VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Project Report

Control System

ECE2010

Winter Semester 2020-21

Face Mask Detection

Submitted By:

ASHUTOSH DEWANGAN

19BEC0628

RAKSHIT KASHYAP
19BEC0653

ANUPAM KUMAR

19BEC0211

SHRUT MAKDE

19BEC0656

Submitted to:

Dr. RAJESH R

Declaration

It is always a pleasure to remind the fine people for their sincere guidance we received to uphold our skills and knowledge while doing this project. First of all, thanks to our parent for giving encouragement, enthusiasm and invaluable assistance to us. Without all this, we might not be able to complete this subject properly.

Second, I would like to thank our HOD for giving us the opportunity to undergo this particular course. He also gives us their guidance and support.

Thirdly, I also want to express my deepest thanks to Prof. Rajesh R sir that has helped us a lot in dealing with this project. He had supported to us by showing different method of information collection about the topics. He helped all time when we needed and he gave right direction toward completion of this project.

We would say thanks to all our class mates for making a healthy, supportive and progressive environment around us that also kept us too much motivated.

Finally, we apologize all other unnamed who helped us in various ways to have good project.

Index

Abstract

The year 2020 & 2021 has been one of the most stressful years in almost everyone's lifetimes because of a pandemic due to the deadly virus causing COVID-19 disease. With over170 million people in the world affected by the disease and taking more than a million people's lives, it has been a major concern on how to restrict the spread of this deadly virus.

COVID-19 pandemic has rapidly affected our day-to-day life disrupting the world trade and movements. Wearing a protective face mask has become a new normal. In the near future, many public service providers will ask the customers to wear masks correctly to avail of their services. Therefore, face mask detection has become a crucial task to help global society.

Our proposed method detects the face from the image correctly and then identifies if it has a mask on it or not. As asurveillance task performer, it can also detect a face along with a mask in motion. The method attains accuracy up to 98%

Objectives

Effective strategies to restrain COVID-19 pandemic need high attention to mitigate negatively impacted communal health and global economy, with the brim-full horizon yet to unfold. In the absence of effective antiviral and limited medical resources, many measures are recommended by WHO to control the infection rate and avoid exhausting the limited medical resources. Wearing a mask is among the non-pharmaceutical intervention measures that can be used to cut the primary source of SARS-CoV2 droplets expelled by an infected individual. Regardless of discourse on medical resources and diversities in masks, all countries are mandating coverings over the nose and mouth in public. To contribute towards communal health, this project aims to devise a highly accurate and real-time technique that can efficiently detect non-mask faces in public and thus, enforcing to wear mask.

Introduction

Even from the beginning of the pandemic, face masks were considered to give a certain amount of protection to the surrounding people from an infected source. Though, due to the shortage of masks around the world, hoarding these masks were not encouraged as themedical workers needed them the most. As more research into the virus came out, it was found that if large number of people wore masks, even simple cloth masks, the reproduction rate of the virus would decrease enormously.

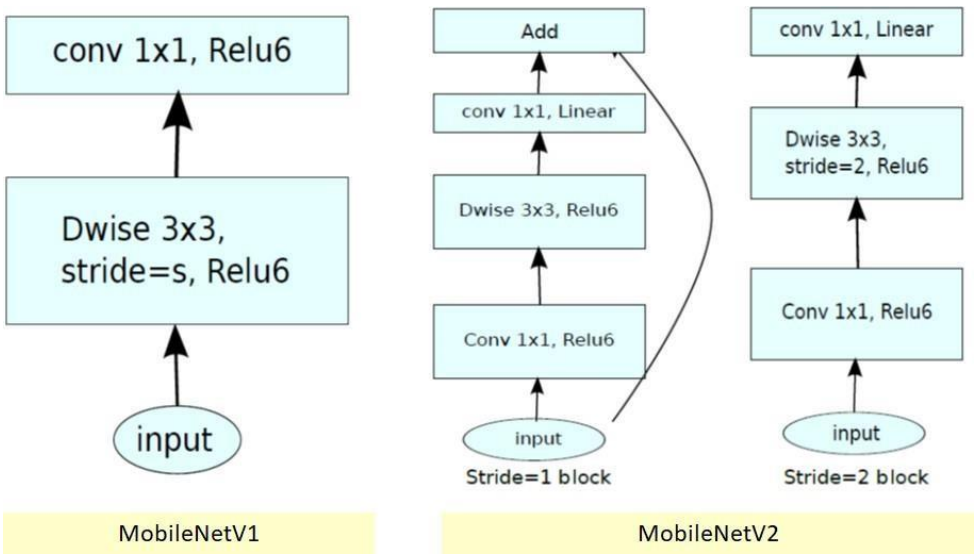
In order to implement the same, a lot of countries have issued strict guidelines on individuals in public places to wear face masks at all times. Due to the enormous amount of people to keep track of, it is usually difficult for any law enforcement agency to keep track of the same. Therefore, we have come with a Machine Learning model, which can capture a photo of the person and verify if he/she is indeed wearing a face mask. This system can be put near the entrance of public places where entry would be only allowed ifface protection, face masks are worn. This would help in containing the deadly virus and would help the world to get back on its feet.

Literature survey

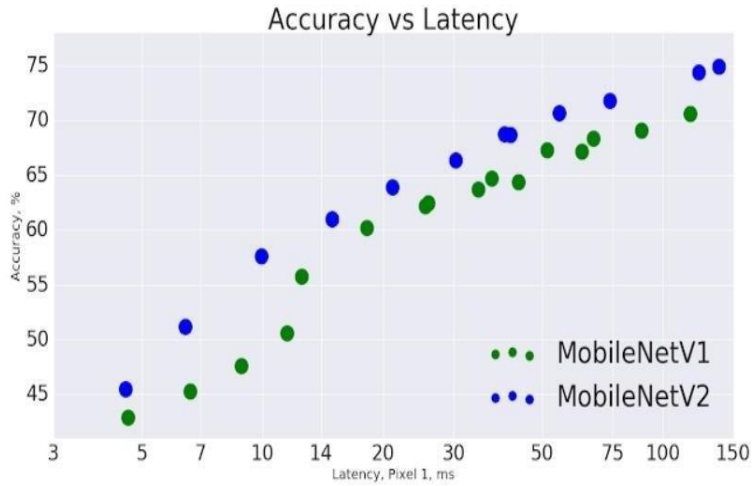
Comparison with other Models

MobilenetV2 vs MobilenetV1

The MobileNetV2 models are faster for the same accuracy across the entire latency spectrum. In particular, the new models use 2x fewer operations, need 30% fewer parameters and are about 30-40% faster on a Google Pixel phone than MobileNetV1 models, all while achieving higher accuracy.



