



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of Aditya Kumar Jha (38110013), Arkit Mehta (38110047) who carried out the project entitled "Face mask Detection using Convolutional Neural Network" under my/our supervision from October 2021 to April 2022.

Internal Guide

External Guide

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

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

I convey my thanks to Dr. Shashikala, Dean, school of computing and Dr. L. Lakshmanan, Head of the department, Dept. of Computer Science for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my project guide Mr. Aroul Cannesane for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

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

Face Mask Detection using Convolutional Neural Network

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Engineering degree in Computer Science Engineering

By

Aditya Kumar Jha (38110013)

Arkit Jitendra Mehta (38110047)



DEPARTMENT OF COMPUTER SCIENCE ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(Deemed to be university)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600119

MAY - 2022

Contents

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	vi
1	INTRODUCTIONS	1
	1.1 PROJECT AIMS AND OBJECTIVES	1
	1.2 BACKGROUND OF PROJECT	1
	1.3 OPERATION ENVIROMENT	2
2	SYSTEM ANALYSIS	3
	2.1 SOFTWARE REQUIIREMENT SPECIFICATION	3
	2.2 SOFTWARE-HARDWARE REQUIREMENT	4
	2.3 SOFTWARE TOOL USED	5
3	SYSTEM TESTING	7
	3.1 TABLE DESIGN	7
	3.2 DATA FLOW DIAGRAM	7
4	SYSTEM IMPLEMENTATIONS	8
	4.1 MODULE DESCRIPTION	8
	4.2 DATASET	9
	4.3 FUTURE SCOPE	9
	REFERENCES	10
	APPENDIX	
	A. SOURCE CODE	11
	B. SCREEN SHOTS	18

CHAPTER 1

INTRODUCTION

This chapter gives an overview about the aim, objectives, background and operation environment of the system.

1.1 PROJECT AIMS and OBJECTIVES

The project aims and objectives that will be achieved after completion of this project are discussed in this subchapter. The aims and objectives are as follows:

- a. Image Cropping
- b. Image Classification
- c. Creation of Unbiased facemask dataset
- d. Requires less memory

1.2 BACKGROUND OF THE PROJECT

Face mask detection System using Convolutional Neural Network (CNN) is proposed system to classify face mask detection using Image classification. It refers to detect whether a person is wearing a mask or not. In fact, the problem is reverse engineering of face detection where the face is detected using machine learning algorithm for purpose of security, authentication and surveillance.

Face detection is a key area in the field of Computer vision and pattern recognition. The primary research on face detection was done in 2001 using the design of handcraft feature and application of traditional machine learning algorithms to train effective classifiers for detection and recognition.

In recent years, face detection methods based on deep convolutional neural networks (CNN) have widely developed to improve detection performance.

1.3 OPERATION ENVIRONMENT

PROCESSOR	INTEL CORE PROCESSOR OR BETTER PERFORMANCE
OPERATING SYSTEM	WINDOWS VISTA, WINDOWS 7, UBUNTU
MEMORY	1GB RAM OR MORE
HARD DISK SPACE	MINIMUM 3 GB FOR DATABASE USAGE FOR FUTURE

Table:1

CHAPTER 2

SYSTEM ANALYSIS

In this chapter, we will discuss and analyse about the developing process of facemask detection System including software requirement specification (SRS) and comparison between traditional and existing system. The functional and non-functional requirements are included in SRS part to provide complete description and overview of system requirement before the developing process is carried out.

2.1 SOFTWARE REQUIREMENT SPECIFICATION

PRODUCT DESCRIPTION:

The task of recognizing the mask over the face in the public area can be achieved by deploying an effective object recognition algorithm through surveillance devices. We review the recent development in region proposal techniques using detectors, general technique for improving and pre-trained models based on these techniques.

2.2 SOFTWARE AND HARDWARE REQUIREMENTS

This section describes the software and hardware requirements of the system.

2.2.1 SOFTWARE REQUIREMENTS

- ◆ Operating system- Windows 7 is used as the operating system as it is stable and supports more features and is more user friendly
- ◆ Development tools and Programming language - Python is used to write the whole source code with PyCharm being the IDE.

2.2.2 HARDWARE REQUIREMENTS

- ◆ Intel core i5 8th generation is used as a processor because it is fast than other processors and provide reliable and stable and we can run our pc for long-time. By using this processor, we can keep on developing our project without any worries.
- ◆ Ram 1 GB is used as it will provide fast reading and writing capabilities and will in turn support in processing.

