A Project Report

on

FACE MASK DETECTION USING OPEN CV(MACHINE LEARNING)

Submitted in partial fulfillment of the requirement for the award of the degree of

B.TECH CSE

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of

Name of Supervisor:
Dr. Avneesh Kumar

Submitted By

Shashank Singh/18SCSE1010561
Shoaib Akhtar/19SCSE1010824

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING /
DEPARTMENT OF COMPUTERAPPLICATION
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
DECEMBER, 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled **"FACE MASK DETECTION USING OPEN CV(MACHINE LEARNING)"** in partial fulfillment of the requirements for the award of the Shashank Singh and Shoiab Akhtar submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of December, 2021 , under the supervision of Name Dr. Avneesh Kumar Associate Professor, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Shashank singh

Shoaib Akhtar

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dr. Avneesh Kumar

Associate Professor

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of **Name:Shashank Singh/18SCSE1010561 and Shoaib Akhtar/19SCSE1010824** has been held on _________________ and his/her work is recommended for the award of B.Tech CSE~~.~~

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: December, 2021

Place: Greater Noida

Table of Contents

Abstract

The novel Coronavirus had brought a new normal life in which the social distance and wearing of face masks plays a vital role in controlling the spread of virus. But most of the people are not wearing face masks in public places which increases the spread of viruses. This may result in a serious problem of increased spreading. Hence to avoid such situations we have to scrutinize and make people aware of wearing face masks. Humans cannot be involved for this process, due to the chance of getting affected by corona.

The COVID-19 epidemic has quickly affected our daily lives by disrupting global trade and movement. Wearing a protective face mask has become a new trend. In the near future, more and more public service providers will be asking customers to wear masks properly to access their services. Therefore, facial recognition has become an important function of helping the international community. This report outlines a simplified approach to achieving this goal using basic machine learning packages such as OpenCV and Scikit-Learn. The proposed method detects the surface from the image well and indicates whether it has a mask on it or not. As a watchmaker, you can also find a face and a movement mask. This approach achieves accuracy of up to 95.77% and 94.58% respectively on two different datasets. We test the adjusted parameters of the parameters using the Scikit-Learn to determine the presence of the mask properly without causing excessive   alignment.

CHAPTER-1

Introduction

The novel coronavirus covid-19 had added a brand new everyday lifestyles. India is suffering to get out of this virus attack and the government carried out lockdown for the lengthy manner. Lockdown located a stress on the global economy. So the authorities gave relaxations in lockdown . Declared by way of the WHO that a capacity speech by using preserving distance and sporting a masks is vital. the most important aid that the government wishes after relaxation is social distancing and sporting of masks via the humans.however many people are becoming out without a face masks this may increase the unfold of covid-19. monetary instances India has stated that " Survey shows that ninety percentage Indians are aware, however simplest 44 percent carrying a mask ". This survey genuinely points that humans are conscious but they're not sporting the mask because of a few soreness in carrying and carelessness.this may result inside the clean spreading of covid-19 in public locations.the arena fitness organisation has definitely said that until vaccines are discovered the sporting of masks and social distancing are key gear to lessen spread of virus.So it is crucial to make people put on mask in public locations. In densely populated areas it's miles hard to find the persons not wearing the face mask and warn them.therefore we're the use of picture processing techniques for identity of people wearing and now not carrying face mask. In real time pics are collected from the camera and it is processed in Raspberry Pi embedded improvement kit.The actual time pix from the camera are in comparison with the skilled dataset and detection of carrying or not carrying a mask is achieved.

The educated dataset is made by means of using system getting to know technique that's the figuring out issue of the result.

The set of rules created by using the usage of a trained dataset will locate the persons with and with out carrying face mask. The internet of things (IOTs) can be used for

connecting objects like smartphones, net TVs, laptops, computer systems, sensors and actuators to the net wherein the devices are related collectively to enable new types of verbal exchange between things and those, and between things themselves. Intimation messages are sent to authority persons by using IOT .The bankruptcy 2 explains in detail the grade by grade manner of the proposed algorithm. The experimental setup of the proposed hardware version is defined in detail in bankruptcy 3. The experimental consequences acquired are in short plotted in chapter four. bankruptcy five concludes the proposed approach overall performance and its related destiny work. According to the World Health Organization's (205) official Situation Report, coronavirus 2019 (COVID-19) has infected more than 20 million people worldwide and caused more than 0.7 million deaths . People with COVID-19 had a wide range of reported symptoms - from mild manifestations to serious illness. Respiratory problems such as shortness of breath or difficulty breathing are one of them. Older people with pneumonia may have more serious complications from COVID-19 as they appear to be at greater risk . Other common human coronaviruses that infect humans worldwide are 229E, HKU1, OC43, and NL63. Before depleting human immunity, viruses such as 2019-nCoV, SARS-CoV, and MERS-CoV infect animals and turn them into human coronaviruses . People with respiratory problems can expose anyone (closest to them) to the beads of infection. The area around a contaminated person can cause contact transmission as droplets carrying the virus can reach their immediate surroundings .

To reduce certain respiratory infections, including COVID-19, wearing a clinical mask is much needed. The public should know whether to wear a resource management mask or COVID-19 disgust. Points may be of interest to the use of masks lying down to reduce the risk of injury from a dangerous person during a "pre-symptomatic" period and the stigma of different people wearing masks to prevent the spread of the virus. The WHO emphasizes prioritizing medical masks and respiratory health care providers . Therefore, facial recognition has become an important activity in the current global community.

Finding a face mask involves finding the location of the face and deciding whether to have a mask on or not. This problem is very close to finding a common object to find categories of objects. Face ID works by distinguishing a specific group of objects called Face. It has many applications, such as private driving, education, surveillance, and so on . This project outlines a simplified way to achieve the above goal using basic Machine Learning (ML) packages such as OpenCV and Scikit-Learn.

CHAPTER-2

LITERATURE REVIEW

TITLE : Face Mask Detector

Single Shot Detector architecture is used for the object detection purpose. In this system face mask detector can be deployed in many areas like shopping malls, airports and other heavy traffic places to monitor the public and to avoid the spread of the disease by checking who is following basic rules and who is not.It takes excessive time for data loading in Google Colab Notebook. It did not allow the access of webcam which posed a hurdle in testing images and video stream.We have modeled a facemask detector using Deep learning. We are processed a system computationally efficient using MobileNetV2 which makes it easier to Extract the data sets. We use   architecture for better performance. We can fix it in any kind of cameras

TITLE : Face detection techniques: a review, Artificial

Human beings have not tremendous ability to identify different faces than machines, so automatic face detection system plays an important role in face recognition, head-pose estimation etc. It has some problems like face occlusion, and non uniform illumination. We use Neural Network to detect face in the Live video stream. Tensor flow is also used in this system . In existing they use Adaboost algorithm, we are using mob net   Architecture model in our proposed system.We will overcome all these problems in this report.

TITLE : Multi-Stage   Architecture for Face Mask Detection

This system consists of a dual-stage architecture capable of detecting masked and unmasked faces and can be integrated with pre-installed CCTV cameras. This will help track safety violations, promote the use of face masks and ensure a safe working environment. Datasets were collected from public domain along with some data scraped from the internet. They use only pretrained datasets for detection. We can use any cameras to detect faces. It will be very useful for society and for peoples to prevent them from virus transmission. Here we use live video detection using open cv (python library)

TITLE : Real time face mask recognition with alarm system using deep learning

This process gives a precise and speedily results for facemask detection. Raspberry pi based real time face mask recognition that captures the facial image. This system uses the architectural features of VGG-16 as the foundation network for face recognition. Deep learning techniques are applied to construct a classifier that will collect image of a person wearing a face mask and no masks. Our proposed study are uses the architectural features of   as the foundation network for face detection .It shows accuracy in detecting person wearing a face mask and not wearing a face mask .This study presence a useful tool in fighting the spread of covid 19 virus .

Combined Packages

A. OpenCV

OpenCV (Open Source Computer Vision Library), an open source computer source and ML library, is used for face recognition and visualization, visualization, group movement in recording, tracking modules, eye tracking, camera action, red-eyed photography using the flash, find comparable images from the image database, see the landscape and set markers to attach to the larger reality and so on [20]. The proposed method uses these OpenCV features to resize and color change data images.

B. Scikit-learn

Scikit-learn is largely written in Python, and uses NumPy extensively for high-performance linear algebra and array operations. Furthermore, some core algorithms are written in Cython to improve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR. In such cases, extending these methods with Python may not be possible.

Scikit-learn integrates well with many other Python libraries, such as Matplotlib and plotly for plotting, Numpy for array vectorization, Panda data frames, Scipy, and many more.

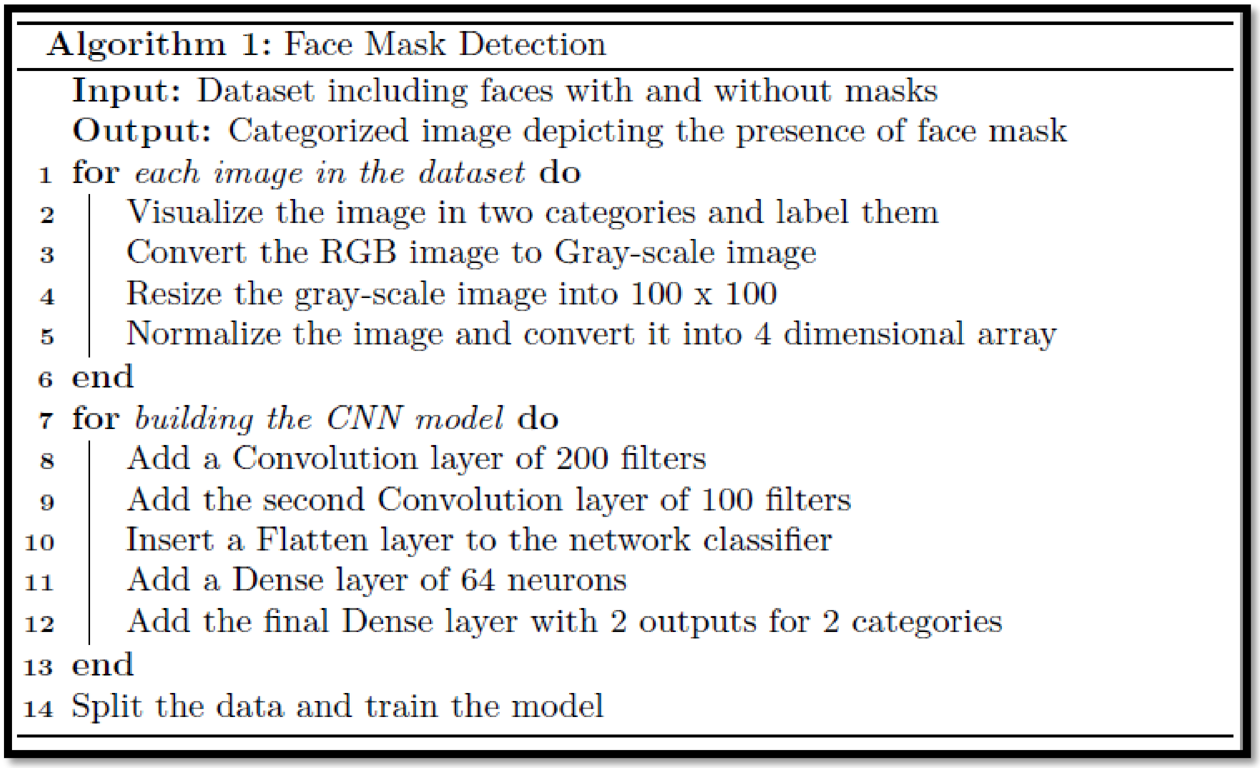
RELATED WORK

In face detection method, a face is detected from an image that has several attributes in it. According to , research into face detection requires expression recognition, face tracking, and pose estimation. Given a solitary image, the challenge is to identify the face from the picture. Face detection is a difficult errand because the faces change in size, shape, color, etc and they are not immutable. It becomes a laborious job for opaque image impeded by some other thing not confronting camera, and so forth. Authors in think occlusive face detection comes with two major challenges: 1) unavailability of sizably voluminous datasets containing both masked and unmasked faces, and 2) exclusion of facial expression in the covered area. Utilizing the locally linear embedding (LLE) algorithm and the dictionaries trained on an immensely colossal pool of masked faces, synthesized mundane faces, several mislaid expressions can be recuperated and the ascendancy of facial cues can be mitigated to great extent. According to the work reported in , convolutional neural network (CNNs) in computer vision comes with a strict constraint regarding the size of the input image. The prevalent practice reconfigures the images before fitting them into the network to surmount the inhibition.

