MINI PROJECT REPORT

ON

**FACE MASK DETECTOR**

**Using Python Programming**

Submitted as fulfillment of the requirements

of the award of the degree of

Bachelor of Technology In

Computer Science and Engineering

BY

**Aditya Mittal – 1809710007**

**Divyanshu Saxena – 1809710039**

Under the supervision of

Mr. Mayank Dixit

Mr. T. Akilan

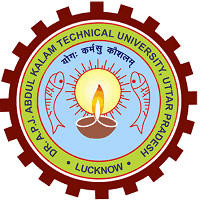


Galgotias College of Engineering and Technology

Greater Noida, Uttar Pradesh

India

Affiliated to



Dr. A.P.J Abdul Kalam Technical University

Lucknow, Uttar Pradesh

India



**FACE MASK DETEctor**

USING PYTHON PROGRAMMING



January 6, 2020

**Aditya Mittal - 1809710007**

**Divyanshu Saxena - 1809710039**

# **ACKNOWLEDGEMENT**

We would like to express our deepest appreciation to all those who provided us the opportunity to complete this project. We would like to thank the department of CSE at Galgotias College of Engineering and Technology for taking these initiatives and motivating students to explore their technical skills.

A special gratitude, we would like to give to our professor Mr. Mayank Dixit, whose contribution in stimulating suggestions and encouragement, helped us to coordinate our project successfully. We would like to thank him for giving us this opportunity and for always being there, helping and guiding us the right path to accomplish our goal. It was only because of his guidance, that we were able to complete our project successfully.

# 

# **ABSTRACT**

The COVID-19 pandemic is considered as the most crucial global health calamity of the century and the greatest challenge that the humankind faced since the 2nd World War. In December 2019, a new infectious respiratory disease emerged in Wuhan, Hubei province, China and was named by the World Health Organization as COVID-19 (coronavirus disease 2019). A new class of corona virus, known as SARS-CoV-2 (severe acute respiratory syndrome coronavirus 2) has been found to be responsible for occurrence of this disease. As far as the history of human civilization is concerned there are instances of severe outbreaks of diseases caused by a number of viruses. According to the report of the World Health Organization (WHO as of April 18 2020), the current outbreak of COVID-19, has affected over 2164111 people and killed more than 146,198 people in more than 200 countries throughout the world. Till now there is no report of any clinically approved antiviral drugs or vaccines that are effective against COVID-19. It has rapidly spread around the world, posing enormous health, economic, environmental and social challenges to

the entire human population. The coronavirus outbreak is severely disrupting the global economy. Almost all the nations are struggling to slow down the transmission of the disease by testing & treating patients, quarantining suspected persons through contact tracing, restricting large gatherings, maintaining complete or partial lock down etc. This paper describes the impact of COVID-19 on society and global environment, and the possible ways in which the disease can be controlled has also been discussed therein.

**INDEX**

1. **Introduction**
2. **Face Detection Algorithm**
3. **Source Code**
4. **Future Aspects and limitations**
5. **Conclusion**
6. **Reference**

**INTRODUCTION**

A new strain which has not previously been identified in humans is novel coronavirus (nCoV). Coronaviruses (CoV) are a wide group of viruses which cause illness that range from colds to deadly infections like Middle East Respiratory Syndrome (MERS) and Severe Acute Respiratory Syndrome (SARS) [1]. The first infected patient of coronavirus has been found in December 2019. From that period, COVID-19 has become a pandemic all over the world [2]. People all over the world are facing challenging situations due to this pandemic. Every day a large number of people are being infected and died. At the time of writing this paper, almost 16,207,130 infected cases have been confirmed where 648,513 are death [3]. This number is increasing day by day. Fever, dry cough, tiredness, diarrhea, loss of taste, and smell are the major symptoms of coronavirus which is declared by the World Health Organization (WHO) [4]. Many precautionary measures have been taken to fight against coronavirus. Among them cleaning hands, maintaining a safe distance, wearing a mask, refraining from touching eyes, nose, and mouth are the main, where wearing a mask is the simplest one. COVID-19 is a disease that spread from human to human which can be controlled by ensuring proper use of a facial mask. The spread of COVID-19 can be limited if people strictly maintain social distancing and use a facial mask. Very sadly, people are not obeying these rules properly which is speeding the spread of this virus. Detecting the people not obeying the rules and informing the corresponding authorities can be a solution in reducing the spread of coronavirus.

A face mask detection is a technique to find out whether someone is wearing a mask or not. It is similar to detect any object from a scene. Many systems have been introduced for object detection. Deep learning techniques are highly used in medical applications [5], [6]. Recently, deep learning architectures [7] have shown a remarkable role in object detection. These architectures can be incorporated in detecting the mask on a face. Moreover, a smart city [8] means an urban area that consists of many IoT sensors to collect data. These collected data are then used to perform different operations across the city.

**FACE DETECTION ALGORITHM**

One of the popular algorithms which we used is Viola Jones Algorithm.

Viola-Jones algorithm is named after two computer vision researchers who proposed the method in 2001, Paul Viola and Michael Jones in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features”. Despite being an outdated framework, Viola-Jones is quite powerful, and its application has proven to be exceptionally notable in real-time face detection. This algorithm is painfully slow to train but can detect faces in real-time with impressive speed.

