

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award
of Bachelor of Technology Degree in Information Technology

By
Santosh Kumar
38120076



DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND
TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119. TAMILNADU

OCTOBER 2020



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" Grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119.

Phone: 044 - 2450 3150 / 51 / 52 / 54 / 55 Fax: 044 - 2450 2344

www.sathyabama.ac.in



SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **Santosh Kumar** who carried out the project entitled "**Face Mask Detector**" under our supervision from August 2020 to January 2021.

Internal Guide

Ms. Brumancia E

Head of the Department

Dr. R. Subhashini M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **Santosh Kumar** hereby declare that the Project Report entitled “**Face Mask Detector**“ done by me under the guidance of **Ms. Brumancia E**, at Sathyabama Institute of Science and Technology (Deemed to be University), Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600119 is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 21/01/2021

Place: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA M.E., Ph.D., Dean, School of Computing** and **Dr. R. SUBHASHINI M.E., Ph.D., Head of the Department, Department of Information Technology** for providing me the necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms. BRUMANCIA E**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of **INFORMATION TECHNOLOGY** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE



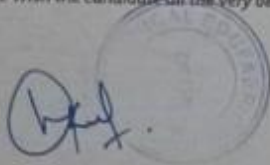
12th Jan 2020TO WHOM IT MAY CONCERN

This is to certify that Santosh Kumar Student of Sathyabama IST College had registered for Edufabrica Winter Camp 2020 @ Delhi with Ethical Edufabrica Pvt. Ltd which was held during the period of 25th Dec. to 12th Jan 2020. As part of the Edufabrica Winter Camp 2020, the candidate was required to undertake a project showcasing the aspects of the chosen course curriculum as a criteria towards finishing the Program.

So hereby with this letter, we acknowledge that the student has done his project on the topic of "Face Detection using Python" as part of training in "Python Development Program" and the project has exhibited the student's ability and skill in adapting the course structure into its practical nature and focusing on the core elements of the curriculum. The Project was satisfactory and had catered to all the key briefs handed to the student as an outline for the project

The candidate, during training with us has, shown diligence and perseverance towards adhering to the training protocols set by the company and to making sure that the given tasks were handled and performed with great outcome.

We Wish the candidate all the very best for the future



Ethical Edufabrica Pvt. Ltd.

Ethical Edufabrica Pvt Ltd

Plot No 12, First Floor, Hargobind Enclave, Karkardooma,
Anand Vihar, New Delhi, Delhi 110092

info@edufabrica.netwww.edufabrica.net[+91-93548 23927](tel:+91-9354823927)

ABSTRACT

FACE MASK DETECTOR

In the current COVID-19 pandemic situation, it is very important to maintain social distancing and following all the advisory in order to control the spread of CORONA Virus. We know that any one person can affect whole society. It has been adviced to wear mask, in order to prevent direct connection of the virus if anyone is already affected by this virus. This project can be used at Railway Stations, Airports, Offices, Public Places, Colleges, Schools,and other places in order to monitor whether the people are wearing mask or not as guidelines issued by the government authorities.

CHAPTER NO	TITLE NAME	PAGE NO
	ABSTRACT	7
	LIST OF FIGURES	9
	LIST OF ABBREVIATION	10
1	INTRODUCTION	11
	1.1 GENERAL	11
	1.2 OUTLINE OF THE PROJECT	11
2	AIM AND SCOPE	12
	2.1 AIM	12
	2.2 PROBLEM STATEMENT	12
	2.3 SCOPE	12
3	SYSTEM ANALYSIS AND DESIGN	13
	3.1 GENERAL	13
	3.2 SOFTWARES USED	14
	3.3 ALGORITHM	16
	3.4 PROJECT DESCRIPTION	17
4	RESULT AND DISCUSSION	18
	4.1 SYSTEM REQUIREMENTS	18
	4.2 RESULT	19
5	CONCLUSION AND FUTURE WORK	20
	5.1 CONCLUSION	20
	5.2 FUTURE WORK	20
	5.3 REFERENCE	20
	APPENDIX	21
	A.SCREENSHOTS	21
	B.SOURCE CODE	23

