DEPARTMENT OF COMPUTER SCIENCE (UBIT)

COVID-19 MASK DETECTION SYSTEM (MDS)

A PROJECT REPORT IS SUBMITTED TO
UNIVERSITY OF KARACHI

SUBMITTED BY:

Raheel Ahmed Khan – EP19101067

(Sr. Software Engineer and Development Manager in miniMAX Solution Pvt.)

Adnan Shahzad – EP19101003

(Sr. Network Engineer in Wateen telecom and Solutions)

Naseer Uddin – EP19101062

(Principal Computer official at SPD, Pakistan Navy)

ACKNOWLEDGEMENT

I thank the almighty for giving us the courage and perseverance in completing the main project. This project itself is acknowledgements for all those people who have give us their heartfelt co-operation in making this project a grand success.

I extend our sincere thanks to **AST. PROFESSOR SIR KHALID JAMAL**, for providing sufficient infrastructure and good environment in the department to complete our course.

TABLE OF CONTENTS

INTRODUCTION:

Face Mask Detector with Computer vision in Python – Critical project for the safety against COVID-19

The hottest topic in the industry is computer vision, similar to humans, computers can also see and take a decision.

In this project, we will build a model to detect whether a person is wearing mask in real-time. We will do this using the concepts of computer vision using the Python, OpenCV library Tensorflow, Keras and couple other libraries for data intelligence and machine learning.

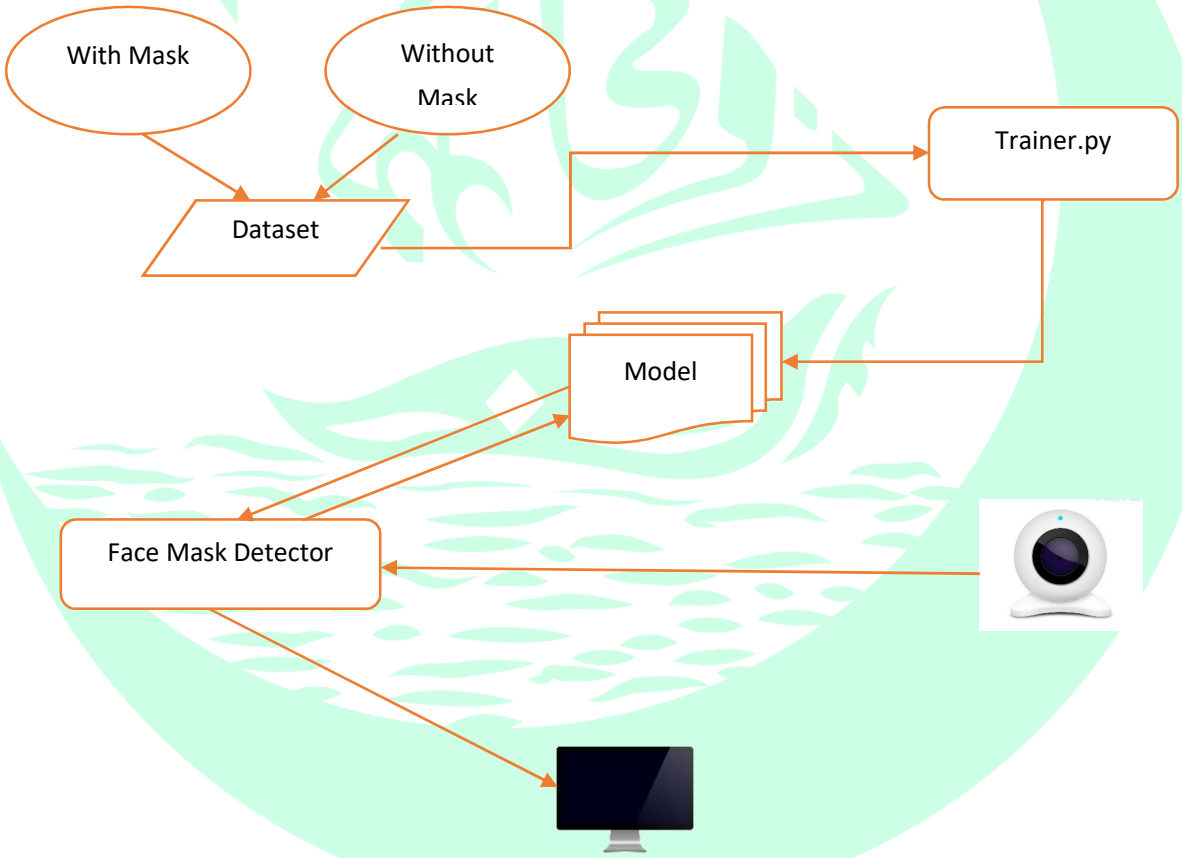
Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled.

To make this possible we have collected number of images of people with mask and without wearing mask for our dataset. Then we train our AI model to understand and process our dataset after that we finally become able to detect mask on face form webcam stream through OpenCV2.

SYSTEM DESIGN

We have divided this project into three major tasks:

1- Get live video stream from webcam
2- Collect and train dataset of with and without mask of faces
3- Detect face masks in real-time video streams



LIBRARIES AND TECHNOLOGIES:

PYTHON

The python language is one of the most accessible programming languages available because it has simplified syntax and not complicated, which gives more emphasis on natural language. Due to its ease of learning and usage, python codes can be easily written and executed much faster than other programming languages.

The python language is one of the most accessible programming languages available because it has simplified syntax and not complicated, which gives more emphasis on natural language. Due to its ease of learning and usage, python codes can be easily written and executed much faster than other programming languages.

Python is a general-purpose coding language—which means that, unlike HTML, CSS, and JavaScript, it can be used for other types of programming and software development besides web development. That includes back end development, software development, data science and writing system scripts among other things.

COVID-19: Face Mask Detector with OpenCV, Keras/TensorFlow, We'll use this Python script to train a face mask detector ... I like to define my deep learning hyperparameters in one place.

NUMPY

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. ... It also discusses the various array functions, types of indexing, etc.

What does Numpy gradient do?

