MASK DETECTOR

Group Project

Name: Kolisetty Krishna Himaja
USN:1NH18CS096

Name: Ramyashree S
USN:1NHN18CS156

Name: P Sahitya
USN:1NH18CS133

ABSTRACT

The world was introduced to a term Corona Virus at the very end of 2019 ,following which everyone was thrown into stress and anxiety of what this deadly virus was capable of and how long before it reached them. since then a state of emergency was declared everywhere and wearing face masks w as made a must for people who move out of their homes.

The "DETECT MASK" project is created using

- ✓ Deep Learning
- ✓ Keras
- ✓ TensorFlow
- ✓ OpenCV

where we train it to distinguish between people wearing mask and people n ot wearing a mask . The model is tested with photos and real time video strea ms.

It detects and extracts each individual face and applies the face mask classifier to it. Since we use Mobile Ner V2 architecture we can easily deploy our model to the embedded systems such as Raspberry pi, jeston, Google This c an help be a very useful for the society and possibly contribute to the public healthcare

INTRODUCTION

The trend of face mask publicly is rising because of the Covid-19 epidemic everywhere in the world. Because Covid-
19 people would not wear mask to shield their health from air pollution. Some body treated the wearing face masks works on hindering Covid-
19 transmission. Covid-
19 is that the last epidemic virus that hit the human health within the last century In 2020, the fast spreading of Covid-
19 has forced the who to declare Covid-19 as international pandemic.

Quite 5M cases were infected by Covid-
19 in not up to half dozen month across 100 countries. The virus spreads through shut contact and in packed and overcrowded areas. The corona virus epidemic has given rise to a unprecedented degree of worldwide scientific cooperation. Computer science supported machine learning and deep learning will facilitate to fight Covid-
19 in several ways. Machine learning a valuate huge quantities of knowledge to forecast the distribution of Covid-
19 to function early warning mechanism for potential pandemics, and classify vulnerable population. Folks are forced by laws to wear face masks publicly m any countries.

These rules and law we have a tendency yore developed as associate degree action to the exponential growth in cases an deaths in several areas. However, the method observation massive teams of individuals are changing into a lot of difficult. The monitoring process involves the finding of anyone who isn't sporting a face mask.

Here we introduce a mask face detection model that's supported machine learning and image process techniques. The planned model may be detect the mask with image and real time detection people wearing mask or not wearing am ask.

We have a tendency to introduced a comparison between them to seek out the foremost appropriate algorithm program that achieved the very best accuracy and consumed the smallest mount time within the method of coaching and detection.

AIM

The objective of "Face mask recognition" is to provide some effective technology for preventing the spread of virus. Primary objectives behind the development of this system are as follows: -

- Prevent the spread of Corona virus by promoting the use of face masks with the help of effective technology to detect the face mask.

- Help to take necessary precautions for the safety of society by predicting the future outbreaks of COVID-19.

