

# Outlines..

<i>Objective</i> .....	2
1. Introduction.....	3
1.1    Symptoms of COVID-19.....	3
1.2    Face Mask Detection.....	3
1.3    Social Distancing Detection.....	3
2. Feasibility Study.....	4
3. Literature Review.....	5
4. Technology Used.....	6
5. Software/Hardware Requirements.....	7
5.1    Software.....	7
5.2    GPU requirement.....	7
5.3    NVIDIA.....	7
5.4    Hardware.....	8
6. Libraries used in Project.....	9
7. CODING.....	10
7.1    Module 1.....	11
7.2    Module 2.....	16
7.3    Module 3.....	18
7.4    Screenshot of the result.....	26
8. Conclusion.....	27
9. Future Scope.....	28
<i>Reference</i> .....	28

# **SOCIAL DISTANCING & FACE MASK DETECTION**

## ***Using TENSORFLOW***

### **Objective**

*According to data obtained by the World Health Organization (WHO), the global pandemic of COVID-19 has severely impacted the world and has now infected more than eight million people worldwide. Wearing face masks and following safe social distancing are two of the enhanced that contributes to public safety, we proposed and efficient computer vision-based approach focused on the real-time automated monitoring of people to detect both safe social violations through cameras. In this proposed system modern deep learning algorithm have been mixed with geometric techniques for building a robust modal which covers three aspects of detection, tracking and validation. Thus, the proposed system favors the society by saving time and helps in lowering the spread of corona virus. It can be implemented effectively in current situation when lockdown is eased to inspect persons in public gatherings, shopping malls, etc. Automated inspection reduces manpower to inspect the public and also can be used in any place.[1]*

# INTRODUCTION

Since COVID-19 has become a pandemic, the entire world is finding ideas and method to stop the spread of it. As the ground rule to stop the spread is to maintain social distance and wearing a mask while going out. The novel COVID or COVID-19 began spreading during 2019 December at first from China, the city guest Wuhan. In china, the infection is started from the creatures and spread generally as a pandemic circumstance everywhere on the world. Corona Virus infection communicates to other however coordinates actual contact with the influenced patients and through air. The infection straight forwardly hit the lung cells through respiratory arrangement of the patients and permits it to recreate the infection and makes an extreme irresistible problem in an exceptionally limited ability to focus.

Our main motive, Face mask detection with Social Distancing is the task of identifying an already detected object as that person wear mask or not and they are walking with maintaining Social Distance to each other.

## 1.1 SYMPTOMS of COVID-19

The most widely recognized manifestations of COVID are dry hack, fever, windedness, discomfort and migraine. The quick send of COVID produce serious muscles torments and empower the individuals with debilitate invulnerable framework got trained by it without any problem. The outrageous phase of COVID-19 prompts demise of numerous people groups with extreme failing of lung and different organs of the body. Different examples of medicines are dealing with by the doctors everywhere on the world to find an effective method of restricting the infection being communicated to most exceedingly terrible stage.

## 1.2 FACE MASK DETECTION

Face Mask Detection Platform utilizes Artificial Network to perceive if a person does/doesn't wear a mask. The application can be associated with any current or new IP cameras to identify individuals with/without a mask. The face mask detection process starts from the image acquisition using a camera. The imaging device and the modules are developed using TensorFlow and Open-CV programming detect the face and evaluate all faces point that to detect is that face wear mask or not. If person wear mask they will be in safe zone it display as green rectangle-box with safe alert where if person don't wear mask then it will be shown in red rectangle-box and with the message of Alert as well.

## 1.3 SOCIAL DISTANCING DETECTION

Social Distancing detection will detect that two or more persons in a single frame are walking with maintain social distancing with at least 0.80 meter of range with each other. By using Euclidean Distance method, it will detect that persons maintaining or following social distancing under guidance of WHO. If they maintain then it will shown in Green Rectangular-box with safe alert message where if they don't following social distancing then system will display an alert message with red rectangular-box.

## **CHAPTER – 2**

### **FEASIBILITY STUDY**

It is not feasible to manually track the implementation of this policy. Technology holds the key here. We introduce a Deep Learning based system that can detect instances where face masks are not used properly.

Our system consists of a dual stage Convolutional Neural Network (CNN) architecture capable of detecting masked and unmasked faces and can be integrated with pre-installed CCTV cameras.

This will help track safety violations, promote the use of face masks, and ensure a safe working environment. The study is carried out by analyzing the necessary technologies involved in the previous research and formulating the efficient model that help the people in real-time. Application development using keras and Tensor flow is most widely used in current trends, henceforth we are suggesting the Python based image processing and machine learning technique to achieve the robust structure

the feasibility of considering a diversity of cases using the proposed method with a high level of confidence and accuracy in social distancing monitoring and risk assessment with the help of Deep Learning and Computer Vision.

## CHAPTER – 3

### LITERATURE REVIEW

Presented a cumulative article that contains the different categories of COVID datasets available. These datasets are publicly available open-source data. The listed sequence of data holds the X-rays images, CT scan images, in few cases with health background is narrowed then MRI images are also used. Another set of datasets used for COVID analysis is textual data based on discussions, medical suggestions made on social media etc. Clinical test results and reports are also considered in many dataset windows that are utilized for analysis of diagnostic procedures. [2]

1. Evaluated a virtual social distancing model that helps out the peoples being alerted in the public places. They graphically represented four types of spacing calling intimate space, personal space, social space and public spacing. Based on distance measurement rule, the spaces are measured. The procedure belongs to scene understanding and geometrical measurement, homograph estimation, metric references and density estimation etc. the secondary analysis consists of two-dimensional people detection and multiple angle people Social Distance Monitoring and Face Mask Detection Using Deep Neural Network 2 detection. [3]
2. Evaluated a human detection framework, for monitoring the social distancing and safety misuse in the pandemic situation. They have utilized the pre-trained model of recurrent CNN to identify the different models. The human detection process is done using blob segmentation. These blobs are tracked with respect to the other to measure the distance between the other persons. They faced the challenge of detecting the person's body blobs during the outdoor area because of the correlations of other objects nearby. They found this challenge need to be rectified in further research. [4]
3. Developed a novel face recognition system using principle component analysis (PCA) and convolutional neural network. The experiment is tested with discriminant algorithms, multi-layered perceptron , naïve bayes model and support vector machines. This journal on face recognition provides the challenges in recognizing the multi face model and how to overcome the same in future research. [5]
4. Evaluated a deep learning approach on face detection and face classification. Clustering of different faces are done using pre-trained facial dataset. FDDB dataset is applied to train and test the proposed model. The proposed model is altered to obtain novelty and high performing in prediction. The accuracy achieved on using convolution neural network is noted high and the future challenges are declared using real time face images and live capturing of videos.[6]
5. Explained in detail on multi-face detection model based on machine learning algorithms Ada-Boost, Haar boost, SVM(support vector machine) and gradient boost models. These techniques are clearly explained to achieve high accuracy with one another. The major challenge faced by the researcher is false positive rate increases in certain facial data.[7]

## CHAPTER – 4

### TECHNOLOGY USED



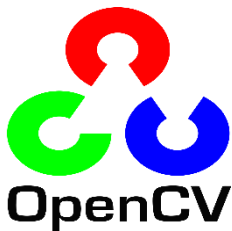
TENSORFLOW



PYTHON



SCIKIT LEARN



COMPUTER VISION (OPEN CV)



Scientific Python (SciPy)

#### ADDITIONAL METHODS AND TECHNIQUES:

- Pretrained **Yolov3** is used to detect the people.
- Pretrained **SSD** is used to detect faces.
- Trained **MobileNetV2** is used as face mask classifier.
- **Euclidean Distance** is used to calculate social distancing violations.

