

A
PROJECT REPORT
ON
Face Mask Detection System

Submitted by

Satyarth Ratnani (18IT444)

Kishan Thakkar (18IT458)

Vinit Patel (18IT462)

Max Christian (18IT470)

**For Partial Fulfillment of the Requirements for Bachelor of Technology in
Information Technology**

Guided by

Prof.Prachi.Shah

Dec, 2021



Information Technology Department
Birla VishvakarmaMahavidyalaya Engineering College
(An Autonomous Institution)
VallabhVidyanagar – 388120
Gujarat, INDIA



Birla VishvakarmaMahavidyalaya Engineering College

(An Autonomous Institution)

Information Technology Department

AY: 2021-22, Semester I

CERTIFICATE

This is to certify that the project work entitled Face Mask Detection System has been successfully carried out by 18IT444 –Ratnani Satyarth 18IT458 – Thakkar Kishan 18IT462 – Patel Vinit 18IT470 – Christian Max for the subject **Project I (4IT31)** during the academic year 2021-22, Semester-I for partial fulfilment of Bachelor of Technology in Information Technology. The work carried out during the semester is satisfactory.

Prof.Prachi.Shah

IT Department
BVM

Dr.KeyurBrahmbhatt

Head, IT Department
BVM

Acknowledgement

We would like to express a special thanks to our guide **Prof. Prachi Shah** who gave us a golden opportunity to do this wonderful project, which also helped us in doing a lot of research and we came to know about so many new things we are really thankful to them.

Secondly, we would like to thank Our course coordinators **Dr. Zankhna Shah** and **Prof. Vishal Polara** and Also The HOD of our department **Dr. Keyur Brahmbhatt** of college **Birla Vishvakarma Mahavidyalaya** for guiding us to the successful completion of the project.

Abstract

Our system accurately detecting the face mask on human faces from image, video as well as from live camera. It provides attractive website interface to upload image or video and detect mask from that and if face mask is not present on face then an email will be sent to admin.

List of Figures

Figure.1 Image Processing.....	03
Figure.2 Haar Features.....	06
Figure.3 general framework for object detection.....	06
Figure.4 MobileNet SDD.....	07
Figure.5 Convolutional filters.....	09
Figure.6 Existing application.....	11
Figure.7 Timeline Chart.....	12
Figure.8 Training Dataset.....	14
Figure.9 Use Case Diagram.....	16
Figure.10 Class Diagram.....	17
Figure.11 ER Diagram.....	18
Figure.12 DFD Level 0.....	19
Figure.13 DFD Level 1.....	19
Figure.14 Flow chart.....	20
Figure.15 Image into numpy array.....	22
Figure.16 Face Detection.....	24
Figure.17 Accuracy/Loss training curve plot	25
Figure.18 Home page.....	26
Figure.19 Demo page.....	27
Figure.20 Upload page.....	28

Figure.21 About page.....	28
Figure.22 Gallery page.....	29
Figure.23 Upload photo result.....	31
Figure.24 Live video result.....	33
Figure.25 Upload video result.....	34
Figure.26 Testing using use cases.....	35

List of Tables

Table 1: Database Design and Normalization of user login details 19

Table of Content

SR.NO.			TITLE	PGNO.
1.			INTRODUCTION	01
	1.1		Brief overview of the	01
	1.2		workProject Objective	01
	1.3		Project Scope	01
	1.4		Project Modules	02
	1.5		Project Hardware/Software Requirements	02
2.			Literature Review	03
3.			System Analysis & Design	10
	3.1		Comparison of Existing Applications with your Project	10
	3.2		Project Feasibility Study	12
	3.3		Project Timeline chart	12
	3.4		Detailed Modules Description	13
	3.5		Project SRS	16
		3.5.1	Use Case Diagrams	16
		3.5.2	Data Flow Diagrams	17
		3.5.3	Flowchart	18
	3.6		Data Dictionary	19
4.			Implementation and Testing	20
	4.1		User Interface and Snapshot	20
	4.2		Testing using Use Cases	31
5			Conclusion and Future work	33
	5.1		Conclusion	33
	5.2		Future work	33
6			References	34

Chapter 1: Introduction

1.1 Brief overview of project:

Face-mask detection represents both detection as well as a classification problem because it requires first the location of faces of people in digital images and then the decision of whether they are wearing a mask or not.

The first part of this problem has been studied extensively in the computer vision literature, due to the broad applicability of face-detection technology. The second part, on the other hand (i.e., predicting whether a face is masked or not), has only gained interest recently, in the context of the COVID-19 pandemic.

1.2 Objective:

To propose a method that will help the authorities and officials to recognize persons who are not wearing a face mask and the same message will be sent to the official's who is controlling the system that the work will be far more easy to control.

1.3 Scope:

Video is captured and then it is converted into frames, then it will automatically recognize the faces with masks and without masks. Whenever a face is detected without mask a message will be sent to the concerned authority. It is based on computer vision technology, enabling computers to understand images, which can be exploited in wide applications such as,

- Covid 19 Control
- Other epidemic Control
- Image analysis
- Other industrial areas

- Face Mask Detection

1.4 Project Modules:

- Face Detection
- Training the model
- Mask Detection
- Frontend

1.5 Project Hardware/Software requirements:

1.5.1 Hardware Interfaces:

- Device with camera
- RAM: 4GB
- Processor : Intel I3 or higher
- Disk Space: 4 GB Minimum

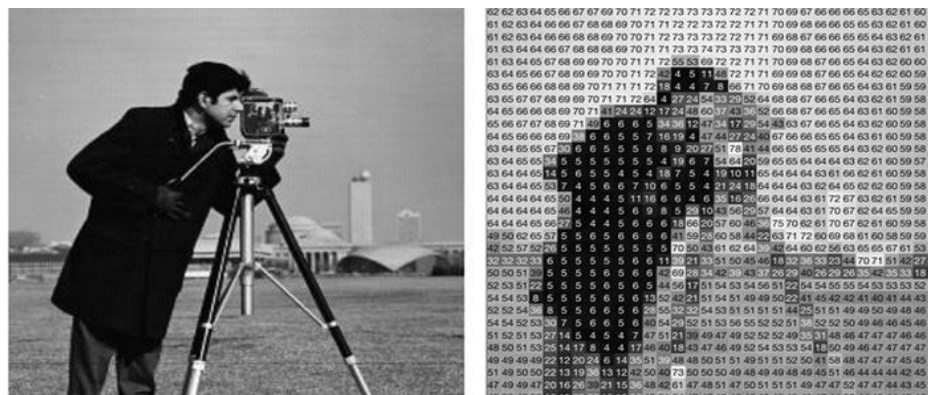
1.5.2 Software Interfaces:

- IDE for python (version 3.9 and above)
- Wamp server for frontend (version 3.0 and above)
- Text editor
- Web browser

Chapter 2: Literature review

2.1 Introduction to Image Processing

Before implementing face mask detection problem, first we need to understand that how to handle images. Images are simply a collection of colors in red, green and blue format. As a human we see an image with some object or shape in it, but for computer it is just an array with color values range from 0 to 255.



(Figure 1 Image Processing)

The way computer sees anything is different from the way human see an image. But that's the good news for us because if we got an array of the image than it becomes simple for us to implement any algorithm on that array.

Steps to Perform Image Processing:

- Load images using Python or any other programming you are working on.
- Converting images into array.
- And finally apply some algorithm on that array.

Another good thing is that we have a library known as OpenCV which will help us to read the image and return array of color pixels.

2.2 Face Detection using OpenCV

Introduction to OpenCV

- OpenCV(Open Source Computer Vision Library)is an open source computer vision and machine learning software library.
- The library has more than 2500optimized algorithms.
- It has C++, Python, Java and MATLAB inter faces and supports Windows, Linux, Android and MacOS.
- Will help us to load images in Python and convert the min to array.
- Each index of array represents (red, green, blue) color pixel which ranges from 0 to 255.

Features of OpenCV

- Face Detection
- Geometric Transformations
- Image Thresholding
- Smoothing Images
- Canny Edge Detection
- Background Removals
- Image Segmentation

2.3 Face Detection Algorithm

In this section, we are going to implement face recognition using OpenCV and Python.

We are using **Viola–Jones object detection algorithm** to detect the face from image.

The Viola–Jones object detection frame work is an object detection frame work which was proposed in 2001 by Paul Viola and Michael Jones. Although it can be trained to detect a variety of object classes, it was motivated primarily by the problem of face

detection.

The algorithm has four stages:

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

Haar Features

All human faces have some similar properties. The regularities may be matched using Haar Features.

A few properties common to human faces:

- The eye region is darker than the upper-cheeks.
- The nose bridge region is brighter than the eyes.