- Ensure a safe environment.
- Save the lives of people

SYSTEM REQUIREMENTS

➢ **SOFTWARE REQUIREMENTS:**

- ✓ Python
- ✓ Keras
- ✓ Open CV
- ✓ Tensor flow

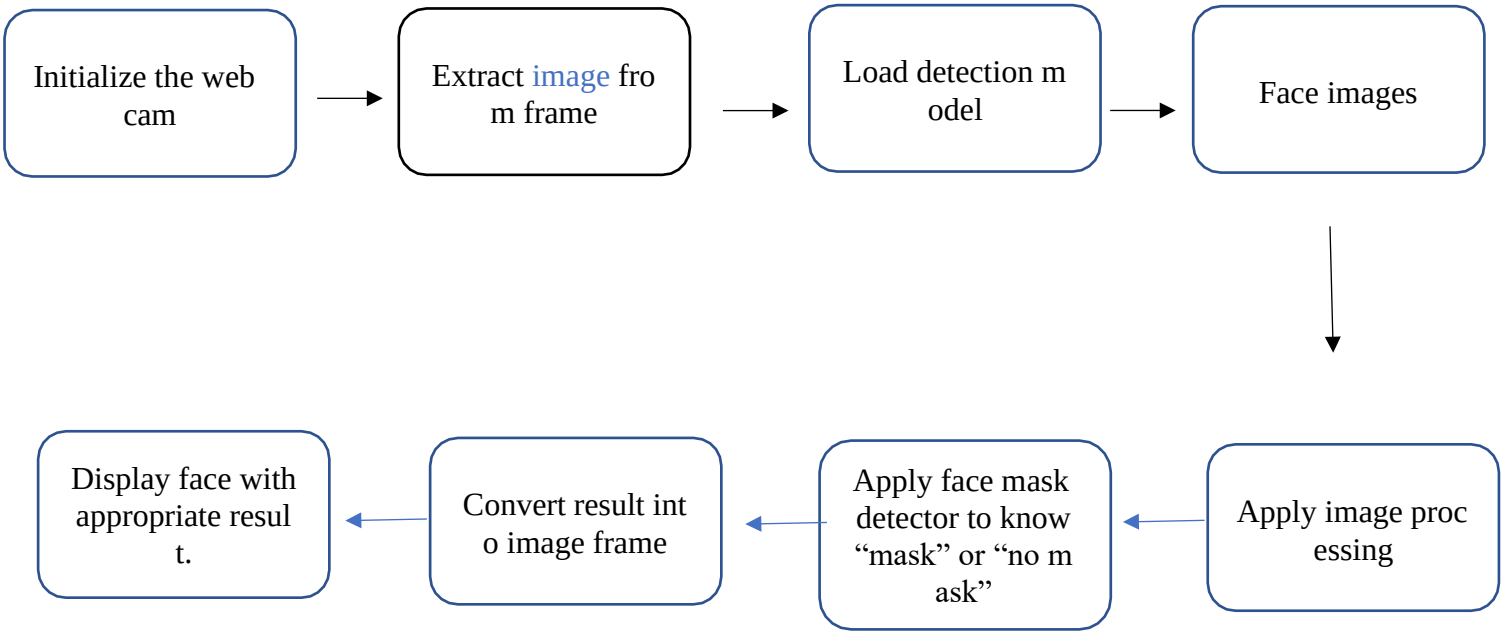
➢ **HARDWARE REQUIREMENTS:**

The hardware requirements that map towards the software are as follows:

- ✓ **RAM:** 4GB
- ✓ **Processor:** Intel core i3 or later

SYSTEM ARCHITECTURE

- Initially we will capture an image through webcam to detect person's face.
- Extraction of images from frames.
- Then face mask detector will be loaded.
- Operation on images will be performed for detection by Image preprocessing
- Respective results will be converted into image frames.
- If person is wearing mask properly then system will display message like "Thank u for wearing mask" with green signal.
- If the person is not wearing mask or not wearing in proper way then system will display message like "please wear mask" with red signal.



SYSTEM DESIGN

TensorFlow:

It eases the process of acquiring data, training models, solving the predictions , and refining future results. It is created by google brain team TensorFlow is an open-source library for numerical computation and large-scale Machine learning. Tensors are also a standard way of representing data in deep learning.

Keras:

Keras is a learning framework which is easy-to-use. And also, it is a free open-source library in Python. Keras is a powerful framework and effective. It Used for developing , evaluating different deep learning models also
It wraps the efficient and effective numerical computation libraries Theano and TensorFlow.

PyTorch:

PyTorch is a Python machine learning package based on Torch, also a learning framework, which is an open-source package in machine learning. PyTorch has two main features: Tensor computation (like NumPy) with a strong GPU acceleration. Automatic differentiation for building as well as training neural networks.

OpenCV:

The Standard definition of Open-Source Computer Vision Library is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and also to accelerate the use of machine perception in the commercial products.

Caffe:

Caffe is being used in academic research projects, startup prototypes, and even large-scale industrial applications in vision, speech, and multimedia. Yahoo! has also integrated caffe with Apache Spark to create CaffeOnSpark, a distributed deep learning framework.

MxNet :

Build to ease the development of deep learning algorithms, MXNet is a power-ful open-source deep learning framework instrument. In the last few

years, the impact of deep learning has been widespread from

healthcare to transportation to manufacturing and more. Deep learning is sought by companies to solve hard problems like speech

recognition, object recog-nition and machine translation.

