

# **Face Mask Detection**

**Using OpenCV & MobileNet**

TE mini-project report submitted in partial fulfilment of the requirements of the degree of Information Technology,

By

Aashish Shetye Roll 46 [TEIT2]  
Amaan Siddiqui Roll 47 [TEIT2]  
Gaurav Wankhede Roll 58 [TEIT2]  
Manish Yadav Roll 60 [TEIT2]

Under the guidance of:  
Prof. Nileema Pathak



**Department of Information Technology,  
Atharva College of Engineering, Malad  
(W)**

**University of Mumbai**

**2021-2022**

## **CERTIFICATE**

This is to certify that the T.E. mini-project entitled “*Face Mask Detection (openCV)*” is a bonafide work of “Aashish Shetye” Roll 46 [TEIT2], “Amaan Siddiqui” Roll 47 [TEIT2], “Gaurav Wankhede” Roll 58 [TEIT2], “Manish Yadav” Roll 60 [TEIT2] submitted to University of Mumbai in partial fulfilment of the requirement for the award of the degree of “**Information Technology**” during the academic year 2021–2022.

Prof. Nileema Pathak  
Guide name

**Prof. Deepali Maste**  
H.O.D.

**Dr. S.P. Kallurkar**  
Principal

# **T.E. Mini-Project Report Approval**

This mini-project synopsis entitled “*Face Mask Detection (openCV)*” by *Aashish Shetye, Amaan Siddiqui, Gaurav Wankhede and Manish Yadav* is approved for the degree of *Information Technology* from *University of Mumbai*.

## **Examiners**

1.

2.

Date: 30/04/2022

Place: Malad (West)

# Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will cause disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Signature

Aashish Shetye Roll 46

Amaan Siddiqui Roll 47

Gaurav Wankhede Roll 58

Manish Yadav Roll 60

Date: 30/04/2022

## ABSTRACT

Global pandemic COVID-19 circumstances emerged in an epidemic of dangerous disease in all over the world. Wearing a face mask will help prevent the spread of infection and prevent the individual from contracting any airborne infectious germs.

Using Face Mask Detection System, one can monitor if the people are wearing masks or not. Here *MobileNet* algorithm is used for image detection. Collating with other existing algorithms, this classifier produces a high recognition rate even with varying expressions, efficient feature selection and low assortment of false positive features.

New feature-based cascade classifier system utilizes only 200 features out of 6000 features to yield a recognition rate of 85-95%.

According to this motivation we demand mask detection as a unique and public health service system during the global pandemic COVID-19 epidemic.

The model is trained by face mask image and non-face mask image.

**Keywords:** COVID-19 epidemic, MobileNet algorithm, mask detection, face mask image, non-face mask image

## **Table of Contents**

<b>Sr. no.</b>	<b>Topic</b>
<b>1</b>	<b>Introduction</b>
<b>2</b>	<b>Problem statement</b>
<b>3</b>	<b>Objective</b>
<b>4</b>	<b>Literature Survey</b>
<b>5</b>	<b>Proposed System</b>
<b>6</b>	<b>Scope of Project</b>
<b>7</b>	<b>Hardware &amp; Software Requirements</b>
<b>8</b>	<b>Methodology</b>
<b>9</b>	<b>Implementation</b>
<b>10</b>	<b>Result Analysis</b>
<b>11</b>	<b>Conclusion</b>
<b>12</b>	<b>Reference</b>

# INTRODUCTION

- As we all know, due to COVID-19 pandemic, society need to embrace and adopt new norm that includes practicing social distance and wearing masks. If implemented properly, it can effectively reduce the transmission and severity of a disease, hence reducing the pressure on healthcare systems and allowing more time for government countermeasures.
- In this project, we propose a method which employs TensorFlow and OpenCV to detect human faces and proper mask wearing in images and webcam streams. Trained on MobileNetV2, a state-of-the-art lightweight deep learning model on image classification, the app is computationally efficient to deploy to help control the spread of the disease.
- While many work on face mask detection has been developed since the start of the pandemic, few distinguishes whether a mask is worn correctly or incorrectly. Given the discovery of the more transmissible coronavirus variant, we aim to provide a more precise detection model to help strengthen enforcement of mask mandate around the world.

