FACE MASK DETECTION USING PYTHON

By

SAI ABHISHEK

18BEC1154

C RISHIT

18BEC1204

MEGHA BHASKER

18BEC1217

ABINESH VEL

18BEC1340

A project report submitted to

Dr. SIVAKUMAR. S

Associate Professor

SCHOOL OF ELECTRONICS ENGINEERING

in partial fulfilment of the requirements for the course of

ECE2006 – DIGITAL SIGNAL PROCESSING

in

B.Tech. ELECTRONICS AND COMMUNICATION ENGINEERING

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Vandalur – Kelambakkam

Road Chennai – 600127

OCTOBER 2020

BONAFIDE CERTIFICATE

Certified that this project report entitled "**FACE MASK DETECTION USING PYTHON"** is a bonafide work of **SAI ABHISHEK 18BEC1154, C RISHIT 18BEC1204, MEGHA BHASKER 18BEC1217, and ABINESH VEL 18BEC1340** who carried out the Project work under my supervision and guidance for **ECE2006 – DIGITAL SIGNAL PROCESSING.**

Dr. Sivakumar. S

Associate Professor

School of Electronics Engineering (SENSE),

VIT University, Chennai

Chennai – 600 127.

ABSTRACT

This report shows a comprehensive project using python, intended to detect if people are wearing face masks or not. This is due the prevalence of SARS- COVID-19 (Severe Acute Respiratory Syndrome – Coronavirus Infectious Disease – 2019) caused by the SARS-COV (Severe Acute Respiratory Syndrome – Coronavirus) and can be implemented using data available in the public domain with the aim of ascertaining the most effective method to combat this and any future pandemics.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. Sivakumar. S** Associate Professor, School of Electronics Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. Sivasubramanian. A,** Dean of School of Electronics Engineering, VIT Chennai, for extending the facilities of the School towards our project and for her unstinting support. We express our thanks to our Head of the Department **Dr. Vetrivelan. P**

for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for

their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

Sai Abhishek **Rishit C** **Megha Bhasker** **Abinesh Vel**

Table of Contents:

CHAPTER 1

1.1 AIM

Face Mask Detection using Python involves the usage of the Haar cascade, cascade classifiers and other related appendages to aid in the detection of masks on a person's face.

1.2 OBJECTIVES

The main objective of the project is to identify whether the person is wearing a mask or not. Since the COVID 19 epidemic hit, public safety and hygiene is number one priority.

1.3 BENEFITS

COVID-19 is a rapidly evolving pandemic that began in the metropolitan city of Wuhan in the Hubei province of the People's Republic of China ; since spreading globally, it has posed an indomitable challenge to governments across the world which have been unsuccessfully attempting to bring it under control. This offers a comprehensive, result-oriented analysis which governments and nation-states can choose to co-opt in a manner which is most useful to them.

COVID-19 spreads mainly from person to person through respiratory droplets. Respiratory droplets travel into the air when you cough, sneeze, talk, shout, or sing. These droplets can then land in the mouths or noses of people who are near you or they may breathe these droplets in. Masks are a simple barrier to help prevent your respiratory droplets from reaching others. Studies show that masks reduce the spray of droplets when worn over the nose and mouth.

This project helps to identify whether the person is wearing a mask or not. This can be used in any public places like Malls, Airport, Railway station as a part of checking procedure by the Law Officers like Police, to make sure only people wearing mask are allowed in to maintain the safety of the public. Of course, this can be done manually, here the officials have to check each individual personally, and it's a redundant and time-consuming job. Our project aims to help such officials by making accurate predictions using the concept of Haar Cascades, to determine if the person is wearing a mask or not, to make the checking procedure faster, and safer for these officials.

1.4 THEORY

Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and based on the concept of features proposed by Paul Viola and Michael Jones in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001.

It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

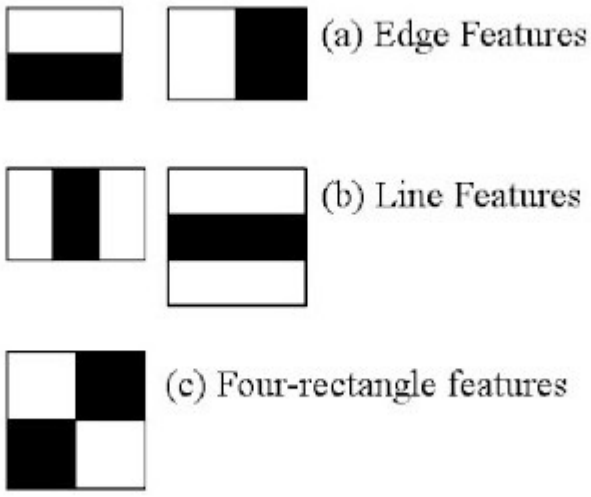
The algorithm has four stages:

1. Haar Feature Selection
2. Creating Integral Images
3. Adaboost Training
4. Cascading Classifiers

It is well known for being able to detect faces and body parts in an image, but can be trained to identify almost any object.

