FACE MASK DETECTION

- AHMED ALAMRI TC0020
- NAIF GANADILY QA0272
- MAAN ALASALI RA0193

EE 429 Independent Design Supervisor: Dr. Amr Yousef

The problem:

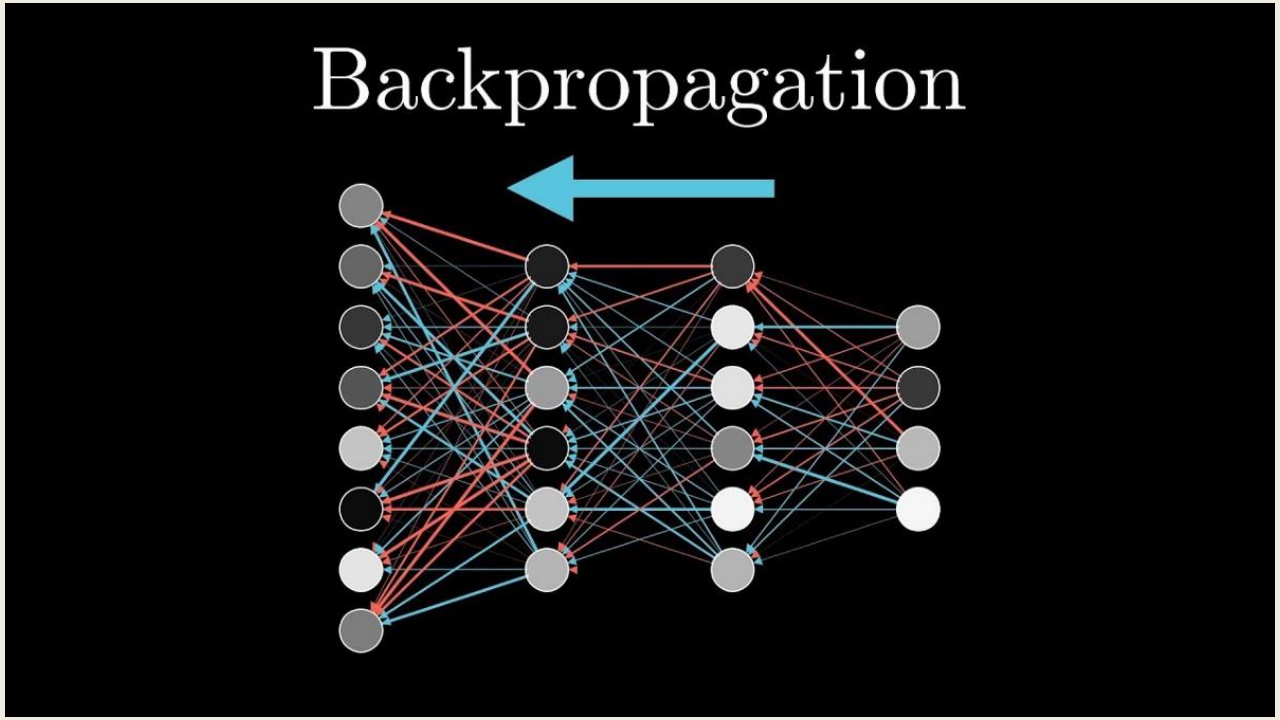
- Pandemic COVID-19, WHO has made wearing masks compulsory.
- Some people in public places do not take precautions.

