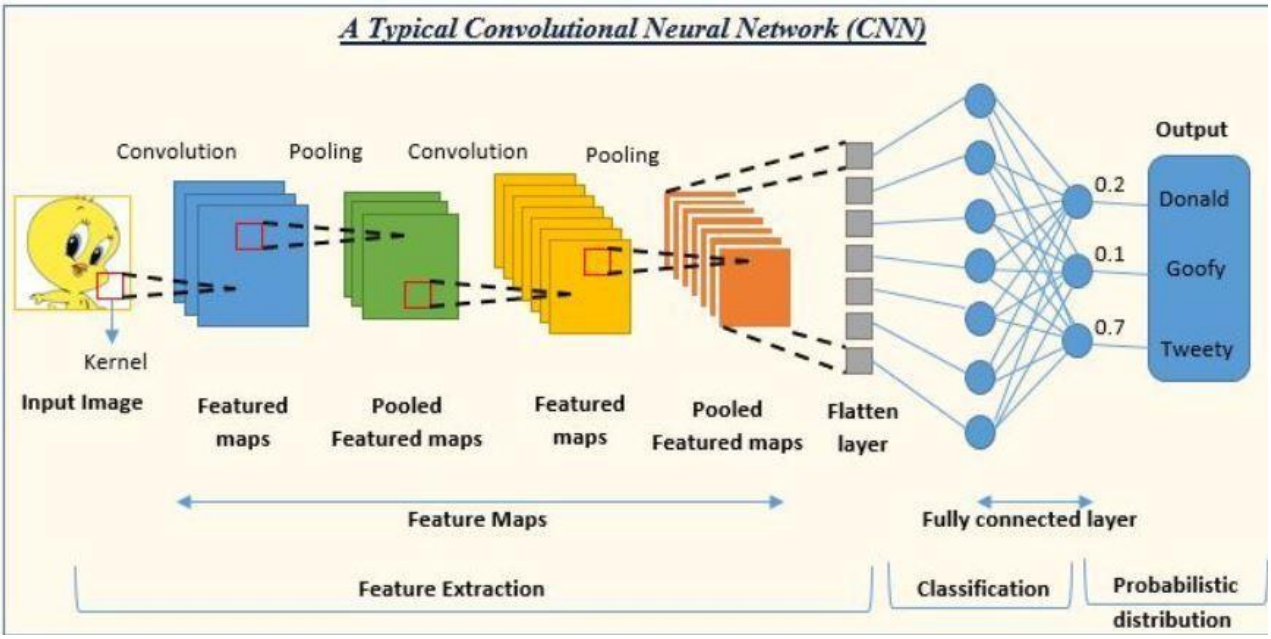
- TensorFlow -TensorFlow is an open-source machine learning software that focuses on deep neural networks from start to finish. TensorFlow is a collection of libraries, tools, and community resources that are diverse and comprehensive. It enables programmers to construct and deploy cutting-edge machine learning-based applications. The Google Brain team first designed the TensorFlow python deep learning library for internal usage. The open-source platform's application in R&D and manufacturing systems has increased since then. There are a few key principles in TensorFlow. Tensors are TensorFlow's fundamental building blocks. In the TensorFlow python deep-learning framework, a tensor is an array that represents many forms of data. Unlike a one-dimensional vector or array or a two-dimensional matrix, a tensor can have n dimensions.
- Keras - Google developed Keras, a high-level deep learning API for building neural networks. It is written in python and helps with neural network development. It is modular, quick, and simple to use. It was created by Google developer Francois Chollet. Low-level computation is not handled by Keras. Instead, it makes use of a library known as the "Backend". Keras provides the ability to swap between different back ends. TensorFlow, Theano, PlaidML, MXNet, and CNTK (Microsoft Cognitive Toolkit) are among the frameworks support by Keras. TensorFlow is the only one of these five frameworks that has accepted Keras as its official high-level API. Keras is a deep learning framework that is built on top of TensorFlow and has built-in modules for all neural network computations. Simultaneously, the TensorFlow core API may be used to build custom computations with tensors, computation graphs, sessions, and so on. It gives users complete control and flexibility over their applications, as well as the ability to quickly implement ideas.