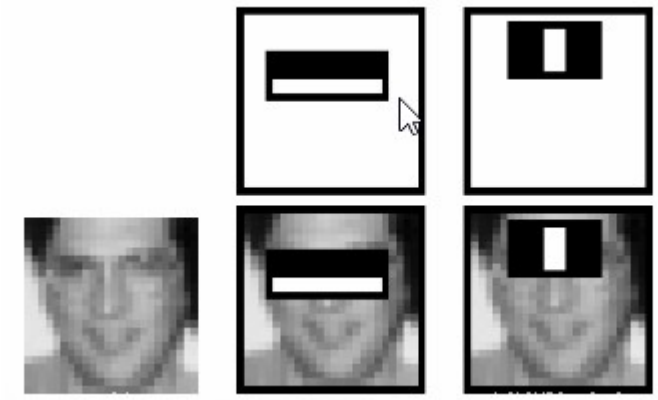
Lets take face detection as an example. Initially, the algorithm needs a lot of positive images of faces and negative images without faces to train the classifier. Then we need to extract features from it.

First step is to collect the Haar Features. A Haar feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums.

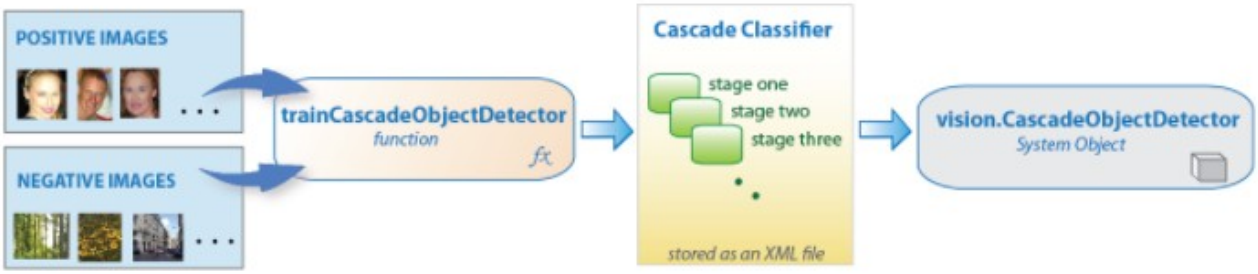


But among all these features we calculated, most of them are irrelevant. For example, consider the image below. Top row shows two good features. The first feature selected seems to focus on the

property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applying on cheeks or any other place is irrelevant.



we select the best features out of 160000+ features using a concept called Adaboost which both selects the best features and trains the classifiers that use them. This algorithm constructs a “strong” classifier as a linear combination of weighted simple “weak” classifiers. The process is as follows.



Cascade Classifier block diagram

Adaboost Classifier:

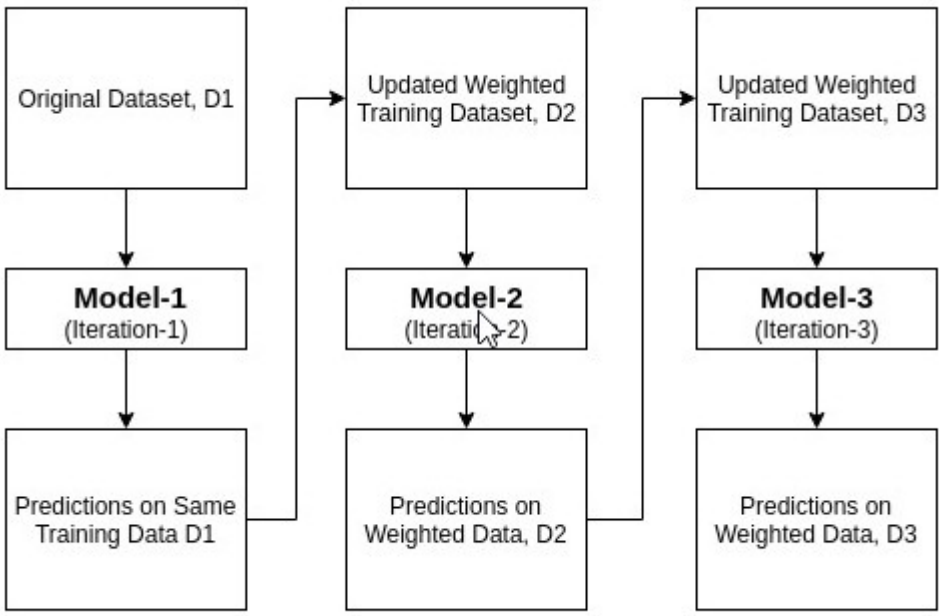
Ada-boost or Adaptive Boosting is one of ensemble boosting classifier proposed by Yoav Freund and Robert Schapire in 1996. It combines multiple classifiers to increase the accuracy of classifiers. AdaBoost is an iterative ensemble method. AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier. The basic concept behind Adaboost is to set the weights of classifiers and training the data sample in each iteration such that it ensures the accurate predictions of unusual observations. Any machine learning algorithm can be used as base classifier if it accepts weights on the training set. Adaboost should meet two conditions:

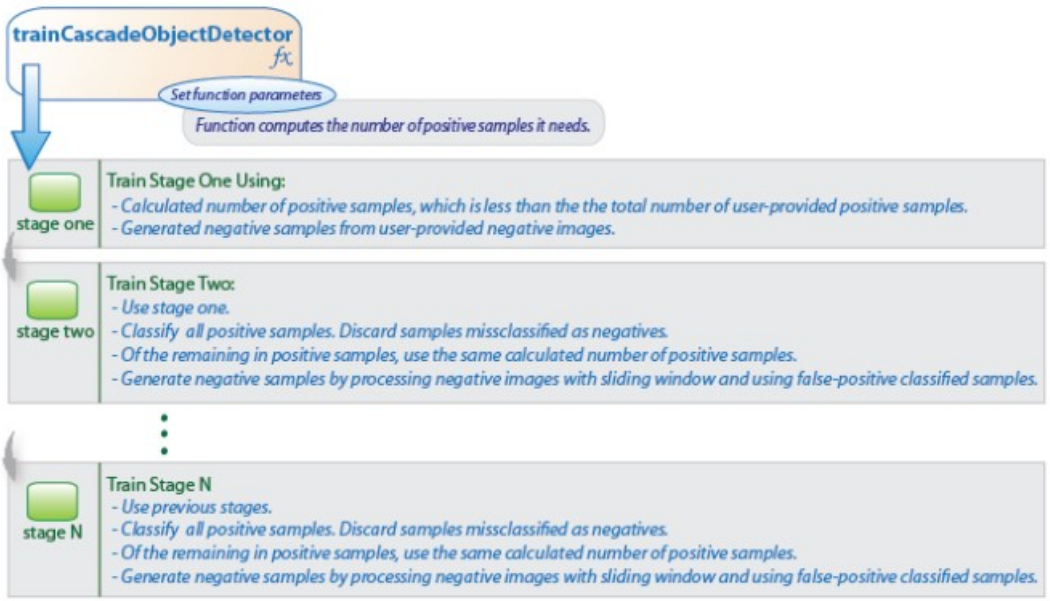
The classifier should be trained interactively on various weighed training examples.
In each iteration, it tries to provide an excellent fit for these examples by minimizing training error.

It works in the following steps:

1. Initially, Adaboost selects a training subset randomly.
2. It iteratively trains the AdaBoost machine learning model by selecting the training set based on the accurate prediction of the last training.
3. It assigns the higher weight to wrong classified observations so that in the next iteration these observations will get the high probability for classification.

4. Also, It assigns the weight to the trained classifier in each iteration according to the accuracy of the classifier. The more accurate classifier will get high weight.
5. This process iterate until the complete training data fits without any error or until reached to the specified maximum number of estimators.
6. To classify, perform a "vote" across all of the learning algorithms you built.





CHAPTER 2:

Implementation, Code and Working:

2.1 IMPLEMENTATION:

We need to make sure the pre-requisites are taken care of. The requirements are listed below.

Libraries and Files Required:

1. OpenCV (version 2.7.10)
2. Haar cascade for frontal face
3. Haar cascade for eye

The code (below) should be ran using a python interpreter, in this case we have used the Microsoft's Visual Studio as the Integrated Development Environment.

Note: This code only works on OpenCV v2.7. The latest versions do not support all the functions used in this code, therefore it will return errors.

