CONTENTS

What have we learnt during this training?

1. Basic of python: Addition, Subtraction, multiplication, condition, loop, function, String handling, Morse code, Hashing : Chaining, Linear probing

2.1. List, Tuple, Set and Dictionary

2.2. Pandas

2.3. Numpy

3.1 Basic of machine learning: What is Machine learning, Supervised, Unsupervised, Reinforcement Learning,

parameters, Effect of parameter in the model as well as data set, hyperplane, classifier, clustering.

3.2 Supervised learning

1. Entropy, information gain, Residual information, tree generation.
2. KNN (KNN solve)
3. D Tree

a) Naive-Bayes
b) SVM

3.3 Unsupervised:

1. KMeans (With example)
2. DBSCAN (With example)
3. Cross validation
4. Mathematical definition of confusion matrix, accuracy, precision, recall, F-Score

4. Data visualization using Matplotlib and Graphviz

1. CSV to bar graph
2. Several bar graph
3. Scatter plot
4. Line plot
5. Subplot
6. Customize the label
7. Tics
8. Axes
9. Colour code
10. Pie chart

5. Several machine learning algorithms:

1) KNN solve
2) D Tree

a) Naive-Bayes
b) SVM
c) KMeans

Training, testing, pre-processing (encoding), model fitting, prediction, confusion matrix, accuracy, precision, recall, F-Score

View the model and result.

6. Image processing with OpenCV:

read, write, show image, properties of images, resize, line, circle, text in image, thresholding, masking, colour detection, image disnoise, smoothing, erosion, dilation, opening, closing, canny edge detection, filtering, machine learning in image processing (KMeans).

7. Web scraping using requests and bs4:

web page parsing from URL, use of different soup.function(), create CSV from website (from Amazon, IMDB), Movie recommendation system

Problem Statement

The objective of this project is to detect whether a person is wearing a mask or not.

This project is carried out in Python Programming language.

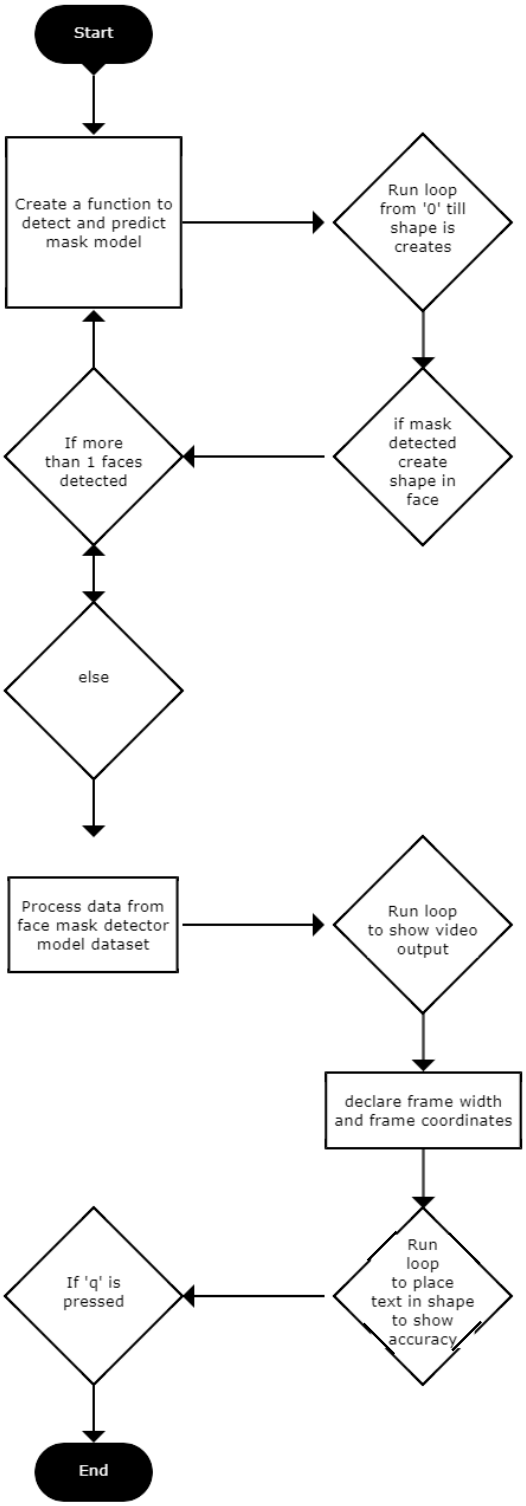
Goals of this project:

1. Real time image processing
2. Face detection
3. Mask detection
4. Model accuracy
5. Set training and testing datasets
6. Mark detected area with green if mask is detected
7. Mark detected area with red if mask is not detected
8. Show accuracy of the model
9. Show real time Video Stream
10. Close Video Stream if ‘q’ is pressed

ALGORITHM

STEP 1 – START
STEP 2 – Create a function **detect_and_predict_mask**
STEP 3 – Create empty lists **faces[], locs[], preds[]**
STEP 4 – Run a loop from **0** till **shape creates**
STEP 5 – Check condition whether **face** is detected
STEP 6 – If face detected then check how many
STEP 7 – If more than one faces detected go back to **Step 4** for each face
STEP 8 – else return **locs** and **preds** and save to **STEP 3**
STEP 9 – exit loop
STEP 10 – Compare found face with pre trained **face mask detector model**
STEP 11 – Detect accuracy of the model
STEP 12 – Print **[INFO] loading face detector model...**
STEP 13 – After comparison done print **[INFO] starting video stream...**
STEP 14 – Create a delay of **2 seconds**
STEP 15 – Run a while loop condition becomes **true**
STEP 16 – Declare video width
STEP 17 – load data from **STEP 3**
STEP 18 – Run a for loop till **preds** are placed in **locs**
STEP 19 – Create a label **Mask** if mask **is detected**
STEP 20 – Create a label **No Mask** if mask **is not detected**
STEP 21 – Set colour of label **Mask** with **Green** and **No Mask** with **Red**
STEP 22 – place label in face locations from data in **locs**
STEP 23 – Set font family as **FONT_HERSHEY_SIMPLEX**
STEP 24 – Start Video Streaming
STEP 25 – Check condition if key ‘q’ is pressed
STEP 26 – Exit loop
STEP 27 – Stop Video Streaming
STEP 28 – Destroy all windows
STEP 28 – EXIT Program

FLOWCHART


Start
Create a function to detect and predict mask model
Run loop from '0' till shape is creates
if mask detected create shape in face
If more than 1 faces detected
else
Process data from face mask detector model dataset
Run loop to show video output
declare frame width and frame coordinates
Run loop to place text in shape to show accuracy
If 'q' is pressed
End

MATHEMATICAL EXPLANATION

Masking

A mask is a binary image consisting of zero- and non-zero values. If a mask is applied to another binary or to a grayscale image of the same size, all pixels which are zero in the mask are set to zero in the output image. All others remain unchanged.

Masking can be implemented either using pixel multiplication or logical AND, the latter in general being faster.

Masking is often used to restrict a point or arithmetic operator to an area defined by the mask. We can, for example, accomplish this by first masking the desired area in the input image and processing it with the operator, then masking the original input image with the inverted mask to obtain the unprocessed area of the image and finally recombining the two partial images using image addition. An example can be seen in the worksheet on the logical AND operator. In some image processing packages, a mask can directly be defined as an optional input to a point operator, so that automatically the operator is only applied to the pixels defined by the mask.

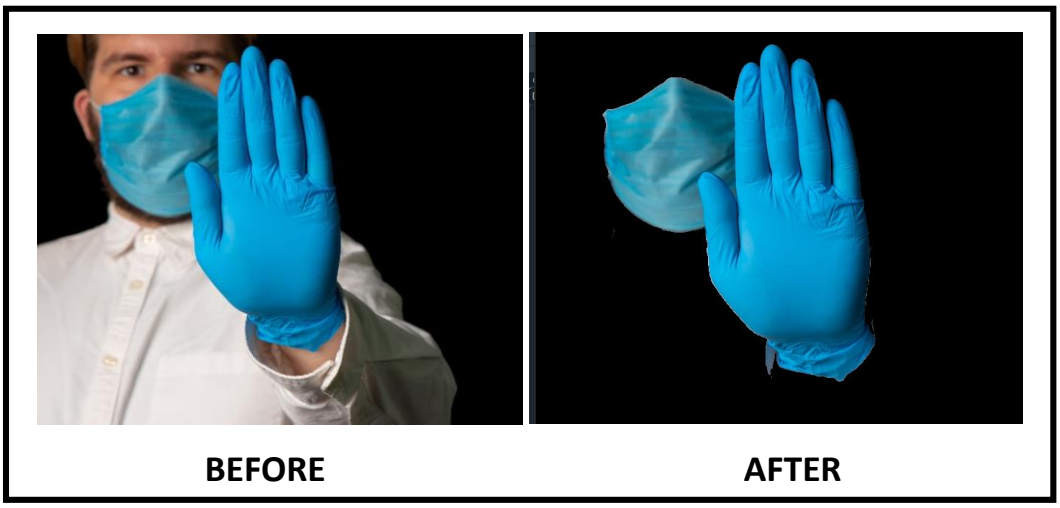


Figure 1

Figure 1 shows example of masking.