MobileNetV2 is a very effective feature extractor for object detection and segmentation. For example, for detection when paired with the newly introduced SSDLite the new model is about 35% faster with the same accuracy than MobileNetV1.



YOLO vs MobilenetV2

An experiment was carried out using the 70:30 YOLO model trained for 132,000 iterations(referred to as the YOLO pith detector) and the 80:20 SSD MobileNet model trained for 152,000 iterations (referred to as the SSD MobileNet pith detector) to evaluate the detectionperformance.

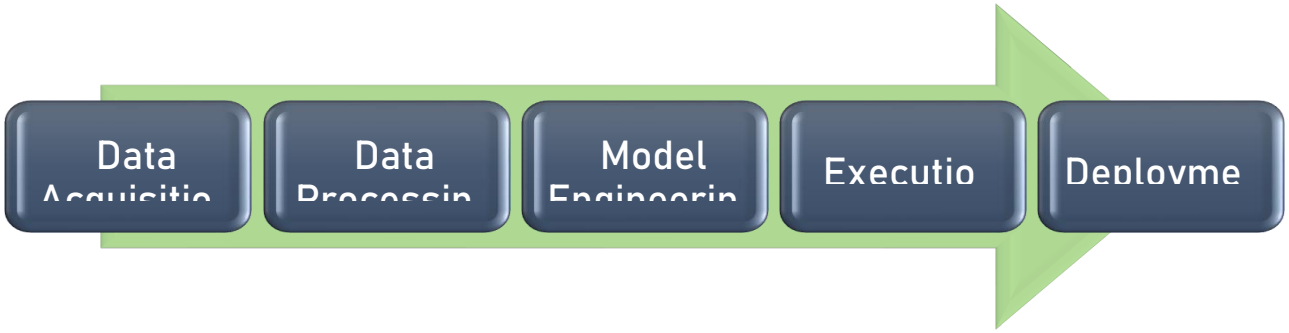
Algorithm	Detection Rate (%)	$\overline{D}$	D_{SD}	D_{max}	D_{min}	$\overline{T}$ (s)
YOLO (70:30) @132,000	76.6	7.61	13.8	96.71	0.05	1.05
SSD MobileNet (80:20) @152,000	87.7	7.04	15.32	193.19	0.11	0.78

The inference dataset was employed in the comparison. The confidence levels of detectionwere fixed to 0.4 (40%) for both the YOLO and SSD MobileNet pith detectors. This meansthat all objects detected within an image that have a confidence level greater than or equal to 0.4 are reported as piths. Table 2 shows the comparison results. The detection rate of theSSDMobileNet pith detector is superior to that of the YOLO pith detector.

It is capable of making correct pith detections 83.6% of the time. However,when the location error is considered, the YOLO pith detector outperformsthe SSD MobileNet pith detector because it has half the

System Model (Architecture)

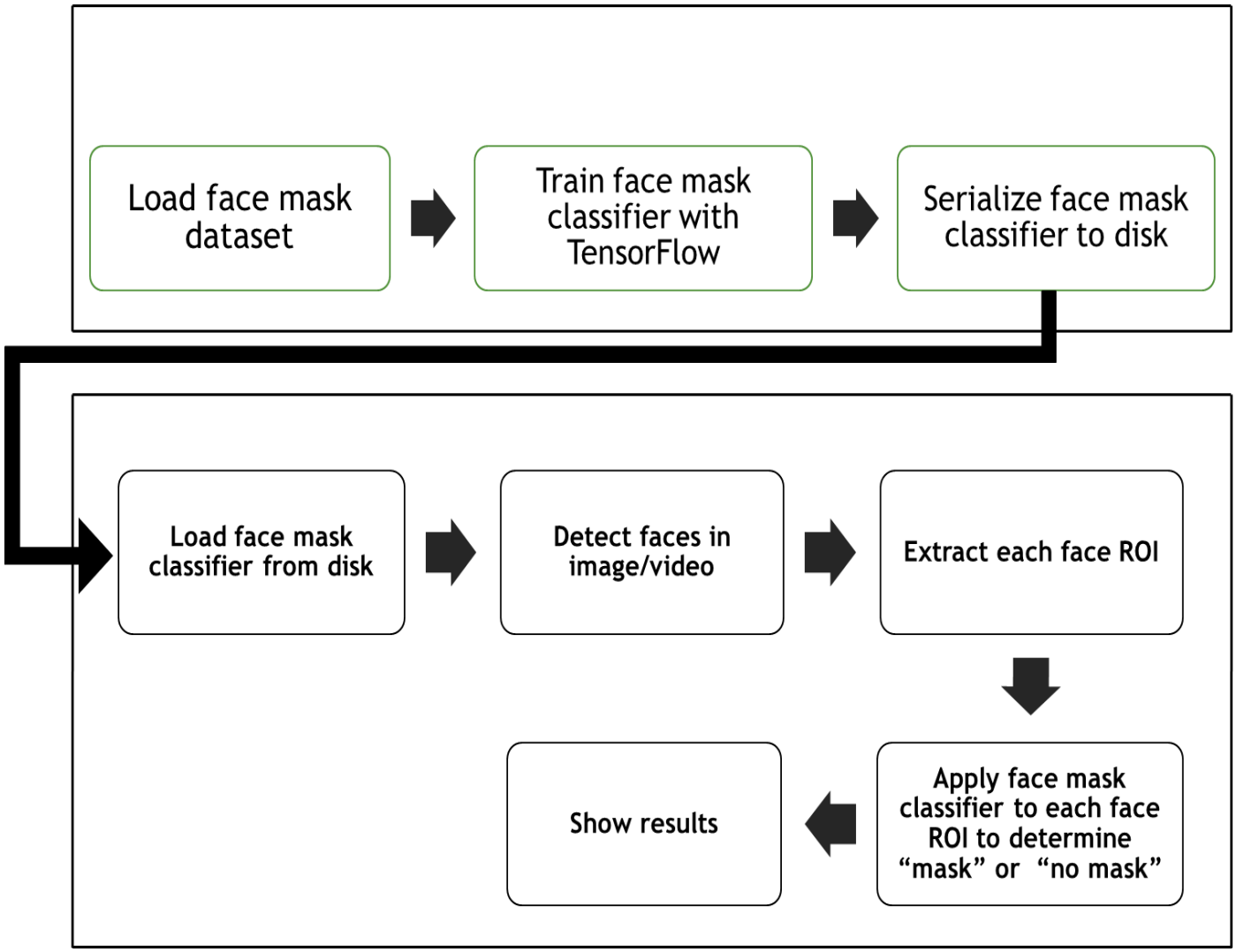
I) General Architecture



The above is the general architecture of our project/model. The steps that are involved are as follows:

- **Data Acquisition:** We have photos as our dataset which will serve as the inputto the training model.
- **Data Processing:** The data (photos) are then processed as our model gets trained for the detection of masks later on.
- **Model Engineering:** The model is being trained in this phase through the photos as the input and this training of our model will help the computer visiondetect the masks from our testing user.
- **Execution:** After the model has been trained, it becomes ready for test cases and final execution, and it is the part where we find out how good our model is in finding out whether the user has his/her mask on or not.
- **Deployment:** This is the part when we know that our project has been implemented correctly and is displaying good accuracy, which is complying with the community standards and can be deployed in the respective areas like the housing societies and industrial and office areas.

II) Face Mask Architecture



Methodology

Training Phase

- First, we have all the photos in a folder and we import that folder.
- There are two folders inside it, one with photos where people have masks on and the other where people who do not have their mask on.
- These two types of folders act as tags for us, i.e. the people with mask and withoutmask. These two helps us training for our two specific purposes, to detect people with masks (training with the masked photos) and to detect people without masks(training with the photos without mask).
- We save this trained model in our disk for future testing purposes.

Applying Face Masks

- Loading the pre saved model from our disk so that we can use it for testing rounds.
- Start the video stream using Imutils.
- Detection of face using the camera.
- Extraction of face from the view of the camera.
- Applying the model and testing the input which we got from the camera and finallyshowing two values.
- The first value will be the confidence percentage of the person wearing the mask, and the second value will be the person not wearing the mask.
- We display the value at the end as our prediction, depending on which value is higher.
- If the person is not wearing a mask, we save the picture of that particular person, so that we can identify that person and not save the picture otherwise (if they are wearing a mask)

MobileNetV2

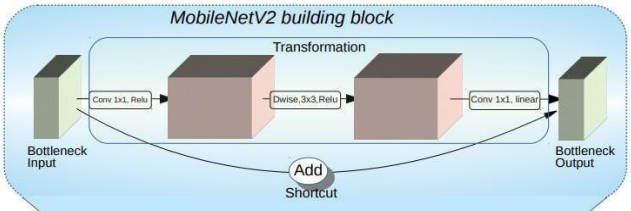
In MobileNetV2, there are two types of blocks.

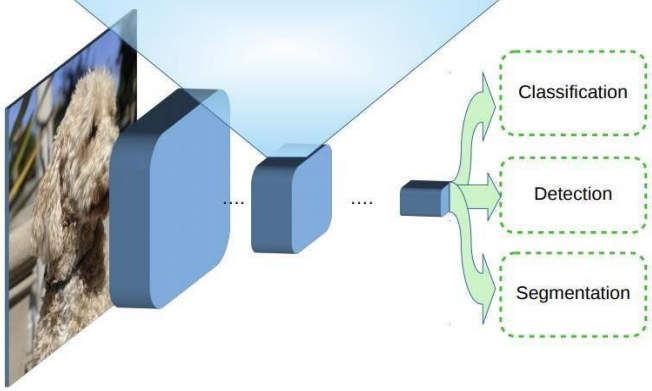
- One is residual block with stride of 1.
- Another one is block with stride of 2 for downsizing.

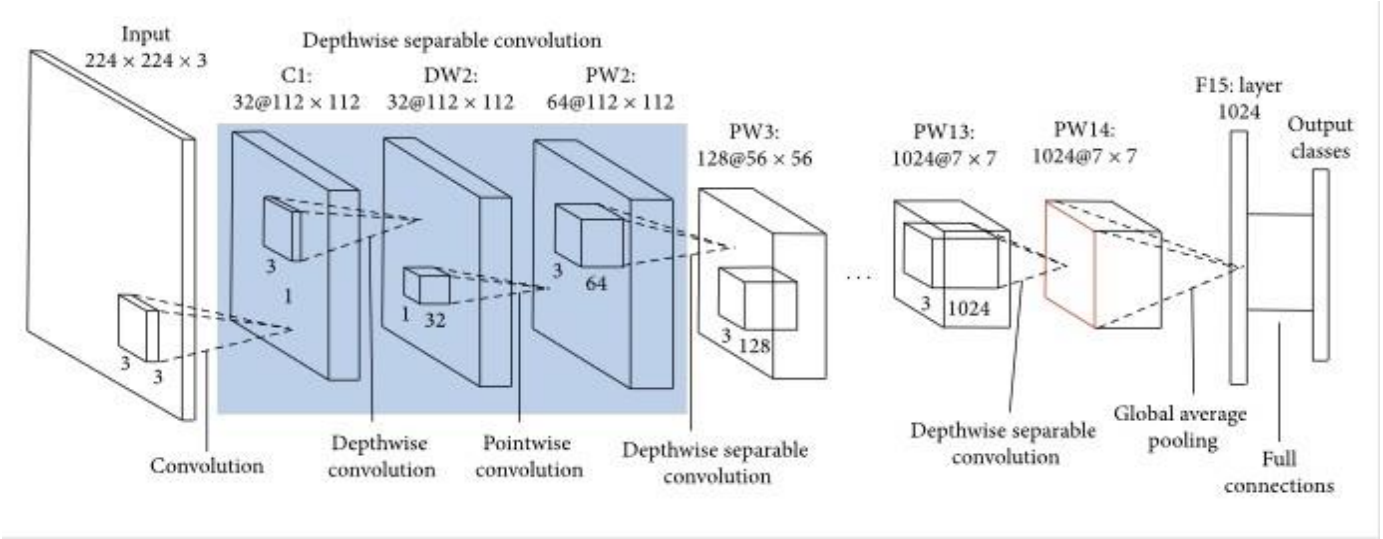
There are 3 layers for both types of blocks.