2.2 Python Code:

```
import numpy as np

import cv2

import random

face_cascade = cv2.CascadeClassifier('data\\xml\\
haarcascade_frontalface_default.xml')

eye_cascade = cv2.CascadeClassifier('data\\xml\\
haarcascade_eye.xml')

mouth_cascade = cv2.CascadeClassifier('data\\xml\\
haarcascade_mcs_mouth.xml')

upper_body = cv2.CascadeClassifier('data\\xml\\
haarcascade_upperbody.xml')


# Adjust threshold value in range 80 to 105 based on your light.

bw_threshold = 80


# User message

font = cv2.FONT_HERSHEY_SIMPLEX

org = (30, 30)

yes_mask_font_color = (0,128,0)

no_mask_color = (0, 0, 255)

thickness = 2
```

```
font_scale = 1
yes_mask = "Mask Detected. Thank You."
no_mask = "Mask Not Detected."



# Read video
cap = cv2.VideoCapture(0)

while 1:
    # Get individual frame
    ret, img = cap.read()
    img = cv2.flip(img,1)

    # Convert Image into gray
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Convert image in black and white
    (thresh, black_and_white) = cv2.threshold(gray, bw_threshold, 255,
cv2.THRESH_BINARY)
    #cv2.imshow('black_and_white', black_and_white)

    # detect face
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)

    # Face prediction for black and white
```

```
    faces_bw = face_cascade.detectMultiScale(black_and_white, 1.1,
4)



    if(len(faces) == 0 and len(faces_bw) == 0):
        cv2.putText(img, "No face found...", org, font, font_scale,
yes_mask_font_color, thickness, cv2.LINE_AA)
    elif(len(faces) == 0 and len(faces_bw) == 1):
        # It has been observed that for white mask covering mouth, with
gray image face prediction is not happening
        cv2.putText(img, yes_mask, org, font, font_scale,
yes_mask_font_color, thickness, cv2.LINE_AA)
    else:
        # Draw rectangle on gace
        for (x, y, w, h) in faces:
            cv2.rectangle(img, (x, y), (x + w, y + h), (255, 255, 255), 2)
            roi_gray = gray[y:y + h, x:x + w]
            roi_color = img[y:y + h, x:x + w]



            # Detect lips counters
            mouth_rects = mouth_cascade.detectMultiScale(gray, 1.5, 5)


        # Face detected but Lips not detected which means person is
wearing mask
        if(len(mouth_rects) == 0):
```

```
            cv2.putText(img, yes_mask, org, font, font_scale,
yes_mask_font_color, thickness, cv2.LINE_AA)
        else:
            for (mx, my, mw, mh) in mouth_rects:


                if(y < my < y + h):
                    # Face and Lips are detected but lips coordinates are
within face cordinates which `means lips prediction is true and
                    # person is not waring mask
                    cv2.putText(img, no_mask, org, font, font_scale,
no_mask_color, thickness, cv2.LINE_AA)


                    #cv2.rectangle(img, (mx, my), (mx + mh, my + mw), (0,
0, 255), 3)
                    break


    # Show frame with results
    cv2.imshow('Mask Detection', img)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

# Release video
cap.release()
cv2.destroyAllWindows()
```

2.3 Working of the code:

The basic idea behind the project is to detect if a person is wearing a mask or not. If a person is wearing a mask then ideally the mask should cover the person's mouth and nose. If he is not wearing a mask then his mouth and nose should be visible.

So instead of looking for the mask, we can use the haar cascades to check if the person's mouth and nose is visible or not. Hence, we have used 2 haar cascades files in the current project. One is the classifier for the frontal face, second is the classifier for the eyes.

Using the concepts of deep learning and haar cascades discussed above, we find if the person's mouth is visible. If the mouth is not visible then that means the person is wearing a mask. Second case is if the eyes are not detected the output will be no face found.

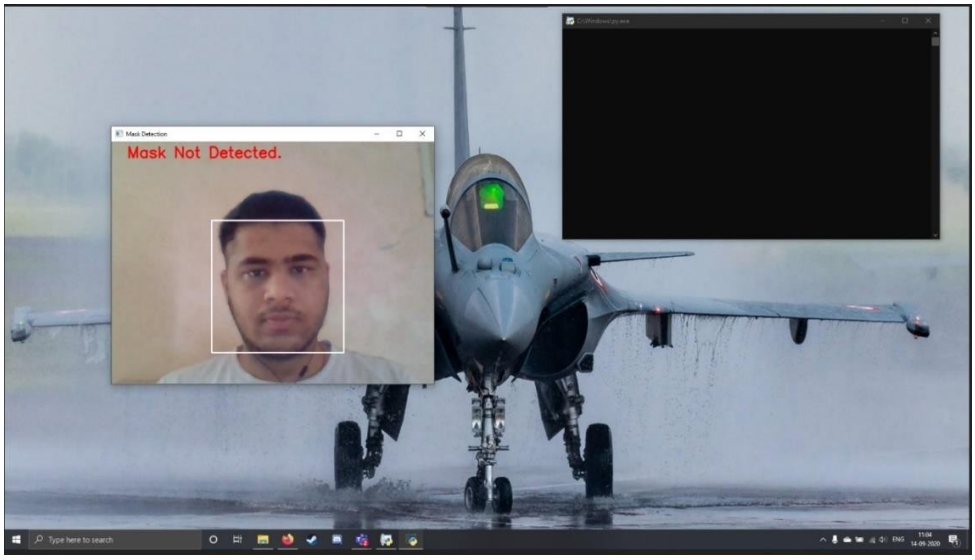
CHAPTER 3:

RESULTS AND INFERENCE

3.1 Results:

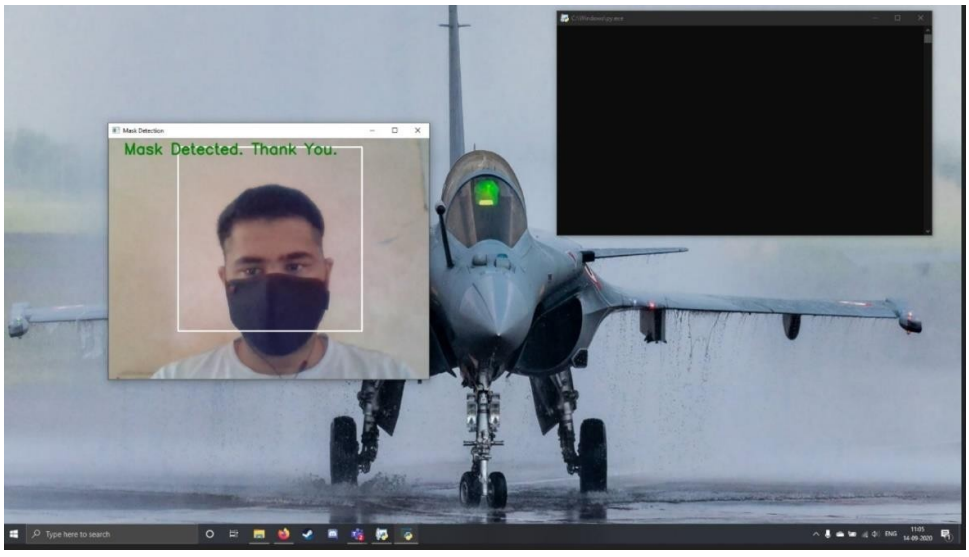
Case 1:

MASK NOT DETECTED:



Case 2:

MASK DETECTED:



3.2 Inference

Using the concepts of deep learning and haar cascades discussed above, we find if the person's mouth is visible. We have seen that in case1, the program does not detect the presence of mouth and nose, so it returns a positive message, and in the second case since the haar cascade detects the presence of mouth and nose, it concludes that the person is not wearing a mask.

CHAPTER 4

Conclusion and Future work

4.1 CONCLUSION

- Thus, we have created a program where it can be ascertained if people are wearing masks or not.
- Furthermore, safety of all people in a pandemic can only be assured when everyone irrespective wage, class, and wealth are truly safe, and the pandemic is completely eradicated.
- We believe that not only have we completed our project, but also satisfied civic duty and responsibility in creation of a reasonably accurate model with 95-98% accuracy and a similar precision as it would help the Government of India.

4.2 FUTURE WORK

The model is most certainly very accurate in terms of the results, but there are certain loopholes. For example, since it looks for the mouth and a nose, if you cover your mouth with your hand it will say that the mask is detected.

Hence to avoid this loophole, we need to redesign it from the fundamentals. Instead of using haar cascade to detect mouth, we need to design our own deep neural network with set of training data and classifiers to directly detect the mask, and not the body features.

References:

1. Wikipedia, Wikipedia. “AdaBoost.” Wikipedia, Wikimedia Foundation, 13 Jan. 2018, en.wikipedia.org/wiki/AdaBoost.

2. Docs, OpenCV. “Face Detection Using Haar Cascades.” OpenCV: Face Detection Using Haar Cascades, 4 Aug. 2017, docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html.

3. Wikipedia, Wikipedia. “Viola–Jones object detection framework.” Wikipedia, Wikimedia Foundation, 4 November. 2017, en.wikipedia.org/wiki/Viola–Jones_object_detection_framework.

4. Wikipedia, Wikipedia. “Cascading Classifiers.” Wikipedia, Wikimedia Foundation, 15 October. 2013, en.wikipedia.org/wiki/Cascading_classifiers.

5. Mathworks, Mathworks. “Train a Cascade Object Detector” Mathworks, 2017, www.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html.

6. OpenCV Documentation - https://docs.opencv.org/master/