- Scikit-learn - Formerly scikits.learn and also known as sklearn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

CHAPTER 3: System Implementation

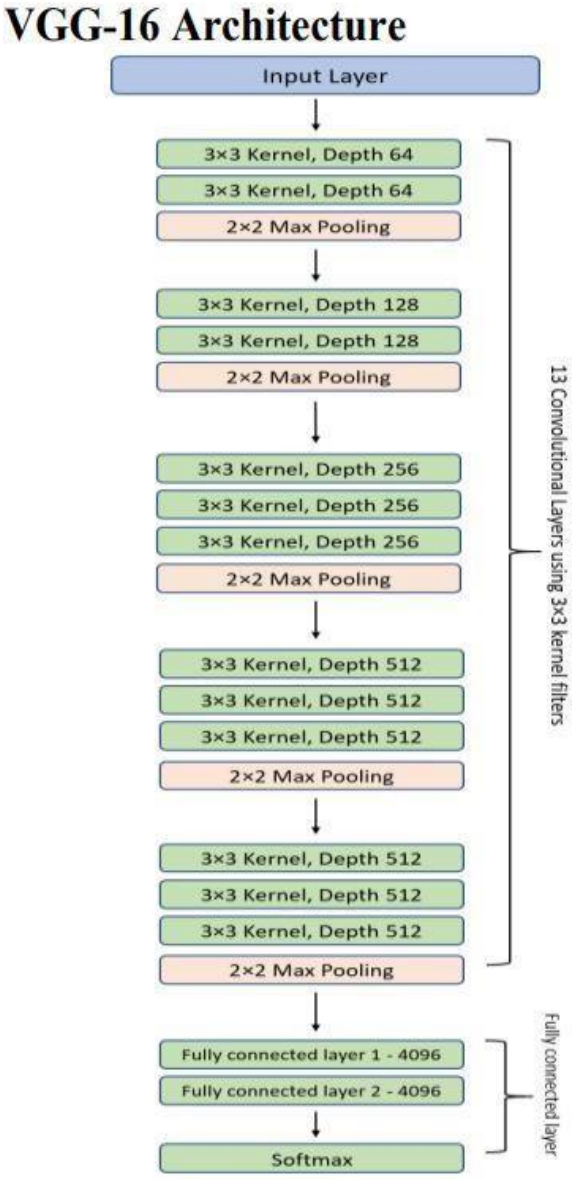
- Proposed Approach - Firstly, use a Convolutional neural network, in which the input pictures are processed through a succession of layers, including convolutional, pooling, flattening, and fully connected layers, and then the output of CNN is formed, which classifies pictures. As a result, use one of the pre-trained models VGG-16 to categorize images and assess accuracy for training and validation data.

  - Convolutional Neural Network - A Convolutional neural network is a sort of artificial neural network that employs multiple perceptron to evaluate picture inputs and have learnable weights and bases to several sections of pictures that may separate each other. The usage of Convolutional Neural Networks has the benefit of leveraging the usage of local spatial coherence in the input pictures, which allows them to have fewer weights because certain parameters are shared. In terms of memory and complexity, this procedure is definitely efficient.



The higher performance of convolutional neural networks with picture, speech, or audio signals inputs distinguishes them from other neural networks. Convolutional layer, pooling layer, and fully connected layer are the main three basic types of layers available. A convolutional network's first layer is the convolutional layer. While further convolutional layers or pooling layers can be added after convolutional layers, the fully connected layer is the last layer. The CNN becomes more complicated with each layer, detecting larger areas of the image. Earlier layers concentrate on basic elements like colors and borders. As the visual data travels through the CNN layers, it begins to distinguish larger elements or features of the item, eventually identifying the target object.

