Face Mask And Social Distancing Detection

Mini Project Report

Submitted by

Ayisha Nidha

Submitted in partial fulfillment of the requirements for the award of the degree of

Master of Computer Applications
Of
A P J Abdul Kalam Technological University

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)®
ANGAMALY-683577, ERNAKULAM(DIST)
FEBRUARY 2022

DECLARATION

I, **Ayisha Nidha** hereby declare that the report of this project work, submitted to the Department of Computer Applications, Federal Institute of Science and Technology (**FISAT**), Angamaly in partial fulfillment of the award of the degree of Master of Computer Application is an authentic record of our original work.

The report has not been submitted for the award of any degree of this university or any other university.

Date :

Place: Angamaly

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)®

ANGAMALY, ERNAKULAM-683577

DEPARTMENT OF COMPUTER APPLICATIONS

CERTIFICATE

This is to certify that the project report titled **"Face Mask and Social Distancing Detection "** sub- mitted by **Ayisha Nidha** towards partial fulfillment of the requirements for the award of the degree of Master of Computer Applications is a record of bonafide work carried out by them during the year 2022.

Project Guide **Head of the Department**

Submitted for the viva-voice held on . . . . . at . . . . . .

ACKNOWLEDGEMENT

I am deeply grateful to **Dr.Manoj George** , Principal, FISAT, Angamaly for providing us with adequate facilities, way and means by which we were able to complete our mini project work and I express my sincere thanks to **Dr. C. Sheela** ,Vice Principal FISAT, Angamaly.

My sincere thanks to **Dr.Deepa Mary Mathwes**, Head of the department of MCA, FISAT, who had been a source of inspiration. I express heartiest thanks to **Dr.Deepa Mary Mathews** our project guide for their encouragement and valuable suggestion . I express our heartiest gratitude to our scrum master **Ms.Rose Mary Mathew** and the faculty members in our department for their constant encouragement and never ending support throughout the project.I would also like to express our sincere gratitude to the lab faculty members for their guidance

Finally I express our thanks to all my friends who gave me wealth of suggestion for successful completion of this project.

ABSTRACT

Since the infectious coronavirus disease (COVID-19) was first reported in Wuhan, it has become a public around the world. This pandemic is having devastating effects on societies and economies around the world. The increase in the number of COVID-19 tests gives more information about the epidemic spread, which may lead to the possibility of surrounding it to prevent further infections. However, wearing a face mask that prevents the transmission of droplets in the air and maintaining an appropriate physical distance between people, and reducing close contact with each other can still be beneficial in combating this pandemic.

Monitoring social distancing regulations and manually checking people's masks is most likely to lead to scarcity of resources, and is supposed to allow errors creeping in due to human intervention. There is an urgent need to solve the virus transmission by studying the ideal social distancing rules. This system includes the detection of people violating social distance regulations and the prediction of masks. Citizens have ensured safety with strict rules checking whether sufficient distance and mask usage are observed. This piece of work discusses a lightweight system that is likely to help in the prevention of COVID-19 as it uses surveillance through video mode for identifying and confirming that the social distancing is maintained by everyone so that the spread can be reduced which further helps in the slowing down of the virus transmission.

This can be achieved by employing deep learning algorithms such as convolutional neural networks(CNN). Pre-trained models such as VGG, SSD-MobileNet for object detection, and CLAHE are used for face mask detection and social distancing detection respectively. This can be used with the existing embedded camera infrastructure to enable these analytics which can be applied to various verticals, as well as in an office building or at airport terminals/gates.

Contents

Chapter 1

Introduction

Since the end of 2019, infectious coronavrus disease (COVID-19) has been reported for the first time in Wuhan, and it has become a public damage fitness issue in China and even worldwide. This pandemic has devastating effects on societies and economies around the world causing a global health crisis [1]. It is an emerging respiratory infectious disease caused by Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2) [2]. All over the world, especially in the third wave, COVID-19 has been a significant healthcare challenge [3]. Many shutdowns in different industries have been caused by this pandemic. In addition, many sectors such as maintenance projects and infrastructure construction have not been suspended owing to their significant effect on people's routine.

To prevent rapid COVID-19 infection, many solutions, such as confinement and lockdowns, are suggested by the majority of the world's governments. Before coronavirus, some people put masks to protect themselves from air pollution, while other people put face masks to hide their faces and their emotions from others. Protection against coronavirus is a mandatory counter measure, according to the WHO . Indeed, wearing a mask is an effective method of blocking 80 of all respiratory infections . Also, the WHO recommends practising physical distancing to mitigate the spread of the virus. All over the world, governments are struggling against this type of virus. Many organizations enforce face mask rules for the personal protection. Checking manually if

individuals entering an organisation are wearing masks is cumbersome and possibly conflicting . This proposed deep learning-based model,aims to prevent human-to-human transmissions of the SARS-CoV-2 and detect faces with or without mask. Two different datasets (IDS1 and IDS2) with over 5200 images are used to train and test the model.

Chapter 2

PROOF OF CONCEPT

Coronavirus (COVID-19) pandemic locks down the world and causes a global recession. This virus spreads mostly because of the droplets produced by sneezing and coughing. Regular hand washing, wearing a face mask, and maintaining the social distance in public places can be preventive measures to reduce the spreading of the virus.

To curb certain respiratory viral ailments, including COVID-19, wearing a clinical mask is very necessary. The public should be aware of whether to put on the mask for source control or aversion of COVID-19. Potential points of interest of the utilization of masks lie in reducing vulnerability of risk from a noxious individual during the "pre-symptomatic" period and stigmatization of discrete persons putting on masks to restraint the spread of virus. WHO stresses on prioritizing medical masks and respirators for health care assistants[4]. Therefore, face mask detection has become a crucial task in present global society..

Here a deep learning model which can predict whether the face mask is ON/OFF a persons face is developed.The model also detects whether social distancing is properly maintained between people. This model can be employed in shopping malls, markets etc were there shall be a group of people. Manualy checking individuals entering an organisation are wearing masks is cumbersome and possibly conflicting .

Chapter 3

IMPLEMENTATION

The primary step of this model is to design a deeplearning CNN model .The proposed method for face mask detection consists of a cascade classifier and a pre-trained CNN which contains two 2D convolution layers connected to layers of dense neurons.

3.1 Architecture

3.1.1 Facemask

The proposed method consists of a cascade classifier and a pre-trained CNN which contains two 2D convolution layers connected to layers of dense neurons.CNN has become ascendant in miscellaneous computer vision tasks [12]. The current method makes use of Sequential CNN.

The First Convolution layer is followed by Rectified Linear Unit (ReLU) and MaxPooling layers. The Convolution layer learns from 200 filters. Kernel size is set to 3 x 3 which specifies the height and width of the 2D convolution window. As the model should be aware of the shape of the input expected, the first layer in the model needs to be provided with information about input shape. Following layers can perform instinctive shape reckoning [13]. In this case, *input_shape* is specified as *data.shape[1:]* which returns the dimensions of the data array from index 1. Default padding is "valid" where the spatial dimensions are sanctioned to truncate and the input volume is non-zero padded. The activation parameter to the Conv2D class is set as "relu". It represents an approximately linear function that possesses all the assets of linear models that can easily be optimized with gradient-descent methods. Considering the performance and generalization in deep learning, it is better compared to other activation functions [14]. Max Pooling is used to reduce the spatial dimensions of the output volume. Pool_size is set to 3 x 3 and

the resulting output has a shape (number of rows or columns) of: shape_of_output = (input_shape – pool_size + 1) / strides), where strides has default value (1,1) [15].