Gradient. Return the gradient of an N-dimensional array. The gradient is computed using second order accurate central differences in the interior points and either first or second order accurate one-sides (forward or backwards) differences at the boundaries.

What is the NumPy in Python?

NumPy is a Python library that provides a simple yet powerful data structure: the n-dimensional array. This is the foundation on which almost all the power of Python's data science toolkit is built, and learning NumPy is the first step on any Python data scientist's.

In the code, we:

Import the numpy package.

Pass the list of lists wines into the array function, which converts it into a NumPy array. Exclude the header row with list slicing. Specify the keyword argument dtype to make sure each element is converted to a float. We'll dive more into what the dtype is later on.

OPEN CV2

OpenCV is the most popular library for computer vision. Originally written in C/C++, it now provides bindings for Python.

OpenCV uses machine learning algorithms to search for faces within a picture. Because faces are so complicated, there isn't one simple test that will tell you if it found a face or not. Instead, there are thousands of small patterns and features that must be matched. The algorithms break the task of identifying the face into thousands of smaller, bite-sized tasks, each of which is easy to solve. These tasks are also called classifiers.

For something like a face, you might have 6,000 or more classifiers, all of which must match for a face to be detected (within error limits, of course). But therein lies the problem: for face detection, the algorithm starts at the top left of a picture and moves down across small blocks of data, looking at each block, constantly asking, "Is this a face? … Is this a face? … Is this a face?" Since there are 6,000 or more tests per block, you might have millions of calculations to do, which will grind your computer to a halt.

To get around this, OpenCV uses cascades. What's a cascade? The best answer can be found in the dictionary: "a waterfall or series of waterfalls."

Like a series of waterfalls, the OpenCV cascade breaks the problem of detecting faces into multiple stages. For each block, it does a very rough and quick test. If that passes, it does a slightly more detailed test, and so on. The algorithm may have 30 to 50 of these stages or cascades, and it will only detect a face if all stages pass.

The advantage is that the majority of the picture will return a negative during the first few stages, which means the algorithm won't waste time testing all 6,000 features on it. Instead of taking hours, face detection can now be done in real time.

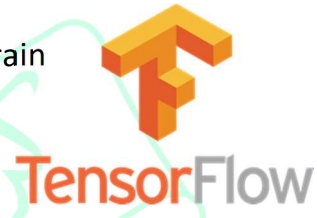
Cascades in Practice

Though the theory may sound complicated, in practice it is quite easy. The cascades themselves are just a bunch of XML files that contain OpenCV data used to detect objects. You initialize your code with the cascade you want, and then it does the work for you.

Since face detection is such a common case, OpenCV comes with a number of built-in cascades for detecting everything from faces to eyes to hands to legs. There are even cascades for non-human things. For example, if you run a banana shop and want to track people stealing bananas, this guy has built one for that!

TENSORFLOW

TensorFlow provides a collection of workflows to develop and train models using Python, JavaScript, or Swift, and to easily deploy.



TensorFlow is an open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning. TensorFlow was originally developed for large numerical computations without keeping deep learning in mind.

What is TensorFlow used for?

Created by the Google Brain team, TensorFlow is an open source library for numerical computation and large-scale machine learning. TensorFlow bundles together a slew of machine learning and deep learning (aka neural networking) models and algorithms and makes them useful by way of a common metaphor.

Is TensorFlow written in Python?

The most important thing to realize about TensorFlow is that, for the most part, the core is not written in Python: It's written in a combination of highly-optimized C++ and CUDA (Nvidia's language for programming GPUs).

KERAS

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.

What is keras and TensorFlow?

Keras is a neural network library while TensorFlow is the open-source library for a number of various tasks in machine learning. TensorFlow provides both high-level and low-level APIs while Keras provides only high-level APIs. ... Both frameworks thus provide high-level APIs for building and training models with ease.

Use keras or TF keras?

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. ... Since Keras provides APIs that TensorFlow has already implemented (unless CNTK and Theano overtake TensorFlow which is unlikely), tf. keras would keep up with Keras in terms of API diversity.

What is keras and TensorFlow?

Keras is a neural network library while TensorFlow is the open-source library for a number of various tasks in machine learning. TensorFlow provides both high-level and low-level APIs while Keras provides only high-level APIs. ... Both frameworks thus provide high-level APIs for building and training models with ease.28-Jun-2019

PROCESS OF FACE DETECTION:

First of all we need to install required modules in system, we are developing this system in Windows 10 with **Python version 3.8.1**

Now we need to need to install open cv2 library to detect face

```
pip install opencv-python
```

This should install the latest version of open cv2 in system

Now create a .py file and with the following code we can simply detect faces through webcam stream

First import cv2 library in .py file and creating creating a face cascade

```python
import cv2
import sys

cascPath = sys.argv[1]
faceCascade = cv2.CascadeClassifier(cascPath)
```

This line sets the video source to the default webcam, which OpenCV can easily capture.

```python
video_capture = cv2.VideoCapture(0)
```

Here, we capture the video. The read() function reads one frame from the video source, which in this example is the webcam. This returns:

The actual video frame read (one frame on each loop)

A return code

The return code tells us if we have run out of frames, which will happen if we are reading from a file. This doesn't matter when reading from the webcam, since we can record forever, so we will ignore it.

```python
while True:
    # Capture frame-by-frame
    ret, frame = video_capture.read()
```

Now we are merely searching for the face in our captured frame.

Python

```
    # Capture frame-by-frame
    ret, frame = video_capture.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(30, 30),
        flags=cv2.cv.CV_HAAR_SCALE_IMAGE
    )

    # Draw a rectangle around the faces
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

    # Display the resulting frame
    cv2.imshow('Video', frame)
```

Again, this code should be familiar. We are merely searching for the face in our captured frame.

We wait for the ‘q’ key to be pressed. If it is, we exit the script.