To apply a mask on an image, filter mask is moved point to point on the image. In the original image, at each point (X, Y), filter is calculated by using a predefined relationship.

There are two types of filters:

1. Linear filter
2. Frequency domain filter

Linear filter

A linear filter is the simplest filter. In linear filter, each pixel is replaced by the average of these pixel values. The entire linear filter works in the same way except when the weighted average is formed instead of a simple average.

The formula for a linear filter

$$g_{ij} = \sum_{k=-m}^{m} \sum_{l=-m}^{m} w_{kl} f_{i+k,j+l} \qquad \text{for } i,j = (m+1), \ldots, (n-m).$$

Example:

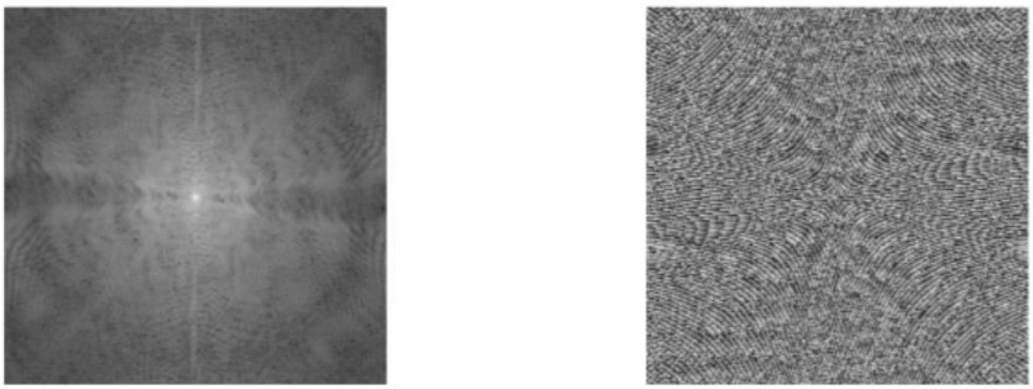
$$\begin{aligned} g_{ij} = \; & w_{-1,-1} \; f_{i-1,j-1} + w_{-1,0} \; f_{i-1,j} + w_{-1,1} \; f_{i-1,j+1} \\ & + w_{0,-1} \; f_{i,j-1} + w_{0,0} \; f_{i,j} + w_{0,1} \; f_{i,j+1} \\ & + w_{1,-1} \; f_{i+1,j-1} + w_{1,0} \; f_{i+1,j} + w_{1,1} \; f_{i+1,j+1}. \end{aligned}$$

Frequency Domain Filter

In frequency domain filter, an image is represented as the sum of many sine waves which have different frequencies, amplitudes and directions. The parameter of sine waves is referred to as Fourier coefficients.

Reasons for using this approach:

1. For getting extra insight.
2. Linear filters can also in the frequency domain use Fast Fourier Transform (FFT)



Fourier transform of X-ray image and phases from an X-ray image

Filters are used for 2 purposes:

1. Blurring and noise reduction.
2. Edge detection and sharpness.

Blurring and noise reduction

Filters can be used for blurring as well as noise reduction from an image. Blurring is used to remove small details from an image. Noise reduction can also be done with the help of blurring.

Commonly used masks for blurring are:

1. Box filter
2. Weighted average filter

Edge Detection and sharpness

Filters can be used for edge detection and sharpness. To increase the sharpness of an image, edge detection is used.