- This time, the first layer is 1×1 convolution with ReLU6.
- The second layer is the depth wise convolution.
- The third layer is another 1×1 convolution but without any non-linearity. It is claimed that if ReLU is used again, the deep networks only have the power of a linear classifier on the non-zero volume part of the output domain.

There is also an expansion factor t. And t=6 for all main experiments. If the input got 64 channels, the internal output would get 64×t=64×6=384 channels.







Architecture

The MobileNetV2 architecture is based on an inverted residual structure where the input and output of the residual block are thin bottleneck layers opposite to traditional residual models which use expanded representations in the input an MobileNetV2 uses lightweightdepth wise convolutions to filter features in the intermediate expansion layer. Additionally,we find that it is important to remove non-linarites in the narrow layers in order to maintainrepresentational power.

We demonstrate that this improves performance and provide an intuition that led to this design. Finally, our approach allows decoupling of the input/output domains from the

expressiveness of the transformation, which provides a convenient framework for further analysis.

Performance Analysis

First, we import the folder that has all the pictures of our dataset.

There are two folders inside it, one with photos where people have masks on and the otherwhere people who do not have their mask on.

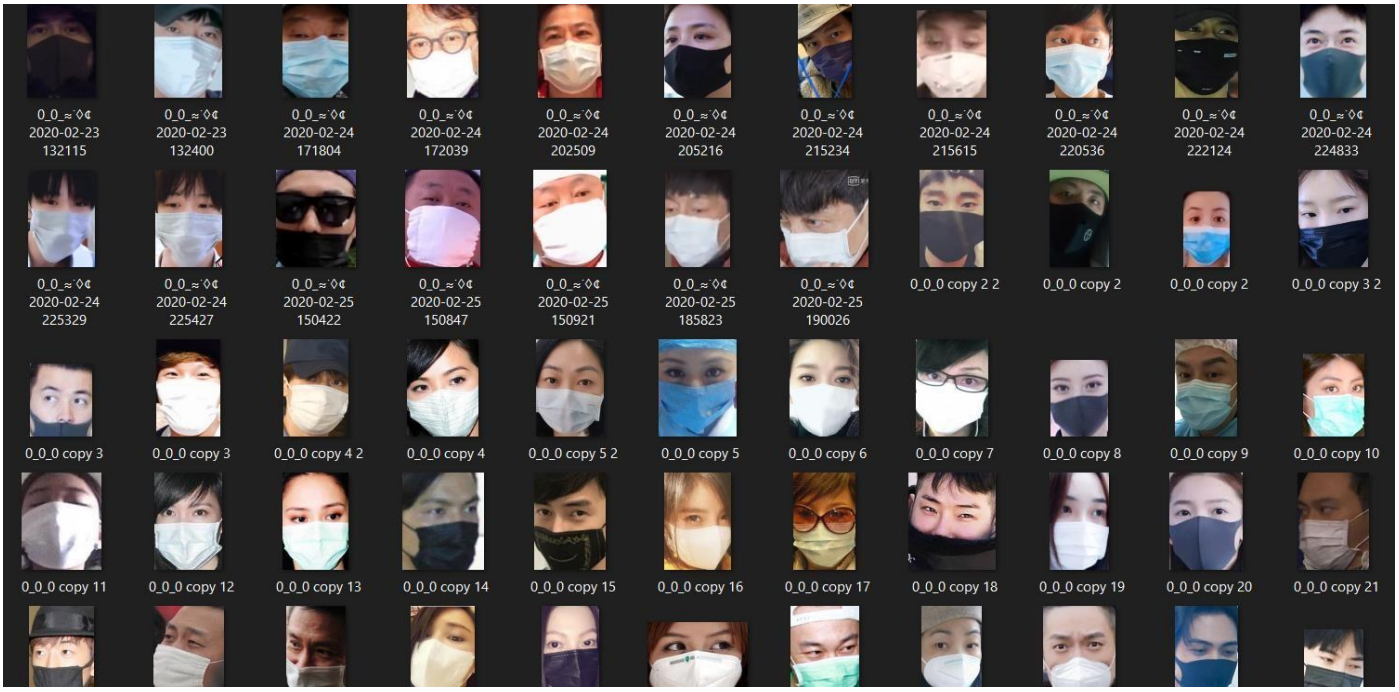


Image Of content inside the folder 'with_mask' in our dataset

Image Of content inside the folder 'without_mask' in our dataset

These two types of folders act as tags for us, i.e. the people with masks and without masks.These two helps us in the training for our two specific purposes, todetect people with masks(training with the masked photos) and to detect people without masks (training with the photos without masks). We save this trained model in our disk for future testing purposes. While running the main part of ourproject, we first load the pre-trained model from our disk.

Then we start accessing the cameras connected to our device. When the camera starts, we can see the current footage the camera can see on our screen and our project accesses it. Then we look in the frame for a face and when we have a face,we extract the details. Thenwe feed them to our mask detector which checks if the nose and mouth of the face is properly covered with something(mask).

Then we generate 2 values, confidence with which our model can say that the face has a mask on and the confidence with which it can say that the face has no mask. Then we compare these values and show results on the frame accordingly. Also, if we determine thatthe person is not wearing a mask, we save the snapshotof the frame at that very instant andprevent infinite images.

Code

Training our model:

```
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 #learning rate should be less,so that the loss will be calculated properly and we get the better accuracy very soon.

EPOCHS = 20

BS = 32 #back value

DIRECTORY = r"/content/drive/MyDrive/control_system_project_dataset/dataset" #location of dataset folder in my laptop

CATEGORIES = ["with_mask", "without_mask"] #storing folders present inside the directory

# 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...") #we just write this so that we have context what's happening

data = [] #append all the image arrays
labels = []# append image which are with mask and without mask

for category in CATEGORIES:
```

```
    path = os.path.join(DIRECTORY, category) #joining directory with elements present in CATAGORIES

        for img in os.listdir(path): #list down all the images present in particular directory

            img_path = os.path.join(path, img) #joining path with images(with_mask joins with its images, without_mask joins with its images)

                image = load_img(img_path, target_size=(224, 224)) #load_image comes from keras.image.preprocessing.it load all the image path. Also we are making all the images of equal size that is 224x224

            image = img_to_array(image) #img_to_array function comes from keras.image.preprocessing module. Convert all the images in array.


            image = preprocess_input(image) #preprocessing of input.

        data.append(image)
        labels.append(category)# with mask and without mask

    #the labels are in text formal. So we need to convert in number format that is binary format.

    # perform one-hot encoding on the labels
    lb = LabelBinarizer() #this function comes from sklearn.preprocessing module. We convert with mask and without mask in categorical values.

    labels = lb.fit_transform(labels)
    labels = to_categorical(labels)

    #now we need to convert into numpy array because only with arrays deep learning model can work

    data = np.array(data, dtype="float32")
    labels = np.array(labels)

    (trainX, testX, trainY, testY) = train_test_split(data, labels,test_size=0.20, stratify=labels, random_state=42) # splitting the train and test data. Size of test data is 20% means out of total images, 20% is used for testing.80% is used for training. Random state is used to get same set of train and test data.

    # 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") #it creates many image with a single image by creating many features like height,width etc.

    # 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))) #creating base model. Imagenet means there are some pre trained model for images, so when we use it those weights get initialize and will get proper results. Include_top is a Boolean value which is said whether to connect fully layered value at the top of our network.Input_tensor is nothing but the same of image going through, 3 represent the 3 channel that is red, blue and green.