As shown in fig, 4, the second Convolution layer has 100 filters and Kernel size is set to 3 x 3. It is followed by ReLu and MaxPooling layers. To insert the data into CNN, the long vector of input is passed through a Flatten layer which transforms matrix of features into a vector that can be fed into a fully connected neural network classifier. To reduce overfitting a Dropout layer with a 50% chance of setting inputs to zero is added to the model. Then a Dense layer of 64 neurons with a ReLu activation function is added. The final layer (Dense) with two outputs for two categories uses the Softmax activation function.

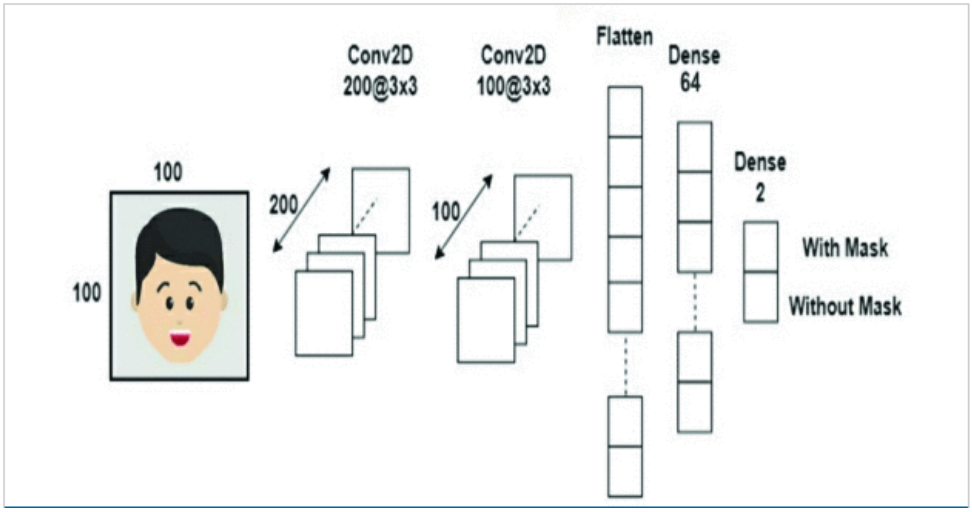


Figure 3.1.1.1 Convolutional Neural Network Architecture

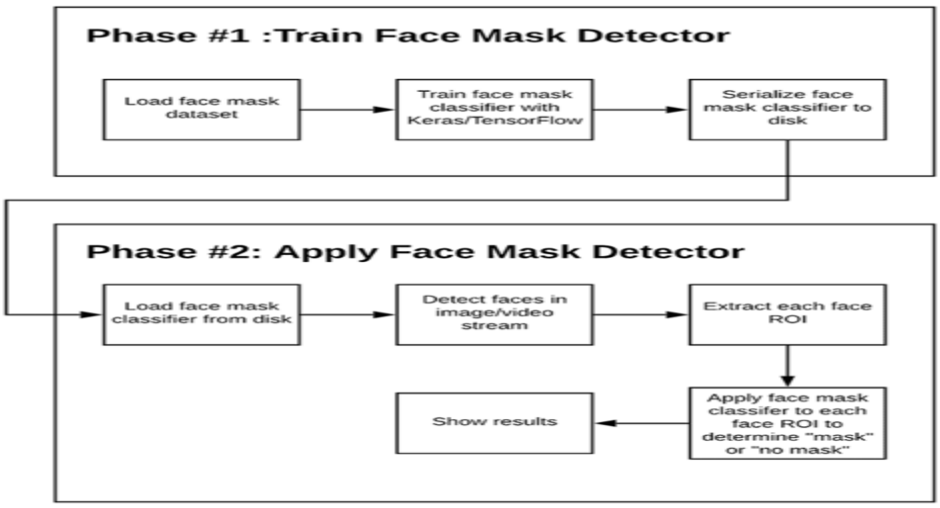


Figure 3.1.1.2: Architecture

3.1.2 Social Distancing

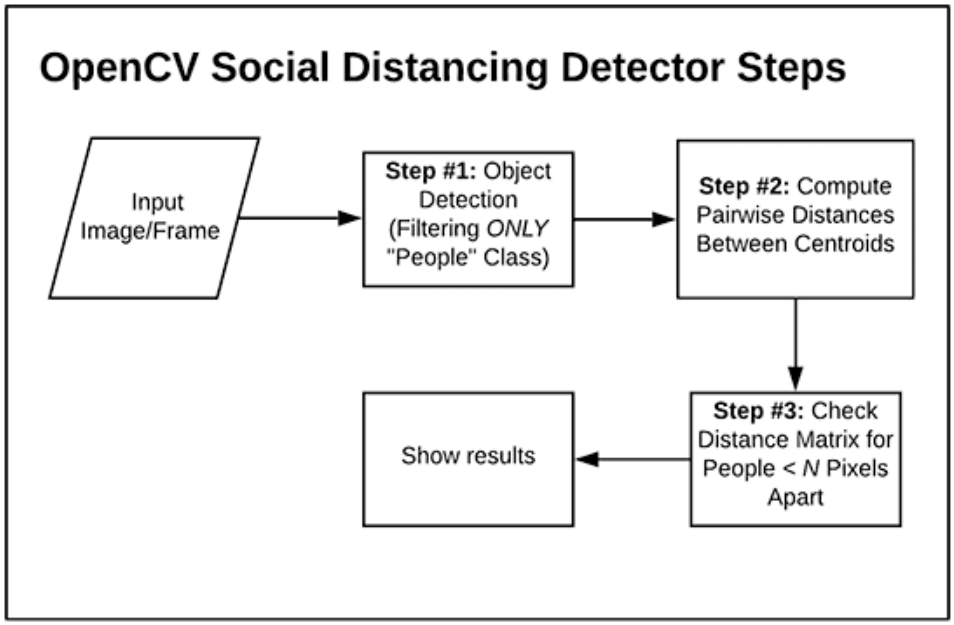


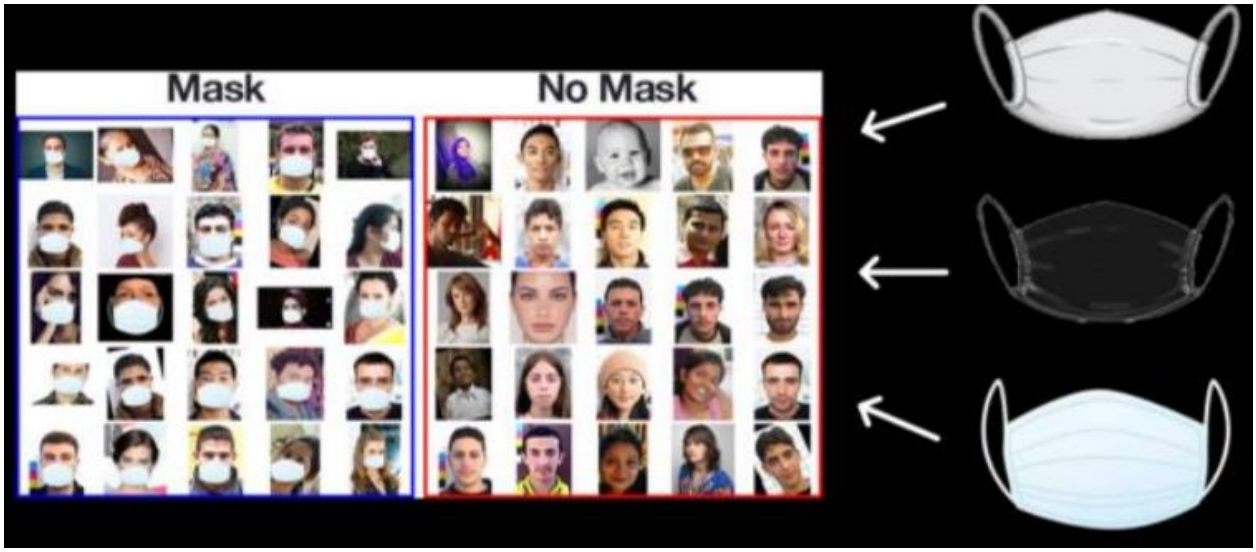
Figure 3.1.2.1: The steps involved in an OpenCV-based social distancing applicatiom

The steps to build a social distancing detector include:

1. Apply **object detection** to detect all people (and *only* people) in a video stream (see this tutorial on building an **OpenCV people counter**)
2. **Compute the pairwise distances** between all detected people
3. Based on these distances, **check to see if any two people are less than *N* pixels apart**

3.2 Datasets

Datasets are used only for training and testing the facemask detection model.