Here the main challenge of the task is to detect the face from the image correctly and then identify if it has a mask on it or not. In order to perform surveillance tasks, the proposed method should also detect a face along with a mask in motion.

THE PROPOSED METHOD

The proposed method consists of a cascade classifier and a pre-trained CNN which contains two 2D convolution layers connected to layers of dense neurons. The algorithm for face mask detection is as follows:



```
Algorithm 1: Face Mask Detection
Input: Dataset including faces with and without masks
Output: Categorized image depicting the presence of face mask
for each image in the dataset do
    Visualize the image in two categories and label them
    Convert the RGB image to Gray-scale image
    Resize the gray-scale image into 100 x 100
    Normalize the image and convert it into 4 dimensional array
end
for building the CNN model do
    Add a Convolution layer of 200 filters
    Add the second Convolution layer of 100 filters
    Insert a Flatten layer to the network classifier
    Add a Dense layer of 64 neurons
    Add the final Dense layer with 2 outputs for 2 categories
end
Split the data and train the model
```

A. Data Processing

Data preprocessing involves conversion of data from a given format to much more user friendly, desired and meaningful format. It can be in any form like tables, images, videos, graphs, etc. These organized information fit in with an information model or composition and captures relationship between different entities . The proposed method deals with image and video data using Numpy and OpenCV.

a) Data Visualization: Data visualization is the process of transforming abstract data to meaningful representations using knowledge communication and insight discovery through encodings. It is helpful to study a particular pattern in the dataset .

The total number of images in the dataset is visualized in both categories – ‘with mask’ and ‘without mask’. The statement categories=os.listdir(data path) categorizes the list of directories in the specified data path. The variable categories now looks like: [‘with mask’, ‘without mask’] Then to find the number of labels, we need to distinguish those categories using labels=[i for i in range(len(categories))]. It sets the labels as: [0, 1]

Now, each category is mapped to its respective label using label dict=dict(zip(categories,labels)) which at first returns an iterator of tuples in the form of zip object where the items in each passed iterator is paired together consequently. The mapped variable label dict looks like: f‘with mask’: 0, ‘without mask’: 1g

b) Conversion of RGB image to Gray image: Modern descriptor-based image recognition systems regularly work on grayscale images, without elaborating the method used to convert from color-to-grayscale. This is because the colorto-grayscale method is of little consequence when using robust descriptors. Introducing nonessential information could increase the size of training data required to achieve good performance. As grayscale rationalizes the algorithm and
diminishes the computational requisites, it is utilized for extracting descriptors instead of working on color images instantaneously .

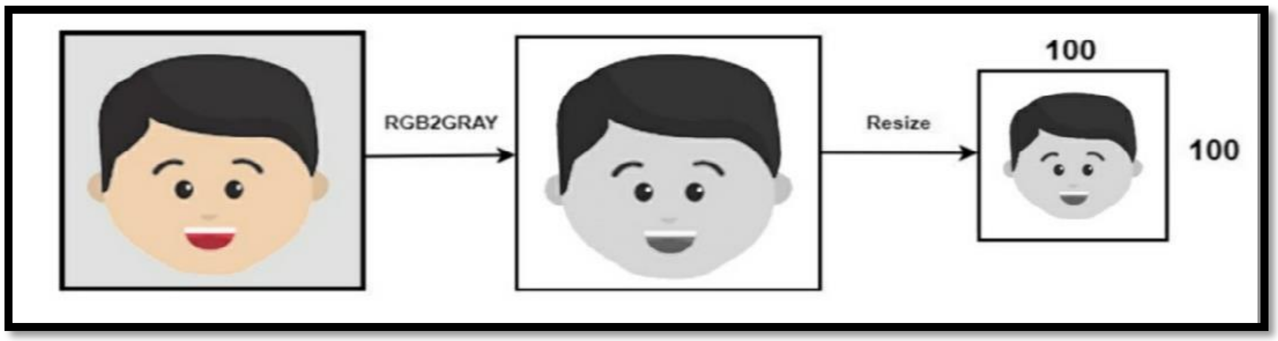


Fig. Conversion of a RGB image to a Gray Scale image of 100 x 100 size

We use the function cv2.cvtColor(input image, flag) for changing the color space. Here flag determines the type of conversion . In this case, the flag cv2.COLOR BGR2GRAY is used for gray conversion. Deep CNNs require a fixed-size input image. Therefore we need a fixed common size for all the images in the dataset. Using cv2.resize() the gray scale image is resized into 100 x 100.

c) Image Reshaping: The input during relegation of an image is a three-dimensional tensor, where each channel has a prominent unique pixel. All the images must have identically tantamount size corresponding to 3D feature tensor. However, neither images are customarily coextensive nor their corresponding feature tensors . Most CNNs can only accept fine-tuned images. This engenders several problems throughout data collection and implementation of model. However, reconfiguring the input images before augmenting them into the network can help to surmount this constraint.

The images are normalized to converge the pixel range between 0 and 1. Then they are converted to 4 dimensional arrays using data=np.reshape(data,(data.shape[0], img size,img size,1)) where 1 indicates the Grayscale image. As, the final layer of the neural network has 2 outputs – with mask and without mask i.e. it has categorical representation, the data is converted to categorical labels.

B. Training of Model

a) Building the model using CNN architecture: CNN has become ascendant in miscellaneous computer vision tasks The current method makes use of Sequential CNN. The First Convolution layer is followed by Rectified Linear Unit (ReLU) and MaxPooling layers. The Convolution layer learns from 200 filters. Kernel size is set to 3 x 3 which specifies the height and width of the 2D convolution window.

As the model should be aware of the shape of the input expected, the first layer in the model needs to be provided with information about input shape. Following layers can perform instinctive shape reckoning . In this case, input shape is specified as data.shape[1:] which returns the dimensions of the data array from index 1. Default padding is “valid” where the spatial dimensions are sanctioned to truncate and the input volume is non-zero padded. The activation parameter to the Conv2D class is set as “relu”. It represents an approximately linear function that possesses all the assets of linear models that can easily be optimized with gradient-descent methods. Considering the performance and generalization in deep learning, it is better compared to other activation functions. Max Pooling is used to reduce the spatial dimensions of the output volume. Pool size is set to 3 x 3 and the resulting output has a shape (number of rows or columns) of: shape of output = (input shape - pool size + 1) / strides), where strides has default value (1,1) [15].

As shown in fig, 4, the second Convolution layer has 100 filters and Kernel size is set to 3 x 3. It is followed by ReLu and MaxPooling layers. To insert the data into CNN, the long vector of input is passed through a Flatten layer which transforms matrix of features into a vector that can be fed into a fully connected neural network classifier. To reduce overfitting a Dropout layer with a 50% chance of setting inputs to zero is added to the model. Then a Dense layer of 64 neurons with a ReLu activation function is added. The final layer (Dense) with two outputs for two categories uses the Softmax activation function.

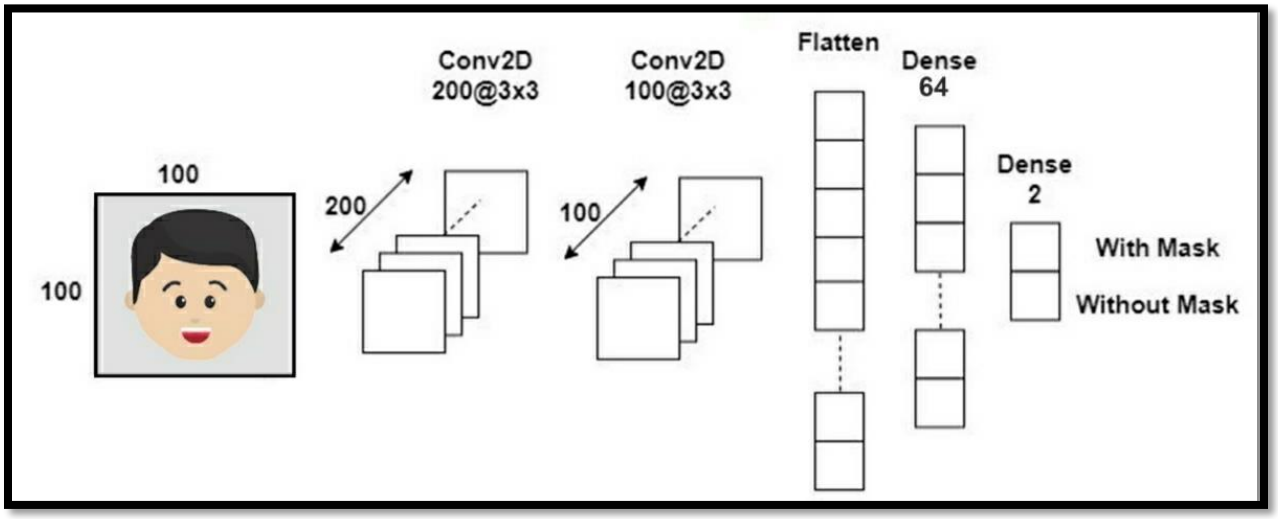


Fig. Convolutional Neural Network architecture

The learning process needs to be configured first with the compile method . Here “adam” optimizer is used. categorical crossentropy which is also known as multiclass log loss is used as a loss function (the objective that the model tries to minimize). As the problem is a classification problem, metrics is set to “accuracy”.
b) Splitting the data and training the CNN model: After setting the blueprint to analyze the data, the model needs to be trained using a specific dataset and then to be tested against a different dataset. A proper model and optimized train test split help to produce accurate results while making a prediction. The test size is set to 0.1 i.e. 90% data of the dataset undergoes training and the rest 10% goes for testing purposes. The validation loss is monitored using ModelCheckpoint. Next, the images in the training set and the test set are fitted to the Sequential model. Here, 20% of the training data is used as validation data. The model is trained for 20 epochs (iterations) which maintains a trade-off between accuracy and chances of overfitting.

METHODOLOGY

System design

The major requirement for implementing this project using python programming language along with Deep learning

,Machine learning , Computer vision and also with python libraries. The architecture consists of Mobile Net as the backbone, it can be used for high and low computation scenarios. We are using   Algorithm in our proposed system.

Implementation:

We have four modules

Datasets Collecting : We collect no of data sets with face mask and without masks. we can get high accuracy depends on collecting the number of images .

Datasets Extracting: We can extract the features using mobile net v2 of mask and no mask sets

Models Training: We will train the the model using open cv,keras (python library).

