Real Time Face-mask Detection with Arduino Based Smart Door

Rafiqul Islam Munna

Samiur Rahman

A Thesis in the Partial Fulfillment of the Requirements

for the Award of Bachelor of Computer Science and Engineering (BCSE)

Department of Computer Science and Engineering
College of Engineering and Technology
IUBAT – International University of Business Agriculture and Technology

Fall 2020

Real Time Face-mask Detection with Arduino Based Smart Door

Rafiqul Islam Munna and Samiur Rahman

A Thesis in the Partial Fulfillment of the Requirements for the Award of Bachelor of Computer Science and Engineering (BCSE)

The thesis has been examined and approved,

_____

Prof. Dr. Utpal Kanti Das
Chairman

_____

Dr. Hasibur Rashid Chayon
Coordinator and Associate Professor

_____

Suhala Lamia

Supervisor and Lecturer

Department of Computer Science and Engineering College of Engineering and Technology IUBAT – International University of Business Agriculture and Technology.

Letter of Transmittal

21 October 2020
The Chairman,
Thesis Defense Committee,
Department of Computer Science and Engineering,
IUBAT–International University of Business Agriculture and Technology,
4 Embankment Drive Road, Sector 10, Uttara Model Town,
Dhaka 1230, Bangladesh.

Subject: Letter of Transmittal.

Dear Sir,

With due respect, it is a great pleasure for us to be able to present the result of our hardship of the thesis report on "Real Time Face-mask Detection with Arduino Based Smart Door". This report is the result of the knowledge which has been acquired from the respective course. We tried our level best for preparing this report. The information of this report is mainly based on the study and research that we have done.

We, therefore, hope that you will find this report worth reading. Please feel free for any query or clarification that you would like us to explain. Hope you will appreciate our work and excuse the minor errors. Thanking you for your cooperation.

Yours sincerely,

Rafiqul Islam Munna	Samiur Rahman
18103116	18103305

Student's Declaration

We declare that this thesis report titled "Real Time Face-mask Detection with Arduino Based Smart Door" has been composed solely by ourselves and that it has not been submitted, in whole or in part, for any other purpose, reward or degree at IUBAT or any other institutions either by me or by any other student. Except where states otherwise by reference or acknowledgment, the work presented is entirely our own. We also declare that there is no plagiarism or data falsification and materials used in this report from various sources have been duly cited.

Rafiqul Islam Munna	Samiur Rahman
18103116	18103305

Supervisor's Certification

This is paper titled "Real Time Face-mask Detection with Arduino Based Smart Door" submitted by the group as mentioned below has been accepted as satisfactory impartial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Engineering in November 2021.

_____

Suhala Lamia

Supervisor and Lecturer
Department of Computer Science and Engineering

IUBAT–International University of Business Agriculture and Technology

Abstract

At present time COVID-19 became the talk of the town. Many peoples are dying by COVID-19. So wearing mask is one of the best way to prevent COVID-19. But many people are not aware to prevent themselves from COVID-19 by violating the rules. For making ourselves and our surrounding safe from COVID-19 we have proposed an Arduino based smart door that will allow to enter a people by the door by checking whether he/she is wearing a mask or not by using image processing and deep learning technologies. Now a day's face mask detection had played a significant progress on the field of image processing and deep learning. There have been a lots of work on face mask detection with different types of models and algorithms. From those we used MobilenetV2 t, tensorflow, Keras, open CV and deep learning for detecting that whether a person is wearing a mask or not and that will give signal to the Arduino. So our system detect mask in real time and for given images and that will automatically open the door if the person is wearing mask otherwise not. That method we have used it's attains 99% accuracy. And the dataset is collected from various sources.

Acknowledgments

We would like to offer our sincere gratitude to our thesis supervisor Suhala Lamia for guiding us on how we should approach the problems and find the most positive solutions accordingly throughout the days. Her guidance and suggestions have been a tremendous value till the last of our work. We are grateful to our parents and some of our beloved teachers who have been always inspiring. They encouraged us to make believe that we have got that potential to do the things in a better manner. We would also like to acknowledge numerous people who shared their ideas and works at discussion sessions.

Table of Content

Table of Figure

List of Table

`

Chapter 1. Introduction

1.1 Background and Context

From 2019, we are suffering from COVID-19 which is caused by corona virus. According to the world health organization 221 countries have been affected by corona. Among 221 countries 23 million peoples have been affected by covid19, 4.7 million peoples have been died and like 0.4 million peoples affected every day [1]. According to our country from 3 January 2020 to till now 1.5 million peoples have been affected and 27k peoples died [2]. So we have to think how to prevent covid19? There are mainly three ways to prevent covid19-

1) Wearing Mask
2) Maintaining social distancing
3) Clean hand regularly. [3]

So we have to make sure that everyone should wear mask when they are being outside. So from that idea we are designing a model which can detect mask by the training through dataset and access to enter any organizations, hospital, market, bus terminals, restaurants, and other public gatherings if they wear a mask.

For training the dataset we have used machine learning with deep learning technology that will preprocess the image by using CNN (Convolutional neural network) and can handle a large number of data at a time.

In the field of image preprocessing, computer vision (open CV) and pattern recognition is the first step to detect a face. And face detection is the very first step for different types of applications that depends on facial analysis algorithms for

identifying and recognizing human faces and also to capturing facial motions in digital images, including the face recognition, face alignment, face verification, age recognition, face modelling, face authentication, access control, forensics, and human-computer interactions [4].

Convolutional neural network (CNN) is composed of multiple building blocks. Convolution layers, pooling layers and fully connected layers. By those layers and backpropagation algorithm it can preprocess and detect objects from image easily [5].

After preprocessing we will use models or algorithms (e.g. Mobile net v2) to detect a mask on face.

There are so many models for detecting the face mask. But all of those models accurecy and effeciency arenot same. So if we use a model that accurecy isnot that much good then there have been some risk to spread corona virue. For that reason we have done some accurecy tests between two popular models. And the higher accurate and efficiency model's has been used in our system.

So the propose model in this paper uses deep learning, tensorflow, open cv, adam, keras which are used for a image classifiers and detect the mask. After detcting the mask it will send a signal to the Arduino uno device and that will controll the open or close of door. By that system we and our socity can remain safe because if eveyone will wear a mask then it helps to prevent COVID-19.

1.2 Motivation

In support of the ongoing fight against this infectious disease, we are motivated to generate new insights, by giving permission who can enter and who can not enter by the facemask detection, using deep learning techniques.

1.3 Objectives

- To check the accuracy of different models for finding which models is much more accurate and efficience.
- Train the dataset that has been collected from different sources.
- To allow a person to enter by a door if he/she is wearing a mask otherwise not.

Chapter 2. Literature Review

P. Narayana, J. Guptha, Deepak and P. Sai explains about the different technologies for mask detection. Like Open CV, MTCNN, CNN, IFTTT, ThingSpea. They used MTCNN via viola jones algorithm to detect the human face. Then they used CNN and MobileNetV2 to detect the mask. So by this way they detect the mask. But the main drawback of their paper is they have not talk much about the door system and they have only test the accuracy for 50 samples. They have mainly focused on only the mask detection [6].

K. N. Baluprithviraj, K. R. Bharathi, S. Chendhuran and P. Lokeshwaran used CNN, Keras and open CV for detecting the mask. They have worked with 1376 images to train their datasets. But the main drawback is they have not shown their accuracy rate and efficiency rate. And they did not talk a lot about their smart door system [7].

S. Teboulbi, S. Messaoud, M. A. Hajjaji and A. Mtibaa worked in face mask and temperature detection. For train their models they have used MobileNet, RestNet, VGG. They have used CXR image. They have tested those models accuracy and those became 93.3%, 89%, 94.52% [8].

S. Sethi, M. Kathuria and T. Kaushik worked in face mask detection. They have worked with ResNet50, AlexNet and MobileNet models. But they mainly focused on ResNet50. In ResNet 50 they have got 98.2% accuracy. They have used MAFA dataset for training the model. So by supervised learning they have trained their model for detecting the mask [9].

P. Nagrath, R. Jain, A. Madan, R. Arora, P. Kataria and J. Hemanth worked with single shot multibox detector system to detect the mask. They have proposed SSDMNV2 model. For introducing the model, they had used Tensorflow, Keras, OpenC, DNN and MobileNetV2. Their system can detect frontal face with mask from image with mask. And their training accuracy became 93% [10].

E. Adhikarla and B. D. Davison talked about face mask detection on real world webcam image. They have use Open CV to access the webcam. After acceding the web cam in real time video its takes an image frame. And by the image frame the system detect that the person is wearing a mask or not. For doing this work they have used COCO (a large scale object detection, segmentation dataset) dataset. And for the model they had used YOLO (You Only Look Once) algorithms [11].

J. M. Shamrat , S. Chakraborty , M. Billah, A. Jubair, S. Islam and R. Ranjari talked about CNN based face mask detection. They had used three deep learning methods for mask detection. They had included Max pooling, Average pooling, and MobileNetV2 architecture. They have got average 98.67% accuracy in Max Pooling,

96.23% accuracy in Average Pooling and 99.82% accuracy in MobileNetV2. In their future work they want to add IOT with their system [12].

K. Kiran, B. V. Kiran, D. C. Sai, G. V. Vamsi and P. R. Salomi talked about Face mask detection by using CNN and deep learning. They have used MovileNetV2 architecture to detect the mask. The main drawback of their paper is they did not show the accuracy and efficiency of their work [13].

H. Adusumalli, Kalyani, K. Sri, Pratapteja and P. Rao made a face mask detector by using Mask R-CNN. They have collected their dataset from Kaggle. By using Keras they have trained their models. By Open CV it takes images and from this image it detects there has a mask or not. And after detecting it's sent a main to authority. But main drawback is it doesn't work for a real time video and they didn't show the accuracy and efficiency rate [14].

Q. V. Truong and . T. N. A. Nguyen developed a face detection model which detects that a whether a person is wearing a mask or not. If not, then the system rings an alarm that someone is without a mask and alerts the peoples of that region. In this model the face has detected by Haar cascade classifier and the face mask is detected by YOLOv3 algorithm. This model accuracy is 90.1%. This accuracy in a real-time environment is very good, hence it is also a good reference model that can be take into consideration before developing our mask detection models [15].

J. Wang and L. Lin used data augmentation, model compressing and transfer learning methods. They have proposed a model of mask detection based on PPYOLO algorithm. In their model single picture mask recognition speed is 11.842ms and real time mask detection accuracy is 86.69%. It shows that their system is pretty good and accurate. This model when compared with RCNN and YOLOv3 the experimental results shows that the detection speed is pretty good than other models and accuracy compared to other models is not that much high [16].

V. Wuttichai and C. Peerasak applied three algorithms called KNN, MobileNetv2 and SVM. They had implemented the mask detection model using all the three algorithms and tried to compare the performance of those models. They had compared recognition time and accuracy and they tried to find which model is working more accurately and efficiently for real-time applications. While testing those models developed by different algorithms, they had found that for the given input video or images MobileNetv2 is more accurate and precise [17].