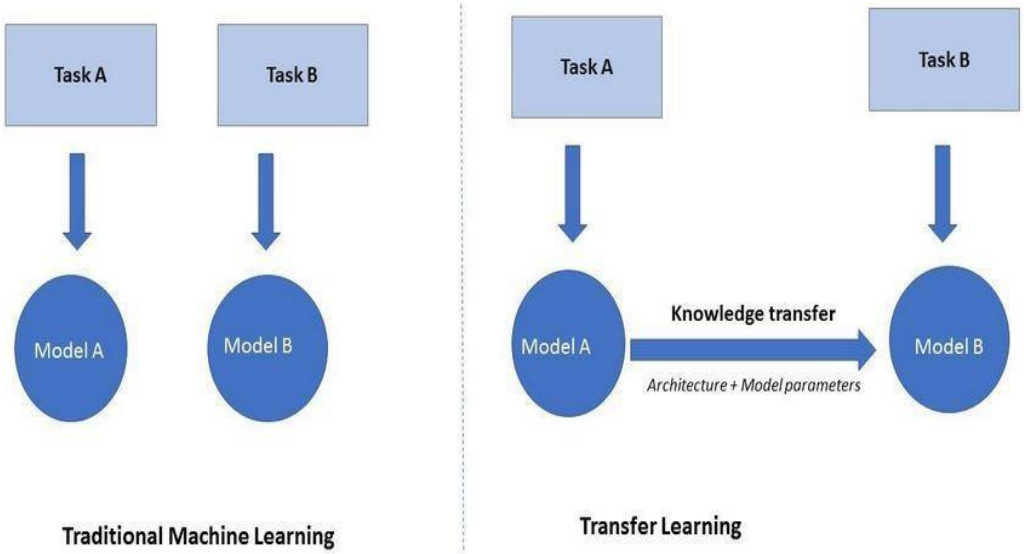
- VGG-16 Architecture – VGG owes its name to the Visual Geometry Group of Oxford University. After being submitted to ILSVRC in 2014[1], the model VGG-Net became as popular as the group itself. VGG-Net is the CNN architecture developed at Oxford university by Karen Simonyan, Andrew Zisserman, and others. VGG-Net is a 16-layer CNN that has been trained on over one billion images and has up to 95 million parameters (1000 classes). It has 4096 convolutional features and can handle 224 by 224 pixels input images.



The precise structure of the VGG-16 network shown in figure is as follows: The first and second convolutional layers are made up of 64 feature kernel filters with a filter size of 3x3. The dimensions of the input picture (RGB picture with depth 3) change

to 224x224x64 as it passes through the first and second convolutional layers. The output is then sent to the max pooling layer with a stride of 2. The third and fourth convolutional layers are made up of 124 feature kernel filters with a filter size of 3x3. Following these two layers is a max pooling layer with stride 2, and the resultant output is 56x56x128. Convolutional layers with kernel size 3x3 are used in the fifth, sixth, and seventh levels. All three make use of 256 feature maps. Following these layers is a max pooling layer with stride 2. Eighth to thirteenth are two groups of convolutional layers with kernel size 3x3. All of these convolutional layer sets contain 512 kernel filters. Following these layers is a max pooling layer with a stride of 1. Fourteen and fifteen levels are completely linked hidden layers of 4096 units, followed by a soft max output layer (sixteenth layer) of 1000 units.

- Transfer Learning - The proposed model use VGG16 architecture and transfer learning, which is a process of applying the knowledge of weights and layers from a current model to new untrained model in order to accelerate the learning of a new model.



In proposed model existing VGG16 is used for transfer learning based face mask detection and proposed model conducts train and validation on face mask detection custom data set with varying facial image position, lighting, and size. The images on the data set contain hundreds faces, hundreds of images with mask and without mask and also used weights of Imagenet that provide better results. VGG16 is a convolution based network model. VGG16 is a network model focused on convolutions and it is widely used in computer vision technologies. In proposed model for better classification, removed the fully connected layer top most layer of existing model and replaced this layer with the flatten, dense and dense softmax layer for better classification. To prevent over fitting, the drop out is used to drop certain values at random. For multi-classification of facial features in an image softmax layer is used. Except for the last layer, the ReLu activation mechanism has been used in all layers. The number of images used in the training phase 80% of the total number of images in the dataset, with the remainder used for confirmation.

- Source code with algorithm –
  - Inside the project path, several required libraries were installed. Packages like numpy, keras, opencv were imported after the installation. These packages take care of their own functions as needed within the application.

```
import os
from keras.preprocessing import image
import cv2
```

  - There is category. Within the category, there are two values: one with mask and the other without. An empty array data is also made. Through these two categories, a loop function has been applied After that, the for-loop function was used to loop through the categories. A link to the route with the directory and category has been generated. “os.lisdir” produces a list of all the images paths inside the loop. Following that, the cv2.imread function is utilized. The reading of image is handled by the imported library “cv2”. The image path has been loaded with imread, and the image size target is (224, 224) pixels and resizing is done using cv2.imread. The dimensions of an image are its width and height. By loading the images, they were saved into an array.

```
categories=['with_mask','without_mask']
data = []
for category in categories:

path=os.path.join('C:\\Users\\pushk\\face_mask_detector\\
train',category)

    label = categories.index(category)
    for file in os.listdir(path):

        img_path=os.path.join(path,file)
        img = cv2.imread(img_path)
        img = cv2.resize(img,(224,224))
        data.append([img,label])
```