2.3 SOFTWARE TOOLS USED

Python is the interpreted high-level general purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

PyCharm is an integrated development (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains (formerly known as IntelliJ). It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control system (VCSes), and supports web development with Django as well as data science with Anaconda.

Machine Learning is a branch of computing which originated from the study of the recognition of forms and the theory of computerized learning in the artificial. Rather than following rigid system instructions, these processes use model generation from model inputs to create data-based forecasts or selections. Arthur Samuel coined the term Machine Learning, in the sense of solving a test machine with a machine.

Machine learning (ML) involves two types of activities:

- i. Supervised machine learning
- ii. Unsupervised machine learning

Convolutional neural network (CNN) is a type of deep learning algorithm that takes an input picture and assigns importance to different picture characteristic as weight and are able to differentiate between

them. Strength of a convolutional neural network comes from a particular kind of layer called the convolutional layer.

CNN contains many convolutional layers assembled on top of each other, each one competent of recognizing more sophisticated shapes. The construction of a convolutional neural network is a multi-layered feed-forward neural network. CNN are very satisfactory at picking up on design in the input image, such as lines, gradients, circles, or even eyes and faces.

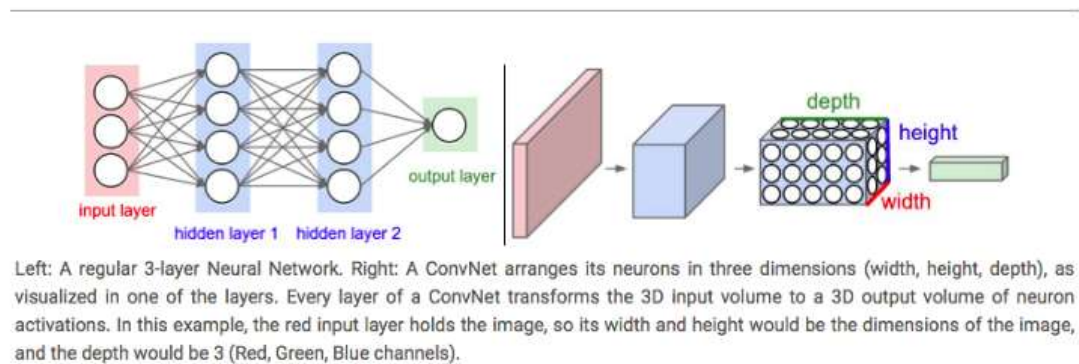


Fig 1: CNN Layers

CHAPTER 3

SYSTEM TESTING

3.1 TABLE DESIGN

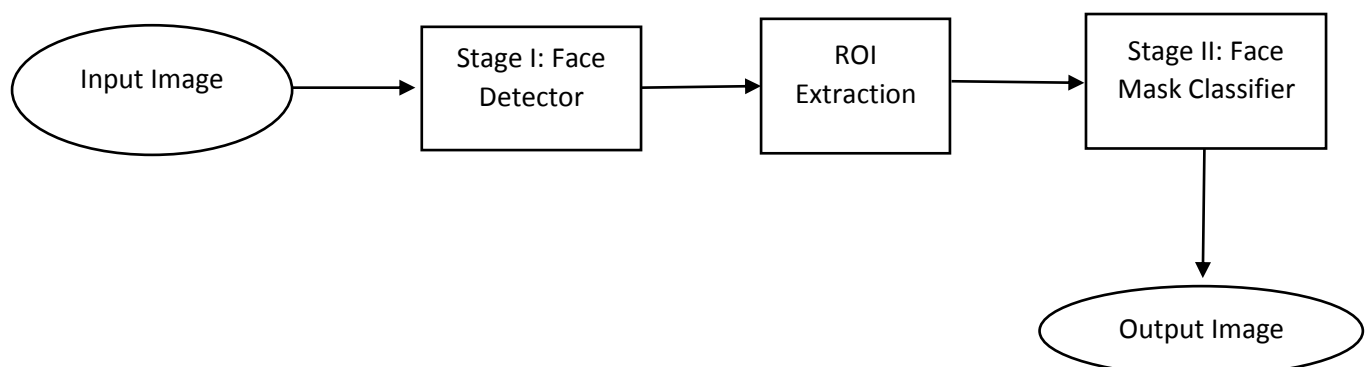
Testing is an important portion that is to be carried out while developing any program as it helps to find the error if there is any error and helps to debug the program. Testing helps us to get familiar with the type of error and handle the errors with the necessary formatting in the program.