The datasets required for this project were collected from:

www.github.com

www.kaggle.com

.

3.3 Algorithms

3.3.1 Facemask Detection

The algorithm used for facemask detection is Convolutional Neural Networks (CNN).

3.3.2 Social Distancing Detection

MobileNet-SSD and CLAHE algorithm for better object detection.

3.4 Modules

3.4.1 Face Mask Detection

3.4.1.1 IMAGE SEPARATION

For the classification task, data were divided into a training and testing partitions.

3.4.1.2DATA PREPOSSESSING

Data preprocessing involves conversion of data from a given format to much more user friendly, desired and meaningful format. It can be in any form like tables, images, videos, graphs, etc. These organized information fit in with an information model or composition and captures relationship between different entities.The proposed method deals with image data using Numpy and OpenCV.

3.4.1.2.1 Data Visualization

Data visualization is the process of transforming abstract data to meaningful representations using knowledge communication and insight discovery through encodings. It is helpful to study a particular pattern in the dataset .

The total number of images in the dataset is visualized in both categories – ‘with mask’ and ‘without mask’.

The statement *categories=os.listdir(data_path)* categorizes the list of directories in the specified data path. The variable *categories* now looks like: [‘with mask’, ‘without mask’]

Then to find the number of labels, we need to distinguish those categories using

labels=[i for i in range(len(categories))]. It sets the labels as: [0, 1]

Now, each category is mapped to its respective

label using *label_dict=dict(zip(categories,labels))* which at first returns an iterator of

tuples in the form of zip object where the items in each passed iterator is paire

d together consequently. The mapped variable *label_dict* looks like: {‘with mask’: 0, ‘without mask’: 1}

3.4.1.2.2 Conversion of RGB image to Gray image

Modern descriptor-based image recognition systems regularly work on grayscale images,

without elaborating the method used to convert from color-to-grayscale. This is because the color-to-grayscale method is of little consequence when using robust descriptors. Introducing nonessential information could increase the size of training data required to achieve good performance. As grayscale rationalizes the algorithm and diminishes the computational requisites, it is utilized for extracting descriptors instead of working on color images instantaneously

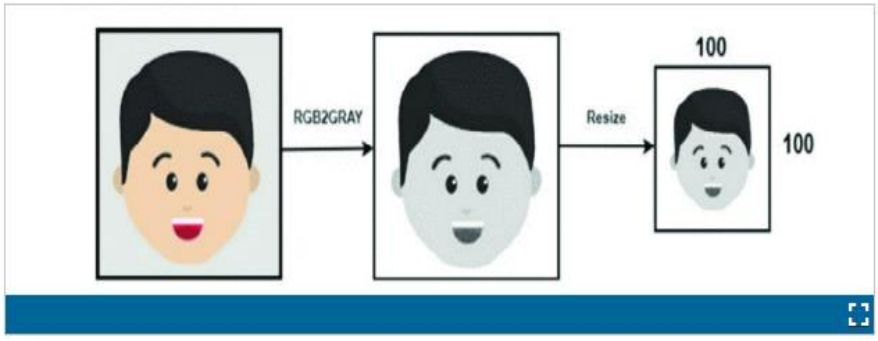


Fig. 3.

Fig 3.4.2.2.1 Conversion of RGB image to one 100*100 GrayScale.

3.4.1.2.3 Image Reshaping

The input during relegation of an image is a three-dimensional tensor, where each channel has a prominent unique pixel. All the images must have identically tantamount size corresponding to 3D feature tensor. However, neither images are customarily coextensive nor their corresponding feature tensors [10]. Most CNNs can only accept fine-tuned images. This engenders several problems throughout data collection and implementation of model. However, reconfiguring the input images before augmenting them into the network can help to surmount this constraint

3.4.1.2 TRAINING OF MODEL

3.4.1.2.1 Building the model using CNN architecture

CNN has become ascendant in miscellaneous computer vision tasks [12]. The current method makes use of Sequential CNN.

The First Convolution layer is followed by Rectified Linear Unit (ReLU) and MaxPooling layers. The Convolution layer learns from 200 filters. Kernel size is set to 3 x 3 which specifies the height and width of the 2D convolution window. As the model should be aware of the shape of the input expected, the first layer in the model needs to be provided with information about input shape. Following layers can perform instinctive shape reckoning . In this case, *input_shape* is specified as *data.shape[1:]* which returns the dimensions of the data array from index 1. Default padding is "valid" where the spatial dimensions are sanctioned to truncate and the input volume is non-zero padded. The activation parameter to the Conv2D class is set as "relu". It represents an approximately linear function that possesses all the assets of linear models that can easily be optimized with gradient-descent methods. Considering the performance and generalization in deep learning, it is better compared to other activation functions . Max Pooling is used to reduce the spatial dimensions of the output volume. Pool_size is set to 3 x 3 and the resulting output has a shape (number of rows or columns) of: shape_of_output = (input_shape – pool_size + 1) / strides), where strides has default value (1,1)

As shown in fig,the second Convolution layer has 100 filters and Kernel size is set to 3 x 3. It is followed by ReLu and MaxPooling layers. To insert the data into CNN, the long vector of input is passed through a Flatten layer which transforms matrix of features into a vector that can be fed into a fully connected neural network classifier. To reduce overfitting a Dropout layer with a 50%

chance of setting inputs to zero is added to the model. Then a Dense layer of 64 neurons with a ReLu activation function is added. The final layer (Dense) with two outputs for two categories uses the Softmax activation function.