## CHAPTER – 5

### SOFTWARE/HARDWARE REQUIREMENTS

#### 5.1 SOFTWARE:

Software:	Anaconda, Python 3.x (3.8 or earlier)
Editor:	VS Code/ PyCharm/ Sublime/ Spyder
Environment:	TensorFlow
GPU Drivers:	Nvidia® CUDA® 11.0 requires 450.x or above CUDA® Toolkit (TensorFlow >= 2.4.0) cuDNN SDK 8.0.4 (TensorFlow >= 2.4.0)

---

#### GPU (Graphics Processing Unit)

- WHY GPU...??

CUDA is a parallel computing platform and programming model developed by Nvidia for general computing on its own GPUs (graphics processing units). CUDA enables developers to speed up compute-intensive applications by harnessing the power of GPUs for the parallelizable part of the computation.

While there have been other proposed APIs for GPUs, such as OpenCL, and there are competitive GPUs from other companies, such as AMD, the combination of CUDA and Nvidia GPUs dominates several application areas, including deep learning, and is a foundation for some of the fastest computers in the world.

- TensorFlow supports all GPUs...??

➔ Current development of TensorFlow Supports Only GPU Deep Learning and Machine Learning using NVIDIA CUDA, NVIDIA toolkits, So, all We need to install all required files or software need to boost TensorFlow file/TensorFlow Model.[8]



➔ CUDA has improved and broadened its scope over the years, more or less in lockstep with improved Nvidia GPUs. As of CUDA version 9.2, using multiple P100 server GPUs, we can realize up to 50x performance improvements over CPUs. The V100 (not shown in this figure) is another 3x faster for some loads. The previous generation of server GPUs, the K80, offered 5x to 12x performance improvements over CPUs.

#### NVIDIA CUDA, TOOLKITS, cuDNN

- CUDA in Deep Learning –

Deep learning has an outsized need for computing speed. For example, to train the models for Google Translate in 2016, the Google Brain and Google Translate teams did hundreds of one-week TensorFlow runs using GPUs; they had bought 2,000 server-grade GPUs from Nvidia for the purpose. Without GPUs, those training runs would have taken months rather than a week to converge.



For production deployment of those TensorFlow translation models, Google used a new custom processing chip, the TPU (tensor processing unit).

In addition to TensorFlow, many other DL frameworks rely on CUDA for their GPU support, including Caffe2, CNTK, Databricks, H2O.ai, Keras, MXNet, PyTorch, Theano, and Torch. In most cases they use the cuDNN library for the deep neural network computations. That library is so important to the training of the deep learning frameworks that all of the frameworks using a given version of cuDNN have essentially the same performance numbers for equivalent use cases. When CUDA and cuDNN improve from version to version, all of the deep learning frameworks that update to the new version see the performance gains. Where the performance tends to differ from framework to framework is in how well they scale to multiple GPUs and multiple nodes.

- NVIDIA CUDA toolkit –

The [CUDA Toolkit](#) includes libraries, debugging and optimization tools, a compiler, documentation, and a runtime library to deploy your applications. It has components that support deep learning, linear algebra, signal processing, and parallel algorithms. In general, CUDA libraries support all families of Nvidia GPUs, but perform best on the latest generation, such as the V100, which can be 3 x faster than the P100 for deep learning training workloads. Using one or more libraries is the easiest way to take advantage of GPUs, as long as the algorithms you need have been implemented in the appropriate library.[9]

- NVIDIA CUDA Deep Learning Libraries–



There are 3 libraries in CUDA deep learning: cuDNN, TensorRT, Deep Stream. I've used **cuDNN**.  
cuDNN— It is the GPU component for most open-source deep learning frameworks.

## 5.2 HARDWARE:

GPU:	Graphics Processor (NVIDIA)—min 2GB
Camera:	CCTV/ Webcam/ Mobile Camera (Sharing Camera)
Storage Disk (Optional):	SSD – Min 400MB/s Read Speed



## CHAPTER – 6

### LIBRARIES USING IN THIS PROJECT

Install require Libraries used in this project.

✓ face-detection	<code>pip install face-recognition==1.3.0</code>
✓ imutils	<code>pip install imutils==0.5.3</code>
✓ keras	<code>pip install Keras==2.4.3</code>
✓ matplotlib	<code>pip install matplotlib==3.3.3</code>
✓ opencv-python	<code>pip install opencv-python== 4.4.0.46</code>
✓ pandas	<code>pip install pandas==1.2.0</code>
✓ scikit-learn	<code>pip install scikit-learn==0.24.0</code>
✓ tensorflow	<code>pip install tensorflow==2.4.0</code>
✓ tensorflow-gpu	<code>pip install tensorflow-gpu==2.4.0</code>
✓ SciPy	<code>pip install scipy==1.6.0</code>

**Dataset required\*** : To train the model to detect face mask, download the required datasets from the link given below.

<https://www.kaggle.com/shantanu1118/face-mask-detection-dataset-with-4k-samples>