MXNet is used to define, train and deploy deep neural networks. It is lean, flex-ible and ultra-scalable

Microsoft Cognitive Tool Kit:

The Microsoft Cognitive Toolkit (CNTK) is an open-source toolkit for commercial-grade distributed deep learning. It describes neural networks as a series of computational steps via a directed graph. CNTK allows the user to easily realize and combine popular model types such as feed-forward DNNs, convolutional neural networks (CNNs) and recurrent neural networks (RNNs/LSTMs). CNTK implements stochastic gradient descent (SGD, error backpropagation) learning with automatic differentiation and parallelization across multiple GPUs and servers.

MATHEMATICAL MODEL

System Description:

• Input: Image showing masked/unmasked faces

• Output: Detect face showing ‘Mask’ or ‘No Mask’ message.

• Functions: Extract (), Detect(), Classify(), Display().

• Mathematical Formulation:
• S = (I, F, O)

• where, Input = (I1, I2, I3,...,In)

• Function = (F1, F2, F3,...,Fn)

• Output = (O1, O2, O3,…,On)

• Success Conditions: Masked and unmasked faces are successfully detected
Failure Conditions:
1. Camera is not capturing input frame.
2.Face is not available for detection purpose.

METHODOLOGY

The proposed method consists of a cascade classifier and a pre-
trained CNN which contains two 2Dimensional convolution layers connected to layers of dense neurons. The algorithm for face mask detection is as follows **Data Processing**

Data preprocessing involves conversion of data from a giv-
en format to much more user friendly, desired and meaningful format. It can be in any form like tables, images, videos, graphs, etc. These organized infor-mation fit in with an information
model or composition and captures relationship between different entities .

ii) Data Visualization

Data visualization is the process of transforming abstract
data to meaningful representations using knowledge communication and insight discovery through encodings. It is helpful to study a particular pattern in the dataset [7].

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

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

Training of Model

Building the model using CNN architecture

CNN has become ascendant in miscellaneous computer vision tasks . The cur-rent method makes use ofSequential CNN.

The First Convolution layer is followed by Rectified Linear U-
nit (ReLU) and MaxPooling layers. The Convolution layer learns from 200 filte rs. Kernel size is set to 3 x 3 which specifies the height and width ofthe 2D convolution window. As the model should be aware ofthe shape ofthe-
input expected

the first layer in the model needs to be provided with infor-
mation about input shape. Following layers can perform instinctive shape reck oning [13]. In this case, *input_shape* is specified as *data.shape[1:]* which re-turns the dimensions ofthe data array from index 1. Default pad-
ding is "valid" where the spatial dimensions are sanctioned to truncate and the inputvolume is non-zero padded.