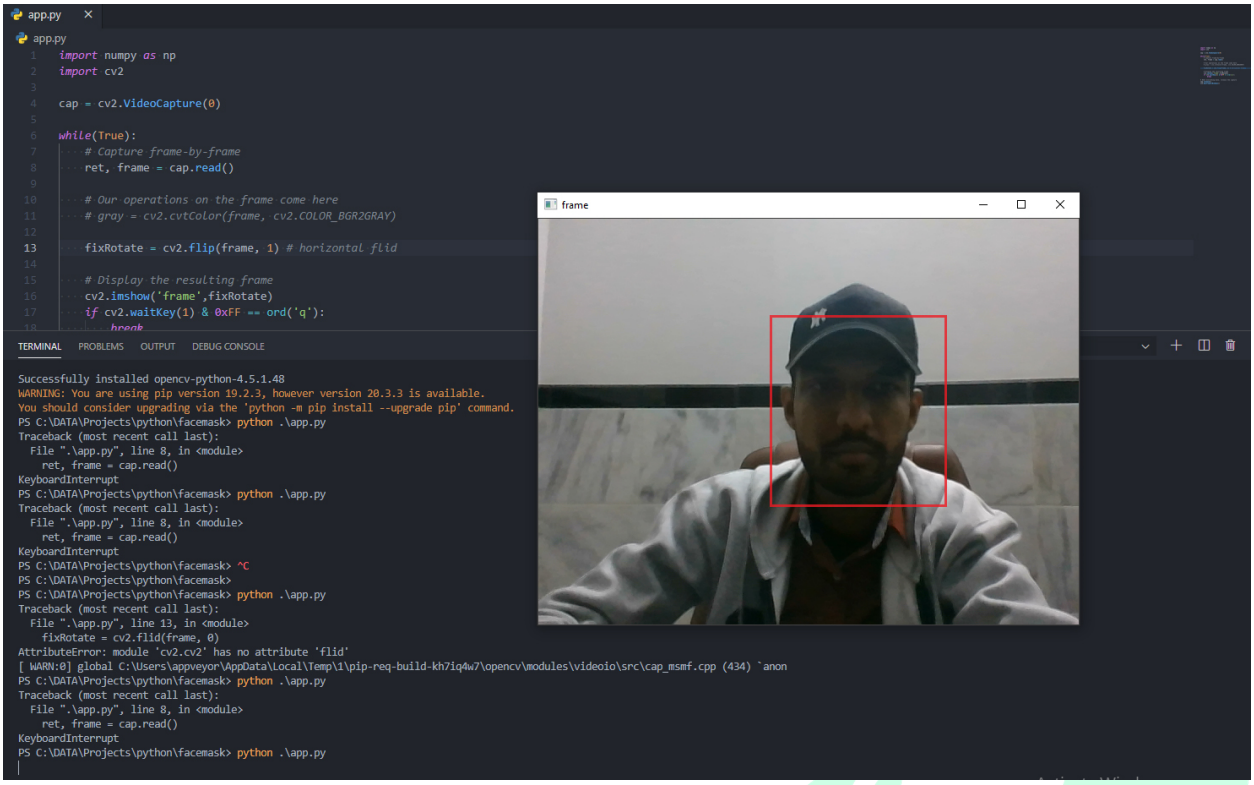
Python

```
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

Python

```
# When everything is done, release the capture
video_capture.release()
cv2.destroyAllWindows()
```

Now we do some testing



PROCESS OF DATA MODEL AND TRANING

Now we are going to uses these high-level TensorFlow concepts:

- Use TensorFlow's default eager execution development environment,
- Import data with the Datasets API,
- Build models and layers with TensorFlow's Keras API.

First of all we need to install tensorflow and other required libraries

- pip3 install tensorflow
- pip3 install keras
- pip3 install imutils
- pip3 install numpy

SETUP DATASET

Now we need to create two folders in root of the project with the name of with_mask and without_mask.

Then we will save some images of people with mask in with_mask folder and some images of people without mask in without_mask folder

For this project we have downloaded opensource dataset of people faces form ffhq-dataset so this speedup our process to create dataset of without mask people faces

And for the with faces dataset we also used opensource dataset from Real-World Masked Face Dataset by X-zhangyang so we can have large number of images of people with mask

TRAIN MODEL

Now we have already setup our dataset so now we will create another .py file to train or model

First of all we will create two arrays for images and labels and store all images from dataset into those array.

```
data = []
labels = []

for category in CATEGORIES:
    path = os.path.join(DIRECTORY, category)
    for img in os.listdir(path):
        img_path = os.path.join(path, img)
        image = load_img(img_path, target_size=(224, 224)) # resize image for faster process
        image = img_to_array(image)
        image = preprocess_input(image)

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


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

Then we will user numpy array function to convert images and labels to numpy array and also set images datatype to float32

Then we will use keras preprocessing image function to generate data augmentation

```
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)

# 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")


baseModel = MobileNetV2(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3)))

```

After that we will create actual model from keras models function

```

    # load keras model and place the head FC model on top
    model = Model(inputs=baseModel.input, outputs=headModel)

```

Then we will train our model using tensorflow and keras and save our model in mast_detector.model file

```
# 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")