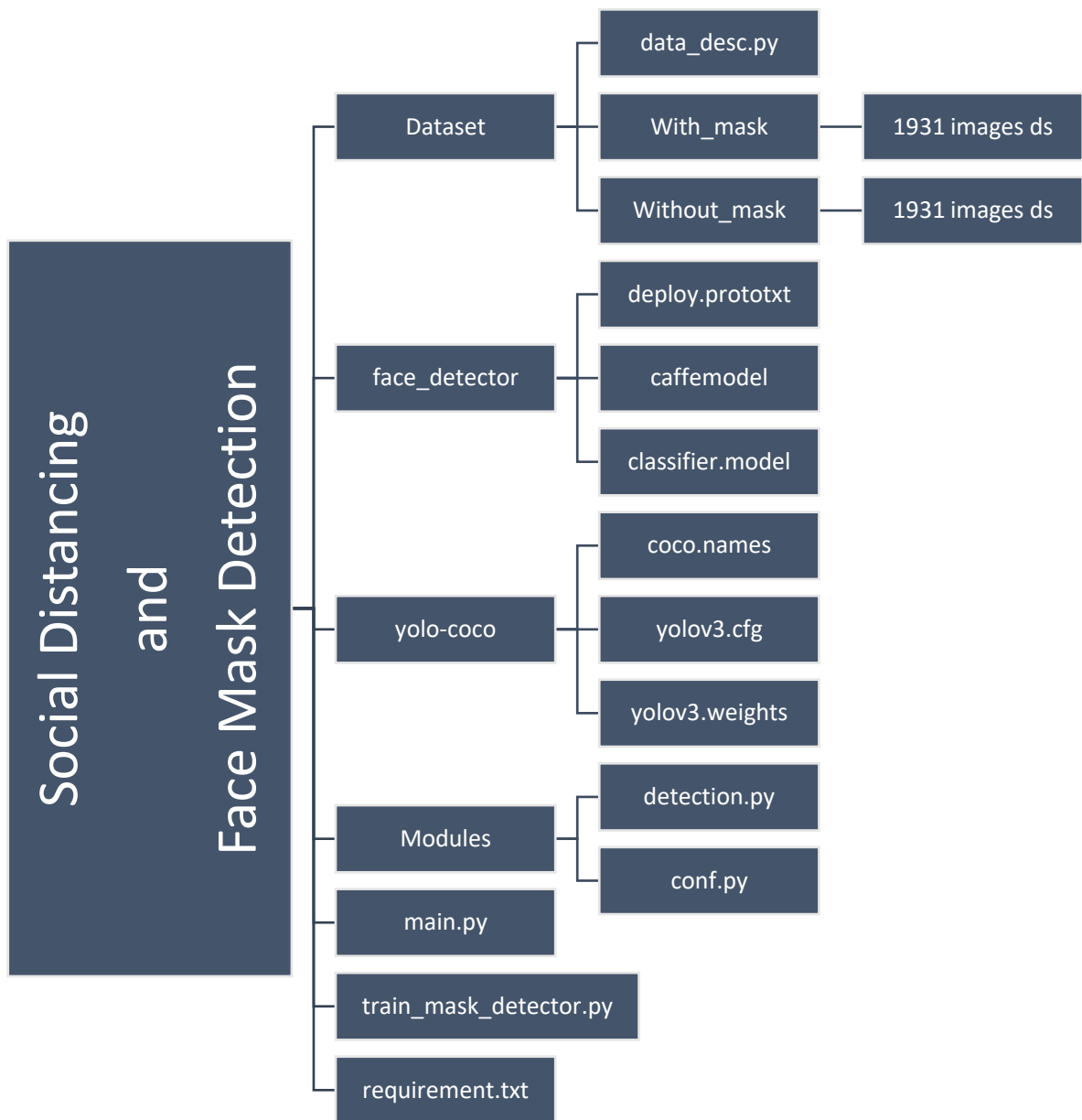
\*dataset of 4000 samples

- With\_mask (2000 Images sample)
- Without\_mask (2000 Images Sample)

## CHAPTER – 7

### CODING

**\*Structure of Folder:**



Total Modules: 3

1. train\_mask\_detector.py
2. detection.py
3. main.py

## MODULE NO. 1

### [ Train\_mask\_detector.py ]

Command to run this trainer file:

```
python train_mask_detector.py --dataset <path/to/dataset>
```

#### Step 1.

Import all required libraries modules for this training script. TensorFlow functions, keras, Scikit Learn, imutils, matplotlib.

```
1. from tensorflow.keras.preprocessing.image import ImageDataGenerator
2. from tensorflow.keras.applications import MobileNetV2
3. from tensorflow.keras.layers import AveragePooling2D
4. from tensorflow.keras.layers import Dropout
5. from tensorflow.keras.layers import Flatten
6. from tensorflow.keras.layers import Dense
7. from tensorflow.keras.layers import Input
8. from tensorflow.keras.models import Model
9. from tensorflow.keras.optimizers import Adam
10. from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
11. from tensorflow.keras.preprocessing.image import img_to_array
12. from tensorflow.keras.preprocessing.image import load_img
13. from tensorflow.keras.utils import to_categorical
14. from sklearn.preprocessing import LabelBinarizer
15. from sklearn.model_selection import train_test_split
16. from sklearn.metrics import classification_report
17. from imutils import paths
18. import matplotlib.pyplot as plt
19. import numpy as np
20. import argparse
21. import os
```

Set of tensorflow.keras imports returns:

- Data augmentation
- Loading the **MobilNetV2** classifier (fine-tune this model with pre-trained **ImageNet** weights).
- Building a new fully-connected (FC) head.
- Pre-processing
- Loading image data.


Use of scikit-learn aka sklearn for binarizing class labels, segmenting dataset and a printing a classification report.

By **imutils** paths implementation will help to find and list images in dataset, and use of **matplotlib** to plot training curves.

## Step 2.

construct the argument parser and parse the arguments in the command box.

eg. `python train_mask_detector.py --dataset <path/to/dataset>`



```
22. # make a argument parser in the terminal or command box.
23. ap = argparse.ArgumentParser()
24. ap.add_argument("-d", "--dataset", required=True,
25.                 help="path to input dataset")
26. ap.add_argument("-p", "--plot", type=str, default="acc_curves.png",
27.                 help="path to output loss/accuracy plot map")
28. ap.add_argument("-m", "--model", type=str,
29.                 default="classifier.model",
30.                 help="path to output face mask detector model")
31. args = vars(ap.parse_args())
```

command line arguments include:

- **--dataset:** The path to the input dataset of faces and faces with masks.
- **--plot:** The path to your output training history plot, which will be generated using Matplotlib.
- **--model:** The path to the resulting serialized face mask classification model.

## Step 3.

Initialize Hyperparameters constraints including **Initial Learning Rate**, number of training **Epochs** and **Batch Size**.

```
32. INIT_LR = 1e-4
33. EPOCHS = 20
34. BS = 32
```

## Step 4.

```
35. print("Loading images...")
36. imagePaths = list(paths.list_images(args["dataset"]))
37. data = []
38. labels = []
39.
40. # loop over the image paths
41. for imagePath in imagePaths:
42.     # extract the class label from the filename
43.     label = imagePath.split(os.path.sep)[-2]
44.     # load the input image (224x224) and preprocess it
45.     image = load_img(imagePath, target_size=(224, 224))
46.     image = img_to_array(image)
47.     image = preprocess_input(image)
```

- Grabbing all of the *imagepaths* in the dataset (line 36).

- Initialize *data* and *labels* lists (line 37-38).

```

48.         # update the data and labels lists, respectively
49.         data.append(image)
50.         labels.append(label)
51.
52.     # convert the data and labels to NumPy arrays
53.     data = np.array(data, dtype="float32")
54.     labels = np.array(labels)

```

- Looping over the *imagePaths* and loading + pre-processing images (Lines 41-50). Pre-processing steps include resizing to 224x224 pixels, conversion to array format, and scaling the pixel intensities in the input image to the range [-1, 1] (via the *preprocess\_input* convenience function)
- Appending the pre-processed *image* and associated *label* to the *data* and *labels* lists, respectively (Lines 49 and 50)
- Ensuring our training data is in NumPy array format (Lines 53 and 54).

```

55.     # perform one-hot encoding on the labels
56.     lb = LabelBinarizer()
57.     labels = lb.fit_transform(labels)
58.     labels = to_categorical(labels)
59.
60.     # partition the data into training and testing splits using 75% of
61.     # the data for training and the remaining 25% for testing
62.     #ratio of testing and training – 25%:75%
63.     (trainX, testX, trainY, testY) = train_test_split(data, labels,
64.                                                         test_size=0.20, stratify=labels, random_state=42)
65.
66.     # construct the training image generator for data augmentation
67.     aug = ImageDataGenerator(
68.         rotation_range=20,
69.         zoom_range=0.15,
70.         width_shift_range=0.2,
71.         height_shift_range=0.2,
72.         shear_range=0.15,
73.         horizontal_flip=True,
74.         fill_mode="nearest")
75.
76.     # load the MobileNetV2 network, ensuring the head FC layer sets are left off
77.     baseModel = MobileNetV2(weights="imagenet", include_top=False,
78.                             input_tensor=Input(shape=(224, 224, 3)))
79.
80.     # construct the head of the model that will be placed on top of the
81.     # the base model
82.     headModel = baseModel.output
83.     headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
84.     headModel = Flatten(name="flatten")(headModel)
85.     headModel = Dense(128, activation="relu")(headModel)
86.     headModel = Dropout(0.5)(headModel)
87.     headModel = Dense(2, activation="softmax")(headModel)

```

(line 56-58) one-hot encode our class labels, it accepts Categorical data as input and returns an Numpy array.