## **PROBLEM STATEMENT**

- The system to be developed, aims to promote wearing masks by providing a classifying machine learning tool using OpenCV and TensorFlow to detect facemasks on people.

## **OBJECTIVE**

- Objective of our project is making a system which is easy to implement in any existing organizational system.
- And To identify the person on image/video stream wearing face mask with the help of computer vision and deep learning algorithm



## **LITERATURE SURVEY**

Here, we review three such existing systems with same system:

- The 1<sup>st</sup> paper is one of the few works which is concentrated on detection of human faces where they are wearing masks. They found that the accuracy in human face detection decreases by 70% when a face mask is present.
- In the 2<sup>nd</sup> Paper the authors have developed a method to identify how a person is wearing the face mask. They were able to classify three categories of facemask-wearing condition namely correct facemask-wearing, incorrect facemask- wearing, and no facemask-wearing. This method achieved over 98% accuracy in detection.
- And 3<sup>rd</sup> paper is about identifying facemask-wearing condition using image super resolution with classification network. They were able to classify three categories of facemask-wearing condition namely correct facemask-wearing, incorrect facemask-wearing, and no facemask-wearing. This method achieved over 98% accuracy in detection.

## PROPOSED SYSTEM

**This section contains information about the enhancements** or updates that are to be done to the proposed system by referring to the already existing works.

- Framework and libraries used in this project are:
  1. OpenCV: It is a computer vision library used to process images
  2. Tensorflow / Keras: deep learning framework used to build and train our models
  3. MobileNet V2: lightweight pre-trained model available in Keras Applications; used as a base model for our transfer learning
- We provide and the **MFN** dataset is built from the MaskedFace-Net.
- The dataset which is being used contains 3833 images out of which 1915 images have people with masks in them and 1918 people without masks in them.
- Hence, the methodology for the project is divided in 3 steps :
- Dataset Collection: The dataset was collected from Google two models trained on two different datasets. Our **RMFD** dataset is built from the Real World Masked Face Dataset
  1. and Kaggle Repository and was split into training and testing data after its analysis.
  2. Training a model to detect face masks: A default OpenCV module was used to train the model to identify face mask.
  3. Detecting the person not wearing a mask: After training the model, it was deployed through MobileNet to detect person with or without masks.

## **SCOPE & FEASIBILITY**

- With the increasing risk of contagious diseases all over the world, a system to replace humans to check masks on the faces of people is greatly needed. This system satisfies that need.
- This system can be employed in public places like railway stations and malls. It will be of a great help in companies and huge establishments where there will be a lot of workers.
- This system will be of a great help there because it is easy find the people who are not wearing the mask and can be easily alerted to take proper precautions.
- The system can be used in the following places to identify people with or without masks:
  1. Offices
  2. Hospitals/healthcare organizations
  3. Airports and railway stations
  4. Sports venues
  5. Entertainment and hospitality industry
  6. Densely populated areas.

## **Hardware & Software Requirements**

- **Hardware/Software Interface:**

This section lists the minimum hardware and software requirements needed to run the system efficiently.

- **Hardware Interface:**

- a) Processor
- b) 250 Gb of hard-drive space
- c) 8 Gb RAM or Above

- **Software Interface:**

- a) Operating System: Windows (7 or above), MacOS or Linux.
- b) Programming Language: Python
- c) Command Prompt for data processing and Training.

## **METHODOLOGY**

- 1.** This system is capable to train the dataset of both persons wearing masks and without wearing masks.
- 2.** After training the model the system can predict whether the person is wearing the mask or not.
- 3.** It can access the webcam and predict the result.
- 4.** Python modules used in the project are:

### **A. OPENCV:**

- It is a cross-platform library using which we can develop real-time computer vision applications.
- It mainly focuses on image processing, video capture and analysis including feature like face detection and object detection.