Contours to Hierarchical Regions:

We consider a contour detector, whose output E (x, y, θ) predicts the probability of an image boundary at location (x, y) and orientation θ. We build hierarchical regions by exploiting the information in this contour signal using a sequence of two transformations, the Oriented Watershed Transform (OWT) and Ultrametric Contour Map (UCM), detailed below

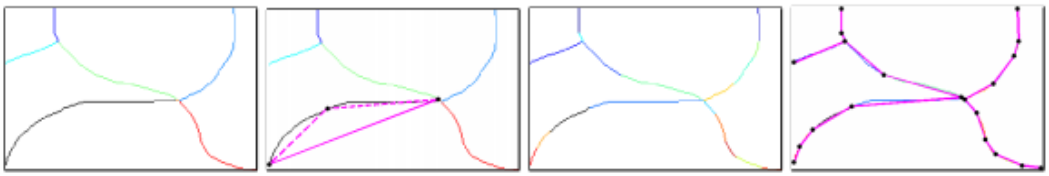


Figure 2

Figure 2: Contour subdivision. **Left:** Initial arcs color-coded. **Middle Left:** For each arc, we consider the straight line segment connecting its endpoints. If the distance from any point on the arc to this line segment is greater than a fixed fraction of the segment length, we subdivide the arc at the maximally distant point. An example is shown for one arc, with the dashed segments indicating the new subdivision. **Middle Right:** The final set of arcs resulting from recursive application of the scale-invariant subdivision procedure. **Right:** Approximating straight line segments overlaid on the subdivided arcs.

Explanation:

To correct this problem, we enforce consistency between the strength of the boundaries of K0 and the underlying E (x, y, θ) signal in a modified procedure, which we call the Oriented Watershed Transform (OWT). As the first step in this reweighting process, we estimate an orientation at each pixel on an arc from the local geometry of the arc itself. These orientations are obtained by approximating the watershed arcs with line segments as shown in ***Figure 2***. We recursively subdivide any arc which is not well fit by the line segment connecting its endpoints. By expressing the approximation criteria in terms of the maximum distance of a point on the arc from the line segment as a fraction of the line segment length, we obtain a scale-invariant subdivision. We assign each pixel (x, y) on a subdivided arc the orientation o (x, y) ∈ [0, π) of the corresponding line segment.

Next, we use the oriented contour detector output E (x, y, θ), to assign each arc pixel (x, y) a boundary strength of E (x, y, o (x, y)). Here we quantize o (x, y) in the same manner as θ, so this operation is a simple lookup. Finally, each original arc in K0 is assigned weight equal to average boundary strength of the pixels it contains.

Python datatype used for this project

Libraries used:

1. Tensorflow
2. Keras
3. Imutils
4. Numpy
5. Argparse
6. Time
7. OpenCV
8. OS

Datatype Used:

1. List
2. String
3. Array

How to install libraries:

In python shell run commands:

1. *pip install numpy*
2. *pip install tensorflow*
3. *pip install keras*
4. *pip install argparse*
5. *pip install opencv-python*
6. *pip install imutils*