Using scikit-learn's convenience method, (Lines 63-64) segment our data into 80% training and the remaining 20% for testing.

During training, we'll be applying on-the-fly mutations to our images in an effort to improve generalization. This is known as data augmentation, where the random rotation, zoom, shear, shift, and flip parameters are established on Lines 77-84. We'll use the aug object at training time.

For **fine-tuning** used to prepare MobileNetV2.

Fine-tuning setup is a three-step process:

- Load MobileNet with pre-trained ImageNet weights, leaving off head of network (Lines 77-78)
- Construct a new FC head, and append it to the base in place of the old head (Lines 82-90)
- Freeze the base layers of the network (Lines 94-95). The weights of these base layers will not be updated during the process of backpropagation, whereas the head layer weights will be tuned.

```
88.
88.     # place the head FC model on top of the base model (this will become the actual model we will train)
89.     model = Model(inputs=baseModel.input, outputs=headModel)
90.
91.     # loop over all layers in the base model and freeze them so they will
92.     # *not* be updated during the first training process
93.     for layer in baseModel.layers:
94.         layer.trainable = False
95.     # compile our model
96.     print("[INFO] compiling model...")
97.     opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
98.     model.compile(loss="binary_crossentropy", optimizer=opt,
99.                  metrics=["accuracy"])
100.
101.     # train the head of the network
102.     print("[INFO] training head...")
103.     H = model.fit(
104.         aug.flow(trainX, trainY, batch_size=BS),
105.         steps_per_epoch=len(trainX) // BS,
106.         validation_data=(testX, testY),
107.         validation_steps=len(testX) // BS,
108.         epochs=EPOCHS)
109.
110.     # make predictions on the testing set
111.     print("[INFO] evaluating network...")
112.     predIdxs = model.predict(testX, batch_size=BS)
113.
114.     # for each image in the testing set we need to find the index of the
115.     # label with corresponding largest predicted probability
116.     predIdxs = np.argmax(predIdxs, axis=1)
```

Lines 98-100 *compile* our model with the *Adam* optimizer, a learning rate decay schedule, and binary cross-entropy. If you're building from this training script with > 2 classes, be sure to use categorical cross-entropy.

Face mask training is launched via Lines 104-109. Notice how our data augmentation object (*aug*) will be providing batches of mutated image data.

Once training is complete, we'll evaluate the resulting model on the test set. In Lines 113-117 make predictions on the test set, grabbing the highest probability class label indices. Then, we print a classification report in the terminal for inspection.

Line 126 serializes our face mask classification model to disk.

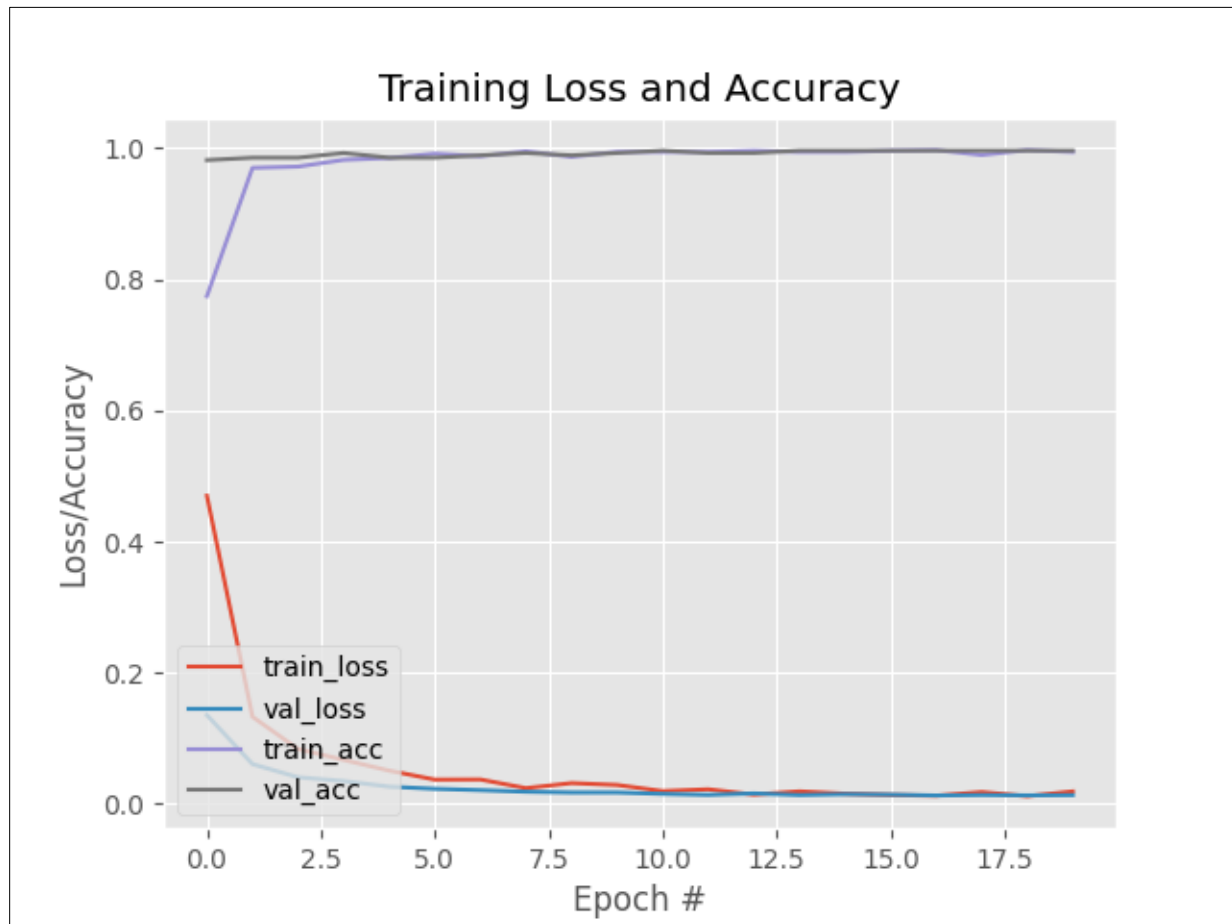
```
118.  
117.     # show a nicely formatted classification report  
118.     print(classification_report(testY.argmax(axis=1), predIdxs,  
119.                               target_names=lb.classes_))  
120.  
121.     # serialize the model to disk  
122.     print("[INFO] saving mask detector model...")  
123.     model.save(args["model"], save_format="h5")  
124.  
125.
```