### **B. TENSORFLOW FRAMEWORK:**

- Tensor flow is also an open-source software library to run deep learning and other statistical and predictive analytics workloads.
- It is a python library that supports many classification and regression algorithms and more generally deep learning.

### **C. KERAS**

- it offers consistent & simple APIs, it contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image easier.
- It was developed to make implementing deep learning models as fast and easy as possible for research and development.

### **D. NUMPY**

- NumPy is a library for the Python, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high level mathematical functions to operate on these arrays.

### **E. MATPLOTLIB**

- Mat plot is a plotting library for the Python.

Finally talking about **MobileNet**, it is a type of convolutional neural network designed for embedded vision applications. They are based on a streamlined architecture that uses depth wise separable convolutions to build lightweight deep neural networks that can have low latency for embedded devices. We used this algorithm because it is vastly smaller in size and faster in performance than many other popular models.

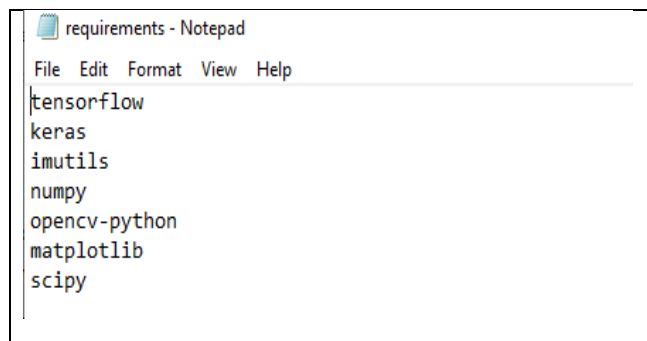
**CNN-** *A convolution neural network is a special architecture of artificial neural network. One of the most popular uses of the architecture is image classification. CNNs have wide applications in image and video recognition, recommender systems and natural language processing. CNNs, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output. The whole network has a loss function and all the tips and tricks that we developed for neural networks still apply on CNNs. In more detail the image is passed through a series of convolution, nonlinear, pooling layers and fully connected layers, then generates the output.*

## METHODOLOGY

Implementation of Project consists of various stages from data pre-processing and training the model to visualizing the result.

### ❖ Installing the Dependencies:

All the modules and libraries of python stored in *requirements.txt* file are installed first. Also, it is checked the compatibility error and resolve it beforehand.



```
requirements - Notepad
File Edit Format View Help
tensorflow
keras
imutils
numpy
opencv-python
matplotlib
scipy
```

### ❖ Collecting the Dataset:

Dataset for training the model is collected from various sources like Google and Kaggle repository under two categories, namely ‘with mask’ and ‘without mask.’



## Without mask:



### ❖ Data pre-processing:

Data pre-processing is converted raw data into an understandable format. Here, the main function of this step is to convert the images of dataset in an array form to create a deploying model to train.

- Now, we create a python file 'train\_mask\_detector', in this file we have imported the libraries and also mentioned the directory containing path of dataset.

```
DIRECTORY = r"D:/Mini Project 2B/Mini project/Face-Mask-Detection/dataset"
CATEGORIES = ["with_mask", "without_mask"]
```

- Then we create few objects, 'data' to store images and 'label' to store label of images. Then we create an object 'path' which will follow the path of directory to dataset in search of images.

```
data = []
labels = []
```

- Then we use list directory property which will help to list down all data in the path.

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



- After joining path to image, we use 'load\_img' of keras.preprocessing library and then use 'img\_to\_array' of same library to convert the images to arrays.