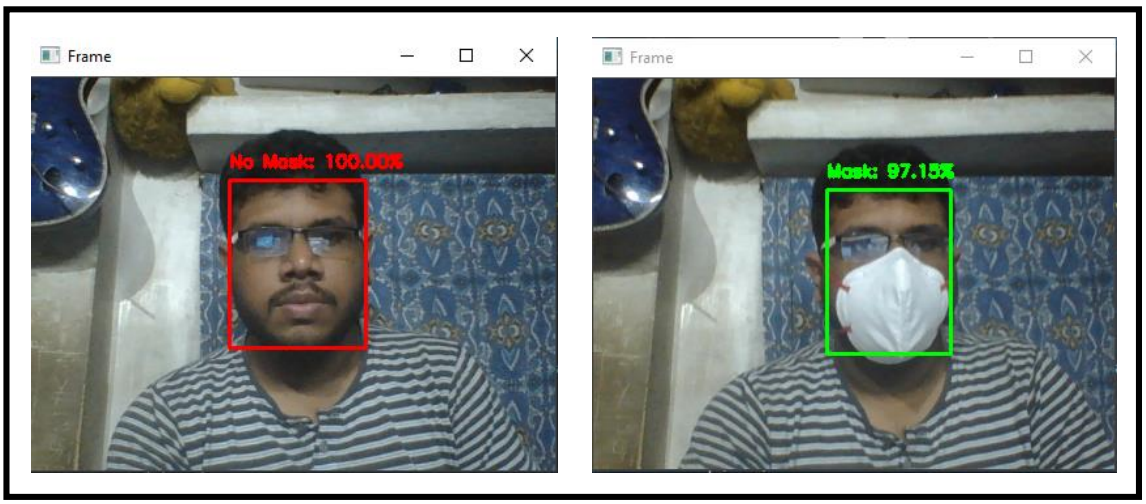
Code:

```
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import argparse
import imutils
import time
import cv2
import os

def detect_and_predict_mask(frame, faceNet, maskNet):
	(h, w) = frame.shape[:2]
	blob = cv2.dnn.blobFromImage(frame, 1.0, (300, 300),
		(104.0, 177.0, 123.0))
	faceNet.setInput(blob)
	detections = faceNet.forward()
	faces = []
	locs = []
	preds = []
	for i in range(0, detections.shape[2]):
		confidence = detections[0, 0, i, 2]
		if confidence > args["confidence"]:
			box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
			(startX, startY, endX, endY) = box.astype("int")
			(startX, startY) = (max(0, startX), max(0, startY))
			(endX, endY) = (min(w - 1, endX), min(h - 1, endY))
			face = frame[startY:endY, startX:endX]
			face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
			face = cv2.resize(face, (224, 224))
			face = img_to_array(face)
			face = preprocess_input(face)
			face = np.expand_dims(face, axis=0)
			faces.append(face)
			locs.append((startX, startY, endX, endY))
	if len(faces) > 0:
		preds = maskNet.predict(faces)
	return (locs, preds)
ap = argparse.ArgumentParser()
ap.add_argument("-f", "--face", type=str,
	default="face_detector",
	help="path to face detector model directory")
ap.add_argument("-m", "--model", type=str,
	default="mask_detector.model",
```

```
        help="path to trained face mask detector model")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
        help="minimum probability to filter weak detections")
args = vars(ap.parse_args())
print("[INFO] loading face detector model...")
prototxtPath = os.path.sep.join([args["face"], "deploy.prototxt"])
weightsPath = os.path.sep.join([args["face"],
        "res10_300x300_ssd_iter_140000.caffemodel"])
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
print("[INFO] loading face mask detector model...")
maskNet = load_model(args["model"])
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)
while True:
        frame = vs.read()
        frame = imutils.resize(frame, width=400)
        (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
        for (box, pred) in zip(locs, preds):
                (startX, startY, endX, endY) = box
                (mask, withoutMask) = pred
                label = "Mask" if mask > withoutMask else "No Mask"
                color = (0, 255, 0) if label == "Mask" else (0, 0, 255)
                label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)
                cv2.putText(frame, label, (startX, startY - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
                cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
        if key == ord("q"):
                break
cv2.destroyAllWindows()
vs.stop()
```

Result:



DATASET

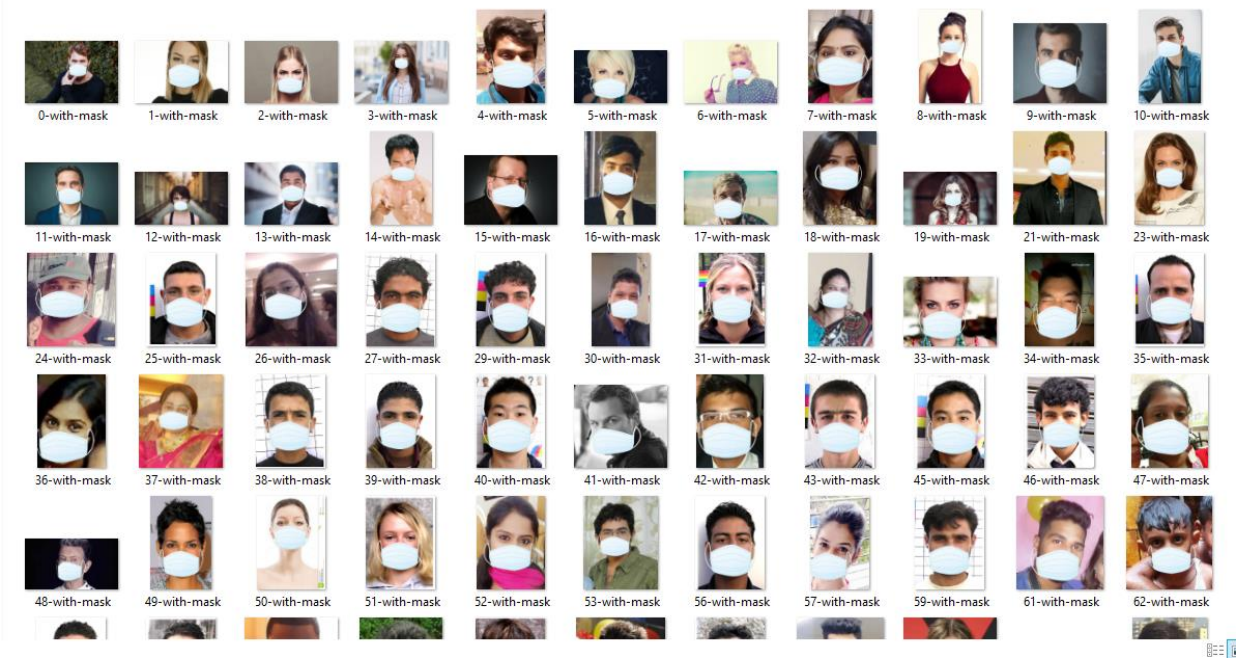
Folder Contains:

Name	Date modified	Type	Size
dataset	09-05-2021 09:41 PM	File folder	
face_detector	09-05-2021 09:41 PM	File folder	
detect_mask_video	07-05-2021 07:09 PM	Python file	3 KB
mask_detector.model	23-06-2020 10:19 AM	MODEL File	11,215 KB
plot	23-06-2020 10:19 AM	PNG File	43 KB

dataset:

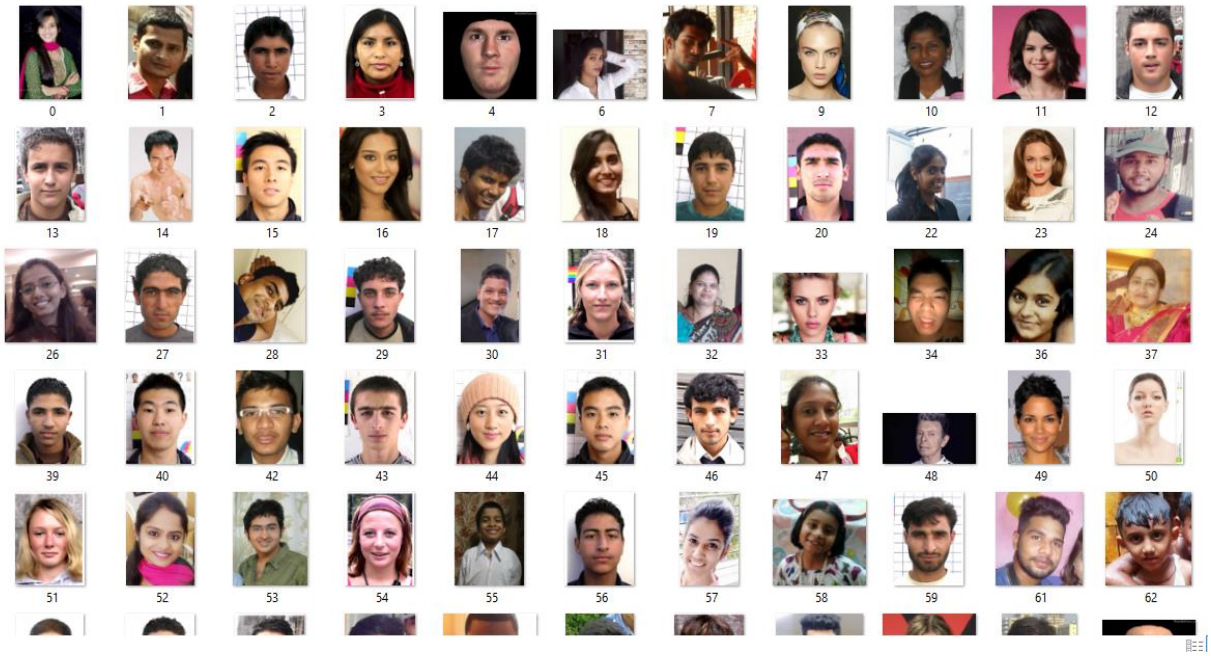
Name	Date modified	Type	Size
with_mask	09-05-2021 09:41 PM	File folder	
without_mask	09-05-2021 09:41 PM	File folder	

with_mask:



There are 690 pictures with mask for training the model

without_mask:

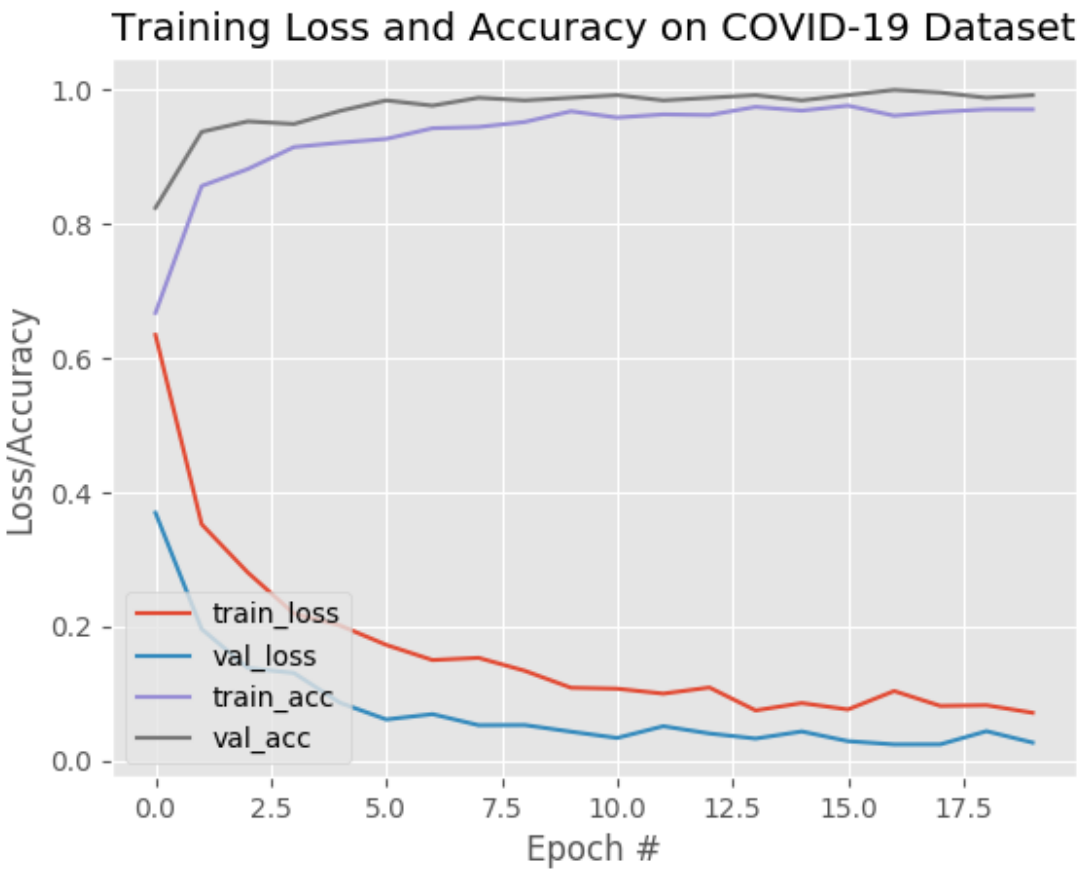


There are 686 images without mask for training the model

face_detector:

Name	Date modified	Type	Size
deploy.prototxt	23-06-2020 10:19 AM	PROTOTXT File	28 KB
res10_300x300_ssd_iter_140000.caffemodel	07-05-2021 04:33 AM	CAFFEMODEL File	10,417 KB

VISUALIZATION



The plot shows the loss and acurracy in training and valuating the model

CONCLUSION

Through the whole of this project I've learnt many new things that I would never know if I didn't got the opportunity to do such a great project on the topic ***Real Time Face Mask Detection*** using python. According to me it's a very useful project specially during this pandemic times of COVID-19. Face masks are now a essential and daily part of our lives. Wearing masks are for our own well being. This project can be implemented in many platforms and some places where wearing mask is mandatory like crowded areas (example - Railway Stations, Airports, Public Parks, Bus Stops, and even in Offices, Hotels and many more places). Government has made strict rule for people caught without mask. This project might be helpful in those places.

I could not complete this without the help of our respected professors who helped me a lot in completing this project. Also, a special thanks to our college ***Meghnad Saha Institute of Technology*** that have introduced me to the light of learning python and other programming languages. I also want to show respect to my parents who have always supported me everyway possible and supplying me with the essential equipments required for this project.

So, I conclude that I've completed this project succesfully and is in the point to submit it. Thanks and respect to everyone who helped me completing this project.