Given an image(this algorithm works on grayscale image), the algorithm looks at many smaller subregions and tries to find a face by looking for specific features in each subregion. It needs to check many different positions and scales because an image can contain many faces of various sizes. Viola and Jones used Haar-like features to detect faces in this algorithm.

**SOURCE CODE**

# 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

from imutils.video import VideoStream

import numpy as np

import imutils

import time

import cv2

import os

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)

# initialize our list of faces, their corresponding locations,

# and the list of predictions from our face mask network

faces = []

locs = []

preds = []

# 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 > 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))

# extract the face ROI, convert it from BGR to RGB channel

# ordering, resize it to 224x224, and preprocess it

face = frame[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)

# add the face and bounding boxes to their respective

# lists

faces.append(face)

locs.append((startX, startY, endX, endY))

# only make a predictions if at least one face was detected

if len(faces) > 0:

# for faster inference we'll make batch predictions on \*all\*

# faces at the same time rather than one-by-one predictions

# in the above `for` loop

faces = np.array(faces, dtype="float32")

preds = maskNet.predict(faces, batch\_size=32)

# return a 2-tuple of the face locations and their corresponding

# locations

return (locs, preds)

# load our serialized face detector model from disk

prototxtPath = r"face\_detector\deploy.prototxt"

weightsPath = r"face\_detector\res10\_300x300\_ssd\_iter\_140000.caffemodel"

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

# 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:

# 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)

# loop over the detected face locations and their corresponding

# locations

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)

# 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(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

# if the `q` key was pressed, break from the loop

if key == ord("q"):

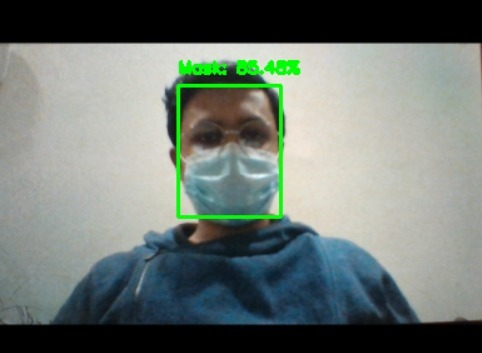
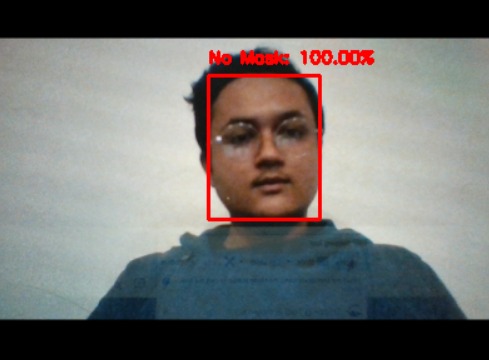
break

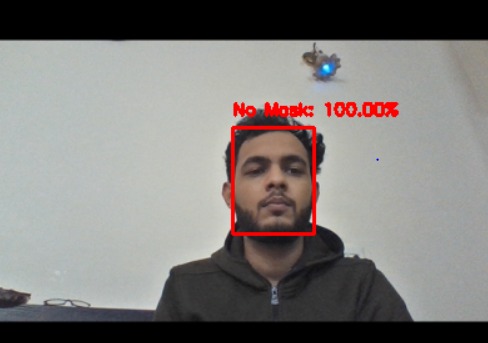
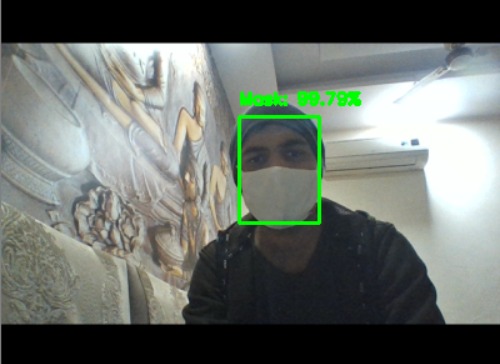
# do a bit of cleanup

cv2.destroyAllWindows()

vs.stop()

**OUTPUT**



 ****

**FUTURE ASPECTS AND LIMITATIONS**

The developed system faces difficulties in classifying faces covered by hands since it almost looks like the person wearing a mask. While any person without a face mask is traveling on any vehicle, the system cannot locate that person correctly. For a very densely populated area, distinguishing the face of each person is very difficult. For this type of scenario, identifying people without face mask would be very difficult for our proposed system. In order to get the best result out of this system, the city must have a large number of CCTV cameras to monitor the whole city as well as dedicated manpower to enforce proper laws on the violators. Since the information about the violator is sent via SMS, the system fails when there is a problem in the network.

The proposed system mainly detects the face mask and informs the corresponding authority with the location of a person not wearing a mask. Based on this, the authority has to send their personnel to find out the person and take necessary actions. But this manual scenario can be automated by using drones and robot technology [22], [23] to take action instantly. Furthermore, people near to the person not wearing a mask may be alerted by an alarm signal on that location, and displaying the violators face in a LED screen to maintain a safe distance from the person would be a further study.

**CONCLUSION**

In this project, we have developed a deep learning model for face mask detection using Python, Keras, and OpenCV. We developed the face mask detector model for detecting whether person is wearing a mask or not. We have trained the model using Keras with network architecture. Training the model is the first part of this project and testing using webcam using OpenCV is the second part.

This is a nice project for beginners to implement their learnings and gain expertise.

**REFERENCES**

1. YOUTUBE
2. GOOGLE
3. COURSERA
4. GITHUB