Facemask Detection : We can detect Pre processing image and also detect via live video . If people wear mask, it will permit them, if not then it will give the buzzer to wear mask to prevent them from virus transmission.

BENEFITS

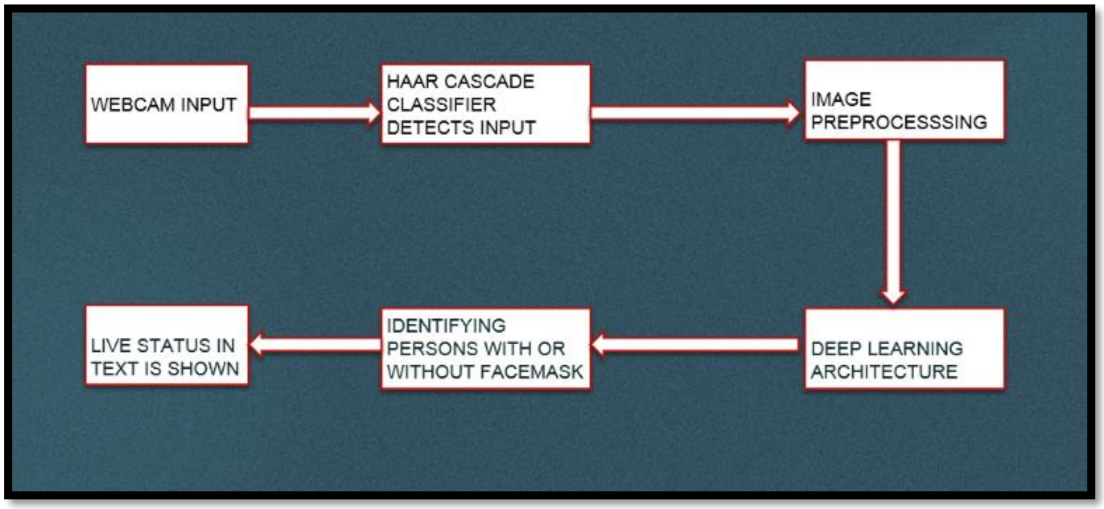
1. Manual Monitoring is very difficult for officers to check whether the peoples are wearing mask or not. So in our technique, We are using web cam to detect peoples faces and to prevent from virus transmission.

2. It has fast and high accuracy

3. This system can be implemented in ATMs, Banks etc

4. We can keep peoples safe from our technique.

Workflow


WEBCAM INPUT
HAAR CASCADE CLASSIFIER DETECTS INPUT
IMAGE PREPROCESSSING
DEEP LEARNING ARCHITECTURE
IDENTIFYING PERSONS WITH OR WITHOUT FACEMASK
LIVE STATUS IN TEXT IS SHOWN

Workflow diagram

Face Mask Detection Algorithm

The Viola-Jones Algorithm

Developed in 2001 by Paul Viola and Michael Jones, the Viola-Jones algorithm is an object-recognition framework that allows the detection of image features in real-time.

Despite being an outdated framework, Viola-Jones is quite powerful and its application has proven to be exceptionally notable in real-time face detection.

How it works

There are 2 stages in the Viola-Jones Algorithm:

1. Training
2. Detection

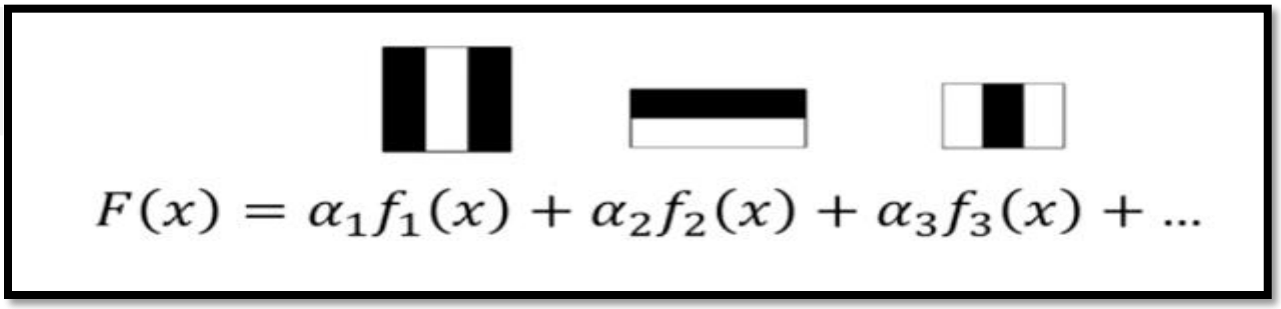
1.Training

The algorithm shrinks the image to 24 x 24 and looks for the trained features within the image. It needs a lot of facial image data to be able to see features in the different and varying forms. That's why we need to supply lots of facial image data to the algorithm so it can be trained. Viola and Jones fed their algorithm 4,960 images (each manually labeled). For some images, you can feed the mirror image of a particular image, which would be brand new information for a computer.

You would also need to supply the algorithm non-facial images so it can differentiate between the two classes. Viola and Jones supplied their algorithm 9,544 non-facial images. Within these, some images may look similar to features in a face, but the algorithm will understand which features are more likely to be on a face and which features would obviously not be on a face.

Adaptive Boosting (AdaBoost)

The algorithm learns from the images we supply it and is able to determine the false positives and true negatives in the data, allowing it to be more accurate. We would get a highly accurate model once we have looked at all possible positions and combinations of those features. Training can be super extensive because of all the different possibilities and combinations you would have to check for every single frame or image.



$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots$$

Let’s say we have an equation for our features that determines the success rate (as seen in the image), with f1, f2 and f3 as the features and a1, a2, a3 as the respective weights of the features. Each of the features is known as a **weak classifier.** The left side of the equation F(x) is called a **strong classifier.** Since one weak classifier may not be as good, we get a strong classifier when we have a combination of two or three weak classifiers. As you keep adding, it gets stronger and stronger. This is called an **ensemble.**

2. Detection

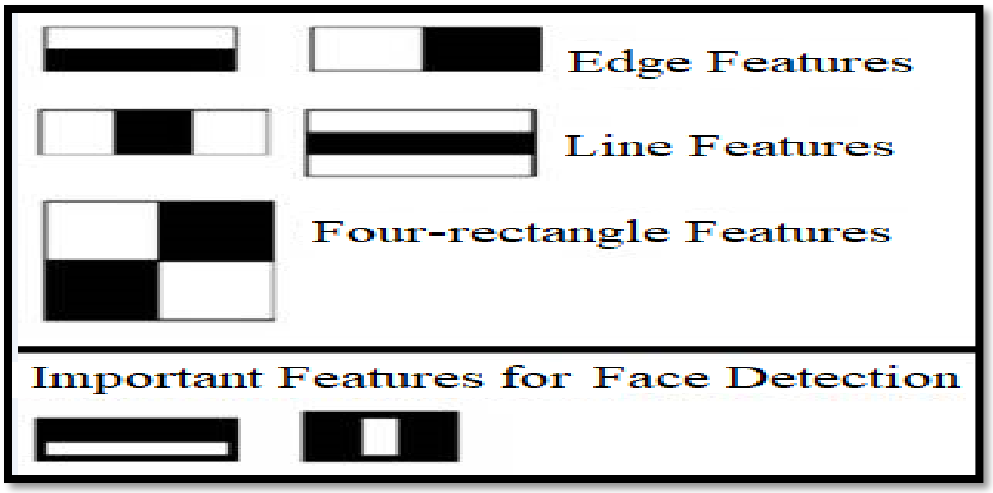
Viola-Jones was designed for frontal faces, so it is able to detect frontal the best rather than faces looking sideways, upwards or downwards. Before detecting a face, the image is converted into grayscale, since it is easier to work with and there's lesser data to process. The Viola-Jones algorithm first detects the face on the grayscale image and then finds the location on the colored image.

Viola-Jones outlines a box and searches for a face within the box. It is essentially searching for these haar-like features, which will be explained later. The box moves a step to the right after going through every tile in the picture. In this case, I've used a large box size and taken large steps for demonstration, but in general, you can change the box size and step size according to your needs.

With smaller steps, a number of boxes detect face-like features (Haar-like features) and the data of all of those boxes put together, helps the algorithm determine where the face is.

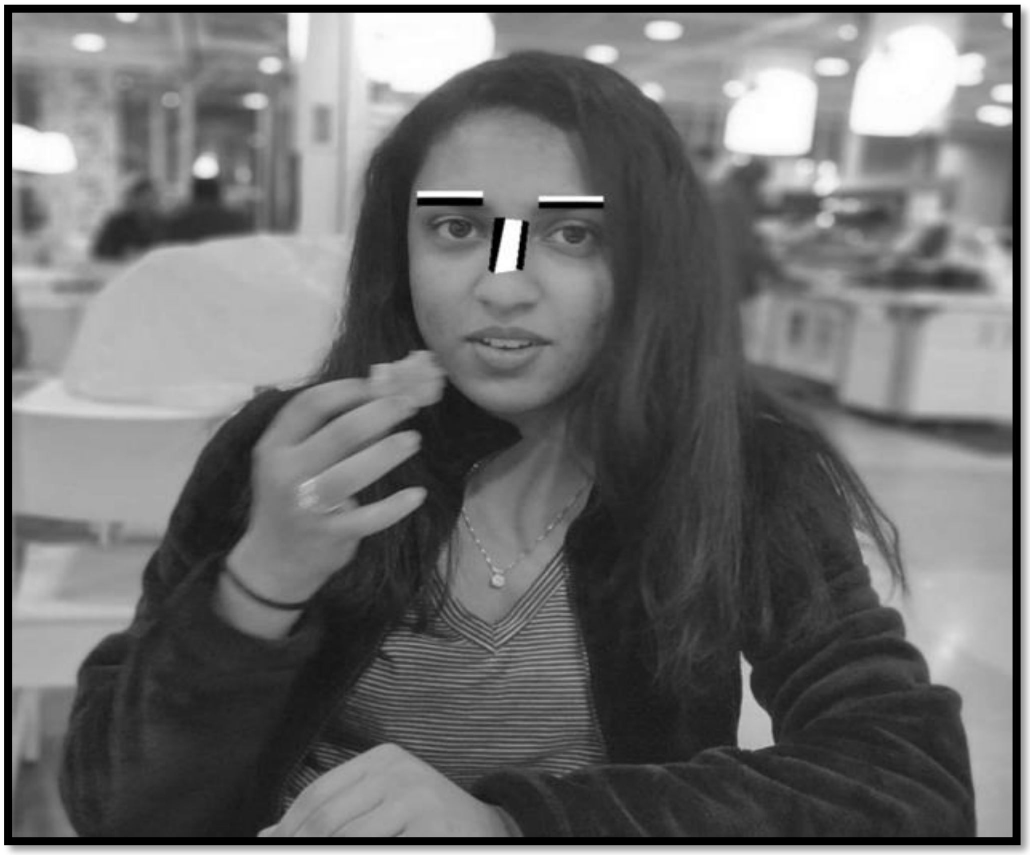
Haar-like Features

Haar-like features are named after Alfred Haar, a Hungarian mathematician in the 19th century who developed the concept of Haar wavelets (kind of like the ancestor of haar-like features). The features below show a box with a light side and a dark side, which is how the machine determines what the feature is. Sometimes one side will be lighter than the other, as in an edge of an eyebrow. Sometimes the middle portion may be shinier than the surrounding boxes, which can be interpreted as a nose.