```

Now we need to run this train model file to train our dataset

```
PS C:\DATA\Projects\python\facemask> python .\train_model.py
2021-02-11 02:10:07.852071: W tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load dynamic library 'cudart64_101.dll'; dlerror: cudart64_101.dll not found
2021-02-11 02:10:07.852923: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
[INFO] loading images...
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [96, 128, 160, 192, 224]. Weights for input shape (224, 224) will be loaded as the default.
2021-02-11 02:10:17.229963: W tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll not found
2021-02-11 02:10:17.254868: W tensorflow/stream_executor/cuda/cuda_driver.cc:312] failed call to cuInit: UNKNOWN ERROR (303)
2021-02-11 02:10:17.268767: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: DESKTOP-UI2G85U
2021-02-11 02:10:17.270436: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-UI2G85U
2021-02-11 02:10:17.306579: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN)to use the following CPU instructions in performan
ce-critical operations:  AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-02-11 02:10:17.362848: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x216f32ee2f0 initialized for platform Host (this does not guarantee that XLA will be used). Devices:
2021-02-11 02:10:17.363507: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Host, Default Version
[INFO] compiling model...
[INFO] training head...
Epoch 1/20
8/8 [==============================] - 9s 1s/step - loss: 1.0855 - accuracy: 0.4375 - val_loss: 0.6033 - val_accuracy: 0.7231
Epoch 2/20
8/8 [==============================] - 7s 904ms/step - loss: 0.5349 - accuracy: 0.7689 - val_loss: 0.4761 - val_accuracy: 0.8308
Epoch 3/20
8/8 [==============================] - 7s 854ms/step - loss: 0.5143 - accuracy: 0.8178 - val_loss: 0.4236 - val_accuracy: 0.8308
Epoch 4/20
8/8 [==============================] - 8s 962ms/step - loss: 0.4158 - accuracy: 0.8359 - val_loss: 0.3550 - val_accuracy: 0.8308
Epoch 5/20
8/8 [==============================] - 7s 917ms/step - loss: 0.3582 - accuracy: 0.8533 - val_loss: 0.3038 - val_accuracy: 0.8615
Epoch 6/20
8/8 [==============================] - 7s 900ms/step - loss: 0.3261 - accuracy: 0.8800 - val_loss: 0.2622 - val_accuracy: 0.8769
Epoch 7/20
8/8 [==============================] - 7s 874ms/step - loss: 0.2689 - accuracy: 0.8889 - val_loss: 0.2277 - val_accuracy: 0.8769
Epoch 8/20
8/8 [==============================] - 7s 842ms/step - loss: 0.2291 - accuracy: 0.9111 - val_loss: 0.1962 - val_accuracy: 0.9077
Epoch 9/20
8/8 [==============================] - 7s 870ms/step - loss: 0.2076 - accuracy: 0.9422 - val_loss: 0.1735 - val_accuracy: 0.9231
Epoch 10/20
8/8 [==============================] - 8s 1s/step - loss: 0.2082 - accuracy: 0.9156 - val_loss: 0.1510 - val_accuracy: 0.9692
Epoch 11/20
8/8 [==============================] - 9s 1s/step - loss: 0.1818 - accuracy: 0.9289 - val_loss: 0.1420 - val_accuracy: 0.9385
Epoch 12/20
8/8 [==============================] - 7s 904ms/step - loss: 0.1295 - accuracy: 0.9556 - val_loss: 0.1290 - val_accuracy: 0.9538
Epoch 13/20
8/8 [==============================] - 7s 851ms/step - loss: 0.1421 - accuracy: 0.9556 - val_loss: 0.1196 - val_accuracy: 0.9538
Epoch 14/20
8/8 [==============================] - 7s 861ms/step - loss: 0.1267 - accuracy: 0.9467 - val_loss: 0.1115 - val_accuracy: 0.9538
Epoch 15/20
8/8 [==============================] - 7s 863ms/step - loss: 0.1389 - accuracy: 0.9378 - val_loss: 0.1006 - val_accuracy: 0.9692
Epoch 16/20
8/8 [==============================] - 10s 1s/step - loss: 0.1387 - accuracy: 0.9556 - val_loss: 0.0905 - val_accuracy: 0.9692
Epoch 17/20
8/8 [==============================] - 14s 2s/step - loss: 0.0850 - accuracy: 0.9805 - val_loss: 0.0830 - val_accuracy: 0.9846
Epoch 18/20
8/8 [==============================] - 14s 2s/step - loss: 0.0873 - accuracy: 0.9844 - val_loss: 0.0759 - val_accuracy: 0.9846
```



```
8/8 [==============================] - 14s 2s/step - loss: 0.0873 - accuracy: 0.9844 - val_loss: 0.0759 - val_accuracy: 0.9846
Epoch 19/20
8/8 [==============================] - 8s 988ms/step - loss: 0.1000 - accuracy: 0.9733 - val_loss: 0.0687 - val_accuracy: 0.9846
Epoch 20/20
8/8 [==============================] - 7s 813ms/step - loss: 0.0799 - accuracy: 0.9778 - val_loss: 0.0643 - val_accuracy: 0.9846
              precision    recall  f1-score   support

   with_mask       1.00      0.91      0.95        11
without_mask       0.98      1.00      0.99        54
    accuracy                           0.98        65
   macro avg       0.99      0.95      0.97        65
weighted avg       0.98      0.98      0.98        65
```

Once our model is train then we can move to our old file to detect facemask.

PROCESS OF DETECTING FACEMASK

Now we will open our old file and new method to detect facemask

This method get images frame in parameter and process image to detect weather face has mask or not.

First we will convert image to blob using cv2 and store in separate variable

Then we will also create 3 another variable to store data of faces, and the location of each faces and also the rate of prediction of each face weather it has mask or not

```
12
13  def detect_and_predict_mask(frame, faceNet, maskNet):
14
15      (h, w) = frame.shape[:2]
16      blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224), (104.0, 177.0, 123.0))
17
18
19      faceNet.setInput(blob)
20      detections = faceNet.forward()
21      #print(detections.shape)
22
23
24      faces = [] # use to store multiple faces
25      locs = [] # use to store locations of each face
26      preds = [] # for prediction data of each face
27
```

Now we will use for loop for all the faces in image frame, and also check confidence of face if the face has at least 50% of confidence because we don't want to run our algorithm on blur face

And store face data in faces variable and its location in location variable

```
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))


        face = frame[startY:endY, startX:endX]
        face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
        face = cv2.resize(face, (224, 224)) # using small image size otherwise its slowing the process
        face = img_to_array(face)
        face = preprocess_input(face)
        #print(face)

        # for multiple faces
        faces.append(face)
        locs.append((startX, startY, endX, endY))
```