## Step 5:

Last step to plot accuracy and loss curves of this model. Once plot is ready it will auto save to Disk self-using *--plot* (Line 26, Step 2) file-path.

```
126.     # plot the training loss and accuracy  
127.     N = EPOCHS  
128.     plt.style.use("ggplot")  
129.     plt.figure()  
130.     plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")  
131.     plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")  
132.     plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")  
133.     plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")  
134.     plt.title("Training Loss and Accuracy")  
135.     plt.xlabel("Epoch #")  
136.     plt.ylabel("Loss/Accuracy")  
137.     plt.legend(loc="lower left")  
138.     plt.savefig(args["plot"])
```

## OUTPUT OF PLOT DIAGRAM



Plot diagram of accuracy and curves of face mask detection model

## MODULE NO. 2 [ detection.py ]

### Step 1:

Import all required libraries and one selfmade module known as **.conf**. this python file contains some global values like *NMS\_THRES* – threshold value, *MIN\_CONF*, *People\_Counter* at same path.

1. `# import the necessary packages`
2. `from .config import NMS_THRESH, MIN_CONF, People_Counter`
3. `import numpy as np`
4. `import cv2`



```

5. def detect_people(frame, net, ln, personIdx=0):
6.     # grab the dimensions of the frame and initialize the list of results
7.     (H, W) = frame.shape[:2]
8.     results = []
9.
10.    # construct a blob from the input frame and then perform a forward
11.    # pass of the YOLO object detector, giving us our bounding boxes and associated probabilities
12.    blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),
13.                                  swapRB=True, crop=False)
14.    net.setInput(blob)
15.    layerOutputs = net.forward(ln)
16.
17.    # initialize our lists of detected bounding boxes, centroids, and confidences, respectively
18.    boxes = []
19.    centroids = []
20.    confidences = []
21.
22.    # loop over each of the layer outputs
23.    for output in layerOutputs:
24.        # loop over each of the detections
25.        for detection in output:
26.            # extract the class ID and confidence (i.e., probability)
27.            # of the current object detection
28.            scores = detection[5:]
29.            classID = np.argmax(scores)
30.            confidence = scores[classID]
31.
32.            # filter detections by (1) ensuring that the object
33.            # detected was a person and (2) that the minimum confidence is met
34.            if classID == personIdx and confidence > MIN_CONF:
35.                # scale the bounding box coordinates back relative to
36.                # the size of the image, keeping in mind that YOLO
37.                # actually returns the center (x, y)-coordinates of
38.                # the bounding box followed by the boxes' width and height
39.                box = detection[0:4] * np.array([W, H, W, H])
40.                (centerX, centerY, width, height) = box.astype("int")
41.
42.                # use the center (x, y)-coordinates to derive the top
43.                # and left corner of the bounding box
44.                x = int(centerX - (width / 2))
45.                y = int(centerY - (height / 2))
46.
47.                # update our list of bounding box coordinates,
48.                # centroids, and confidences
49.                boxes.append([x, y, int(width), int(height)])
50.                centroids.append((centerX, centerY))
51.                confidences.append(float(confidence))

```

```

52.         # apply non-maxima suppression to suppress weak, overlapping
53.         # bounding boxes
54.         idxs = cv2.dnn.NMSBoxes(boxes, confidences, MIN_CONF, NMS_THRESH)
55.         #print("Total people count:", len(idxs))
56.         # compute the total people counter
57.         if People_Counter:
58.             human_count = "Person count: {}".format(len(idxs))
59.             cv2.putText(frame, human_count, (frame.shape[0] - 170, 50),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 0, 255), 2)
60.
61.         # ensure at least one detection exists
62.         if len(idxs) > 0:
63.             # loop over the indexes we are keeping
64.             for i in idxs.flatten():
65.                 # extract the bounding box coordinates
66.                 (x, y) = (boxes[i][0], boxes[i][1])
67.                 (w, h) = (boxes[i][2], boxes[i][3])
68.
69.                 # update our results list to consist of the person
70.                 # prediction probability, bounding box coordinates, and the centroid
71.                 r = (confidences[i], (x, y, x + w, y + h), centroids[i])
72.                 results.append(r)
73.
74.         # return the list of results
75.         return results

```

This Module **detection.py** is made for to count the total persons in a frame and returns total number to output display.

### MODULE NO. 3 [ main.py ]

#### Step 1:

Imports all required libraries in this module.

Such that tensorflow, Keras, numpy and SciPy. SciPy is used to compute distance using **Euclidean Distance**.

```

1.     from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
2.     from tensorflow.keras.preprocessing.image import img_to_array
3.     from tensorflow.keras.models import load_model
4.     import numpy as np
5.     import imutils
6.     import time
7.     import cv2
8.     import math
9.     from modules.detection import detect_people
10.    from scipy.spatial import distance as dist

```

## Step 2:

There is a file called *coco.names* (*Pre-Trained model*) that has the list of 80 object class that the model will be able to detect. The model has been trained only on these 80 object classes.

Call and Open pre-trained model (Lines 12-13), similarly set path of YOLO Version3 pre-trained weights of neural network i.e. *.weights* and neural network architecture model i.e. *.cfg* file to *weightsPath* and *ConfigPath* respectively (Line 20-21).

*readNetFromDarknet* returns Network object that ready to do forward, throw an exception in failure cases(Lines 23), and to assign the pixel values in relation to the threshold value is 0.4.

```
11.
12. labelsPath = "yolo-coco/coco.names"
13. LABELS = open(labelsPath).read().strip().split("\n")
14. np.random.seed(42)
15. COLORS = np.random.randint(0,
16.                             255,
17.                             size=(len(LABELS), 3),
18.                             dtype="uint8")
19.
20. weightsPath = "yolo-coco/yolov3.weights"
21. configPath = "yolo-coco/yolov3.cfg"
22.
23. net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
24.
25. # face mask classification
26. confidence_threshold = 0.4
27.
28. if True:
29.     # set CUDA as the preferable backend and target
30.     print("")
31.     print("[INFO] Looking for GPU")
32.     net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
33.     net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)
34.
35. # load our serialized face detector model from disk
36. print("loading face detector model...")
37. prototxtPath = "models/deploy.prototxt"
38. weightsPath = "models/res10_300x300_ssd_iter_140000.caffemodel"
39. faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
40.
41. # load the face mask detector model from disk
42. model_store_dir= "models/classifier.model"
43. maskNet = load_model(model_store_dir)
```

Load the GPU for backend support (Lines 28-33). Call all pre-trained face detect models *deploy\_prototxt* & *caffemodel*.