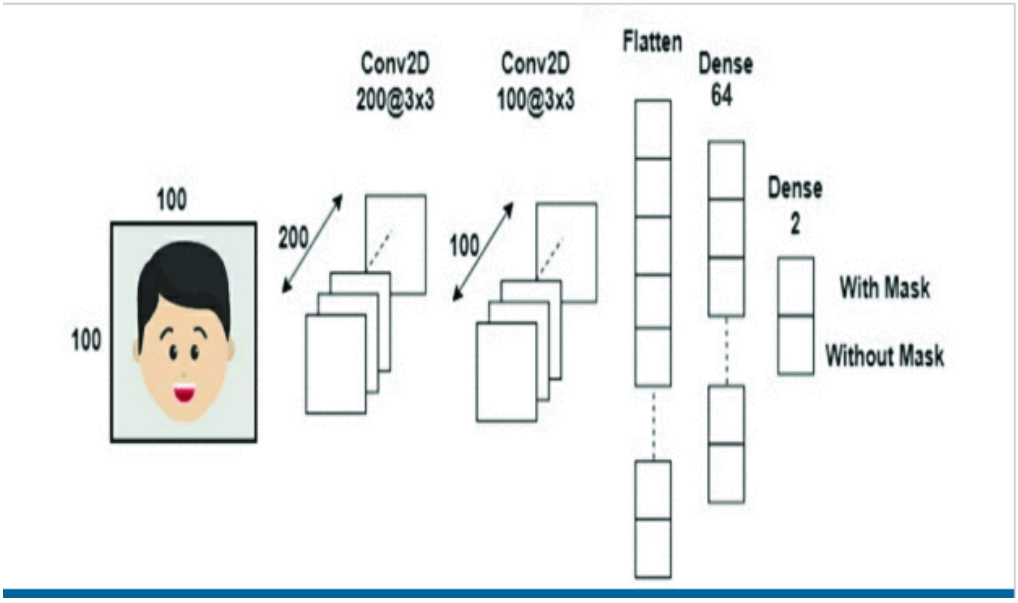


Fig: Convolutional Neural Network Architecture

3.4.1.2.2 Splitting the data and training the CNN model

After setting the blueprint to analyze the data, the model needs to be trained using a specific dataset and then to be tested against a different dataset. A proper model and optimized *train_test_split* help to produce accurate results while making a prediction. The test_size is set to 0.1 i.e. 90% data of the dataset undergoes training and the rest 10% goes for testing purposes. The validation loss is monitored using *ModelCheckpoint.* Next, the images in the training set and the test set are fitted to the Sequential model. Here, 20% of the training data is used as validation data. The model is trained for 20 epochs (iterations) which maintains a trade-off between accuracy and chances of overfitting. Fig. 5 depicts visual representation of the proposed model**.**

Input: Dataset including faces with and without masks
Output: Categorized image depicting the presence of face mask
1 **for** *each image in the dataset* **do**
2 | Visualize the image in two categories and label them
3 | Convert the RGB image to Gray-scale image
4 | Resize the gray-scale image into 100 x 100
5 | Normalize the image and convert it into 4 dimensional array
6 **end**
7 **for** *building the CNN model* **do**
8 | Add a Convolution layer of 200 filters
9 | Add the second Convolution layer of 100 filters
10 | Insert a Flatten layer to the network classifier
11 | Add a Dense layer of 64 neurons
12 | Add the final Dense layer with 2 outputs for 2 categories
13 **end**
14 Split the data and train the model

Algorithm used for face mask detection model.

3.4.2 Social Distancing Detection

The implementation:

1. Using the YOLO object detector to detect people in a video stream
2. Determining the centroids for each detected person
3. Computing the pairwise distances between all centroids
4. Checking to see if any pairwise distances were $< N$ pixels apart, and if so, indicating that the pair of people violated social distancing rules

The main purpose of this system is to process captured video footage for person detection and further processing for social distancing or safety violation. So, the process starts with reading the frames of a video feed one by one. This is shown in Fig. 3 which illustrates the whole sequence of activities in a flowchart.

The most important feature of this study is the object detection framework. This is due to the element of this study that focuses on determining the location of a person from the input frame. Hence, choosing the most suitable object detection model is important to avoid any problems in detecting persons.

3.4.2.1 Object Detection Model

The model used is MobileNet-SSD , since it takes less time for execution.
In this study, Caffe deep learning model framework is used to run the object detection model.*MobilenetSSD* is an object detection model that computes the bounding box and category of an object from an input image. This *Single Shot Detector* (SSD) object detection model uses *Mobilenet* as backbone and can achieve fast object detection optimized for mobile devices.

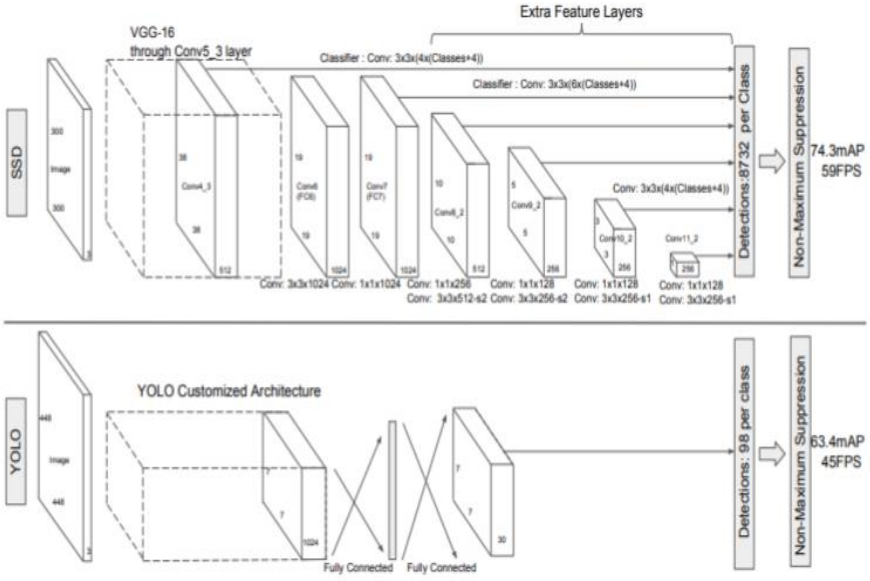


Fig. 2: A comparison between two single shot detection models: SSD and YOLO [5]. Our SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. SSD with a 300 × 300 input size significantly outperforms its 448 × 448 YOLO counterpart in accuracy on VOC2007 `test` while also improving the speed.

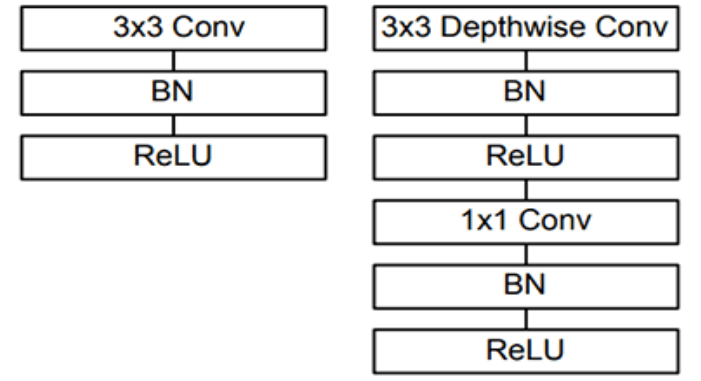


Figure 2: *(Left)* Standard convolutional layer with batch normalization and ReLU. *(Right)* Depthwise separable convolution with depthwise and pointwise layers followed by batch normalization and ReLU (figure and caption from Liu et al.).

3.4.2.2 Applying the CLAHE Algorithm

CLAHE is a variant of *Adaptive histogram equalization (AHE)* which takes care of over-amplification of the contrast. CLAHE operates on small regions in the image, called tiles, rather than the entire image. The neighboring tiles are then combined using bilinear interpolation to remove the artificial

boundaries.
This algorithm can be applied to improve the contrast of images.
We can also apply CLAHE to color images, where usually it is applied on the luminance channel and the results after equalizing only the luminance channel of an HSV image are much better than equalizing all the channels of the BGR image.
In this tutorial, we are going to learn how to apply CLAHE and process a given input image for histogram equalization

3.4.2.3 Determine person location

In determining the position of a person's bounding box as well as the segment involved, each ground plane point is used to compare the ROI range.