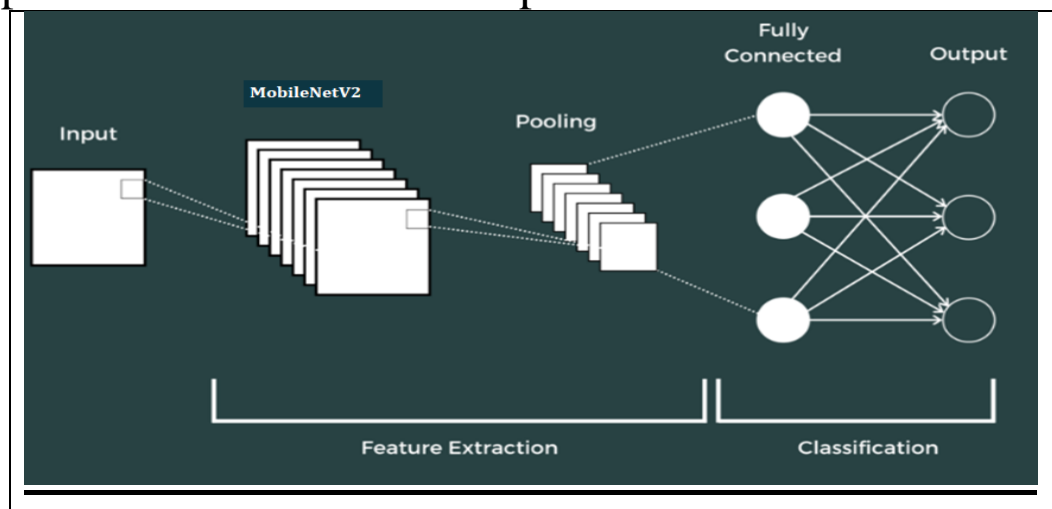
```
img_path = os.path.join(path, img)
image = load_img(img_path, target_size=(224, 224))
image = img_to_array(image)
image = preprocess_input(image)
```

- Labels of the images are also converted into numpy array using np.array(label).

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

### ❖ Model Training:

- Coming to the training part, after input image is processed as an array, we will use 'MobileNet' to train the model. Then that model is pooled, which means to accumulate the filtered images and reduce the size of representation for better computation in network.



- After pooling a fully connected layer of filtered array is formed which then gives us the output.
- To understand it more easily, we have 2 models in MobileNet: base model which accesses the preprocessed input and fixes the representation size of image through 'input\_tensor'. This model is then passed to the head model where pooling takes place.

```
# Let's call
baseModel = MobileNetV2(weights="imagenet", include_top=False,
| input_tensor=Input(shape=(224, 224, 3)))
```

- Then finally a function 'model' is called, with two attribute input of base model and output of head model.

```
# The second model we will create
model = Model(inputs=baseModel.input, outputs=headModel)
```

- Then the generated model is saved and accuracy is plotted using Matplotlib and it is saved in image (png) form.

```
plt.plot(np.arange(0, N), H.history["val_
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig("plot.png")
```

## MODEL DEPLOYMENT

In this section, we will see how the trained model is deployed to get the final result.

### ❖ Applying the trained model:

- Here, we create a python file where we use openCV for face detection.
- In the python file we will create objects, 'readNet' of cv2 module for detecting the faces, 'maskNet' to detect the mask using load model.

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

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

- Then we will load the camera using videostream, another object frame is created to read the frame of videostream.

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

... ..
```

### ❖ Creating a function:

- Here, we will define a function using all three objects and will return the function with attributes 'location' and 'prediction'.

```
frame = vs.read()
frame = imutils.resize(frame, width=400)

(locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
```

- Location gives the x,y coordinates of rectangle surrounding the face and prediction gives the accuracy of person wearing the mask.

```
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
```

#### ❖ Predicting the results:

- Here, with the coordinates in table form we predict the result using our trained model.
- Since we trained our model with two categories – mask and without mask, the result will be percentage of each category predicted.