S Meivel, I. D. K, U. M. S and V. M. J introduced mask detection system by using Matlab. According to authors while integrated with Matlab data set allocation and R-CNN algorithms work more efficiently. Matlan dealt with the Complex pictures with large crowds and low light. The R-CNN algorithm is normally used in complex security and medical systems. A lots of complexities like color changes, contrast changes, brightness changes, balanced face restricting dealt by Matlab. The researchers who are involved during this work have mainly focused on the complexities in these face detection models generally and brought it as a challenge to boost the models efficiency [18].

Chapter 3. Proposed Approach

3.1 Dataset

Our dataset has been collected from the kaggle. The dataset contains 1376 images. The dataset has been used VGGFACE Dataset and after processing it converts into FRGC Dataset. The dataset has been created based on two scenarios. Those are best case and worst case. Dataset is divided in two classes, With mask and without mask. For creating the dataset first of all they take normal images of faces then create a custom CV python script to add face mask on the images. The process of deploying face mask is given in Fig 1.

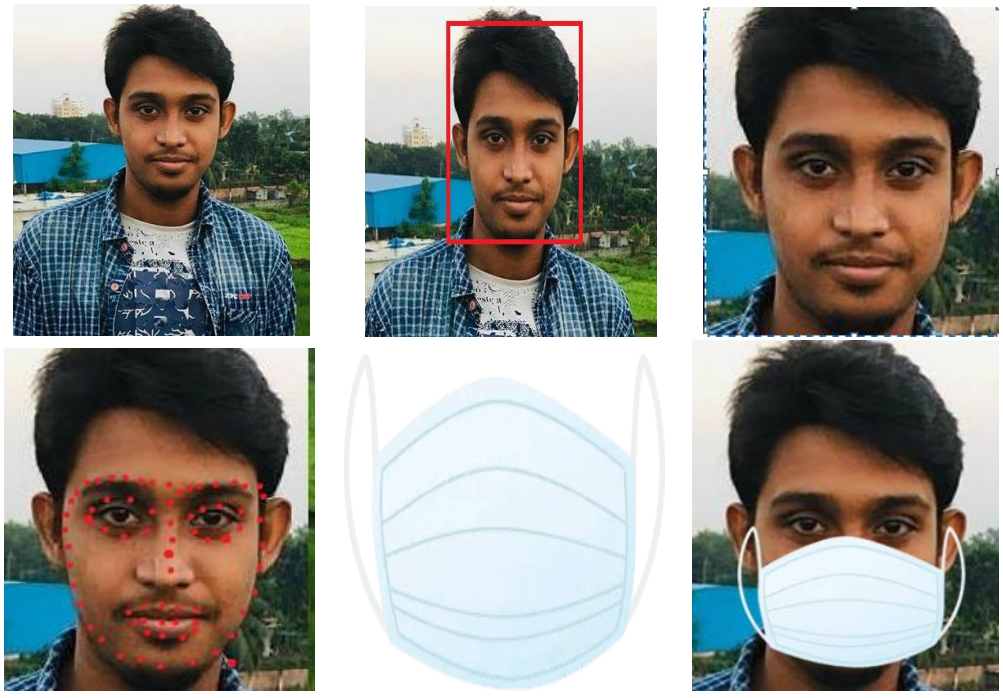


Figure 3.1: Process of Deploying Face mask

3.2 Tensorflow

TensorFlow is a library of python that is open source and free. It was created to perform large numerical computations for deep learning. It supports machine learning which allow to do quick and big numerical computations. This library can be used for a variety of activities, but mainly it's focused on deep neural network inferences and training. Deep learning models can be generated directly using TensorFlow, creat the base library to simplify the process by using wrapper libraries built on top of TensorFlow. TensorFlow enables the creation of dataflow graphs and structures to determine how the data flows through the graph by receiving inputs as a multi-dimensional tensor array. It allows building a flow chart for the inputs which is carried out on the one end and is performed on the other.

3.3 Open CV

Open Source Computer Vision Library (OpenCV) is an open source computer vision and machine learning library, is utilized to differentiate and recognize faces, recognize objects, group movements in recordings, trace progressive modules, follow eye gesture, track camera actions, expel red eyes from pictures taken utilizing flash, find comparative pictures from an image database, perceive landscape and set up markers to overlay it with increased reality and so forth [19] [20]. When all the training and classification is ready to be executed then it needed an eye for the designed system to capture real-time images. Open CV adds intelligence to Deep Learning models for visualization image processing [21].

3.4 Keras

Keras is a deep learning application programming interface(API) which is written in python, running on top of machine learning platform named Tensorflow. It is an approachable, high productive interface for solving the problems of machine learning with a focus on the [22]modern deep learning and image processing. It gives fundamental reflections for creating and transporting of ML arrangements with a high iteration velocity. It takes full advantage of the scalability and cross-platform capabilities of TensorFlow. The core data structures of Keras are layers and models [22]. All the layers used in the CNN model are implemented using Keras. Along with the conversion of the class vector to the binary class matrix in data processing, it helps to compile the overall model [22].

3.5 Arduino

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board. [23].

The System has been done by python that is connected by a USB to micro controller Arduino UNO. And Arduino UNO does not support python code directly. We should execute that by Standard Firmata. For execute it we should follow the directory.

File→Examples→Firmata→StandardFirmata.

For connecting the port, we need PySerial. And PySerial have many features such as:

• Same class based interface on all supported platforms.

• Access to the port settings through Python properties.

• Support for different byte sizes, stop bits, parity and flow control with RTS/CTS and/or Xon/Xoff.

• Working with or without receive timeout.

• File like API with “read” and “write” (“readline” etc. also supported).

• The files in this package are 100% pure Python.

• The port is set up for binary transmission. No NULL byte stripping, CR-LF translation etc. (which are many times enabled for POSIX.) This makes this module universally useful.

Chapter 4. Research Methodology

4.1 System Procedure

The present study explains and analyses naturally. The development of a research methodology helps the researcher to find a systematic approach to the research process. The system has been designed for detecting the faces in a real-time video and to determine whether a person wears a mask or not. Using the data, system can decide that a person can be allowed inside public places or not allowed. This project can be used in any types of public gatherings where the monitoring has to be done.

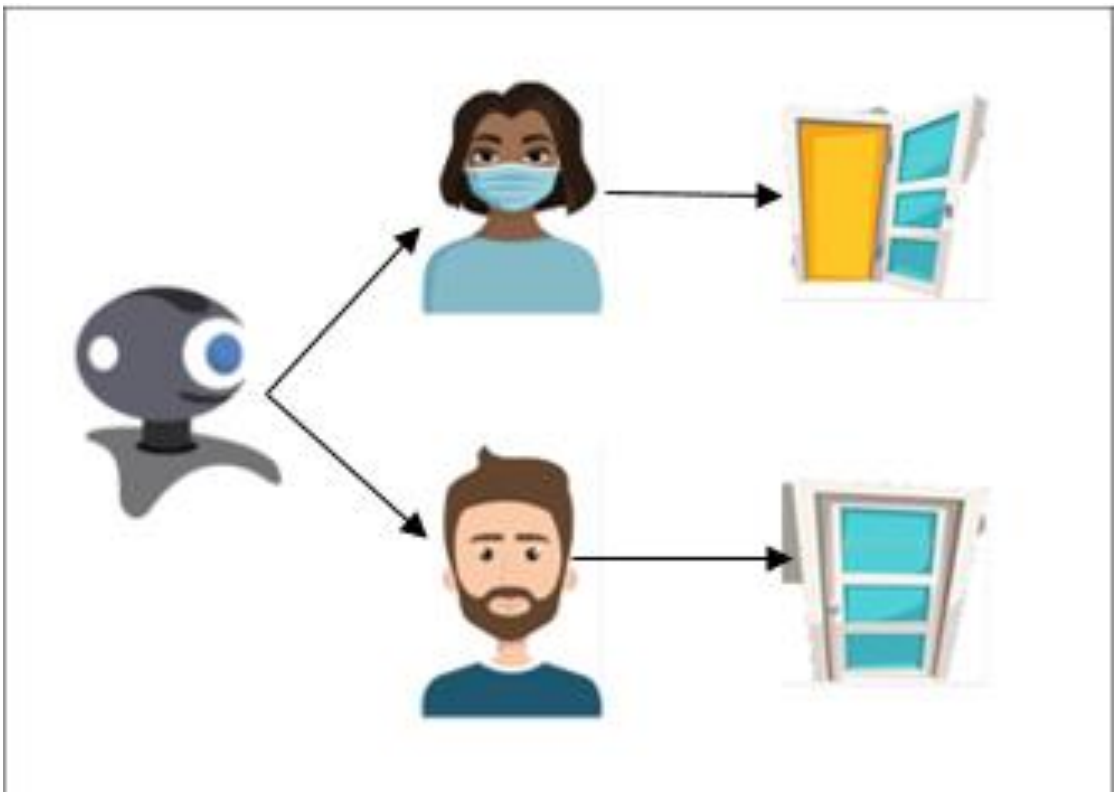


Figure 4.1:Smart Door by Face Mask Detection

This System consists of a camera which will placed in front of door, the camera request Open CV for the video frame which will running on a PC. If the system detects a face it will capture the image of a person, then the detected image will send to the system for testing by Tensorflow that the person wears a face mask or not. After the data processing, System will send the result to Arduino. And the Arduino will command whether to open the door or not.

4.2 System Requirements

The following software's and hardware's requires to run the system.

4.2.1 Software Requirements:

1) Python (3.5 or newer version)
2) Py Serial
3) Tensorflow
4) Keras
5) Imutils
6) Numpy
7) Open CV
8) Matplotlib

4.2.2 Hardware Requirements:

1) Camera
2) Computer
3) Arduino Uno
4) Servo Motor
5) Male to Male wire.

4.3 System Overview

The System Contains two phases

1) Software Phase: It is responsible for detecting the face mask by using tensorflow, open cv, adam, keras and cnn. It also consis two parts
    a) Training phase
    b) Deployment phase.

2) Hardware Phases: It is responsible for opening or closing the door by Arduino. Arduino will control the whole door system. For us It is servo motor.

The algorithmic process is given on the figure

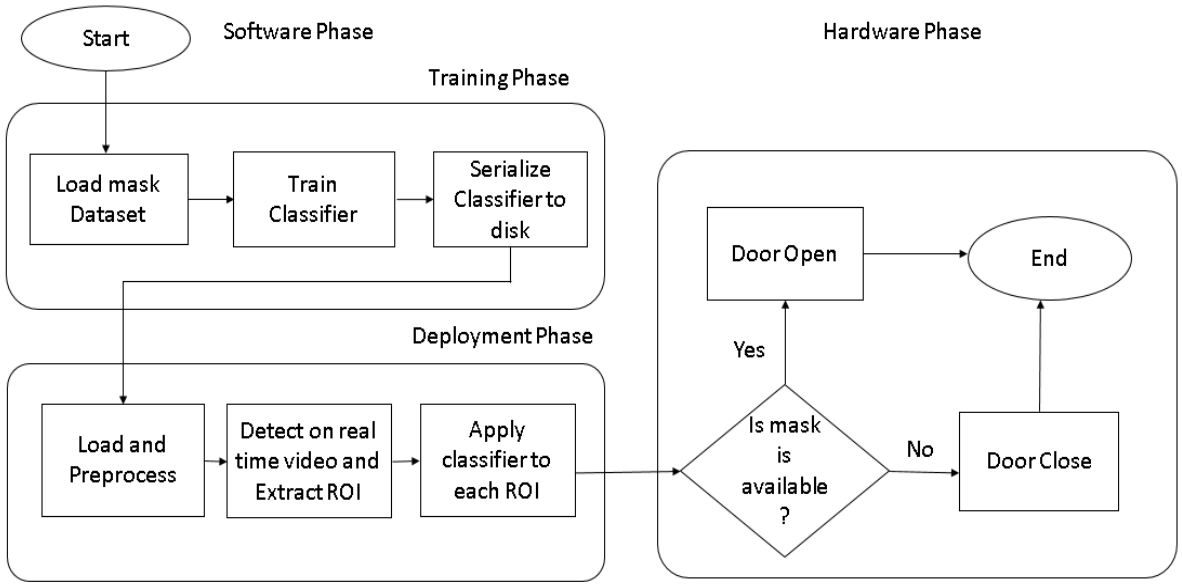


Figure 4.2: Work Flow Diagram

Software Phase:

Training phase:

- The datasets with and without mask are collected.
- Train the facemask classifier using Keras/Tensor Flow and
- Build a model using the datasets.