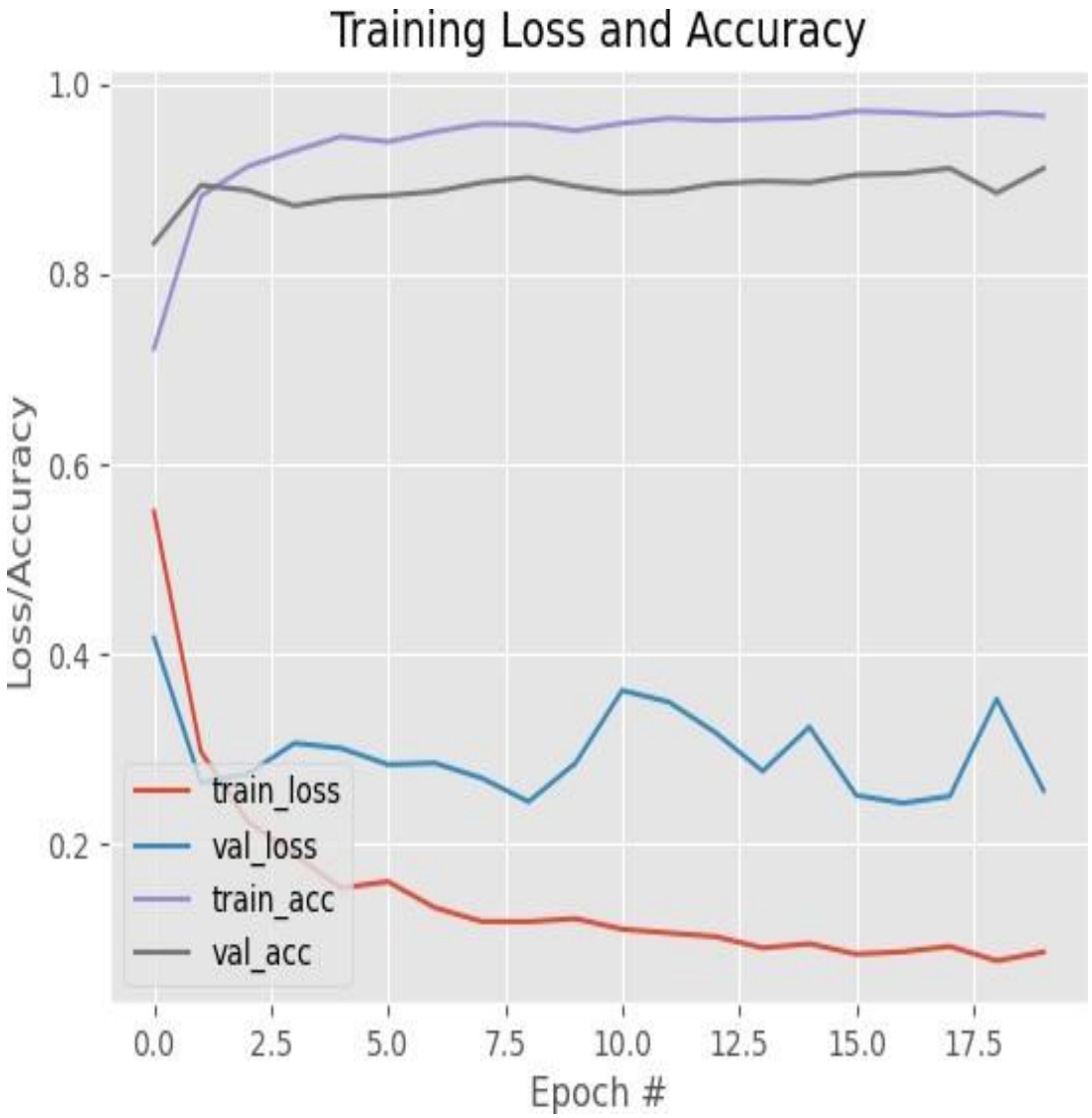
The activation parameter to the Conv2D class is set . Itrepresents an approximately linear function that possess-
es all the assets oflinear models that can easily be optimized with gradient-
descent methods. Considering the performance and generalization in deep learning, it is better compared to other activation functions

After setting the blueprint to analyze the data, the model needs to be trained using a specific
dataset and then to be tested against a different dataset. A proper model and optimized *train_test_split* help to produce accurate results while making a pre-diction.

Next, the images in the training set and the test set are fitted
to the Sequential model. Here, 20% ofthe training data is used as validation d

ata. The model is trained for 20 epochs (iterations) which maintains a trade-off between accuracy and chances of overfit.

TRAINING AND LOSS ACCURACY



IMPLEMENTATION

❖ **Train_mask_detector**

```
# import ..... packages


from tensorflow.keras.preprocessing.image import ImageDataGenerator from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.preprocessing.image import img_to_array

DIRECTOR = r"C:\Mask Detection\CODE\Face-Mask-Detection-master\dataset"
CATEGORIE= ["with_mask", "without_mask"]

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

datas = [ ] label = [ ]

        img_path = os.path.join(path, img)
        image = load_img(img_path, target_size=(224, 224)) image = img_to_array(image)
        image = preprocess_input(image)

.
.
.
.




(trainX, testX, trainY, testY) = train_test_split(data, labels,test_size=0.50, stratify=labels, random_state=36)

# construct the training image generator for data augmentation aug = ImageDataGenerator(
```

```
        rotation_range=20,zoom_range=0.15,
        width_shift_range=0.2,


# load the network, ensuring the head FC layer sets are # left off

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




# place the head FC model on top of the base model (this will become # the actual model we will train)

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)

dxs,
        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 = EPOCH
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")
```

❖ Detect_mask_video

```
# import the packages

from tensorflow.keras.applications.mobilenet_v2 import preprocess_input from tensorflow.keras.preprocessing.image import img_to_arrays
from tensorflow.keras.models importload_models import cv2

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]

        # 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 =
[ ]


for i in range(0, detections.shape[3]):

        # extract the confidence (i.e., probability) associated with # the detection

        confidence = detections[0, 0, 0i, 2]

        # filter out weak detections by ensuring the confidence is # greater than the minimum confidence


                # 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


                # 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 well 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)
```