### Step 3:

```
44. cap = cv2.VideoCapture(0)    #Start Video Streaming
45. while (cap.isOpened()):
46.     ret, image = cap.read()
47.
48.     if ret == False:
49.         break
50.
51.     image = cv2.resize(image, (720, 640))
52.     # cv2.namedWindow("output", cv2.WINDOW_NORMAL)
53.
54.     (H, W) = image.shape[:2]
55.     ln = net.getLayerNames()
56.     ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]
57.     blob = cv2.dnn.blobFromImage(image,
58.                                   1/255.0,
59.                                   (416, 416),
60.                                   swapRB=True,
61.                                   crop=False)
62.
63.     results = detect_people(image, net, ln,
64.                             personIdx=LABELS.index("person"))
65.
66.     net.setInput(blob)
67.     start = time.time()
68.     layerOutputs = net.forward(ln)
69.     end = time.time()
70.     print("Time taken to predict the image: {:.6f}seconds".format(end-start))
71.     boxes = []
72.     confidences = []
73.     classIDs = []
74.
75.     for output in layerOutputs:
76.         for detection in output:
77.             scores = detection[5:]
78.             classID = np.argmax(scores)
79.             confidence = scores[classID]
80.             if confidence > 0.1 and classID == 0:
81.                 box = detection[0:4] * np.array([W, H, W, H])
82.                 (centerX, centerY, width, height) = box.astype("int")
83.                 x = int(centerX - (width / 2))
84.                 y = int(centerY - (height / 2))
85.                 boxes.append([x, y, int(width), int(height)])
86.                 confidences.append(float(confidence))
87.                 classIDs.append(classID)
88.     idxs = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.3)
89.     ind = []
90.     for i in range(0, len(classIDs)):
91.         if (classIDs[i] == 0):
92.             ind.append(i)
93.
```

To Start the videoStreaming set the path of Camera main camera value is '0' for System inbuilt Camera, and to connect with other out source of camera such as external camera like CCTV and webcam set the value in *VideoCapture()* is 1,2.. as the no. connected of cameras.

Start VideoStreaming to load real-time images to detect the face and check to maintain social detecting. After Video Streaming it set size of output windows to 720x640 and for analyzing an image that has undergone binarization processingsuch as the presence, number, area, position, length, and direction of lumps (Lines 44-61).

```

94.     a = []
95.     b = []
96.     if len(idxs) > 0:
97.         for i in idxs.flatten():
98.             (x, y) = (boxes[i][0], boxes[i][1])
99.             (w, h) = (boxes[i][2], boxes[i][3])
100.            a.append(x)
101.            b.append(y)
102.
103.    distance = []
104.    nsd = []
105.    for i in range(0, len(a) - 1):
106.        for k in range(1, len(a)):
107.            if (k == i):
108.                break
109.            else:
110.                x_dist = (a[k] - a[i])
111.                y_dist = (b[k] - b[i])
112.                d = math.sqrt(x_dist * x_dist + y_dist * y_dist)
113.                distance.append(d)
114.                if (d <= 100.0):
115.                    nsd.append(i)
116.                    nsd.append(k)
117.                nsd = list(dict.fromkeys(nsd))
118.
119.    color = (0, 0, 255)
120.    for i in nsd:
121.        (x, y) = (boxes[i][0], boxes[i][1])
122.        (w, h) = (boxes[i][2], boxes[i][3])
123.        cv2.rectangle(image, (x, y), (x + w, y + h), color, 1)
124.        text = "Alert"
125.        cv2.putText(image, text, (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 1)
126.        cv2.putText(image, text, (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 1)

```

Initialize a & b lists (Line 95-96) and append all points such as (x, y), (w, h) to x and y. after calculate the distance between 2 points using formulae  $\sqrt{x^2 + y^2}$  . and convert it into list from dictionary.

For every list elements put a rectangle and text for detecting the faces with their result that they wear mask or not. If yes, then both rectangle and text will display in Green color else in red color i.e. Alert.

```

127.     color = (138, 68, 38)
128.     if len(idxs) > 0:
129.         for i in idxs.flatten():
130.             if (i in nsd):
131.                 break
132.             else:
133.                 (x, y) = (boxes[i][0], boxes[i][1])
134.                 (w, h) = (boxes[i][2], boxes[i][3])
135.                 cv2.rectangle(image, (x, y), (x + w, y + h), color, 1)
136.                 text = "SAFE"
137.                 cv2.putText(image, text, (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 1)
138.
139.     (h, w) = image.shape[:2]
140.     blob = cv2.dnn.blobFromImage(image, 1.0, (416, 416), (104.0, 177.0, 123.0))
141.
142.     faceNet.setInput(blob)
143.     detections = faceNet.forward()
144.
145.     for i in range(0, detections.shape[2]):
146.         confidence = detections[0, 0, i, 2]
147.
148.         if confidence > confidence_threshold:
149.             box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
150.             (startX, startY, endX, endY) = box.astype("int")
151.
152.             (startX, startY) = (max(0, startX), max(0, startY))
153.             (endX, endY) = (min(w - 1, endX), min(h - 1, endY))
154.
155.             face = image[startY:endY, startX:endX]
156.             face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
157.             face = cv2.resize(face, (224, 224))
158.             face = img_to_array(face)
159.             face = preprocess_input(face)
160.             face = np.expand_dims(face, axis=0)

```

construct a blob, detect faces, and initialize lists, two of which the function is set to return. These lists include our faces (i.e., ROIs).

Inside the loop, we filter out weak detections (Lines 146-149) and extract bounding boxes while ensuring bounding box coordinates do not fall outside the bounds of the image (Lines 150-154).

After extracting face ROIs and pre-processing (Lines 156-161), it will expand the shape of an array. Insert a new axis that will appear at the axis position in the expanded array shape.

```

161.      # ----- Counting Sources -----
162.      serious = set()
163.      abnormal = set()
164.
165.      # there are *at least* two people detections (required in
166.      # order to compute our pairwise distance maps)
167.      if len(results) >= 2:
168.          # extract all centroids from the results and compute the
169.          # Euclidean distances between all pairs of the centroids
170.          centroids = np.array([r[2] for r in results])
171.          D = dist.cdist(centroids, centroids, metric="euclidean")
172.
173.          # loop over the upper triangular of the distance matrix
174.          for i in range(0, D.shape[0]):
175.              for j in range(i + 1, D.shape[1]):
176.                  # check to see if the distance between any two
177.                  # centroid pairs is less than the configured number of pixels
178.                  if D[i, j] < 50:
179.                      # update our violation set with the indexes of the centroid pairs
180.                      serious.add(i)
181.                      serious.add(j)
182.                      # update our abnormal set if the centroid distance is below max distance limit
183.                      if (D[i, j] < 80) and not serious:
184.                          abnormal.add(i)
185.                          abnormal.add(j)
186.
187.          for (i, (prob, bbox, centroid)) in enumerate(results):
188.              # extract the bounding box and centroid coordinates, then
189.              # initialize the color of the annotation
190.              (startX, startY, endX, endY) = bbox
191.              (cX, cY) = centroid
192.              color = (0, 255, 0)
193.
194.              # if the index pair exists within the violation/abnormal sets, then update the color
195.              if i in serious:
196.                  color = (0, 0, 255)
197.              elif i in abnormal:
198.                  color = (0, 255, 255) #orange = (0, 165, 255)
199.
200.              # draw (1) a bounding box around the person and (2) the
201.              # centroid coordinates of the person,
202.              cv2.circle(image, (cX, cY), 2, color, 2)

```

To detect and calculate total persons in a frame and who maintain the social distancing at a time there are **at least** two people detections (required in order to compute our pairwise distance maps). extract all centroids from the results and compute the Euclidean distances between all pairs of the centroids (Line 168-186).