Now instead of checking each face in for loop we will convert faces through numpy array and process all faces at once this will save lot of system resources.

```
    # check if at least one face was detected
    if len(faces) > 0:
        # check all faces in bulk using numpy
        faces = np.array(faces, dtype="float32")
        preds = maskNet.predict(faces, batch_size=32)

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

The rest of the code is same we need to detect face using open cv and load our data model and process each video frame

We will also create boxes on face and also show label if face has mask or not

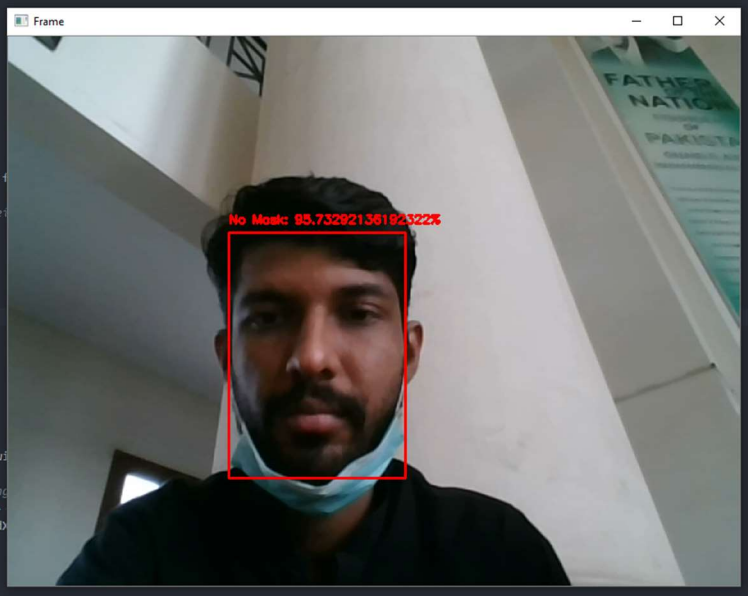
We will also change color of each boxes based on the detect result

```python
# Loop over the frames from the video stream
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=800)

    # run face dector on each frame of video
    (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

    # loop over the detected face locations and their corresponding predection
    for (box, pred) in zip(locs, preds):

        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred

        label = "Mask"
        color = (0, 255, 0)
        if mask < withoutMask:
            label = "No Mask"
            color = (0, 0, 255)


        # include the probability in the label
        label = "{}: {}%".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
```

Now all is done we will test the result ☺

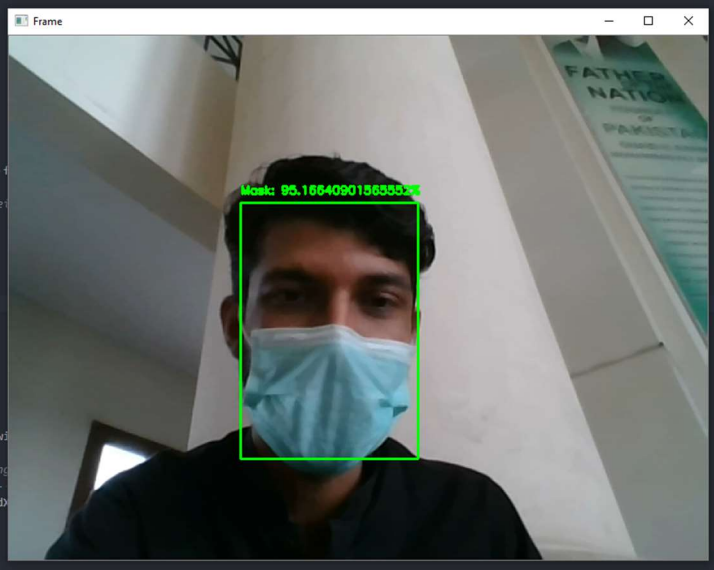
RESULT

```python
# 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()

# loop over the frames from the video stream
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=800)

    # run face dector on each frame of video
    (locs, preds) = detect_and_predict_mask(frame, f

    # loop over the detected face locations and thei
    for (box, pred) in zip(locs, preds):

        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred

        label = "Mask"
        color = (0, 255, 0)
        if mask < withoutMask:
            label = "No Mask"
            color = (0, 0, 255)


        # include the probability in the label
        label = "{}: {}%".format(label, max(mask, wi

        # display the label and bounding box rectang
        cv2.putText(frame, label, (startX, startY -
        cv2.rectangle(frame, (startX, startY), (endX