(Figure 2.1 Haar features)

Composition of properties forming matchable facial features:

- Location and size: eyes, mouth, bridge of nose
- Value : oriented gradients of pixel intensities



(Figure 2.2 Haar Features)

The four features matched by this algorithm are then sought in the image of a face Rectangle features:

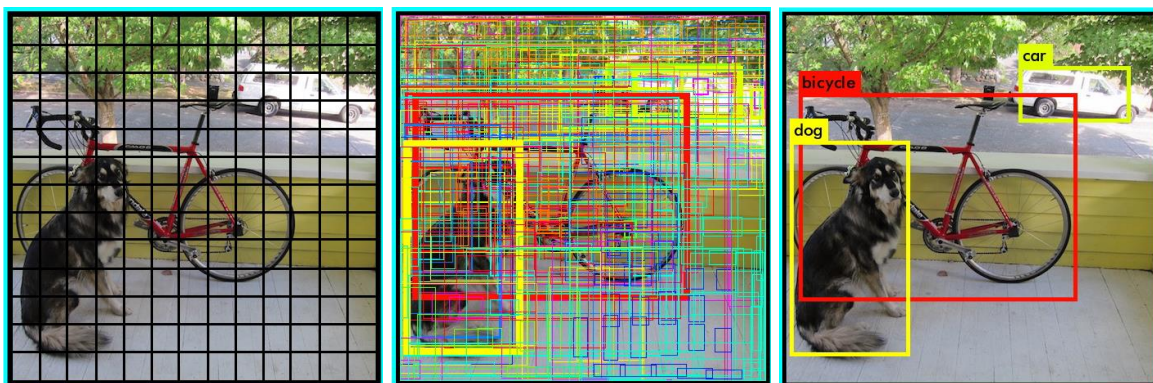
- $\text{Value} = \Sigma(\text{pixel in black area}) - \Sigma(\text{pixel in white area})$
- Three types: two-, three-, four-rectangles, Viola & Jones used two-rectangle features
- For example : the difference in brightness between the white & black rectangles over a specific area
- Each feature is related to a special location in the sub-window

2.4 Object detection using tensorflow

2.4.1 A General Framework for Object Detection

Typically, we follow three steps when building an object detection framework:

1. First, a deep learning model or algorithm is used to generate a large set of bounding boxes spanning the full image (that is, an object localization component)



(Figure 3 General framework for object detection)

2. Next, visual features are extracted for each of the bounding boxes. They are evaluated and it is determined whether and which objects are present in the boxes based on visual features (i.e. an object classification component)
3. In the final post-processing step, overlapping boxes are combined into a single bounding box (that is, non-maximum suppression)

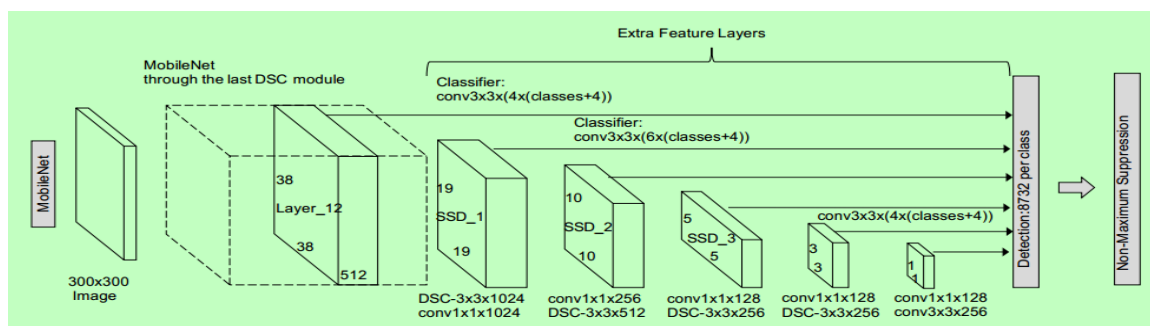
2.4.2 TensorFlow Object Detection API

The TensorFlow object detection API is the framework for creating a deep learning network that solves object detection problems.

There are already pretrained models in their framework which they refer to as Model Zoo. This includes a collection of pretrained models trained on the COCO dataset, the KITTI dataset, and the Open Images Dataset. These models can be used for inference if we are interested in categories only in this dataset.

2.4.3 MobileNet-SSD

The SSD architecture is a single convolution network that learns to predict bounding box locations and classify these locations in one pass. Hence, SSD can be trained end-to-end. The SSD network consists of base architecture (MobileNet in this case) followed by several convolution layers:



(Figure 4 MobileNet SDD)

SSD operates on feature maps to detect the location of bounding boxes. Remember – a feature map is of the size $D_f * D_f * M$. For each feature map location, k bounding boxes are predicted.

2.4.4 Loss in MobileNet-SSD

With the final set of matched boxes, we can compute the loss like this:

$$L = 1/N (L_{\text{class}} + L_{\text{box}})$$

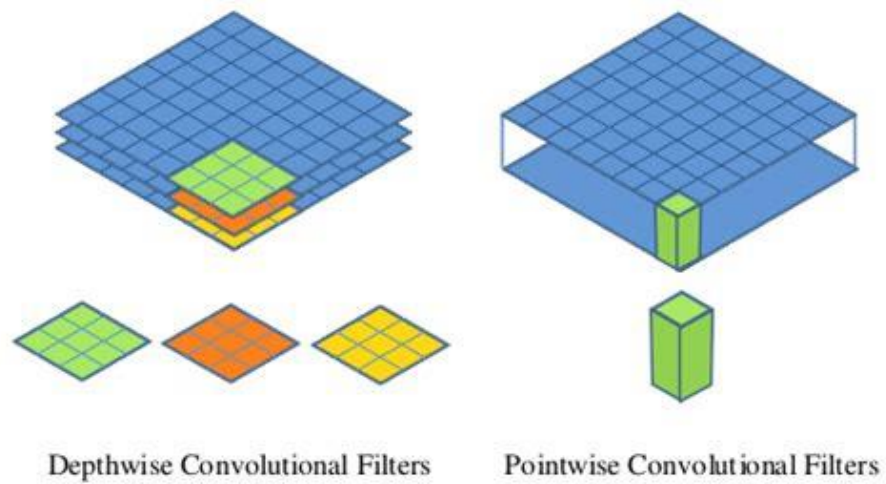
Here, N is the total number of matched boxes. L_{class} is the softmax loss for classification and ‘ L_{box} ’ is the L1 smooth loss representing the error of matched boxes. L1 smooth loss is a modification of L1 loss which is more robust to outliers. In the event that N is 0, the loss is set to 0 as well.

MobileNet

The MobileNet model is based on depthwise separable convolutions which are a form of factorized convolutions. These factorize a standard convolution into a depthwise convolution and a 1×1 convolution called a pointwise convolution.

For MobileNets, the depthwise convolution applies a single filter to each input channel. The pointwise convolution then applies a 1×1 convolution to combine the outputs of the depthwise convolution.

A standard convolution both filters and combines inputs into a new set of outputs in one step. The depthwise separable convolution splits this into two layers – a separate layer for filtering and a separate layer for combining. **This factorization has the effect of drastically reducing computation and model size.**



Depthwise Separable Convolution

(Figure 5 Convolutional filters)

Chapter 3: System Analysis & Design

3.1 Comparison of Existing Applications with your Project with merits and demerits:

3.1.1 Uber:

Uber to use face detection technology to ensure drivers wear masks

Before drivers start accepting rides, they will be required to complete a "Go Online Checklist" in the application. This list includes taking a photo of themselves wearing a face covering.

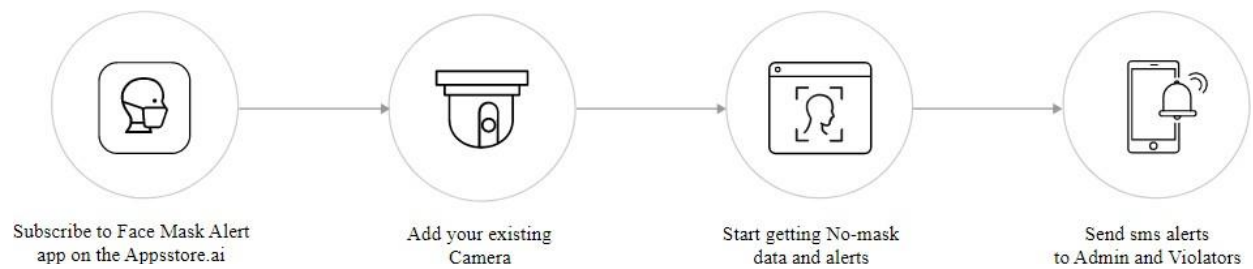
Uber's Global Head of Safety Products Sachin Kansal demonstrated to ABC News how the new software in the app can determine whether a driver is wearing a face covering based on the photo. If a face covering is not detected, the driver cannot go online.



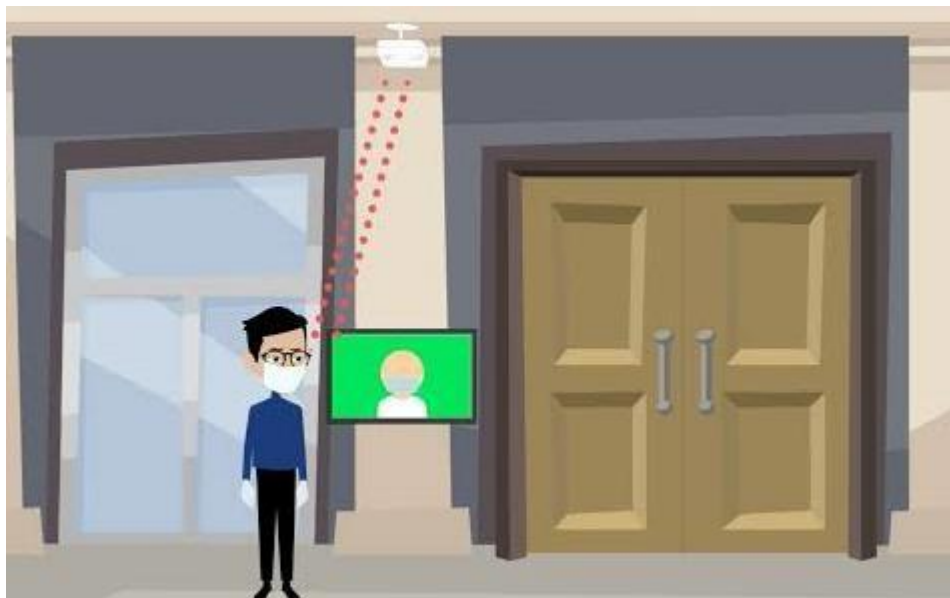
3.1.2 leewayhertz:

Face Mask Detection Platform uses Artificial Network to recognize if a user is not wearing a mask. The app can be connected to any existing or new IP mask detection cameras to detect people without a mask. App users can also add faces and phone numbers to send them an alert in case they are not wearing a mask. If the camera captures an unrecognized face, a notification can be sent out to the administrator.