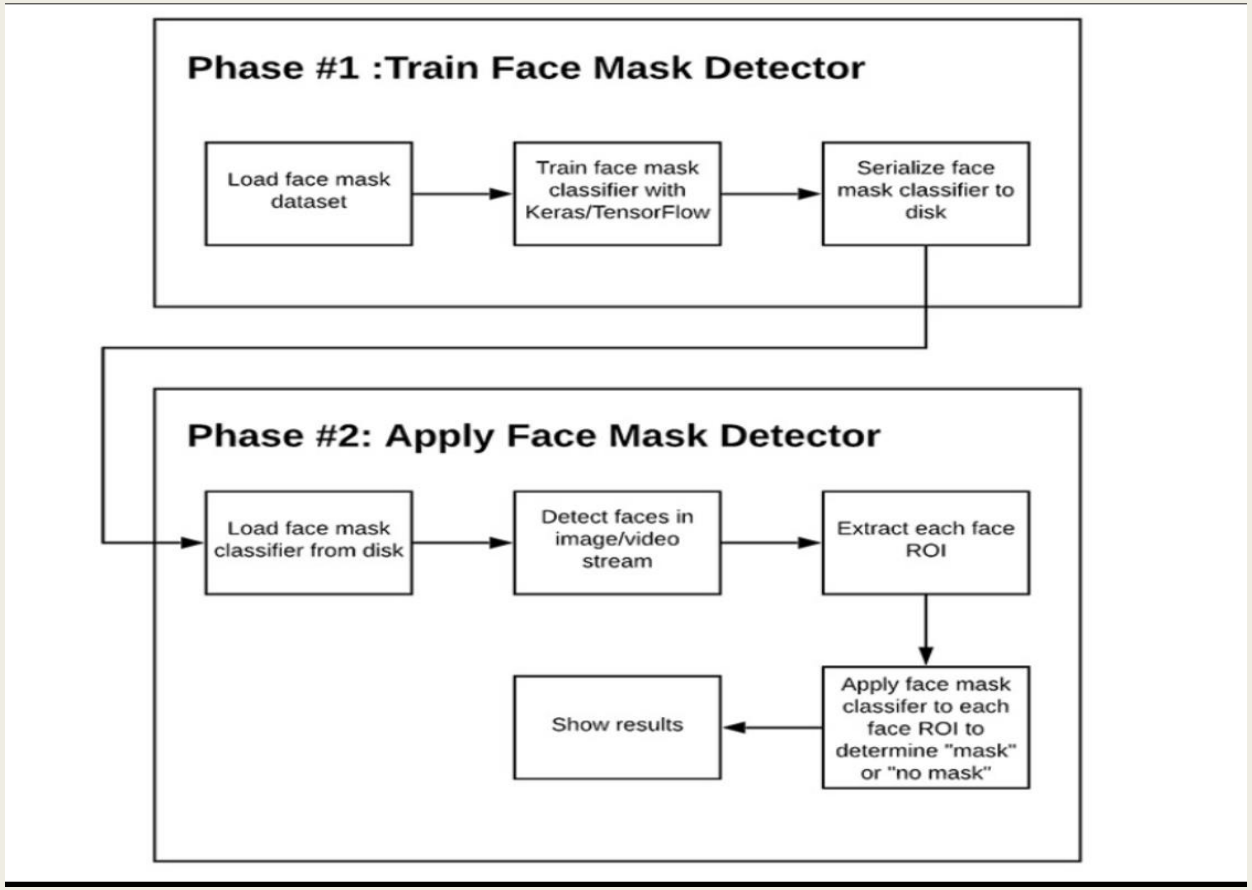
The Potential of the Face Mask:

- The COVID-19 mask detector we're building here today could potentially be used to help ensure your safety and the safety of others.
- Detecting whether person is wearing a mask or not.

Backpropagation (Algorithm)

- In machine learning, backpropagation is a widely used algorithm for training feedforward neural networks. Generalizations of backpropagation exist for other artificial neural networks, and for functions generally.




Phase #1 :Train Face Mask Detector
Load face mask dataset
Train face mask classifier with Keras/TensorFlow
Serialize face mask classifier to disk
Phase #2: Apply Face Mask Detector
Load face mask classifier from disk
Detect faces in image/video stream
Extract each face ROI
Apply face mask classifer to each face ROI to determine "mask" or "no mask"
Show results

- Data augmentation
- Loading the MobilNetV2 classifier (we will fine-tune this model with pre-trained **ImageNet** weights)
- Building a new fully-connected (FC) head
- Pre-processing
- Loading image data

Step one: Install The Dependencies.

- Required libraries:
  - *tensorflow>=1.15.2*
  - *keras==2.3.1*
  - *imultils==0.5.3*
  - *numpy==1.18.2*
  - *opencv-python==4.2.0.**
  - *matplotib==3.2.1*
  - *argparse==1.1*
  - *scipy==1.4.1*

STEP 2: Let’s A take a look at dataset

- Inside the dataset folder there will be two folders:

1- with_mask
  - *With mask contain all images of people wearing masks on.*

2- without_mask

STEP 3: Data Preprocessing

- In this part we will convert all the images from both folders with-without mask.
- These are the imports:

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
```

STEP 3: Data Preprocessing count.

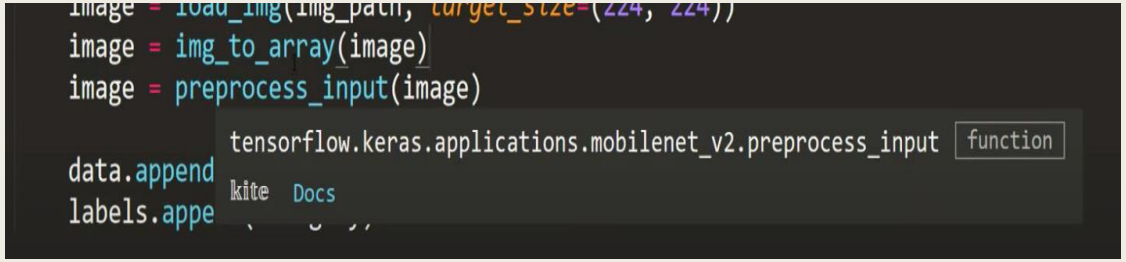
- There is directory and categories:
- Inside directory I mentioned where is my data folders are actually present.
- Inside the category: I'm having 2 values with-without mask
- And I just mentioned the print("[INFO] loading images. . .")
- And there are two lists I operated two lists
- 1- data= [ ] , inside Data I'll attach all the images inside this data list
- 2- labels=[ ] , inside the labels list I'll append all those corresponding with-without masks basically they contain the label of those images are with & without mask.
- Go to —> path = os.path.join(directory , category)
- Once data is done I get the path of this particular with mask and without mask.

```
for category in CATEGORIES:
    path = os.path.join(DIRECTORY, category)
```

word listdir:

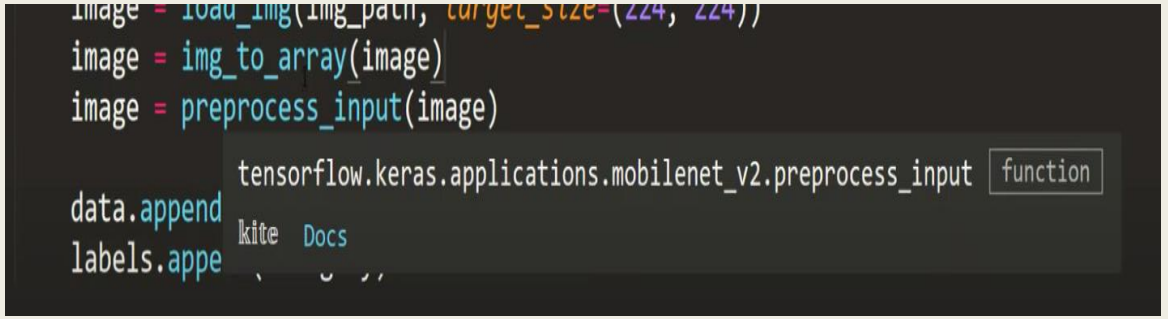
- It’s kind of images hold inside that particular directory so with mask there is around thousand and nine images which will listdir by this particular so. listdir method.After I’ll be joining the paths to this particular with mask corresponding image.
- Onc I’m done I’ll call the function called”load_img”
- So, this “load_img is coming from the keras processing
- Inside keras there a function called load img

```
for img in os.listdir(path):
    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)
```



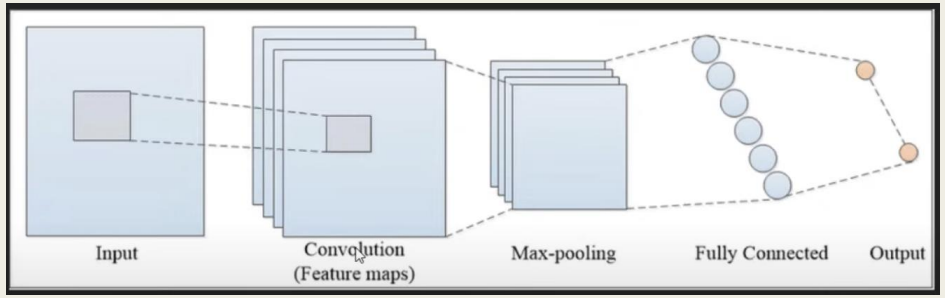
STEP 3: Data Preprocessing count.

- Once I load the image I just save it and now convert to an array using img_to_array function.
- This image to array coming from the keras
- Once I convert the function to array and then I'll use
- image = preprocess_input(image)
- Once we convert our pictures successfully then open our image in [data.append(image)] once that is done need to open the corresponding labels inside this labels
- With and without masks.



STEP4: Training

- We are going to follow the “CONVOLUTION NUERAL NETWORK”
- Mobil nets are very faster in process when compared to CONVOLUTION NUERAL NETWORKS AND MOBIL NETS uses lesser parameters.



STEP4: Training count.

- giving the initial rate to INIT_LR = 1e-4
- always make sure your learning rate is less so your when your learning that is less your loss will be calculated properly which means you will get the better accuracy.
- my learning to be 0.0001 and given 20 epochs my back size size 32
- these values to get a better result.