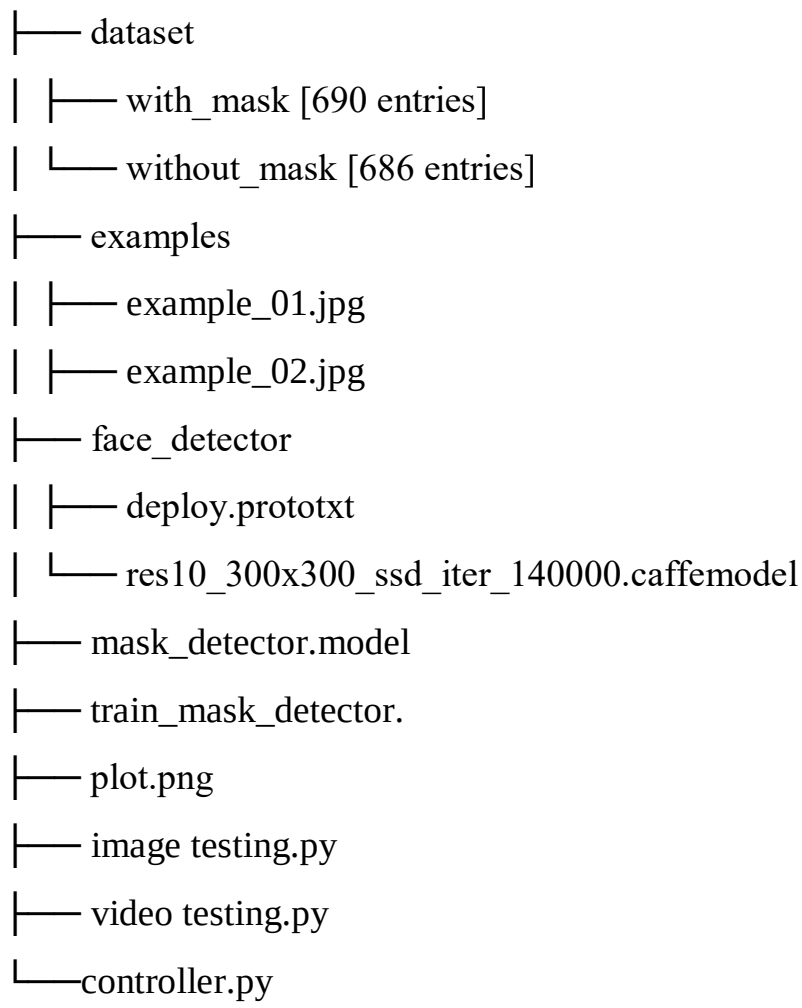
Deployment phase:

- Load the built model.
- Cameras are switched on and the faces are detected using CV2.
- Extract Face ROI in camera video.

Hardware Phase:

- Applying the built model to detect facemask.
- If facemask is not found, the door will not open.
- If facemask is found, the door will open automatically.

The structure of the system given below

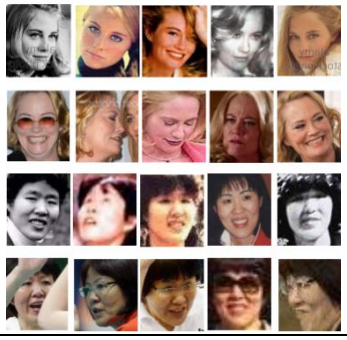


4.4 Dataset Collection

All face detection system uses dataset of images for training and testing, evaluating and comparing the system accuracy in verifying individuals as shown in table

Table 4.1: Different Types of Datasets

Dataset Name	Sample
ORL Dataset	
FERET Dataset	
XM2VTS Dataset	
FRGC Dataset	
LFW Dataset	
CFP Dataset	

IARPA Janus Benchmark-A	
VGGFACE Dataset	
VGGFACE2 Dataset	

Our dataset has been collected from the kaggle. The dataset contains 1376 images. The dataset has been used VGGFACE Dataset and after processing it converts into FRGC Dataset. The dataset has been created based on two scenarios. Those are best case and worst case. Dataset is divided in two classes, With mask and without mask. For creating the dataset first of all they take normal images of faces then create a custom CV python script to add face mask on the images. The example images of our dataset given below-

Figure 4.3: With Mask Dataset

Figure 4.4: Without Mask Dataset

4.5 Pre-processing

The pre-processing phase is the first phase of training and testing the data. There are mainly four steps in pre-processing. They are resizing the images, converting the image into an array, pre-processing input by using the MobileNetV2 algorithm, and performing encoding on labels.

Resizing images is one of the critical pre-processing steps in computer vision (CV) due to maintaining the effectiveness of training models. If the image size becomes smaller, the model will run better. For my system, I resize the images into 224 x 224 pixels.

Then we have to process all the images in my dataset into an array. The image is which is converted into the array we call them by the loop function. After that, the images will be used to pre-process input by using the MobileNetV2 algorithm.

Finally, we have to perform encoding on labels because many machine learning algorithms cannot operate data by labelling directly. They require the input variables and the output variables in a numerical form, including this algorithm. The labelled data are going to be transformed into a numerical label so that the algorithm can understand the labelling and process the data.

4.6 Split the Data

After pre-processing, the data's need to split into two batches, which are training data namely 80 percent, and rest of 20 percent is testing data. Each batch contain both of with-mask and without-mask images.

4.7 Building the Model

The next step is building the model. There are six steps in building the model. Those are for augmentation constructing the training image generator, the base model with MobileNetV2, add model parameters, compiling the model, training the model and saving the model for the future prediction process.

4.8 Training

We have to use keras and tensorflow to train the classifier to detect that whether a person is wearing mask or not. For doing the task we have been used mobilenetv2 architecture, which is highly efficient architecture. It can be used in embedded system. Tensorflow and keras helps us for data augmentation, load the architecture, preprocessing and loading image data. Scikitlearn helps to binarizing the class label, segmentation the dataset and printing the classification report. Imutils helps to find the list or making the list of images in the datasets. And for making the graphs we have been used matplotlib, that helps us to plot the data on the graphs. So by those dependencies we have preprocessed our model and trained our model.

4.9 Testing

To make sure that the model can predict well, there are some steps in testing method. The initial step is making predictions on the testing set. The result for 20 epochs is checking the loss and accuracy rate at the time of training the model.

4.10 Implementing the model

The model implemented for image and real time video. For the image testing there has been given some image on project repertory. From those picture it detects that there is a mask or not in face. If mask has been detected, then the box color became green and there has some percentage. And If there is no mask then there arise a red box with some percentage.

For the video testing it test frame by frame. By using the OpenCV its read frame by frame and detect whether there is a mask or not.

4.11 Circuit design:

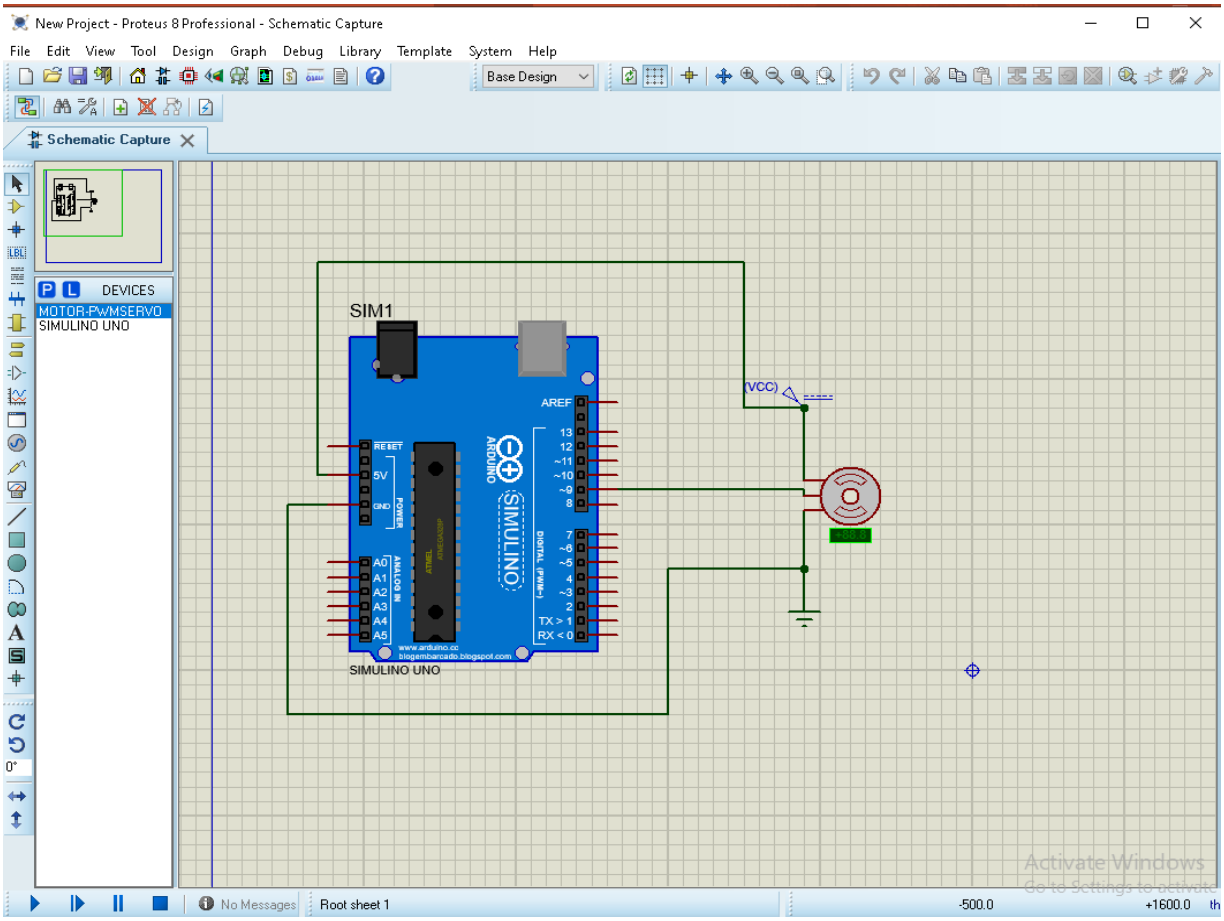


Figure 4.5: Circuit Diagram for the System

For the circuit design we have used

1) Arduino Uno
2) Servo Motor SG90
3) 3 males to male wire.