```
# determine the class label and color we'll use to draw  
# the bounding box and text  
label = "Mask" if mask > withoutMask else "No Mask"
```

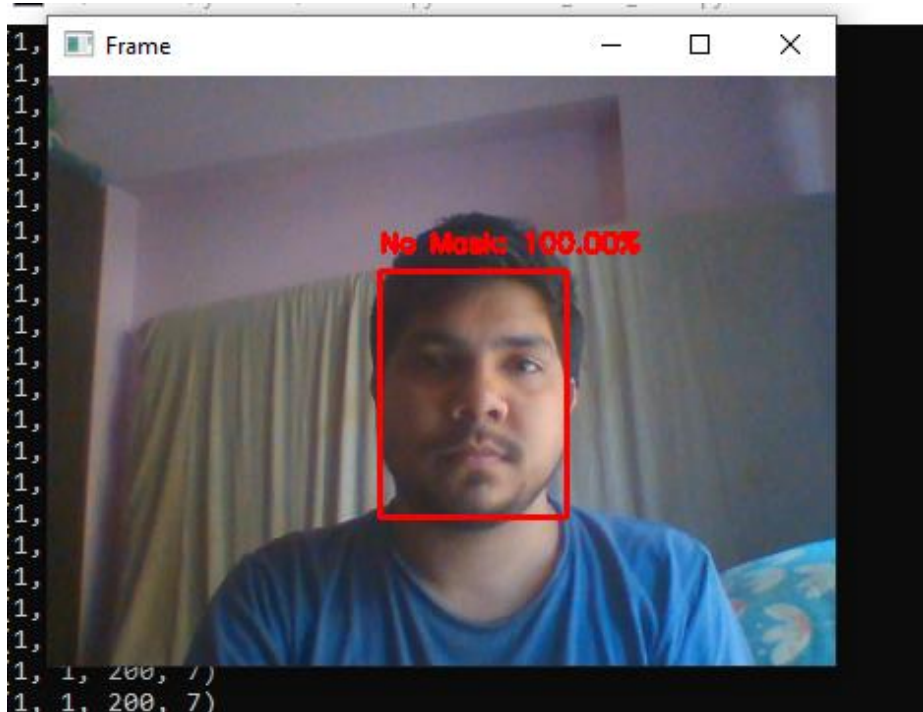
#### ❖ Displaying the result:

- We label each category, if the mask is predicted label it as “mask” else “no mask”.
- Finally we use color attribute for more comprehension of the result. If the label says, mask then the value of green color is increased else the value of red color is increased.

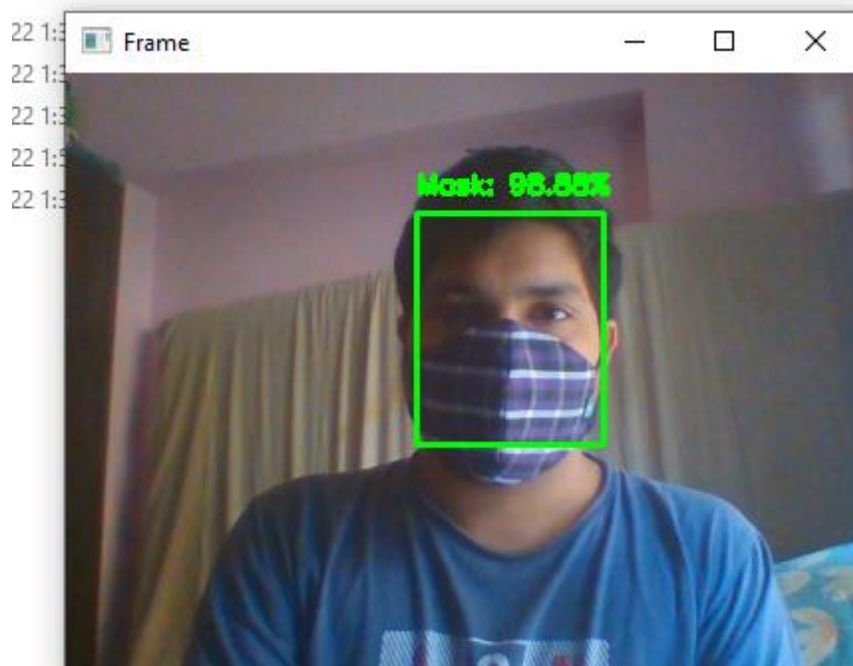
```
color = (0, 255, 0) if label == "Mask" else (0, 0, 255)
```

## RESULT ANALYSIS

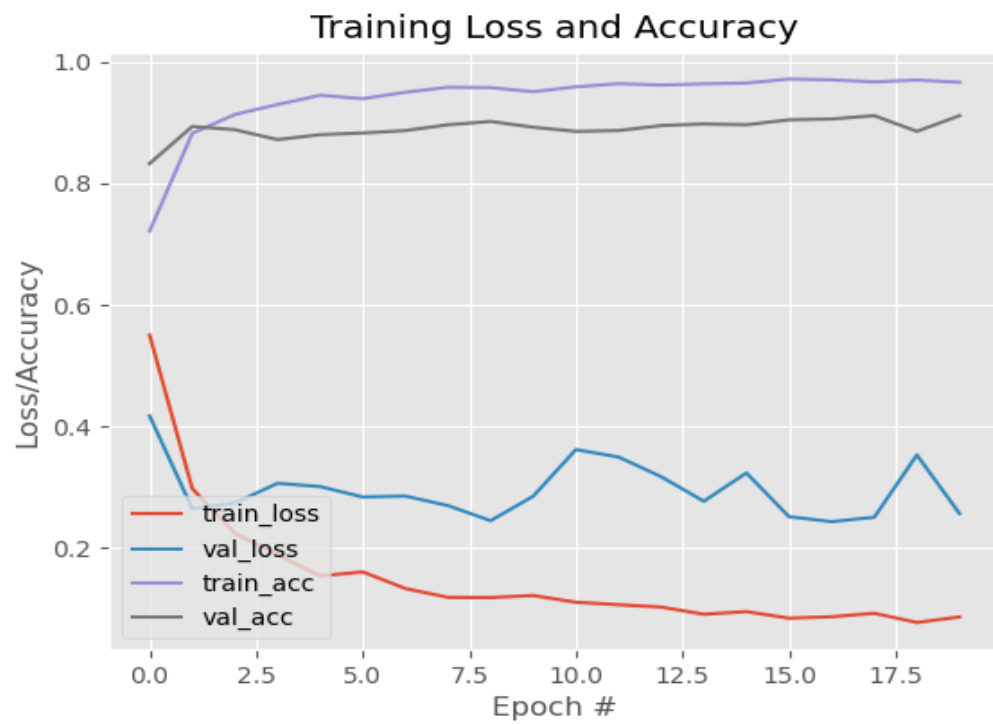
- ❖ After the model is trained and deployed, we will see the output.
- ❖ Without Mask:



- ❖ Without Mask:



❖ Accuracy:



## **CONCLUSION**

As the technology are blooming with emerging trends the availability so we have novel face mask detector which can possibly contribute to public healthcare. The architecture consists of Mobile Net as the backbone it can be used for high and low computation scenarios. The models were tested with images and real-time video streams. The accuracy of the model is achieved and, the optimization of the model is a continuous process.

## REFERENCE

❖ OpenCV tutorial:

[https://docs.opencv.org/4.x/d9/df8/tutorial\\_root.html](https://docs.opencv.org/4.x/d9/df8/tutorial_root.html)

❖ Image/Object classification with MobileNet:

<https://medium.com/analytics-vidhya/image-classification-with-mobilenet-cc6fbb2cd470>

❖ Classification using Keras:

<https://www.tensorflow.org/tutorials/keras/classification>

❖ Surveyed material:

1. <https://ieeexplore.ieee.org/document/8934543>
2. <https://www.mdpi.com/1424-8220/20/18/5236>