Test-1

Action	To check whether it can handle exceptions or not
Expected output	It should able to handle any sum of input of user
Actual output	The program was able to handle even the wrong input of user
Test result	The program runs in its flow until user closes it

Table 2

3.2 DATA FLOW DIAGRAM



CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 MODULE DESCRIPTION

The project “Facemask Detection system” has been made using Python, Machine Learning algorithms and codes were written in PyCharm software. Public use of Facemask has been common in recent times since the beginning of new coronavirus disease outbreak. We now know from recent studies that a significant portion of individuals with Covid-19 symptoms (pre-symptomatic) can transmit the virus to others before showing symptoms.

So, it is inevitable for people of an overpopulated country like India to wear masks and let the work go on. Nobody can keep an eye on every people coming in the work space is wearing mask or not. So, the need of face mask detection system arose. The model in this paper uses Convolutional Neural Network. It is a deep neural network model used for analysing any visual imagery.

CNN takes image data as input, captures all the data, and send to layers of neurons. It has a fully connected layer, which processes the final output that represents the prediction about the image. As a whole, the architecture contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers.

4.2 Dataset

Dataset is the major part for building the CNN model. More the number of dataset images is, more accurate the model will be.

The dataset used in this project is a Face dataset (with/without mask dataset) from Kaggle.com. The dataset consists of images divided into two masks classes:

- Images taken with a mask
- Images taken without a mask



4.3 Future scope

Across the globe many countries have made it mandatory to wear masks at all public places. You have to cover your face in schools, supermarkets, transports, offices, stores and all other public places. For this reason, we plan to update the mask recognition system and announce it as an open-source project.

This system can be implemented and the result can be seen on digital screens. This app can be used with any digital camera for identifying people who aren't wearing masks. The proposed technique efficiently handles occlusions in dense situations.

Finally, the work opens interesting future directions for researcher. Firstly, the proposed technique can be integrated into any high-resolution surveillance devices and not be limited to mask detection only. Secondly, the model can be extended to detect facial landmarks with a facemask for biometric purpose.

References

- **CNN:** [Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network | up Grad blog](#)
- **PYTHON:** [Learn Python Programming Masterclass | Udemy](#)
- [Face Mask Detection using CNN \(98% Accuracy\) | Kaggle](#)

- [Face Mask Detection using CNN | IEEE Conference Publication | IEEE Xplore](#)
- Negi, A., Chauhan, Kumar, K., Rajput, R.: Category facial mask acquisition and modelling of a doctor with keras surgery. 2020 IEEE International Conference on Recent Development and Innovation in Engineering (ICRAIE), pages 1–6 (2020). IEEE
- Ejaz, M.S., Islam, M.R., Sifatullah, M., Sarkar, A.: Implementation of a key component in the identification of covered and uncovered faces. In: Springer Nature 2021 LATEX Template Topic 9 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), pp. 1–5 IEEE (2019)
- Qin, B., Li, D .: Identifies the condition of wearing a face mask using image super resolution and a separation network to prevent covid-19.
- Lin, K., Zhao, H., Lv, J., Zhan, J., Liu, X., Chen, R., Li, C., Huang, Z .: Face detection and subdivision by standard roads over a union based on the mask r-CNN. At: International Conference on Brain-inspired Memory Programs, pages 106–116 (2019). Springer

Source Code

- Importing the libraries

```
import
cv2,os
    from tensorflow.keras.models import load_model
    import NumPy as np
```

- Determining the accuracy of CNN

```
model = load_model('model-007.model') #comment this line if running resnet model
#model = load_model('model-562.model') #uncomment this line if running resnet
model
```

```
folder_path=r"data\mask_test"
img_names=os.listdir(folder_path)
count=0
correct=0
```

```
for img_name in img_names:
    count=count+1
    img_path=os.path.join(folder_path,img_name)
    img=cv2.imread(img_path)
    img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    img=img/255.0
    img=np.reshape(img,(1,100,100,1))
    #img=np.reshape(img,(1,100,100,3))
    result=model.predict(img)
    if (result[0][0] > result[0][1]):
        correct=correct+1
```

```
folder_path=r"data\unmask_test"
img_names=os.listdir(folder_path)
```

```
for img_name in img_names:
    count=count+1
    img_path=os.path.join(folder_path,img_name)
    img=cv2.imread(img_path)
    img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    img=img/255.0
    img=np.reshape(img,(1,100,100,1))
```

```

=np.reshape(img,(1,100,100,3))
result=model.predict(img)
if (result [0][0] < result [0][1]):
    correct=correct+1
print(correct/count)