  - We will use random.shuffle from random so that the model doesn’t develop a bias towards the data. After shuffling we will separate x and y and will convert them into numpy array.

```
import random
random.shuffle(data)

x = []
y = []

for features,label in data:
    x.append(features)
    y.append(label)

import numpy as np
x=np.array(x)
y=np.array(y)
x.shape
```

- Training and testing data were separated into trainX and testY, as well as trainY and testY. Within the classification process, there were two types of datasets: training and testing datasets. The training set was used to train a model, whereas the test set was used to test the model that have been trained. As shown 20% of the testing sets were used, while the remaining 80% was used for additional training. Because the datasets were separated into training and testing, data preparation was carried out. The "sklearn.model_selection import train_test_split" package was used to split the data into train and test.

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test =
train_test_split(x,y,test_size=0.2)
```

- Now we will do the transfer learning with vgg16. At first we will remove the last layer then we will freeze the layers so that during training parameters wouldn't update. After freezing we will ad our own layer and then we will do the compilation part.

```
from keras.applications.vgg16 import VGG16
vgg=VGG16()
vgg.summary()

#remove the last layer
from keras import Sequential
model = Sequential()
for layer in vgg.layers[:-1]:
    model.add(layer)
```

```
model.summary()

#now freeze the layers
for layer in model.layers:
    layer.trainable=False
model.summary()

#add the layer
from keras.layers import Dense
model.add(Dense(1,activation='sigmoid'))
model.summary()

#compilation
model.compile(optimizer='Adam',loss='binary_crossentropy'
,metrics=['accuracy'])
model.fit(x_train,y_train,epochs=2,validation_data=(x_tes
t,y_test))
```

- Now we will make functions named detect_face_mask, draw_label, and detect_face for detecting face mask, positioning text and detecting face respectively.

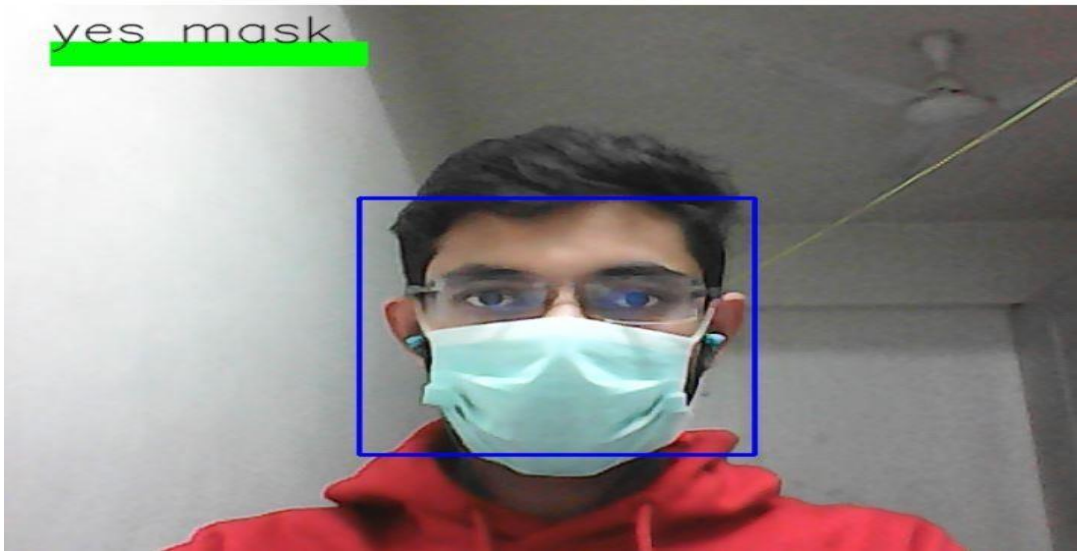
```
def detect_face_mask(img):
    y_pred = model.predict(img.reshape(1,224,224,3))
    y_pred = np.round(y_pred).astype(int)
    return y_pred[0][0]

def draw_label(img,text,pos,bg_color):

text_size=cv2.getTextSize(text,cv2.FONT_HERSHEY_SIMPLEX,1
,cv2.FILLED)
    end_x=pos[0] + text_size[0][0] + 2
    end_y=pos[0] + text_size[0][1] - 2


cv2.rectangle(img,pos,(end_x,end_y),bg_color,cv2.FILLED)

cv2.putText(img,text,pos,cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,
0),1,cv2.LINE_AA)