FIGURE NO	FIGURE NAME	PAGE NO
3.1	DATA FLOW	17
4.1	Output	19

LIST OF ABBREVIATIONS

ABBREVIATIONS

EXPANSION

OpenCv

Open Source Computer Vision Library

CNN

Convolutional Neural Network

IDE

Integrated Development Environment

CHAPTER 1

INTRODUCTION

1.1 GENERAL

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves. Face Mask Detection is one of the essential job especially at Public Places like Railway Stations, Colleges, Schools, Offices etc. in order to monitor whether people are following Guidelines given by the Government since even one affected person can affect others at large extent. This problem requires the use of several datasets since mask detection yield depends on many different factors such as positioning of Eyes, Eyebrows, Nose, Mouth and Jawline.

1.2 OUTLINE OF THE PROJECT

Machine Learning studies consist of different challenges when aiming to build a high-performance predictive model. It is crucial to select the right algorithms to solve the problem at hand, and in addition, the algorithms and the underlying platforms need to be capable of handling the volume of data. This project mainly focuses on Detecting whether people are wearing mask in Places like Colleges, Schools, Railway Stations, Offices, etc. First, we get the image with the face and run it through a cascade classifier. The classifier will give the region of interest of the face (height and width). Secondly, we will resize the region of interest into a 100x100 and pass it to a pre-trained CNN, it will give us the probability as an output.

CHAPTER 2

AIM AND SCOPE

2.1 AIM

The main objective of this project is to detect whether people are wearing mask or not at places of large gathering. This detector is recommended so as to monitor public in order to minimize the chances of spreading of virus.

2.2 PROBLEM STATEMENT

Even one COVID affected person can affect others at a huge extent. But many people doesn't follow the instructions issued by the concerned authorities. This project is designed so as to help the government to get monitor places where the crowd might be at large extent.

2.3 SCOPE

This Project can be implemented at Railway Stations, Airports, Offices, Public Places, Colleges, Schools, and other places where the chances of public gathering is huge by connecting it to the CCTV Cameras. The scope of the project is to detect whether people are adhering to the guidelines issued by the Government and Concerned authorities. This project will also help govt. in order to take the appropriate decision and strategies in order to fight against COVID19. This may save many lives. This can also help government to identify the places where most people are not aware of wearing mask.

CHAPTER 3

SYSTEM ANALYSIS

3.1 GENERAL

3.1.1 MACHINE LEARNING

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

3.1.2 PYTHON

Python is an interpreted, high-level and general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

3.1.3 IDE (INTEGRATED DEVELOPMENT ENVIRONMENT)

IDE stands for Integrated Development Environment. It's a coding tool which allows you to write, test, and debug your code in an easier way, as they typically offer code completion or code insight by highlighting, resource management, debugging tools,... And even though the IDE is a strictly defined concept, it's starting to be redefined as other tools such as notebooks start gaining more and more features that traditionally belong to IDEs. The IDE used in this project is Anaconda Navigator.

3.1.4 PYTHON MACHINE LEARNING FRAMEWORK

A Machine Learning framework is a collection of packages or modules which allow developers to implement Machine Learning Algorithms. The Python deep Learning Frameworks and Library used in this project are OpenCv, Mobilenet, Keras, Tensorflow, Numpy, Matplotlib.

3.2 SOFTWARES USED

3.2.1 CODE EDITORS USED

SUBLIME TEXT EDITOR

Sublime Text is one the best text editor for programmers. Packed with numerous features and great looking interface, Sublime is available for all three major Desktop i.e. Windows, Linux and MacOS.

ANACONDA

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is an integrated development environment developed by data scientists and is best suitable for Machine learning and Data science.

3.2.2 LIBRARIES AND FRAMEWORKS USED

OPENCV

OpenCV is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision

applications and to accelerate the use of machine perception in the commercial products.