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



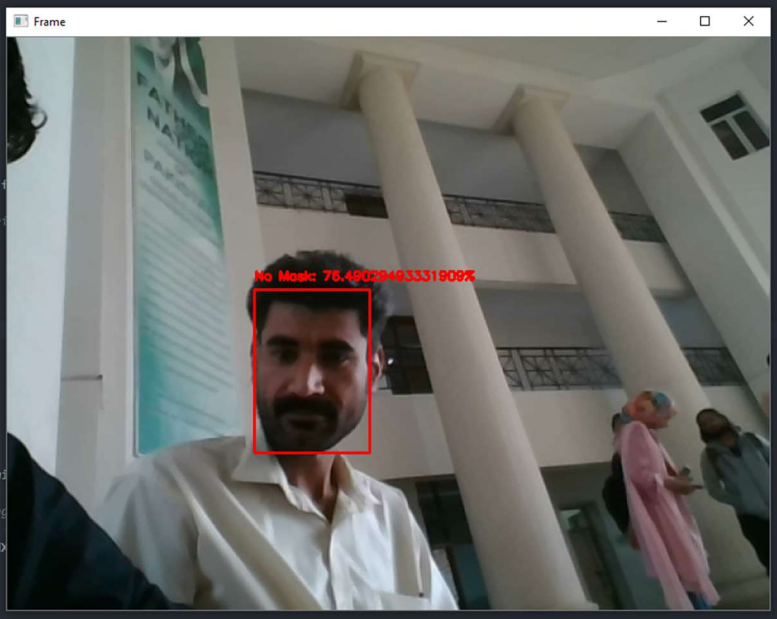
app.py

```python
# 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()

# loop over the frames from the video stream
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=800)

    # run face dector on each frame of video
    (locs, preds) = detect_and_predict_mask(frame, f

    # loop over the detected face locations and the
    for (box, pred) in zip(locs, preds):

        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred

        label = "Mask"
        color = (0, 255, 0)
        if mask < withoutMask:
            label = "No Mask"
            color = (0, 0, 255)


        # include the probability in the label
        label = "{}: {}%".format(label, max(mask, wi

        # display the label and bounding box rectang
        cv2.putText(frame, label, (startX, startY -
        cv2.rectangle(frame, (startX, startY), (endX

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

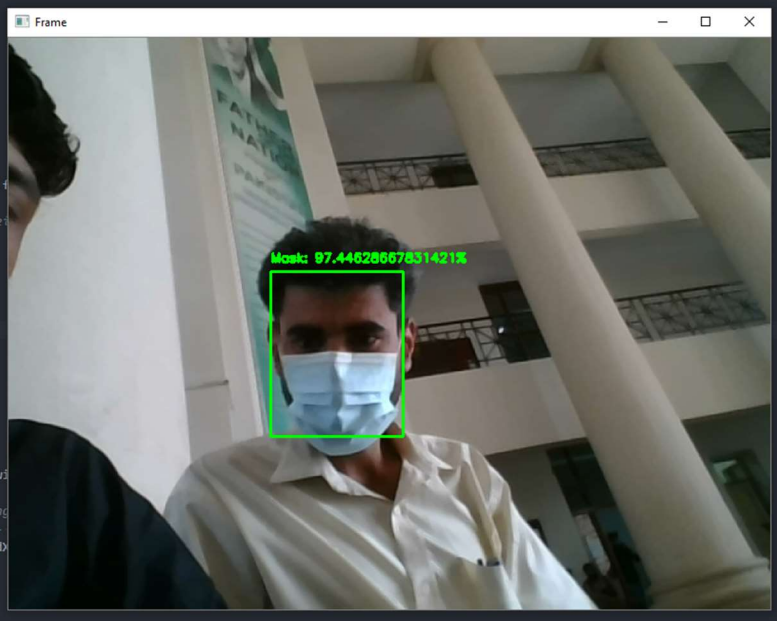


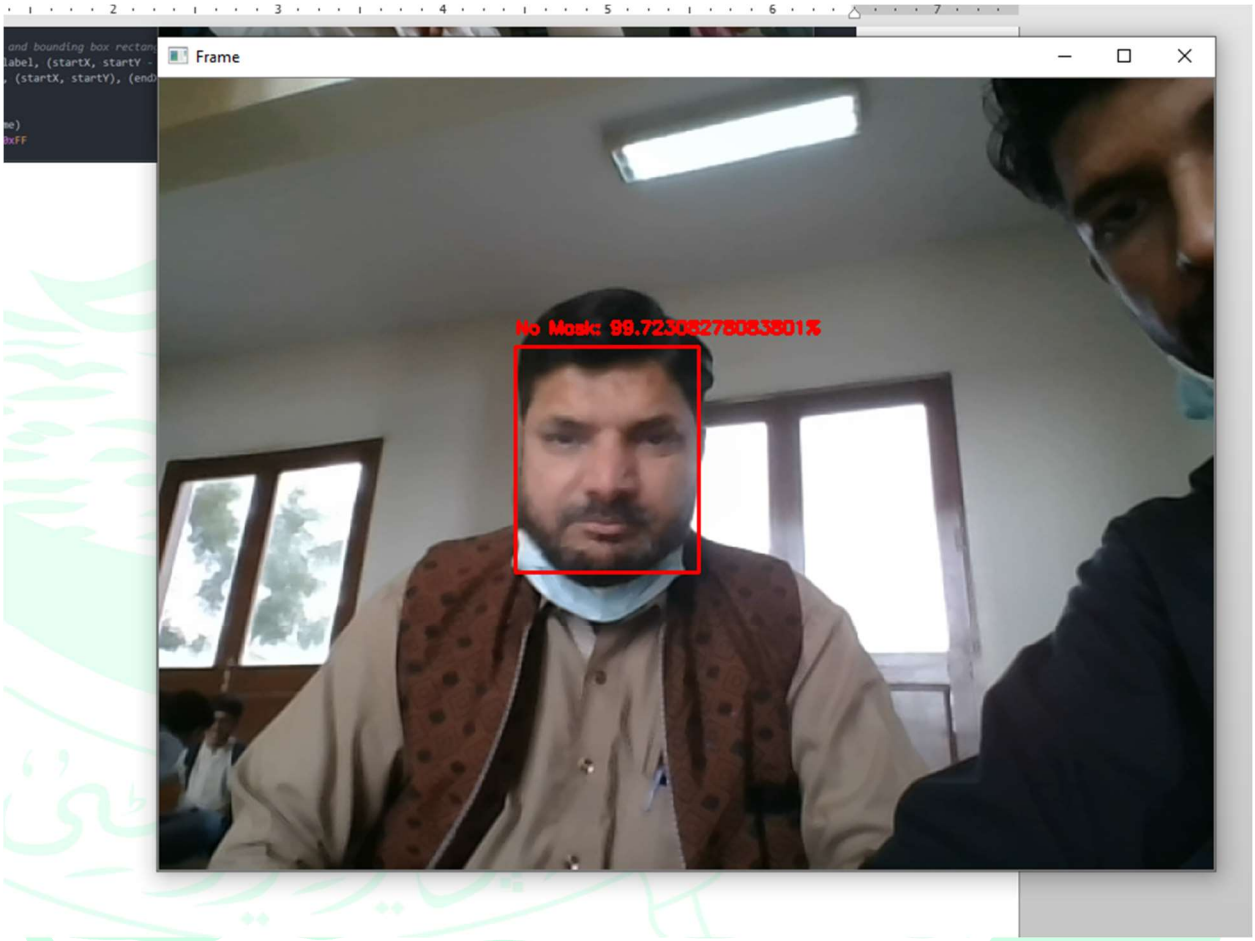
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE 1: python

```python
# 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()

# loop over the frames from the video stream
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=800)

    # run face dector on each frame of video
    (locs, preds) = detect_and_predict_mask(frame, f

    # Loop over the detected face locations and the
    for (box, pred) in zip(locs, preds):

        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred

        label = "Mask"
        color = (0, 255, 0)
        if mask < withoutMask:
            label = "No Mask"
            color = (0, 0, 255)


        # include the probability in the label
        label = "{}: {}%".format(label, max(mask, wi

        # display the label and bounding box rectang
        cv2.putText(frame, label, (startX, startY -
        cv2.rectangle(frame, (startX, startY), (endX