How does it work?



3.1.3 Digital Barriers launches face mask detection and people counting solution to keep businesses COVID-19-secure:



(Figure 6 Existing application)

3.2 Project Feasibility Study:

3.2.1 Technical Feasibility

Technical feasibility assesses the current resources (hardware and software) and technologies, which are required to accomplish user requirements. It requires a computer with python installed. Today every organization has computer, so it is not an extra cost.








3.2.2 Economical Feasibility

Economic feasibility is the most frequently used method for evaluating the effectiveness of proposed system. The proposed model is cost effective.

3.2.3 Operational feasibility

The proposed system performs effective than the Existing system. The system recognizes a person without wearing a mask.

3.3 Project Timeline chart:

Development Phase	90 Days						Duration (Day)
	0 to 15 Day	16 to 30 Day	31 to 45 Day	46 to 60 Day	61 to 75 Day	76 to 90 Day	
Requirement Gathering							10
Analysis							15
Design							30
Coding							25
Testing							12
Implementation							08
Documentation							80
Total Time (Days)							90

(Figure 7 Timeline chart)

3.4 Detailed Modules Description:

- **Face Detection**

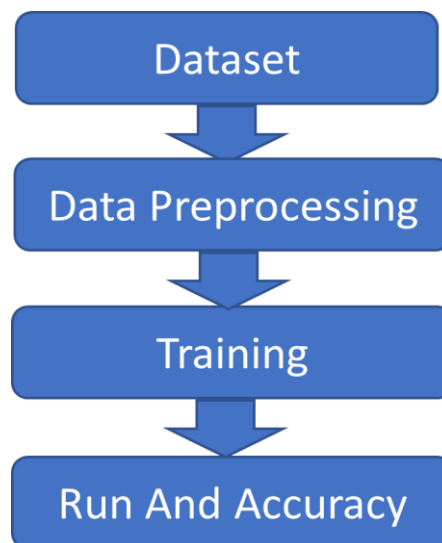
In our project for face detection we are using haar cascade classifier which feature proposed by the viola-Jones algorithm.

The Viola-Jones algorithm first detects the face on the grayscale image and then finds the location on the colored image.

Haar cascade classifier is an Object Detection Algorithm used to identify faces in an image or a real time video. The algorithm uses edge or line detection features

- **Training the model**

Below figure show the step of implementation of our project



- **Dataset**

We have dataset inside the database folder we have two folder with mask and without mask with mask contain all the people who wear a mask and without mask contain all the people who were not wear a mask. Both folders contain around 4000 images.

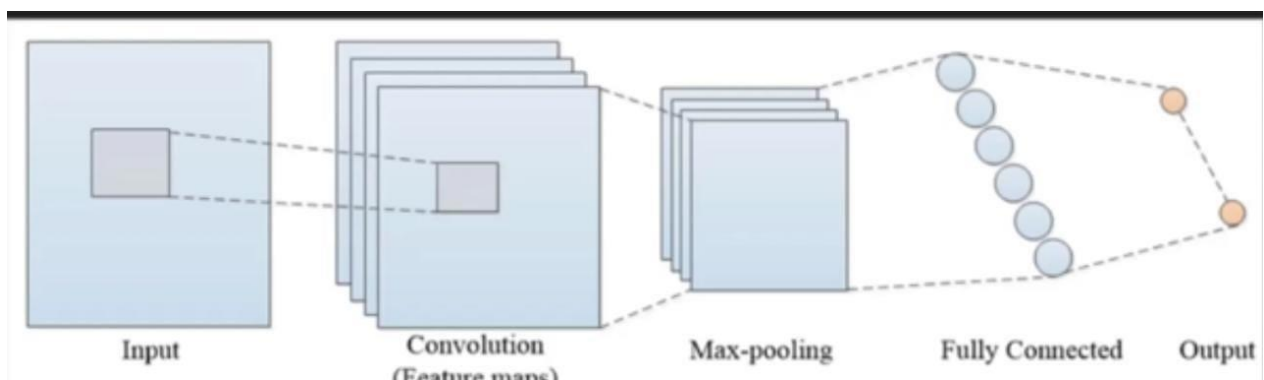
- **Data Preprocessing**

We are going to convert all the images of folder what we have with mask and without mask into arrays.

- Load_image
- Img_to_array
- LabelBinarizer
- Train_test_split

- **Training**

- Image data generator
- Base Model
- Head model



(Figure 8 Training dataset)

- **Mask detection**

User can also upload the photo and check whether the person is wearing a mask or not
user can also upload the video and also check through live video detection.

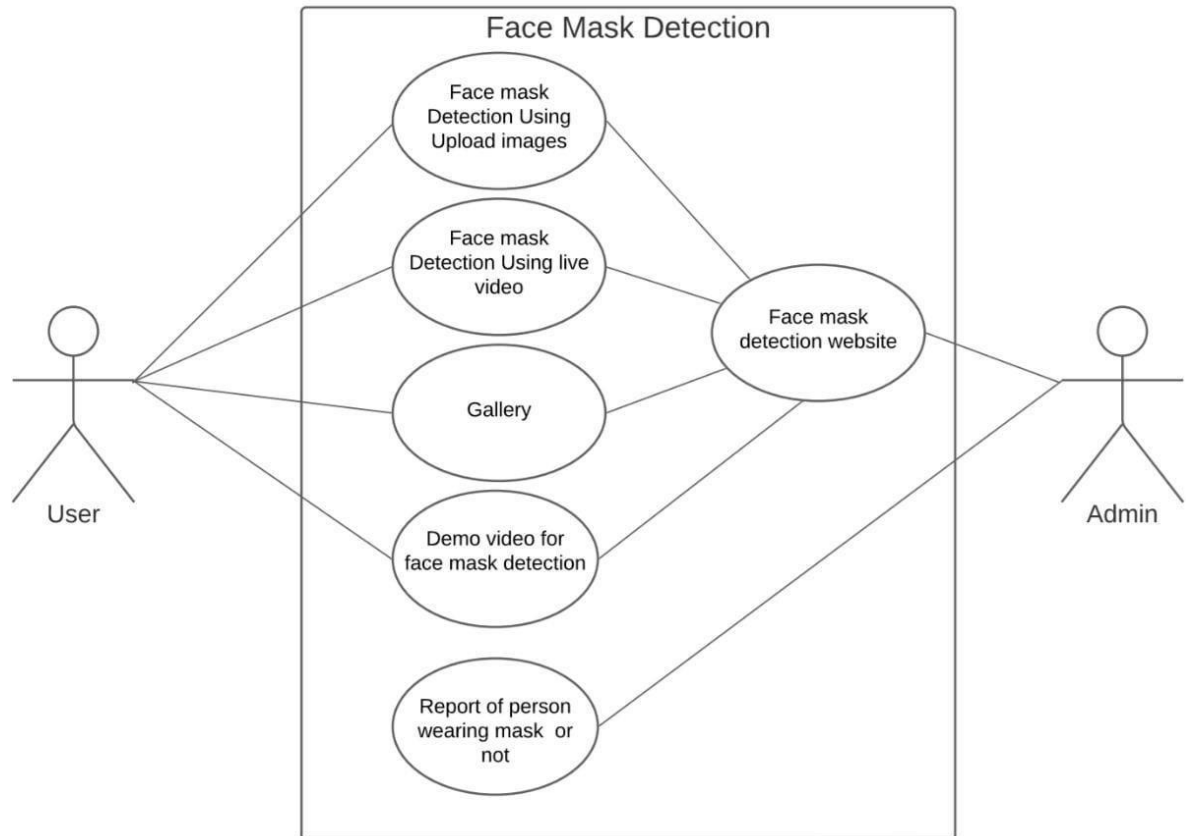
Our system also predict the percentage of wearing mask

- **Front end**

The website where user can upload Image, video and detect the face mask or by live
Video using device camera also can detect face mask and see the results.