MOBILENET

MobileNet is an efficient and portable CNN architecture that is used in real world applications.

KERAS

Keras is an open-source software library that provides a python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

TensorFlow

TensorFlow is an end to end open source platform for Machine Learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in Machine Learning and developers easily build and deploy ML powered applications.

NumPy

NumPy is a python library used for working with arrays. It stands for Numerical Python. It also has functions for working in domain of linear algebra, fourier transform, and matrices. It is an open source project and we can use it freely.

Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.

3.3 ALGORITHM

3.3.1 CNN

Convolutional Neural Network is a Deep Learning algorithm which can take in an input image, assign importance to various aspects or objects in the image and be able to differentiate one from the other.

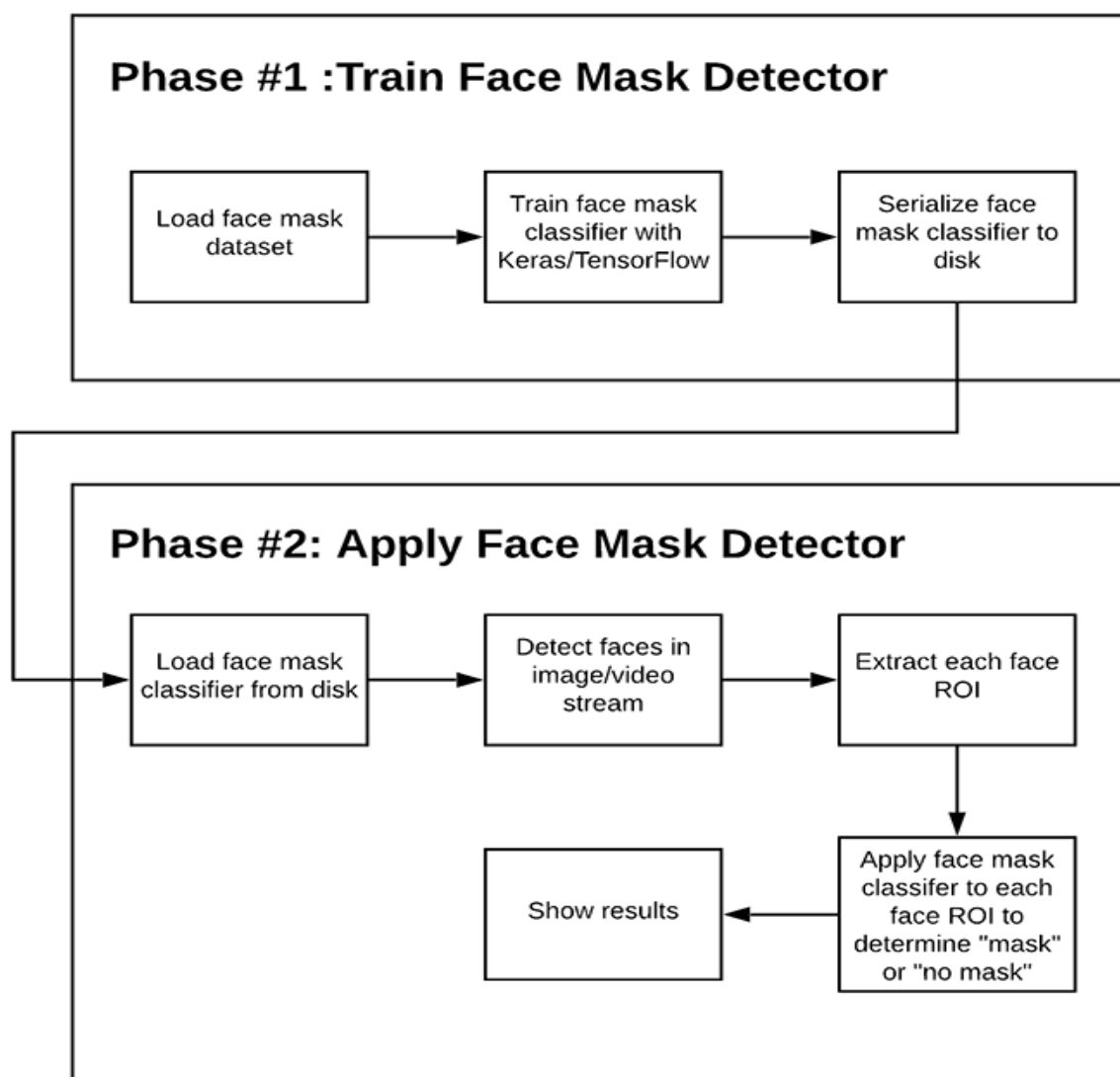
3.3.2 MobileNet

The MobileNet algorithm uses deep convolution layer as the basic layer network, which optimizes the delay while considering the size of the model, it's processing is much faster than CNN.

3.4 PROJECT DESCRIPTION

This project aims at detect the Face Mask using Deep Learning. I have created a several files which contains the machine learning algorithms and also used pretrained model for recognizing Face. I have used around 1000 photos of people wearing mask and without mask. I have used MobileNet instead of CNN usual model. MobileNet is much faster than CNN in case of Processing. I have used OpenCv, Keras, Tensorflow, Matplotlib