```
# initialize the initial learning rate, number of epochs to train for,
# and batch size
INIT_LR = 1e-4
EPOCHS = 20
BS = 32

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

STEP4: Training count.

- Image data generator kind accommodations so it creates many images with a single image by adding various properties like flipping rotating the image the image and many other properties to the can create more data set.
- rotation angle is 20
- zoom range is 0.15

```
# 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)))
```

STEP4: Training count.

- what we are doing as we are saying the model that we generated now and we are saying it in the h5 format and finally what we do is we are going to plot accuracy and a matrix by using matplotlib.

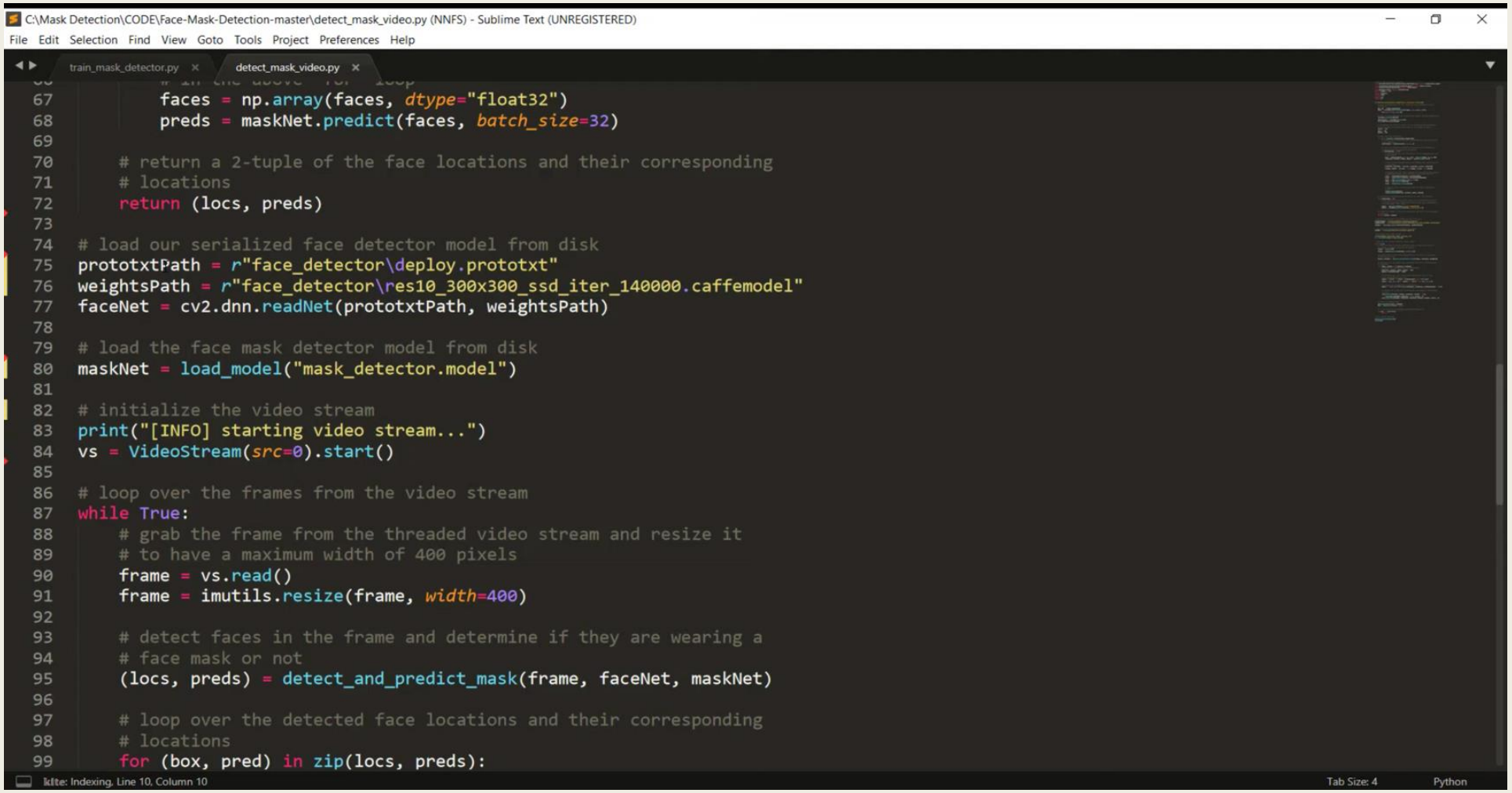
```
# serialize the model to disk
print("[INFO] saving mask detector model...")
model.save("mask_detector.model", save_format="h5")

# 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")
```

STEP 5: Run View The Accuracy:

- According the plot the model is pretty good at the moment.