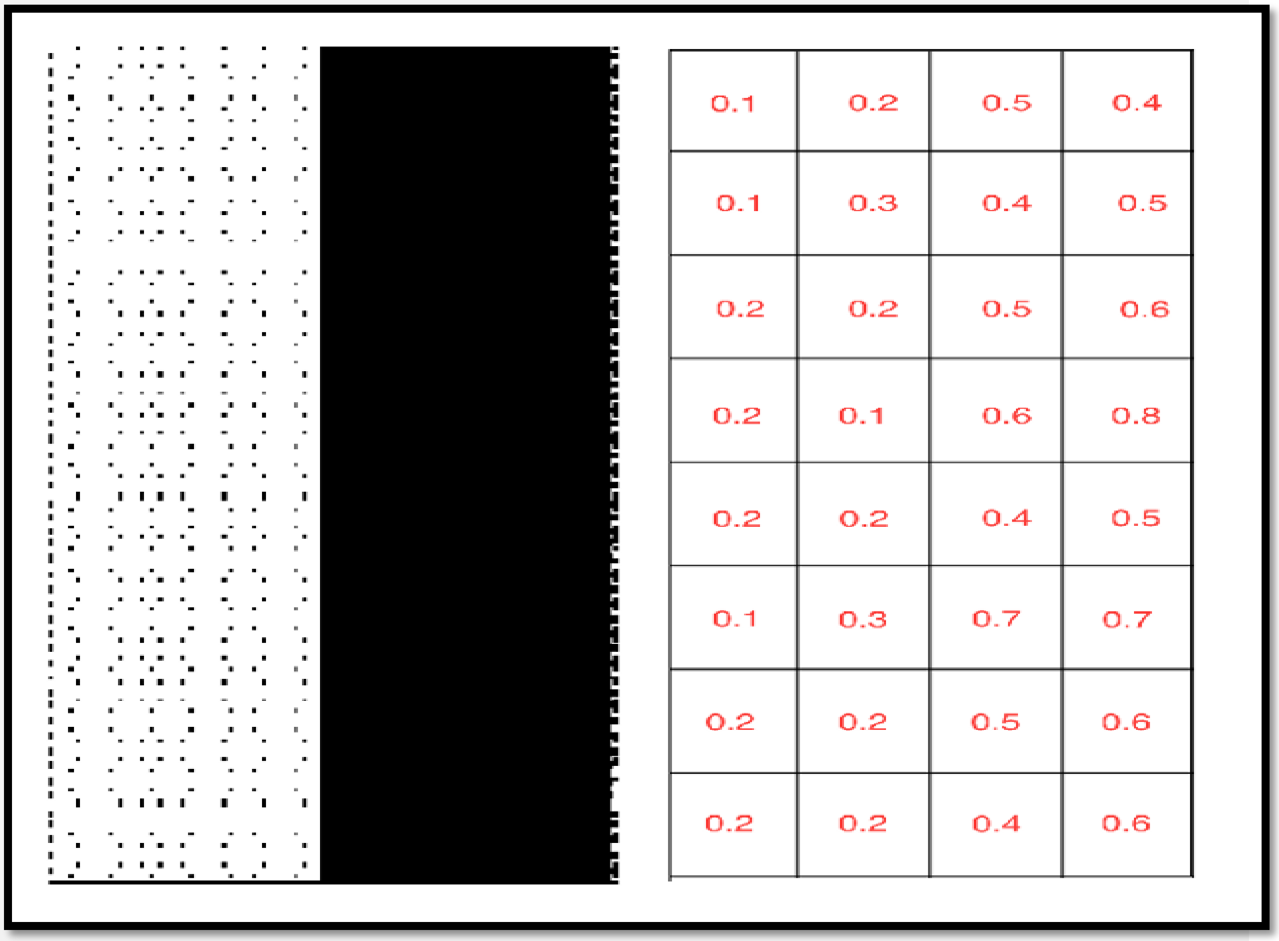
There are 3 types of Haar-like features that Viola and Jones identified in their research:

- Edge features
- Line-features
- Four-sided features



These features help the machine understand what the image is. Imagine what the edge of a table would look like on a b&w image. One side will be lighter than the other, creating that edge like b&w feature as you can see in the picture above.

In the two important features for Face Detection, the horizontal and the vertical features describe what eyebrows and the nose, respectively, look like to the machine. Additionally, when the images are inspected, each feature has a value of its own. It’s quite easy to calculate: Subtract White area from the Black area. For example, look at the image below.



0.1	0.2	0.5	0.4
0.1	0.3	0.4	0.5
0.2	0.2	0.5	0.6
0.2	0.1	0.6	0.8
0.2	0.2	0.4	0.5
0.1	0.3	0.7	0.7
0.2	0.2	0.5	0.6
0.2	0.2	0.4	0.6

Imagine our haar-like feature was converted into a grid. Each square represents a pixel. For demonstration, I chose a 4 x 8 grid, but in reality, there would be many more pixels and thus a much larger grid for a certain feature. The numbers in the boxes represent the darkness of the features. The higher it is, the darker the pixel. Thus, you can see the numbers are higher on the right side than on the left side. Now if you add up the numbers on the two left-sided (white) columns, and subtract it from the sum of the right-sided columns, you will get the value of the particular feature.

So in this case, the value of our feature is →

(0.5 + 0.4 + 0.5 +0.6 + 0.4 + 0.7 + 0.5 + 0.4 +

0.4 + 0.5 + 0.6 + 0.8 + 0.5 + 0.7 + 0.6 + 0.6) -

(0.1 + 0.1 + 0.2 + 0. 2+ 0.2 + 0.1 + 0.2 + 0.2 +

0.2 + 0.3 + 0.2 + 0.1 + 0.2 + 0.3 + 0.2 + 0.2)

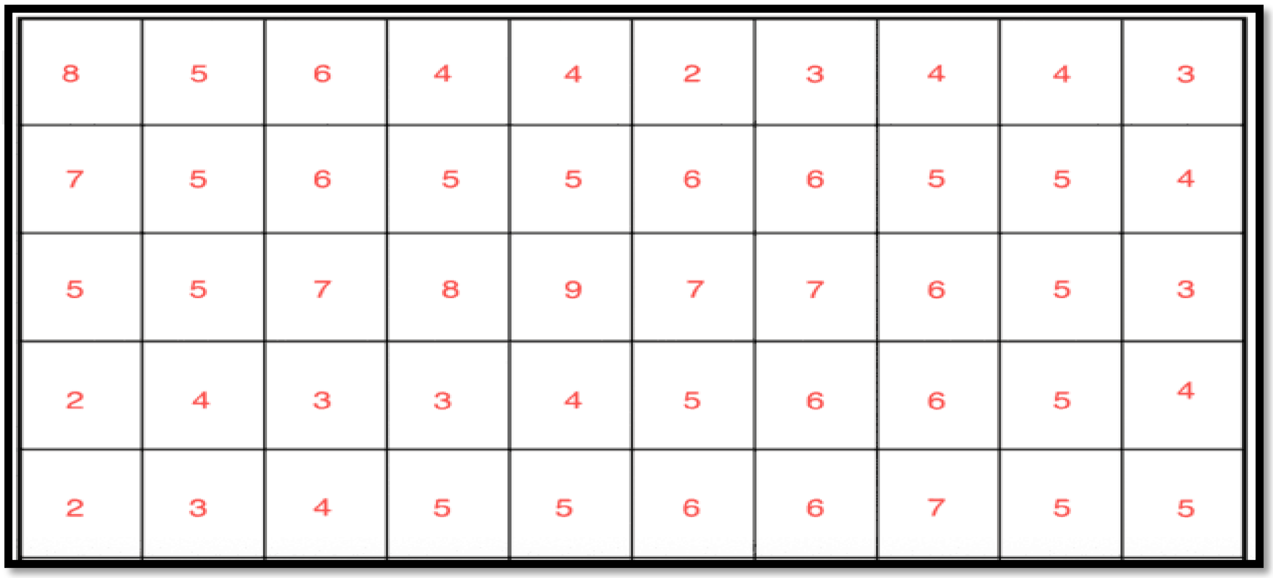
B - W= 8.7 - 3

= 5.7

Integral Image

So in the last section, we calculated the value of a feature. In reality, these calculations can be very intensive since the number of pixels would be much greater within a large feature.

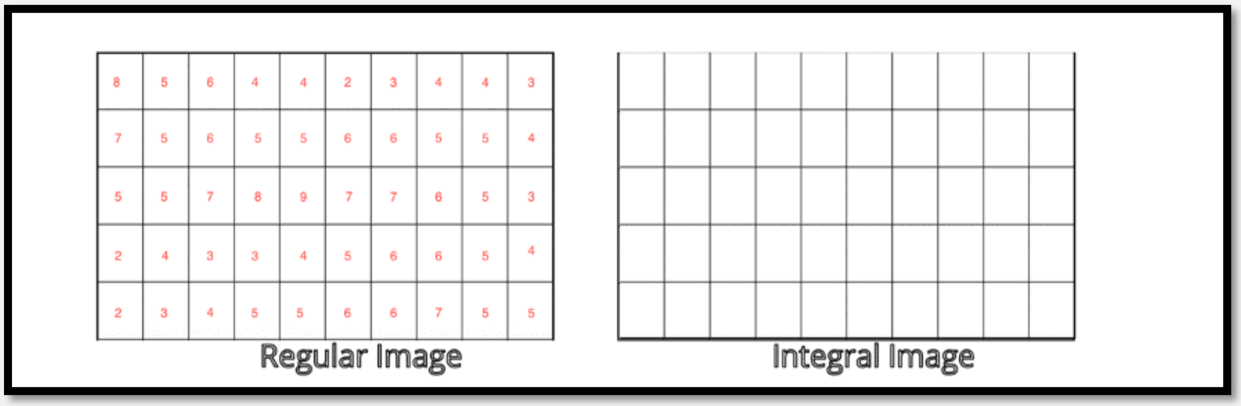
The integral image plays its part in allowing us to perform these intensive calculations quickly so we can understand whether a feature of a number of features fit the criteria.