and Numpy. This model face as input and give the probability of a person wearing mask or without mask. To run the model, we need to open command prompt, for this purpose I have used Anaconda command prompt, type the directory path and then type `python detect_mask_video.py`. Initially it takes some to warm up camera and then it starts webcam.

The work flow of the model is demonstrated in the following diagram.



CHAPTER 4

RESULTS AND DISCUSSION

4.1 SYSTEM REQUIREMENTS

- Operating System: Windows/Mac
- RAM: 4GB
- Processor: 64x 1.0Ghz processor
- ROM: 8GB

4.2 RESULT

The Results shows that we can attain an accurate crop yield prediction using the Random Forest algorithm. Random Forest algorithm achieves a largest number of crop yield models with a lowest models. It is suitable for massive crop yield prediction in agricultural planning. This makes the farmers to take the right decision for right crop such that the agricultural sector will be developed by innovative ideas.

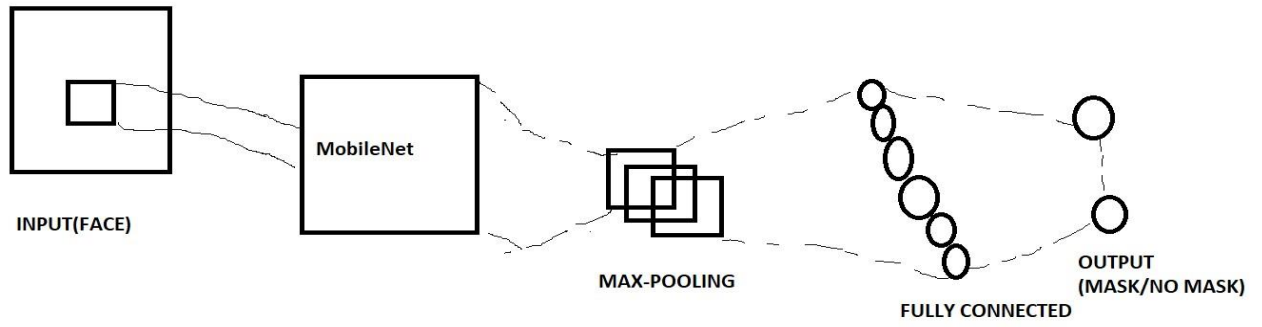


FIG 4.2 Work Flow In MobileNet Model

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

This project is undertaken using Deep Learning and detects the Face CNN and MobileNet algorithms. In our proposed model among all the two algorithms MobileNet gives the output faster as compared to CNN. Whenever I tried with more Learning the accuracy was very poor but once I tried with less Learning rate then it gave more accuracy. This Project will be very much helpful in places like Railway Stations, Airports, Offices, Public Places, Colleges, Schools, and other places by connecting it to the CCTV Cameras.