To evaluate the distance between two or more persons I've used distance differentiate method i.e., **Euclidean Distance Method**.

$$D(Q, P) = \sqrt{\sum_{i=1}^n (Q_i - P_i)^2}$$

And find the centroid of each detected person to match the safe distance as well.

```
203.         (mask, without_mask) = maskNet.predict(face)[0]
204.         label = "Mask" if mask > without_mask else "No Mask"
205.         color = (0, 255, 0) if label == "Mask" else (0, 0, 255)
206.
207.         label = "{: {:.2f}%}".format(label, max(mask, without_mask) * 100)
208.         c = cv2.putText(image, label, (startX, startY - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 1)
209.         x = cv2.rectangle(image, (startX, startY), (endX, endY), color, 1)
210.         text = "Total serious violations: {}".format(len(serious))
211.         cv2.putText(image, text,
212.                     (10, image.shape[0] - 55),
213.                     cv2.FONT_HERSHEY_SIMPLEX, 0.70,
214.                     (0, 0, 255), 2)
215.
216.         text1 = "Total abnormal violations: {}".format(len(abnormal))
217.         cv2.putText(image, text1,
218.                     (10, image.shape[0] - 25),
219.                     cv2.FONT_HERSHEY_SIMPLEX, 0.70,
220.                     (0, 255, 255), 2)
221.
222.         print("End of classifier")
223.         #Display the output
224.         cv2.imshow("Image", image)
225.         if cv2.waitKey(1) & 0xFF == ord('q'):
226.             break
227.
228.     cap.release()
229.     cv2.destroyAllWindows()
```

Finally, it display the results and perform clean up.

After the *frame* is displayed, we capture *key* presses. If the user presses 'q' (quit), we *break* out of the loop and perform housekeeping.



## Screenshots of Result:

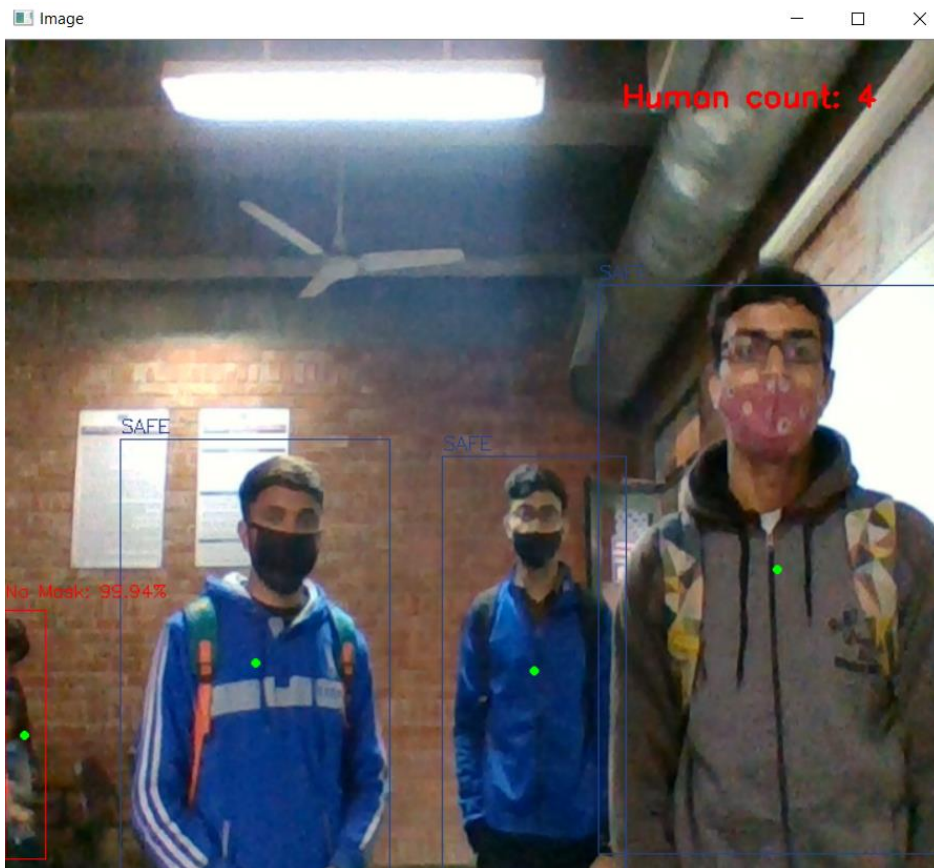
1. After launch the model using TensorFlow Environment, command for terminal/Command Prompt.