8	5	6	4	4	2	3	4	4	3
7	5	6	5	5	6	6	5	5	4
5	5	7	8	9	7	7	6	5	3
2	4	3	3	4	5	6	6	5	4
2	3	4	5	5	6	6	7	5	5

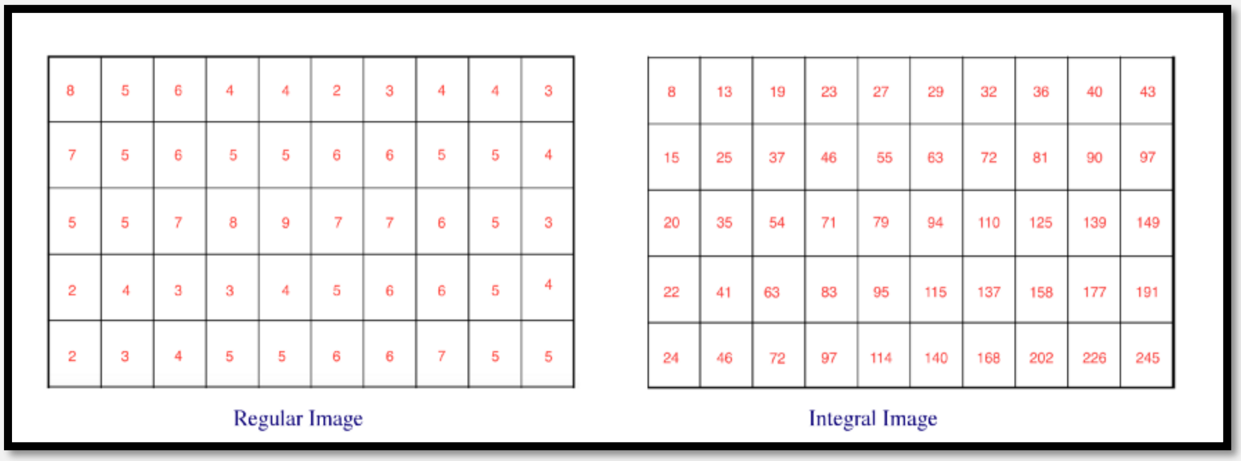
Regular Image

To calculate the value of a single box in the integral image, we take the sum of all the boxes to its left. The image below shows an example:


Regular Image
Integral Image

8	5	6	4	4	2	3	4	4	3
7	5	6	5	5	6	6	5	5	4
5	5	7	8	9	7	7	6	5	3
2	4	3	3	4	5	6	6	5	4
2	3	4	5	5	6	6	7	5	5

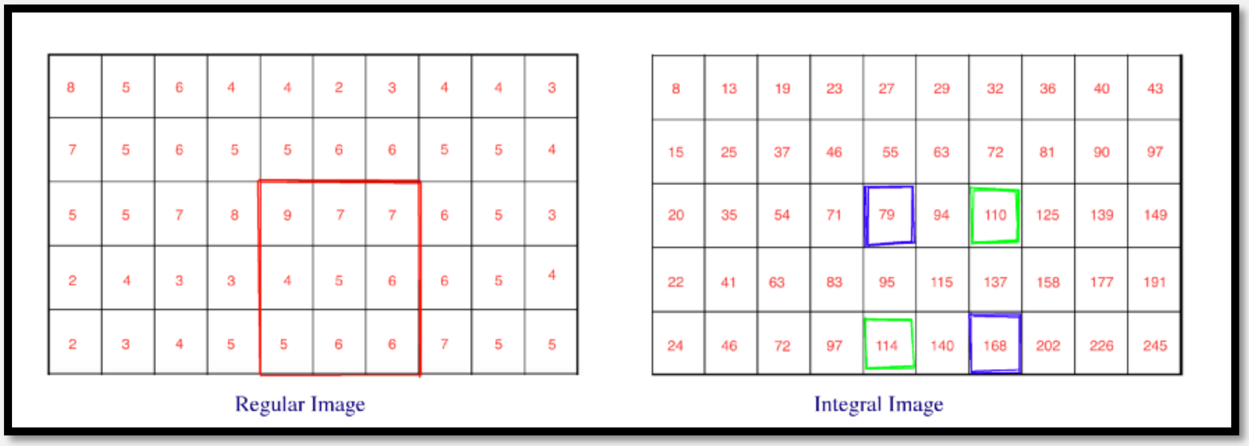
The green box in the integral image is calculated as the sum of the highlighted area in the regular image. If we did this for every box, we would have a sequence going through the grid and it may look something like the image below.


Regular Image
Integral Image

8	5	6	4	4	2	3	4	4	3
7	5	6	5	5	6	6	5	5	4
5	5	7	8	9	7	7	6	5	3
2	4	3	3	4	5	6	6	5	4
2	3	4	5	5	6	6	7	5	5

8	13	19	23	27	29	32	36	40	43
15	25	37	46	55	63	72	81	90	97
20	35	54	71	79	94	110	125	139	149
22	41	63	83	95	115	137	158	177	191
24	46	72	97	114	140	168	202	226	245

Let’s look at the value of the example I had picked earlier:



8	5	6	4	4	2	3	4	4	3
7	5	6	5	5	6	6	5	5	4
5	5	7	8	9	7	7	6	5	3
2	4	3	3	4	5	6	6	5	4
2	3	4	5	5	6	6	7	5	5

Regular Image

8	13	19	23	27	29	32	36	40	43
15	25	37	46	55	63	72	81	90	97
20	35	54	71	79	94	110	125	139	149
22	41	63	83	95	115	137	158	177	191
24	46	72	97	114	140	168	202	226	245

Integral Image

All we have to do is look at the 4 corners of our feature, and add the purples, subtract the greens.

→ **168–114 + 79–110 = 23**

CHAPTER-3

IMPLEMENTATION

In this , we are gone through to use OpenCV to do real-time face detection from a live stream via our webcam.

As videos are basically made up of frames, which are still images. We perform the face detection for each frame in a video. So when it comes to detecting a face in still image and detecting a face in a real-time video stream, there is not much difference between them.

We will be using Haar Cascade algorithm, also known as Voila-Jones algorithm to detect faces. It is basically a machine learning object detection algorithm which is used to identify objects in an image or video. In OpenCV, we have several trained Haar Cascade models which are saved as XML files. Instead of creating and training the model from scratch, we use this file. We are going to use "haarcascade_frontalface_alt2.xml" file in this project. Now let us start coding this up

The first step is to find the path to the "haarcascade_frontalface_alt2.xml" file. We do this by using the os module of Python language.

```
haar_data = cv2.CascadeClassifier(r'C:\Users\SHASHANK\data.xml')
```

```
haar_data.detectMultiScale(img)
```

The next step is to load our classifier. The path to the above XML file goes as an argument to CascadeClassifier() method of OpenCV.

```
faceCascade = cv2.CascadeClassifier(cascPath)
```

After loading the classifier, let us open the webcam using this simple OpenCV one-liner code

```
video_capture = cv2.VideoCapture(0)
```

Next, we need to get the frames from the webcam stream, we do this using the read() function. We use it in infinite loop to get all the frames until the time we want to close the stream.

```
while True:

    # Capture frame-by-frame

    ret, frame = video_capture.read()
```

The read() function returns:

1. The actual video frame read (one frame on each loop)
2. A return code

The return code tells us if we have run out of frames, which will happen if we are reading from a file. This doesn’t matter when reading from the webcam since we can record forever, so we will ignore it.

For this specific classifier to work, we need to convert the frame into greyscale.

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

The faceCascade object has a method detectMultiScale(), which receives a frame(image) as an argument and runs the classifier cascade over the image. The term Multi Scale indicates that the algorithm looks at subregions of the image in multiple scales, to detect faces of varying sizes.

```
faces = faceCascade.detectMultiScale(gray,

scaleFactor=1.1,

minNeighbors=5,

minSize=(60, 60),

flags=cv2.CASCADE_SCALE_IMAGE)
```

Let us go through these arguments of this function:

- Scale Factor – Parameter specifying how much the image size is reduced at each image scale. By rescaling the input image, you can resize a larger face to a smaller one, making it detectable by the algorithm. 1.05 is a good possible value for this, which means you use a small step for resizing, i.e. reduce the size by 5%, you increase the chance of a matching size with the model for detection is found.
- Min Neighbors – Parameter specifying how many neighbours each candidate rectangle should have to retain it. This parameter will affect the quality of the detected faces. Higher value results in fewer detections but with higher quality. 3~6 is a good value for it.
- flags –Mode of operation

- min Size – Minimum possible object size. Objects smaller than that are ignored.

The variable faces now contain all the detections for the target image. Detections are saved as pixel coordinates. Each detection is defined by its top-left corner coordinates and width and height of the rectangle that encompasses the detected face.

To show the detected face, we will draw a rectangle over it. OpenCV's rectangle() draws rectangles over images, and it needs to know the pixel coordinates of the top-left and bottom-right corner. The coordinates indicate the row and column of pixels in the image. We can easily get these coordinates from the variable face.

```
for (x,y,w,h) in faces:

cv2.rectangle(frame, (x, y), (x + w, y + h),(0,255,0), 2)
```

rectangle() accepts the following arguments:

- The original image
- The coordinates of the top-left point of the detection
- The coordinates of the bottom-right point of the detection
- The colour of the rectangle (a tuple that defines the amount of red, green, and blue (0-255)).In our case, we set as green just keeping the green component as 255 and rest as zero.
- The thickness of the rectangle lines

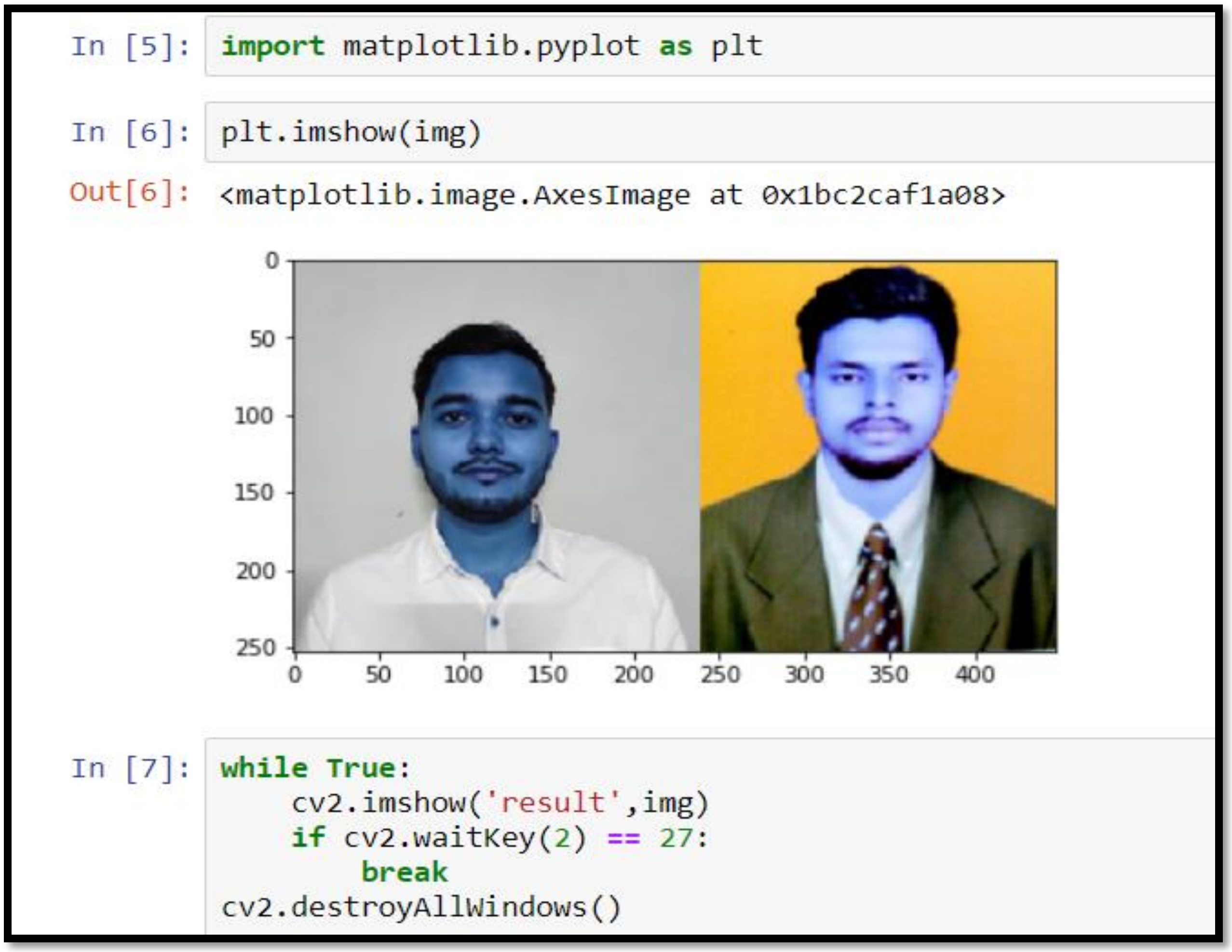
Next, we just display the resulting frame and also set a way to exit this infinite loop and close the video feed. By pressing the 'q' key, we can exit the script here

```
 cv2.imshow('Video', frame)
```

```
    if cv2.waitKey(1) & 0xFF == ord('q'):

        break
```

The next two lines are just to clean up and release the picture.