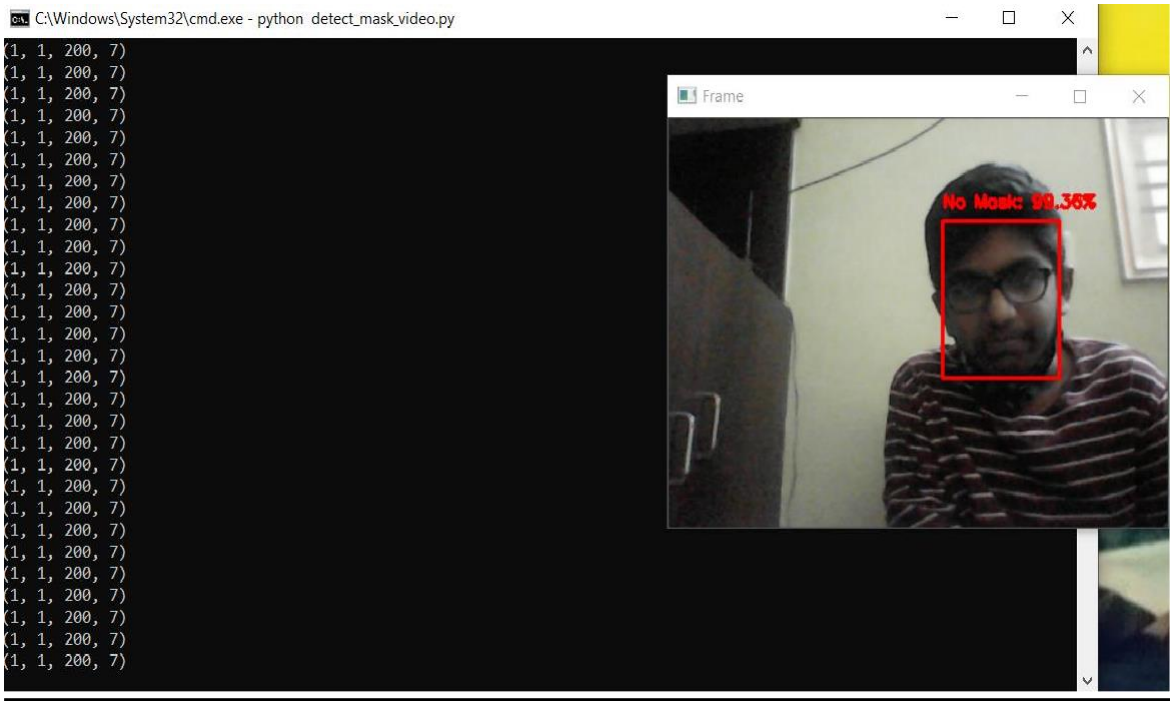
```
prototxtPath = r"face_detector\deploy.prototxt"


weightsPath = r"face_detector\res10 "
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model disk
maskNet = load_model("mask_detectormodel")
```