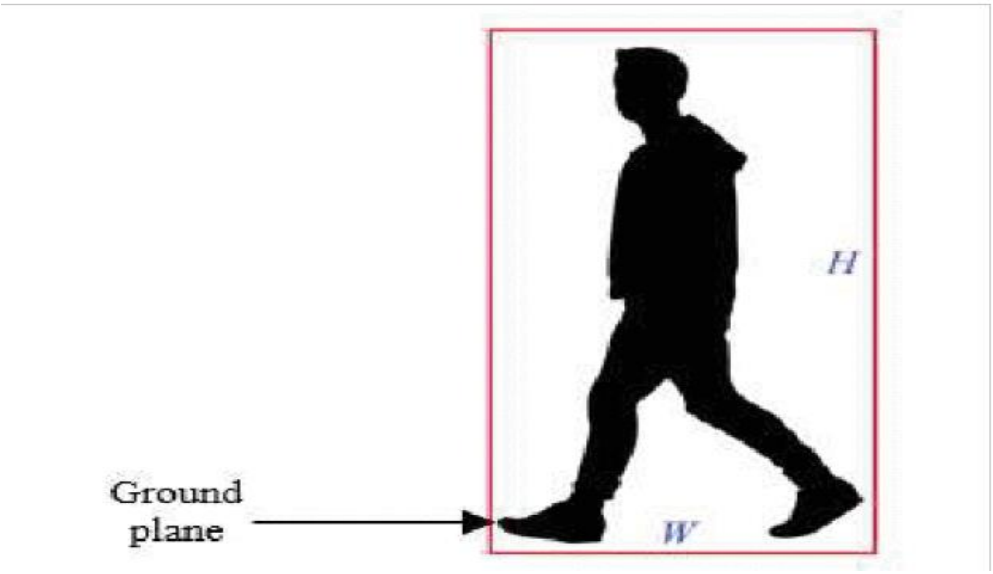


Fig. The ground plane of a human walking model

Surveillance cameras are usually placed at high places as the overhead camera especially to monitor a certain area e.g. highrisk area or areas of interest for an organization. In this case, it is more suitable to compare the ground plane for the detection box instead of using the center point value.

3.4.2.4 Calculate the centroid of a bounding box

To measure the center point, $C(x, y)$, of the bounding box for the detected person, midpoint equation is used as in (2).

$$C(x, y)=(x_{min}+x_{max}2, y_{min}+y_{max}2)$$

Each of the minimum and maximum value for the corresponding width, x_{min} and x_{max}, and height, y_{min} and y_{max}, of the bounding box will be used to calculate the center point of the bounding box.

3.4.2.5 Calculate the distance between the bounding boxes

To measure the distance, C1(xmin , ymin) and C2(xmax , ymax), between each of the detected person in the frame, distance equation is used as in (3).

$$d(C1′C2)=sqrt((xmax−xmin)2+(ymax−ymin))2$$

3.5DEPLOYMENT

Model deployment is simply the engineering task of exposing an ML model to real use. The term is often used quite synonymously with making a model available via real-time APIs.

Chapter 4

RESULT ANALYSIS

The facemask detection model is trained, validated and tested upon two datasets. Corresponding to dataset 1, the method attains accuracy up to 95.77% (shown in fig. 7). Fig. 6 depicts how this optimized accuracy mitigates the cost of error. Dataset 2 is more versatile than dataset 1 as it has multiple faces in the frame and different types of masks having different colors as well. Therefore, the model attains an accuracy of 94.58% on dataset 2 as shown in Fig. 9. Fig. 8 depicts the contrast between training and validation loss corresponding to dataset 2. One of the main reasons behind achieving this accuracy lies in *MaxPooling*. It provides rudimentary translation invariance to the internal representation along with the reduction in the number of parameters the model has to learn. This sample-based discretization process down-samples the input representation consisting of image, by reducing its dimensionality. Number of neurons has the optimized value of 64 which is not too high. A much higher number of neurons and filters can lead to worse performance. The optimized filter values and pool_size help to filter out the main portion (face) of the image to detect the existence of mask correctly without causing over-fitting.

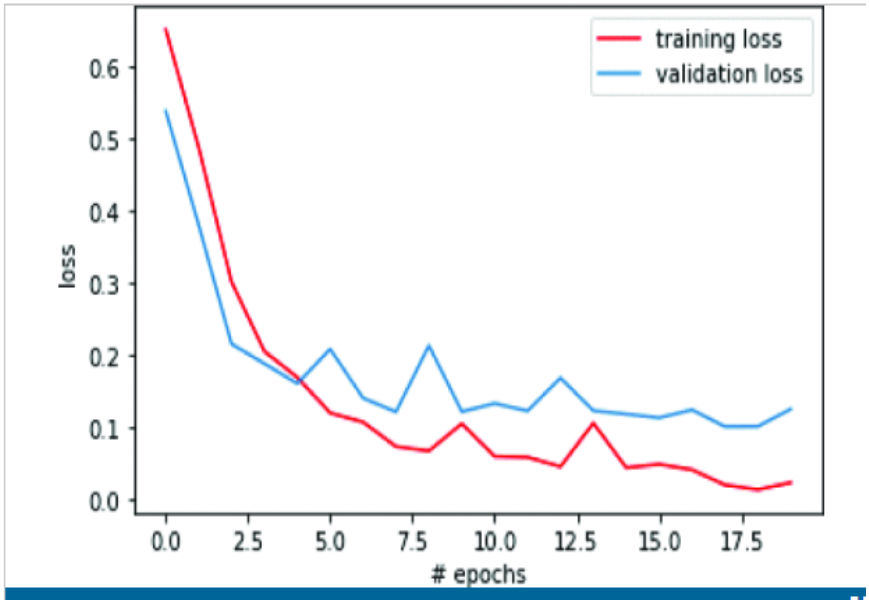


Fig.
epochs vs loss corresponding to dataset 1

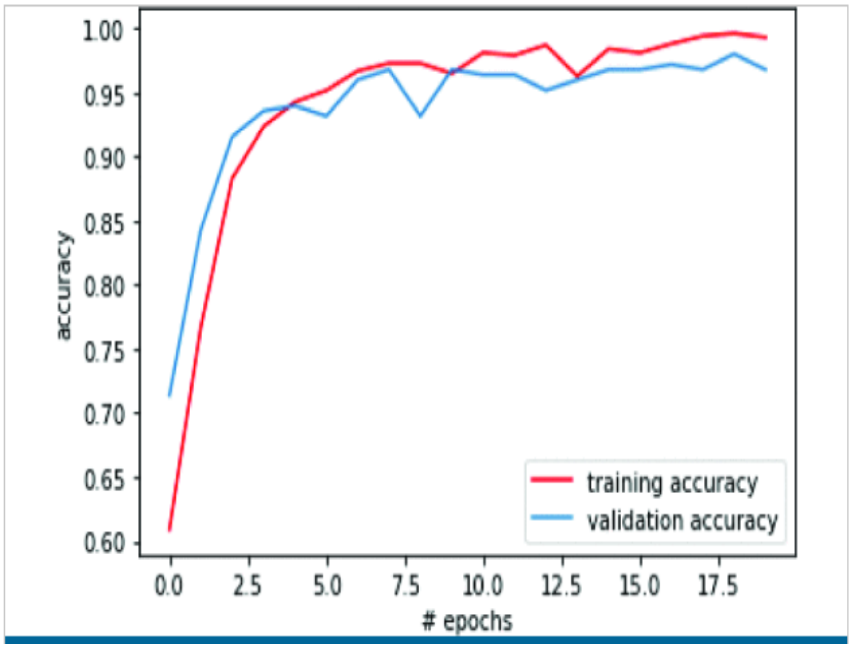


Fig.
epochs vs accuracy corresponding to dataset 1

The system can efficiently detect partially occluded faces either with a mask or hair or hand. It considers the occlusion degree of four regions – nose, mouth, chin and eye to differentiate between annotated mask or face covered by hand. Therefore, a mask covering the face fully including nose and chin will only be treated as “with mask” by the model.