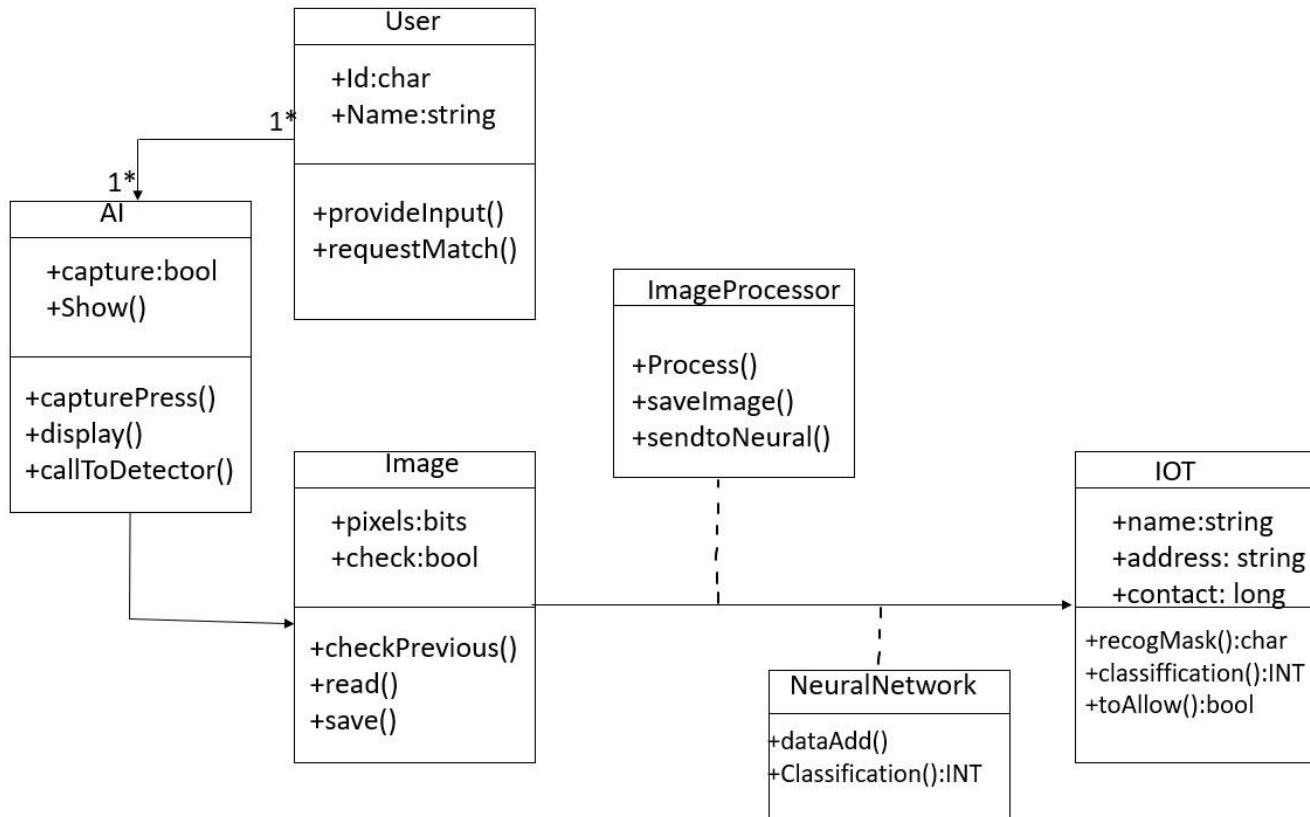
3.5 Project SRS:

3.5.1 USE CASE DIAGRAM:



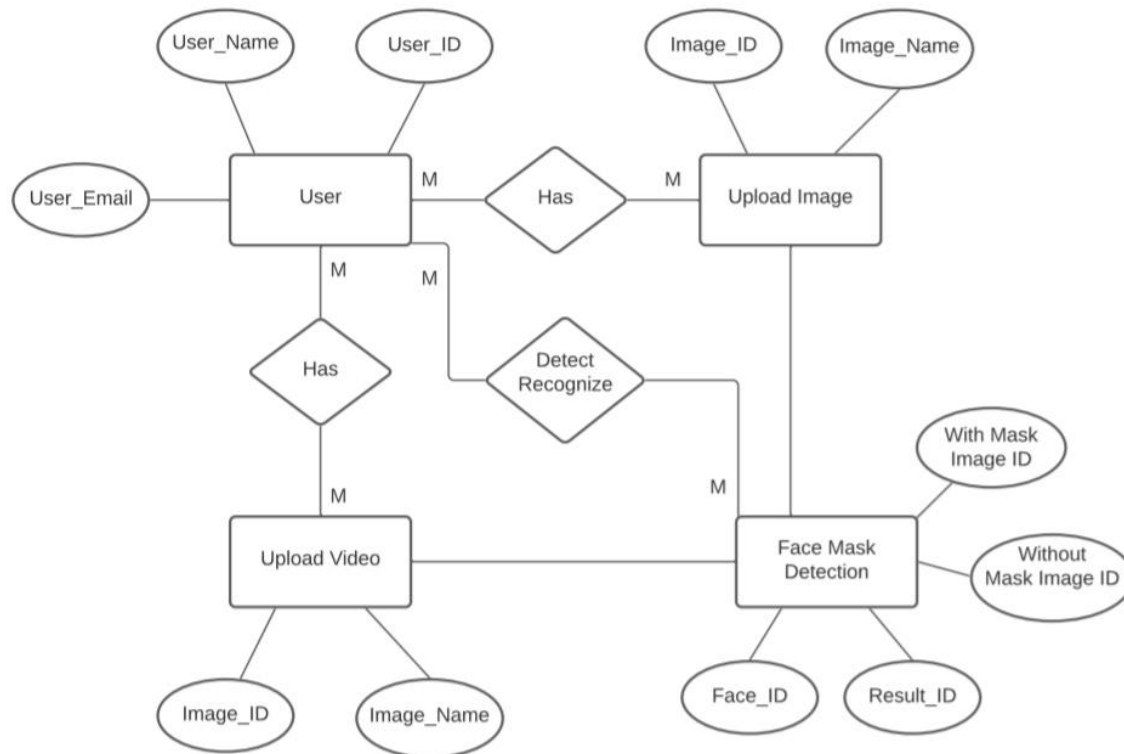
(Figure 9 Use Case Diagram)

3.5.2 CLASS DIAGRAM



(Figure 10 Class diagram)

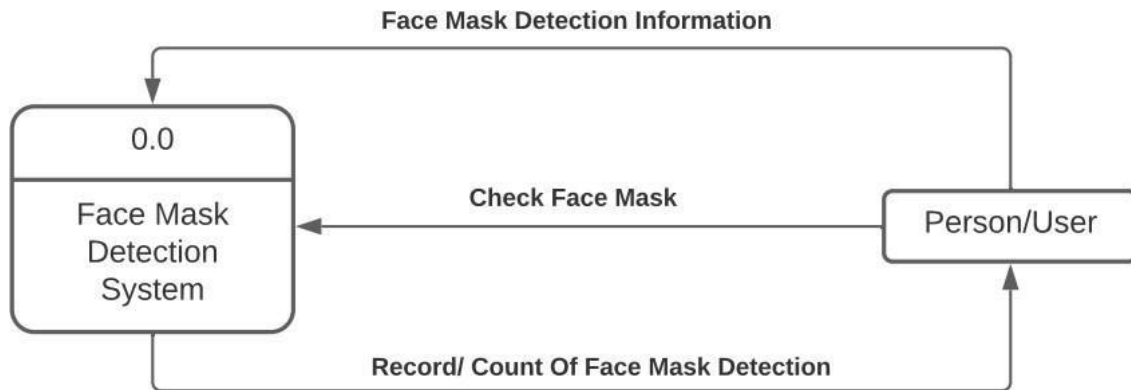
3.5.3 ER DIAGRAM



(Figure 11 ER diagram)

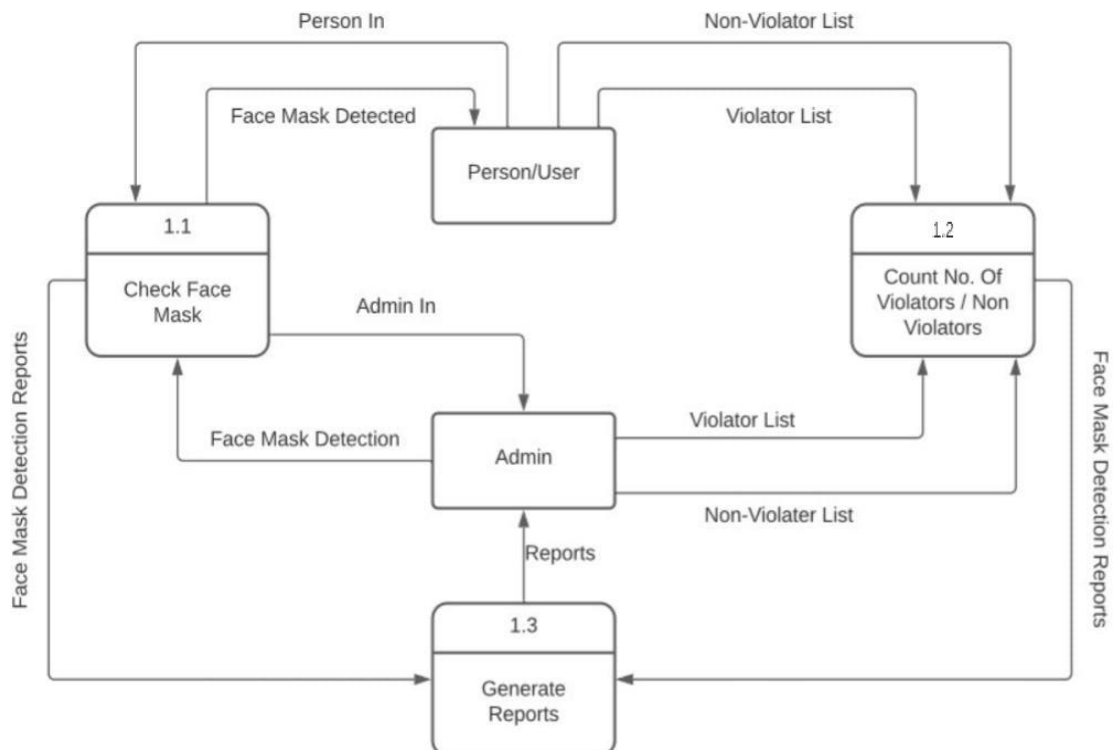
Data Flow Diagram:

- LEVEL 0 of DFD:



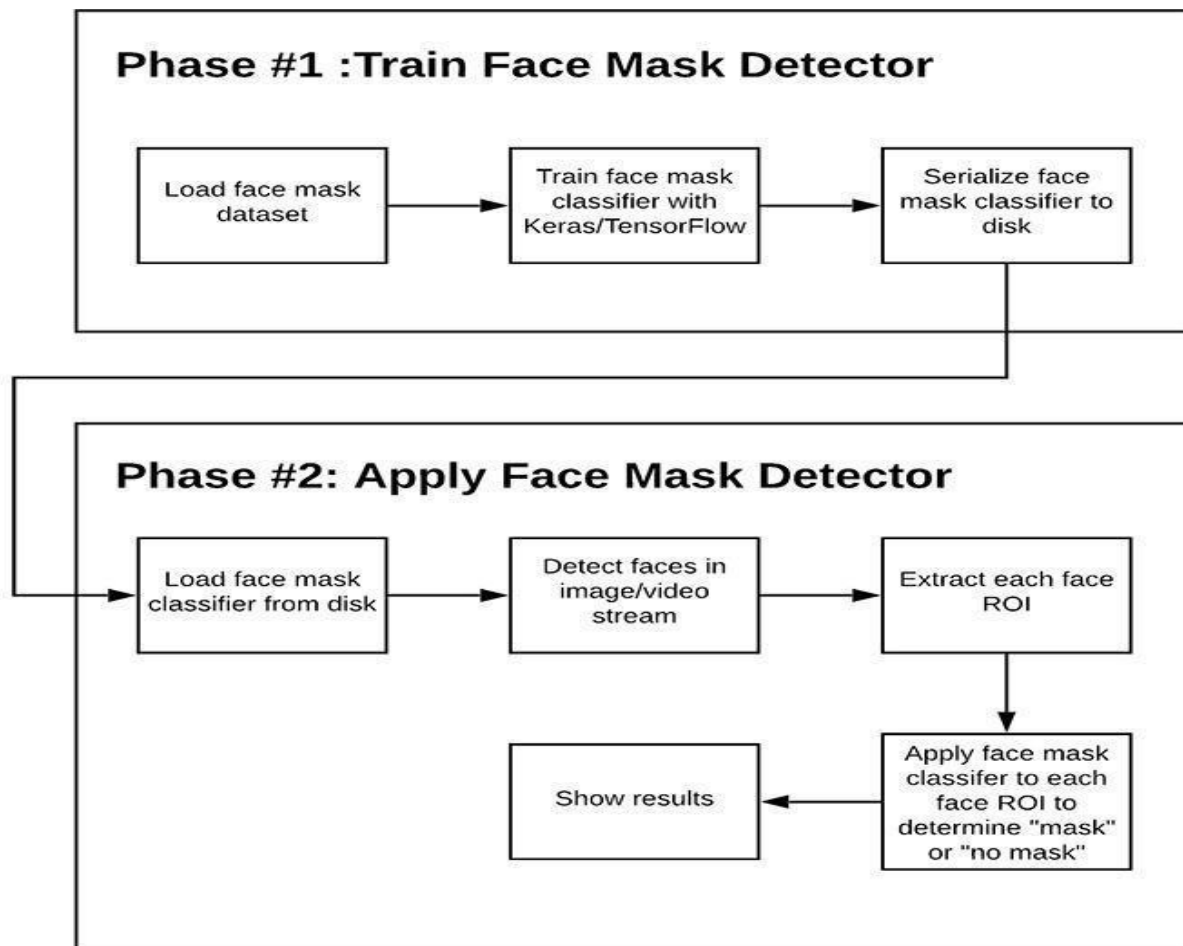
(Figure 12 DFD level 0)

- LEVEL 1 DFD:



(Figure 13 DFD level 1)

3.5.2 Flowchart:



(Figure 14 Flow chart)

3.6 Data Dictionary:

- Table Name : Login table

Description: To Store the admin and user login details.

(Table 1)

Sr No.	Name	Data type	Constraint	Description
1	User_name	Varchar(10)	Primary Key	To store the user name
2	User_type	Varchar(10)	Not null	To Store the user type
3	Password	Varchar(10)	Not null	To Store the Password

Chapter 4: Implementation and Testing

4.1 User Interface and Snapshot

4.1.1 Face detection

First we did image processing using OpenCV Package. So first load OpenCV package. after importing OpenCV let's read an image.

```
import cv2

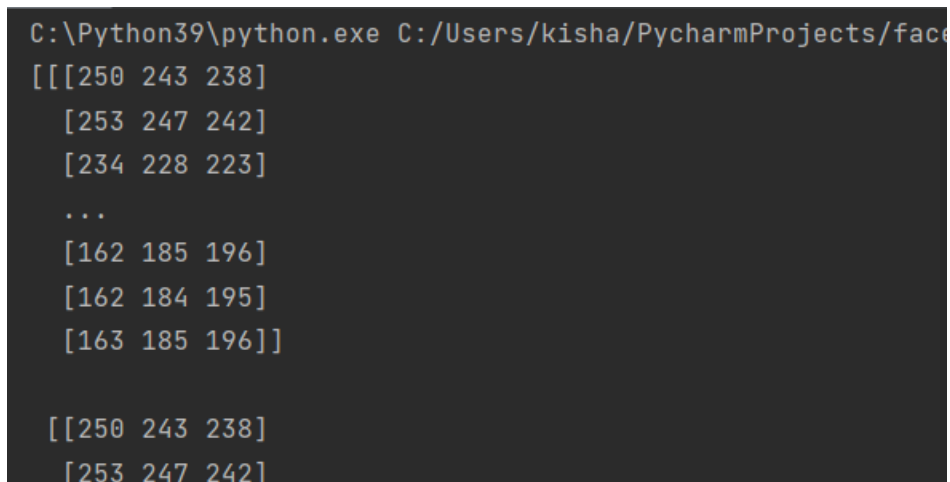
haar_data = cv2.CascadeClassifier('data.xml')

img = cv2.imread('image3.png')

print(img)

print("\n")
```

So, when we read any image using OpenCV it returns object of numpy array by default and using `img.shape` we are checking the height and width of image and also it returns 3 which is the color channel of image. Now we can see the values of array are the color values actually.



```
C:\Python39\python.exe C:/Users/kisha/PycharmProjects/face
[[250 243 238]
 [253 247 242]
 [234 228 223]
 ...
 [162 185 196]
 [162 184 195]
 [163 185 196]]

[[250 243 238]
 [253 247 242]]
```

(Figure 15 Image into a numpy array)

So now we are going to see how to detect face from an image. Face detection algorithm was introduced by Viola and Jones in 2001. They divided this algorithm in four stages :

1. Haar Features Selection
2. Integral Images
3. AdaBoost
4. Cascading Classifier

Data.xml file is a dataset which is going to help us to detect faces from the image.

```
haar_data = cv2.CascadeClassifier('data.xml')
```

```
while True:
```

```
    faces = haar_data.detectMultiScale(img)
```

```
    print(faces)
```

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

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

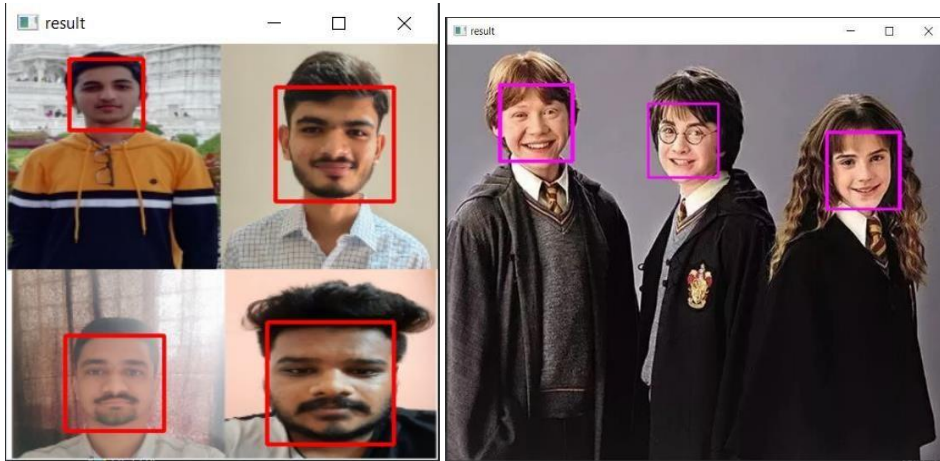
```
cv2.imshow('result',img)
```

```
if cv2.waitKey(0) == 27:
```

```
    break
```

```
cv2.destroyAllWindows()
```

It will detect the faces from uploaded image and highlight it by colored rectangle.



(Figure 16 Face detection)

4.1.2 Training the model

- **TensorFlow(For Better Accuracy):**

It is an open source artificial intelligence library, using data flow graphs to build models. It allows developers to create large-scale neural networks with many layers. TensorFlow is mainly used for: **Classification, Perception, Understanding, Discovering, Prediction and Creation**

The images used in the dataset are real images of people wearing mask i.e. the dataset doesn't contain morphed masked images. The model is accurately trained and, also the system can therefore be used in real-time applications which require face-mask detection.