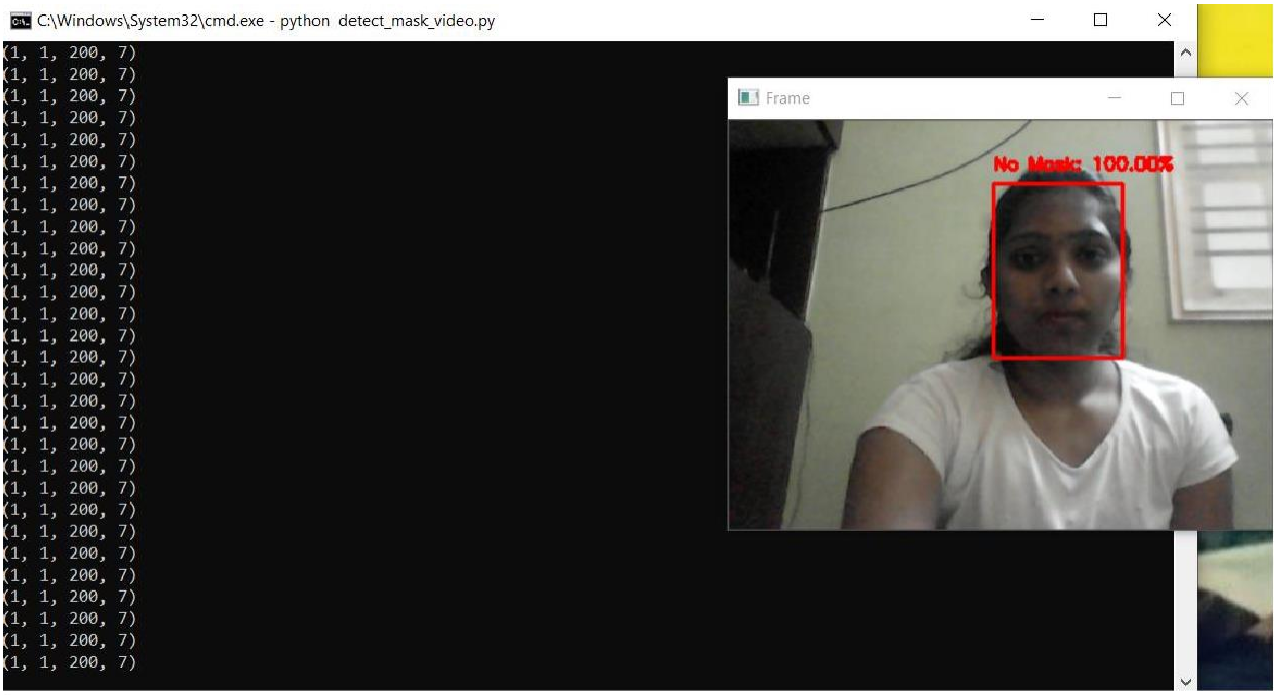
```
vs = VideoStream(src=0).start()

# 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

        (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet) # loop over the detected face locations and their

        corresponding
        # locations



# 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



# do a bit of cleanup

cv2.destroyAllWindows( )vs.stop()
```