One wire connected the yellow wire point of the servo motor with 10 no pin of Arduino Uno.

Another wire connected the red wire point of the servo motor with 5V source of Arduino Uno.

And the last wire connected the black wire point of the servo motor with ground source of Arduino Uno.

4.12 Circuit Control:

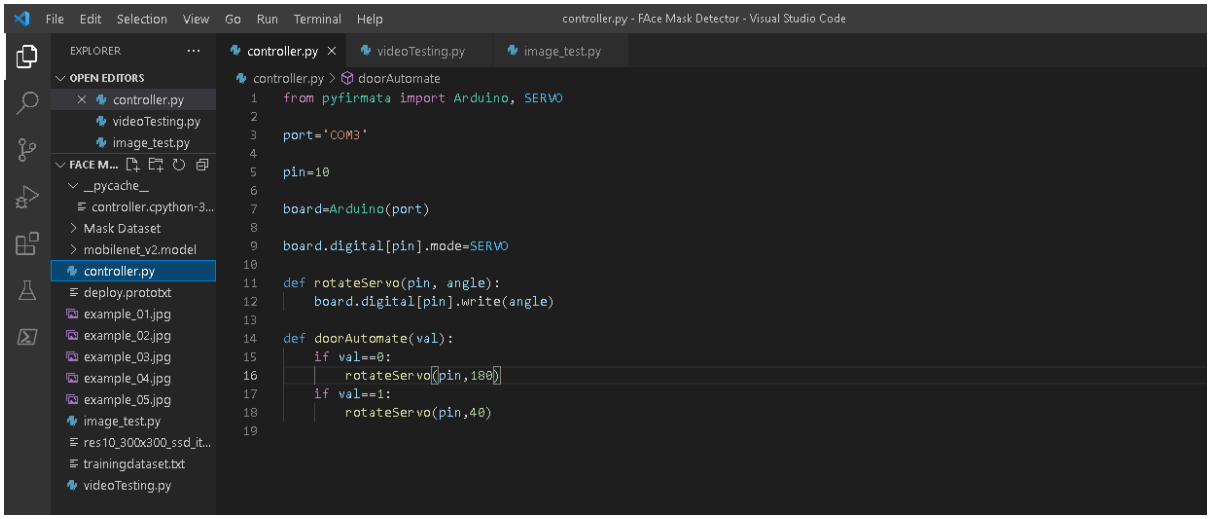


Figure 4.6: Code for Circuit Control

For controlling the Arduino, we have used that python code. But the python code does not directly execute on the Arduino. So for that we need PyFirmata. And we will import Arduino and servo from that library.

Now for controlling the Arduino Uno we have to download the Firmata from the Arduino software and for connecting we have to go to

File→Examples→Firmata→StandardFirmata

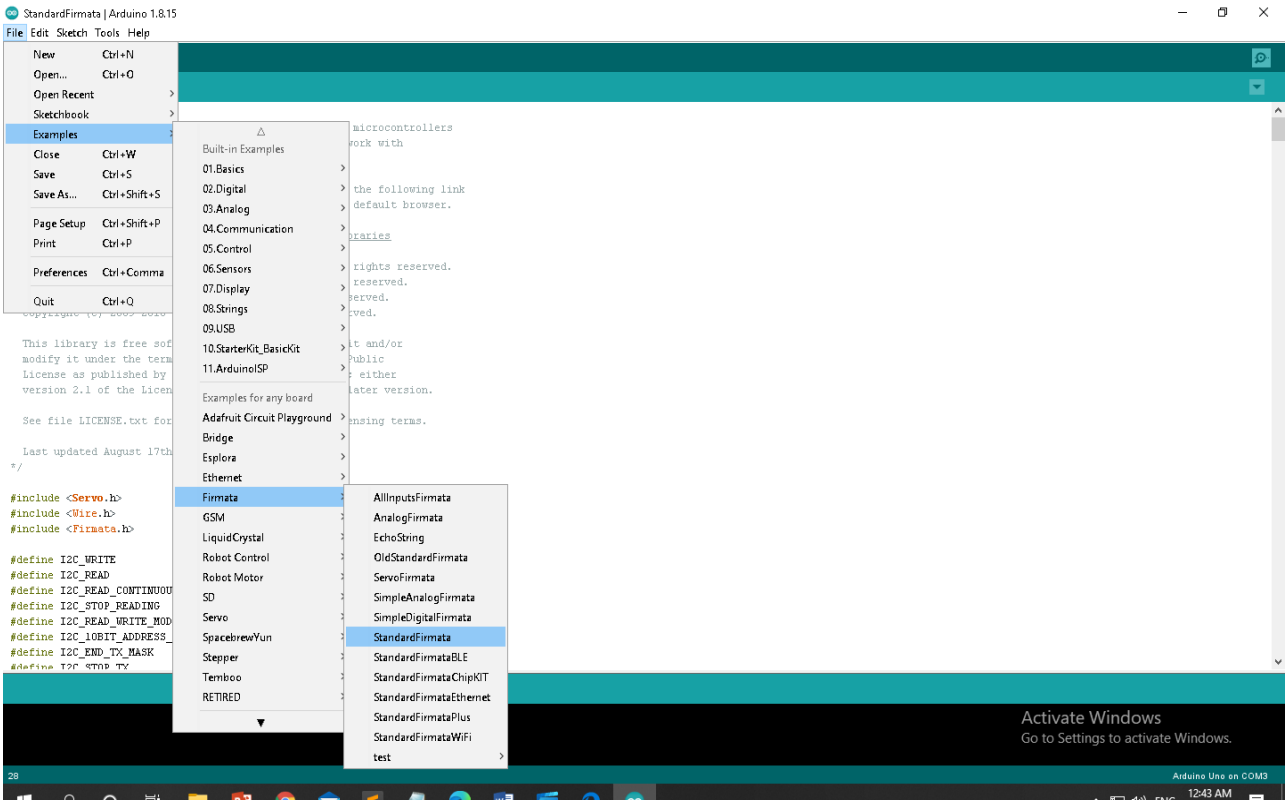


Figure 4.7: Arduino for Standard Firmata

After Executing we have to go to

Tools→ports

And check the serial port and check it with the python code port variable.

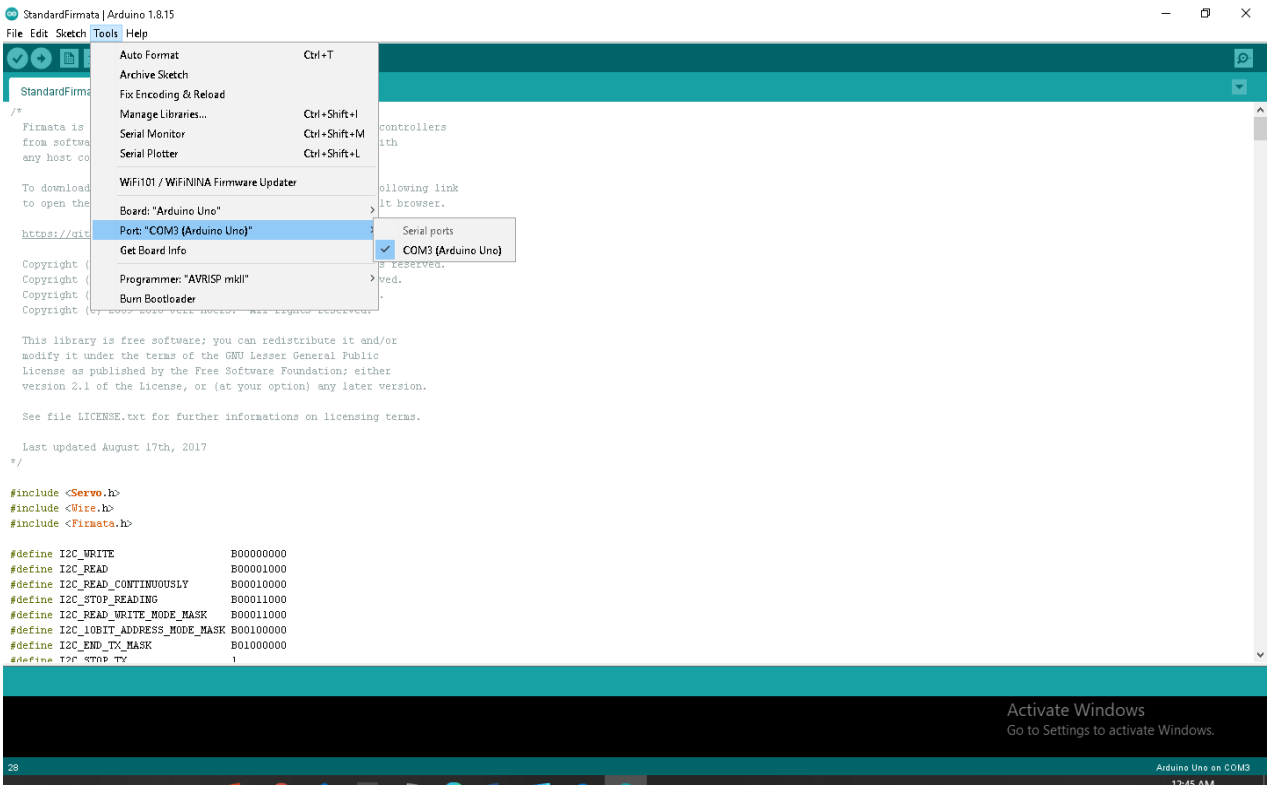


Figure 4.8: Arduino for Finding Port

Then we have to compile the code and burn the code on bootloader.

That's the methodology We have used for our research.