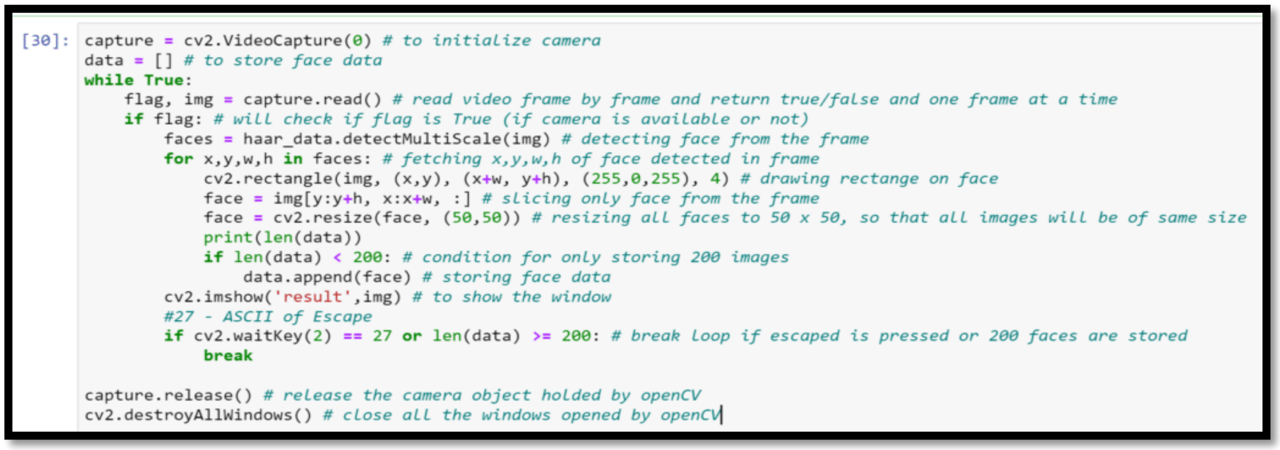
```
video_capture.release()

cv2.destroyAllWindows()
```



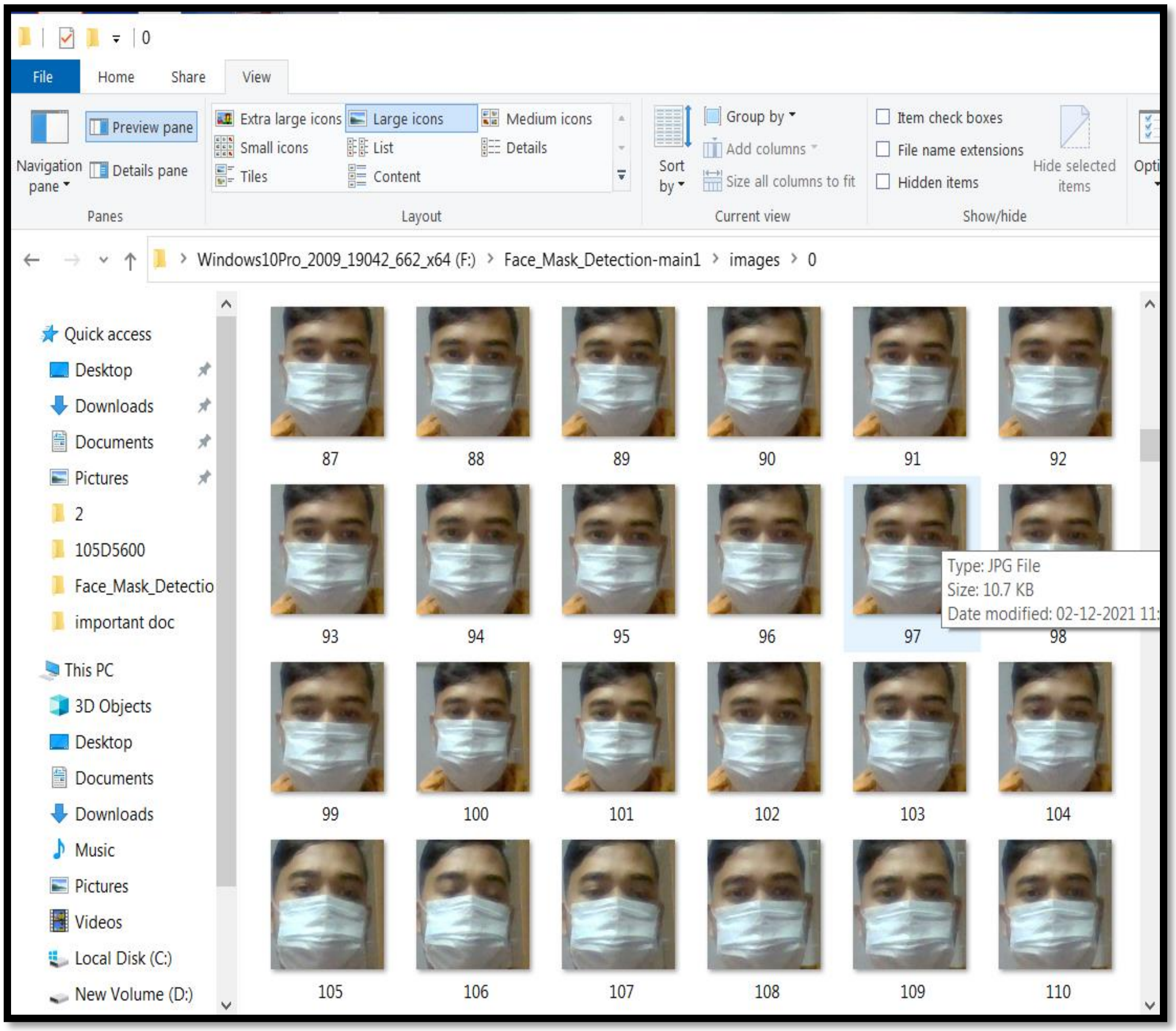
```
In [5]: import matplotlib.pyplot as plt

In [6]: plt.imshow(img)

Out[6]: <matplotlib.image.AxesImage at 0x1bc2caf1a08>
```

```
In [7]: while True:
            cv2.imshow('result',img)
            if cv2.waitKey(2) == 27:
                break
        cv2.destroyAllWindows()
```

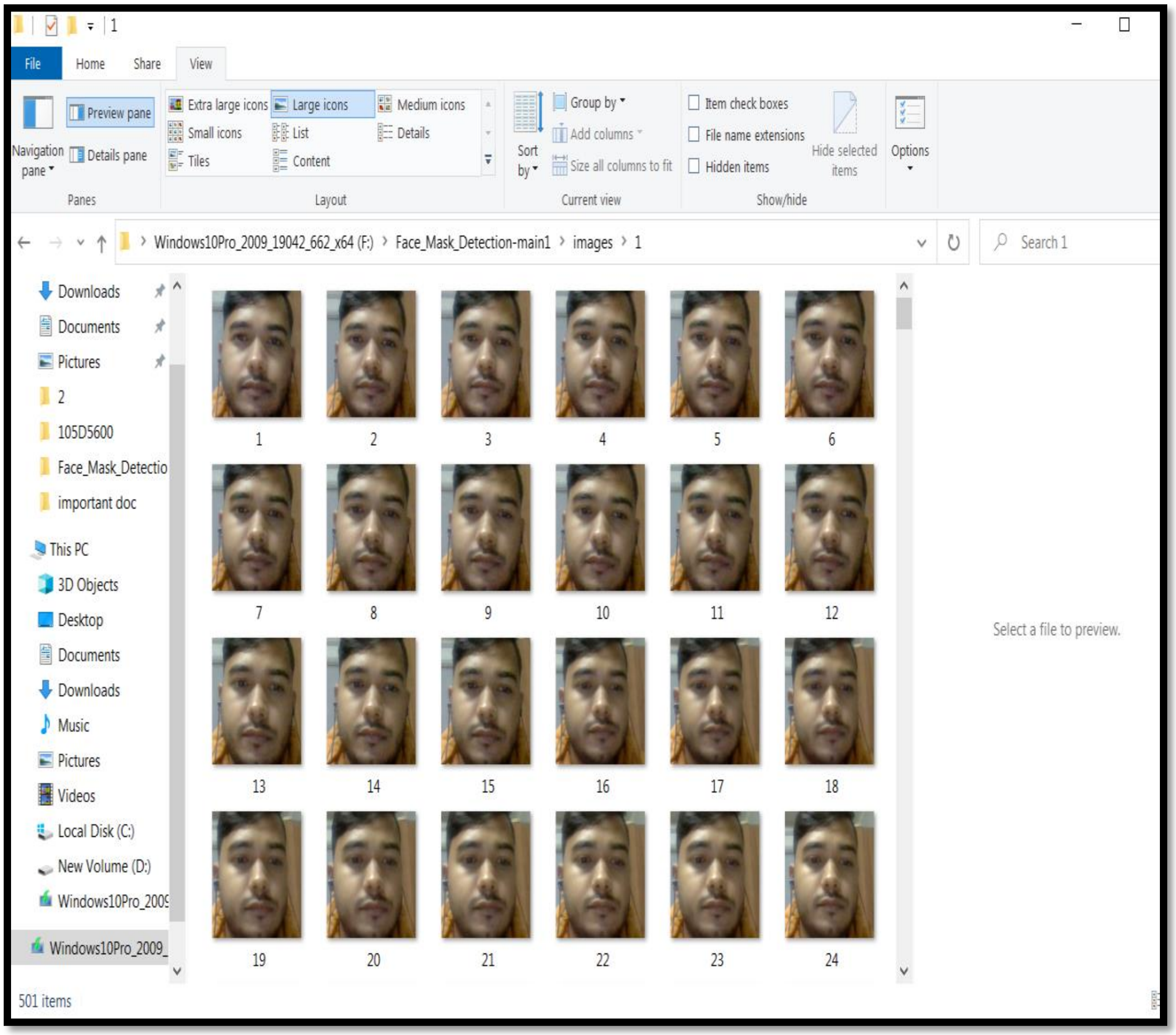
So first we need to collect data and we are going to collect data using our own camera. Here is the complete code to perform face detection using camera and storing face data only:



```
[30]: capture = cv2.VideoCapture(0) # to initialize camera
      data = [] # to store face data
      while True:
          flag, img = capture.read() # read video frame by frame and return true/false and one frame at a time
          if flag: # will check if flag is True (if camera is available or not)
              faces = haar_data.detectMultiScale(img) # detecting face from the frame
              for x,y,w,h in faces: # fetching x,y,w,h of face detected in frame
                  cv2.rectangle(img, (x,y), (x+w, y+h), (255,0,255), 4) # drawing rectange on face
                  face = img[y:y+h, x:x+w, :] # slicing only face from the frame
                  face = cv2.resize(face, (50,50)) # resizing all faces to 50 x 50, so that all images will be of same size
                  print(len(data))
                  if len(data) < 200: # condition for only storing 200 images
                      data.append(face) # storing face data
              cv2.imshow('result',img) # to show the window
              #27 - ASCII of Escape
              if cv2.waitKey(2) == 27 or len(data) >= 200: # break loop if escaped is pressed or 200 faces are stored
                  break

      capture.release() # release the camera object holded by openCV
      cv2.destroyAllWindows() # close all the windows opened by openCV
```