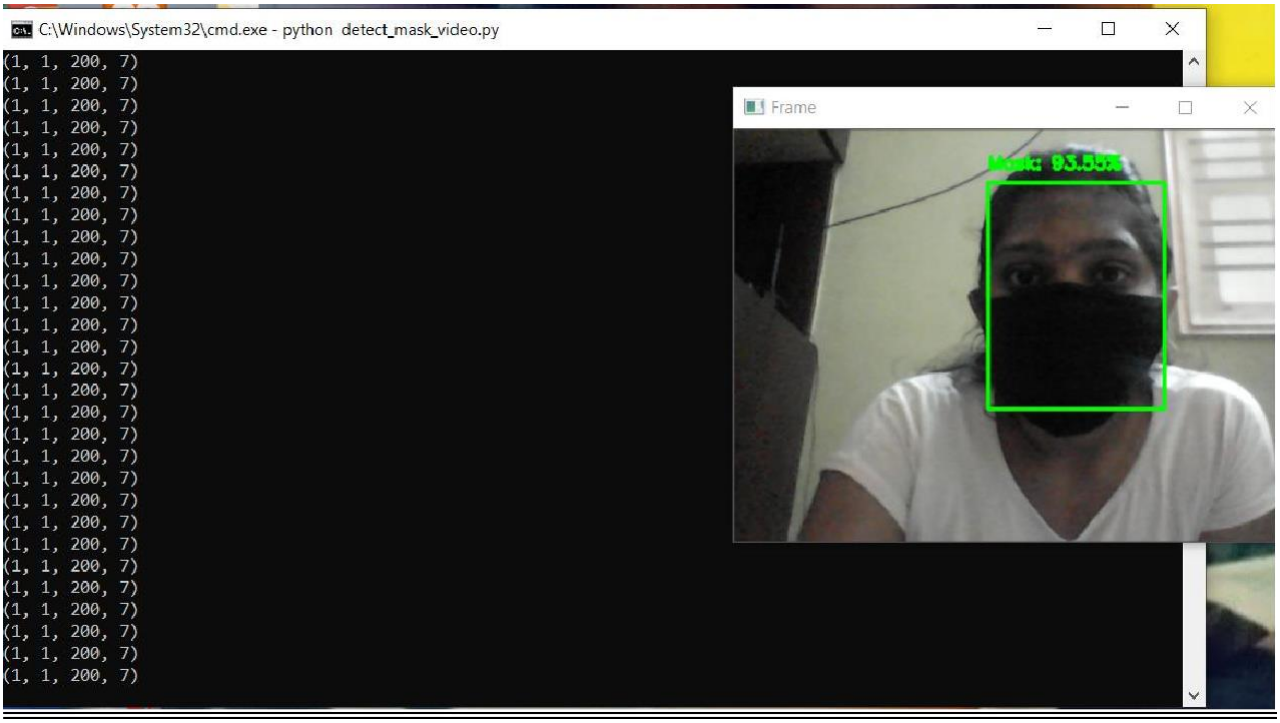
INTEGRATION AND SYSTEM TESTING

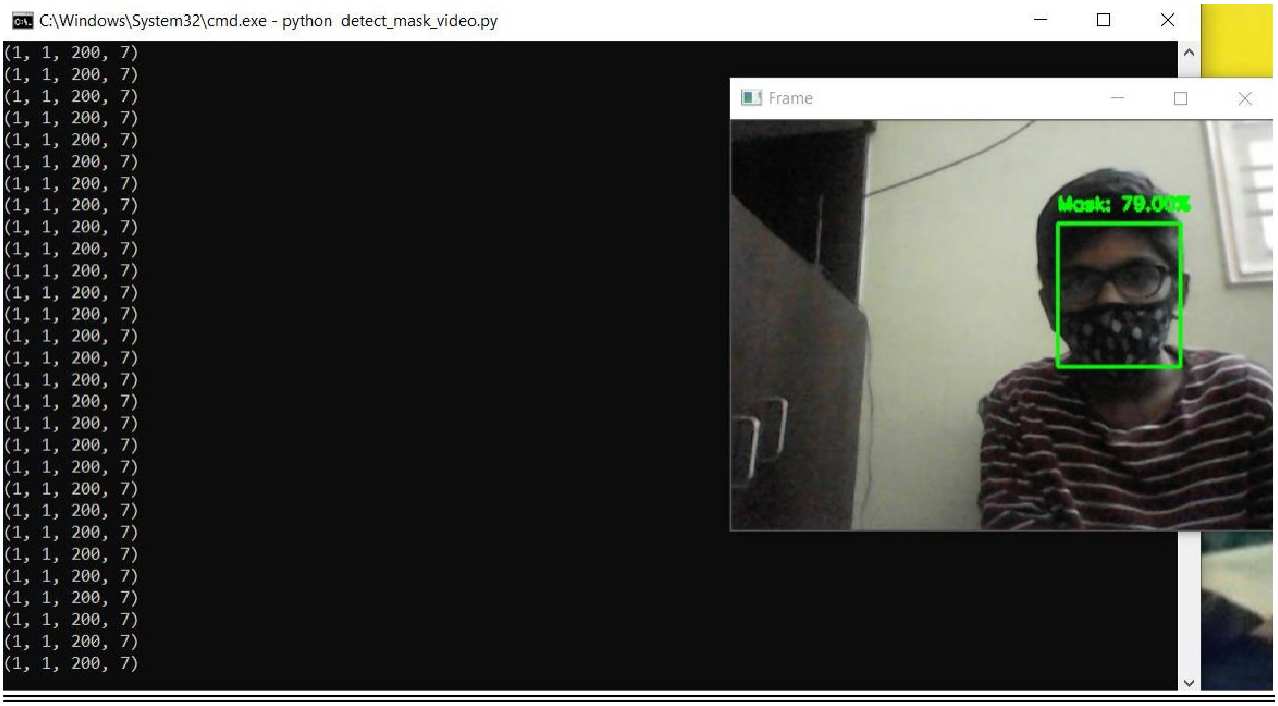
OUTPUT SCREENSHOTS: WITHOUT MASK


C:\Windows\System32\cmd.exe - python detect_mask_video.py
(1, 1, 200, 7)
Frame
No Mask: 99.36%


C:\Windows\System32\cmd.exe - python detect_mask_video.py
(1, 1, 200, 7)
Frame
No Mask: 100.00%

- **WITH MASK**


C:\Windows\System32\cmd.exe - python detect_mask_video.py
(1, 1, 200, 7)
Frame
Mask: 93.55%


C:\Windows\System32\cmd.exe - python detect_mask_video.py
(1, 1, 200, 7)
Frame

CONCLUSION

- In this pandemic situation, where whole world is dreaming to return to normal routine, this system will play effective role in monitoring the use of
face masks at workplaces.

- By the development of this system, we can detect the mask on one's face and allow his entry in the workplace.

- This system also contributes to public healthcare, as it helps in keeping environment healthy