- **Training of CNN Model :**

Open terminal. Go into the project directory folder and type the following command:

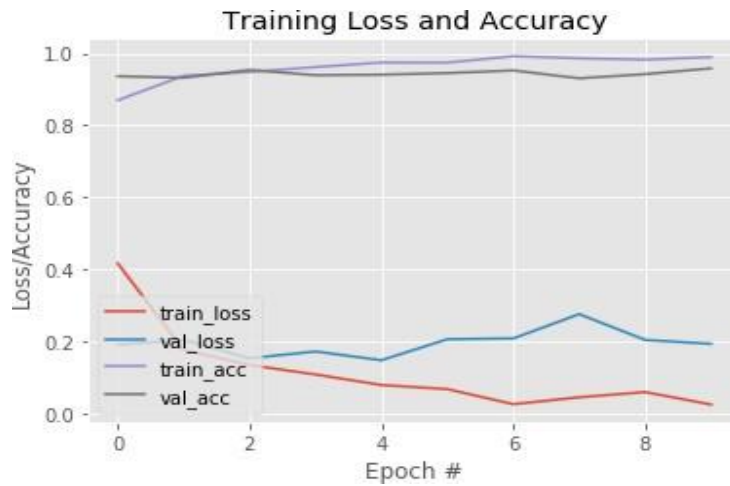
```
python train.py --dataset
```

- **Testing of CNN Model :**

For detecting face mask in images, run the following command :


```
python Mask_Detection_in_Image.py --image data/image1.jpg
```

- **Accuracy/Loss training curve plot:**

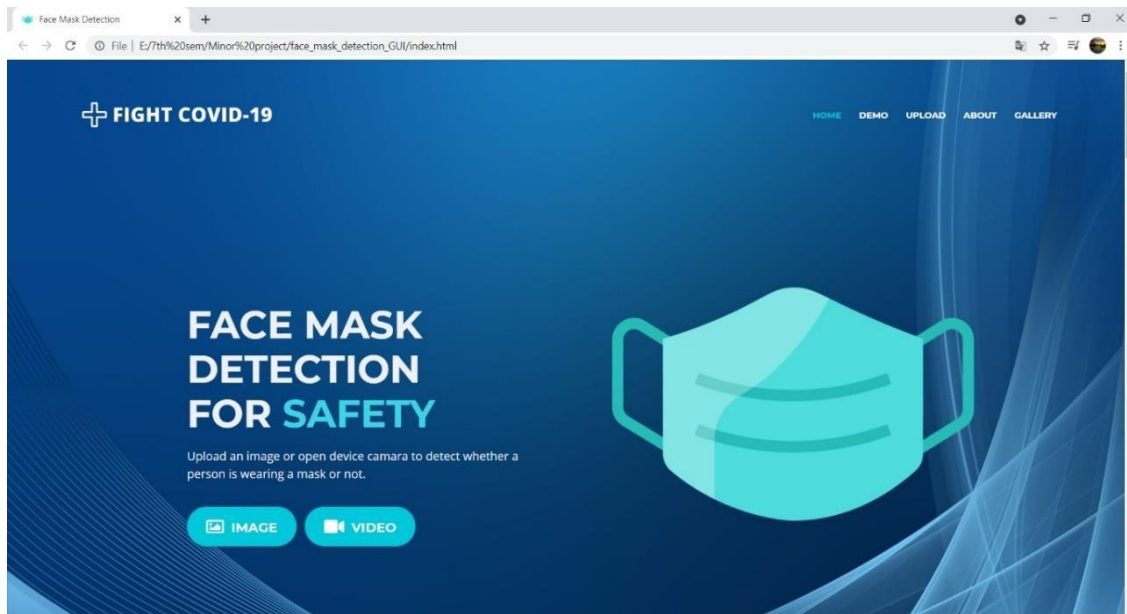


(Figure 17 Accuracy/Loss training curve plot)

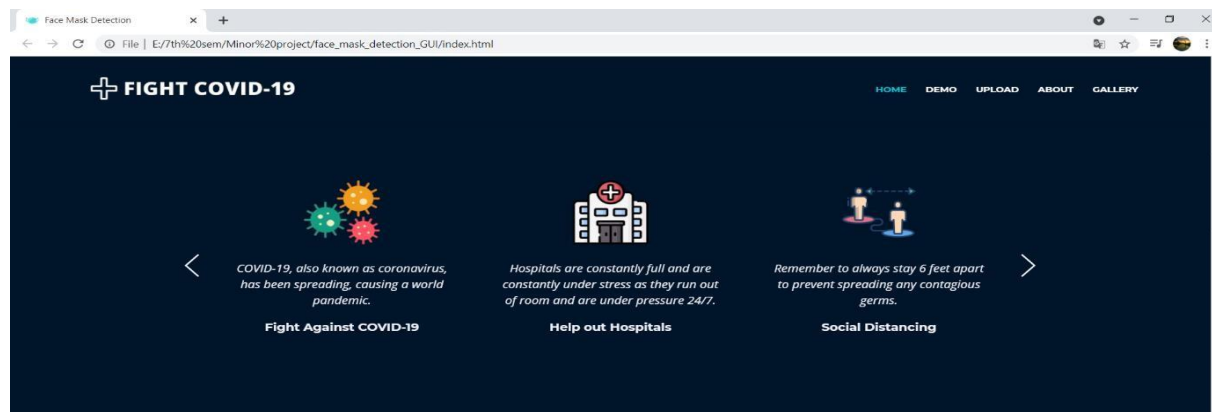
4.1.3 Website interface:

Homepage:

Main page that demonstrates title, logo, navigation bar, and an iconic carousel for important reminders and relevant information.



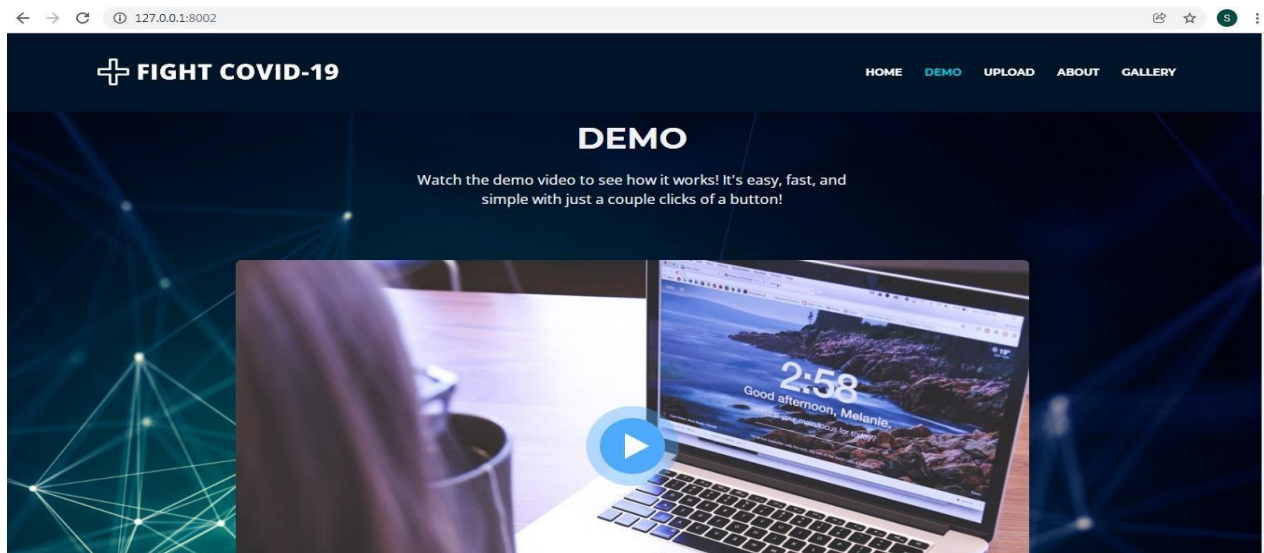
After home page we have a slider which is showing some general safety instruction to increase awareness about covid -19.



(Figure 18 Home page)

Demo Page:

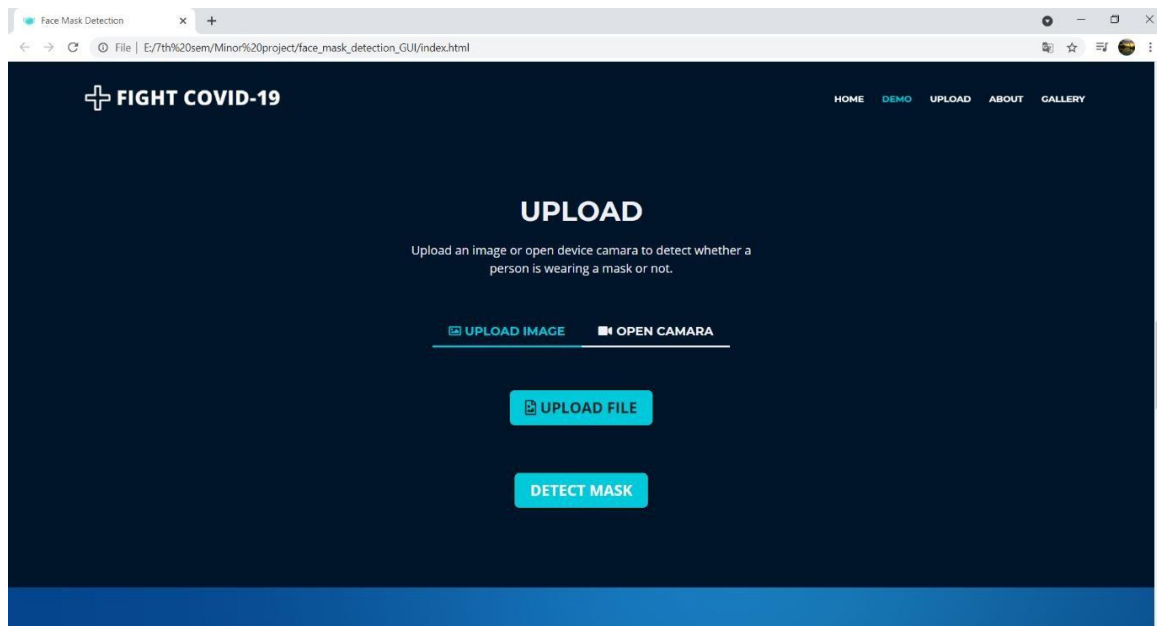
- To demonstrate the easy and simple steps of how to upload an image or video to create an output.
- By clicking on video icon it will open a video which is explaining features of website.