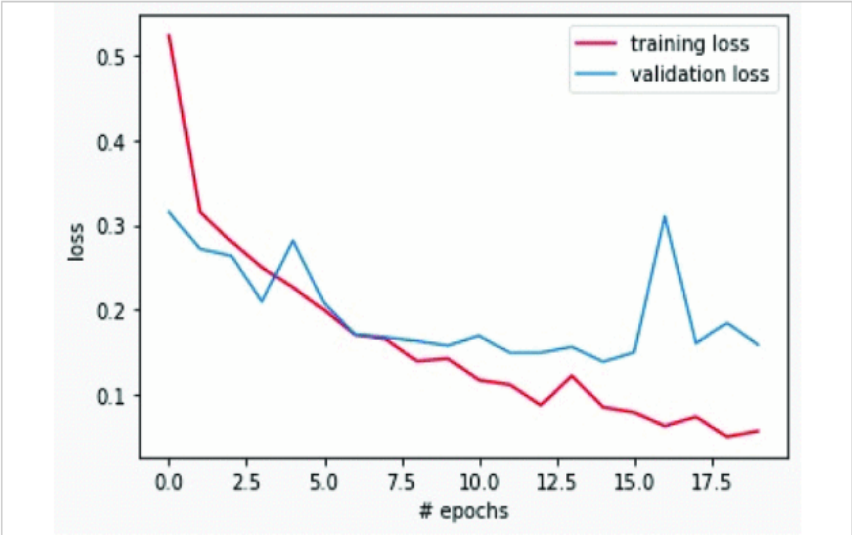


Fig.
epochs vs loss corresponding to dataset 2

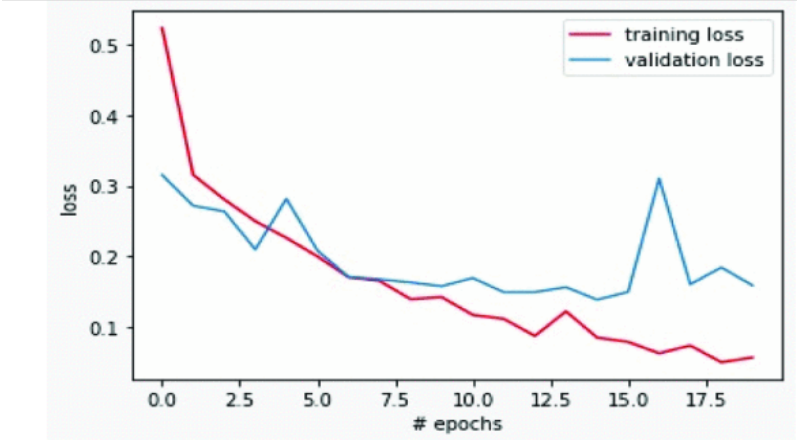


Fig.
epochs vs accuracy corresponding to dataset 2

System development for social distancing project has been completed based on Python 3, OpenCV for image processing techniques and Caffe object detection model framework. Based on this developed system, some analysis has been performed to test its effectiveness and results have been obtained. Specifically, the MobileNet SSD Caffe model has been used in this study as the key algorithm in person detection

Chapter 5

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

.

Using basic ML tools and simplified techniques the method has achieved reasonably high accuracy. It can be used for a variety of applications. Wearing a mask may be obligatory in the near future, considering the Covid-19 crisis. Many public service providers will ask the customers to wear masks correctly to avail of their services. The deployed model will contribute immensely to the public health care system.

Social distancing is one of the important precautions in reducing physical contact that may lead to the spread of coronavirus. Consequences of non-compliance with these guidelines will be causing the higher rates of virus transmission. A system has been developed using Python and OpenCV library to implement two proposed features.

The first feature is on detecting violations of social distancing, while the second feature is on detecting violations of entering restricted areas. Both features have been tested for accuracy. Based on the overall results, this study is seen to meet all of its objectives. However, there are some limitations to the results obtained. Based on the tests performed on the system, the results show that the object detection model used for detecting persons is having the difficulty in detecting people correctly in the outdoor environment and difficult scenes with distant scenes. For further improvement in the future, a better object detection model can be implemented.

5.2 Future Scope

The deployed face mask detection model will contribute immensely to the public health care system. In future it can be extended to detect if a person is wearing the mask properly or not. The model can be further improved to detect if the mask is virus prone or not i.e. the type of the mask is surgical, N95 or not.

Based on the overall results, the social distancing detection study is seen to meet all of its objectives. However,there are some limitations to the results obtained. Based on the tests performed on thesystem, the results show that the object detection model used for detecting persons ishaving the difficulty in detecting people correctly in the outdoor environment and difficult scenes with distant scenes. For further improvement in the future, a better object detection model can be implemented.

Chapter 6

CODING

6.1

6.1.1 app.py (Face Mask Detection)

```
from flask import Flask,render_template,request,send_file,send_from_directory

import numpy as np

import pandas as pd

import sklearn.metrics as m

from keras.utils.np_utils import to_categorical

import os
```

```
import cv2

from sklearn.model_selection import train_test_split

from keras.models import Sequential

from keras.layers import Dense,Conv2D,Flatten,Activation,MaxPooling2D

from keras.preprocessing import image

from keras.models import load_model

import keras.backend.tensorflow_backend as tb


from skimage import transform

import argparse

from keras.applications.vgg16 import VGG16

from keras.models import Model

import tensorflow as tf


tb._SYMBOLIC_SCOPE.value = True


model = load_model('model-facemask.h5')
```

```
def processesing(arr):
  for i in arr:
    if(i[0]>i[1]):
      return 0
    else:
      return 1
def images(img):
    image_read=[]
    image1=image.load_img(img)
    image2=image.img_to_array(image1)
    image3=cv2.resize(image2,(224,224))
    image_read.append(image3)
    img_array=np.asarray(image_read)
    return img_array
app = Flask(__name__,static_folder='static',template_folder='templates')
```

```
@app.route('/')
def home():
      return render_template("index.html")


def percentage(u,pre):
  sum=u[0][0]+u[0][1]
  return 100*u[0][pre]/sum


@app.route('/predict',methods=['POST','GET'])
def predict():
  if request.method=='POST':
    img=request.files['ima'].read()
    print(img)
    npimg = np.fromstring(img, np.uint8)
# convert numpy array to image
    img = cv2.imdecode(npimg,cv2.IMREAD_COLOR)
    cv2.imwrite("images/output.png",img)
```

```
image3=cv2.resize(img,(224,224))

image = np.expand_dims(image3, axis=0)


imgarray=image

print(imgarray)

u=model.predict(imgarray)

pre=processesing(u)

print(u)

perc=percentage(u,pre)


if pre==0:

  print(0)

  response="Mask ON! You are Safe"

  return render_template("prediction.html",predict=response,percent=str(perc)+" %")
```

```
    if pre==1:
      print(1)
      response="Mask OFF! Please wear the Mask"
      return render_template("prediction.html",predict=response,percent=str(perc)+" %")

  if request.method=='GET':
    return render_template("index.html")


if __name__=='__main__':
    app.run(debug=False,threaded=False)
```