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

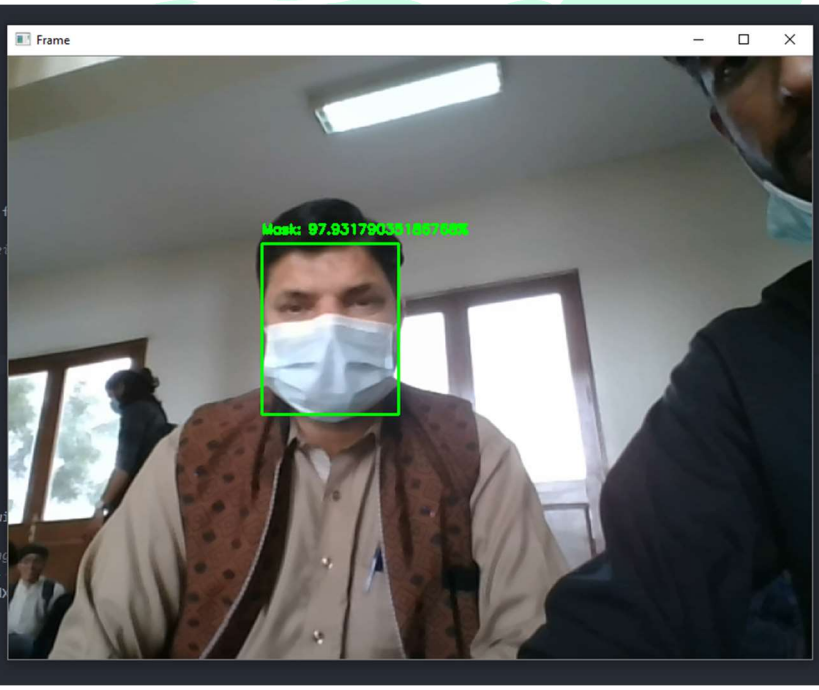



```python
# load the face mask detector model from disk
maskNet = load_model("mask_detector.model")

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

# loop over the frames from the video stream
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=800)

    # run face dector on each frame of video
    (locs, preds) = detect_and_predict_mask(frame, f

    # Loop over the detected face locations and the
    for (box, pred) in zip(locs, preds):

        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred

        label = "Mask"
        color = (0, 255, 0)
        if mask < withoutMask:
            label = "No Mask"
            color = (0, 0, 255)


        # include the probability in the label
        label = "{}: {}%".format(label, max(mask, wi

        # display the label and bounding box rectang
        cv2.putText(frame, label, (startX, startY -
        cv2.rectangle(frame, (startX, startY), (endX

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

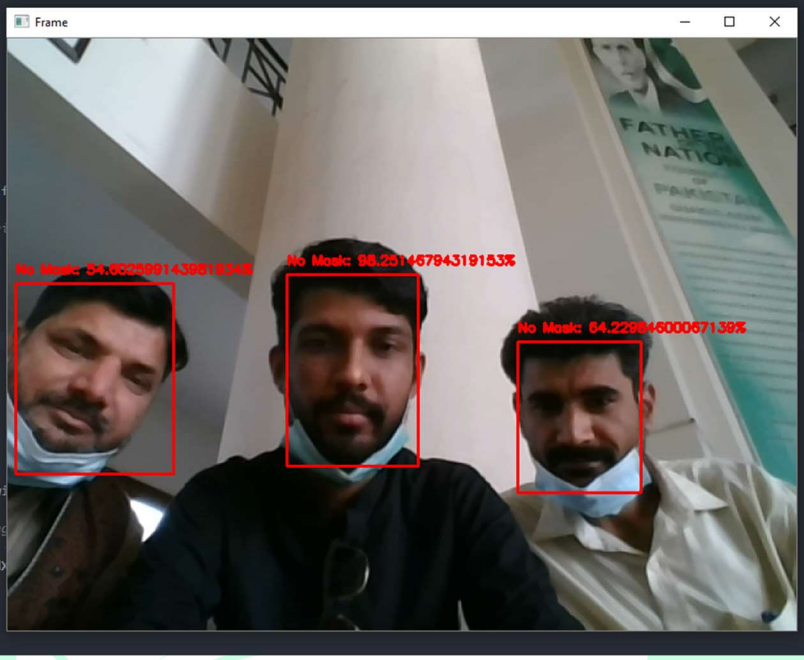




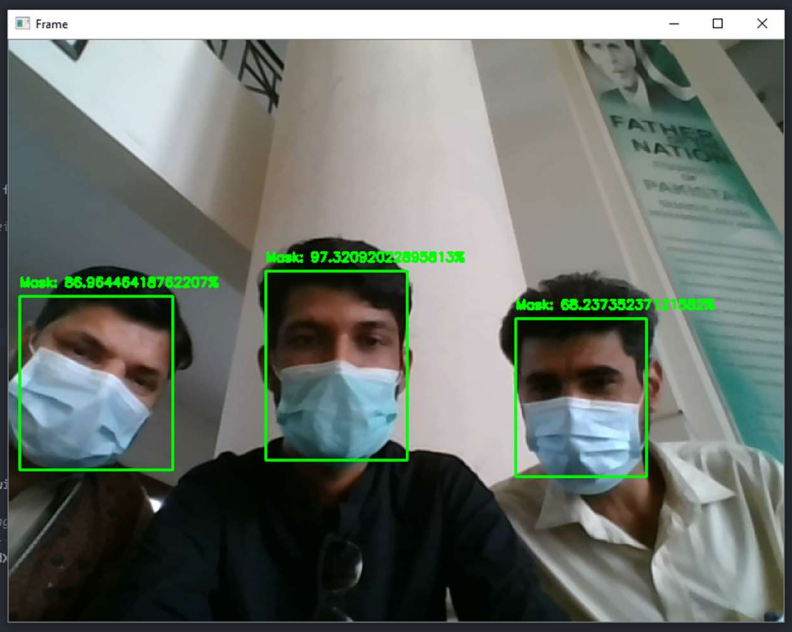

```
# initialize the video stream
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()