(Figure 19 Demo page)

Upload Page:

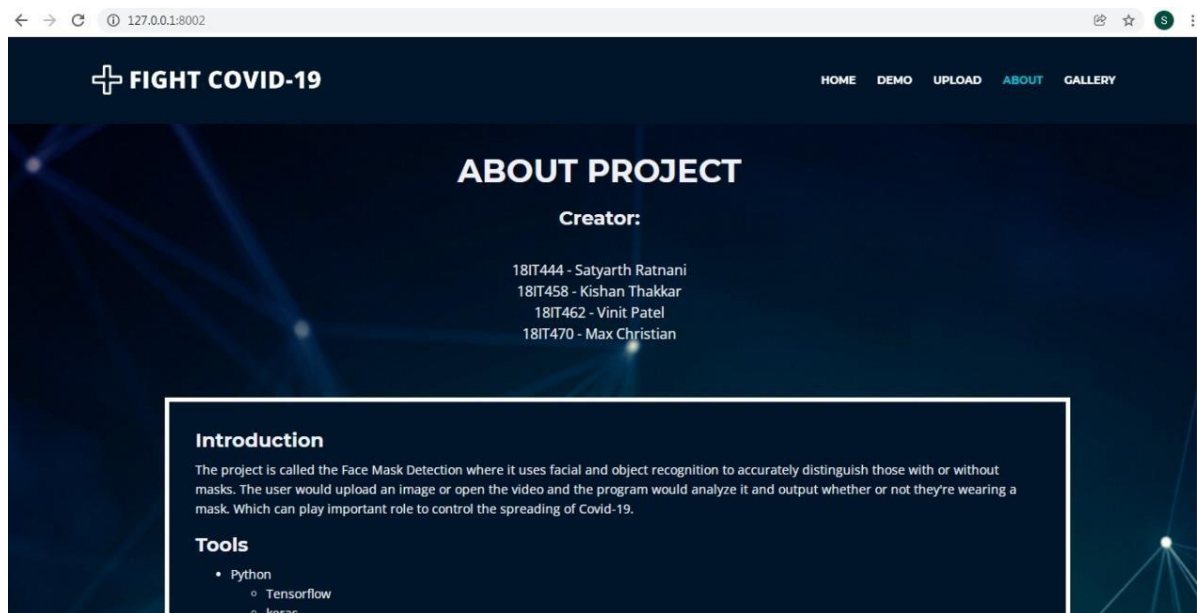
The area where the user will upload their image or open device camera to capture a video and the output will display back with a frame and percentage that shows if the person is wearing a mask or not.



(Figure 20 Upload page)

About Page:

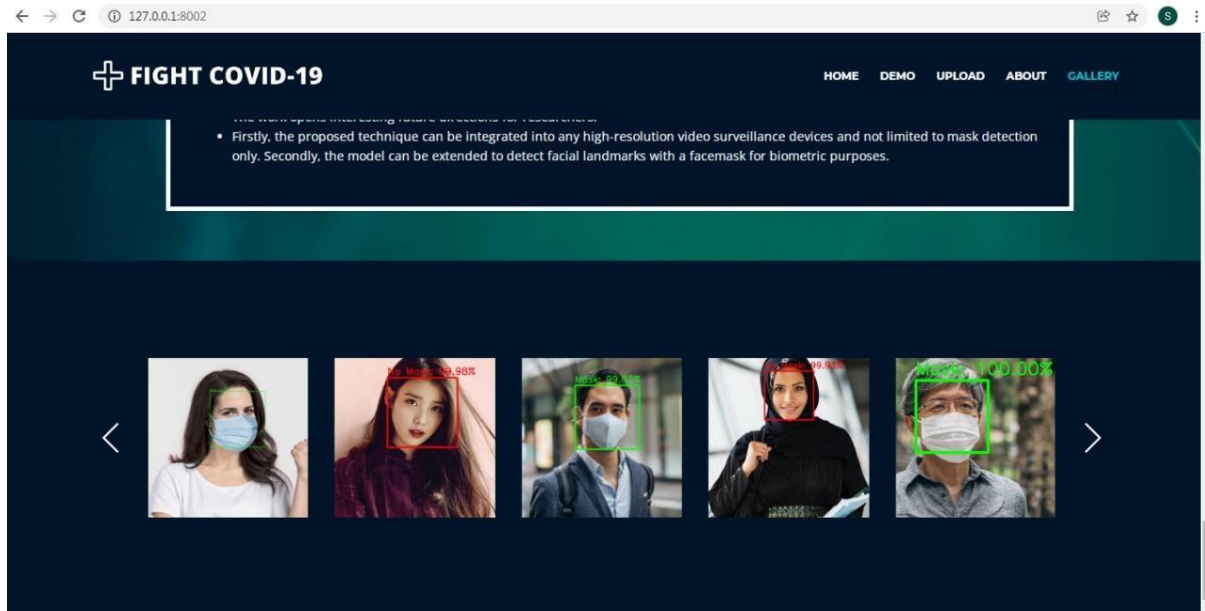
The project description, information, and future discussion are posted there.



(Figure 21 About page)

Gallery Page:

A carousel containing lots of images that I tested out to show some samples of what similar results could be produced.



(Figure 22 Gallery page)

4.1.4 Mask detection:

- **From image**

User can upload an image by clicking on upload image button. The preview of image will be shown. By clicking on detect mask button face mask will be detected from the image.

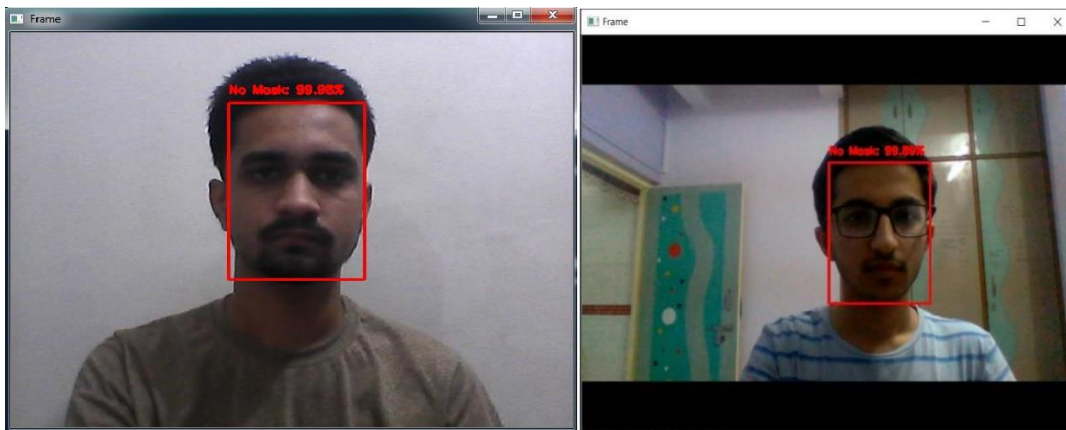
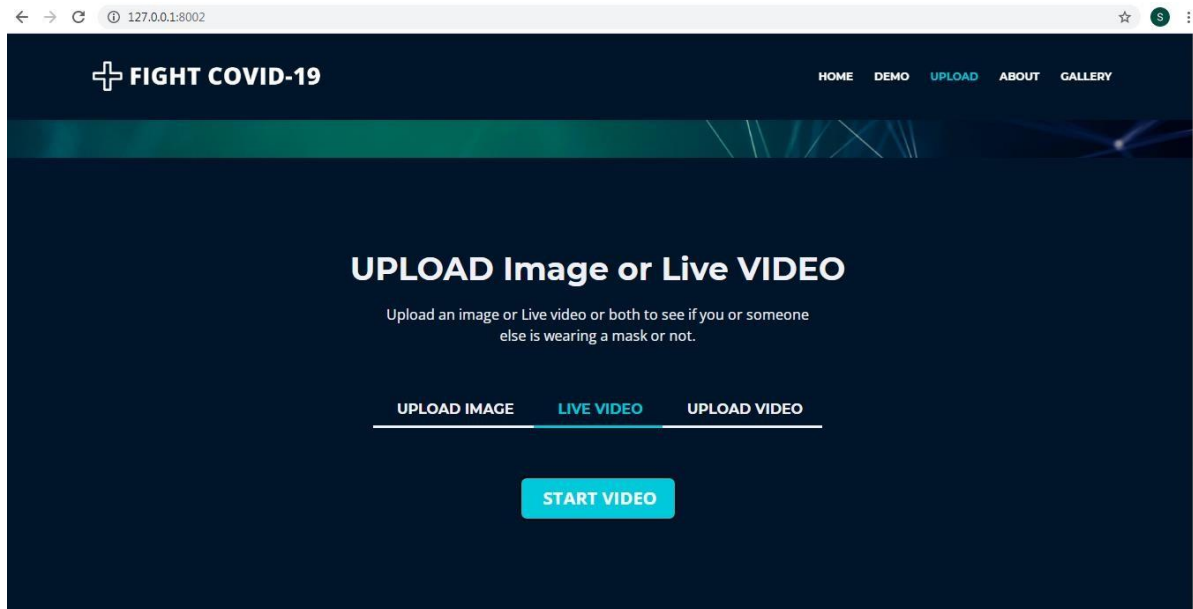




(Figure 23 Upload photo and result)

- **From live video**

By clicking on start video button the device camera will be turned on and from the video our model will detect whether a face mask is present or not.