```
# construct the head of the model that will be placed on top of the base model

headModel = baseModel.output #passing the base model output

headModel = AveragePooling2D(pool_size=(7, 7))(headM del)# creating pooling of size 7 by 7.

headModel = Flatten(name="flatten")(headModel) #flattens the layer

headModel = Dense(128, activation="relu")(headModel)#dense layer=128 neurons, relu is go to activation for non linear use cases. Whenever we deak with images we go to the relu.

headModel = Dropout(0.5)(headModel)#we are using drop out just to avoid overfitting of the model.

headModel = Dense(2, activation="softmax")(headModel)#output is of 2 layer, one is for with_mask and one is for without_mask. Since we are dealing with binary classification here, so we use softmax as activation function because they are probability based 0 and 1 values.

# 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)# accept two model one is input that is base model  other one is output that is 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) #giving initial learning

model.compile(loss="binary_crossentropy", optimizer=opt, "])#in the loss fuction I have given binary_crossentropy and for optimizer we are using adom optimizer

metrics=["accuracy”] #we are going to track matrics here which is accuracy matrix

# call image data generator is always better to use the image data generated when you have a lesser date of the tandoor for the validation data by using the testing their self destruction and during the number of the box that we mentioned before lichtman tv box

# 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...")#evaluate our work using model.predict method
 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
 #we need to find the index of the label corresponding to the largest building the probability that a leap year
selected data and integrating the classification report with the without formatting
 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...")#finally we are saving the model in h5 formate
 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")

 #finally we will plot the image using matplot library
```

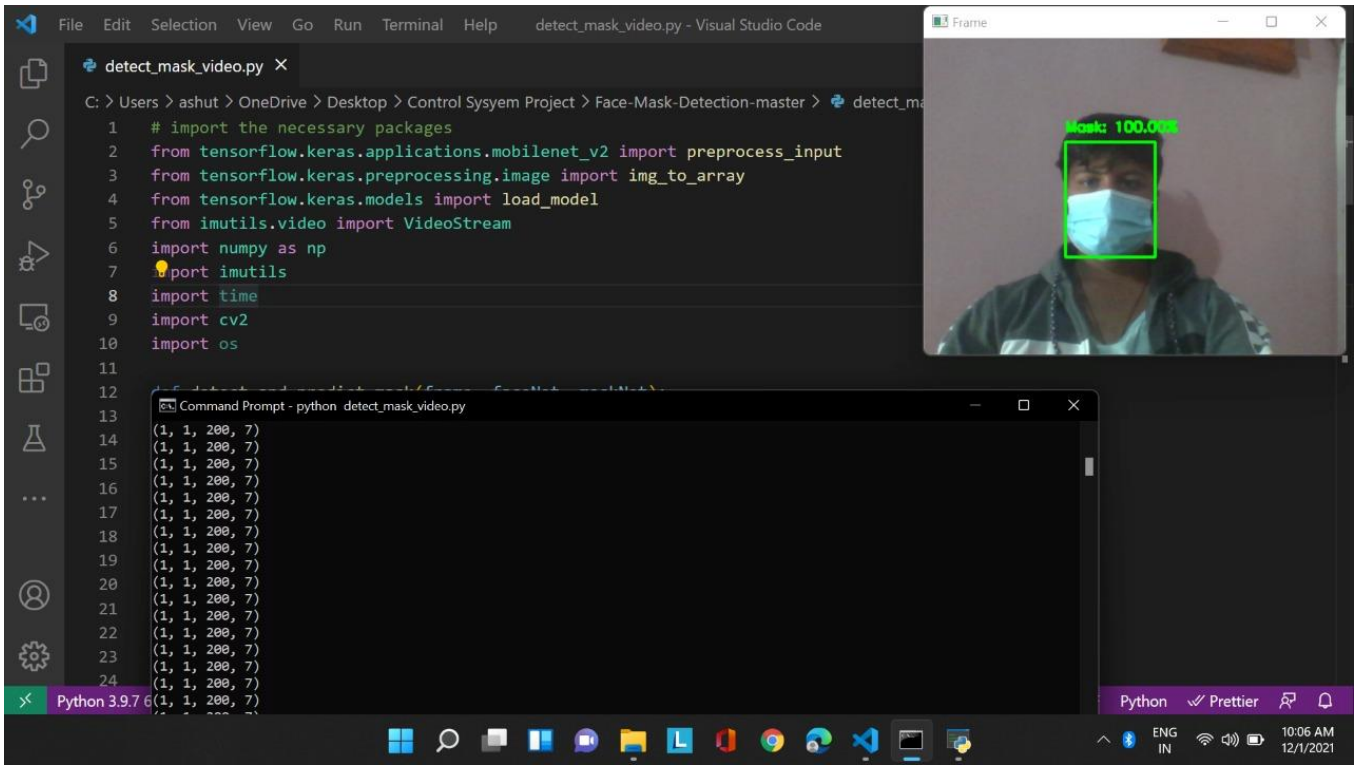
Detection Of Face Mask:

```
import tensorflow

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 VideoStreamimport 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="pathtofacedetectormodeldirectory") 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="minimumprobabilitytofilterweakdetections") 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]loadingfacemaskdetectormodel...") maskNet =load_model(args["model"])

print("[INFO] starting video stream...") vs

= VideoStream(src=0).start()time.sleep(2.0)


check = 0x=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" ifmask > withoutMask else"No Mask"color=(0, 255, 0) if label=="Mask" else (0,0, 255)
```