Dataset with mask



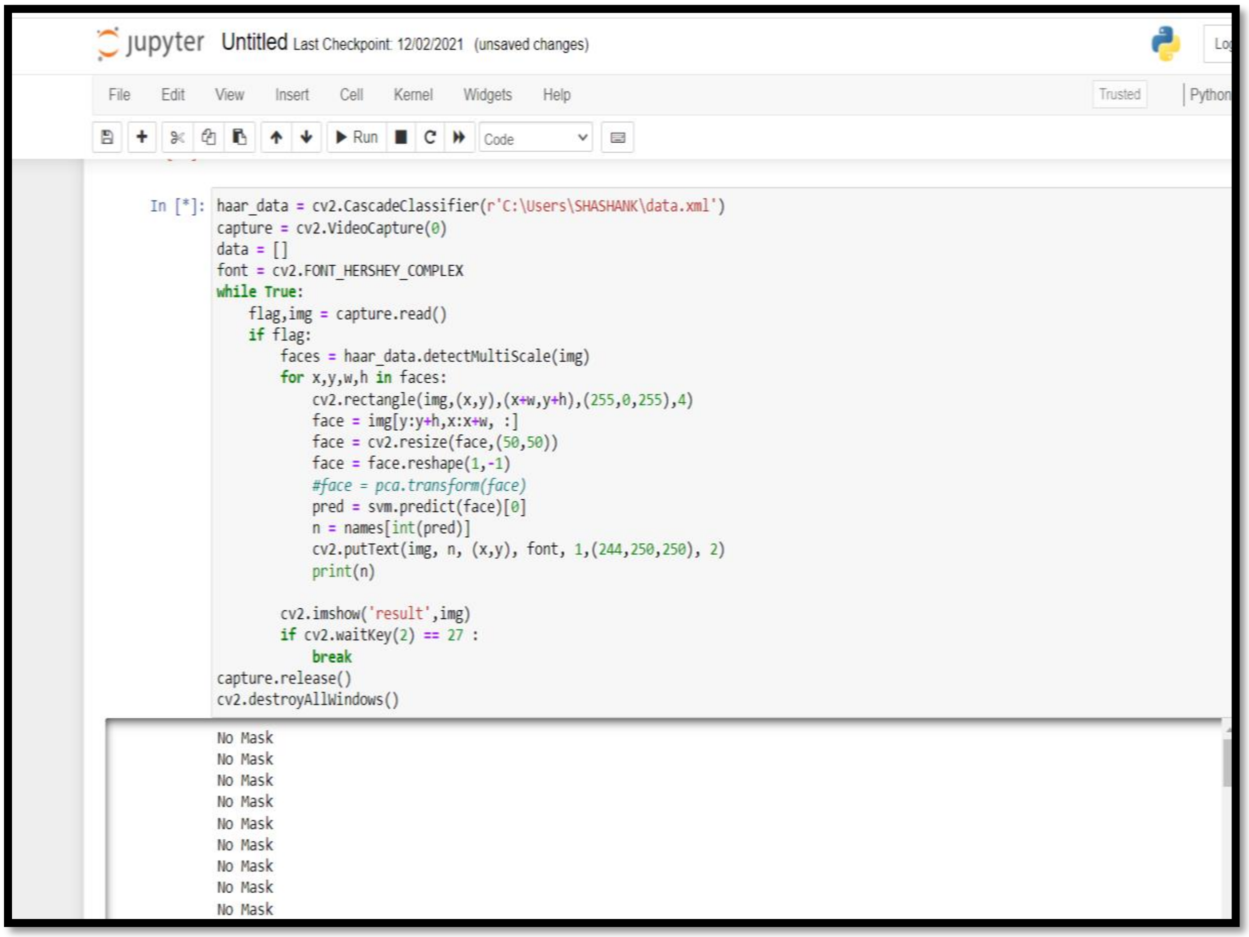
Dataset without mask

CHAPTER-4

EXPERIMENTS AND RESULTS

1.NO-MASK

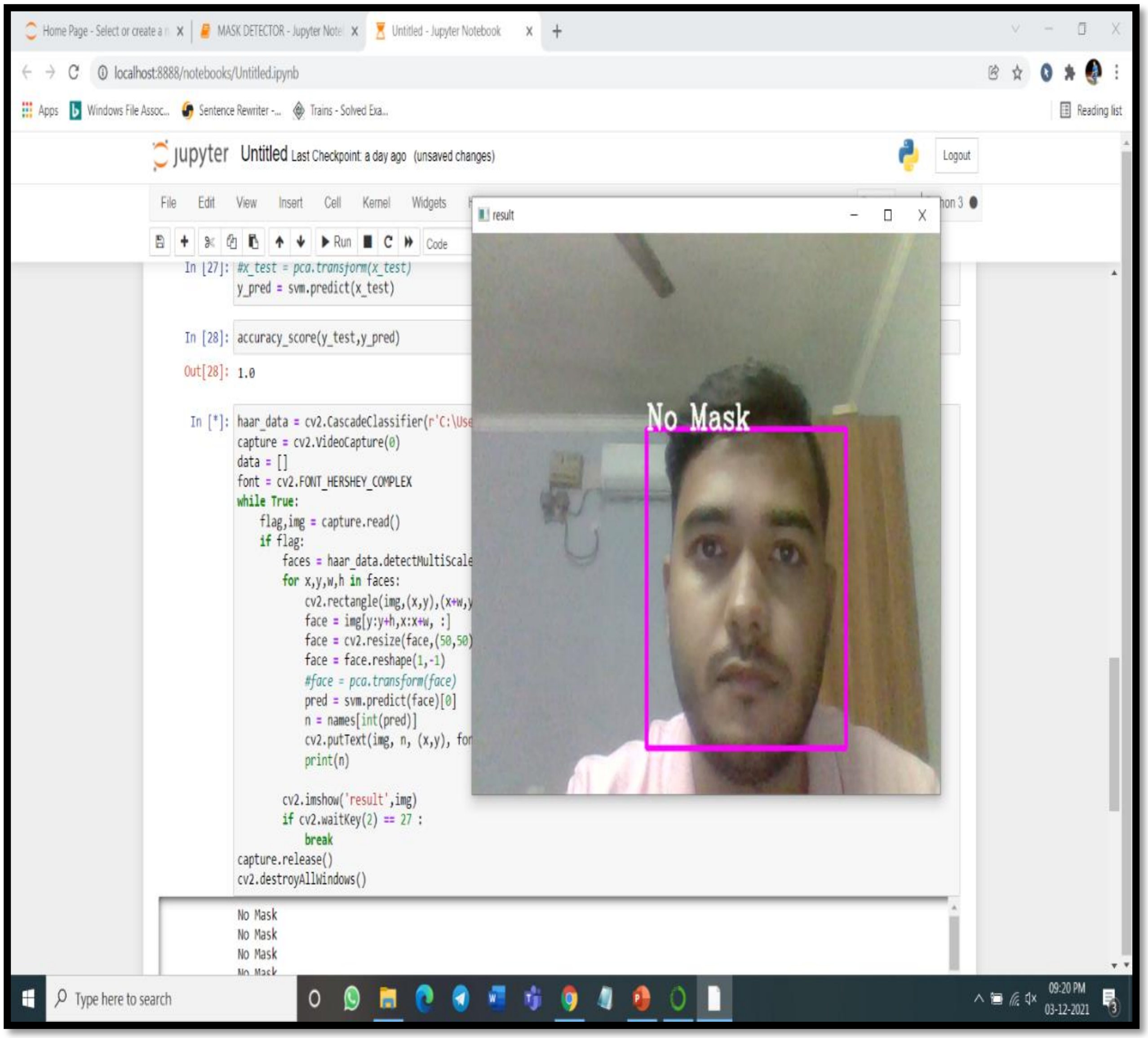
CODE:


jupyter Untitled Last Checkpoint: 12/02/2021 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python
Run
Code
In [*]:

```
haar_data = cv2.CascadeClassifier(r'C:\Users\SHASHANK\data.xml')
capture = cv2.VideoCapture(0)
data = []
font = cv2.FONT_HERSHEY_COMPLEX
while True:
    flag,img = capture.read()
    if flag:
        faces = haar_data.detectMultiScale(img)
        for x,y,w,h in faces:
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,255),4)
            face = img[y:y+h,x:x+w, :]
            face = cv2.resize(face,(50,50))
            face = face.reshape(1,-1)
            #face = pca.transform(face)
            pred = svm.predict(face)[0]
            n = names[int(pred)]
            cv2.putText(img, n, (x,y), font, 1,(244,250,250), 2)
            print(n)

        cv2.imshow('result',img)
        if cv2.waitKey(2) == 27 :
            break
capture.release()
cv2.destroyAllWindows()
```

```
No Mask
No Mask
No Mask
No Mask
No Mask
No Mask
No Mask
No Mask
No Mask
```

RESULT:

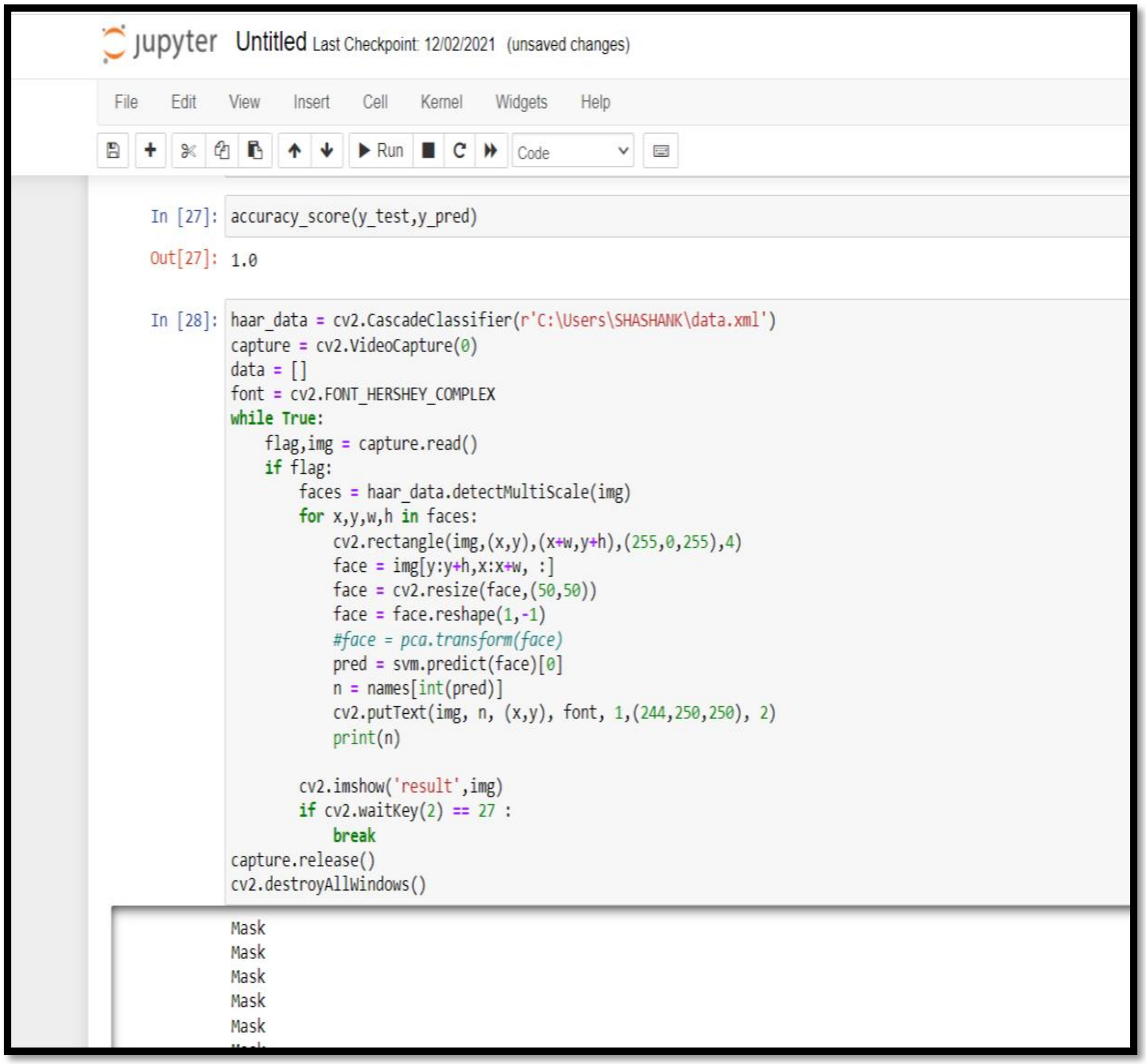

Untitled Last Checkpoint a day ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets
result
No Mask
Type here to search
09:20 PM
03-12-2021

```
In [27]: #x_test = pca.transform(x_test)
         y_pred = svm.predict(x_test)

In [28]: accuracy_score(y_test,y_pred)

Out[28]: 1.0

In [*]:  haar_data = cv2.CascadeClassifier(r'C:\Use
         capture = cv2.VideoCapture(0)
         data = []
         font = cv2.FONT_HERSHEY_COMPLEX
         while True:
             flag,img = capture.read()
             if flag:
                 faces = haar_data.detectMultiScale
                 for x,y,w,h in faces:
                     cv2.rectangle(img,(x,y),(x+w,y
                     face = img[y:y+h,x:x+w, :]
                     face = cv2.resize(face,(50,50)
                     face = face.reshape(1,-1)
                     #face = pca.transform(face)
                     pred = svm.predict(face)[0]
                     n = names[int(pred)]
                     cv2.putText(img, n, (x,y), fon
                     print(n)

                 cv2.imshow('result',img)
                 if cv2.waitKey(2) == 27 :
                     break
         capture.release()
         cv2.destroyAllWindows()
```

```
No Mask
No Mask
No Mask
```

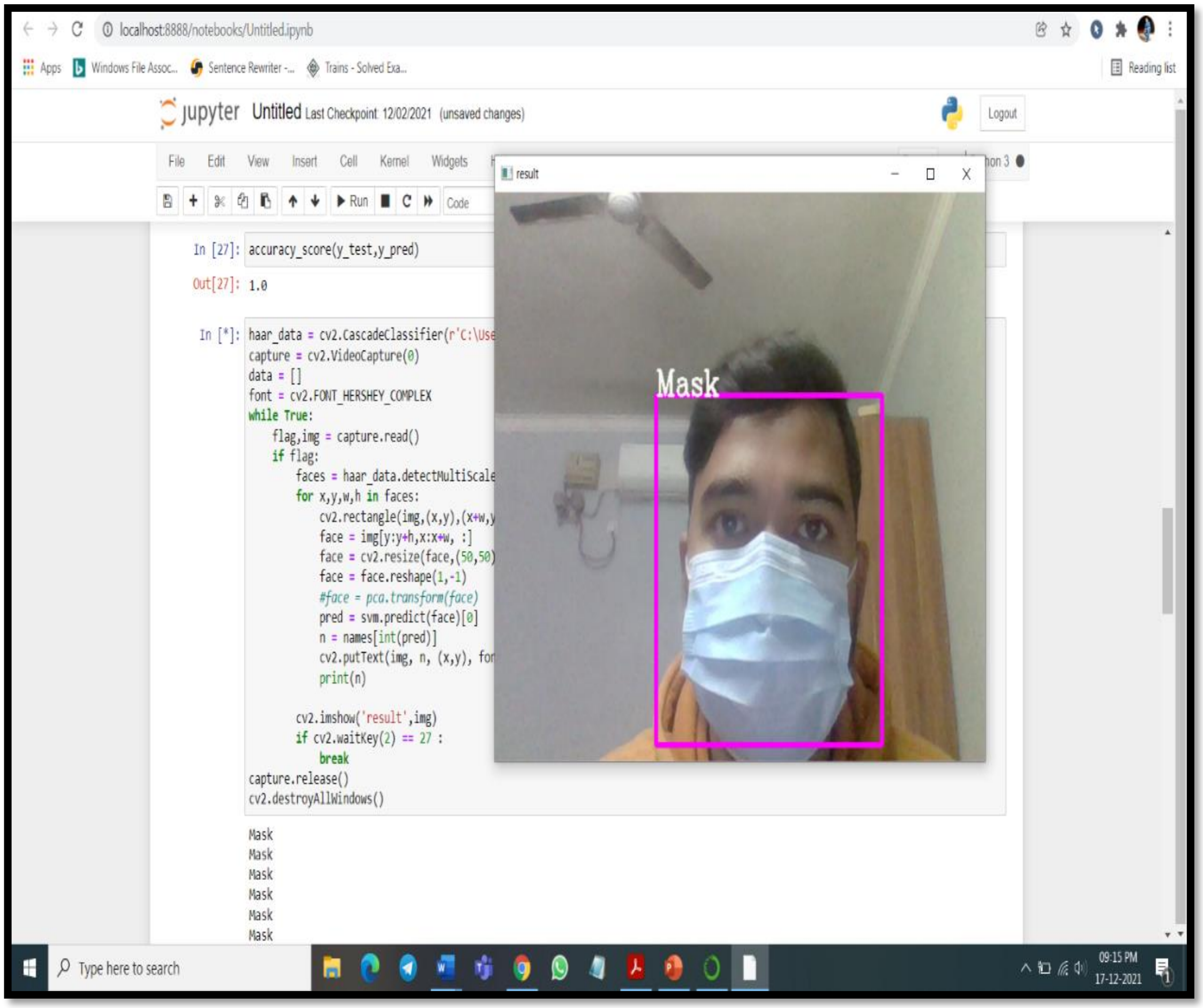
2.MASK

CODE:


jupyter Untitled Last Checkpoint: 12/02/2021 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Run
Code

```
In [27]: accuracy_score(y_test,y_pred)

Out[27]: 1.0

In [28]: haar_data = cv2.CascadeClassifier(r'C:\Users\SHASHANK\data.xml')
         capture = cv2.VideoCapture(0)
         data = []
         font = cv2.FONT_HERSHEY_COMPLEX
         while True:
             flag,img = capture.read()
             if flag:
                 faces = haar_data.detectMultiScale(img)
                 for x,y,w,h in faces:
                     cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,255),4)
                     face = img[y:y+h,x:x+w, :]
                     face = cv2.resize(face,(50,50))
                     face = face.reshape(1,-1)
                     #face = pca.transform(face)
                     pred = svm.predict(face)[0]
                     n = names[int(pred)]
                     cv2.putText(img, n, (x,y), font, 1,(244,250,250), 2)
                     print(n)

                 cv2.imshow('result',img)
                 if cv2.waitKey(2) == 27 :
                     break
         capture.release()
         cv2.destroyAllWindows()

         Mask
         Mask
         Mask
         Mask
         Mask
```

RESULT:


localhost:8888/notebooks/Untitled.ipynb
jupyter Untitled Last Checkpoint: 12/02/2021 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets
Run
Code
result
Mask
Type here to search
09:15 PM
17-12-2021

```
In [27]: accuracy_score(y_test,y_pred)

Out[27]: 1.0

In [*]: haar_data = cv2.CascadeClassifier(r'C:\Use
        capture = cv2.VideoCapture(0)
        data = []
        font = cv2.FONT_HERSHEY_COMPLEX
        while True:
            flag,img = capture.read()
            if flag:
                faces = haar_data.detectMultiScale
                for x,y,w,h in faces:
                    cv2.rectangle(img,(x,y),(x+w,y
                    face = img[y:y+h,x:x+w, :]
                    face = cv2.resize(face,(50,50)
                    face = face.reshape(1,-1)
                    #face = pca.transform(face)
                    pred = svm.predict(face)[0]
                    n = names[int(pred)]
                    cv2.putText(img, n, (x,y), fon
                    print(n)

                cv2.imshow('result',img)
                if cv2.waitKey(2) == 27 :
                    break
        capture.release()
        cv2.destroyAllWindows()

        Mask
        Mask
        Mask
        Mask
        Mask
        Mask
```

CHAPTER-5

CONCLUSION AND FUTURE WORK

In this report, we briefly describe the motivation for the work at the beginning. After that, we demonstrated the learning function and performance of the model. Basic ML tools and simple techniques are used to achieve high accuracy. It can be used for a variety of applications. Wearing a mask may be mandatory soon, considering the Covid-19 problem. The model used will contribute significantly to the public health care system. This detection can also be used for video stream or camera fed inputs. To get improved performance and speed, Raspberry Pi of higher variant such as 4GB or 8GB RAM can be used to implement the detection algorithm.

Future Scope:

The Future development of the project is planned to involve the identification of a person and sent the intimation message to the persons mobile who were not wearing face masks. This can be implemented in offices and institutions by means of training the database with employees images or students images and by means of face recognition the person is identified by which the mobile number and other details of the person is obtained from database and hence it will be easy to notify that particular person or useful for taking any actions regarding not wearing face mask. The proposed model can also be enhanced by means of including various parameters like peoples count, social distance and temperature measurement. This project will be very helpful and can be implemented in hospitals, airports, schools, colleges, offices, shops, malls, theaters, temples, apartments etc. and can also be implemented for Covid free event management. In the future it can be expanded to find out whether a person is wearing a mask properly or not. The model can be further developed to determine whether the mask is infected or not ,the type of mask is undergoing.

REFERENCES

1. Ariyanto, Mochammad & Haryanto, Ismoyo & Setiawan, Joga & Muna, Munadi & Radityo, M.. (2019). Real-Time Image Processing Method Using Raspberry Pi for a Car Model. 46-51.

2. V. K. Bhanse and M. D. Jaybhaye,(2018) "Face Detection and Tracking Using Image Processing on Raspberry Pi," 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, pp. 1099-1103.

3. A. Das, M. Wasif Ansari and R. Basak, (2020) "Covid-19 Face Mask Detection Using TensorFlow, Keras and OpenCV," 2020 IEEE 17th India Council International Conference (INDICON), New Delhi, India,

4. M. S. Islam, E. Haque Moon, M. A. Shaikat and M. Jahangir Alam, (2020) "A Novel Approach to Detect Face Mask using CNN," 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), Thoothukudi, India, pp. 800-806.

5. Joseph Redmon, S. D. (2016) “You Only Look Once(YOLO) Unified, Real Time Object Detection” IEEE.

6. A. Lodh, U. Saxena, A. khan, A. Motwani, L. Shakkeera and V. Y. Sharmasth, (2020) "Prototype for Integration of Face Mask Detection and Person Identification

Model – COVID-19," 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, pp. 1361-1367.

7. Luigi Atzori, Antonio iera and Giacomo Morabito. (2010) ‘The Internet of Things: A Survey’, Journal of Computer Networks Vol.54, No.15,pp.2787-2805.

8. Lu Tan and Neng Wang. (2010) ‘Future internet: The Internet of Things’, IEEE Xplore Proc., 3rd IEEE Int.Conf. Adv. Comp. Theory. Engg. (ICACTE), pp:1-9.

9. J. Marot and S. Bourennane, (2017)"Raspberry Pi for image processing education,"25th European Signal Processing Conference (EUSIPCO), Kos, Greece, 2017, pp. 2364-2366.