Chapter 5. Result and Discussion

5.1 Accuracy Testing:
we have tasted the accuracy test between two models. Those are

1) Mobile Net V2

2) Viola Jones

The accuracy test results images are given below:

5.1.1 Mobile Net V2:

```
91/91 [==============================] - 41s 455ms/step - loss: 0.0681 - accuracy: 0.9761 - val_loss: 0.0353 - val_accurac
y: 0.9855
Epoch 4/20
91/91 [==============================] - 37s 402ms/step - loss: 0.0459 - accuracy: 0.9844 - val_loss: 0.0471 - val_accurac
y: 0.9783
Epoch 5/20
91/91 [==============================] - 36s 397ms/step - loss: 0.0439 - accuracy: 0.9835 - val_loss: 0.0370 - val_accurac
y: 0.9819
Epoch 6/20
91/91 [==============================] - 33s 364ms/step - loss: 0.0309 - accuracy: 0.9890 - val_loss: 0.0601 - val_accurac
y: 0.9746
Epoch 7/20
91/91 [==============================] - 29s 318ms/step - loss: 0.0345 - accuracy: 0.9890 - val_loss: 0.0344 - val_accurac
y: 0.9819
Epoch 8/20
91/91 [==============================] - 34s 369ms/step - loss: 0.0243 - accuracy: 0.9899 - val_loss: 0.0304 - val_accurac
y: 0.9819
Epoch 9/20
91/91 [==============================] - 39s 427ms/step - loss: 0.0312 - accuracy: 0.9908 - val_loss: 0.0456 - val_accurac
y: 0.9819
Epoch 10/20
91/91 [==============================] - 36s 397ms/step - loss: 0.0189 - accuracy: 0.9926 - val_loss: 0.0550 - val_accurac
y: 0.9746
Epoch 11/20
91/91 [==============================] - 33s 360ms/step - loss: 0.0252 - accuracy: 0.9890 - val_loss: 0.0483 - val_accurac
y: 0.9746
Epoch 12/20
91/91 [==============================] - 34s 374ms/step - loss: 0.0262 - accuracy: 0.9908 - val_loss: 0.0439 - val_accurac
y: 0.9819
Epoch 13/20
91/91 [==============================] - 31s 335ms/step - loss: 0.0140 - accuracy: 0.9945 - val_loss: 0.0427 - val_accurac
y: 0.9928
Epoch 14/20
91/91 [==============================] - 33s 367ms/step - loss: 0.0216 - accuracy: 0.9926 - val_loss: 0.0376 - val_accurac
y: 0.9855
Epoch 15/20
91/91 [==============================] - 36s 397ms/step - loss: 0.0248 - accuracy: 0.9899 - val_loss: 0.0402 - val_accurac
y: 0.9819
Epoch 16/20
91/91 [==============================] - 34s 378ms/step - loss: 0.0232 - accuracy: 0.9899 - val_loss: 0.0425 - val_accurac
y: 0.9855
Epoch 17/20
91/91 [==============================] - 32s 350ms/step - loss: 0.0252 - accuracy: 0.9908 - val_loss: 0.0535 - val_accurac
y: 0.9783
Epoch 18/20
91/91 [==============================] - 32s 346ms/step - loss: 0.0168 - accuracy: 0.9926 - val_loss: 0.0487 - val_accurac
y: 0.9855
Epoch 19/20
91/91 [==============================] - 31s 339ms/step - loss: 0.0222 - accuracy: 0.9899 - val_loss: 0.0713 - val_accurac
y: 0.9746
Epoch 20/20
91/91 [==============================] - 30s 331ms/step - loss: 0.0255 - accuracy: 0.9899 - val_loss: 0.0453 - val_accurac
y: 0.9855
```

Figure 5.1: MobileNetV2 Epoch Testing

We have divided our dataset testing in 20 epoch. In every epoch there has been tested in 91 iteration. From the first epoch we can see that the accurcy for the testing 97.61% and the data loss percentage became 6.81%. In the second epoch we can see that the accurecy rate increase and the data loss percentage became decreasing. For the second epoch the data accurecy rise 97.6 to 98.4% and data loss percentage became 4.59%. For every epoch the data accurecy is rising rapidly and the data loss percentage became decreasing. At the last epoch the data accuracy became 99% and data loss percentage became 4.53%.

	precision	recall	f1-score	support
with_mask	0.99	0.99	0.99	138
without_mask	0.99	0.99	0.99	138
accuracy			0.99	276
macro avg	0.99	0.99	0.99	276
weighted avg	0.99	0.99	0.99	276

Figure 5.2: MobileNetV2 Testing Accuracy

From Fig 5.2 we see the testing accuracy result for the algorithm. We can see that the f1 accuracy result is 99% it is for both with mask and without mask dataset. For the both dataset the recall and precision accuracy is also 99%. So from that picture we can say that the overall testing accuracy rate is 99%.

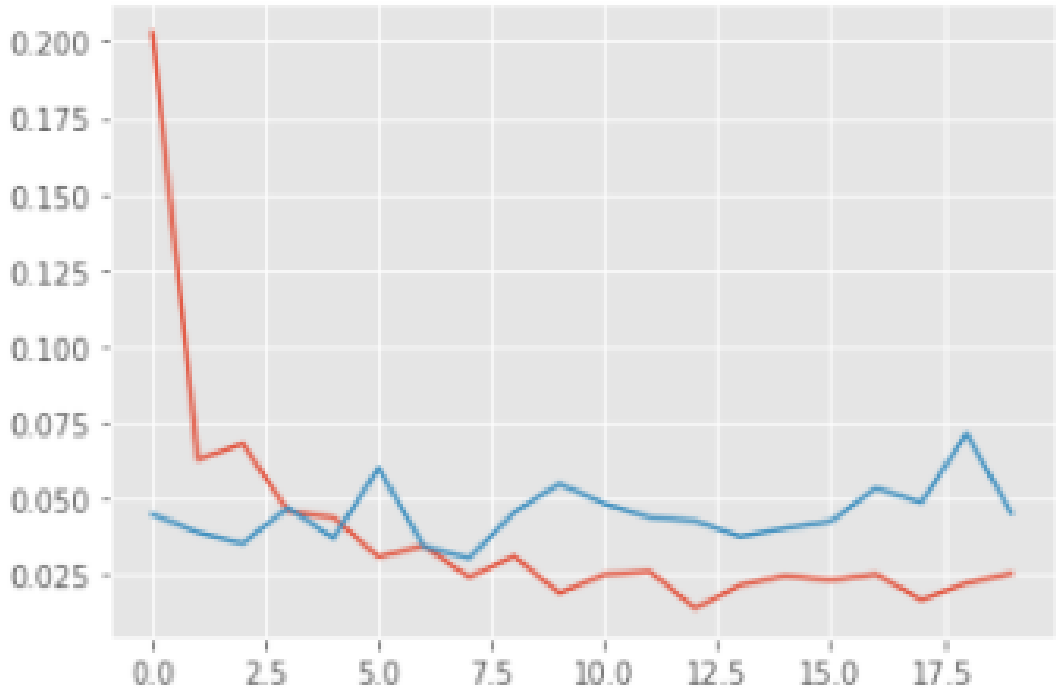


Figure 5.3: MobileNetV2 Data Loss Graph

This is the data loss diagram. The horizontal row shows the times in microsecond and the vertical row shows the percentage value of data loss. The blue line shows over time how data loss is increasing or decreasing. And the red line shows over time how the value loss is increasing or decreasing. From the graph we can see that overtime the value loss has been decreased and also over time the data loss is increasing and decreasing. Finally, we can see the data loss is less than 5%.

5.1.2 Viola Jones:

```
[INFO] training head...
Epoch 1/20
34/34 [==============================] - 2s 38ms/step - loss: 0.6617 - accuracy: 0.5824 - val_loss: 0.5208 - val_accuracy: 0.7754
Epoch 2/20
34/34 [==============================] - 1s 26ms/step - loss: 0.5443 - accuracy: 0.7260 - val_loss: 0.4706 - val_accuracy: 0.8007
Epoch 3/20
34/34 [==============================] - 1s 26ms/step - loss: 0.5141 - accuracy: 0.7358 - val_loss: 0.4226 - val_accuracy: 0.8261
Epoch 4/20
34/34 [==============================] - 1s 26ms/step - loss: 0.5174 - accuracy: 0.7495 - val_loss: 0.3931 - val_accuracy: 0.8478
Epoch 5/20
34/34 [==============================] - 1s 26ms/step - loss: 0.4915 - accuracy: 0.7595 - val_loss: 0.4271 - val_accuracy: 0.8225
Epoch 6/20
34/34 [==============================] - 1s 26ms/step - loss: 0.4808 - accuracy: 0.8007 - val_loss: 0.4050 - val_accuracy: 0.8297
Epoch 7/20
34/34 [==============================] - 1s 26ms/step - loss: 0.4952 - accuracy: 0.7420 - val_loss: 0.4331 - val_accuracy: 0.8080
Epoch 8/20
34/34 [==============================] - 1s 25ms/step - loss: 0.4767 - accuracy: 0.7662 - val_loss: 0.4420 - val_accuracy: 0.7826
Epoch 9/20
34/34 [==============================] - 1s 26ms/step - loss: 0.4668 - accuracy: 0.7874 - val_loss: 0.3803 - val_accuracy: 0.8442
Epoch 10/20
34/34 [==============================] - 1s 26ms/step - loss: 0.4101 - accuracy: 0.8150 - val_loss: 0.3414 - val_accuracy: 0.8587
Epoch 11/20
34/34 [==============================] - 1s 26ms/step - loss: 0.4162 - accuracy: 0.8105 - val_loss: 0.3343 - val_accuracy: 0.8696
Epoch 12/20
34/34 [==============================] - 1s 26ms/step - loss: 0.3867 - accuracy: 0.8170 - val_loss: 0.3160 - val_accuracy: 0.8804
Epoch 13/20
34/34 [==============================] - 1s 26ms/step - loss: 0.4498 - accuracy: 0.7856 - val_loss: 0.3166 - val_accuracy: 0.8659
Epoch 14/20
34/34 [==============================] - 1s 26ms/step - loss: 0.4224 - accuracy: 0.7983 - val_loss: 0.3233 - val_accuracy: 0.8696
Epoch 15/20
34/34 [==============================] - 1s 26ms/step - loss: 0.3940 - accuracy: 0.8303 - val_loss: 0.3094 - val_accuracy: 0.8659
Epoch 16/20
34/34 [==============================] - 1s 26ms/step - loss: 0.3829 - accuracy: 0.8473 - val_loss: 0.3057 - val_accuracy: 0.8768
Epoch 17/20
34/34 [==============================] - 1s 26ms/step - loss: 0.3553 - accuracy: 0.8449 - val_loss: 0.2759 - val_accuracy: 0.8913
Epoch 18/20
34/34 [==============================] - 1s 27ms/step - loss: 0.3279 - accuracy: 0.8689 - val_loss: 0.2695 - val_accuracy: 0.9022
Epoch 19/20
34/34 [==============================] - 1s 26ms/step - loss: 0.3480 - accuracy: 0.8356 - val_loss: 0.2960 - val_accuracy: 0.8804
Epoch 20/20
34/34 [==============================] - 1s 26ms/step - loss: 0.3282 - accuracy: 0.8655 - val_loss: 0.2756 - val_accuracy: 0.8804
```

Figure 5.4: Viola Jones Epoch Testing

Here also, we have divided our dataset testing in 20 epoch. In every epoch there has been tested in 34 iteration. From the first epoch we can see that the accurcy for the testing 58.24% and the data loss percentage became 66.17%. In the second epoch we can see that the accurecy rate increase and the data loss percentage became decreasing. For the second epoch the data accurecy rise 58.24 to 72.6% and data loss percentage became 54.43%. For every epoch the data accurecy is rising rapidly and the data loss percentage became decreasing. At the last epoch the data accuracy became 86.55% and data loss percentage became 32.82%.

```
[INFO] evaluating network...
                              precision    recall  f1-score   support

   ./dataset_1/with_mask       0.90      0.86      0.88       138
./dataset_1/without_mask       0.86      0.91      0.88       138

                accuracy                           0.88       276
               macro avg       0.88      0.88      0.88       276
            weighted avg       0.88      0.88      0.88       276
```

Figure 5.5: Viola Jones Testing Accuracy

From Fig 5.5 we see the testing accuracy result for the algorithm. We can see that the f1 accuracy result is 88% it is for both with mask and without mask dataset. For with mask dataset the recall accuracy is 86% and for without mask the recall accuracy is 91% and for with mask dataset the precision accuracy is also 90% and for the without mask the accuracy is 86%. So from that picture we can say that the overall testing accuracy rate is 88%.