```
label="{}:{:.2f}%".format(label,max(mask,withoutMask)*100) ifcheck == 0 and mask<withoutMask:check = 1
y='defaulters/img' + str(x) + '.jpg'cv2.imwrite(y,frame)

x=x+1

elif

mask>withoutMask:

check = 0


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.destroyAllWindo  ws()  vs.stop()
```

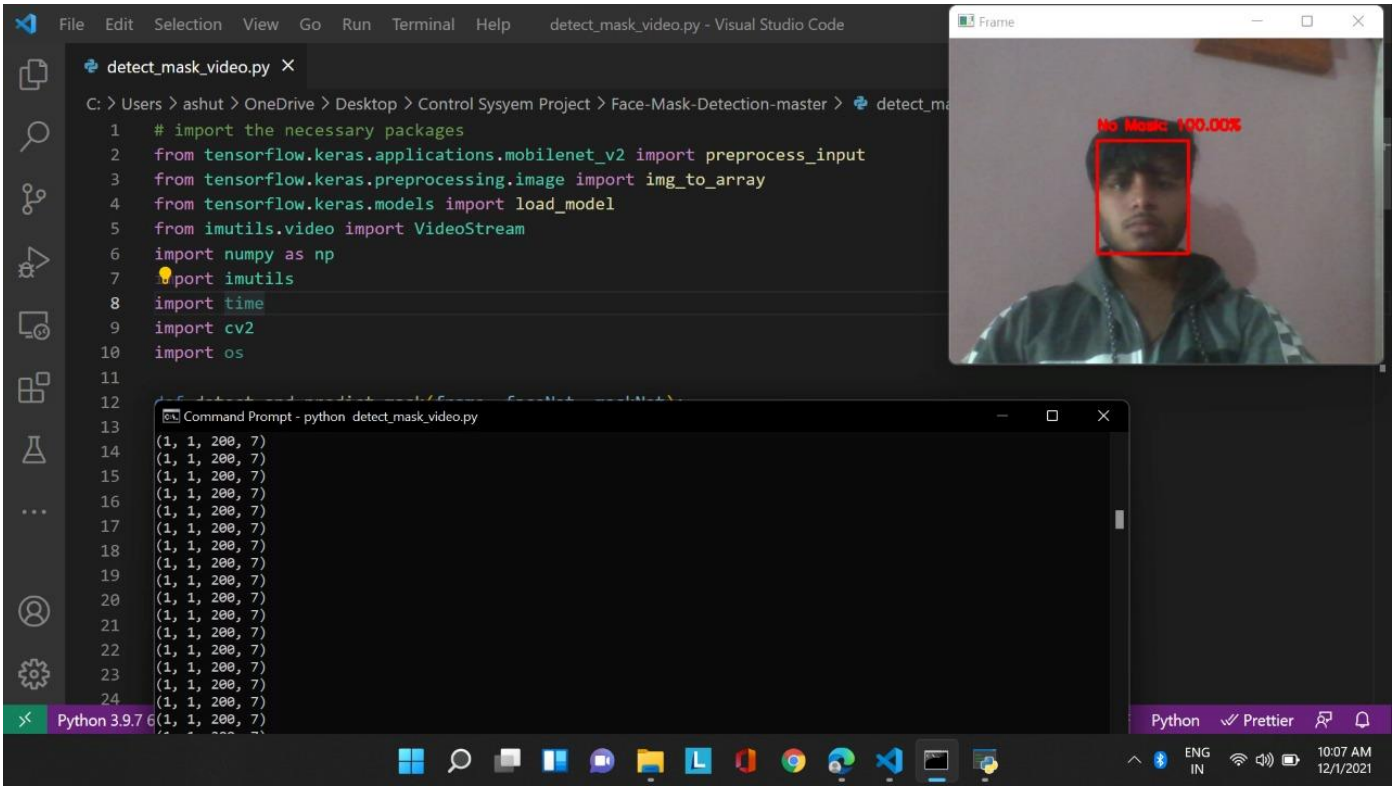
Result and Discussion



This was a demo of one of our teammates wearing a mask. Using the webcam of the laptop as input.

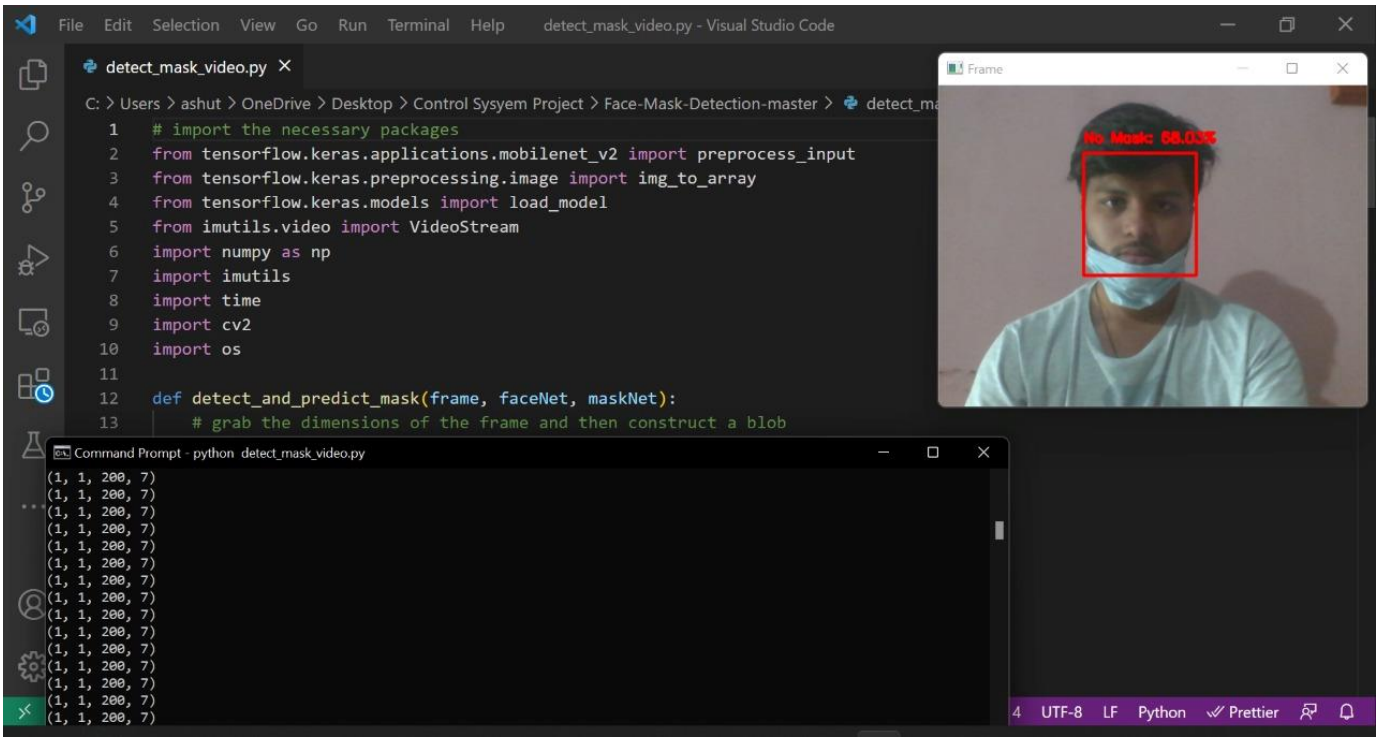
The mask worn is rightly confirmed by our model, which means it is successful in its first test, which was to identify correctly if a person has his/her mask on.

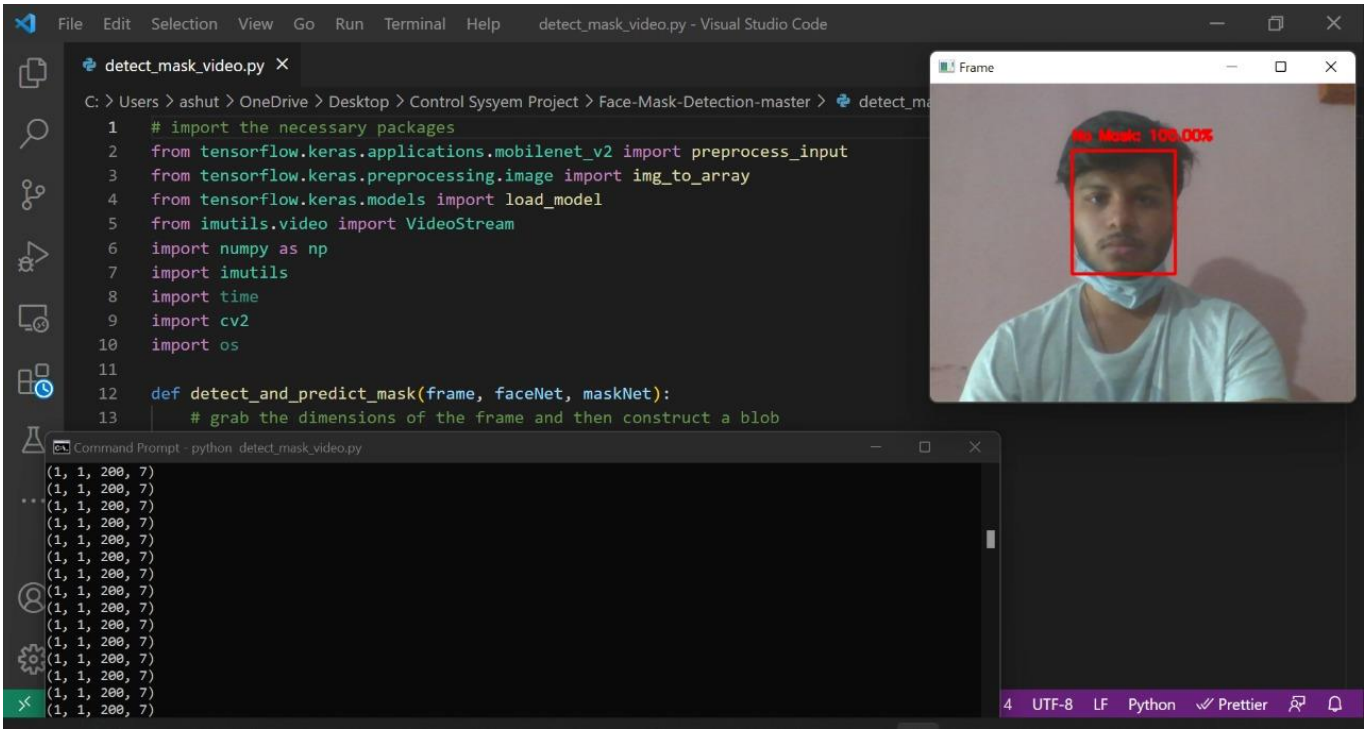
The accuracy of the model, that is the confidence level of the prediction, which is clearly very high at about 99.55%, which is very much in compliance with the industrial standards.

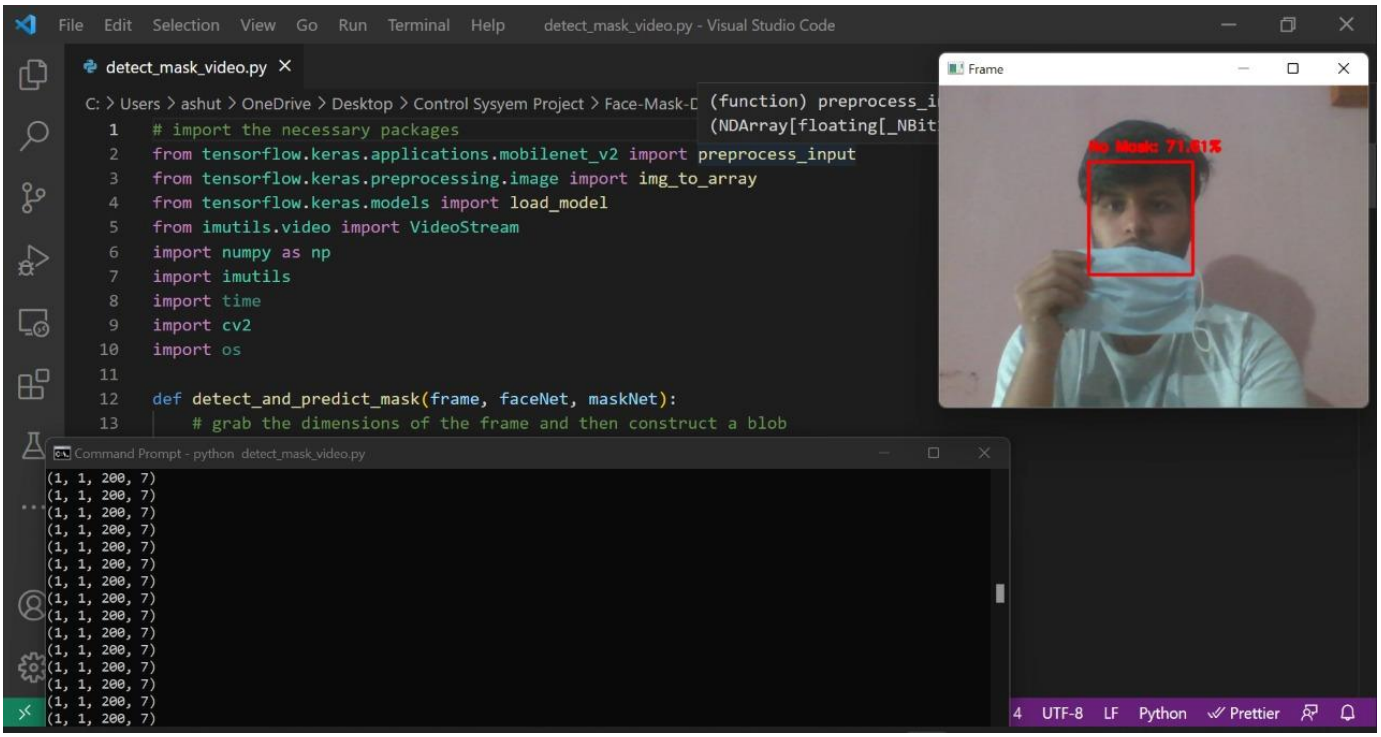


Here, our model correctly identifies the person without a mask, which fulfils its second objective, which is to identify the person if he/she is not wearing the mask.

The accuracy of the model, that is the confidence level of the prediction, which is clearlyvery high at about 99.81%, which is very much in compliance with the industrial standards.




File Edit Selection View Go Run Terminal Help detect_mask_video.py - Visual Studio Code
detect_mask_video.py
C: > Users > ashut > OneDrive > Desktop > Control Sysyem Project > Face-Mask-Detection-master > detect_ma
1 # import the necessary packages
2 from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
3 from tensorflow.keras.preprocessing.image import img_to_array
4 from tensorflow.keras.models import load_model
5 from imutils.video import VideoStream
6 import numpy as np
7 import imutils
8 import time
9 import cv2
10 import os
11
12 def detect_and_predict_mask(frame, faceNet, maskNet):
13 # grab the dimensions of the frame and then construct a blob
Command Prompt - python detect_mask_video.py
(1, 1, 200, 7)
Frame
No Mask: 100.00%