STEP 6: Apply The Model In Camera



STEP 6: Apply The Model In Camera count.

IN ORDER TO USE THEM WE HAVE TO USE A METHOD CALLED READ NET WHICH IS UNDER CV2 AND THE MODEL CALLED DNN
DNN STANDS FOR = DEEP NEURAL NETWORKS
CV2 RECENTLY DEVELOP DNN IN MANY USING METHODS.

```
77  faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
```

WE CAN USE THIS PARTICULAR "FACENET" TO DETECT FACES.
WE ARE LOADING OUR MODEL USING "LOAD MODEL"
THE MODEL WE USE FOR FACE DETECTION
THIS IS THE MODEL THAT WE ACTUALLY DEVELOP.

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

STEP 6: Apply The Model In Camera count.

AND NOW WE HAVE THE T]MODELS FOR FACE DETECTION AND MASK DETECTION.
NEXT THING WE NEED TO DO IS TO LOAD THE CAMERA
IN ORDER TO DO THAT WE ARE USING VIDEO STREAM
INSIDE THIS VIDEO STREAM FUNCTION THERE IS SOMETHING CALLED "SRC=0" IF YOU HAVE MORE THAN A CAMER THE ZERO
WILL CHANGE TO HOW MANY CAMERAS ARE USED.
AND START ACTUALLY LOADS THE CAMERA.

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

AND THEN WE WILL GO TO THE "WHILE TRUE"
HERE WE ARE READING THE FRAME
SO EVERY FRAME IS NOTHING BUT AN IMAGE.

```
# 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)
```

STEP 6: Apply The Model In Camera count.

AND NOW HE HAVE " FACE NET , MASK NET , FRAME "
FACE NET: FOR FACE DETECTION
MASK NET: FOR MASK DETECTION
FRAME: FOR THE VIDEO LOADING FROM THE CAMERA
AFTER WE HAVE ALL THIS WE WILL DEFINE A FUNCTION
"CALLED DEFINE AND PREDICT MASK"

```
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)
```

WITH THESE THREE ARGUMENTS AND WE JUST DO NORMAL MANIPULATIONS HERE AND RETURN BACK TO RETURN

```
    preds = maskNet.predict(faces, batch_size=32)

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

STEP 6: Apply The Model In Camera count.

LOCS = IS NOTHING BUT X AND Y COORDINATES OF PARTICULAR RECTANGLE THAT IS GOING TO SURROUNDING THE FACE.
PREDS = IS THE PREDICTIONS OF THE PERSON WEARING THE MASK OR NOT AND IT CAN BE AS PERCENTAGES.

NOW WE GET THE X,Y AS FOR EXAMPLE
(T X1,Y1 , X2,Y2) WITH THESE POINTS WE WILL DRAW WE WILL DRAW THE RECTANGLE
AND WITH THE PREDICTIONS

```
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)
```

LABEL = “MASK” IS MASK > WITHOUTMASK ELSE “NO MASK”
AND FOR COLOR WE WILL CHOOSE B , G AND R
0= IS THE LEAST VALUE 255= IS HIGHEST VALUE
AS WE SAID B= 0 G= 255 R= 0
THESE COLORS LEAD TO GREEN COLOR
SO GREEN COLOR IF THERE IS MASK

AND IF THERE IS NO MASK WE WILL PUT (0,0,255)
RE IS THE MAXIMUM VALUE HERE I]AND THIS WILL LEADS TO RED.

STEP 6: Apply The Model In Camera count.

WE ARE JUST DISPLAYING THE PERCENTAGE OF THE PREDICTION WE HAVE HERE

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

- THE TEXT HERE.
- DRAWING THE RECTANGLE

```
cv2.putText(frame, label, (startX, startY - 10),
    cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
```

- THE FRAME IS THE SEQUENCE OF PICTURES THAT IS GOING TO FLOW THROUGH.
- FINALLY WE ARE BREAKING THE WHILE LOOP

```
# 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
```

WE FINALLY DESTROY ALL THE WINDOWS AND STOP THE VIDEOS

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

STEP7: Let’s See The Result