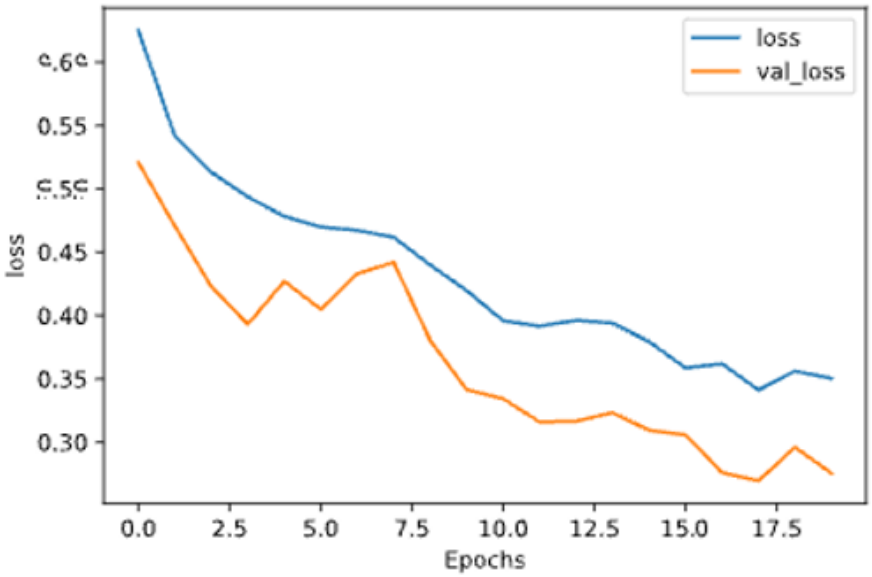


Figure 5.6: Viola Jones Data Loss Graph

This is the data loss diagram. The horizontal row shows the times in microsecond and the vertical row shows the percentage value of data loss. The blue line shows over time how data loss is increasing or decreasing. And the red line shows over time how the value loss is increasing or decreasing. From the graph we can see that overtime the value loss has been decreased and also over time the data loss is increasing and decreasing.

5.2 Implemented methods Output

From those figure, we can see the clear difference between those two model. 1st model accuracy rate is much more different from the 2nd model. The accuracy table is given below in table

Table 5.1: Difference Between MobileNetV2 and Viola Jones

Mobile Net V2		Viola Jones	
Data Training Accuracy	98.99%	Data Training Accuracy	86.55%
Data Training Losses	4.53%	Data Training Losses	27.56%
F1 Accuracy	99%	F1 Accuracy	88%
Macro and Weighted Score	99%	Macro and Weighted Score	88%

So, from that result we can see that the accuracy rate of mobilenetv2 is better than the other model. And it is more compatible. So we have used mobile net v2 for the smart door management.

We assume the servo motor as a door. If it rotates in 180 degrees that means the door is close and if the servo rotates in 40 degrees that means the door is open. so if the system detect that a person is wearing a mask then the servo will rotate in 180 degrees or a person is not wearing a mask then the servo will rotate in 40 degrees.

The system can work for images and real time videos. So the resultant parts images are given below-

5.2.1 Image testing without mask:

Figure 5.7: System Testing for Non-Mask Face for Image

5.2.2 Image testing with mask:

Figure 5.8:System Testing for With Mask Face for Image

In figure 5.7 and 5.8 we can see that for image if the system can detect the is no mask then the servo motor rotates in 180 degrees that means the door is close. And if the

system can detect that there is mask in the image person face then the servo motor rotate in 40 degrees. That means the door is open. Now the person can enter the place.

5.2.3 video testing without mask:

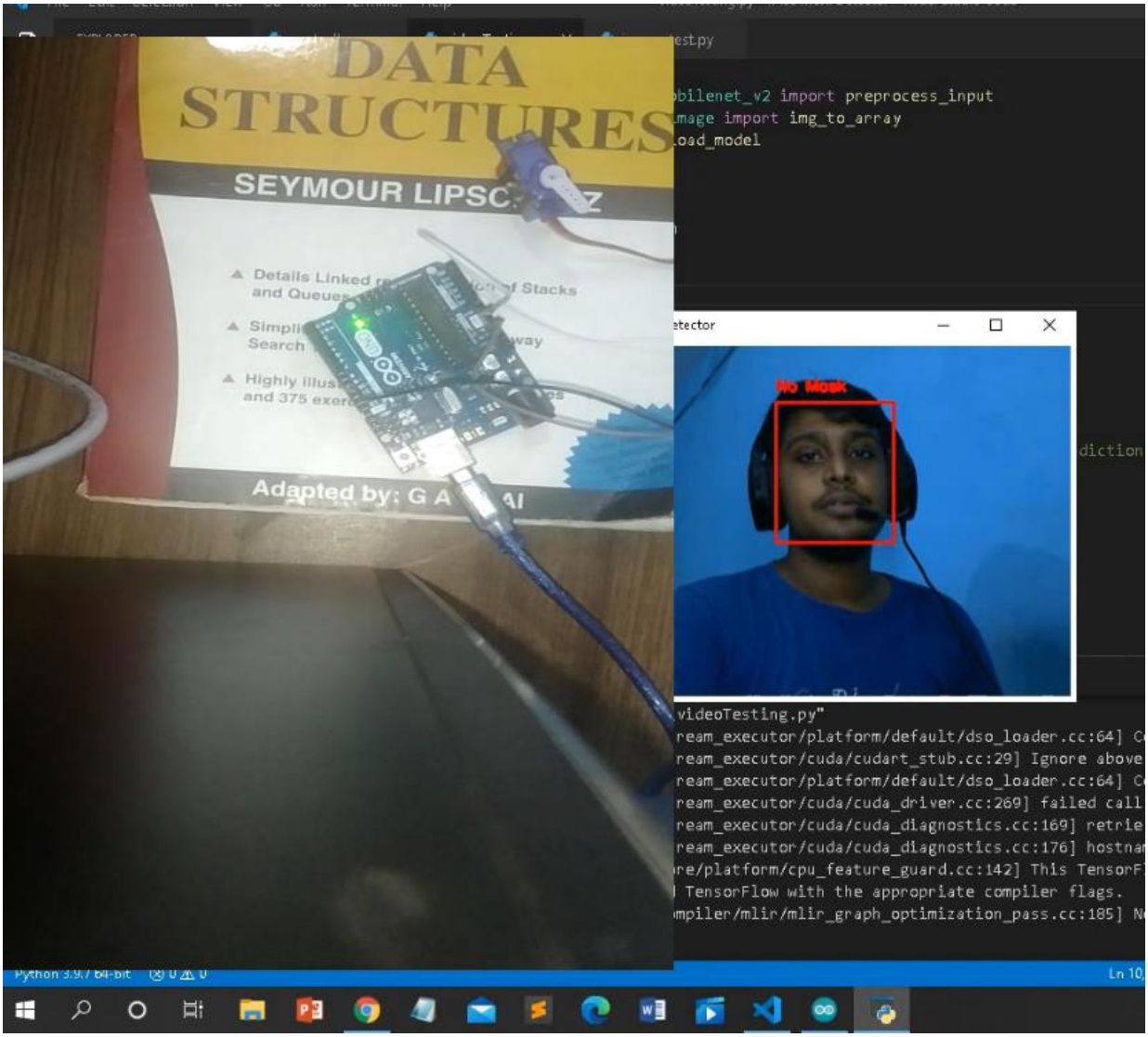


Figure 5.9: System Testing for Non-Mask Face for Video

5.2.4 video testing with mask:

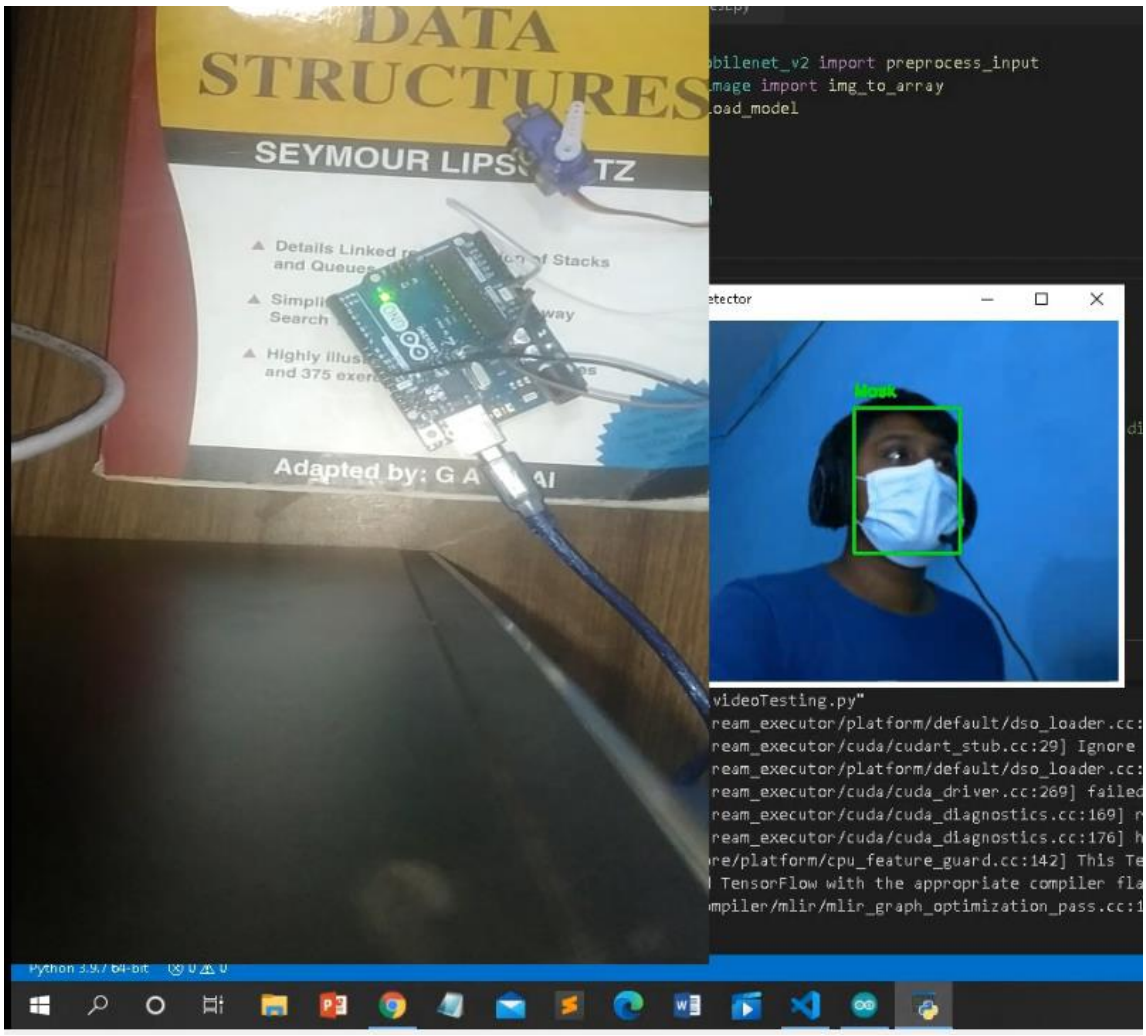


Figure 5.10: System Testing for With Mask Face for Video

For the real time video testing its detects frame by frame. For real time video the servo motor is always in 180 degrees' position. If anyone wants to enter by the door with a mask, then the motor rotates in 40 degrees and that means the door is open. And when the person passes the camera then the servo back its previous position in 180 degree. And if it can detect that the person isn't wearing a mask then it remains same place in 180 degree. That we can see from fig 5.9 and 5.10.