6.1.2 social_distamce.py

```
def CLAHE(bgr_image: np.array) -> np.array:
    hsv = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2HSV)
    hsv_planes = cv2.split(hsv)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    hsv_planes[2] = clahe.apply(hsv_planes[2])
    hsv = cv2.merge(hsv_planes)
    return cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)

def centroid(startX,endX,startY,endY):
    centroid_x = round((startX+endX)/2,4)
    centroid_y = round((startY+endY)/2,4)
    bboxHeight = round(endY-startY,4)
    return centroid_x,centroid_y,bboxHeight

def calcDistance(bboxHeight):
    distance = (calculateConstant_x * calculateConstant_y) / bboxHeight
    return distance

def drawResult(frame,position):
    for i in position.keys():
        if i in highRisk:
            rectangleColor = RED
        elif i in mediumRisk:
            rectangleColor = YELLOW
        else:
            rectangleColor = GREEN
        (startX, startY, endX, endY) = detectionCoordinates[i]

        cv2.rectangle(frame, (startX, startY), (endX, endY), rectangleColor, 2)
```

```
if __name__== "__main__":

    caffeNetwork = cv2.dnn.readNetFromCaffe("./SSD_MobileNet_prototxt.txt", "./SSD_MobileNet.caffemodel")
    cap = cv2.VideoCapture("./pedestrians.mp4")
    fourcc = cv2.VideoWriter_fourcc(*"XVID")
    output_movie = cv2.VideoWriter("./result.avi", fourcc, 24, (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)),
int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))))


    while cap.isOpened():

        debug_frame, frame = cap.read()
        highRisk = set()
        mediumRisk = set()
        position = dict()
        detectionCoordinates = dict()

        if not debug_frame:
            print("Video cannot opened or finished!")
            break

        if preprocessing:
            frame = CLAHE(frame)

        (imageHeight, imageWidth) = frame.shape[:2]
        pDetection = cv2.dnn.blobFromImage(cv2.resize(frame, (imageWidth, imageHeight)), 0.007843,
(imageWidth, imageHeight), 127.5)

        caffeNetwork.setInput(pDetection)
        detections = caffeNetwork.forward()

        for i in range(detections.shape[2]):

            accuracy = detections[0, 0, i, 2]
```

```
  if accuracy > accuracyThreshold:
    # Detection class and detection box coordinates.
    idOfClasses = int(detections[0, 0, i, 1])
    box = detections[0, 0, i, 3:7] * np.array([imageWidth, imageHeight, imageWidth, imageHeight])
    (startX, startY, endX, endY) = box.astype('int')

    if idOfClasses == personLabelID:
      # Default drawing bounding box.
      bboxDefaultColor = (255,255,255)
      cv2.rectangle(frame, (startX, startY), (endX, endY), bboxDefaultColor, 2)
      detectionCoordinates[i] = (startX, startY, endX, endY)

      # Centroid of bounding boxes
      centroid_x, centroid_y, bboxHeight = centroid(startX,endX,startY,endY)
      distance = calcDistance(bboxHeight)
      # Centroid in centimeter distance
      centroid_x_centimeters = (centroid_x * distance) / calculateConstant_y
      centroid_y_centimeters = (centroid_y * distance) / calculateConstant_y
      position[i] = (centroid_x_centimeters, centroid_y_centimeters, distance)

#Risk Counter Using Distance of Positions
for i in position.keys():
  for j in position.keys():
    if i < j:
      distanceOfBboxes = sqrt(pow(position[i][0]-position[j][0],2)
                  + pow(position[i][1]-position[j][1],2)
                  + pow(position[i][2]-position[j][2],2)
                  )
      if distanceOfBboxes < 150: # 150cm or lower
        highRisk.add(i),highRisk.add(j)
      elif distanceOfBboxes < 200 > 150: # between 150 and 200
        mediumRisk.add(i),mediumRisk.add(j)


cv2.putText(frame, "Person in High Risk : " + str(len(highRisk)) , (20, 20),
```

```
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
    cv2.putText(frame, "Person in Medium Risk : " + str(len(mediumRisk)) , (20, 40),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 2)
    cv2.putText(frame, "Detected Person : " + str(len(detectionCoordinates)), (20, 60),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

    drawResult(frame, position)
    if write_video:
      output_movie.write(frame)
    cv2.imshow('Result', frame)
    waitkey = cv2.waitKey(1)
    if waitkey == ord("q"):
      break

  cap.release()
  cv2.destroyAllWindows()
```

Chapter 7

SCREEN SHOTS

Here are some sample screenshots of the proposed system which includes:

Home Screen, Image Upload Form, and the Prediction Screen

Figure 7.1: Home Screen

Figure 7.2: Image Upload Form

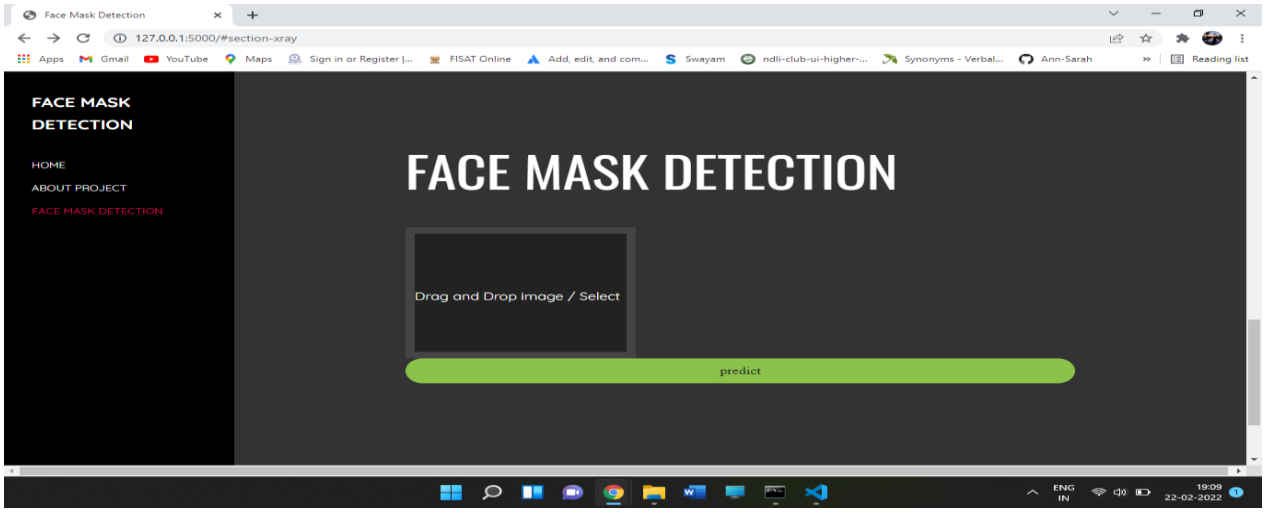
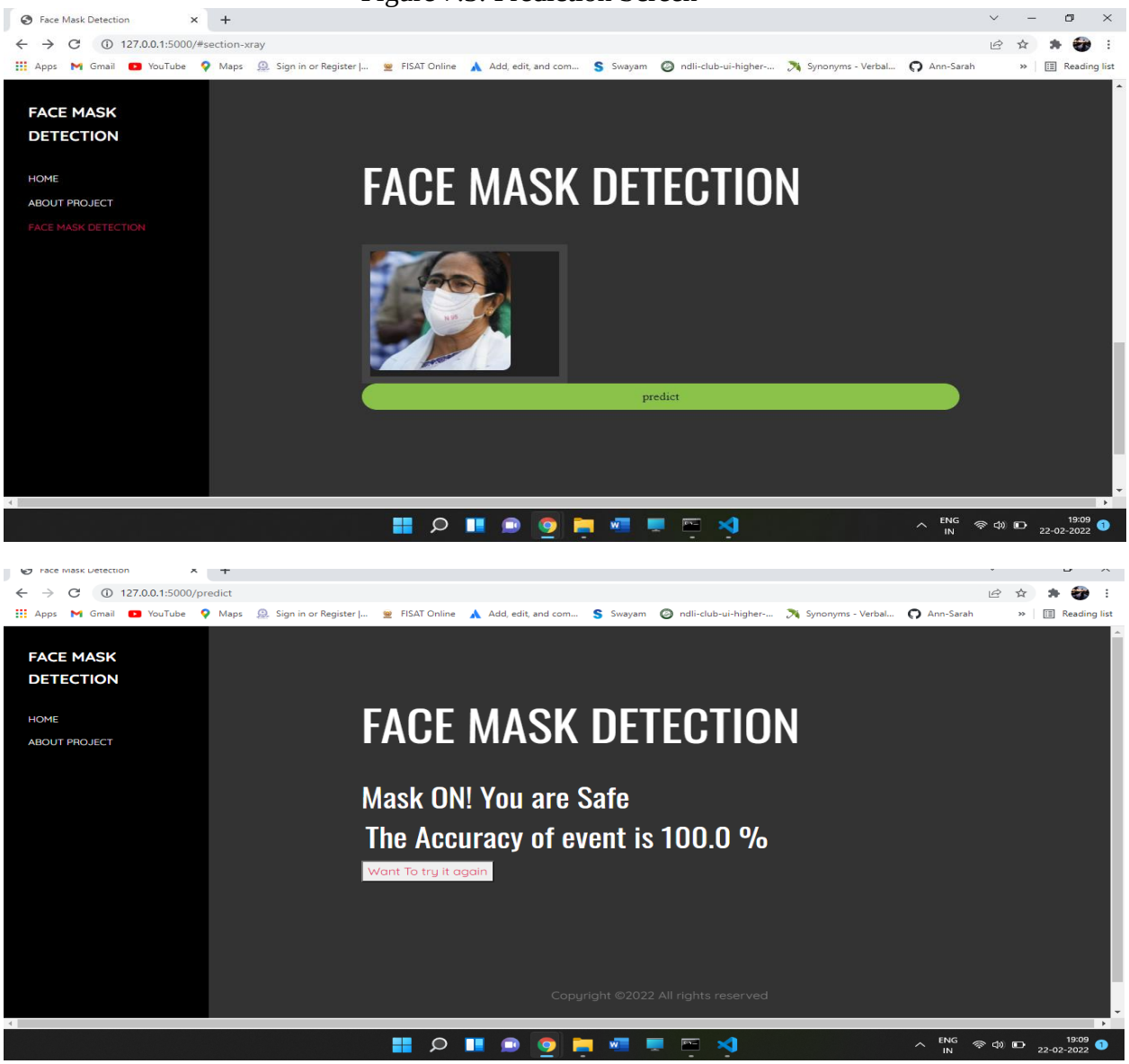
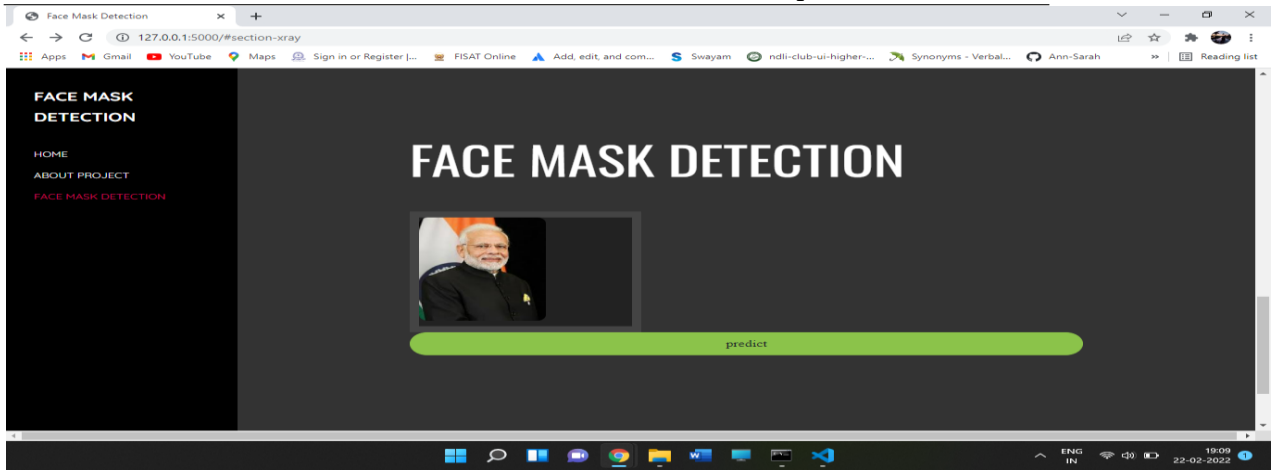
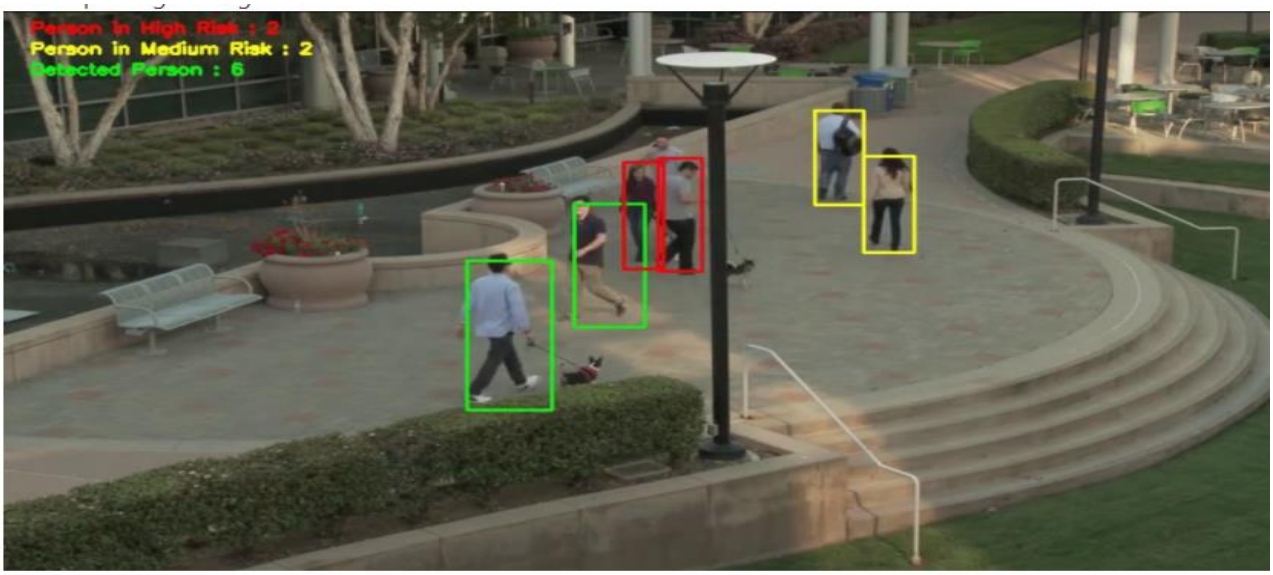


Figure 7.3: Prediction Screen




Face Mask Detection
127.0.0.1:5000/#section-xray
FACE MASK DETECTION
HOME
ABOUT PROJECT
FACE MASK DETECTION
FACE MASK DETECTION
predict


Person in High Risk : 2
Person in Medium Risk : 2
Detected Person : 6


Total People = 17 (12 Safe 5 Unsafe) (0 Masked 5 Unmasked 12 Unknown)

Chapter 8

REFERENCES

(a) www.google.com

(b) www.youtube.com

(c) www.wikipedia.com

(d) Ieeeexplore.iee.com