haar =
cv2.CascadeClassifier('C:\\Users\\pushk\\face_mask_detect
or\\haarcascade_frontalface_default.xml')
def detect_face(img):
    coods = haar.detectMultiScale(img)
    return coods
```

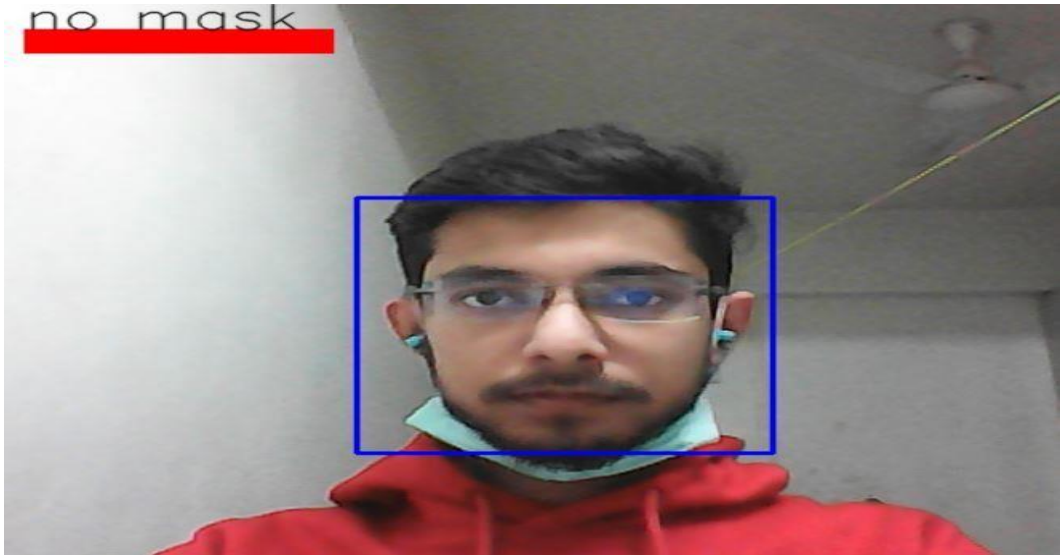
- Now comes the final stage where we will be calling our functions and we will connect the webcam with our model and will do the live face mask detection. cv2 is extensively used in this step.

```
cap=cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    img = cv2.resize(frame,(224,224))
    y_pred=detect_face_mask(img)

    coods =
detect_face(cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY))
    for x,y,w,h in coods:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)


    if y_pred==0:
        draw_label(frame,"yes mask ",(30,30),(0,255,0))
    else:
        draw_label(frame,"no mask ",(30,30),(0,0,255))

    cv2.imshow("window",frame)

    if cv2.waitKey(1) & 0xFF == ord('x'):
        break

cv2.destroyAllWindows()
```

CHAPTER 4 : Results

The machine ran a face detection algorithm. In a window named frame, a face and face has been detected. The detector categorized the face as “yes mask” and “no mask’ with an accuracy score of indicating that the face mask was identified. Face mask detection was performed in real time using facial images.Also with yes mask there is a green signal and with no mask there is a red signal.





CHAPTER 5 : Conclusion and future scope.

- Conclusion - The challenge of the face detection system is the non-frontal face position and the use of accessories that cover the face area. This paper presents the real-time face mask detection system that can perform face detection with better accuracy. This system detects human faces and also the presence or absence of facemask on it. The sensible outcomes are generated when the framework is tried on different postures of faces. This work illustrates the model's preliminary steps toward a high accuracy of the real-time face detection device. The goal is achieved by putting the concept into practice using cutting-edge technologies like opencv, machine learning, and deep learning. In future work, making the model more precise and gaining more accuracy and efficiency, model train on a larger dataset and also work on face key points and a better camera will be used for capturing face images in real-time, so the system can work on deem light also.

- Future Scope - Object detection is a technology that falls under the umbrella of computer vision. Objects in images and videos can be recognized and tracked using this technique. Object identification, also known as object detection, can be used for face recognition, vehicles recognition, self-driving vehicles, security systems, and a range of other applications. Object detection using deep learning and machine learning has become a study focus on recent years.
  Face detection can be used in several applications such as in surveillance, identification in the login system, and personalized technology.
  Masks are becoming increasingly trendy these days. In the absence of immunization, masks are one of the few ways to protect against the corona virus, and they play an important role in protecting people's health from respiratory illnesses. To ensure human safety, this project can be integrated with embedded technology and deployed in a range of public venues, such as airports, train stations, offices, schools, and publicspaces