Chapter 6. Conclusion

6.1 Conclusion

For preventing COVID-19 we all should work together. For preventing the virus, it is a small effort from us. This system can be used for gathering places like Mall, School, Office etc. The motivation for doing the work comes from seeing the present situation. Many peoples are disobeying the rules and regulation that is given to them for preventing COVID-19. So our system will not allow to enter them at their desiring place. And by that way they cannot spread COVID-19. Our system accuracy become 99%. If anyone want to reuse this project idea with some changes and new features, then they can make it possible. It will be considerable as an open source system so that anyone can make a very good use of it because our concern is about preventing the COVID-19.

6.2 Limitation

Every work is not perfect. So our work is also not perfect. It's has some limitations. First of all, for our system we have used a servo motor instead of a real door for simulation. So we consider this as a limitation for this project. Then our system has some problem to detect the face mask on low light. If the lighting is very low, then the system faces some problem to detect the mask. If anyone is with a mask in low light sometime the system detects there is no mask in his/her face and the door remain close. It's face some problem to detect a continuous movement face. If anyone move so fast, then the system face a problem to detect if the person is wearing a mask or not.

6.3 Future Work

In future we will upgrade our work. We will try to add the system with database. So for a specific office if anyone doesn't maintain the COVID-19 rules then the system will send a notification to authority that who isn't maintaining the COVID-19 rules. we will also try to make the system more accurate and more efficient. And we will also try to add retina scanning and monitor the body temperature.

References

WHO, "World Health Organization," [Online]. Available: https://covid19.who.int/?gclid=CjwKCAjwhuCKBhADEiwA1HegOTmUKwinEMib961rcCKHLciOVokb6DUNHJKABroiWm3_IuuaaHzv6BoCIO8QAvD_BwE. [Accessed 3 10 2021].

WHO, 3 1 2020. [Online]. Available: https://covid19.who.int/region/searo/country/bd. [Accessed 1 10 2021].

N. Iftikhar, "healthline," 4 6 2020. [Online]. Available: https://www.healthline.com/health/coronavirus-prevention. [Accessed 1 10 2021].

R. I. R. M. S. A. P.S. Othman, "Image Processing Techniques for Identifying Impostor Documents Through Digital Forensic Examination," *Image Process. Tech.*, p. 62, 2020.

R. N. M. D. R. e. a. Yamashita, "Convolutional neural networks: an overview and application in radiology," *Insights Imaging*, 2018.

P. Narayana, J. Guptha, Deepak and P. Sai, "Smart Door / COVID-19 Face Mask Detection," *International Journal of Innovative Technology and Exploring Engineering*, vol. 10, no. 9, 2021.

K. N. Baluprithviraj, K. R. Bharathi, S. Chendhuran and P. Lokeshwaran, "Artificial Intelligence based Smart Door with Face Mask Detection," *IEEE*, 2021.

S. Teboulbi, S. Messaoud, M. A. Hajjaji and A. Mtibaa, "Real-Time Implementation of AI-Based Face Mask Detection and Social Distancing

Measuring System for COVID-19 Prevention," *Hindawi*, vol. 2021, 2021.

S. Sethi, M. Kathuria and T. Kaushik, "Face mask detection using deep learning: An approach to reduce risk of Coronavirus spread," *National Center for Biotechnology Information*, 2021.

P. Nagrath, R. Jain, A. Madan, R. Arora, P. Kataria and J. Hemanth,

"SSDMNV2: A real time DNN-based face mask detection system using single 0 shot multibox detector and MobileNetV2," *Sustainable Cities and Society*, vol. 66, p. 102692, 2020.

E. Adhikarla and B. D. Davison, "Face Mask Detection on Real-World Webcam Images," *GoodIT*, pp. 139-144, 2021.

J. M. Shamrat , S. Chakraborty , M. Billah, A. Jubair, S. Islam and R. Ranjari,

"Face Mask Detection using Convolutional Neural Network (CNN) to reduce the spread of Covid-19," in *5th International Conference on Trends in Electronics and Informatics (ICOEI 2021)*, Tirunelveli, India, 2021.

K. Kiran, B. V. Kiran, D. C. Sai, G. V. Vamsi and P. R. Salomi, "Face Mask Detection Using Machine Learning," *SSPN*, p. 7, 2021.

H. Adusumalli, Kalyani, K. Sri, Pratapteja and P. Rao, "Face Mask Detection

Using OpenCV," in *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, India, 2021.

Q. V. Truong and . T. N. A. Nguyen , "Real-Time Face Mask Detector Using YOLOv3 Algorithm And Haar Cascade Classifier," in *International Conference on Advanced Computing And Applications (ACOMP)*, 2021.

J. Wang and L. Lin, "Face mask detection based on Transfer learning and PP YOLO," in *2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, 2021.

V. Wuttichai and C. Peerasak, "Study of the Performance of Machine Learning Algorithms for Face Mask Detection," *IEEE,* 2020.

S Meivel, I. D. K, U. M. S and V. M. J, "Real time data analysis of face mask detection and social distance measurement using Matlab," in *ELSEVIER 2020*, 2021.

D. Someshwar, D. Bhanushali, V. Chaudhari and S. Nadkarni, "Implementation of Virtual Assistant with Sign Language using Deep Learning and TensorFlow," *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA),* 2020.

H. Lai, "Real-Time Dynamic Hand Gesture Recognition," in *2014 International Symposium on Computer, Consumer and Control*, 2021.

R. R. Asaad, "Review on Deep Learning and Neural Network Implementation for Emotions Recognition," *Qubahan Academic Journal,* vol. 1, 2021.

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

B. Mohammed and H. Ahmad, ""Advanced car-parking security platform using Arduino along with automatic license and number recognition," *Academic Journal of Nawroz University*, vol. 10, p. 1, 2021.

Appendix

A. Source Code for Importing Libraries

```
In [4]: import numpy as np
        import os
        import matplotlib.pyplot as plt
        from imutils import paths

        from tensorflow.keras.applications import MobileNetV2
        from tensorflow.keras.layers import AveragePooling2D
        from tensorflow.keras.layers import Dropout
        from tensorflow.keras.layers import Flatten
        from tensorflow.keras.layers import Dense
        from tensorflow.keras.layers import Input
        from tensorflow.keras.models import Model
        from tensorflow.keras.optimizers import Adam
        from tensorflow.keras.applications.mobilenet_v2 import preprocess_input

        from tensorflow.keras.preprocessing.image import ImageDataGenerator
        from tensorflow.keras.preprocessing.image import img_to_array
        from tensorflow.keras.preprocessing.image import load_img
        from tensorflow.keras.utils import to_categorical
        from sklearn.preprocessing import LabelBinarizer
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import classification_report
```

B. Source Code for Training Dataset

```
train_X,test_X,train_Y,test_Y=train_test_split(data,labels,test_size=0.20,stratify=labels,random_state=10)

aug=ImageDataGenerator(rotation_range=20,zoom_range=0.15,width_shift_range=0.2,height_shift_range=0.2,shear_range=0.15,horiz
```

```
baseModel=MobileNetV2(weights='imagenet',include_top=False,input_tensor=Input(shape=(224,224,3)))
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [96, 128, 160, 192, 224]. Weights for inp
ut shape (224, 224) will be loaded as the default.
```

C. Source Code for Layering Model

```
In [11]: for layer in baseModel.layers:
             layer.trainable=False


         print(model.summary())
```

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 224, 224, 3) 0
__________________________________________________________________________________________________
Conv1 (Conv2D)                  (None, 112, 112, 32) 864         input_1[0][0]
__________________________________________________________________________________________________
bn_Conv1 (BatchNormalization)   (None, 112, 112, 32) 128         Conv1[0][0]
__________________________________________________________________________________________________
```

D. Source Code For Training the Epoch

```
In [ ]: H=model.fit(
            aug.flow(train_X,train_Y,batch_size=BS),
            steps_per_epoch=len(train_X)//BS,
            validation_data=(test_X,test_Y),
            validation_steps=len(test_X)//BS,
            epochs=Epochs
        )
```