python main.py

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

E:\CodeMork\Machine Learning\Facemask\Social Distancing and Face Mask Detection>conda activate tf_python
(tf_python) E:\CodeMork\Machine Learning\Facemask\Social Distancing and Face Mask Detection>python main.py
2021-01-02 15:57:24.997356: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cudart64_110.dll
[INFO] Looking for GPU
loading face detector model...
2021-01-02 15:57:35.722394: I tensorflow/compiler/jit/xla_cpu_device.cc:41] Not creating XLA devices, tf_xla_enable_xla_devices not set
2021-01-02 15:57:35.728622: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library nvcuda.dll
2021-01-02 15:57:36.164498: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1720] Found device 0 with properties:
pciBusID: 0000:01:00.0 name: GeForce RTX10 computeCapability: 5.0
coreClock: 1.0006GHz coreCount: 2 deviceMemorySize: 2.0001GB deviceMemoryBandwidth: 17.1361GB/s
2021-01-02 15:57:36.164701: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cudart64_110.dll
2021-01-02 15:57:36.987998: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cublas64_11.dll
2021-01-02 15:57:36.988255: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cublas64_11.dll
2021-01-02 15:57:36.979702: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cufft64_10.dll
2021-01-02 15:57:36.982268: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library curand64_10.dll
2021-01-02 15:57:37.069081: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cusolver64_10.dll
2021-01-02 15:57:37.341724: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cusparse64_11.dll
2021-01-02 15:57:37.373472: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cudnn64_8.dll
2021-01-02 15:57:37.555536: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1862] Adding visible gpu devices: 0
2021-01-02 15:57:37.559289: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in perf
ormance-critical operations: AVX2
To enable then in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-01-02 15:57:37.561142: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1720] Found device 0 with properties:
pciBusID: 0000:01:00.0 name: GeForce RTX10 computeCapability: 5.0
coreClock: 1.0006GHz coreCount: 2 deviceMemorySize: 2.0001GB deviceMemoryBandwidth: 17.1361GB/s
2021-01-02 15:57:37.561286: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cudart64_110.dll
2021-01-02 15:57:37.561526: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cublas64_11.dll
2021-01-02 15:57:37.561627: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cufft64_10.dll
2021-01-02 15:57:37.561739: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library curand64_10.dll
2021-01-02 15:57:37.561856: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cusolver64_10.dll
2021-01-02 15:57:37.561908: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cusparse64_11.dll
2021-01-02 15:57:37.562063: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cudnn64_8.dll
2021-01-02 15:57:37.562205: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1862] Adding visible gpu devices: 0
2021-01-02 15:57:40.041252: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1261] Device interconnect StreamExecutor with strength 1 edge matrix:
2021-01-02 15:57:40.041418: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1267] 0
2021-01-02 15:57:40.041684: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1280] 0: N
2021-01-02 15:57:40.044605: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1406] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 1373 MB memory) -> physical GPU (device: 0, name:
GeForce RTX10, pci bus id: 0000:01:00.0, compute capability: 5.0)
2021-01-02 15:57:40.049915: I tensorflow/compiler/jit/xla_cpu_device.cc:99] Not creating XLA devices, tf_xla_enable_xla_devices not set
[ WARN:0] global C:\Users\appveyor\AppData\Local\Temp\1\pip-req-build-5201relq\opencv\modules\dnn\src\dmn.cpp (1429) cv::dnn::dnn_v20200609::Net::Impl::setUpNet DNN module was not built with CUDA backend; switc
hing to CPU
Time taken to predict the image: 0.741835seconds
Time taken to predict the image: 0.456779seconds
[ WARN:1] global C:\Users\appveyor\AppData\Local\Temp\1\pip-req-build-5201relq\opencv\modules\videoio\src\cap_msf.cpp (435) 'anonymous namespace':SourceReaderCB::SourceReaderCB terminating async callback
(tf_python) E:\CodeMork\Machine Learning\Facemask\Social Distancing and Face Mask Detection>
```

2. Frame will pop up with the name of image.



## Separate Outputs from Social Distancing and Face Mask Detection:

Social Distancing Detector:



Distancing Detector: Shows red Block – Alert to maintain the distance

Face Mask Detector:



Face Mask Detector: Show red block – Alert to wear face mask



After combining both outputs:



It shows both social Distancing and face mask alert message

## CHAPTER – 8

## CONCLUSION

Corporate giants from various verticals are turning to AI and ML, leveraging technology at the service of humanity amid the pandemic. Digital product development companies are launching mask detection API services that enable developers to build a face mask detection system quickly to serve the community amid the crisis.

we proposed an approach that uses computer vision and MobileNet V2, SSD architecture to help maintain a secure environment and ensure individuals protection by automatically monitoring public places to avoid the spread of the COVID-19 virus and assist police by minimizing their physical surveillance work in containment zones and public areas where surveillance is required by means of camera feeds The technology assures reliable and real-time face detection of users wearing masks. Besides, the system is easy to deploy into any existing system of a business while keeping the safety and privacy of users' data. this proposed system will operate in an efficient manner in the current situation when the lockout is eased and helps to track public places easily in an automated manner. We have addressed in depth the tracking of social distancing and

the identification of face masks that help to ensure human health. So, the Social Distancing and face mask detection system is going to be the leading digital solution for most industries, especially retail, healthcare, temples, shopping complex, metro stations, airports and corporate sectors.[1]

## CHAPTER – 10

### FUTURE SCOPE

Objects are usually recognized by their unique features. There are many features in a human face, which can be recognized between a face and many other objects. It locates faces by extracting structural features like eyes, nose, mouth etc. and then uses them to detect a face. Typically, some sort of statistical classifier qualified then helpful to separate between facial and non-facial regions.[1],[3]

Human faces have particular textures which can be used to differentiate between a face and other objects. Moreover, the edge of features can help to detect the objects from the face.

The above-mentioned use cases are only some of the many features that were incorporated as part of this solution. We assume there are several other cases of usage that can be included in this solution to offer a more detailed sense of safety. Several of the currently under development features are listed below in brief:

1. **Coughing and Sneezing Detection:** Chronic coughing and sneezing is one of the key symptoms of COVID-19 infection as per WHO guidelines and also one of the major routes of disease spread to non-infected public. Deep learning-based approach can be proved handy here to detect & limit the disease spread by enhancing our proposed solution with body gesture analysis to understand if an individual is coughing and sneezing in public places while breaching facial mask and social distancing guidelines and based on outcome enforcement agencies can be alerted.
2. **Temperature Screening:** Elevated body temperature is another key symptom of COVID-19 infection, at present scenario thermal screening is done using handheld contactless IR thermometers where health worker need to come in close proximity with the person need to be screened which makes the health workers vulnerable to get infected and also its practically impossible to capture temperature for each and every person in public places, the proposed use-case can be equipped with thermal cameras based screening to analyze body temperature of the peoples in public places that can add another helping hand to enforcement agencies to tackle the pandemic effectively.[1],[2]

## REFERENCES

- |   |   |
|---|---|
| <p>[1] Yadav, S. (2020). Deep Learning based Safe Social Distancing and Face Mask Detection in Public Areas for COVID19 Safety Guidelines Adherence. <i>IJRASET</i>, 8(VII), 4.<br/>doi:10.22214/ijraset.2020.30560</p> <p>[2] Sohan, M. (2020). <i>So you need datasets for your COVID-19 Detection Research using Machine Learning</i>. ArXiv.<br/>doi:abs/2008.05906</p> | <p>[3] M. Cristani, A. D. Bue, V. Murino, F. Setti and A. Vinciarelli, "The Visual Social Distancing Problem," in <i>IEEE Access</i>, vol. 8, pp. 126876- 126886, 2020, doi: 10.1109/ACCESS.2020.3008370</p> <p>[4] A. H. Ahamad, N. Zaini and M. F. A. Latip, "Person Detection for Social Distancing and Safety Violation Alert based on Segmented ROI," 2020 10th IEEE International Conference on Control</p> |
|---|---|

- System, Computing and Engineering (ICCSCE), Penang, Malaysia, 2020, pp. 113-118, doi: 10.1109/ICCSCE50387.2020.9204934.*
- [5] S. Sharma, M. Bhatt and P. Sharma, "Face Recognition System Using Machine Learning Algorithm," 2020 5th International Conference on Communication and Electronics Systems (ICCES), COIMBATORE, India, 2020, pp. 1162-1168, doi: 10.1109/ICCES48766.2020.9137850
- [6] D. Garg, P. Goel, S. Pandya, A. Ganatra and K. Kotecha, "A Deep Learning Approach for Face Detection using YOLO," 2018 IEEE Punecon, Pune, India, 2018, pp. 1-4, doi: 10.1109/PUNECON.2018.8745376.
- [7] H. Filali, J. Riffi, A. M. Mahraz and H. Tairi, "Multiple face detection based on machine learning," 2018 International Conference on Intelligent Systems and Computer Vision (ISCV), Fez, 2018, pp. 1-8, doi: 10.1109/ISACV.2018.8354058
- [8] GPU in Tensorflow, Software Requirements, <https://www.tensorflow.org/install/gpu> Information Networking and Applications Workshops (WAINA), Taipei, 2017, pp. 514-518, doi: 10.1109/WAINA.2017.135.
- [9] Cuda Versions, NVIDIA CUDA toolkits, <https://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html> ease-notes/index.html