5.2 FUTURE WORK

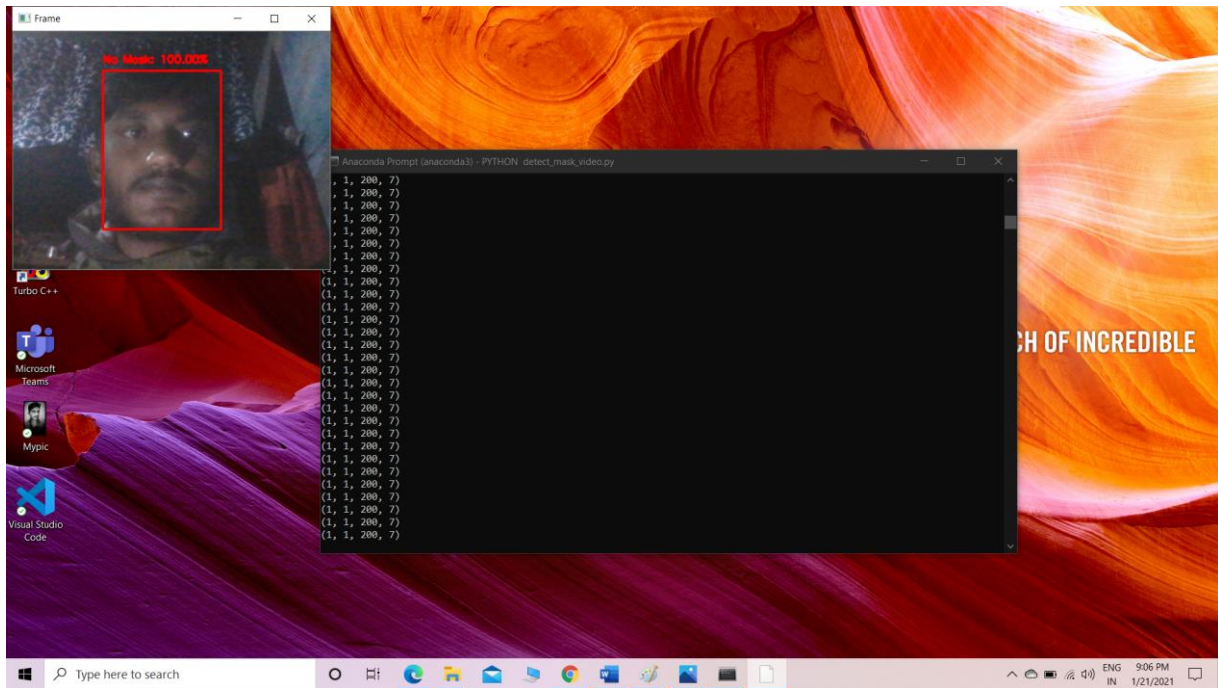
This project describes Face Mask Detection. In future we can determine the efficient algorithm based on their accuracy metrics that will help to choose an efficient algorithm for Face Mask Detector.