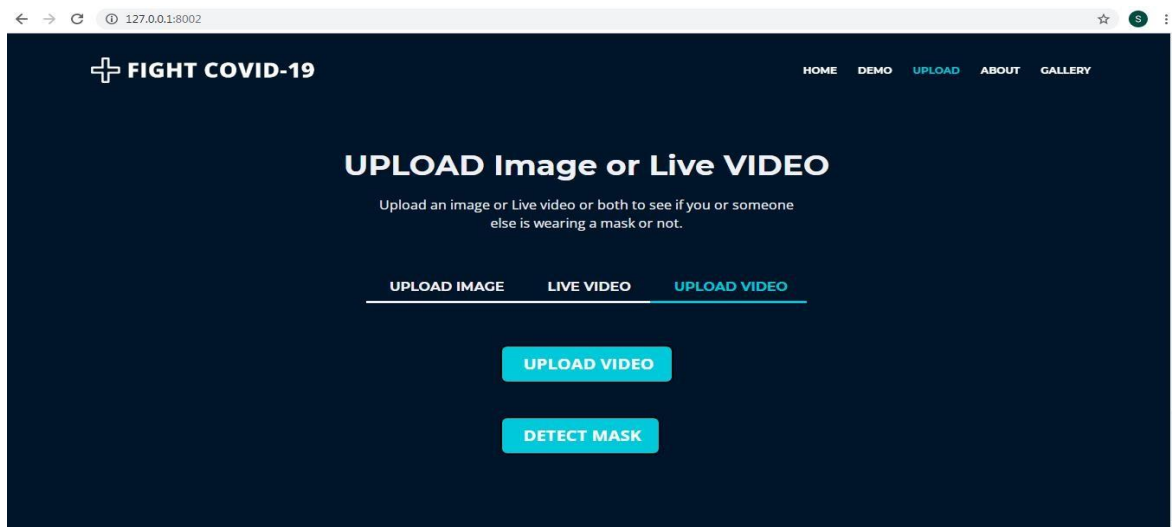




(Figure 24 From live video)

- **From uploaded video**

User can upload video and check whether a person is wearing a mask or not in the video.

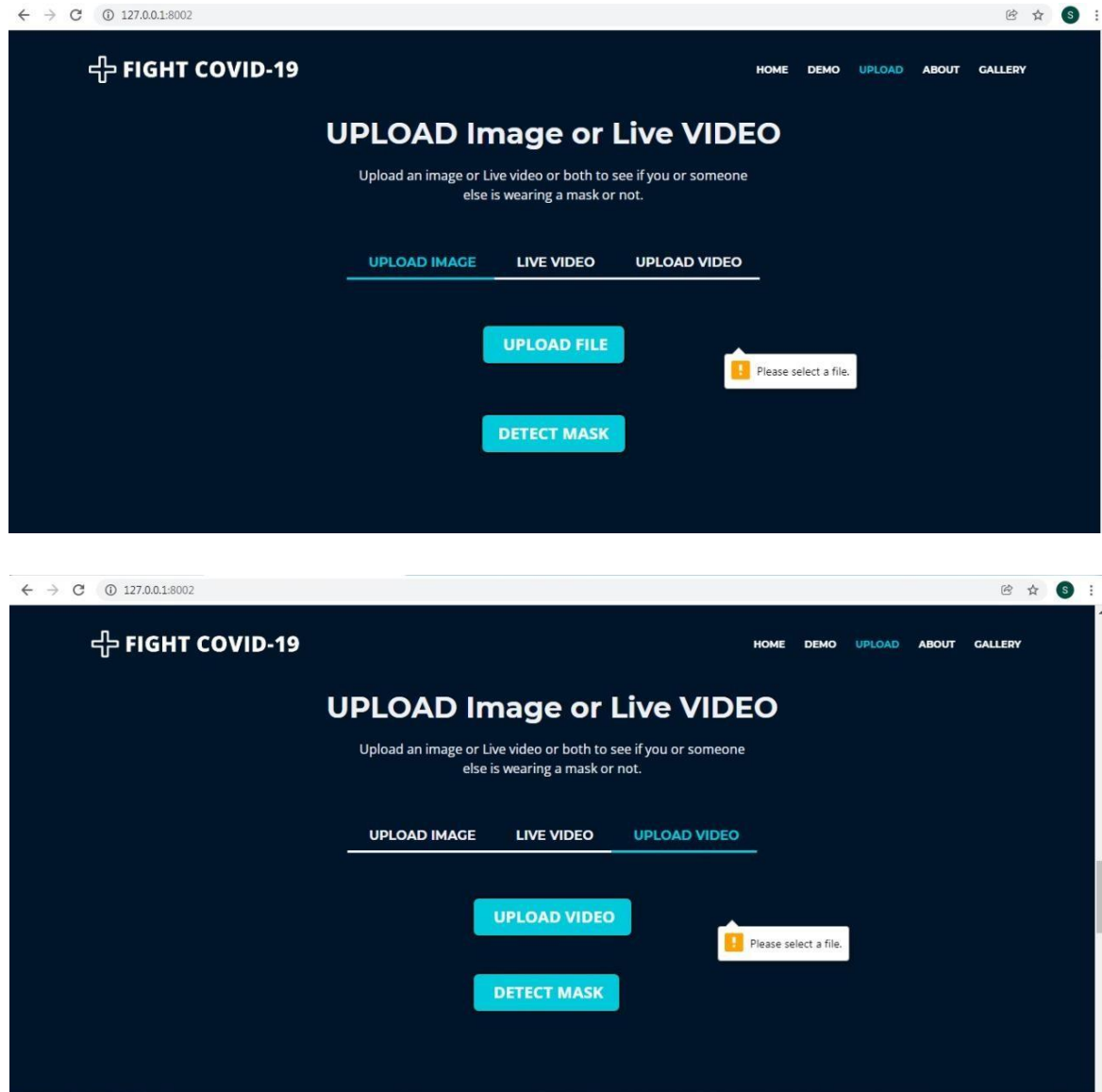




(Figure 25 From uploaded video)

4.2 Testing using Use Cases

In upload image for upload video section If a user directly click on detect mask button without uploading a file then a message will be shown “please select a file”.



(Figure 26 Testing using use cases)

If uploaded image file for video file is correct then it will perfectly detect the mask from face like we have seen above.

Chapter 5: Conclusion & Future work

5.1 Conclusion

To mitigate the spread of COVID-19 pandemic, measures must be taken. We have modeled a facemask detector using tensorflow, OpenCV and CNN. To train,validate and test the model, we used the dataset that consisted around 2000 masked and unmasked faces images. These images were taken from various resources like Kaggle. The model was inferred on images and live video streams. To select a base model,we evaluated the metrics like accuracy, precision and recall and selected MobileNetV2 architecturewith the best performance. The face maskdetector can be deployed in many areas like shopping malls, airports and other heavy traffic placesto monitor the public and to avoid the spread of the disease.

5.2 Future work

This work opens interesting future directions for researchers. Firstly, the proposed techniquecan be integrated into any high-resolution video surveillance devices and not limited to mask detection only. Secondly, the model can be extended to detect facial landmarks with a facemask forbiometric purposes.

Chapter 6: References

- [1] John V Guttag, “*Introduction to Computation and Programming Using Python*”, Prentice Hall of India.
 - [2] Rich and Knight, “*A Artificial Intelligence*”, Tata Mcgraw Hills publication, 3rd edition.
 - [3] Kevin Murphy, “*Machine Learning: A Probabilistic Perspective*”, MIT Press, 2012.
 - [4] Adjabi, I.; Ouahabi, A.; Benzaoui, A.; Taleb-Ahmed, A. Past, present, and future of face recognition: A review. *Electronics* **2020**, *9*, 1188.
 - [5] Voulodimos, A.; Doulamis, N.; Doulamis, A.; Protopapadakis, E. Deep Learning for Computer Vision: A Brief Review. *Comput. Intell. Neurosci.* 2018, 2018.
 - [6] Dang, K.; Sharma, S. Review and comparison of face detection algorithms. In *Proceedings of the 7th International Conference Confluence 2017 on Cloud Computing, Data Science and Engineering*, Noida, India, 12–13 January 2017; pp. 629–633.
 - [7] Ngan, M.; Grother, P.; Hanaoka, K. Ongoing Face Recognition Vendor Test (FRVT)—Part 6A: Face recognition accuracy with masks using pre-COVID-19 algorithms. *Natl. Inst. Stand. Technol.* **2020**.
 - [8] Galterio, M.G.; Shavit, S.A.; Hayajneh, T. A review of facial biometrics security for smart devices. *Computers* 2018, *7*, 37.
 - [9] www.github.com
 - [10] www.youtube.com
 - [11] [Scikit-learn.org](https://scikit-learn.org)
 - [12] www.stackoverflow.com
 - [13] <https://www.kaggle.com/omkargurav/face-mask-dataset>
 - [14] <https://www.lucidchart.com/>
-