4 UTF-8 LF Python Prettier


File Edit Selection View Go Run Terminal Help detect_mask_video.py - Visual Studio Code
detect_mask_video.py
C: > Users > ashut > OneDrive > Desktop > Control Sysyem Project > Face-Mask-D
(function) preprocess_i
(NDArray[floating[_NBit
1 # import the necessary packages
2 from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
3 from tensorflow.keras.preprocessing.image import img_to_array
4 from tensorflow.keras.models import load_model
5 from imutils.video import VideoStream
6 import numpy as np
7 import imutils
8 import time
9 import cv2
10 import os
11
12 def detect_and_predict_mask(frame, faceNet, maskNet):
13 # grab the dimensions of the frame and then construct a blob
Command Prompt - python detect_mask_video.py
(1, 1, 200, 7)
Frame
4 UTF-8 LF Python Prettier

The program then saves the picture of the user without the mask into a folder for the first time. When this happens, we change our flag value so it avoids our system from going into an infinite loop and taking infinite pictures of the user and thus, also preventthe system fromgetting stuck.

Following this, an extra photo is stored in the same folder, this is because when the firstphotois captured, it stores the photo for the first time and the flag is set appropriately, and when the user wears the mask, the flag is reset and then no photos are captured. Again, when the user removes the mask, the system spots it through the camera, and then clicks a fresh picture of the user without the mask, updating the previous one

Conclusion and Future Work

We can say that our model works completely fine and is able to demonstrate the two main objectives, which is to identify the people with and without masks, and also do that with very high accuracy, as depicted in the above screenshots. Thus, we can conclude that our project meets the basic standards of working and accuracy and can bedeployed to the places mentioned in our objective and help us create ahealthier and safer environment to live in.