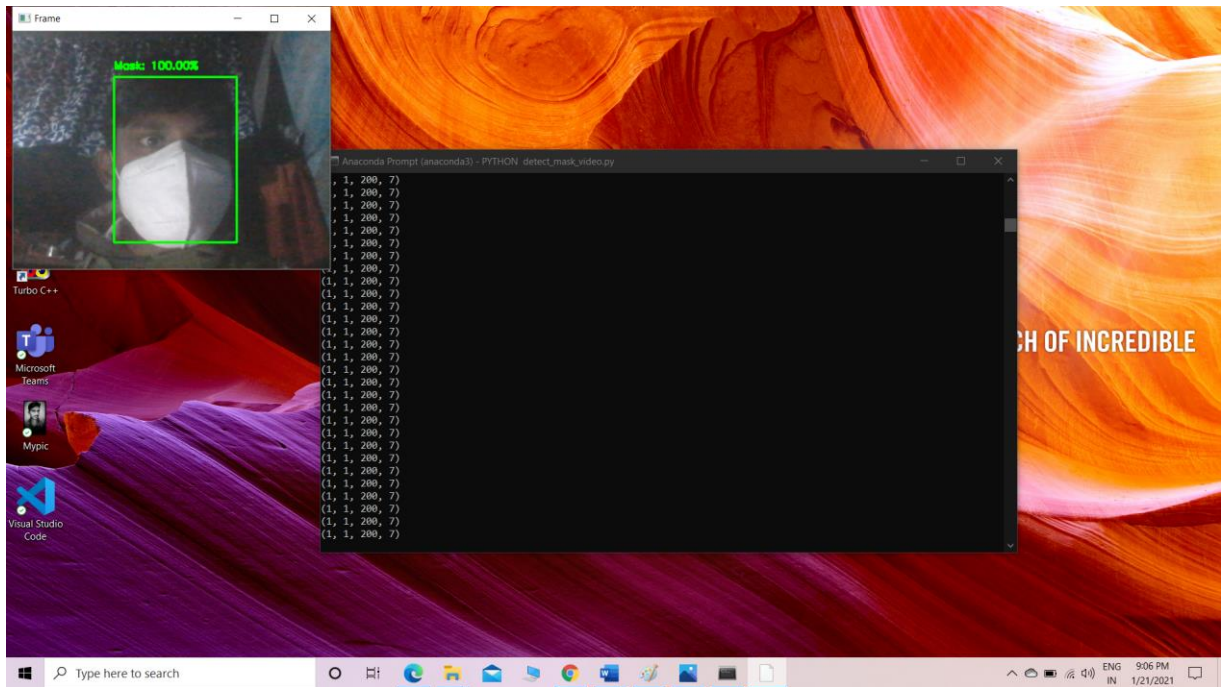
5.3 REFERENCES

1. <https://keras.io/api/applications/mobilenet/>
2. <https://keras.io/>
3. <https://www.anaconda.com/>
4. <https://numpy.org/doc/>

APPENDIX

A.SCREENSHOTS





B.SOURCE CODE

1. detect_mask_video.py

```
# import the necessary packages
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 imutils
import time
import cv2
import os

def detect_and_predict_mask(frame, faceNet, maskNet):
    # grab the dimensions of the frame and then construct a blob
    # from it
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
                                  (104.0, 177.0, 123.0))

    # pass the blob through the network and obtain the face detections
    faceNet.setInput(blob)
    detections = faceNet.forward()
    print(detections.shape)

    # initialize our list of faces, their corresponding locations,
    # and the list of predictions from our face mask network
    faces = []
    locs = []
    preds = []

    # loop over the detections
    for i in range(0, detections.shape[2]):
        # extract the confidence (i.e., probability) associated with
        # the detection
        confidence = detections[0, 0, i, 2]

        # filter out weak detections by ensuring the confidence is
        # greater than the minimum confidence
        if confidence > 0.5:
            # compute the (x, y)-coordinates of the bounding box for
            # the object
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")
```

```

        # ensure the bounding boxes fall within the dimensions of
        # the frame
        (startX, startY) = (max(0, startX), max(0, startY))
        (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

        # extract the face ROI, convert it from BGR to RGB channel
        # ordering, resize it to 224x224, and preprocess it
        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)

        # add the face and bounding boxes to their respective
        # lists
        faces.append(face)
        locs.append((startX, startY, endX, endY))

    # only make a predictions if at least one face was detected
    if len(faces) > 0:
        # for faster inference we'll make batch predictions on *all*
        # faces at the same time rather than one-by-one predictions
        # in the above `for` loop
        faces = np.array(faces, dtype="float32")
        preds = maskNet.predict(faces, batch_size=32)

    # return a 2-tuple of the face locations and their corresponding
    # locations
    return (locs, preds)

# load our serialized face detector model from disk
prototxtPath = r"face_detector\deploy.prototxt"
weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk
maskNet = load_model("mask_detector.model")

# initialize the video stream
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start() #using primary camera

# loop over the frames from the video stream
while True:
    # grab the frame from the threaded video stream and resize it
    # to have a maximum width of 400 pixels
    frame = vs.read()