# loop over the frames from the video stream
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=800)

    # run face dector on each frame of video
    (locs, preds) = detect_and_predict_mask(frame,

    # loop over the detected face locations and the
    for (box, pred) in zip(locs, preds):

        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred

        label = "Mask"
        color = (0, 255, 0)
        if mask < withoutMask:
            label = "No Mask"
            color = (0, 0, 255)


        # include the probability in the label
        label = "{}: {}%".format(label, max(mask, w

        # display the label and bounding box rectang
        cv2.putText(frame, label, (startX, startY -
        cv2.rectangle(frame, (startX, startY), (endX

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



```
maskNet = load_model("mask_detector.model")

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

# loop over the frames from the video stream
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=800)

    # run face dector on each frame of video
    (locs, preds) = detect_and_predict_mask(frame, f

    # loop over the detected face locations and the
    for (box, pred) in zip(locs, preds):

        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred

        label = "Mask"
        color = (0, 255, 0)
        if mask < withoutMask:
            label = "No Mask"
            color = (0, 0, 255)


        # include the probability in the label
        label = "{}: {}%".format(label, max(mask, wi

        # display the label and bounding box rectang
        cv2.putText(frame, label, (startX, startY -
        cv2.rectangle(frame, (startX, startY), (endX

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



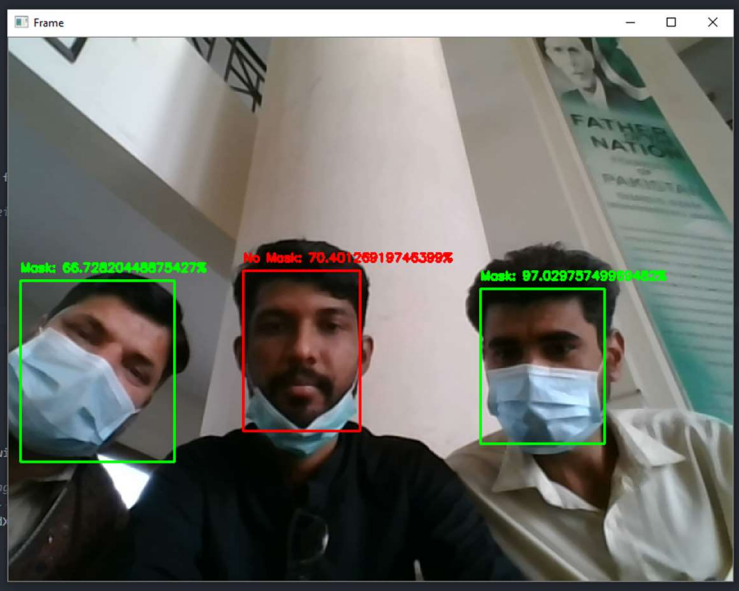

```
# load the face mask detector model from disk
maskNet = load_model("mask_detector.model")

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

# loop over the frames from the video stream
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=800)

    # run face dector on each frame of video
    (locs, preds) = detect_and_predict_mask(frame, f

    # loop over the detected face locations and the
    for (box, pred) in zip(locs, preds):

        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred

        label = "Mask"
        color = (0, 255, 0)
        if mask < withoutMask:
            label = "No Mask"
            color = (0, 0, 255)


        # include the probability in the label
        label = "{}: {}%".format(label, max(mask, wi

        # display the label and bounding box rectang
        cv2.putText(frame, label, (startX, startY -
        cv2.rectangle(frame, (startX, startY), (endX

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



TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE 1: python

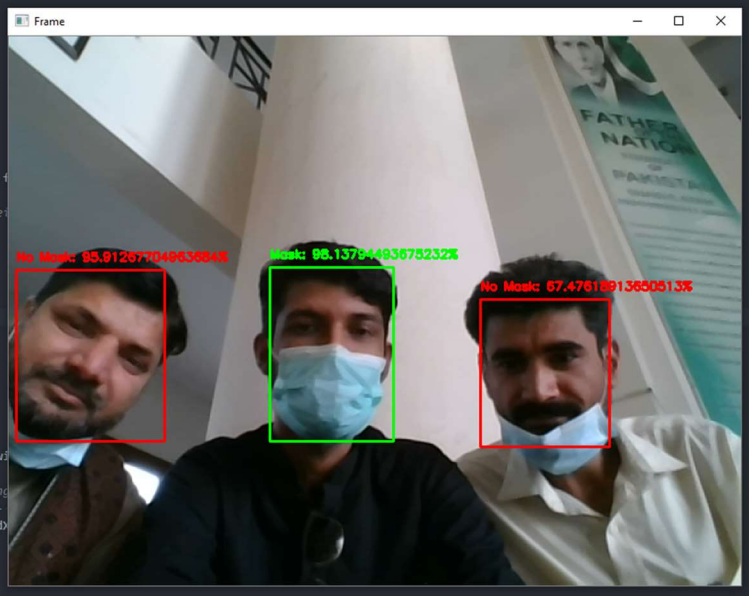
app.py

```
# 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()

# loop over the frames from the video stream
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=800)

    # run face dector on each frame of video
    (locs, preds) = detect_and_predict_mask(frame,

    # loop over the detected face locations and the
    for (box, pred) in zip(locs, preds):

        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred

        label = "Mask"
        color = (0, 255, 0)
        if mask < withoutMask:
            label = "No Mask"
            color = (0, 0, 255)


        # include the probability in the label
        label = "{}: {}%".format(label, max(mask, wi

        # display the label and bounding box rectang
        cv2.putText(frame, label, (startX, startY -
        cv2.rectangle(frame, (startX, startY), (endX

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



TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE 1: python

```
maskNet = load_model("mask_detector.model")

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

# loop over the frames from the video stream
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=800)

    # run face dector on each frame of video
    (locs, preds) = detect_and_predict_mask(frame,

    # loop over the detected face locations and the
    for (box, pred) in zip(locs, preds):

        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred

        label = "Mask"
        color = (0, 255, 0)
        if mask < withoutMask:
            label = "No Mask"
            color = (0, 0, 255)


        # include the probability in the label
        label = "{}: {}%".format(label, max(mask, wi

        # display the label and bounding box rectang
        cv2.putText(frame, label, (startX, startY -
        cv2.rectangle(frame, (startX, startY), (endX

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



END