```

- Pre-processing the image

```

import
os

import mtcnn
import cv2
import matplotlib.pyplot as plt
from PIL import Image
import PIL.ImageOps

data_path='data'
folder_path=os.path.join(data_path,"unmask")
img_names=os.listdir(folder_path)
detector = mtcnn.MTCNN()

img_num = 0
for img_name in img_names:
    pixels = plt.imread(os.path.join(folder_path, img_name))
    img = cv2.imread(os.path.join(folder_path, img_name))
    try:
        faces = detector.detect_faces(pixels)
        print(img_num, img_name, len(faces))
        count = 0
        for face in faces:
            x, y, width, height = face['box']
            img = img[y:y+height, x:x+width]
            if img.size == 0:
                continue
            if len(img.shape) != 3:
                continue
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            gray = cv2.resize(gray, (100,100))
            new_image_name = os.path.join(r'data\unmask_train',
img_name + str(count) + '.jpg')
            cv2.imwrite(new_image_name, gray)
            count = count + 1
        img_num = img_num + 1

```

```
except Exception as e:  
    print('Exception:',e)
```

- Training mask detector with CNN

```
# USAGE  
# python train_mask_detector.py --dataset dataset  
  
# 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 argparse  
import os  
  
# construct the argument parser and parse the arguments  
ap = argparse.ArgumentParser()  
ap.add_argument("-d", "--dataset", required=True,  
                help="path to input dataset")  
ap.add_argument("-p", "--plot", type=str, default="plot.png",  
                help="path to output loss/accuracy plot")  
ap.add_argument("-m", "--model", type=str,  
                default="mask_detector.model",  
                help="path to output face mask detector model")  
args = vars(ap.parse_args())  
  
# initialize the initial learning rate, number of epochs to train for,  
# and batch size  
INIT_LR = 1e-4  
EPOCHS = 20
```

```

BS = 32

# 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...")
imagePaths = list(paths.list_images(args["dataset"]))
data = []
labels = []

# loop over the image paths
for imagePath in imagePaths:
    # extract the class label from the filename
    label = imagePath.split(os.path.sep)[-2]

    # load the input image (224x224) and preprocess it
    image = load_img(imagePath, target_size=(224, 224))
    image = img_to_array(image)
    image = preprocess_input(image)

    # update the data and labels lists, respectively
    data.append(image)
    labels.append(label)

# convert the data and labels to NumPy arrays
data = np.array(data, dtype="float32")
labels = np.array(labels)

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

# partition the data into training and testing splits using 75% of
# the data for training and the remaining 25% for testing
(trainX, testX, trainY, testY) = train_test_split(data, labels,
    test_size=0.20, stratify=labels, random_state=42)

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

# load the MobileNetV2 network, ensuring the head FC layer sets are

```

```

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

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

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

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

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

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

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

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

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

```

```

# serialize the model to disk
print("[INFO] saving mask detector model...")
model.save(args["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(args["plot"])

```

- Detecting mask image

```

# USAGE
# python detect_mask_image.py --image examples/example_01.png

# import the necessary packages
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
import numpy as np
import argparse
import cv2
import os

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
    help="path to input image")
ap.add_argument("-f", "--face", type=str,
    default="face_detector",
    help="path to face detector model directory")
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="minimum probability to filter weak detections")
args = vars(ap.parse_args())

# load our serialized face detector model from disk

```

```

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"])
net = cv2.dnn.readNetFromCaffe(prototxtPath, weightsPath)

# load the face mask detector model from disk
print("[INFO] loading face mask detector model...")
model = load_model(args["model"])

# load the input image from disk, clone it, and grab the image spatial
# dimensions
image = cv2.imread(args["image"])
orig = image.copy()
(h, w) = image.shape[:2]

# construct a blob from the image
blob = cv2.dnn.blobFromImage(image, 1.0, (300, 300),
    (104.0, 177.0, 123.0))

# pass the blob through the network and obtain the face detections
print("[INFO] computing face detections...")
net.setInput(blob)
detections = net.forward()

# loop over the detections
for i in range(0, detections.shape[2]):
    # extract the confidence (i.e., probability) associated with
    # the detection
    confidence = detections[0, 0, i, 2]
    # filter out weak detections by ensuring the confidence is
    # greater than the minimum confidence
    if confidence > args["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
        face = image[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)

# pass the face through the model to determine if the face
# has a mask or not
(mask, withoutMask) = model.predict(face)[0]

# 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)

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

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

# show the output image
cv2.imshow("Output", image)
cv2.waitKey(0)

```

Output screenshot