```

```

frame = imutils.resize(frame, width=400)

# detect faces in the frame and determine if they are wearing a
# face mask or not
(locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

# loop over the detected face locations and their corresponding
# locations
for (box, pred) in zip(locs, preds):
    # unpack the bounding box and predictions
    (startX, startY, endX, endY) = box
    (mask, withoutMask) = pred

    # determine the class label and color we'll use to draw
    # the bounding box and text
    label = "Mask" if mask > withoutMask else "No Mask"
    color = (0, 255, 0) if label == "Mask" else (0, 0, 255) #color bgr (0 least
255 max)

    # include the probability in the label
    label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

    # display the label and bounding box rectangle on the output
    # frame
    cv2.putText(frame, label, (startX, startY - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
    cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

# show the output frame
cv2.imshow("Frame", frame)
key = cv2.waitKey(1) & 0xFF

# if the `q` key was pressed, break from the loop
if key == ord("q"):
    break

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()

```

2. train_mask_detector.py

```

# import the necessary packages
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2

```

```

from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os

# initialize the initial learning rate, number of epochs to train for,
# and batch size
INIT_LR = 1e-4 #always learning rate should be less, loss will be calculated
perfectly hence better accuracy very soon,0.0001
EPOCHS = 20
BS = 32

#location of dataset directory
DIRECTORY = r"C:\Mask Detection\CODE\Face-Mask-Detection-master\dataset"
CATEGORIES = ["with_mask", "without_mask"]

# grab the list of images in our dataset directory, then initialize

```

```

# the list of data (i.e., images) and class images
print("[INFO] loading images...")

data = [] #empty list, appending all the images in this list
labels = [] #label of images( with or without mask)

#looping through the categories
for category in CATEGORIES: #inside CATEGORIES there are 2 values(with and
without mask)
    path = os.path.join(DIRECTORY, category) #go to path joinig the diretory and
category
    for img in os.listdir(path):
        img_path = os.path.join(path, img)
        image = load_img(img_path, target_size=(224, 224)) #coming from
keras.preprocessing, converting the size of all the images to 224 and 224 (ht and
width resp.)
        image = img_to_array(image)
        image = preprocess_input(image)#using mobilenet

        data.append(image)
        labels.append(category)

# perform one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

data = np.array(data, dtype="float32") #converting data and labels into numpy array
labels = np.array(labels)

(trainX, testX, trainY, testY) = train_test_split(data, labels,

```

```

test_size=0.20, stratify=labels, random_state=42)

# construct the training image generator for data augmentation
aug = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode="nearest")

# load the MobileNetV2 network, ensuring the head FC layer sets are
# left off
baseModel = MobileNetV2(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3))) #ht,wt,channels,rgb

# construct the head of the model that will be placed on top of the
# the base model
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel) #relu is goto activation
function for non linear use cases
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel) #2 layers

# place the head FC model on top of the base model (this will become
# the actual model we will train)
model = Model(inputs=baseModel.input, outputs=headModel)

```

```

# loop over all layers in the base model and freeze them so they will
# *not* be updated during the first training process
for layer in baseModel.layers:
    layer.trainable = False

# compile our model
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
              metrics=["accuracy"])

# train the head of the network
print("[INFO] training head...")
H = model.fit(
    aug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
    epochs=EPOCHS)

# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)

# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)

# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs,
                          target_names=lb.classes_))

```

```

# serialize the model to disk
print("[INFO] saving mask detector model...")
model.save("mask_detector.model", save_format="h5")

# plot the training loss and accuracy
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig("plot.png")

```