The basic Idea behind our project is to make a facial mask detector thatcan be used in apartments, offices, etc. through the CCTV cameras to later catch and penalise people who were not wearing masks.

We want to make our product is dynamic. That is, it should show results on the display screen in real time and also click(single) pictures of people who are not wearing a mask.

This will instill a fear of penalty among people and make them followrules. This will helpin creating a healthier work space environment inthe industrial/office areas. Also, it will help in maintaining a healthierand safer society to live in.

We will modify this project in future to obtain the desired output.

References

- A. Das, M. Wasif Ansari and R. Basak, "Covid-19 Face Mask Detection Using TensorFlow, Keras and OpenCV," 2020 IEEE 17th India Council International Conference (INDICON), New Delhi, India, 2020, pp. 1-5, doi: 10.1109/INDICON49873.2020.9342585. (Dec 2020)
- S. A. Sanjaya and S. Adi Rakhmawan, "Face Mask Detection Using MobileNetV2 in The Era of COVID-19 Pandemic," 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI), Sakheer, Bahrain, 2020, pp. 1-5, doi: 10.1109/ICDABI51230.2020.9325631. (Oct 2020)
- M. S. Islam, E. Haque Moon, M. A. Shaikat and M. Jahangir Alam, "A Novel Approach to Detect Face Mask using CNN," 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), Thoothukudi, India, 2020, pp. 800-806, doi: 10.1109/ICISS49785.2020.9315927. (Dec 2020)
- S. V. Militante and N. V. Dionisio, "Real-Time Facemask Recognition with Alarm System using Deep Learning," 2020 11th IEEE Control and System Graduate Research Colloquium (ICSGRC), Shah Alam, Malaysia, 2020, pp. 106-110, doi: 10.1109/ICSGRC49013.2020.9232610.
- Preeti Nagratha, Rachna Jaina and AgamMadan, “A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2,” Sustainable Cities and Society, Volume 66, March 2021, 102692
- Mamata S. Kalas, “Real time face detection and tracking using OPENCV” Proceedings of IRF International Conference, 5th & 6th February 2014, Pune India. ISBN: 978-93-82702-56-6