```
Epoch 1/20
91/91 [==============================] - 33s 327ms/step - loss: 0.2601 - accuracy: 0.9044 - val_loss: 0.0565 - val_accurac
y: 0.9819
Epoch 2/20
91/91 [==============================] - 31s 335ms/step - loss: 0.0740 - accuracy: 0.9770 - val_loss: 0.0446 - val_accurac
y: 0.9819
Epoch 3/20
91/91 [==============================] - 35s 387ms/step - loss: 0.0551 - accuracy: 0.9789 - val_loss: 0.0355 - val_accurac
y: 0.9855
Epoch 4/20
91/91 [==============================] - 35s 381ms/step - loss: 0.0462 - accuracy: 0.9862 - val_loss: 0.0394 - val_accurac
y: 0.9783
Epoch 5/20
91/91 [==============================] - 32s 356ms/step - loss: 0.0470 - accuracy: 0.9835 - val_loss: 0.0335 - val_accurac
y: 0.9855
Epoch 6/20
91/91 [==============================] - 28s 312ms/step - loss: 0.0332 - accuracy: 0.9871 - val_loss: 0.0335 - val_accurac
y: 0.9819
Epoch 7/20
91/91 [==============================] - 30s 334ms/step - loss: 0.0289 - accuracy: 0.9881 - val_loss: 0.0408 - val_accurac
y: 0.9891
Epoch 8/20
91/91 [==============================] - 35s 380ms/step - loss: 0.0259 - accuracy: 0.9917 - val_loss: 0.0355 - val_accurac
y: 0.9855
Epoch 9/20
91/91 [==============================] - 32s 353ms/step - loss: 0.0258 - accuracy: 0.9890 - val_loss: 0.0528 - val_accurac
y: 0.9783
```


E. Source Code for Classification Report

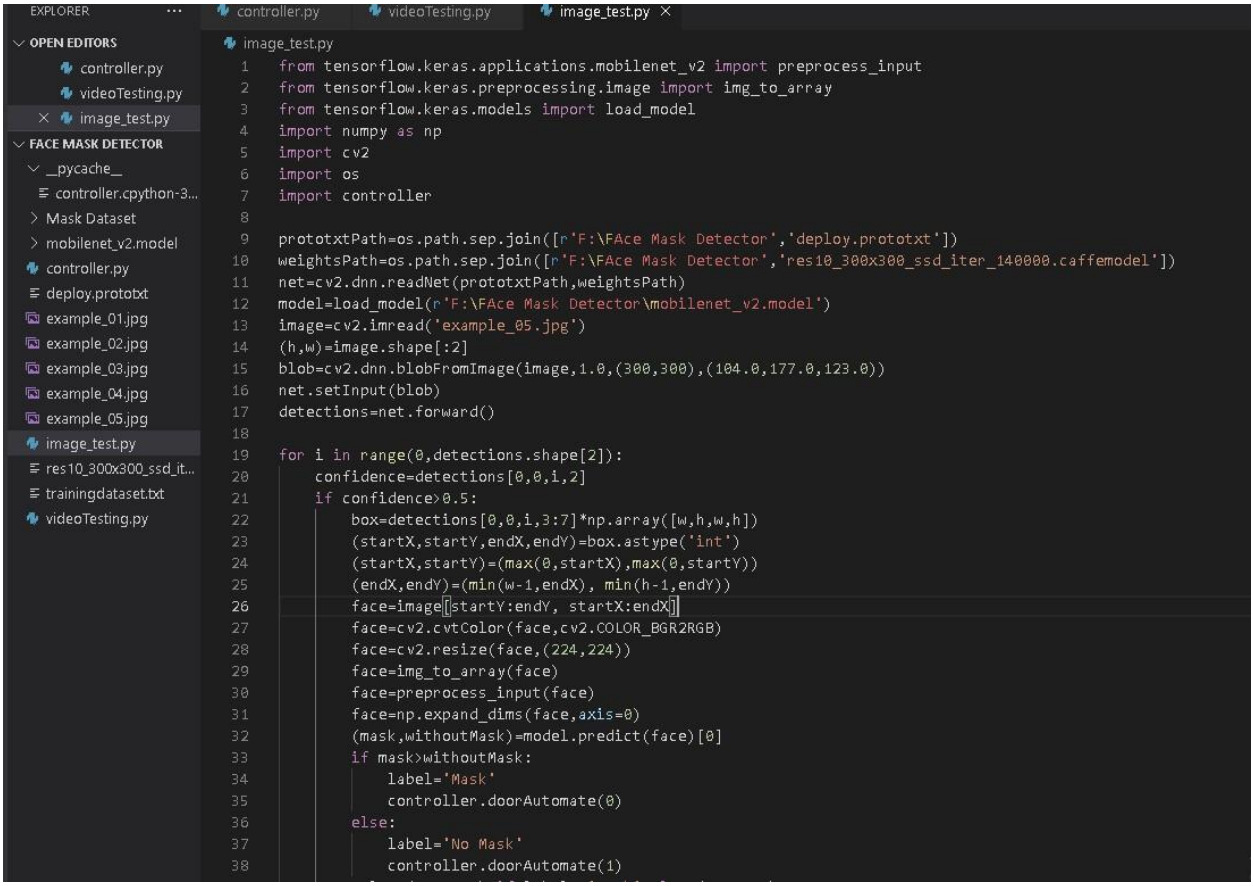
```
In [15]: predict=model.predict(test_X,batch_size=BS)
         predict=np.argmax(predict,axis=1)
         print(classification_report(test_Y.argmax(axis=1),predict,target_names=lb.classes_))
```

```
              precision    recall  f1-score   support

   with_mask       0.97      0.99      0.98       138
without_mask       0.99      0.97      0.98       138

    accuracy                           0.98       276
   macro avg       0.98      0.98      0.98       276
weighted avg       0.98      0.98      0.98       276
```

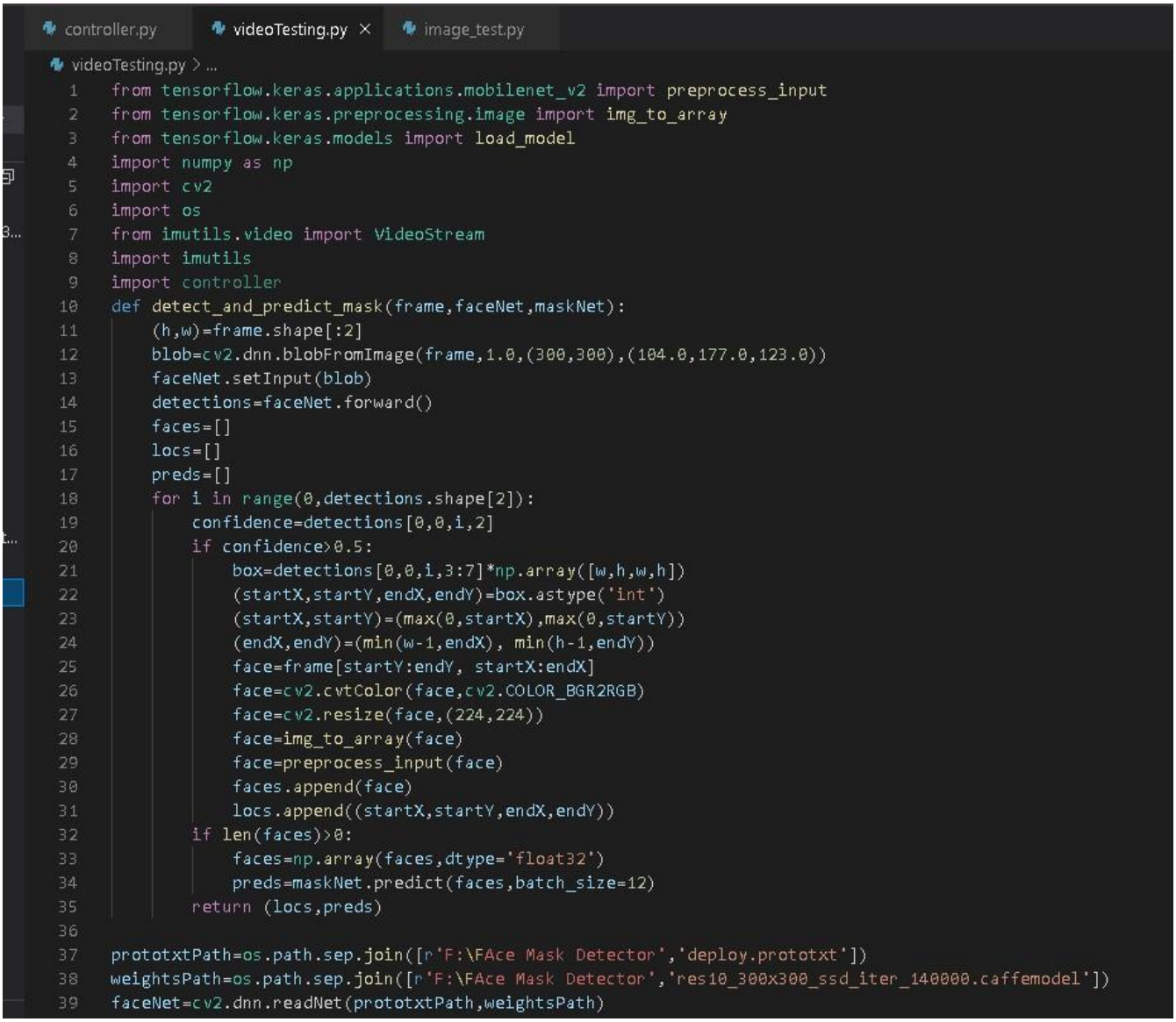
F. Source Code for Image Testing




```
        if mask>withoutMask:
            label='Mask'
            controller.doorAutomate(0)
        else:
            label='No Mask'
            controller.doorAutomate(1)
        color=(0,255,0) if label=='Mask' else (0,0,255)
        label="{}: {:.2f}%".format(label,max(mask,withoutMask)*100)
        cv2.putText(image,label,(startX,startY-10),cv2.FONT_HERSHEY_SIMPLEX,0.45,color,2)
        cv2.rectangle(image,(startX,startY),(endX,endY),color,2)

cv2.imshow("OutPut",image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

G. Source Code for Video Testing



controller.py | videoTesting.py | image_test.py

videoTesting.py > ...

```python
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
import numpy as np
import cv2
import os
from imutils.video import VideoStream
import imutils
import controller
def detect_and_predict_mask(frame,faceNet,maskNet):
    (h,w)=frame.shape[:2]
    blob=cv2.dnn.blobFromImage(frame,1.0,(300,300),(104.0,177.0,123.0))
    faceNet.setInput(blob)
    detections=faceNet.forward()
    faces=[]
    locs=[]
    preds=[]
    for i in range(0,detections.shape[2]):
        confidence=detections[0,0,i,2]
        if confidence>0.5:
            box=detections[0,0,i,3:7]*np.array([w,h,w,h])
            (startX,startY,endX,endY)=box.astype('int')
            (startX,startY)=(max(0,startX),max(0,startY))
            (endX,endY)=(min(w-1,endX), min(h-1,endY))
            face=frame[startY:endY, startX:endX]
            face=cv2.cvtColor(face,cv2.COLOR_BGR2RGB)
            face=cv2.resize(face,(224,224))
            face=img_to_array(face)
            face=preprocess_input(face)
            faces.append(face)
            locs.append((startX,startY,endX,endY))
        if len(faces)>0:
            faces=np.array(faces,dtype='float32')
            preds=maskNet.predict(faces,batch_size=12)
        return (locs,preds)

prototxtPath=os.path.sep.join([r'F:\FAce Mask Detector','deploy.prototxt'])
weightsPath=os.path.sep.join([r'F:\FAce Mask Detector','res10_300x300_ssd_iter_140000.caffemodel'])
faceNet=cv2.dnn.readNet(prototxtPath,weightsPath)
```

```
prototxtPath=os.path.sep.join([r'F:\FAce Mask Detector','deploy.prototxt'])
weightsPath=os.path.sep.join([r'F:\FAce Mask Detector','res10_300x300_ssd_iter_140000.caffemodel'])
faceNet=cv2.dnn.readNet(prototxtPath,weightsPath)
maskNet=load_model(r'F:\FAce Mask Detector\mobilenet_v2.model')
vs=VideoStream(src="http://192.168.0.2:4747/video").start()
while True:
    frame=vs.read()
    frame=imutils.resize(frame,width=400)
    (locs,preds)=detect_and_predict_mask(frame,faceNet,maskNet)
    for (box,pred) in zip(locs,preds):
        (startX,startY,endX,endY)=box
        (mask,withoutMask)=pred
        if mask>withoutMask:
            label='Mask'
            #controller.doorAutomate(0)
        else:
            label='No Mask'
            #controller.doorAutomate(1)
        color=(0,255,0) if label=='Mask' else (0,0,255)
        cv2.putText(frame,label,(startX,startY-10),cv2.FONT_HERSHEY_SIMPLEX,0.45,color,2)
        cv2.rectangle(frame,(startX,startY),(endX,endY),color,2)
    cv2.imshow("Mask Detector",frame)
    key=cv2.waitKey(1) & 0xFF
    if key==ord('q'):
        break

cv2.destroyAllWindows()
vs.stop()
```

H. Source Code for Control Arduino

controller.py > doorAutomate

```
1   from pyfirmata import Arduino, SERVO
2
3   port='COM3'
4
5   pin=10
6
7   board=Arduino(port)
8
9   board.digital[pin].mode=SERVO
10
11  def rotateServo(pin, angle):
12      board.digital[pin].write(angle)
13
14  def doorAutomate(val):
15      if val==0:
16          rotateServo(pin,180)
17      if val==1:
18          rotateServo(pin,40)
19
```